Réalisabilité et paramétricité dans les systèmes de types purs

Marc Lasson

École Normale Supérieure de Lyon

Laboratoire de l'Informatique du Parallélisme

École doctorale Informatique & Mathématiques

THÈSE

en vue de l'obtention du grade de

Docteur de l'École Normale Supérieure de Lyon – Université de Lyon

Spécialité : Informatique

Réalisabilité et Paramétricité dans les Systèmes de Types Purs

---

Présentée et soutenue publiquement le 20 novembre 2012 par :
Marc Lasson

Directeur de thèse :

Patrick Baillot

Co-encadrant :

Olivier Laurent

Après l'avis de :

Thierry Coquand
Martin Hofmann

Devant la commission d'examen formée de :

Patrick Baillot,	Directeur
Nick Benton,	Membre
Martin Hofmann,	Rapporteur
Olivier Laurent,	Co-encadrant
Christine Paulin-Mohring	Présidente

Table des matières

Remerciements

Je tiens à remercier en tout premier lieu mes deux directeurs de thèse Patrick Baillot et Olivier Laurent pour leur constant soutien ainsi que pour l'immense liberté scientifique qu'ils m'ont offerte. Merci également pour les nombreuses corrections et leurs conseils avisés ; leur efficacité redoutable et leur disponibilité constante m'ont été précieuses en particulier dans les moments difficiles.

Un très grand merci à Thierry Coquand et Martin Hoffman d'avoir accepté de relire mon travail. Je remercie également chaleureusement Nick Benton et Christine Paulin d'avoir consenti à faire partie de ce jury.

Je remercie mes coauteurs, Jean-Philippe Bernardy et Chantal Keller, avec lesquels j'ai eu énormément de plaisir à collaborer ; leur influence a été à bien des égards décisive.

Merci à tous les membres de l'équipe Plume, ce manège enchanté était un environnement à la fois stimulant et convivial au sein duquel ces trois années de thèse sont passées trop vite. Merci beaucoup à Catherine Desplanches, mais également à tous les assistants et assistantes qui font vivre le laboratoire.

J'aimerais remercier Alexandre Guitton, Éric Wegrzynowski et Daniel Hirschkoff qui m'ont fait confiance et encouragé à des moments clefs de mon parcours universitaire.

Merci aussi à tous les copains et copines qui m'ont accompagné pendant cette thèse. Merci beaucoup à Aurélien pour son amitié précieuse ; mais également pour son humour légèrement supérieur à celui de Kazoo. Merci à Louis pour son inconditionnelle disponibilité et son enthousiasme. Merci à Romain pour son bureau et sa cuisine si pratique à consommer. Merci à Claire Metz pour sa gentillesse et son hospitalité. Merci à Élise d'être cool. Merci à Lionel pour m'avoir bien souvent écouté parler tout seul. Merci à Guillaume de m'avoir appris le hockey. Merci à Barbara pour sa bonne humeur permanente et son bichromisme. Merci à Paul pour toutes les invitations. Merci à Damien qui sait si bien se déguiser. Merci aux batraciens pour la musique, et en particulier merci encore au trompettiste à pilosité variable qui m'aura soutenu jusqu'au bout.

Un grand merci à la bibliothèque Denis Diderot d'être ouverte aussi tard, ainsi qu'à toutes les petites souris géniales qui y vivent et qui m'ont soutenu pendant ma rédaction : Magali et ses cafés de l'amitié, Charlotte et ses éclats de rire contagieux, mais aussi Claire, Alison, Peg, Noemi, Noémie et Élise.

Merci à mes parents et à mon cher petit frère.

Enfin, un très grand merci à Corinne d'avoir toujours été à mes côtés, pour tous ses encouragements, ses relectures, son soutien logistique mais surtout pour tout son amour.

Cambridge, le 30 octobre 2012.

Introduction

David Hilbert proposa au début des années 1920 un système formel élémentaire capable de représenter les démonstrations. Son projet, que l'on a ensuite appelé *Programme de Hilbert*, avait pour but de justifier de façon définitive les fondements des mathématiques. Un des points essentiels de ce programme était de montrer qu'un tel système minimal devait être en mesure de prouver sa propre cohérence et être suffisamment expressif pour démontrer la cohérence des théories moins élémentaires, en outre, l'analyse réelle.

Le fameux théorème d'incomplétude, publié en 1931 par Kurt Gödel [Gö31, vH02], marque l'anéantissement du programme de Hilbert, ainsi que le début de la théorie de la démonstration moderne. En effet, il affirme qu'il est impossible qu'une théorie cohérente et "suffisamment expressive" puisse démontrer sa propre cohérence. Afin de donner un sens formel à la condition d'expressivité, il montre comment l'arithmétique peut être utilisée pour réifier les démonstrations valides, puis énoncer la cohérence d'une théorie et donc sa propre cohérence. Il introduit alors une notion de calcul élémentaire, exprimable en arithmétique, aujourd'hui connue sous le nom de *récursivité primitive*.

Cette notion de calcul lui permet alors de formaliser le concept de démonstration valide en "programmant la procédure de vérification de preuve". Nous pouvons constater que, très tôt, les fondateurs du domaine ont compris qu'il n'était pas possible de décrire les preuves en tant qu'objets formels, sans comprendre ce qu'est le calcul. Ainsi, ils ont très profondément ancré les fondements des mathématiques dans leur rapport à l'informatique théorique. Ce n'est pas un hasard si le λ-calcul de Church (1932, [Chu36]) et les machines de Turing (1936, [Tur36]) ont été développés peu de temps après les travaux de Gödel. Ces deux formalismes ont permis de donner une signification précise à ce que l'on considère comme une fonction "calculable". Distincts du cadre des fonctions primitives récursives, ces deux modèles de calcul permettent de prendre en compte des procédures qui ne "terminent pas" (on dit dans ce cas qu'elles implémentent des fonctions *partielles*, tandis que les fonctions primitives récursives sont *totales*).

Une des ambitions du programme de Hilbert était de donner une réponse finale au problème des fondements des mathématiques. Or, le théorème d'incomplétude a montré qu'il était voué à l'échec dans la mesure où il affirme l'existence de "vérités mathématiques" qui échappent à tous les systèmes. Le théorème de Gödel indique une impasse, en réfutant le projet réductionniste de Hilbert, mais il a aussi conduit les logiciens à explorer et à comparer de nombreux systèmes de démonstrations, dans le but d'étudier leurs propriétés relatives. Une des méthodes souvent employée pour comparer des systèmes consiste à s'intéresser aux calculs qu'ils sont capables de décrire, aux fonctions dont ils prouvent la terminaison. Ainsi, il est possible d'associer à tout système de démonstrations, suffisamment expressif pour formaliser les notions de calculabilité, la classe des fonctions prouvablement totales de ce système.

En pratique, la classe des fonctions prouvablement totales dans un système est relativement large, car les preuves de terminaison des algorithmes usuels nécessitent rarement des arguments logiques avancés. Néanmoins, une des conséquences du théorème de Gödel est que, pour une théorie cohérente donnée, il existe un programme qui termine toujours et dont la théorie ne peut prouver la terminaison [1].

La théorie de la réalisabilité est un des outils méta-théoriques centraux de la théorie de la démonstration. Elle a été introduite par Stephen Cole Kleene en 1945 [Kle45], afin de construire une sémantique pour l'arithmétique intuitionniste. La réalisabilité consiste à associer des programmes aux formules, que l'on nomme réalisateurs. Par exemple, les réalisateurs d'une implication correspondent aux programmes qui acceptent en entrée les réalisateurs de la prémisse et construisent un réalisateur de la conclusion. Une fois les définitions posées, on montre que la réalisabilité généralise la prouvabilité en montrant que l'on peut calculer, à partir de n'importe quelle démonstration d'une formule, un réalisateur de cette formule. En donnant un contenu calculatoire aux démonstrations, la réalisabilité offre de nombreuses applications méta-théoriques. En outre, elle permet de montrer une propriété essentielle des systèmes intuitionnistes : la propriété de l'existence. Celle-ci affirme que si une formule de la forme $\forall x \exists y P(x, y)$ est prouvable, alors il existe une fonction f calculable, qui accepte en entrée un entier n et retourne un entier $f(n)$ tel que $P(n, f(n))$ est vérifié. On comprend donc qu'une telle propriété constitue un outil de choix pour étudier la classe des fonctions prouvablement totales d'un système. La théorie de la réalisabilité a par la suite été étendue à de nombreux systèmes afin de donner un sens calculatoire aux diverses constructions logiques. Une des extensions qui nous intéressera tout particulièrement, dans ce travail de thèse, est la réalisabilité dite "modifiée" conçue par Georg Kreisel (1959, [Kre59]). Il développe une théorie de la réalisabilité dans laquelle les programmes sont représentés par des termes d'un langage typé, aujourd'hui appelé "système $\mathcal{T}$ de Gödel". Il établit un *théorème de représentation* qui affirme que les fonctions prouvablement totales dans l'arithmétique du premier ordre sont exactement les fonctions que l'on peut programmer dans système $\mathcal{T}$. Ce résultat a ensuite été appliqué à l'arithmétique du second ordre par Jean-Yves Girard [Gir71]. Pour cette tâche, il a introduit un nouveau langage de programmation : système $\mathcal{F}$ et il a montré que ce système capturait exactement les fonctions prouvablement totales dans l'arithmétique du second-ordre. C'est donc pour étudier la logique du second-ordre que système $\mathcal{F}$ a été introduit par Girard en 1971 ; mais c'est en grande partie grâce à son exploitation par les informaticiens que ce système a été popularisé, suite à sa redécouverte par John Charles Reynolds [Rey74]. Depuis, de nombreux langages de programmation fonctionnels typés ont pour base théorique un fragment de système $\mathcal{F}$ où l'inférence de types est décidable ; à savoir au sein duquel l'utilisateur n'a pas besoin de fournir explicitement les types des fonctions qu'il programme : ils sont inférés par le système [Mil78].

Suite à ses travaux sur système $\mathcal{F}$, Reynolds a introduit en 1983 la notion de *paramétricité* afin d'étudier la quantification de type dans système $\mathcal{F}$ [Rey83]. Son but était de donner un sens formel au fait que les fonctions, dont le type est quantifié universellement, se comportent uniformément. Par exemple, il n'est pas possible qu'un habitant du type $\forall\alpha.\alpha \to \alpha \to \alpha$ retourne son premier argument, lorsqu'il est appliqué à un type, et son second argument, lorsqu'il est appliqué à un autre type. C'est cette notion que l'on appelle paramétricité ; elle est formalisée par le fait que les programmes satisfont une relation définie inductivement sur la structure des types. Nous verrons dans ce travail que ces

1. Le programme qui accepte en entrée un entier n et qui diverge si et seulement si n code une démonstration de l'absurdité logique dans la théorie concernée ; un tel programme termine pour toute entrée si et seulement si la théorie est cohérente.

relations, appelées *relations logiques*, sont très étroitement liées à la théorie de la réalisabilité.

La paramétricité a par la suite été largement popularisée par Philip Wadler [Wad89]. Il a montré qu'elle pouvait être très utile aux programmeurs pour déduire les propriétés d'uniformité de programmes concrets. La théorie de la paramétricité a été introduite comme une construction sémantique, et, par la suite, les pré-requis nécessaires pour la définir, ont été formalisés afin d'obtenir des logiques spécifiques, suffisamment expressives pour y définir les relations logiques. Notre travail s'inscrit dans cette démarche qui consiste à formaliser une "logique des programmes". Cette dernière permet d'énoncer et de prouver des propriétés sur les programmes et peut en particulier être utilisée pour les spécifier et pour prouver leur correction. En cela, notre travail est fortement inspiré du système d'aide à la preuve Coq et ne se bornera pas uniquement aux programmes du système $\mathcal{F}$.

L'assistant de preuve Coq est un programme informatique qui aide son utilisateur, de façon interactive, à énoncer des propriétés et à construire des preuves valides [Coq04]. On peut diviser son champ d'application en deux axes principaux : la certification des programmes et la formalisation des mathématiques. Les exemples d'applications dans ces domaines sont très nombreux ; ceux que nous donnons à présent sont emblématiques :

- Le *projet compcert* [2], initié en 2005 par Xavier Leroy, a pour but de développer un compilateur d'un large sous-ensemble du langage C vers des architectures réalistes (PowerPC, ARM, x86, ...). Le développement de compcert est un programme Coq d'une centaine de milliers de lignes dont est *extrait* un compilateur programmé en Caml qui compile du code C vers une de ces architectures. La formalisation en Coq permet de prouver que ce compilateur préserve la sémantique du langage de départ. Cela assure donc à l'utilisateur du compilateur que ce dernier n'introduira aucun bogue en traduisant le C en assembleur.
- Le *théorème de l'ordre impair* est un résultat important de la théorie des groupes [3] ; il a été démontré en 1963 par Walter Feit et John Griggs Thompson [FT63]. Contrairement à la plupart des théorèmes de théorie des groupes, le théorème de Feit-Thompson est réputé pour l'impressionnante longueur de sa démonstration [4]. En septembre 2012, c'est-à-dire six ans après le début du projet, George Gonthier et son équipe sont parvenus à formaliser une preuve de ce théorème dans son intégralité [5].

Les fondements logiques de Coq sont basés sur l'*isomorphisme de Curry-Howard*, aussi appelé la correspondance preuve-programme. Elle met en place une correspondance formelle entre les systèmes de démonstrations et les systèmes de typage pour les programmes. La paternité de cette idée est souvent attribuée à Haskell Curry et à William Alvin Howard car ils l'ont explicitée dans le cadre de la logique combinatoire [6] pour Curry [Cur34] et, plus tard, du λ-calcul pour Howard [How69]. Dans ce contexte, les preuves sont représentées par des termes dont le type est la proposition qu'ils prouvent. Originellement présentée pour la logique propositionnelle, la correspondance a depuis été étendue à des logiques de plus en plus expressives. Ainsi, le calcul typé sous-jacent à Coq s'appelle le *calcul des constructions inductives* [CP88, PM93].

Aujourd'hui, la correspondance preuve-programme constitue un programme de recherche dont le but est de mettre en relation les différentes constructions de la programmation avec les constructions

2. http://compcert.inria.fr/
3. Il affirme que tout groupe fini d'ordre impair est résoluble.
4. L'article originel fait plus de 250 pages !
5. http://www.msr-inria.inria.fr/events-news/feit-thompson-proved-in-coq
6. Un autre modèle de calcul équivalant aux machines de Turing et au λ-calcul.

logiques. Ainsi, tout un courant de recherche [7] s'attache à donner des interprétations calculatoires des preuves, afin de chercher les "programmes derrière les preuves".

Dans ce travail, notre approche sera légèrement décalée : au lieu de fixer un système de démonstrations et de chercher à l'interpréter de façon calculatoire, nous aborderons la question réciproque suivante : dans quelle mesure un langage de programmation donné peut-il induire une logique qui peut être interprétée de façon calculatoire par ce même langage de programmation ? Nous allons proposer une construction qui, étant donnée la description d'un langage de programmation, nous fournira un système logique, qui permettra d'énoncer des propriétés à propos des programmes ainsi que les preuves de ces propriétés. Nous verrons ensuite que les constructions logiques de ce système sont réalisées par le langage de programmation de départ et que cette logique offre un cadre adéquat pour développer une théorie de la paramétricité.

Pour parvenir à notre objectif, nous avons dû trouver un cadre formel dans lequel il était possible de représenter de façon uniforme des langages de programmation, ainsi que des systèmes de démonstrations : les *systèmes de types purs* [Bar92]. Ces derniers fournissent un cadre syntaxique au sein duquel il est possible de décrire une grande famille de calculs typés dont certains peuvent être vus, à travers l'isomorphisme de Curry-Howard, comme des systèmes formels de démonstrations. Parmi les systèmes présentables sous forme de systèmes de types purs, on trouve par exemple système $\mathcal{F}$, mais également le calcul des constructions de Gérard Huet et Thierry Coquand [CH88] (dont le calcul des constructions inductives, que est la base de Coq, est une extension).

L'un des leurs attraits principaux est de pouvoir factoriser les théorèmes méta-théoriques de tous les systèmes qu'ils décrivent. Ils permettent parfois de mieux comprendre les problèmes : en effet, en se plaçant dans un cadre très abstrait, on a alors la possibilité de s'éloigner des spécificités inhérentes aux systèmes particuliers. Mais cette sensation peut être illusoire car, plutôt que de repousser les cas particuliers, il est parfois nécessaire de les considérer tous.

Une particularité essentielle des systèmes de types purs réside dans le fait que leurs expressions permettent de décrire à la fois les termes représentant les calculs et les types : il n'y a donc pas deux catégories syntaxiques distinctes qui scinderaient les programmes et les types. Certains systèmes de types purs sont ainsi capables de représenter des calculs sur les types. Les expressions de ces systèmes étendent donc les termes du λ-calcul avec une construction qui permet d'exprimer les types : les *produits dépendants*. Ils permettent, étant données deux expressions A et B, de construire le type que l'on note [8] ici $\forall x : A.B$. Ils représentent le type des fonctions qui prennent en argument un objet x de type A et retournent un objet de type B. Le point important qui les distingue des types simples (notés [9] $A \to B$) est que le type de retour B peut dépendre de la valeur de x. Les types dépendants s'avèrent être très utiles pour la formalisation des mathématiques. On peut considérer à titre d'exemple l'opérateur $\mathbf{M}_{n,m}(\mathcal{R})$ qui désigne les matrices de taille n fois m sur un anneau $\mathcal{R}$. On pourrait alors lui donner le type :

$$\mathbf{M} : \mathsf{nat} \to \mathsf{nat} \to \mathsf{Ring} \to \mathsf{Ring}$$

7. On peut par exemple citer les travaux récents de Jean-Louis Krivine dans lesquels il étend la réalisabilité afin de construire des modèles de l'arithmétique classique avec axiome du choix dépendant [Kri09] ou de la théorie des ensembles [Kri12].

8. On trouve également souvent la notation $\Pi x : A.B$.

9. Nous utiliserons la notation $A \to B$ pour désigner le produit dépendant $\forall x : A.B$ lorsque la variable x n'apparaît pas dans B.

On conçoit donc cet opérateur **M** comme une fonction qui, étant donnés deux entiers et un anneau, retourne un anneau. Le type de la multiplication de deux matrices peut alors être donné par le type dépendant suivant :

$$\forall n\ k\ m : \mathsf{nat}.\mathbf{M}_{n,k}(\mathcal{R}) \to \mathbf{M}_{k,m}(\mathcal{R}) \to \mathbf{M}_{n,m}(\mathcal{R})$$

De plus, nous utiliserons le constructeur de produit dépendant pour représenter les quantificateurs universels dans les propositions logiques : si P est une proposition logique qui dépend d'un individu x de type A, alors $\forall x : A.P$ désignera la proposition "pour tout x de type A, P". On observe ici l'isomorphisme de Curry-Howard à l'œuvre puisqu'une preuve de cette proposition sera alors un terme qui, lorsque qu'on lui fournit un objet de type A, retourne une preuve de P.

Chaque système de types purs est défini par des paramètres ; ceux-ci spécifient les constructions autorisées dans le calcul. Ils sont constitués de trois ensembles : le premier décrit un ensemble de constantes, appelées *sortes*, le second décrit le type des sortes et enfin le troisième décrit les produits dépendants autorisés. C'est donc en fixant ces paramètres que l'on caractérise le système que l'on veut étudier. Tout l'intérêt du cadre des systèmes de types purs est de pouvoir étudier les propriétés des systèmes en se ramenant aux propriétés des paramètres. Hélas, tous les calculs typés ne peuvent pas être décrits comment des systèmes de types purs. C'est pourquoi nous présenterons deux extensions naturelles des systèmes de types purs : la *cumulativité* et les *types inductifs*. Elles permettront d'intégrer dans notre cadre à la fois système $\mathcal{T}$ et le calcul des constructions inductives.

Ainsi, nous montrerons qu'il est possible de développer une théorie de la paramétricité pour les termes du calcul des constructions inductives. Ce qui nous permettra de déployer, au sein de Coq, de nouveaux outils pour prouver la correction des programmes polymorphes (on pourra en particulier formaliser tous les exemples de Philip Wadler de [Wad89]). Nous verrons également un exemple original d'application de la paramétricité à la formalisation des mathématiques concernant les structures algébriques discrètes (nous prendrons l'exemple des groupes finis).

Dans le chapitre 1, nous considérons deux extensions des systèmes de types purs (PTS) qui nous permettront d'englober davantage de systèmes : les systèmes de types purs cumulatifs (CTS) et leurs extensions aux types inductifs (CTSI). Les CTS permettent de capturer les systèmes qui utilisent une relation de sous-typage appelée la *cumulativité*, introduite par Zhaohui Luo dans le *calcul des constructions étendu* [Luo89]. Les CTS accolent donc un nouveau paramètre aux PTS, permettant de décrire les relations de sous-typage entre les sortes. La deuxième section du chapitre 1 se compose d'une étude complète des propriétés des systèmes en fonction de la forme de leurs paramètres. Nous verrons que les CTS permettent de capturer beaucoup de systèmes et seule une faible partie d'entre eux constitue des systèmes raisonnables. Enfin, les deux dernières sections présenteront deux nouvelles notions : les systèmes prédicatifs et les systèmes faiblement imprédicatifs. Ces deux notions se définissent par un critère syntaxique sur les paramètres des CTS mais nous verrons qu'ils correspondront chacun au fait qu'il existe un plongement du système vers un système de référence : la théorie des types de Martin-Löf, dans le cas prédicatif et le calcul de construction avec univers, dans le cas faiblement imprédicatif. À partir de ces critères syntaxiques sur la forme des paramètres, nous en déduisons un test qui permet en pratique de prouver la forte normalisation (et donc la cohérence) de nombreux systèmes. Si ce premier chapitre ne contient pas de résultats fondamentalement nouveaux, il nous a paru nécessaire de voir comment les propriétés des systèmes particuliers pouvaient s'exprimer dans le cadre général de notre travail. Les outils ainsi développés permettent de défricher l'ensemble des systèmes et de factoriser les

résultats méta-théoriques qui serviront à étudier les différents systèmes introduits par la suite.

Le chapitre 2 est consacré à l'extension aux types inductifs. Il étend les paramètres des CTS par le biais de deux nouveaux paramètres : le premier, afin de décrire les inductifs autorisés et le second, pour décrire les éliminations autorisées (à savoir les contextes dans lesquels il est autorisé de faire une distinction de cas sur un inductif). Les définitions introduites sont volontairement très proches de la présentation des inductifs du système Coq : cela nous permettra de proposer une présentation du calcul des constructions inductives fidèle à son implémentation dans Coq.

Le chapitre 3 décrit la transformation qui à un CTS $\mathcal{P}$ associe un CTS $\mathcal{P}^2$, qui permet d'engendrer la logique sur les programmes de $\mathcal{P}$. Dans ce chapitre, la construction est essentiellement illustrée par le système $\mathcal{F}^2$, généré par système $\mathcal{F}$. Nous introduisons ensuite la notion de *système de notations pour l'arithmétique* qui formalise des conditions suffisantes pour représenter les preuves d'arithmétique dans nos systèmes. Cet expédiant s'avérera essentiel pour énoncer les théorèmes de représentation du chapitre 4. Nous montrons ensuite comment ajouter aux termes du système des annotations qui simplifieront les définitions des différentes transformations que nous introduisons sur les termes de $\mathcal{P}^2$. Enfin, le chapitre conclut sur la description d'une large famille de CTS, les *systèmes stratifiés engendrés* ; ces derniers nous intéresseront car la transformation $\cdot \mapsto \cdot^2$, appliquée à ces systèmes, se comporte bien vis-à-vis des propriétés liées à la cumulativité.

Dans le chapitre 4, nous développons la théorie de la réalisabilité dans $\mathcal{P}^2$. Nous introduisons la définition de la relation de réalisabilité, puis nous montrons le *théorème d'adéquation* qui affirme que de toute preuve π d'une proposition P de $\mathcal{P}^2$, on peut extraire un programme, noté $\lfloor \pi \rfloor$, et construire une preuve, notée $\|\pi\|$ que $\lfloor \pi \rfloor$ réalise P, ce que l'on notera $\lfloor \pi \rfloor \Vdash P$. Dans la seconde section, nous montrons comment appliquer ce résultat pour prouver les théorèmes de représentation dans les systèmes qui encodent leurs données grâce aux encodages de système $\mathcal{F}$. Enfin, nous concluons sur l'extension de la construction du chapitre 3 aux types inductifs, afin d'en déduire un théorème de représentation pour les systèmes qui représentent leurs données par des types inductifs. Ceci nous permettra d'étendre nos résultats aux extensions du système $\mathcal{T}$.

Enfin, dans le chapitre 5, nous constaterons un lien syntaxique entre la réalisabilité et les relations logiques. Dans ce chapitre, nous montrons que ces relations peuvent être définies en termes de réalisabilité. Ainsi, il est possible d'utiliser le théorème d'adéquation de la réalisabilité afin de démontrer que tout terme clos satisfait la relation engendrée par son type. L'approche, qui consiste à formaliser la méta-théorie dans laquelle il est possible de formaliser la paramétricité, a été suivie par de nombreux auteurs pour système $\mathcal{F}$. Puis, elle a été étendue par Jean-Philippe Bernardy, Patrik Jansson, et Ross Paterson [BJP10] à une classe de systèmes de types purs. La différence entre la présentation faite ici et celle de Bernardy *et al.* réside dans le fait que ces derniers considèrent une classe de systèmes stables par la construction $\cdot \mapsto \cdot^2$. Ainsi, le mérite principal de notre approche est de pouvoir être appliquée à de nombreux systèmes, même s'ils ne sont pas suffisament expressifs pour satisfaire la condition de stabilité. La seule condition étant qu'ils soient décrits comme des systèmes de types purs.

Une fois les définitions explicitées, nous discuterons des diverses applications de la construction de paramétricité. En particulier, nous nous attacherons à montrer comment cette approche peut être déployée dans le système d'aide à la preuve Coq. Nous donnerons ensuite quelques exemples d'application. Enfin, nous concluons sur l'ébauche d'une implémentation dans le système Coq d'une *tactique* de démonstration automatique basée sur la paramétricité.

Chapitre 1

Étude des systèmes de types cumulatifs

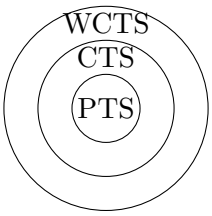
1.1 Les systèmes de types cumulatifs

Les systèmes de types purs ont été introduits indépendamment par Berardi [Ber89] et Terlouw [Ter89] pour étudier les systèmes avec types dépendants et ils ont été popularisés dans leur forme actuelle par Geuvers et Nederhof [GN91]. Ils constituent une généralisation englobant une grande famille de variantes du λ-calcul dites "à la Church", c'est-à-dire où les termes à typer sont annotés afin de faciliter la génération de la dérivation de type. Nous verrons dans la suite qu'elles contiennent, par exemple, le λ-calcul simplement typé, le λ-calcul polymorphe (système $\mathcal{F}$) et son extension d'ordre supérieur (système $\mathcal{F}_\omega$), différentes variantes du Calcul des Constructions et certains exemples historiques de systèmes incohérents (tous les types y sont habités et certains termes bien typés sont divergents).

L'intérêt principal des systèmes de types purs est de fournir un cadre général et unifié pour étudier les systèmes de types pouvant contenir des types dépendants.

Ils permettent de modéliser les systèmes de typage des langages de programmation fonctionnels (comme par exemple *Haskell* [HPJW+92], *Objective Caml* [LDF+]) mais également le noyau logique des assistants de preuves dans lesquels les propositions sont représentées par des types (comme par exemple Coq [Coq04], Agda [BDN09]). Dans ce travail, nous utiliserons le cadre des PTS pour représenter à la fois des langages de programmation et des systèmes logiques en déduction naturelle construits au moyen de la correspondance preuves-programmes.

On appelle *système de types cumulatifs faible* (*WCTS* en abrégez pour *weak cumulative type system*) la donnée $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{C})$ de quatre ensembles : l'ensemble $\mathcal{S}$ des *sortes*, la relation $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$ des *axiomes*, l'ensemble $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{S}$ des *règles* et enfin la relation $\mathcal{C} \subseteq \mathcal{S} \times \mathcal{S}$ de *cumulativité*. On appelle *système de types purs* (*PTS* en abrégé pour *pure type system*), tout WCTS tel que $\mathcal{C} = \emptyset$ et on parle de *système de types cumulatifs* (*CTS* en abrégé pour *cumulative type system*) lorsque $\mathcal{C}$ est un ordre strict.



On notera $\mathcal{P} \models$ PTS (resp. $\mathcal{P} \models$ CTS) pour signifier que le WCTS $\mathcal{P}$ est un PTS (resp. un CTS). Et on se permettra d'omettre le $\mathcal{P}$ lorsqu'il n'y aura pas d'ambiguïté sur le WCTS considéré.

Dans le reste du texte, nous ferons en sorte que les définitions et les propriétés standards énoncées pour les CTS et les WCTS se spécialisent dans le cas des PTS dans leur équivalent dans la littérature [Bar92, Ber89].

La définition de CTS a été introduite par Bruno Barras [BG05, Bar04] afin d'étudier les systèmes de types dépendants munis d'une relation de cumulativité. Elle permet en particulier de capturer le noyau fonctionnel de systèmes tels que le Calcul des Constructions Étendu de Luo [Luo89] (qui est également le noyau fonctionnel du calcul sous-jacent au système Coq [Coq04]), ou encore la théorie des types de Martin-Löf intentionnelle qui peut être présentée comme un CTS (le système obtenu est alors très proche de celui implémenté dans Agda [BDN09]).

Enfin, la notion de WCTS est introduite ici afin d'étudier dans un premier temps l'impact des conditions imposées par la relation d'inclusion sur les propriétés des systèmes. Mais en pratique, tous les exemples concrets qui seront considérés par la suite seront présentés comme des CTS.

1.1.1 Syntaxe

Étant donné un WCTS $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{C})$, on définit l'ensemble des expressions de $\mathcal{P}$ comme étant les termes générés par la grammaire suivante :

$$\begin{array}{rcll} A, B, \ldots & := & (A\ B) & \textit{application} \\ & | & \lambda x : A.B & \textit{abstraction} \\ & | & \forall x : A.B & \textit{produit} \\ & | & x & \textit{variable} \\ & | & s & \textit{sorte} \end{array}$$

où x parcourt un ensemble de variables $\mathcal{V}$ et s l'ensemble des sortes $\mathcal{S}$.

Les constructions λ et $\forall$ se comportent comme des lieurs et l'on considère les expressions à renommage des variables liées près (α-équivalence). On suppose également que l'on dispose d'une bonne notion de substitution, c'est-à-dire préservant l'α-équivalence : on notera $A[B/x]$ l'expression obtenue en remplaçant dans A les occurrences libres de la variable x par B. L'ensemble des variables libres d'un terme A sera désigné par $\mathcal{FV}(A)$.

On notera $A \to B$ les produits non-dépendants, c'est-à-dire les produits de la forme $\forall x : A.B$ avec $x \notin \mathcal{FV}(B)$. On adoptera les conventions suivantes pour noter les imbrications de produit, d'abstraction et d'applications :

$$\begin{array}{rcl} M_1\, M_2 \cdots M_n & \text{désignera} & ((\cdots (M_1\, M_2) \cdots) M_n) \\ \lambda x_1 : A_1, \ldots, x_n : A_n.M & \text{désignera} & (\lambda x_1 : A_1. \cdots (\lambda x_n : A_n.M) \cdots) \\ \forall x_1 : A_1, \ldots, x_n : A_n.B & \text{désignera} & (\forall x_1 : A_1. \cdots (\lambda x_n : A_n.B) \cdots) \\ A_1 \to A_2 \to \cdots \to A_n \to B & \text{désignera} & A_1 \to (A_2 \to \cdots (A_n \to B) \cdots) \end{array}$$

On définit également la relation $\twoheadrightarrow$ comme étant la plus petite relation entre les expressions qui satisfait les règles suivantes :

$$\frac{}{(\lambda x : A.B)\, C \twoheadrightarrow B[C/x]}$$

$$\frac{M \twoheadrightarrow M'}{(M\ N) \twoheadrightarrow (M'\ N)} \qquad \frac{N \twoheadrightarrow N'}{(M\ N) \twoheadrightarrow (M\ N')}$$

$$\frac{A \twoheadrightarrow A'}{\lambda x : A.M \twoheadrightarrow \lambda x : A'.M} \qquad \frac{M \twoheadrightarrow M'}{\lambda x : A.M \twoheadrightarrow \lambda x : A.M'}$$

$$\frac{A \twoheadrightarrow A'}{\forall x : A.B \twoheadrightarrow \forall x : A'.B} \qquad \frac{B \twoheadrightarrow B'}{\forall x : A.B \twoheadrightarrow \forall x : A.B'}$$

On notera $\rhd$ la clôture transitive de $\twoheadrightarrow$, $\trianglerighteq$ la clôture réflexive de $\rhd$ et $\equiv$ la clôture symétrique et transitive de $\twoheadrightarrow$. On dira d'un terme A qu'il est en *forme normale* s'il n'existe pas de terme A' tel que $A \rhd A'$. Un corollaire de la confluence de ce calcul (voir 1.1.3) est l'unicité des formes normales. On désignera donc par $\mathrm{NF}(A)$ la forme normale de A quand elle existe.

1.1.1 Définition (Normalisation faible et forte)
Soit $\mathcal{P}$ un WCTS. On dit d'une expression A de $\mathcal{P}$ qu'elle est faiblement normalisante (resp. fortement normalisante si A a une forme normale (resp. s'il n'existe pas de chaîne infinie de réduction pour $\rhd$ qui part de A). On dit de $\mathcal{P}$ qu'il est faiblement normalisant (resp. fortement normalisant) si toutes les expressions bien typées (voir plus avant) sont faiblement normalisantes (resp. fortement normalisante). On notera cette propriété WEAKLY-NORMALISING (resp. NORMALISING).

On mentionne au passage une des conjectures les plus fameuses [1] [Geu93] :

1.1.2 Question ouverte (Conjecture de Barendregt–Geuvers–Klop)
Tous les PTS faiblement normalisants sont fortement normalisants.

Évidemment cette conjecture s'étend aux WCTS : on ne connaît pas de WCTS qui normalisent faiblement mais pas fortement.

Le lemme suivant énonce la confluence de la réduction des termes. Il dépend uniquement de la syntaxe des PTS qui est la même que celle des WCTS. Et la preuve est similaire à celle du λ-calcul pur, voir [Bar84] par exemple.

1.1.3 Lemme (Propriété de Church-Rosser)
Si $A \equiv A'$ alors il existe C tel que $A \trianglerighteq C$ et $A' \trianglerighteq C$.

On en déduit alors facilement le lemme suivant :

1.1.4 Lemme (Injection de la syntaxe)

$$\begin{aligned}
\forall x : A.B \equiv \forall x : A'.B' &\Rightarrow A \equiv A' \wedge B \equiv B' \\
\lambda x : A.B \equiv \lambda x : A'.B' &\Rightarrow A \equiv A' \wedge B \equiv B' \\
s \equiv s' &\Rightarrow s = s' \\
x \equiv x' &\Rightarrow x = x'
\end{aligned}$$

1. c'est le neuvième problème ouvert de TLCA, voir http://tlca.di.unito.it/opltlca/

1.1.2 Cumulativité

La définition suivante introduit la relation de sous-typage $\prec$ qui est paramétrée par $\mathcal{C}$; $A \prec B$ signifie que A est convertible en un terme de la forme $\forall x_1 : C_1, \cdots, x_n : C_n.s$ (on dira alors que A est une *arité* de sorte s) et B est convertible en un terme de la forme $\forall x_1 : C_1, \cdots, x_n : C_n.s'$ (pour les mêmes C_i) avec $(s, s') \in \mathcal{C}$. Plus formellement :

1.1.5 Définition (Relation de cumulativité)
On définit les ω relations $\prec_0, \prec_1, ...$, ainsi que leur union $\prec$ et leur clôture réflexive $\preccurlyeq$ vis-à-vis de la conversion $\equiv$ par les équations suivantes :

$$\begin{aligned} A \prec_0 B &\Leftrightarrow A \equiv s \wedge B \equiv s' \wedge (s, s') \in \mathcal{C} \\ A \prec_{n+1} B &\Leftrightarrow A \prec_n B \vee (A \equiv \forall x : C.A' \wedge B \equiv \forall x : C.B' \wedge A' \prec_n B') \\ A \prec B &\Leftrightarrow \exists n \in \mathbb{N}, A \prec_n B \\ A \preccurlyeq B &\Leftrightarrow A \equiv B \vee A \prec B \end{aligned}$$

Tout comme celle du Calcul des Constructions Étendu (ECC) Zhaohui Luo [Luo89], notre règle de cumulativité n'est pas contra-variante pour la construction de produit. En effet, on ne peux pas déduire de $A' \preccurlyeq A$ et de $B \preccurlyeq B'$ que $\forall x : A.B \preccurlyeq \forall x : A'.B'$. Ce choix de conception est essentiellement imposé par deux conditions :

- Premièrement, nous souhaitons interpréter (en théorie des ensembles) la cumulativité par l'inclusion des interprétations des types "Si on a $A \preccurlyeq B$, on veut que $[\![A]\!] \subseteq [\![B]\!]$". Ce qui n'est pas possible pour une relation de sous-typage qui serait pleinement contra-variante.
- On souhaite que, sous hypothèse de forte normalisation, la relation de cumulativité soit décidable. Or, on sait par exemple que la relation de sous-typage de système $F_{<:}$ est indécidable (et donc le problème de la vérification de type l'est également) [Pie94].

On notera qu'il existe néanmoins une généralisation des PTS avec quantification bornée [Zwa99], mais celle-ci ne permet pas le sous-typage entre sortes et ne permet donc pas de capturer les systèmes de types avec univers qui nous intéressent tout particulièrement dans ce travail.

La relation de cumulativité est stable par substitution :

1.1.6 Lemme

$$A \preccurlyeq B \Rightarrow A[C/x] \preccurlyeq B[C/x]$$

Démonstration On prouve

$$A \prec_n B \Rightarrow A[C/x] \prec_n B[C/x]$$

par induction sur n :

- Si $A \prec_0 B$ cela signifie qu'il existe s_A et s_B tels que $A \equiv s_A$ et $B \equiv s_B$. Et donc $A[C/x] \equiv s_A$ et $B[C/x] \equiv s_B$ et on a bien $A[C/x] \prec_0 B[C/x]$.
- Si $A \prec_{n+1} B$, on distingue deux cas :
  - Soit on a directement $A \prec_n B$, et l'on conclut directement d'après l'hypothèse d'induction.
  - Soit il existe A', B' et D tels que $A \equiv \forall x : D.A'$, $B \equiv \forall x : D.B'$ et $A' \prec_n B'$. Par hypothèse d'induction, on en déduit que $A'[C/x] \prec_n B'[C/x]$. Et donc $\forall x : D[C/x].A'[C/x] \prec_{n+1} \forall x : D[C/x].B'[C/x]$. Or $\forall x : D[C/x].A'[C/x] \equiv A$ et $\forall x : D[C/x].B'[C/x] \equiv B$, donc $A \prec_{n+1} B$.

Supposons que $A \preccurlyeq B$, alors deux cas sont possibles :
- Soit $A \equiv B$, et alors $A[C/x] \equiv B[C/x]$ et donc $A[C/x] \preccurlyeq B[C/x]$.
- Soit $A \prec B$ et dans ce cas il existe un n tel que $A \prec_n B$ et d'après ce que l'on vient juste de démontrer $A[C/x] \prec_n B[C/x]$ et donc $A[C/x] \preccurlyeq B[C/x]$. ☺

1.1.7 Lemme
Les relations $\prec$, $\prec_n$, $\preccurlyeq$ sont compatibles avec $\equiv$:

$$\left.\begin{array}{l} A \prec_n B \\ A \equiv A' \\ B \equiv B' \end{array}\right\} \Rightarrow A' \prec_n B' \qquad \left.\begin{array}{l} A \prec B \\ A \equiv A' \\ B \equiv B' \end{array}\right\} \Rightarrow A' \prec B' \qquad \left.\begin{array}{l} A \preccurlyeq B \\ A \equiv A' \\ B \equiv B' \end{array}\right\} \Rightarrow A' \preccurlyeq B'$$

Démonstration La preuve de la première implication est immédiate par induction sur n et les deux autres en découlent. ☺

1.1.3 Règles de typage

On appelle *contexte* toute liste Γ ordonnée $a_1 : A, \ldots, a_n : A_n$ où les a_i sont des variables et les A_i des expressions, et on désigne par $\langle\rangle$ le contexte vide. On notera $x : A \in \Gamma$ pour signifier que Γ est de la forme $\Gamma', x : A, \Gamma''$ et on notera $x \in \Gamma$ s'il existe un A tel que $x : A \in \Gamma$. On ne confondra pas avec $x \in \mathcal{FV}(\Gamma)$ où $\mathcal{FV}(\Gamma)$ est défini récursivement par :

$$\begin{aligned} \mathcal{FV}(\langle\rangle) &= \emptyset \\ \mathcal{FV}(x : A, \Gamma) &= \mathcal{FV}(A) \cup (\mathcal{FV}(\Gamma) \setminus \{x\}) \end{aligned}$$

On définit ensuite le système de règle de typage paramétré par $\mathcal{P}$, c'est à dire la plus petite relation ternaire $\vdash_{\mathcal{P}}$ (qu'on notera simplement $\vdash$ lorsqu'il n'y aura pas d'ambiguïté sur le WCTS considéré) entre les contextes et les couples d'expressions qui satisfait les huit règles de la figure 1.1. On notera $\Gamma \vdash A : B$ pour dire que (Γ, A, B) sont en relation et on dira que A est une expression bien formée de type B dans le contexte Γ. On dit d'un contexte Γ qu'il est un *contexte bien formé* s'il existe deux expressions A et B telles que $\Gamma \vdash A : B$ (ce que l'on notera $\mathrm{WF}(\Gamma)$, voir définition 1.1.25). On dit de A qu'elle est une *expression bien formée* dans un contexte Γ si c'est une sorte ou s'il existe une autre expression B telle que $\Gamma \vdash A : B$. On définit les types comme étant les expressions qui habitent les sortes ; pour être précis, on dit d'une expression A qu'elle est un *type* de sorte s dans le contexte Γ si on a $\Gamma \vdash A : s$. On notera plus avant (définition 1.1.25) $\mathrm{WF}_\Gamma(A)$ le fait que A soit un type bien-formé dans un contexte Γ.

Les sortes servent donc à donner un type aux types, elles sont donc elles-mêmes des types ; c'est le paramètre $\mathcal{A}$ qui fixe les types des sortes grâce à la règle ci-dessous.

$$\frac{}{\vdash s : s'}\ (s, s') \in \mathcal{A}$$
AXIOME

On dispose également de deux règles structurelles, l'une pour donner un type aux variables libres, l'autre pour rajouter des variables dans le contexte. On voit que les règles assurent que le type de la variable est bien formé.

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\, x \notin \Gamma \qquad \text{Variable}$$

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}\, x \notin \Gamma \qquad \text{Affaiblissement}$$

Nous disposons également de deux règles pour typer l'abstraction et l'application ; il est important de noter que la règle d'abstraction vérifie à la fois que le corps de l'abstraction est bien formé, et que son type est bien formé.

$$\frac{\Gamma, x : A \vdash B : C \qquad \Gamma \vdash \forall x : A.C : s}{\Gamma \vdash \lambda x : A.B \;:\; \forall x : A.C} \qquad \text{Abstraction}$$

$$\frac{\Gamma \vdash M : \forall x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : B[N/x]} \qquad \text{Application}$$

C'est la règle de produit ci-dessous qui indique si un produit est autorisé, c'est-à-dire bien formé. Elle est paramétrée par l'ensemble de règles $\mathcal{R}$ qui décrit la sorte du produit en fonction des sortes des types du domaine et du codomaine du produit.

$$\frac{\Gamma \vdash A : s_A \qquad \Gamma, x : A \vdash B : s_B}{\Gamma \vdash \forall x : A.B : s}\, (s_A, s_B, s) \in \mathcal{R} \qquad \text{Produit}$$

Enfin, les deux dernières règles du système sont les règles de cumulativité, elles permettent d'introduire le sous-typage paramétré par l'ensemble $\mathcal{C}$.

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s \qquad B \preccurlyeq B'}{\Gamma \vdash A : B'} \qquad \text{Cumulativité}$$

$$\frac{\Gamma \vdash A : C \qquad C \preccurlyeq s}{\Gamma \vdash A : s} \qquad \text{Cumulativité-Sortes}$$

La règle Cumulativité permet de remplacer un type par un sur-type à condition que ce dernier soit bien formé. Les types bien formés sont divisés en deux classes disjointes (voir 1.1.25) : les arités qui admettent pour type une sorte et les sortes maximales pour la relation $\mathcal{A}$. On les appellera tout simplement les *sortes maximales* lorsqu'il n'y aura pas d'ambiguïté.

La règle de conversion permet de remplacer le type d'une expression par un type équivalent pour la relation $\equiv$ à condition que ce dernier soit bien formé.

1.1.8 Lemme
La règle suivante est admissible :

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'}\, B \equiv B' \qquad \text{Conversion}$$

Dans le cas des PTS, Cumulativité se réduit à la règle de conversion car :

$$\mathcal{C} = \emptyset \qquad \Rightarrow \qquad A \preccurlyeq B \Leftrightarrow A \equiv B$$

De plus, la règle Cumulativité-Sortes devient superflue car $s_1 \preccurlyeq s_2 \Leftrightarrow s_1 = s_2$. On retrouve donc bien la présentation usuelle des PTS de la littérature.

$$\frac{}{\vdash s : s'}\ (s, s') \in \mathcal{A}$$
Axiome

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\ x \notin \Gamma \qquad \frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}\ x \notin \Gamma$$
Variable — Affaiblissement

$$\frac{\Gamma, x : A \vdash B : C \qquad \Gamma \vdash \forall x : A.C : s}{\Gamma \vdash \lambda x : A.B \quad : \quad \forall x : A.C} \qquad \frac{\Gamma \vdash M : \forall x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B[N/x]}$$
Abstraction — Application

$$\frac{\Gamma \vdash A : s_A \qquad \Gamma, x : A \vdash B : s_B}{\Gamma \vdash \forall x : A.B : s}\ (s_A, s_B, s) \in \mathcal{R}$$
Produit

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s \qquad B \preccurlyeq B'}{\Gamma \vdash A : B'} \qquad \frac{\Gamma \vdash A : C \qquad C \preccurlyeq s}{\Gamma \vdash A : s}$$
Cumulativité — Cumulativité-Sortes

Figure 1.1 – Règles de typage des WCTS .

1.1.4 Le λ-cube de Barendregt

Parmi, les PTS il existe une sous-classe particulièrement naturelle :

1.1.9 Définition (PTS fonctionnel)
Un PTS est fonctionnel si $\mathcal{A}$ et $\mathcal{R}$ sont des relations fonctionnelles, c'est-à-dire si

1. $(s_1, s_2) \in \mathcal{A} \quad \wedge \quad (s_1, s_2') \in \mathcal{A} \qquad \Rightarrow \qquad s_2 = s_2'$
2. $(s_1, s_2, s_3) \in \mathcal{R} \quad \wedge \quad (s_1, s_2, s_3') \in \mathcal{R} \qquad \Rightarrow \qquad s_3 = s_3'$

On notera $\mathcal{P} \models$ FUNCTIONAL pour signifier qu'un WCTS $\mathcal{P}$ est un PTS fonctionnel (ou simplement $\models$ FUNCTIONAL s'il n'y a pas d'ambiguïté sur le WCTS concerné).

Ces systèmes ont la propriété d'unicité du typage :

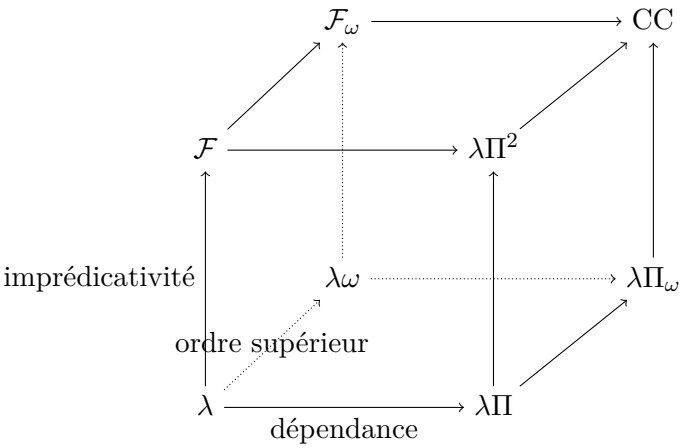
1.1.10 Lemme (Unicité du typage dans les PTS fonctionnels)
Si $\models$ FUNCTIONAL, *alors on a :*

$$\left.\begin{array}{l}\Gamma \vdash M : A_1\\ \Gamma \vdash M : A_2\end{array}\right\} \qquad \Rightarrow \qquad A_1 \equiv A_2$$

Démonstration On trouve par exemple une preuve dans [GN91]. C'est également une conséquence du lemme 1.2.31 comme nous le verrons plus avant. ☺

1.1.11 Définition (WCTS homogène)
On dira d'un WCTS $\mathcal{P}$ qu'il est homogène si $(s_1, s_2, s_3) \in \mathcal{R}$, implique $s_2 = s_3$. On notera cette propriété HOMOGENEOUS.

Le λ-cube est une représentation graphique des 8 PTS fonctionnels et homogènes avec deux sortes $\mathcal{S} = \{ \star, \Box \}$, un unique axiome $\mathcal{A} = \{(\star, \Box)\}$, et contenant la règle $(\star, \star, \star)$. Cette présentation est due à Henk Barendregt [Bar91].



Voici leurs règles respectives :

$$\begin{aligned}
\mathcal{R}_{\lambda} &= \{(\star,\star,\star)\} \\
\mathcal{R}_{\mathcal{F}} &= \{(\star,\star,\star),(\Box,\star,\star)\} \\
\mathcal{R}_{\lambda\omega} &= \{(\star,\star,\star),(\Box,\Box,\Box)\} \\
\mathcal{R}_{\mathcal{F}_{\omega}} &= \{(\star,\star,\star),(\Box,\star,\star),(\Box,\Box,\Box)\} \\
\mathcal{R}_{\lambda\Pi} &= \{(\star,\star,\star),(\star,\Box,\Box)\} \\
\mathcal{R}_{\lambda\Pi^2} &= \{(\star,\star,\star),(\Box,\star,\star),(\star,\Box,\Box)\} \\
\mathcal{R}_{\lambda\Pi_{\omega}} &= \{(\star,\star,\star),(\Box,\Box,\Box),(\star,\Box,\Box)\} \\
\mathcal{R}_{\mathrm{CC}} &= \{(\star,\star,\star),(\Box,\star,\star),(\Box,\Box,\Box),(\star,\Box,\Box)\}
\end{aligned}$$

Voici une brève description du rôle que jouent ces règles :

– La règle $(\star,\star,\star)$ de formation des produits entre types de sorte $\star$. Nous donnons à présent un exemple de dérivation qui ne requiert que cette règle (le symbole $(*)$ indique son utilisation). On peut voir que la règle $(\star,\star,\star)$ permet d'autoriser la formation des types simples :

$$\dfrac{\dfrac{\dfrac{}{\vdash \star : \Box}}{\alpha : \star \vdash \alpha : \star}}{\alpha : \star, x : \alpha \vdash x : \alpha} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \star : \Box}}{\alpha : \star \vdash \alpha : \star} \qquad \dfrac{\dfrac{}{\vdash \star : \Box} \qquad \dfrac{}{\vdash \star : \Box}}{\alpha : \star \vdash \star : \Box}}{\alpha : \star, \_ : \alpha \vdash \alpha : \star}}{\alpha : \star \vdash \alpha \to \alpha : \star}\,(*)$$

$$\alpha : \star \vdash \lambda x : \alpha . x : \alpha \to \alpha$$

– Imprédicativité : La règle $(\Box,\star,\star)$ de quantification permet de fabriquer un produit de sorte $\star$ en quantifiant un type de sorte $\star$ par un type de sorte $\Box$. Elle permet donc d'autoriser la quantification sur les variables de types. Il s'agit de la quantification de Système $\mathcal{F}$; elle est dite imprédicative car elle permet de fabriquer un type de sorte $\star$ en quantifiant sur les objets de sorte $\star$. Voici un exemple de son utilisation (le symbole $(*)$ indique l'utilisation de cette règle) :

$$\dfrac{\alpha : \star \vdash \lambda x : \alpha . x : \alpha \to \alpha \qquad \dfrac{\dfrac{}{\vdash \star : \Box} \qquad \alpha : \star \vdash \alpha \to \alpha : \star}{\vdash \forall \alpha : \star . \alpha \to \alpha : \star}\,(*)}{\vdash \lambda \alpha : \star, x : \alpha . x : \forall \alpha : \star . \alpha \to \alpha}$$

– Ordre supérieur : La règle $(\Box,\Box,\Box)$ autorise les constructeurs de type qui dépendent d'autres types. On peut par exemple définir un "constructeur de type" qui prend comme paramètre un type α et retourne le type $\alpha \to \alpha$. Le type de ce constructeur de type est alors $\star \to \star$ qui est lui-même typable en utilisant cette règle. En exemple, la dérivation suivante :

$$\dfrac{\alpha : \star \vdash \alpha \to \alpha : \star \qquad \dfrac{\dfrac{}{\vdash \star : \Box} \qquad \dfrac{\dfrac{}{\vdash \star : \Box} \qquad \dfrac{}{\vdash \star : \Box}}{\_ : \star \vdash \star : \Box}}{\vdash \star \to \star : \Box}\,(*)}{\vdash \lambda \alpha : \star . \alpha \to \alpha : \star \to \star}$$

On peut donner des exemples de constructeurs de type plus utiles : le constructeur de listes de type $\star \to \star$, le produit cartésien et la somme disjointes tous deux de type $\star \to \star \to \star$.

– Dépendance : La règle $(\star, \square, \square)$ autorise les constructeurs de type qui dépendent de valeurs. Elle permet d'écrire un constructeur de type qui prend en paramètre un objet dont le type est de sorte $\star$ comme par exemple dans la dérivation suivante :

$$\dfrac{\dfrac{\dfrac{}{\vdash \star : \square}}{\alpha : \star \vdash \alpha : \star} \qquad \dfrac{\dfrac{}{\vdash \star : \square} \quad \dfrac{}{\vdash \star : \square}}{\alpha : \star \vdash \star : \square}}{\alpha : \star \vdash \alpha \to \star : \square}\,(*)$$

Les types dépendants peuvent s'avérer très utiles en programmation. On peut citer par exemple le constructeur de type des vecteurs qui fabriquent des listes de taille fixée. Le Calcul des Constructions permet de donner de donner le type $\mathsf{nat} \to \star \to \star$ au constructeur de type des vecteurs [2]. Dans la sous-section suivante, on verra que les types dépendants sont également un cadre de choix pour représenter les preuves en logique intuitionniste.

La plupart des systèmes du λ-cube ont été étudiés avant la présentation de Barendregt :

1. λ, le λ-calcul simplement typé :
   Initialement introduit par Alonzo Church en 1940 [Chu40], il constitue le plus simple exemple de calcul typé. Il est important de noter que la seule dérivation valide dans un contexte non-vide est le séquent $\vdash \star : \square$. En effet, dans ce système on ne dispose pas de règle pour former des types sans variable libre.
2. $\mathcal{F}$, le λ-calcul polymorphe, aussi connu sous le nom de système $\mathcal{F}$:
   Introduit indépendamment par Jean-Yves Girard [Gir71] et par John Reynolds [Rey74], il ajoute la quantification imprédicative au λ-calcul simplement typé. Le système obtenu a depuis été très étudié en particulier pour son caractère très expressif.
3. $\mathcal{F}_\omega$, le λ-calcul polymorphe d'ordre supérieur, aussi connu sous le nom de système $\mathcal{F}_\omega$:
   Également introduit par Jean-Yves Girard [Gir72]. Il s'agit d'une extension de système $\mathcal{F}$ dans laquelle on peut exprimer les constructeurs de type.
4. $\lambda\Pi$, aussi connu sous le nom de LF :
   C'est le noyau fonctionnel du système Twelf [PS99] et c'est aussi une approximation raisonnable du système Automath conçu par DeBruijn [Nic70] (voir aussi [KLN] pour une discussion sur la représentation d'Automath dans les PTS).
5. CC, le calcul des constructions de Thierry Coquand et Gérard Huet [CH88] :
   Il s'agit du système le plus complet du λ-cube, à l'origine de l'assistant de preuve Coq. Il a donné lieu à de multiples extensions dont les plus notables sont probablement le Calcul des Constructions Inductives (CiC) [CP88] et le Calcul des Constructions Étendu (ECC) [Luo89].

1.1.5 Programmer dans le λ-cube

Les systèmes du λ-cube peuvent être perçus comme des langages de programmation purement fonctionnels. Néanmoins, ils ne contiennent aucune notion de type de données primitif ; c'est donc au

2. Nous verrons néanmoins que cette présentation n'est pas suffisamment expressive pour que l'on puisse utiliser effectivement le type des vecteurs sans ajouter des types inductifs munis de schéma d'élimination dite "forte".

programmeur de choisir un moyen de les représenter. Les *entiers de Church* constituent un moyen commode pour encoder les entiers en terme d'abstractions et d'applications. Étant donnés un entier n et une expression τ, on définit *l'entier de Church ouvert* $\overline{n}^\tau$ comme étant le terme

$$\lambda f : \tau \to \tau, x : \tau. \overbrace{f\,(f\,\cdots(f}^{n \text{ fois}}\ x)\cdots)$$

On vérifie alors facilement que $\alpha : \star \vdash_\lambda \overline{n}^\alpha : \nu(\alpha)$ où $\nu(\alpha) = (\alpha \to \alpha) \to \alpha \to \alpha$. Plus généralement, on notera $\nu^i(\tau)$ pour le type défini par la récurrence sur i suivante :

$$\begin{array}{rcl} \nu^0(\tau) & = & \nu(\tau) \\ v^{i+1} & = & \nu^i(\tau) \to \nu^i(\tau) \end{array}$$

et on pose par convention $\nu^{-1}(\tau) = \tau \to \tau$ et $\nu^{-2}(\tau) = \tau$. On montre alors facilement que les seuls habitants de $\nu^i(\alpha)$ dans le contexte $\alpha : \star$ sont extensionnellement équivalents à un entier de Church ouvert :

1.1.12 Lemme
Si $\mathcal{P}$ est un système du cube et si $\alpha : \star \vdash_\mathcal{P} A : \nu^i(\alpha)$ pour un certain i, alors il existe un entier n tel que $A\,f\,x \equiv \overline{n}^{\nu^{i-1}(\alpha)}\,f\,x$.

Démonstration Quitte à le réduire, on peut supposer que A est en forme normale (par préservation du typage, voir la sous-section 1.1.6). En inspectant les formes possibles de la dérivation (ce qui se fait formellement à l'aide du lemme 1.1.32 dit "d'inversion" que l'on verra plus avant), on montre que A est nécessairement de la forme $\lambda f : \nu^{i-1}(\alpha).A'$ et que A' est soit égal à f, soit il est de la forme $\lambda x : \nu^{i-2}(\alpha).M$ avec M de type $\nu^{i-2}(\alpha)$ dans le contexte $f : v^{i-1}(\alpha), x : \nu^{i-2}(\alpha)$. Enfin, les seuls termes de type $\nu^{i-2}(\alpha)$ que l'on peut obtenir sont construits en appliquant f un certain nombre de fois à x. On a donc montré que A est :

- Soit de la forme $\lambda f : \nu^{i-1}(\alpha).f$, et donc $A\,f\,x \equiv f\,x \equiv \overline{1}^{\nu^{i-1}}$.
- Soit de la forme $\overline{n}^{\nu^{i-1}(\alpha)}$ pour un certain n et la conclusion est immédiate. ☺

On en déduit donc que les habitants des types de la forme $\nu^i(\alpha)$ représentent tous un entier modulo réduction.

Programmation dans le λ-calcul simplement typé. Dans ce paragraphe, on s'intéressera aux fonctions représentables dans le PTS λ.

On peut implémenter la fonction retournant le successeur des entiers :

$$\begin{array}{rcl} \mathsf{succ}_{i,\alpha} & : & \nu^i(\alpha) \to \nu^i(\alpha) \\ \mathsf{succ}_{i,\alpha} & = & \lambda n : \nu^i(\alpha), f : \nu^{i-1}(\alpha), x : \nu^{i-2}(\alpha).f\,(n\,(f\,x)) \\ \mathsf{succ}_{i,\alpha}\ \overline{n}^{\nu^i(\alpha)} & \rhd & \overline{n+1}^{\nu^i(\alpha)} \end{array}$$

De même on implémente l'addition :

$$\begin{array}{rcl} \mathsf{plus}_{i,\alpha} & : & \nu^i(\alpha) \to \nu^i(\alpha) \to \nu^i(\alpha) \\ \mathsf{plus}_{i,\alpha} & = & \lambda n\,m : \nu^i(\alpha), f : \nu^{i-1}(\alpha), x : \nu^{i-2}(\alpha).n\,f\,(m\,f\,x) \\ \mathsf{plus}_{i,\alpha}\ \overline{n}^{\nu^i(\alpha)}\ \overline{m}^{\nu^i(\alpha)} & \rhd & \overline{n+m}^{\nu^i(\alpha)} \end{array}$$

. . . et la multiplication :

$$\begin{array}{rcl} \mathsf{mult}_{i,\alpha} & : & \nu^i(\alpha) \to \nu^i(\alpha) \to \nu^i(\alpha) \\ \mathsf{mult}_{i,\alpha} & = & \lambda n\, m : \nu^i(\alpha), f : \nu^{i-1}(\alpha), x : \nu^{i-2}(\alpha).n\,(m\,f)\,x \\ \mathsf{mult}_{i,\alpha}\ \overline{n}^{\nu^i(\alpha)}\ \overline{m}^{\nu^i(\alpha)} & \rhd & \overline{n \times m}^{\nu^i(\alpha)} \end{array}$$

La donnée d'un entier est en réalité la donnée d'un opérateur qui permet d'itérer n fois une fonction à un argument. On peut donc utiliser cet opérateur pour construire de nouvelles fonctions. Il est par exemple possible d'itérer une fonction constante pour implémenter une fonction qui retourne 0, si on lui passe un entier nul et 1 sinon :

$$\begin{array}{rcl} \mathsf{sgn}_{i,\alpha} & : & \nu^i(\alpha) \to \nu^i(\alpha) \\ \mathsf{sgn}_{i,\alpha} & = & \lambda n : \nu^i(\alpha), f : \nu^{i-1}(\alpha), x : \nu^{i-2}(\alpha).n\,(\lambda\_ : \nu^{i-2}(\alpha).(f\,x))\,x \\ \mathsf{sgn}_{i,\alpha}\ \overline{n}^{\nu^i(\alpha)} & \trianglerighteq & \overline{\min(1,n)}^{\nu^i(\alpha)} \end{array}$$

On peut utiliser le même principe pour implémenter l'exponentiation en itérant la multiplication :

$$\begin{array}{rcl} \mathsf{exp}_{i,\alpha} & : & \nu^i(\alpha) \to \nu^{i+2}(\alpha) \to \nu^i(\alpha) \\ \mathsf{exp}_{i,\alpha} & = & \lambda n : \nu^i(\alpha), m : \nu^{i+2}(\alpha).m\,(\mathsf{mult}_{i,\alpha}\ n)\overline{1}^{\nu^i(\alpha)} \\ \mathsf{exp}_{i,\alpha}\ \overline{n}^{\nu^i(\alpha)}\ \overline{m}^{\nu^{i+1}(\alpha)} & \rhd & \overline{n^m}^{\nu^i(\alpha)} \end{array}$$

Mais comme nous pouvons le constater, il y a un coût pour pouvoir utiliser un entier comme itérateur : il faut incrémenter l'indice du type de l'entier utilisé comme itérateur.

1.1.13 Définition (Fonction représentable)
On dira d'une fonction (au sens ensembliste) $f : \mathbb{N}^k \longrightarrow \mathbb{N}$ qu'elle est représentable dans un WCTS qui contient une sorte s et la règle $(s, s, s) \in \mathcal{R}$ s'il existe un terme M et des indices i_1, …, i_{k+1} tel que $\alpha : s \vdash M : \nu^{i_1} \to \cdots \to \nu^{i_{k+1}}$ et vérifiant :

$$M\ \overline{n_1}^{\nu^{i_1}}\ \ldots\ \overline{n_1}^{\nu^{i_k}} \rhd \overline{f(n_1, \ldots, n_k)}^{\nu^{i_{k+1}}}$$

On peut prouver qu'il n'existe pas d'implémentation avec un type "uniforme" de l'exponentiation. Il s'agit d'une conséquence du théorème classique suivant (le sens direct de la preuve se montre facilement à l'aide des exemples que l'on a vus jusqu'à présent et nous renvoyons les lecteurs germanophones à [Sch75] pour une preuve de la réciproque) :

1.1.14 Théorème (Helmut Schwichtenberg [Sch75])
Les fonctions représentables dans le λ-calcul simplement typé avec un type de la forme $\nu \to \cdots \to \nu$ sont exactement les polynômes multivariés étendus avec un test d'égalité à zéro. Plus formellement, c'est le plus petit ensemble de fonctions contenant les fonctions constantes, les projections, la fonction $x \mapsto \min(1, x)$ et stable par addition et multiplication point à point.

Les fonctions fabriquées par itération ne peuvent pas être à leur tour itérées car elles n'ont pas un type uniforme (de la forme $\nu^i(\alpha) \to \nu^i(\alpha)$). On ne peut donc pas itérer la fonction $\mathsf{exp}_{i,\alpha}$ par exemple pour implémenter la fonction $(k, n) \mapsto 2^{\cdot^{\cdot^{2^n}}}$ où k est la hauteur de la tour. C'est ce que prouve le théorème suivant :

1.1.15 Théorème (Steven Fortune, Daniel Leivant, et Michael O'Donnell [FLO83])
Les fonctions représentables dans le λ-calcul simplement typé sont bornées par une tour d'exponentielle.

Les auteurs de [FLO83] rapportent que Richard Statman a prouvé que le test d'égalité et d'ordre n'est pas représentable par une expression du λ-calcul simplement typé. Ceci implique que la soustraction des entiers (la fonction $(n, m) \mapsto \max(n - m, 0)$) n'est également pas représentable. De ce fait, l'ensemble des fonctions représentables dans le λ-calcul simplement typé est un sous-ensemble strict des fonctions *élémentaires* (les fonctions bornées par une tour d'exponentielles[3]).

Le λ-calcul simplement typé est donc un langage de programmation peu expressif, *a fortiori* car on ne dispose pas d'un unique type pour représenter les entiers. Nous allons voir à présent que l'ajout de l'impredicativé permet à système $\mathcal{F}$ d'avoir à la fois un type plus naturel pour typer les entiers de Church et de représenter beaucoup plus de fonctions.

Programmer avec l'impredicativé. Notons nat le type $\forall\alpha : \star.(\alpha \to \alpha) \to \alpha \to \alpha$, on définit alors le n-ième entier de Church (fermé) $\overline{n}$ comme étant le terme $\lambda\alpha : \star.\overline{n}^{\alpha}$. Nous obtenons immédiatement que pour tout n, $\vdash_{\mathcal{F}} \overline{n} : \mathsf{nat}$. On peut alors implémenter facilement les opérations de base sur les entiers :

$$\begin{aligned}
\mathsf{succ} &= \lambda n : \mathsf{nat}, \alpha : \star.\, \mathsf{succ}_{0,\alpha}\,(n\,\alpha) \\
\mathsf{plus} &= \lambda n\, m : \mathsf{nat}, \alpha : \star.\, \mathsf{plus}_{0,\alpha}\,(n\,\alpha)\,(m\,\alpha) \\
\mathsf{mult} &= \lambda n\, m : \mathsf{nat}, \alpha : \star.\, \mathsf{mult}_{0,\alpha}\,(n\,\alpha)\,(m\,\alpha) \\
\mathsf{exp} &= \lambda n\, m : \mathsf{nat}, \alpha : \star.\, \mathsf{exp}_{0,\alpha}\,(n\,\alpha)\,(m\,\nu(\alpha))
\end{aligned}$$

En généralisant, on montre aisément que toutes les fonctions représentables dans le λ-calcul simplement typé par un terme f de type $\nu^{i_1}(\alpha) \to \cdots \to \nu^{i_k}(\alpha) \to \nu(\alpha)$ (dans le contexte "$\alpha : \star$") peuvent être représentées dans $\mathcal{F}$ par le terme $\lambda n_1 : \mathsf{nat}, \ldots, n_k : \mathsf{nat}, \alpha : \star.(f\,(n_1\,\nu^{i_1}(\alpha))\,(n_k \cdots \nu^{i_k}(\alpha))$ de type $\mathsf{nat} \to \cdots \to \mathsf{nat} \to \mathsf{nat}$.

La quantification impredicative nous donne un accès beaucoup plus souple à l'itérateur utilisé pour encoder les entiers.

Il est possible de s'en servir pour implémenter le test d'égalité à zéro qui retourne son second argument ou son troisième argument selon que la valeur de son premier argument vaut 0 ou non :

$$\begin{aligned}
\mathsf{is_zero} &: \mathsf{nat} \to \forall\alpha : \star.\alpha \to \alpha \to \alpha \\
\mathsf{is_zero} &= \lambda n : \mathsf{nat}, \alpha : \star, x\, y : \alpha.n\,\alpha\,(\lambda_ : \alpha.y)\,x \\
\mathsf{is_zero}\ \overline{0}\,\alpha\,x\,y &\rhd x \\
\mathsf{is_zero}\ \overline{n+1}\,\alpha\,x\,y &\rhd y
\end{aligned}$$

De plus, on observe que le type de retour de $\mathsf{is\_zero}$ est un bon candidat pour représenter les valeurs boolèenes. Notons $\mathsf{bool} = \forall\alpha : \star.\alpha \to \alpha \to \alpha$. On montre que bool n'a que deux habitants en forme normale $\lambda\alpha : \star, x\, y : \alpha.x$ et $\lambda\alpha : \star, x\, y : \alpha.y$ que l'on nommera respectivement true et false. On peut

3. Les fonctions élémentaires sont le troisième étage ξ_3 de la hiérarchie de Grzegorczyk, voir [Ros] par exemple.

alors implémenter sans difficultés les opérateurs usuels sur les booléens :

$$\begin{array}{rcl} \mathsf{and} & : & \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool} \\ \mathsf{and} & = & \lambda p\, q : \mathsf{bool}\, .p\ \mathsf{bool}\ q\ \mathsf{false} \\ \mathsf{or} & : & \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool} \\ \mathsf{or} & = & \lambda p\, q : \mathsf{bool}\, .p\ \mathsf{bool}\ \mathsf{true}\ q \\ \mathsf{not} & : & \mathsf{bool} \to \mathsf{bool} \\ \mathsf{not} & = & \lambda p : \mathsf{bool}\, .p\ \mathsf{bool}\ \mathsf{false}\ \mathsf{true} \end{array}$$

Et on vérifie facilement qu'ils satisfont les tables de vérité habituelles. Tout comme pour les entiers, cette représentation des booléens contient en elle-même son éliminateur. Ainsi, la construction if standard s'effectue en appliquant au booléen que l'on veut tester un type et deux valeurs de ce type chacune correspondant à l'une des branches du test (c'est par ailleurs ce que l'on a fait pour vérifier la correction de is_zero ci-dessus). Nous pouvons par ailleurs généraliser cet encodage pour représenter les types de données à n éléments par $\forall\alpha : \star.\ \overbrace{\alpha \to \cdots \to \alpha}^{n \text{ fois}} \to \alpha$. On s'intéressera ici surtout au type $\mathsf{unit} = \forall\alpha : \star.\alpha \to \alpha$ à 1 élément noté $\mathsf{id} = \lambda\alpha : \star, x : \alpha.x$ et au type $\mathsf{empty} = \forall\alpha : \star.\alpha$ à 0 élément. On peut alors prouver le lemme :

1.1.16 Lemme
Il y a exactement n habitants en forme normale du type

$$\forall\alpha : \star.\ \overbrace{\alpha \to \cdots \to \alpha}^{n\ fois} \to \alpha$$

dans un contexte vide.

Démonstration Si $\vdash A : \forall\alpha : \star.\ \overbrace{\alpha \to \cdots \to \alpha}^{n \text{ fois}} \to \alpha$ et que A est en forme normale, alors, en étudiant les cas possibles, on montre qu'il est nécessairement de la forme $\lambda\alpha : \star, x_1 : \alpha, \ldots, x_n : \alpha.x_i$ avec $1 \leq i \leq n$. ☺

En particulier, on en déduit que empty ne peut avoir d'habitant dans un contexte vide (car la normalisation du calcul nous donnerait un habitant en forme normale et contredirait le lemme précédent).

Le fait que les types des opérations sur les entiers soient homogènes, permet d'utiliser le schéma d'itération afin obtenir de nouvelles fonctions dont le type sera également homogène. Ainsi, grâce à l'itération, on pourra donc obtenir *ad libitum* des fonctions de plus en plus croissantes. À titre d'exemple, on peut représenter la fonction d'Ackermann[4] qui s'implémente comme une itération de

4. On rappelle que la fonction d'Ackermann est définie par les équations récursives suivante : $ack(0, n) = n + 1$, $ack(n+1, 0) = ack(n, 1)$, $ack(n+1, m+1) = ack(n, ack(n+1, m))$ (la terminaison est triviale, car l'ordre lexicographique des arguments décroît strictement à chaque appel récursif).

fonctions construites par itération :

$$\begin{array}{rcl}
\mathsf{ack} & : & \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \\
\mathsf{ack} & = & \lambda n : \mathsf{nat}\,. \\
 & & \quad n\,(\mathsf{nat} \to \mathsf{nat})(\lambda p : \mathsf{nat}, f : \mathsf{nat} \to \mathsf{nat}, m : \mathsf{nat}\,. \\
 & & \quad\quad m\ \mathsf{nat}\ f\,(f\,1))\ \mathsf{succ} \\
\mathsf{ack}\ \overline{n}\,\overline{m} & \trianglerighteq & \overline{ack(n,m)}
\end{array}$$

L'imprédicativé permet également de représenter des constructeurs de type : étant donnés deux types bien formés $\Gamma \vdash \sigma : \star$ et $\Gamma \vdash \tau : \star$, on peut construire le type bien formé $\mathsf{prod}_{\sigma,\tau}$ défini par $\forall\gamma : \star, (\sigma \to \tau \to \gamma) \to \gamma$ et on a bien $\Gamma \vdash \mathsf{prod}_{\sigma,\tau} : \star$. On notera par la suite $\sigma \times \tau$ le type $\mathsf{prod}_{\sigma,\tau}$. On peut ensuite implémenter le constructeur de paires et les deux projections par les termes suivants :

$$\begin{array}{rcl}
\mathsf{pair}_{\sigma,\tau} & = & \lambda x : \sigma, y : \tau, \gamma : \star, f : \sigma \to \tau \to \gamma . f\,x\,y \\
\mathsf{proj}^k_{\tau_1,\tau_2} & = & \lambda c : \tau_1 \times \tau_2 . c\,\tau_k\,(\lambda x_1 : \tau_1, x_2 : \tau_2 . x_k)
\end{array}$$

Et on vérifie aisément que

$$\mathsf{proj}^k_{\tau_1,\tau_2}\,(\mathsf{pair}_{\tau_1,\tau_2}\ A_1\,A_2) \rhd A_k$$

On dispose également du même genre d'encodage, pour implémenter le type $\sigma + \tau = \forall\alpha : \star, (\sigma \to \gamma) \to (\tau \to \gamma) \to \gamma$ représentant les sommes disjointes, ou encore le type $\mathsf{list}_\tau = \forall\alpha : \star, (\tau \to \alpha \to \alpha) \to \alpha \to \alpha$ des listes, il est en réalité possible de représenter de cette façon tous les types de données récursifs dit "strictement positifs".

Voici les primitives qui permettent de manipuler ces constructeurs de type :

$$\begin{array}{rcl}
\mathsf{left} & : & \forall\alpha\,\beta : \star . \alpha \to \alpha + \beta \\
\mathsf{left} & = & \lambda\alpha\,\beta : \star, x : \alpha, \gamma : \star, f : \alpha \to \gamma, g : \beta \to \gamma . f\,x \\
\mathsf{right} & : & \forall\alpha\,\beta : \star . \beta \to \alpha + \beta \\
\mathsf{right} & = & \lambda\alpha\,\beta : \star, x : \beta, \gamma : \star, f : \alpha \to \gamma, g : \beta \to \gamma . g\,x \\
\mathsf{disj} & : & \forall\alpha\,\beta\,\gamma : \star . \alpha + \beta \to (\alpha \to \gamma) \to (\beta \to \gamma) \to \gamma \\
\mathsf{disj} & = & \lambda\alpha\,\beta\,\gamma : \star, h : a + \beta, f : \alpha \to \gamma, g : \beta \to \gamma . h\,\gamma\,f\,g
\end{array}$$

Les fonctions left et right implémentent les deux injections canoniques et la fonction disj implémente la disjonction de cas : si l'on a deux méthodes, l'une pour fabriquer un objet de type γ à partir d'un objet de type α et l'autre à partir d'un objet de type β, alors on saura fabriquer un objet de type γ à partir d'un objet de type $\alpha + \beta$. On vérifie sans peine que ces fonctions sont conformes au comportement calculatoire attendu :

$$\begin{array}{rcl}
\mathsf{disj}\ \alpha\,\beta\,f\,g\,(\mathsf{left}\ x) & \rhd & f\,x \\
\mathsf{disj}\ \alpha\,\beta\,f\,g\,(\mathsf{right}\ x) & \rhd & g\,x
\end{array}$$

Pour les listes, on obtient un encodage extrêmement proche de celui des entiers :

$$\begin{array}{rcl}
\mathsf{nil} & : & \forall\alpha : \star.\, \mathsf{list}_\alpha \\
\mathsf{nil} & = & \lambda\alpha : \star, \beta : \star, f : \alpha \to \beta \to \beta, x : \beta.x \\
\mathsf{cons} & : & \forall\alpha : \star.\alpha \to \mathsf{list}_\alpha \to \mathsf{list}_\alpha \\
\mathsf{cons} & = & \lambda\alpha : \star, hd : \alpha, tl : \mathsf{list}_\alpha, \beta : \star, f : \alpha \to \beta \to \beta, x : \beta.f\ hd\,(tl\ \beta\, f\, x) \\
\mathsf{fold} & : & \forall\alpha\,\beta : \star.(\alpha \to \beta \to \beta) \to \alpha \to \mathsf{list}_\alpha \to \beta \\
\mathsf{fold} & = & \lambda\alpha\,\beta : \star, f : \alpha \to \beta \to \beta, x : \alpha, l : \mathsf{list}_\alpha\, .l\,\beta\, f\, x
\end{array}$$

Si l'on note $[x_1; \ldots; x_n]$ le terme $\mathsf{cons}\ \alpha\, x_1\, (\mathsf{cons}\ \alpha\, x_2 \ldots (\mathsf{cons}\ \alpha\, x_n\, (\mathsf{nil}\ \alpha) \ldots))$ (dans un contexte où les x_i sont de type α), alors l'on vérifie que l'on obtient bien :

$$\begin{array}{rcl}
\mathsf{nil}\ \alpha & = & [] \\
\mathsf{cons}\ \alpha\, x_0\, [x_1; \ldots; x_n] & = & [x_0; x_1; \ldots; x_n] \\
\mathsf{fold}\ \alpha\, f\, y\, [x_1; \ldots; x_n] & \rhd & f\, x_1\, (f\, x_2 \cdots (f\, x_n\, y) \cdots)
\end{array}$$

On peut par exemple se servir de fold pour implémenter la fonction $\mathsf{fold\ nat\ plus}\ \overline{0}$ de type $\mathsf{list\ nat} \to \mathsf{nat}$ qui calcule la somme des éléments d'une liste d'entiers. On peut également utiliser l'itérateur fold pour implémenter la concaténation de deux listes :

$$\begin{array}{rcl}
\mathsf{concat} & : & \forall\alpha : \star.\, \mathsf{list}_\alpha \to \mathsf{list}_\alpha \to \mathsf{list}_\alpha \\
\mathsf{concat} & = & \lambda\alpha : \star, l_1\, l_2 : \mathsf{list}_\alpha\, .\, \mathsf{fold}\ \alpha\, (\mathsf{cons}\ \alpha)\, l_2\, l_1
\end{array}$$

La présence de constructeurs de type ajoute de l'expressivité au langage ; on peut en particulier s'en servir pour implémenter la fonction prédécesseur et un schéma de récursion. Si l'on note $\langle M, N\rangle$ le terme $\mathsf{pair}_{\sigma,\tau}\ M\, N$ et $(C)^k$ pour $k = 1, 2$ les termes $(\mathsf{proj}^k_{\sigma,\tau}\ C)$ où σ, τ et $\sigma \times \tau$ sont les types respectifs de M, N et C dans le contexte où ils sont définis. Alors la fonction ci-dessous implémente le prédécesseur des entiers :

$$\begin{array}{rcl}
\mathsf{pred} & : & \mathsf{nat} \to \mathsf{nat} \\
\mathsf{pred} & = & \lambda n : \mathsf{nat}\,. \\
& & \Big(n \quad (\mathsf{nat} \times \mathsf{nat}) \\
& & \quad (\lambda c : \mathsf{nat} \times \mathsf{nat}\,.\langle \mathsf{succ}\,(c)^1, (c)^1\rangle) \\
& & \quad \langle \overline{0}, \overline{0}\rangle\Big)^2
\end{array}$$

Elle procède en appliquant n fois la fonction $p, q \mapsto p + 1, q$ sur le couple $(0, 0)$ puis en récupérant la seconde composante. On obtient alors $(0, 0)$, $(1, 0)$, $\ldots$, $(n, n-1)$. On en déduit donc bien que :

$$\begin{array}{rcl}
\mathsf{pred}\ \overline{n+1} & \rhd & \overline{n} \\
\mathsf{pred}\ \overline{0} & \rhd & \overline{0}
\end{array}$$

On peut donc l'itérer pour implémenter la soustraction :

$$\begin{array}{rcl}
\mathsf{minus} & : & \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \\
\mathsf{minus} & = & \lambda n\, m : \mathsf{nat}\,.m\ \mathsf{nat}\ \mathsf{pred}\ n
\end{array}$$

et en déduire le test de supériorité large :

$$\begin{array}{lcl} \mathsf{greater\_or\_equal} & : & \mathsf{nat} \to \mathsf{nat} \to \mathsf{bool} \\ \mathsf{greater\_or\_equal} & = & \lambda n\, m : \mathsf{nat}, x\, y : \alpha.\, \mathsf{is\_zero}\,(\mathsf{minus}\; n\, m)\, x\, y \end{array}$$

ainsi que le test d'égalité :

$$\begin{array}{lcl} \mathsf{equal} & : & \mathsf{nat} \to \mathsf{nat} \to \mathsf{bool} \\ \mathsf{equal} & = & \lambda n\, m : \mathsf{nat}\,.\, \mathsf{and}\,(\mathsf{greater\_or\_equal}\; n\, m)\,(\mathsf{greater\_or\_equal}\; m\, n) \end{array}$$

En suivant la même technique, il est possible d'implémenter le récurseur des entiers :

$$\begin{array}{lcl} \mathsf{rec} & : & \forall\alpha.(\alpha \to \mathsf{nat} \to \alpha) \to \alpha \to \mathsf{nat} \to \alpha. \\ & & \lambda\alpha : \star, f : \alpha \to \mathsf{nat} \to \alpha, x : \alpha. \\ & & \Big(n \quad (\alpha \times \mathsf{nat}) \\ & & \quad (\lambda c : \alpha \times \mathsf{nat}\,.\langle f\,(c)^1\,(c)^2, \mathsf{succ}\,(c)^2\rangle) \\ & & \quad \langle x, \overline{0}\rangle\Big)^1 \end{array}$$

Étant donnée une fonction $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, le récurseur est construit en appliquant n fois la fonction $p, q \mapsto (f(p,q), q+1)$ sur le couple $(x, 0)$ puis en récupérant la première composante du résultat. Elle vérifie donc bien que :

$$\begin{array}{rcl} \mathsf{rec}\;\alpha\, f\, x\, \overline{0} & \rhd & x \\ \mathsf{rec}\;\alpha\, f\, x\, \overline{n+1} & \rhd & \overbrace{f\,(f\,\cdots\,(f}^{n+1 \text{ fois}}\; x\,\overline{0})\,\cdots\,\overline{n-1})\,\overline{n} \end{array}$$

On en déduit que toutes les fonctions primitives récursives[5] sont représentables dans le système $\mathcal{F}$. Ce schéma de récursion est plus général que le schéma de récursion primitive car c'est un schéma d'ordre supérieur : il peut également être utilisé pour générer des fonctions par récursion. On a d'ailleurs vu que l'on pouvait représenter la fonction d'Ackermann qui est essentiellement connue pour ne pas être primitive récursive. En réalité, le schéma d'ordre supérieur prouve que toutes les fonctions représentables dans le système $\mathcal{T}$ de Gödel (voir sous-section 2.4) sont également représentables dans le système $\mathcal{F}$. Le système $\mathcal{F}$ est encore bien plus expressif que système $\mathcal{T}$, en effet, Jean-Yves Girard [Gir72] a prouvé que les fonctions représentables dans système $\mathcal{F}$ sont exactement les fonctions dont on peut prouver la totalité (on dira *prouvablement totale*) dans l'arithmétique du second-ordre $\mathbf{PA}_2$. Tandis que Gödel a prouvé que les fonctions représentable dans système $\mathcal{T}$ sont les fonctions prouvablement totales dans l'arithmétique du premier-ordre **PA**. Nous y reviendrons quand nous étudierons les caractérisations logiques de ces systèmes dans la sous-section 4.2.6.

5. L'ensemble des fonctions primitives récursives est le plus petit ensemble de fonctions qui contient les projections, les fonctions constantes, la fonction successeur et qui est stable par composition et par le schéma de récursion du premier ordre : si $f(\overrightarrow{x})$ et $g(\overrightarrow{x}, y)$ sont primitives récursives alors la fonction $h(n, \overrightarrow{x})$ telle que $h(0, \overrightarrow{x}) = f(\overrightarrow{x})$ et $h(n+1, \overrightarrow{x}) = g(\overrightarrow{x}, h(n, \overrightarrow{x}))$ est primitive récursive.

Apport des types d'ordre supérieur dans $\lambda\omega$ Dans le système $\lambda\omega$, les types d'ordre supérieur ne sont que superficiels, ils n'aident pas à la programmation. En effet, si un terme ne contient pas de rédex formé à l'aide de la règle $(\Box, \Box, \Box)$, alors tous ses réduits n'en contiendrons pas. Les rédexs de type de $\lambda\omega$ forment donc une couche de sucre syntaxique que l'on peut réduire dans une phase préalable ; ils ne font rien de plus que d'internaliser les notations du niveau méta.

Cela nous permet de justifier la définition des trois fonctions ci-dessous.

- La fonction $\mathcal{E}^2$ efface les dépendances dans les types,
- La fonction $\mathcal{E}^1$ applique $\mathcal{E}^2$ aux annotations de type.
- Enfin, la fonction $\mathcal{E}$ applique $\mathcal{E}^2$ aux types de sorte $\star$ du contexte et remplace les types de sorte $\Box$ par $\star$ (qui est le seul habitant de $\Box$ dans λ).

$$
\begin{aligned}
\mathcal{E}(\langle\rangle) &= \langle\rangle \\
\mathcal{E}(\Gamma, x : A) &= \begin{cases} \mathcal{E}(\Gamma), x : \mathcal{E}^2(\mathrm{NF}(A)) & \text{si } \Gamma \vdash A : \star \\ \mathcal{E}(\Gamma), x : \star & \text{si } \Gamma \vdash A : \Box \end{cases} \\
\\
\mathcal{E}^1(\lambda x : A.M) &= \lambda x : \mathcal{E}^2(\mathrm{NF}(A)).\mathcal{E}^1(M) \\
\mathcal{E}^1(M\,N) &= \mathcal{E}^1(M)\,\mathcal{E}^1(N) \\
\mathcal{E}^1(x) &= x \\
\\
\mathcal{E}^2(\forall x : A.B) &= \forall x : \mathcal{E}^2(A).\mathcal{E}^2(B) \\
\mathcal{E}^2(\lambda x : A.B) &= \mathcal{E}^2(B) \\
\mathcal{E}^2(M\,N) &= \mathcal{E}^2(M) \\
\mathcal{E}^2(x) &= x
\end{aligned}
$$

La fonction $\mathcal{E}^2$ se comporte bien uniquement quand elle est appliquée à des termes en forme normale dont le type vit dans $\Box$. Si le terme n'est pas en forme normale, alors elle peut détruire des liaisons. Par exemple si on applique $\mathcal{E}^2$ au terme $M = (\lambda\alpha : \star.\alpha)\,\beta$, on obtient $\mathcal{E}^2(M) = \alpha \notin \mathcal{FV}(M)$. C'est pour éviter ce cas dégénéré que les fonctions $\mathcal{E}$ et $\mathcal{E}^1$ normalisent les types avant de les passer à $\mathcal{E}^2$.

On prouve alors aisément le lemme suivant :

1.1.17 Lemme

- *Si* $\Gamma \vdash_{\lambda\omega} A : K : \Box$, *alors* $\mathcal{E}(\Gamma) \vdash_\lambda \mathcal{E}^2(\mathrm{NF}(A)) : \star$.
- *Si* $\Gamma \vdash_{\lambda\omega} M : A : \star$, *alors* $\mathcal{E}(\Gamma) \vdash_\lambda \mathcal{E}^1(M) : \mathcal{E}^2(\mathrm{NF}(A))$.

À titre d'exemple, si on a :

$$\alpha : \star, l : \star \to \star, c : \alpha \to l\,\alpha \to l\,\alpha, e : l\,\alpha \vdash_{\lambda\omega} \lambda x : \alpha, y : \alpha.c\,x\,(c\,y\,e) : \alpha \to \alpha \to l\,\alpha$$

alors en le lemme précédent nous montre que :

$$\alpha : \star, l : \star, c : \alpha \to l \to l, e : l \vdash_\lambda \lambda x : \alpha, y : \alpha.c\,x\,(c\,y\,e) : \alpha \to \alpha \to l$$

L'effacement $\mathcal{E}^1$ normalise tous les redex de type mais il conserve la structure du niveau objet. Nous en déduisons donc que :

1.1.18 Lemme
Si $\Gamma \vdash_{\lambda\omega} M : A : \star$ *et* $M \rhd M'$, *alors* $\mathcal{E}^1(M) \trianglerighteq \mathcal{E}^1(M')$.

De même la structure des entiers de Church est préservée (seules les annotations de type changent) :

1.1.19 Lemme
Si $\Gamma \vdash A : \star$, *alors* $\mathcal{E}^1(\overline{n}^A) = \overline{n}^{\mathcal{E}^2(\mathrm{NF}(A))}$.

On en conclut donc :

1.1.20 Lemme
Les fonctions représentables dans $\lambda\omega$ *sont exactement les fonctions représentables de* λ.

Démonstration Il faut d'abord remarquer que les entiers de Church se typent dans $\lambda\omega$ avec un type de la forme $\nu(A)$. Ainsi un terme F représente une fonction f dans $\lambda\omega$ et dans un contexte Γ si et seulement s'il existe $A_1, \ldots, A_k, A$, en forme normale tels que

$$\Gamma \vdash_{\lambda\omega} F : \nu(A_1) \to \cdots \to \nu(A_k) \to \nu(A)$$

et pour tous entiers $n_1, \ldots, n_k$ on a $F\,\overline{n_1}^{\nu(A_1)} \ldots \overline{n_k}^{\nu(A_k)} \trianglerighteq \overline{f(n_1, \ldots, n_k)}^{\nu(A)}$. Or d'après les lemmes précédents, on en conclut que

$$\mathcal{E}(\Gamma) \vdash_{\lambda} \mathcal{E}^1(F) : \upsilon(\mathcal{E}^2(A_1)) \to \cdots \to \upsilon(\mathcal{E}^2(A_k)) \to \upsilon(\mathcal{E}^2(A))$$

et que

$$\mathcal{E}^1(F\,\overline{n_1}^{\nu(A_1)} \ldots \overline{n_k}^{\nu(A_k)}) = \mathcal{E}^1(F)\ \overline{n_1}^{\upsilon(\mathcal{E}^2(A_1))} \ldots \overline{n_k}^{\upsilon(\mathcal{E}^2(A_k))} \trianglerighteq \overline{f(n_1, \ldots, n_k)}^{\nu(\mathcal{E}^2(A))}$$

En d'autres termes, $\mathcal{E}^1(F)$ représente f dans λ et dans le contexte $\mathcal{E}(\Gamma)$. ☺

Apport des types d'ordre supérieur dans Système $\mathcal{F}_\omega$. Dans $\mathcal{F}_\omega$, l'ordre supérieur des types apporte un réel gain d'expressivité. Il est possible de s'en servir pour internaliser les opérateurs de type. Ainsi on peut définir dans $\mathcal{F}_\omega$, le terme $\mathsf{prod} = \lambda\alpha\,\beta : \star.\alpha \times \beta$ et on a $\vdash \mathsf{prod} : \star \to \star \to \star$. Mais on peut surtout utiliser l'imprédicativité pour définir des fonctions qui prennent en paramètre un constructeur de type abstrait ainsi qu'une famille de primitives sur ce constructeur. On aura ensuite la garantie que la fonction n'aura pas accès à l'implémentation effective du type. Ce genre d'abstraction est essentielle dans la pratique de la programmation, elle permet aux programmeurs de rendre public uniquement l'interface des bibliothèques qu'ils implémentent afin de se réserver le droit de changer l'implémentation des structures de données et des primitives dans les versions futures. Le système de modules du langage de programmation *Objective Caml* [Ler95, LDF+] implémente un sous-système[6] de $\mathcal{F}_\omega$ pour réaliser ce mécanisme.

À titre d'exemple, on peut construire une fonction M qui prend en paramètre l'implémentation d'une monade donnée par un constructeur de type t de type $\star \to \star$ et ses deux primitives return et bind de type respectif $\forall\alpha : \star.\alpha \to t\,\alpha$ et $\forall\alpha\,\beta : \star.(\alpha \to t\,\beta) \to t\,\alpha \to t\,\beta$. Cette fonction retournera

6. Ce fragment a été choisi de façon à garantir que les annotations de typage (en particulier les applications de type) soit inférées plutôt que données explicitement par le programmeur.

une implémentation des fonctions map et join de types respectifs $\forall \alpha\, \beta : \star.(\alpha \to \beta) \to (t\,\alpha) \to (t\,\beta)$ et $\forall \alpha : \star.t\,(t\,\alpha) \to \alpha$. Le type de M sera donc :

$$\begin{array}{rl} \mathtt{M} : \forall & t : \star \to \star, \\ & \text{return} : \forall \alpha : \star.\alpha \to t\,\alpha, \\ & \text{bind} : \forall \alpha\, \beta : \star.(\alpha \to t\,\beta) \to t\,\alpha \to t\,\beta. \\ & \quad (\forall \alpha\, \beta : \star.(\alpha \to \beta) \to (t\,\alpha) \to (t\,\beta)) \times (\forall \alpha : \star.t\,(t\,\alpha) \to \alpha) \end{array}$$

Et une implémentation possible serait alors :

$$\begin{array}{rl} \mathtt{M} = \lambda & t : \star \to \star, \\ & \text{return} : \forall \alpha : \star.\alpha \to t\,\alpha, \\ & \text{bind} : \forall \alpha\, \beta : \star.(\alpha \to t\,\beta) \to t\,\alpha \to t\,\beta. \\ & \quad \langle \lambda \alpha\, \beta : \star, f : \alpha \to \beta.\, \text{bind}\,(t\,\alpha)\,\beta\,(\lambda x : \alpha.\, \text{return}\,(f\,x)), \\ & \quad\ \lambda \alpha : \star.\, \text{bind}\,(t\,\alpha)\,\alpha\,(\lambda x : t\,\alpha.x) \rangle \end{array}$$

et elle peut être utilisée avec la monade des listes :

$$M\ \mathsf{list}\,(\lambda \alpha : \star, x : \alpha.[x])\,(\lambda \alpha\, \beta : \star, f : \alpha \to \mathsf{list}\ \beta.\, \mathsf{fold}\ \beta\,(\lambda\, hd\ tl\,.\, \mathsf{concat}\ \beta\,(f\ hd)\ tl))$$

Néanmoins Système $\mathcal{F}_\omega$ n'a pas le pouvoir expressif pour garantir que les paramètres passés à M vérifient effectivement les axiomes que l'on attend d'une monade. On verra dans le paragraphe suivant que les types dépendants permettront eux d'exprimer la correction des paramètres par des formules logiques.

Apport des types dépendants : La correspondance de Curry-Howard. Les types dépendants sont introduits en ajoutant la règle $(\star, \Box, \Box)$, elle permet d'inclure au système la possibilité d'avoir des types qui dépendent des individus. Ainsi dans les systèmes qui ne contiennent pas cette règle (la face gauche du λ-cube), les produits formés par la règle $(\star, \star, \star)$ sont tous non-dépendants : Si $\Gamma \vdash \forall x : A.B : \star$ et $\Gamma \vdash A : \star$, alors $x \notin \mathcal{FV}(B)$ et l'on note alors le produit $A \to B$. Ainsi dans $\lambda\Pi$, le plus petit système du cube avec des types dépendants, on peut dériver le séquent ci-dessous. Soit Γ le contexte suivant :

$$\begin{array}{rcl} \Gamma & = & \alpha : \star, \\ & & H : \alpha \to \star, \\ & & G : \alpha \to \star, \\ & & M : \alpha \to \star, \\ & & h_1 : \forall x : \alpha.G\,x \to H\,x \\ & & h_2 : \forall x : \alpha.H\,x \to M\,x \\ & & s : \alpha \end{array}$$

On peut alors dériver :

$$\Gamma \vdash_{\lambda\Pi} \lambda h : G\,s.h_2\,s\,(h_1\,s\,h) : G\,s \to M\,s$$

À présent, si l'on *interprète* α comme étant le type des individus, H comme le prédicat "être un homme", G comme le prédicat "être grec" et M le prédicat "être mortel" et s comme "Socrate", alors on peut voir

le terme $\lambda h : G\,s.h_2\,s\,(h_1\,s\,h)$ comme une preuve du fait que "si Socrate est un grec, alors il est mortel" sous les hypothèses que (h_1) "tous les grecs sont des hommes" et (h_2) "tous les hommes sont mortels". Il s'agit de la correspondance de Curry-Howard [How69] : il est possible de mettre en relation les preuves et les programmes. Dans cette perspective, l'hypothèse h_1 est interprétée comme un programme qui transforme les preuves de nationalité en preuves d'humanité et h_2 comme un programme qui transforme les preuves d'humanité en preuve de mortalité. En composant ces deux programmes, on obtient donc un programme qui transforme les preuves de nationalité en preuve de mortalité.

Comme tous les systèmes de la face inférieure du cube, $\lambda\Pi$ n'est pas très expressif puisque que la sorte $\star$ n'est jamais habitée dans un contexte vide. C'est une fois encore l'imprédicativé qui va permettre d'effectuer des encodages intéressants. Dans $\lambda\Pi^2$ on peut par exemple l'utiliser pour dériver le séquent suivant :

$$\alpha : \star, x : \alpha, y : \alpha \vdash_{\lambda\Pi^2} \forall X : \alpha \to \star.X\,x \to X\,y : \star$$

Si l'on interprète la quantification comme une quantification sur les prédicats d'individus, alors ce type peut se lire comme une proposition affirmant que tous les prédicats vérifiés par x sont vérifiés par y. En d'autres termes, cela dit qu'il n'existe pas de prédicat capable de discerner x et y. Cette proposition qui dépend de α, x et y est en fait une bonne notion d'égalité ; c'est l'égalité de Leibniz : "on dit de deux choses qu'elles sont égales si elles sont indiscernables". On notera donc cette formule $x =_\alpha y$. On peut alors prouver (dans $\lambda\Pi^2$) par les termes ci-dessous que $=_\alpha$ est une relation d'équivalence pour tout α :

$$\begin{aligned}
\mathsf{eq_refl} &: \forall \alpha : \star, x : \alpha.x =_\alpha x \\
\mathsf{eq_refl} &= \lambda \alpha : \star, x : \alpha, X : \alpha \to \star, h : X\,x.h \\
\mathsf{sym} &: \forall \alpha : \star, x\,y : \alpha.x =_\alpha y \to y =_\alpha x \\
\mathsf{sym} &= \lambda \alpha : \star, x\,y : \alpha, h : x =_\alpha y.h\,(\lambda z : \alpha.z =_\alpha x)\,(\mathsf{eq_refl}\ \alpha\,x) \\
\mathsf{trans} &: \forall \alpha : \star, x\,y\,z : \alpha.x =_\alpha y \to y =_\alpha z \to x =_\alpha z \\
\mathsf{trans} &= \forall \alpha : \star, x\,y\,z : \alpha, h_1 : x =_\alpha y, h_2 : y =_\alpha z.h_2\,(\lambda z : \alpha.x =_\alpha z)\,h_1
\end{aligned}$$

Si l'on ajoute l'ordre supérieur des types à $\lambda\Pi^2$, on obtient le Calcul des Constructions CC dans lequel on peut définir le terme $\mathsf{eq} = \lambda \alpha : \star, x\,y : \alpha.x =_\alpha y$ et on a $\vdash_{\mathrm{CC}} \mathsf{eq} : \forall \alpha : \star.\alpha \to \alpha \to \star$ qui peut être considéré comme un constructeur de propositions : eq construit une proposition (l'égalité) à partir d'un type et de deux individus de ce type.

Les encodages imprédicatifs des constructeurs de type de Système $\mathcal{F}_\omega$ vus précédemment prennent une nouvelle dimension dans CC. En effet, ils peuvent être perçus à travers la correspondance de Curry-Howard comme des encodages des connecteurs logiques propositionnels. Ainsi, le produit cartésien $A \times B$ s'interprète par la conjonction de A et B et la paire construite à partir d'une preuve de A et de B et du constructeur pair constituera une preuve d'une conjonction. De la même façon, les projections proj^1 et proj^2 implémentent les règles d'élimination habituelles de la conjonction. La somme disjointe que l'on notait $A + B$ implémente la disjonction entre deux propositions, left et right implémentent les règles d'introduction et disj la règle d'élimination de la disjonction logique. Enfin le type unit à un élément représentera la proposition toujours vérifiée (elle a une unique preuve) et empty représentera l'absurdité logique (elle n'a aucune preuve). On remarque que le lemme 1.1.16, est donc une preuve de la cohérence de Calcul des Constructions, car elle implique que le type vide qui représente l'absurdité logique n'est pas prouvable.

Il y a donc bien une superposition complète entre la notion de proposition et celle de type de données, dans le Calcul des Constructions présenté. Nous verrons plus bas qu'il existe une variante dans laquelle on sépare la sorte $\star$ en deux sortes `Prop` et `Set`, l'une servant à représenter les propositions et l'autre, les types de données. Nous utiliserons alors des notations différentes pour les distinguer : nous garderons $(\times) : \texttt{Set} \to \texttt{Set} \to \texttt{Set}$ pour le produit cartésien et nous utiliserons $(\wedge) : \texttt{Prop} \to \texttt{Prop} \to \texttt{Prop}$ pour la conjonction. De la même façon, pour les autres connecteurs, nous suivrons le tableau suivant :

Set	Prop
$\times$	$\wedge$
$+$	$\vee$
unit	$\top$
empty	$\bot$

Dans le calcul des constructions avec une unique sorte, nous utiliserons l'une ou l'autre notation selon que l'on veut interpréter le type considéré comme un type de données ou une proposition.

L'imprédicativité permet également d'encoder le quantificateur existentiel suivant :

$$\begin{array}{rcl}
\mathsf{ex} & : & \forall \alpha : \texttt{Prop}.(\alpha \to \texttt{Prop}) \to \texttt{Prop} \\
\mathsf{ex} & = & \lambda \alpha : \texttt{Prop}, P : \alpha \to \texttt{Prop}.\forall X : \texttt{Prop}.(\forall x : \alpha.P\,x \to X) \to X \\
\mathsf{ex_intro} & : & \forall \alpha : \star, P : \alpha \to \star, x : \alpha.P\,x \to \mathsf{ex}\ \alpha\, P \\
\mathsf{ex_intro} & = & \lambda \alpha : \star, P : \alpha \to \star, x : \alpha, h : P\,x, X : \texttt{Prop}, f : \forall x : \alpha.P\,x \to X.f\,x\,h \\
\mathsf{ex_elim} & : & \forall \alpha : \star, P : \alpha \to \star.\mathsf{ex}\ \alpha\, P\ \to \forall X : \texttt{Prop}.(\forall x : \alpha.P\,x \to X) \to X \\
\mathsf{ex_elim} & : & \lambda \alpha : \star, P : \alpha \to \star, c : \mathsf{ex}\ \alpha\, P, X : \texttt{Prop}, f : \forall x : \alpha.P\,x \to X.c\,X\,f
\end{array}$$

Nous pouvons alors noter $\exists x : A.P$ le terme $\mathsf{ex}\ A\,(\lambda x : A.P)$ et les termes $\mathsf{ex\_intro}$ et $\mathsf{ex\_elim}$ représentent respectivement le règle d'introduction et d'élimination de ce nouveau connecteur. Il est important de noter que ce nouveau connecteur n'est pas une généralisation dépendante du produit cartésien, car on ne peut pas construire de terme qui implémente la seconde projection. On peut programmer la première projection de la manière suivante :

$$\begin{array}{rcl}
\pi_1 & : & \forall \alpha : \texttt{Prop}, P : \alpha \to \texttt{Prop}\,.\,\mathsf{ex}\ \alpha\, P \to \alpha \\
\pi_1 & = & \lambda \alpha : \texttt{Prop}, P : \alpha \to \texttt{Prop}, c : \mathsf{ex}\ \alpha\, P.c\,\alpha\,(\lambda x : \alpha, h : P\,x.x)
\end{array}$$

Mais il a été démontré [Coq86] que toute tentative d'ajout d'un terme π_2 de type $\forall \alpha : \texttt{Prop}, P : \alpha \to \texttt{Prop}, c : \mathsf{ex}\ \alpha\, P.P\,(\pi_1\,\alpha\,P\,c)$ vérifiant $\Pi_1\,\alpha\,P\,(\mathsf{ex\_intro}\ \alpha\,P\,x\,h) \equiv h$ rendrait le calcul incohérent. Enfin, notons également qu'il n'es pas possible de représenter le quantificateur du second ordre "$\exists X : \texttt{Prop}.P$" dans ce calcul, mais nous verrons que ce sera possible en utilisant un univers (c'est-à-dire une sorte au dessus de $\Box$) dans le Calcul des Constructions avec univers présenté dans la sous-section 1.4.1.

Revenons à notre exemple de "module" `M` du paragraphe précédent (page 30) qui implémente les opérations map et join à partir d'un constructeur de t sensé représenter une monade et des deux opérations return et bind. Dans le calcul des construction, on peut utiliser le pouvoir expressif du système pour formaliser la spécifications des types de données et de leurs primitives.

Considérons le contexte Γ suivant :

$$
\begin{aligned}
\Gamma = \quad & t : \star \to \star, \\
& \text{return} : \forall \alpha : \star . \alpha \to t\,\alpha, \\
& \text{bind} : \forall \alpha\,\beta : \star . (\alpha \to t\,\beta) \to t\,\alpha \to t\,\beta, \\
& h_1 : \forall \alpha : \star, x : \alpha . \text{bind}\ \alpha\,\alpha\,(\text{return}\ \alpha)\,x\ =_{t\,\alpha}\ x, \\
& h_2 : \forall \alpha\,\beta : \star, f : \alpha \to \beta, x : \alpha . \text{bind}\ \alpha\,\beta\,f\,(\text{return}\ \alpha\,x)\ =_{t\,\beta}\ f\,x, \\
& h_3 : \forall \alpha\,\beta\,\gamma : \star, f :\ \alpha \to t\,\beta, g : \beta \to t\,\gamma, m : t\,\alpha, \\
& \qquad \text{bind}\ \beta\,\gamma\ g\,(\text{bind}\ \alpha\,\beta\,f\,m) =_{t\,\gamma} \text{bind}\ \alpha\,\gamma\,(\lambda x : \alpha . \text{bind}\ \beta\,\gamma\,g\,(f\,x))\,m, \\
& h_4 : \forall \alpha\beta : \star, f\,g : \alpha \to t\,\beta, m : t\,\alpha, \\
& \qquad (\forall x : \alpha . (f\,x) =_{\alpha} (g x)) \to (\text{bind}\ \alpha\,\beta\,f\,m) =_{t\,\beta} (\text{bind}\ \alpha\,\beta\,g\,m). \\
& (\forall \alpha\,\beta : \star . (\alpha \to \beta) \to (t\,\alpha) \to (t\,\beta)) \times (\forall \alpha : \star . t\,(t\,\alpha) \to \alpha)
\end{aligned}
$$

Être sous le contexte Γ c'est disposer d'un opérateur de type t, de ses deux primitives return et bind ainsi que de quatre hypothèse h_1, h_2, h_3 et h_4 prouvant que les opérations return et bind satisfont bien les axiomes attendus d'une monade. Ainsi h_1 devra prouver que return est un élément neutre à gauche pour bind, h_2 prouver que c'est un élément neutre à droite, h_3 prouvera que bind est associatif et enfin h_4 prouvera que bind retourne le même résultat si on lui passe comme argument des fonctions extensionnellement égales.

Posons map $= (M\,t\ \text{return bind})_1$ et join $= (M\,t\ \text{return bind})_2$, alors on peut construire dans CC des preuves de types suivants (nous conseillons vivement au lecteur qui voudrait les retrouver d'utiliser un assistant de preuve) :

$$
\begin{aligned}
\texttt{map\_id} \quad &: \quad \forall \alpha : \star, m : t\,\alpha, \text{map}\ \alpha\,\alpha\ \mathsf{id}\ m =_{t\,\alpha} m \\
\texttt{map\_comp} \quad &: \quad \forall \alpha\beta : \star, f : \beta \to \alpha, g : \alpha \to \beta, m : t\alpha. \\
&\qquad \text{map}\ \alpha\,\alpha\,(\lambda x : \alpha . f\,(g\,x))\,m =_{t\,\alpha} \text{map}\ \beta\,\alpha\,f\,(\text{map}\ \alpha\,\beta\,g\,m) \\
\texttt{map\_return} \quad &: \quad \forall \alpha\,\beta : \star, f : \alpha \to \beta, x : \alpha . \text{return}\ \beta\,(f\,x) =_{t\,\beta} \text{map}\ \alpha\,\beta\,f\,(\text{return}\ \alpha\,x). \\
\texttt{join\_map} \quad &: \quad \forall \alpha : \star, m : t(t(t\alpha)) . \text{join}\ \alpha\,(\text{join}\,(t\,\alpha)\,m) =_{t\,\alpha} \text{join}\ \alpha\,(\text{map}\,(t\,(t\,\alpha))\,(t\,\alpha)\,(\text{join}\ \alpha)\,m) \\
\texttt{join\_return} \quad &: \quad \forall \alpha (m :\ t\alpha) . \text{join}\ \alpha\,(\text{map}\ \alpha\,(t\,\alpha)\,(\text{return}\ \alpha)\,m) =_{t\,\alpha} m \\
\texttt{join\_join} \quad &: \quad \forall \alpha\,\beta : \star, m : t(t\alpha), f : \alpha \to \beta. \\
&\qquad \text{join}\ \beta\,(\text{map}\,(t\,\alpha)\,(t\,\beta)\,(\text{map}\ \alpha\,\beta\,f)\,m) =_{t\,\beta} \text{map}\ \alpha\,\beta\,f\,(\text{join}\ \alpha\,m)
\end{aligned}
$$

Ce que l'on peut paraphraser en :

- Les preuves `map_id` et `map_comp` montrent que t a une structure de *foncteur* (au sens catégorique).
- Les preuves `map_return` et `join_map` montrent que return et join sont des transformations naturelles vis-à-vis du foncteur t.
- Enfin `join_return` et `join_join` montrent respectivement que return est un élément neutre pour join et que join est associative.

Maintenant si l'on veut utiliser ces preuves pour prouver qu'une instance particulière, comme par exemple $t = \mathsf{list}$, return $= \lambda \alpha : \star, x : \alpha . [x]$) et

$$\text{bind} = (\lambda \alpha\,\beta : \star, f : \alpha \to \mathsf{list}\ \beta . \mathsf{fold}\ \beta\,(\lambda\, hd\ tl\,.\,\mathsf{concat}\ \beta\,(f\ hd)\ tl))$$

il faut alors fournir des preuves concrètes pour h_1, h_2 et h_3 et h_4. Nous verrons dans le prochain paragraphe que ce n'est pas possible sans rajouter d'axiomes au Calcul des Constructions. Nous verrons que l'on touche ici aux limites des encodages imprédicatifs.

Les types dépendants sont donc très utiles pour représenter les preuves et ils peuvent servir à rajouter des contraintes de spécification dans les types. Mais il ne permettent néanmoins pas de représenter plus de fonctions : on peut effacer les dépendances des termes dans les types pour obtenir un type moins contraint. De cette façon, on projette naturellement tous les systèmes de la face droite du cube vers la face gauche.

Nous allons maintenant esquisser à gros traits le fonctionnement de cette projection, nous renvoyons le lecteur à [Bar92, PM89b, PM89a] pour une description plus détaille de la fonction d'effacement des dépendances (dans le cas CC vers $\mathcal{F}_\omega$).

On pose :

$$\begin{array}{rcl}
\mathcal{E}^{\mathrm{dep}}(\langle\rangle) & = & \langle\rangle \\
\mathcal{E}^{\mathrm{dep}}(\Gamma, x:A) & = & \mathcal{E}^{\mathrm{dep}}(\Gamma), x : \mathcal{E}^{\mathrm{dep}}_\Gamma(A) \\
 & & \\
\mathcal{E}^{\mathrm{dep}}_\Gamma(\forall x:A.B) & = & \mathcal{E}^{\mathrm{dep}}_{\Gamma,x:A}(B) \text{ si } \Gamma \vdash A : \star, \text{ et } \Gamma, x:A \vdash B : \Box \\
\mathcal{E}^{\mathrm{dep}}_\Gamma(\lambda x:A.M) & = & \mathcal{E}^{\mathrm{dep}}_{\Gamma,x:A}(M) \text{ si } \Gamma \vdash A : \star, \text{ et } \Gamma, x:A \vdash M : B : \Box \\
\mathcal{E}^{\mathrm{dep}}_\Gamma(M\,N) & = & \mathcal{E}^{\mathrm{dep}}_\Gamma(M) \text{ si } \Gamma \vdash M : \forall x:A.B, \Gamma \vdash \forall x:A.B : \Box \text{ et } \Gamma \vdash A : \star \\
 & & \text{sinon :} \\
\mathcal{E}^{\mathrm{dep}}_\Gamma(M\,N) & = & \mathcal{E}^{\mathrm{dep}}_\Gamma(M)\,\mathcal{E}^{\mathrm{dep}}_\Gamma(N) \\
\mathcal{E}^{\mathrm{dep}}_\Gamma(\lambda x:A.B) & = & \lambda x : \mathcal{E}^{\mathrm{dep}}_\Gamma(A).\mathcal{E}^{\mathrm{dep}}_{\Gamma,x:A}(B) \\
\mathcal{E}^{\mathrm{dep}}_\Gamma(\forall x:A.B) & = & \forall x : \mathcal{E}^{\mathrm{dep}}_\Gamma(A).\mathcal{E}^{\mathrm{dep}}_{\Gamma,x:A}(B) \\
\mathcal{E}^{\mathrm{dep}}_\Gamma(s) & = & s \\
\mathcal{E}^{\mathrm{dep}}_\Gamma(x) & = & x
\end{array}$$

Les fonctions $\mathcal{E}^{\mathrm{dep}}_\Gamma(\cdot)$ et $\mathcal{E}^{\mathrm{dep}}(\cdot)$ ne sont bien définis que lorsque que Γ est bien formé (ce que l'on notera plus avant $\mathrm{WF}(\Gamma)$) et aussi parce que nous avons l'unicité du typage dans le cube.

On utilisera également la notation $\mathcal{E}^{\mathrm{dep}}(\mathcal{P})$ pour désigner la projection des systèmes de la face droite vers la face gauche :

$$\begin{array}{rcl}
\mathcal{E}^{\mathrm{dep}}(\lambda\Pi) & = & \lambda \\
\mathcal{E}^{\mathrm{dep}}(\lambda\Pi_\omega) & = & \lambda\omega \\
\mathcal{E}^{\mathrm{dep}}(\lambda\Pi^2) & = & \mathcal{F} \\
\mathcal{E}^{\mathrm{dep}}(\mathrm{CC}) & = & \mathcal{F}_\omega
\end{array}$$

Nous montrons alors facilement par induction que la face droite du cube est envoyée vers la face gauche :

1.1.21 Lemme
Si $\mathcal{P} \in \{\lambda\Pi, \lambda\Pi_\omega, \lambda\Pi_\omega, \lambda\Pi^2\}$, *on a :*

$$\Gamma \vdash_{\mathcal{P}} A : B \text{ implique } \mathcal{E}^{dep}(\Gamma) \vdash_{\mathcal{E}^{dep}(\mathcal{P})} \mathcal{E}^{dep}_{\Gamma}(A) : \mathcal{E}^{dep}_{\Gamma}(B).$$

Tout comme la fonction $\mathcal{E}^1$ d'effacement de l'ordre supérieur dans $\lambda\omega$, la fonction $\mathcal{E}^{\text{dep}}$ préserve les réductions :

$$A \rhd B \quad \Rightarrow \quad \mathcal{E}^{\text{dep}}_{\Gamma}(A) \trianglerighteq \mathcal{E}^{\text{dep}}_{\Gamma}(B)$$

et conserve le type des entiers de Church :

$$\Gamma \vdash_{\mathcal{P}} A : \star \Rightarrow \mathcal{E}^{\text{dep}}_{\Gamma}(\overline{n}^A) = \overline{n}^{\mathcal{E}^{\text{dep}}_{\Gamma} A}$$

Nous en concluons donc que

1.1.22 Lemme
Si $\mathcal{P} \in \{\lambda\Pi, \lambda\Pi_\omega, \text{CC}, \lambda\Pi^2\}$, alors les fonctions représentables dans $\mathcal{P}$ sont exactement les fonctions représentables de $\mathcal{E}^{dep}(\mathcal{P})$.

Pour résumer, il n'y a que trois classes de fonctions différentes représentables par les systèmes du λ-cube : les fonctions représentables dans λ, les fonctions représentables dans $\mathcal{F}$ et celles représentables dans $\mathcal{F}_\omega$.

Les limites des encodages imprédicatifs dans le Calcul des Constructions. Lorsque l'on cherche à faire des preuves dans le Calcul des Constructions en utilisant les encodages imprédicatifs pour représenter les données, on tombe rapidement sur les deux problèmes suivants :

- **L'existence de modèles triviaux.** Il est possible de construire un modèle ensembliste du Calcul des Constructions dans lequel la sorte $\star$ est interprétée par l'ensemble à deux éléments $\{\emptyset, \{\emptyset\}\}$. Dans une telle interprétation, tous les habitants de $\star$ sont interprétés comme des ensembles à au plus un élément [MW02]. Dans ce modèle les interprétations des termes true et false sont égales et puisque l'égalité de Leibniz y est effectivement interprétée par l'égalité ensembliste, l'existence de ce modèle implique que la proposition $\mathsf{true} \neq_{\mathsf{bool}} \mathsf{false}$ ne peut pas être démontable (c'est-à-dire habitable) dans le Calcul des Constructions. En conséquence, le modèle satisfait également $\overline{0} =_{\mathsf{nat}} \overline{1}$ ce qui permet de conclure que $\overline{0} \neq_{\mathsf{bool}} \overline{1}$ n'est pas démontrable. Le calcul des constructions n'est donc pas suffisamment puissant pour contenir l'arithmétique, ce qui limite beaucoup son expressivité. Un moyen de pallier ce problème est d'étendre le système avec des types inductifs munis de schémas d'élimination forte. Nous verrons dans le chapitre 2, que l'on peut se servir de ces derniers pour prouver que $\mathsf{true} \neq_{\mathsf{nat}} \mathsf{false}$.
- **L'existence de modèles non-paramétriques.** Pour justifier que le type des booléens bool ne contenait moralement que deux éléments, nous avons utilisé un argument méta-théorique : il n'existe que deux termes différents en forme normale qui habitent le type bool. Or, sans l'aide d'axiomes, cet argument ne peut pas s'internaliser dans le Calcul des Constructions. En effet, s'il est effectivement impossible de construire par la syntaxe un terme de type bool qui ne se normalise ni sur true, ni sur false, on peut toute fois avoir dans l'interprétation du type bool des fonctions différentes à la fois de l'interprétation de true et de l'interprétation de false. En conséquence, la proposition $\forall x : \mathsf{bool}, x =_{\mathsf{bool}} \mathsf{true} \vee x =_{\mathsf{bool}} \mathsf{false}$ ne sera pas démontrable dans le calcul des constructions. Nous verrons toutefois dans le chapitre 5 que l'on peut exprimer dans le Calcul des Constructions les conditions nécessaires d'uniformité à satisfaire pour que les habitants des encodages imprédicatifs aient un comportement analogue à ceux construits par la syntaxe.

1.1.6 Métathéorie

Le typage des expressions dans les WCTS satisfait la plupart des propriétés que l'on attend d'un système de types à l'exception de la préservation du typage et de la propriété de renforcement qui restent des problèmes ouverts (nous verrons néanmoins des conditions raisonnables pour obtenir ces propriétés dans les cas qui nous intéressent).

1.1.23 Lemme (Substitution)

Si $\Gamma, x : A, \Delta \vdash B : C$ *et* $\Gamma \vdash D : A$, *alors* $\Gamma, \Delta[D/x] \vdash B[D/x] : C[D/x]$.

Démonstration Le théorème s'obtient aisément par induction sur la structure de la dérivation $\Gamma, x : A, \Delta \vdash B : C$ (voir [Bar92] lemme 5.2.11). Le cas des règles de cumulativité ne pose ici aucun de problème (on utilise le lemme 1.1.6). ☺

1.1.24 Lemme (Affaiblissement généralisé)
La règle suivante est admissible :

$$\frac{\Gamma, \Delta \vdash M : A \qquad \Gamma \vdash B : s}{\Gamma, x : B, \Delta \vdash M : A}\, x \notin \Gamma, \Delta$$

Démonstration On prouve le résultat par induction sur la dérivation de $\Gamma, \Delta \vdash M : A$ en supposant $\Gamma \vdash B : s$ et $x \notin \Gamma, \Delta$. L'induction est facile, on ne traitera que les cas Axiome, Variable et Cumulativité :

- Axiome : Dans ce cas, on a $\Gamma, \Delta = \langle\rangle$, $M = r$, $r' = A$ et $(r, r') \in \mathcal{A}$. Et l'on doit prouver $x : B \vdash r : r'$, on utilise pour ça la règle Affaiblissement :

$$\frac{\dfrac{}{\vdash r : r'} \qquad \vdash B : s}{x : B \vdash r : r'}$$

- Variable : Il faut distinguer les deux cas possibles :
  - Soit $\Delta = \langle\rangle$, et Γ est de la forme $\Gamma', y : A$ avec $\Gamma' \vdash A : r$ pour un certain r et $x \neq y$. On montre alors $\Gamma, x : B \vdash y : A$ grâce à la règle Affaiblissement :

$$\frac{\dfrac{\Gamma' \vdash A : r}{\Gamma', y : A \vdash y : A} \qquad \Gamma \vdash B : s}{\Gamma', y : A, x : B \vdash y : A}$$

  - Soit Δ est de la forme $\Delta', y : A$ avec $\Gamma, \Delta' \vdash A : r$ pour un certain r et $x \neq y$. Alors par hypothèse d'induction on en déduit que $\Gamma, x : B, \Delta' \vdash A : r$ et d'après la règle Variable, on a bien $\Gamma, x : B, \Delta \vdash y : A$.
- Cumulativité : Dans ce cas on a $\Gamma, \Delta \vdash M : A'$, avec $\Gamma, \Delta \vdash A : r$ pour un certain r et $A' \preccurlyeq A$. Par hypothèse d'induction, on en déduit que $\Gamma, x : B, \Delta \vdash M : A'$ et que $\Gamma, x : B, \Delta \vdash A : r$, et donc d'après la règle Cumulativité, on en déduit $\Gamma, x : B, \Delta \vdash M : A$. ☺

1.1.25 Définition (Type bien formé)
On notera $\mathrm{WF}_\Gamma(A)$ pour signifier que le terme A est un type bien formé, c'est-à-dire que A est soit syntaxiquement égal à une sorte, soit A admet comme type dans Γ une sorte (il existe s tel que $\Gamma \vdash A : s$). Et on note $\mathrm{WF}(\Gamma)$ le fait qu'un contexte soit bien formé, c'est-à-dire $\mathrm{WF}(\Gamma)$ s'il existe A et B tel que $\Gamma \vdash A : B$.

Un contexte est bien formé si et seulement si les types qui y figurent ont tous pour type une sorte (en particulier si $x : s \in \Gamma$ et $\mathrm{WF}(\Gamma)$, alors s ne peut pas être une sorte maximale) :

1.1.26 Lemme
On a $\mathrm{WF}(x_1 : A_1, \ldots, x_n : A_n)$ *si et seulement s'il existe* $s_1, \ldots, s_n$ *telles que pour tout* $1 \leq i \leq n$ *on a* $x_1 : A_1, \ldots, x_{i-1} : A_{i-1} \vdash A_i : s_i$.

Démonstration On note Γ_i le contexte $x_1 : A_1, \ldots, x_i : A_i$.

$\Rightarrow$: On procède par induction sur $\Gamma_n \vdash A : B$, on traite uniquement les cas Affaiblissement et Abstraction (les autres cas sont tout aussi immédiats) :

- Affaiblissement : Dans ce cas, on a $\Gamma_{n-1} \vdash A : B$ et $\Gamma_{n-1} \vdash A_n : s_n$ pour un certain s_n. Par hypothèse d'induction, on sait qu'il existe $s_1, \ldots, s_n$ telles que pour tout $1 \leq i \leq n-1$ on ait $\Gamma_{i-1} \vdash A_i : s_i$ et on utilise s_n pour conclure.
- Abstraction : Dans ce cas, on a A de la forme $\lambda x_{n+1} : A_{n+1}.A'$ et B de la forme $\forall x : A_{n+1}.B'$ avec $\Gamma_n, x_{n+1} : A_{n+1} \vdash A' : B'$. Par hypothèse d'induction on sait qu'il existe $s_1, \ldots, s_{n+1}$ telles que pour tout $1 \leq i \leq n+1$ on ait $\Gamma_{i-1} : A_{i-1} \vdash A_i : s_i$. C'est donc vérifié en particulier $1 \leq i \leq n$.

$\Leftarrow$: On peut montrer facilement que si $\mathcal{A}$ est vide alors le WCTS ne type aucun terme dans aucun contexte (c'est-à-dire pour tout Γ, A, B on $\Gamma \not\vdash A : B$). On peut donc supposer qu'il existe $(s, r) \in \mathcal{A}$. On montre alors que $\Gamma \vdash s : r$ en enchaînant n règles Affaiblissement :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash s : r} \quad \vdash A_1 : s_1}{\Gamma_1 \vdash s : r}}{\vdots}}{\dfrac{\Gamma_{n-2} \vdash A_{n-1} : s_{n-1}}{\Gamma_{n-1} \vdash s : r} \qquad \Gamma_{n-1} \vdash A_n : s_n}{\Gamma_n \vdash s : r}$$

☺

Soit Δ et Γ deux contextes, on note $\Delta \subseteq \Gamma$ si et seulement si Δ peut être obtenu à partir de Γ en enlevant certaines déclarations. Ce que l'on peut formaliser par les règles suivantes :

$$\frac{}{\langle\rangle \subseteq \langle\rangle} \qquad \frac{\Delta \subseteq \Gamma}{\Delta, x : A \subseteq \Gamma, x : A} \qquad \frac{\Delta \subseteq \Gamma}{\Delta \subseteq \Gamma, x : A}$$

On montre alors facilement que $\Delta \subseteq \Gamma$ si et seulement si Γ peut s'écrire de la forme $\Gamma = \Gamma_1, x_1 : A_1, \Gamma_2, x_2 : A_2, \ldots, \Gamma_{n-1}, x_n : A_n, \Gamma_n$ avec $\Delta = \Gamma_1, \Gamma_2, \ldots, \Gamma_n$.

On en déduit la généralisation du lemme 1.1.24 suivante :

1.1.27 Lemme (Affaiblissement généralisé étendu)
Si $\Delta \subseteq \Gamma$ *et* $\Delta \vdash A : B$ *avec* $\mathrm{WF}(\Gamma)$*, alors* $\Gamma \vdash A : B$*.*

Démonstration Supposons $\Delta \subseteq \Gamma$, alors nous avons vu que Γ peut s'écrire de la forme $\Gamma = \Gamma_1, x_1 : A_1, \Gamma_2, x_2 : A_2, \ldots, \Gamma_{n-1}, x_n : A_n, \Gamma_n$ avec $\Delta = \Gamma_1, \Gamma_2, \ldots, \Gamma_n$. On procède par récurrence sur n :
- Supposons $n = 0$, alors $\Gamma = \Delta = \langle\rangle$ et on a $\Delta \vdash A : B$ implique $\Gamma \vdash A : B$.
- Supposons Γ de la forme $\Gamma', x_{n+1} : A_{n+1}, \Gamma_{n+1}$ avec $\Gamma' = \Gamma_1, x_1 : A_1, \Gamma_2, x_2 : A_2, \ldots, \Gamma_n$ et $\Delta' = \Gamma_1, \Gamma_2, \ldots, \Gamma_n$ et $\Delta = \Delta', \Gamma_{n+1}$. Par hypothèse d'induction, on obtient que $\Gamma', \Gamma_{n+1} \vdash A : B$. Alors, comme on a $\mathrm{WF}(\Gamma)$ on obtient d'après le lemme précédent que $\Gamma' \vdash A_n : s$ pour un certain s. De plus, comme nous avons $\Delta', \Gamma_{n+1} \vdash A : B$, nous en déduisons que $x_{n+1} \notin \mathcal{FV}(A)$ et donc, d'après le lemme 1.1.24, on obtient $\Gamma', x_{n+1} : A_{n+1}, \Gamma_{n+1} \vdash A : B$. ☺

La notion de type bien formé est souvent commode, elle permet en particulier de subsumer les deux règles de cumulativité :

1.1.28 Lemme (Lemme de cumulativité étendus)
La règle suivante est admissible :

$$\frac{\Gamma \vdash A : B \qquad \mathrm{WF}_\Gamma(B')}{\Gamma \vdash A : B'}\, B \preccurlyeq B'$$

Dans les CTS, on a $\preccurlyeq^* = \preccurlyeq$ (voir le lemme 1.2.11 démontré plus loin dans la section 1.2), nous verrons dans la section 1.2 que si $\mathcal{C}$ n'est pas transitive, alors la règle suivante n'est en général pas admissible :

$$\frac{\Gamma \vdash A : B \qquad \mathrm{WF}_\Gamma(B')}{\Gamma \vdash A : B'}\, B \preccurlyeq^* B'$$

C'est pourquoi nous introduisons la définition suivante :

1.1.29 Définition (Relation de cumulativité entre un terme et un type bien formé)
On note $\preccurlyeq_\Gamma$ la relation définie par :

$$A \preccurlyeq_\Gamma B \quad \Leftrightarrow \quad A \preccurlyeq B \wedge \mathrm{WF}_\Gamma(B)$$

a et on note $\preccurlyeq_\Gamma^*$ sa clôture transitive.

On a alors trivialement :

1.1.30 Lemme
La règle suivante est admissible :

$$\frac{\Gamma \vdash A : B}{\Gamma \vdash A : B'}\, B \preccurlyeq_\Gamma^* B'$$

Démonstration Si on a $B_1 \preccurlyeq_\Gamma \cdots \preccurlyeq_\Gamma B_n$ alors on a $B_1 \preccurlyeq \cdots \preccurlyeq B_n$ et $\mathrm{WF}_\Gamma(B_i)$ pour tout $1 < i \leq n$. On peut donc appliquer $n - 1$ fois le lemme 1.1.28 pour prouver $\Gamma \vdash A : B_n$ à partir de $\Gamma \vdash A : B_1$. ☺

1.1.31 Lemme
Si $\mathrm{WF}(\Gamma)$*, alors*

1. $(s, r) \in \mathcal{A}$ *implique* $\Gamma \vdash s : r$,
2. $x : A \in \Gamma$ *implique* $\Gamma \vdash x : A$.

Démonstration Par induction sur la taille de Γ :

1. Si $\Gamma = \langle\rangle$:
   (a) On prouve $\vdash s : r$ avec la règle Axiome.
   (b) Trivialité car $x : A \in \Gamma$ est impossible
2. Si $\Gamma = \Delta, x : B$, alors par définition de WF(Γ), on a WF(Δ) et $\Delta \vdash B : s$ pour un $s \in \mathcal{S}$.
   (a) On a $\Delta \vdash s : r$ par hypothèse d'induction, et on utilise la règle Affaiblissement pour prouver $\Gamma \vdash s : r$.
   (b) Deux cas sont à distinguer :
   – Soit $x = y$ et $A = B$, dans ce cas on utilise la règle Variable pour prouver que $\Gamma \vdash x : A$.
   – Soit $x \neq y$ et $x : A \in \Delta$ et par hypothèse de récurrence on a $\Delta \vdash x : A$. On utilise alors la règle Affaiblissement pour conclure $\Gamma \vdash x : A$. ☺

1.1.32 Lemme (Inversion)

1. $\Gamma \vdash s : A \Rightarrow \exists r \preccurlyeq^*_\Gamma A, (s, r) \in \mathcal{A}$,
2. $\Gamma \vdash x : A \Rightarrow \exists B \preccurlyeq^*_\Gamma A, (x : B) \in \Gamma$,
3. $\Gamma \vdash \lambda x : A.M : C \Rightarrow \exists (B, s), \Gamma \vdash \forall x : A.B : s \wedge \Gamma, x : A \vdash M : B \wedge \forall x : A.B \preccurlyeq^*_\Gamma C$,
4. $\Gamma \vdash \forall x : A.B : C \Rightarrow \exists (s_1, s_2, s_3) \in \mathcal{R}, \Gamma \vdash A : s_1 \wedge \Gamma, x : A \vdash B : s_2 \wedge s_3 \preccurlyeq^*_\Gamma C$,
5. $\Gamma \vdash MN : C \Rightarrow \exists (A, B), \Gamma \vdash M : \forall x : A.B \wedge \Gamma \vdash N : A \wedge B[N/x] \preccurlyeq^*_\Gamma C$.

Démonstration On procède dans chaque cas par induction sur la structure de la dérivation. Le système de types est globalement *dirigé par la syntaxe*, hormis les deux règles de cumulativité et celle d'affaiblissement qui ne "marquent" pas le terme dérivé. En conséquence, dans chaque cas, on lit ce qu'il faut démontrer dans les prémisses de la règle d'introduction associée (c'est-à-dire Axiome pour (1), Variable pour (2), Abstraction pour (3), Produit pour (4) et Application pour (5)). Et on utilise l'hypothèse d'induction et éventuellement la règle d'affaiblissement généralisé pour traiter les cas où la dernière règle utilisée est Affaiblissement, Cumulativité, ou Cumulativité-Sortes.

1.1.33 Lemme

On a $\Gamma \vdash x : A$ *si et seulement s'il existe* A' *tel que* $x : A' \in \Gamma$, $A' \preccurlyeq^*_\Gamma A$ *et* WF(Γ).

Démonstration Le sens direct est prouvé par le lemme d'inversion (lemme 1.1.32). Supposons que Γ contient $x : A'$ et que WF(Γ). D'après le lemme précédent, on a $\Gamma \vdash x : A'$. Et d'après le lemme 1.1.30, on a alors $\Gamma \vdash x : A$. ☺

1.1.34 Lemme

Si $\Gamma \vdash M : \forall x : A.B$, *alors il existe* $(s_1, s_2, s_3) \in \mathcal{R}$ *telles que* $\Gamma \vdash A : s_1$ *et* $\Gamma, x : A \vdash B : s_2$ *(et donc* $\Gamma \vdash \forall x : A.B : s_3$*).*

Démonstration On procède par induction sur la dérivation de $\Gamma \vdash M : \forall x : A.B$:
- Les règles VARIABLE, CUMULATIVITÉ, ABSTRACTION imposent de par l'une de leurs prémisses qu'il existe s telle que $\Gamma \vdash \forall x : A.B : s$ et on peut conclure avec le lemme d'inversion.
- Dans le cas AFFAIBLISSEMENT, Γ est de la forme $\Delta, y : C$ et on a $\Delta \vdash M : \forall x : A.B$ et $\Delta \vdash C : s$. D'après l'hypothèse d'induction, on en conclut qu'il existe $(s_1, s_2, s_3) \in \mathcal{R}$ telles que $\Delta \vdash A : s_1$ et $\Delta, x : A \vdash B : s_2$. Et le lemme d'affaiblissement (lemme 1.1.24), nous permet de conclure que $\Gamma \vdash A : s_1$ et $\Gamma, x : A \vdash B : s_2$.
- Dans le cas APPLICATION, M est de la forme $(M_1\, M_2)$ et il existe C, A' et B' tels que $\Gamma \vdash M_1 : \forall y : C.\forall x : A'.B'$, $\forall x : A.B = \forall x : A'[y/M_2].B'[y/M_2]$, $\Gamma \vdash M_2 : C$. Par hypothèse d'induction, on en déduit qu'il existe s tel que $\Gamma, y : C \vdash \forall x : A'.B' : s$, puis d'après le lemme d'inversion, on en déduit qu'il existe $(s_1, s_2, s_3) \in \mathcal{R}$ telles que $\Gamma, y : C \vdash A' : s_1$ et $\Gamma, y : C, x : A' \vdash B' : s_2$. Enfin le lemme 1.1.23, nous permet de conclure que $\Gamma \vdash A : s_1$ et $\Gamma, x : A \vdash B : s_2$.
- Les cas AXIOME, PRODUIT et CUMULATIVITÉ-SORTES sont impossibles. ☺

Le lemme suivant affirme que les types habitables sont bien formés :

1.1.35 Lemme
Si $\Gamma \vdash M : A$, *alors* $\mathrm{WF}_\Gamma(A)$.

Démonstration On procède par induction sur la dérivation de $\Gamma \vdash M : A$.
- Les cas des règles AXIOME, PRODUIT et CUMULATIVITÉ-SORTES ne posent aucun problème car A est dans ce cas une sorte.
- Les cas ABSTRACTION, VARIABLE, CUMULATIVITÉ sont également directs car une des prémisses impose que A soit typable par une sorte.
- Le cas AFFAIBLISSEMENT consiste simplement à appliquer l'hypothèse d'induction.
- On traite le cas APPLICATION. Dans ce cas, M est de la forme $(M_1\, M_2)$ et A de la forme $A_2[M_2/x]$, et il existe A_1 tel que $\Gamma \vdash M_1 : \forall x : A_1.A_2$ et $\Gamma \vdash M_2 : A_1$. Ainsi d'après le lemme 1.1.34, il existe bien une sorte s telle que $\Gamma, x : A_1 \vdash A_2 : s$ et d'après le lemme de substitution (lemme 1.1.23), on a $\Gamma \vdash A_2[M_2/x] : s$. ☺

1.1.36 Lemme
Si $\models$ WEAKLY-NORMALISING *(resp.* NORMALISING*), alors* $\mathrm{WF}_\Gamma(A)$ *implique* A *est faiblement normalisant (resp. fortement normalisant). Et ainsi* $\Gamma \vdash M : A$ *implique* A *est faiblement normalisant (resp. fortement normalisant).*

Démonstration Si $\mathrm{WF}_\Gamma(A)$, alors par définition :
- Soit A est une sorte s, et donc A est en forme normale.
- Soit il existe une sorte s telle que $\Gamma \vdash A : s$, et donc A est faiblement normalisant (resp. fortement normalisant).

Si $\Gamma \vdash M : A$, alors d'après le lemme 1.1.35, on a $\mathrm{WF}_\Gamma(A)$ et on peut conclure. ☺

1.1.37 Lemme (Cumulativité du contexte)
La règle suivante est admissible :

$$\frac{\Gamma, x : A', \Delta \vdash B : C \qquad \Gamma \vdash A : s}{\Gamma, x : A, \Delta \vdash B : C}\, A \preccurlyeq A'$$

Démonstration On procède par induction sur la dérivation de $\Gamma, x : A', \Delta \vdash B : C$. On ne traite que le cas variable (les autres consistent simplement à propager l'hypothèse d'induction). Dans le cas variable, on distingue les deux cas possibles :

- Soit $\Delta = \langle\rangle$,$B = x$ et $C = A'$ avec $\Gamma \vdash A' : r$ pour un certain r. On peut dériver $\Gamma, x : A \vdash x : A'$ par :

$$\frac{\dfrac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \qquad \dfrac{\Gamma \vdash A' : r \qquad \Gamma \vdash A : s}{\Gamma, x : A \vdash A' : r}\, A \preccurlyeq A'}{\Gamma, x : A \vdash x : A'}$$

- Soit Δ est de la forme $\Delta', y : D$ avec $y \notin \Gamma, x : A, \Delta'$, $\Gamma, x : A, \Delta' \vdash B : C$ et $\Gamma, x : A, \Delta' \vdash D : r$, alors par hypothèse d'induction, on peut prouver $\Gamma, x : A', \Delta' \vdash B : C$ et appliquant la règle Affaiblissement on conclut que $\Gamma, x : A', \Delta \vdash B : C$. ☺

1.1.38 Définition
On dit d'un WCTS $\mathcal{P}$ qu'il est un sous-WCTS de $\mathcal{P}'$ (on note $\mathcal{P} \subseteq \mathcal{P}'$), lorsque $\mathcal{S}_\mathcal{P} \subseteq \mathcal{S}_{\mathcal{P}'}$, $\mathcal{A}_\mathcal{P} \subseteq \mathcal{A}_{\mathcal{P}'}$, $\mathcal{R}_\mathcal{P} \subseteq \mathcal{R}_{\mathcal{P}'}$ et $\mathcal{C}_\mathcal{P} \subseteq \mathcal{C}_{\mathcal{P}'}$.

1.1.39 Lemme (Compacité)
Soit $\Gamma \vdash_\mathcal{P} A : B$, alors il existe un sous-WCTS fini $\mathcal{P}'$ de $\mathcal{P}$ tel que $\Gamma \vdash_{\mathcal{P}'} A : B$.

Démonstration Cela se prouve facilement par induction sur la dérivation de typage de $\Gamma \vdash_\mathcal{P} A : B$. L'idée étant de "collecter" les sortes qui apparaissent dans la dérivation (qui est un objet fini). ☺

1.1.7 Préservation du typage

Dans cette sous-section, nous expliquerons pourquoi la préservation du typage semble difficile à prouver dans le cas général des WCTS, bien que la preuve standard s'adapte très bien au cas des CTS.

1.1.40 Définition (Préservation du typage par réduction)
Soit $\mathcal{P}$ un WCTS, on dit que $\mathcal{P}$ satisfait la préservation du typage par réduction s'il vérifie la propriété :

$$\text{Si } \Gamma \vdash_\mathcal{P} A : B \text{ et } A \trianglerighteq A', \text{ alors } \Gamma \vdash_\mathcal{P} A' : B.$$

On notera $\mathcal{P} \models \mathrm{SR}_\beta$ pour signifier que $\mathcal{P}$ préserve les types, ou simplement $\models \mathrm{SR}_\beta$ s'il n'y a pas d'ambiguïté sur le WCTS concerné.

Le lemme suivant, vrai dans tous les WCTS, constituera un point essentiel pour prouver la préservation du typage dans les CTS :

1.1.41 Lemme (Substitutivité non-typée de la cumulativité des produits)
Pour tout terme N,

$$\forall x : A.B \preccurlyeq \forall x : A'.B' \qquad \Rightarrow \qquad A \equiv A' \wedge B[N/x] \preccurlyeq B'[N/x]$$

Démonstration Supposons $\forall x : A.B \preccurlyeq \forall x : A'.B'$. Par définition de $\preccurlyeq$ et de $\prec$, on distingue trois cas :

– Soit $\forall x : A.B \equiv \forall x : A'.B'$ et dans ce cas d'après le lemme 1.1.4, on a $A \equiv A'$, $B \equiv B'$ et donc $B[N/x] \equiv B'[N/x]$.
– Soit $\forall x : A.B \prec_0 \forall x : A'.B'$. Ce cas est impossible car cela signifierait que les deux produits sont convertibles en des sortes ce qui contredit la confluence.
– Soit $\forall x : A.B \prec_{n+1} \forall x : A'.B'$. Par définition, on sait qu'il existe A_0, B_0 et B'_0 tels que $\forall x : A.B \equiv \forall x : A_0.B_0$, $\forall x : A'.B' \equiv \forall x : A_0.B'_0$, et $B_0 \prec_n B'_0$. Et donc d'après le lemme 1.1.4, on a bien $A \equiv A_0 \equiv A'$ et d'après lemme 1.1.6, on a $B_0[N/x] \prec B'_0[N/x]$ et donc par compatibilité avec la conversion (lemme 1.1.7), on a $B[N/x] \prec B'[N/x]$. ☺

On en déduit donc :

1.1.42 Lemme (Préservation du typage des CTS)
Les CTS préservent les types :

$$\models \text{CTS} \qquad \Rightarrow \qquad \models \text{SR}_\beta$$

Démonstration La preuve a été formalisée en coq par Bruno Barras et Benjamin Grégoire [BG05], de plus on prouvera dans la suite un lemme plus général (lemme 1.1.45). On ne traitera donc ici uniquement le cas clef. Supposons que la dernière règle appliquée dans la dérivation soit :

$$\frac{\begin{array}{cc}(1) & (2)\\ \Gamma \vdash \lambda x : A.M : \forall x : A'.B' & \Gamma \vdash N : A'\end{array}}{\begin{array}{c}\Gamma \vdash (\lambda x : A.M)\, N : B'[N/x]\\ (0)\end{array}}$$

On cherche à montrer $\Gamma \vdash M[N/x] : B'[N/x]$ (*). Si l'on applique le lemme d'inversion (lemme 1.1.32) sur (1), on obtient qu'il existe B et s tels que $\Gamma, x : A \vdash M : B$ (2), $\Gamma \vdash \forall x : A.B : s$ (3) et $\forall x : A.B \preccurlyeq^*_\Gamma \forall x : A'.B'$ (4). Or comme on se trouve dans un CTS on a $\preccurlyeq^*=\preccurlyeq$ et ainsi on déduit de (4) que $\forall x : A.B \preccurlyeq \forall x : A'.B'$ (4′) et que $\text{WF}_\Gamma(\forall x : A'.B')$ (4″). D'après le lemme 1.1.41 on déduit de (4′) que $A \equiv A'$ et que $B[N/x] \preccurlyeq B'[N/x]$. Or d'après le lemme d'inversion 1.1.32, (3) implique qu'il existe s_A telle que $\Gamma \vdash A : s_A$ (5) et la règle de conversion (lemme 1.1.8) on en déduit $\Gamma \vdash N : A$ (6). Puis de (6), (2) et grâce au lemme de substitution (lemme 1.1.23), on a $\Gamma \vdash M[N/x] : B[N/x]$. Enfin, le lemme 1.1.35 et (0) nous permettent de déduire que $\text{WF}_\Gamma(B'[N/x])$ et lemme 1.1.28, nous permet de conclure (*). ☺

Comme nous pouvons le deviner dans la démonstration, si on veut étendre la préservation du typage au WCTS, il semble nécessaire de prouver la substitutivité de la cumulativité des produits :

1.1.43 Définition (Substitutivitée de la cumulativité des produits)
On dit d'un WCTS qu'il a la propriété de substitutivité de la cumulativité des produits si

$$\forall x : A.B \preccurlyeq^*_\Gamma \forall x : A'.B' \wedge \Gamma \vdash N : A' \qquad \Rightarrow \qquad A \equiv A' \wedge B[N/x] \preccurlyeq^*_\Gamma B'[N/x]$$

On notera $\mathcal{P} \models \text{SCP}$ pour signifier qu'un WCTS à cette propriété ou tout simplement $\models \text{SCP}$ s'il n'y a pas d'ambiguïté sur le WCTS concerné.

En réalité, cette propriété est nécessaire pour prouver la préservation du typage. En effet, les WCTS qui préservent les types vérifient la propriété de substitutivité de la cumulativité des produits :

1.1.44 Lemme

$$\models \mathrm{SR}_\beta \quad \Rightarrow \quad \models \mathrm{SCP}$$

Démonstration On procède par induction sur la longueur de la chaîne qui mène de $\forall x : A.B$ à $\forall x : A'.B'$:

- Si $\forall x : A.B = \forall x : A'.B'$ (avec $\mathrm{WF}_\Gamma(\forall x : A'.B')$), on a $A \equiv A'$ et $B[N/x] \preccurlyeq^* B'[N/x]$. Par inversion (lemme 1.1.32), on a $\mathrm{WF}_{\Gamma,x:A'}(B')$ et par substitution (lemme 1.1.23), $\mathrm{WF}_\Gamma(B'[N/x])$. Et donc on a bien $B[N/x] \preccurlyeq^*_\Gamma B'[N/x]$.
- Supposons $\forall x : A.B = P_1 \preccurlyeq_\Gamma \cdots \preccurlyeq_\Gamma P_n \preccurlyeq_\Gamma P_{n+1} = \forall x : A'.B'$ D'après le lemme 1.2.2, il existe A_0, B_0 tel que $P_n \trianglerighteq \forall x : A_0.B_0$, $A' \trianglerighteq A_0$, et $B_0 \preccurlyeq B'$. Par préservation du typage et comme $\mathrm{WF}_\Gamma(P_{n+1})$ on a $\mathrm{WF}_\Gamma(\forall x : A_0.B_0)$ et donc $\forall x : A.B \preccurlyeq_\Gamma P_2 \preccurlyeq_\Gamma \cdots \preccurlyeq_\Gamma P_{n-1} \preccurlyeq_\Gamma \forall x : A_0.B_0$ et par hypothèse de récurrence, on en déduit que $A \equiv A_0$ et $B[N/x] \preccurlyeq^*_{\Gamma,x:A} B_0[N/x]$. D'après le lemme d'inversion (lemme 1.1.32), $\mathrm{WF}_\Gamma(P_{n+1})$ implique $\mathrm{WF}_{\Gamma,x:A'}(B')$. D'après le lemme de substitution (lemme 1.1.23), on en déduit que $\mathrm{WF}_\Gamma(B'[N/x])$. Enfin, le lemme 1.1.6, permet de conclure que $B_0[N/x] \preccurlyeq B'[n/x]$ et donc que $B_0[N/x] \preccurlyeq_\Gamma B'[n/x]$. ☺

Mais elle est également suffisante. Un WCTS préserve les types si et seulement s'il vérifie la propriété de substitutivité de la cumulativité des produits :

1.1.45 Lemme (Préservation du typage dans certains WCTS)

$$\models \mathrm{SCP} \quad \Rightarrow \quad \models \mathrm{SR}_\beta$$

Démonstration On montre SR_β en prouvant l'implication plus forte suivante :

$$\Gamma \vdash M : A \quad \Rightarrow \quad (M \twoheadrightarrow M' \Rightarrow \Gamma \vdash M' : A \quad (1)) \quad \wedge \quad (\Gamma \twoheadrightarrow \Gamma' \Rightarrow \Gamma' \vdash M : A \quad (2))$$

par induction sur la dérivation $\Gamma \vdash M : A$.

- Axiome : On est dans le cas où $\Gamma = \langle\rangle$, $M = s$ et $A = r$ avec $(s, r) \in \mathcal{A}$.
  1. Ce cas est trivial car $M = s$ ne peut pas contenir de redex.
  2. Ce cas est également trivial car $\Gamma = \langle\rangle$ et on ne peut donc pas avoir $\Gamma \twoheadrightarrow \Gamma'$.
- Variable : On est dans le cas où $\Gamma = \Delta, x : A$, $M = x$ et où $\Delta \vdash A : s$ pour une sorte s.
  1. Ce cas est trivial car $M = x$ ne peut pas contenir de redex.
  2. Si on a $\Gamma \twoheadrightarrow \Gamma'$ alors deux cas sont à distinguer :
     - Soit $\Gamma' = \Delta', x : A$ et $\Delta \twoheadrightarrow \Delta'$. Par hypothèse d'induction, on déduit $\Delta' \vdash A : s$ et donc d'après la règle Variable, $\Delta', x : A \vdash x : A$.
     - Soit $\Gamma' = \Delta, x : A'$ et $A \twoheadrightarrow A'$. Par hypothèse d'induction, on déduit $\Delta \vdash A' : s$ et donc $\Delta, x : A' \vdash x : A'$, puis grâce à la dérivation suivante :

$$\dfrac{\dfrac{\dfrac{\Delta \vdash A' : s}{\Gamma' \vdash x : A'}\,\textsc{Variable} \qquad \dfrac{\Delta \vdash A : s \qquad \Delta \vdash A : s}{\Gamma' \vdash A : s}\,\textsc{Affaiblissement}}{\Gamma' \vdash x : A}\,\textsc{Conversion}$$

- Affaiblissement : On est dans le cas où $\Gamma = \Delta, x : B$, $x \notin \Delta$, $\Delta \vdash M : A$ et $\Delta \vdash B : s$.

1. Si $M \twoheadrightarrow M'$, alors par hypothèse d'induction, $\Delta \vdash M' : A$ et d'après la règle Affaiblissement, $\Gamma \vdash M' : A$.
2. Si $\Gamma \twoheadrightarrow \Gamma'$, on distingue deux cas :
   - Soit $\Gamma' = \Delta', x : B$ et $\Delta \twoheadrightarrow \Delta'$. Par hypothèse d'induction, on déduit $\Delta' \vdash M : A$, $\Delta' \vdash B : s$ et donc d'après la règle Affaiblissement, $\Delta', x : B \vdash M : A$.
   - Soit $\Gamma' = \Delta, x : B'$ et $B \twoheadrightarrow B'$. Par hypothèse d'induction, on déduit $\Delta \vdash B' : s$ et donc $\Delta, x : B' \vdash M : A$ d'après la règle Affaiblissement.

– Abstraction : Ici, M est de la forme $\lambda x : A_1.N$ et A de la forme $\forall x : A_1.A_2$ avec $\Gamma, x : A_1 \vdash N : A_2$ et $\Gamma \vdash \forall x : A_1.A_2 : s$.

1. Si $M \twoheadrightarrow M'$, on distingue deux cas :
   - Soit M' est de la forme $\lambda x : A_1.N'$ avec $N \twoheadrightarrow N'$, auquel on a par hypothèse d'induction que $\Gamma, x : A_1 \vdash N' : A_2$ et on conclut grâce à la règle Abstraction que $\Gamma \vdash \lambda x : A_1.N' : \forall x : A_1.A_2$.
   - Soit M' est de la forme $\lambda x : A_1'.N$ avec $A_1 \twoheadrightarrow A_1'$, alors par hypothèse de récurrence on a $\Gamma \vdash \forall x : A_1'.A_2 : s$ (car $\forall x : A_1.A_2 \twoheadrightarrow \forall x : A_1'.A_2$) et $\Gamma, x : A_1' \vdash N : A_2$ et ainsi la règle Abstraction nous prouve que $\Gamma \vdash M' : A$.
2. Si $\Gamma \twoheadrightarrow \Gamma'$, alors par hypothèse d'induction on a : $\Gamma', x : A_1 \vdash N : A_2$ et $\Gamma' \vdash \forall x : A_1.A_2 : s$ et la règle Abstraction, nous donne $\Gamma' \vdash \lambda x : A_1.N : \forall x : A_1.A_2$.

– Application : Dans ce cas, M est de la forme $M_1\, M_2$ avec $\Gamma \vdash M_1 : \forall x : B'.C'$ (1) et $\Gamma \vdash M_2 : B'$ (2) et A est de la forme $C'[M_2/x]$.

1. Si $M \twoheadrightarrow M'$, alors on distingue trois cas :
   - M' est de la forme $(M_1'\, M_2)$ avec $M_1 \twoheadrightarrow M_1'$, alors par hypothèse d'induction on a $\Gamma \vdash M_1' : \forall x : B'.C'$ et donc d'après la règle Application, on conclut que $\Gamma \vdash M_1'\, M_2 : C'[M_2/x]$.
   - M' est de la forme $(M_1\, M_2')$ avec $M_2 \twoheadrightarrow M_2'$, alors par hypothèse d'induction on a $\Gamma \vdash M_2' : B'$. Or d'après le lemme 1.1.34, on déduit de (1) qu'il existe une sorte s telle que $\Gamma, x : B' \vdash C' : s$ et d'après le lemme de substitution (lemme 1.1.23) et (2), on en déduit que $\Gamma \vdash C'[M_2/x] : s$. Enfin, on a $C'[M_2/x] \equiv C'[M_2'/x]$ on peut donc finir avec la dérivation suivante :
   $$\dfrac{\dfrac{\Gamma \vdash M_1 : \forall x : B'.C' \qquad \Gamma \vdash M_2' : B'}{\Gamma \vdash M_1\, M_2' : C'[M_2'/x]} \qquad \Gamma \vdash C'[M_2/x] : s}{\Gamma \vdash M_1\, M_2' : C'[M_2/x]}\ \text{Conversion}$$
   - M est de la forme $\lambda x : B.N$ et M' est de la forme $N[M_2/x]$. On est ici dans le cas clef de la démonstration (c'est le seul que l'on avait traité pour la preuve du lemme 1.1.42). Reprenons donc cette démonstration. On cherche à montrer que $\Gamma \vdash N[M_2/x] : C'[M_2/x]$ (*). Si on applique le lemme d'inversion (lemme 1.1.32) sur (1), on obtient qu'il existe C tel que $\Gamma, x : B \vdash N : C$ (2), $\mathrm{WF}_\Gamma(\forall x : B.C)$ (3) et $\forall x : B.C \preccurlyeq^*_\Gamma \forall x : B'.C'$ (4). Comme on se trouve dans WCTS qui a la propriété de substitutivité de la cumulativité des produits, on déduit de (4) et de (2) que $B \equiv B'$ et que $C[M_2/x] \preccurlyeq^*_{\Gamma, x:B'} C'[M_2/x]$. Par inversion de (2), on a $\mathrm{WF}_\Gamma(B)$ et donc $\Gamma \vdash M_2 : B$ (5). Enfin, le lemme de substitution (lemme 1.1.23) avec (2) et (5), nous donne $\Gamma \vdash N[M_2/x] : C[M_2/x]$ et le lemme 1.1.28 nous permet de conclure (*).

2. Si $\Gamma \twoheadrightarrow \Gamma'$, alors par hypothèse d'induction on a $\Gamma' \vdash M_1 : \forall x : B'.C'$ et $\Gamma' \vdash M_2 : B'$ et la règle Application permet de conclure que $\Gamma' \vdash M_1\, M_2 : C'[M_2/x]$.

– Produit : Ici, M est de la forme $\forall x : A_1.A_2$ et A est une sorte s_3 et on a $\Gamma \vdash A_1 : s_2$ et $\Gamma, x : A_1 \vdash A_2 : s_2$ pour $(s_1, s_2, s_3) \in \mathcal{R}$.

1. Si $M \twoheadrightarrow M'$, on distingue deux cas :
   – Soit M' est de la forme $\forall x : A_1'.A_2$ avec $A_1 \twoheadrightarrow A_1'$, et par hypothèse d'induction on a $\Gamma \vdash A_1' : s_2$ et $\Gamma, x : A_1' \vdash A_2 : s_2$ et donc d'après la règle Produit on en conclut $\Gamma \vdash \forall x : A_1'.A_2 : s_3$.
   – Soit M' est de la forme $\forall x : A_1.A_2'$ avec $A_2 \twoheadrightarrow A_2'$, et par hypothèse d'induction on a $\Gamma, x : A_1 \vdash A_2' : s_2$ et donc d'après la règle Produit on en conclut $\Gamma \vdash \forall x : A_1.A_2' : s_3$.
2. Si $\Gamma \twoheadrightarrow \Gamma'$, alors par hypothèse d'induction on a $\Gamma' \vdash A_1 : s_2$ et $\Gamma', x : A_1 \vdash A_2 : s_2$ et donc d'après la règle Produit, on conclut $\Gamma' \vdash \forall x : A_1.A_2 : s_3$.

– Cumulativité : Dans ce cas, on a $\Gamma \vdash M : B$ et $\Gamma \vdash A : s$ pour un certain B tel que $B \preccurlyeq A$.

1. Si $M \twoheadrightarrow M'$, alors par hypothèse d'induction, on a $\Gamma \vdash M' : B$ et d'après la règle Cumulativité, on a $\Gamma \vdash M' : A$.
2. De même si $\Gamma \twoheadrightarrow \Gamma'$, alors par hypothèse d'induction, on a $\Gamma' \vdash M : B$ et d'après la règle Cumulativité, on a $\Gamma' \vdash M : A$.

– Cumulativité-Sortes : Dans ce cas, on a $\Gamma \vdash M : C$ et $A = s$ avec $C \preccurlyeq A$.

1. Si $M \twoheadrightarrow M'$, alors par hypothèse d'induction, on a $\Gamma \vdash M' : C$ et d'après la règle Cumulativité-Sortes, on a $\Gamma \vdash M' : s$.
2. De même si $\Gamma \twoheadrightarrow \Gamma'$, alors par hypothèse d'induction, on a $\Gamma' \vdash M : C$ et d'après la règle Cumulativité-Sortes, on a $\Gamma' \vdash M : s$. ☺

Ainsi, il apparaît que la preuve habituelle pour démontrer la préservation du typage ne fonctionne pas dans le cas le plus général puisqu'elle nécessite de prouver un lemme qui lui est équivalent. Et il ne semble pas y avoir de décroissance évidente qui permettrait de prouver SR_β et SCP par une récurrence mutuelle. Le problème reste donc ouvert :

1.1.46 Question ouverte
A-t-on pour tout WCTS $\mathcal{P}$, $\mathcal{P} \models \mathrm{SR}_\beta$?

1.1.47 Lemme (Réduction du type)
Si $\models \mathrm{SR}_\beta$, alors la règle suivante est admissible :

$$\frac{\Gamma \vdash A : B}{\Gamma \vdash A : B'}\, B \rhd B'$$

Démonstration Si $\Gamma \vdash A : B$, alors B ne peut pas être une sorte (car B ne contiendrait pas de redex) et donc d'après le lemme 1.1.35 il existe s telle que $\Gamma \vdash B : s$. Par préservation du typage, on en déduit que $\Gamma \vdash B' : s$ et la règle de conversion nous permet de conclure que $\Gamma \vdash A : B'$. ☺

1.1.8 Théorèmes de renforcement

Le renforcement est la règle duale de l'affaiblissement généralisé (lemme 1.1.24), il permet d'enlever les variables du contexte qui n'apparaissent pas dans le terme à typer ni dans les types des autres variables.

1.1.48 Définition (Renforcement)
Soit $\mathcal{P}$ un WCTS, on dit que $\mathcal{P}$ admet la propriété de renforcement s'il vérifie :

$$\text{si } \Gamma, x : A, \Delta \vdash_{\mathcal{P}} M : B \text{ et que } x \notin \mathcal{FV}(\Delta), \mathcal{FV}(M), \mathcal{FV}(B), \text{ alors } \Gamma, \Delta \vdash_{\mathcal{P}} M : B.$$

En d'autres termes, la règle suivante est admissible :

$$\frac{\Gamma, x : A, \Delta \vdash_{\mathcal{P}} M : B}{\Gamma, \Delta \vdash_{\mathcal{P}} M : B}\, x \notin \mathcal{FV}(\Delta), \mathcal{FV}(M), \mathcal{FV}(B)$$

On note $\mathcal{P} \models$ STRENGTHENING pour signifier que $\mathcal{P}$ admet la propriété de renforcement ou simplement $\models$ STRENGTHENING s'il n'y a pas d'ambiguïté sur le WCTS concerné.

Le renforcement est utilisé fréquemment dans l'implémentation des assistants de preuves. Il implémente le déchargement d'hypothèses. Par exemple, dans le système Coq, lorsque l'on ferme une `Section`, le système utilise le renforcement pour éliminer les hypothèses non-utilisées :

```
Require Import Omega.
Section Renf.
  Variable x : nat.
  Variable y : nat.
  Lemma t1 : x + x = 2 * x.
   omega.
  Qed.
  Lemma t2 : y + x = y + x.
   omega.
  Qed.
End Renf.
Check t1. (* Affiche ``t1 : forall x : nat, x + x = 2 * x'' *)
```

De plus, nous verrons ensuite qu'il est essentiel afin de prouver le typage des constructions syntaxiques définies par induction sur la structure des termes (en particulier, lorsque celles-ci peuvent rajouter des variables dans le contexte).

Une des conséquences du renforcement est la possibilité de faire commuter deux déclarations du contexte lorsque la seconde ne dépend pas de la première :

1.1.49 Lemme (Commutation des variables du contexte)
Si $\mathcal{P} \models$ STRENGTHENING, alors la règle suivante est admissible :

$$\frac{\Gamma, x : A, y : B, \Delta \vdash M : T}{\Gamma, y : B, x : A, \Delta \vdash M : T}\, x \notin B$$

Démonstration On procède par induction sur la dérivation de $\Gamma, x : A, y : B, \Delta \vdash M : T$ tous les cas à l'exception de VARIABLE et AFFAIBLISSEMENT consiste à propager les hypothèses d'inductions (le cas AXIOME est impossible puisque le contexte est non-vide par hypothèse). On traite donc ces deux cas :

- VARIABLE : On distingue deux cas :
  - Dans le cas où $\Delta \neq \langle\rangle$, il suffit d'appliquer l'hypothèse d'induction puis d'appliquer la règle VARIABLE.
  - Dans le cas où $\Delta = \langle\rangle$, on a $M = y$ et $B = T$ avec $\Gamma, x : A \vdash B : s$. Or, d'après la propriété STRENGTHENING, on a $\Gamma \vdash B : s$. De plus, on a, d'après le lemme 1.1.35, $\mathrm{WF}(\Gamma, x : A)$ c'est-à-dire $\Gamma \vdash A : r$ pour une sorte r. On peut donc conclure à l'aide de la dérivation suivante :

$$\dfrac{\dfrac{\Gamma \vdash B : s}{\Gamma, y : B \vdash M : T} \qquad \dfrac{\Gamma \vdash A : r \qquad \Gamma \vdash B : s}{\Gamma, y : B \vdash A : r}}{\Gamma, y : B, x : A \vdash M : T}$$

- AFFAIBLISSEMENT : On distingue deux cas :
  - Dans le cas où $\Delta \neq \langle\rangle$, il suffit d'appliquer l'hypothèse d'induction puis d'appliquer la règle AFFAIBLISSEMENT.
  - Dans le cas où $\Delta = \langle\rangle$ avec $\Gamma, x : A \vdash M : T$ et $\Gamma, x : A \vdash B : s$. On en déduit que $y \notin \mathcal{FV}(M) \cup \mathcal{FV}(T)$. Or d'après STRENGTHENING, on a $\Gamma \vdash B : s$. Et donc d'après le lemme 1.1.24, on en déduit que $\Gamma, y : B, x : A \vdash M : T$. ☺

Il peut être assez surprenant à première vue d'apprendre que la propriété de renforcement est difficile à prouver. La première preuve de la propriété de renforcement dans un système de types dépendants est due à van Daalen [vD80], puis est venue celle de Luo pour ECC, un système de types dépendants avec cumulativité [Luo90].

1.1.50 Théorème (Renforcement de ECC, Luo, 1990)

$$\text{ECC} \models \text{STRENGTHENING}$$

Dans ECC, le produit dépendant entre deux types bien-formés est toujours autorisé et c'est pourquoi l'équivalent de la règle ABSTRACTION dans ECC n'a qu'une seule prémisse : il n'y a pas besoin de vérifier que le produit est autorisé lorsque l'on cherche à typer une abstraction. C'est cette simplification qui permet à Luo d'écrire une preuve naturelle et simple de la propriété de renforcement. Cette preuve a ensuite été adaptée sans difficulté pour prouver le renforcement des PTS fonctionnels par Geuvers et Nederhof [GN91].

1.1.51 Théorème (Renforcement dans les PTS fonctionnels, Geuvers, 1991)

$$\models \text{FUNCTIONAL} \quad \Rightarrow \quad \models \text{STRENGTHENING}$$

Mais c'est van Benthem Jutting [vBJ93] qui a réussi le tour de force de prouver le renforcement pour la classe de tous les PTS. Pour cela, il est lui aussi parti de la preuve de Luo et il a su développer des outils non-triviaux pour adapter le résultat de Luo. La preuve obtenue est étonnamment technique.

1.1.52 Théorème (Renforcement dans les PTS, van Benthem Jutting, 1993)

$$\models \text{PTS} \quad \Rightarrow \quad \models \text{STRENGTHENING}$$

Enfin, Jiménez [Jim99] a adapté la preuve de van Benthem Jutting pour prouver que les CTS, dont la relation de cumulativité préserve le typabilité vers le haut et vers le bas, vérifient la propriété de renforcement. Hélas, la présentation des CTS de Jiménez ne dispose pas de la règle Cumulativité-Sortes : dans son système, on ne peut donc pas utiliser la cumulativité entre sortes maximales. Ainsi pour importer son résultat, nous nous restreindrons aux CTS qui ne comparent par la cumulativité que des sortes qui ne sont pas maximales :

$$\forall (t,s) \in \mathcal{C}, \exists r, (s,r) \in \mathcal{A}$$

On notera cette propriété NICE-TOPSORT.

1.1.53 Lemme
Si $\models$ NICE-TOPSORT *et si* $\Gamma \vdash M : A$ *est dérivable, alors* $\Gamma \vdash M : A$ *est également dérivable sans utiliser la règle* Cumulativité-Sortes.

Démonstration Par induction sur la dérivation $\Gamma \vdash M : A$, on ne traite que le cas de la règle Cumulativité-Sortes, les autres cas consistant simplement à propager les hypothèses d'induction.

- Cumulativité-Sortes : On dispose par hypothèses d'induction d'une dérivation de $\Gamma \vdash M : C$ qui n'utilise pas la règle Cumulativité-Sortes, on sait que $C \preccurlyeq s$ et on veut montrer que $\Gamma \vdash M : s$ sans utiliser la règle Cumulativité-Sortes. Si $C \preccurlyeq s$, par définition cela signifie que $C \equiv t$ avec $(t,s) \in \mathcal{C}$. On sait alors d'après $\models$ NICE-TOPSORT, qu'il existe r telle que $(s,r) \in \mathcal{A}$. On a donc bien $\Gamma \vdash s : r$. On peut alors conclure à l'aide de Cumulativité que $\Gamma \vdash M : s$. ☺

La préservation de la typabilité par la cumulativité est définie par :

1.1.54 Définition (Préservation de la typabilité par la cumulativité)
On dit que la relation de cumulativité d'un WCTS préserve la typabilité vers le haut (resp. vers le bas), si $\Gamma \vdash A : s$ et $A \preccurlyeq A'$ (resp. $A' \preccurlyeq A$), alors il existe A'' tel que $A' \trianglerighteq A''$ et $\text{WF}_\Gamma(A'')$. On notera ces deux propriétés respectivement WEAK-SR$_{\preccurlyeq}$ et WEAK-SR$_{\succcurlyeq}$.

On notera que la préservation de la typabilité est toujours vérifiée dans les PTS :

1.1.55 Lemme

$$\models \text{PTS} \quad \Rightarrow \quad \models \text{WEAK-SR}_{\preccurlyeq} \wedge \text{WEAK-SR}_{\succcurlyeq}$$

Démonstration Il faut montrer que si $\Gamma \vdash A : s$ et que $B \equiv A$, alors il existe C tel que $B \trianglerighteq C$ et $\text{WF}_\Gamma(C)$. Or on montre facilement que $\Gamma \vdash C : s$ par confluence et par préservation du typage. ☺

Et le théorème de Jiménez peut alors s'énoncer de la manière suivante :

1.1.56 Théorème (Renforcement pour une classe de CTS, Jiménez, 1999)
Si $\models$ NICE-TOPSORT, *alors*

$$\models \text{CTS} \wedge \text{WEAK-SR}_{\preccurlyeq} \wedge \text{WEAK-SR}_{\succcurlyeq} \quad \Rightarrow \quad \models \text{STRENGTHENING}$$

On ne confondra pas WEAK-SR$_{\preccurlyeq}$ et WEAK-SR$_{\succcurlyeq}$ avec les propriétés plus fortes suivantes :

1.1.57 Définition (Préservation du typage par la cumulativité)
On dit que la relation de cumulativité d'un WCTS préserve le typage vers le haut (resp. vers le bas), si $\Gamma \vdash A : s$ et $A \preccurlyeq A'$ (resp. $A' \preccurlyeq A$), alors il existe A'' et s' tels que $s \preccurlyeq^* s'$ (resp. $s' \preccurlyeq^* s$), $A' \trianglerighteq A''$ et $\Gamma \vdash A'' : s'$. On notera ces deux propriétés respectivement SR$_{\preccurlyeq}$ et SR$_{\succcurlyeq}$.

La préservation du typage par la cumulativité est également vérifiée dans les PTS :

1.1.58 Lemme

$$\models \text{PTS} \quad \Rightarrow \quad \models \text{SR}_{\preccurlyeq} \wedge \text{SR}_{\succcurlyeq}$$

Démonstration Il faut montrer que si $\Gamma \vdash A : s$ et que $B \equiv A$, alors il existe C et r tel que $B \trianglerighteq C$, $s \equiv r$ et $\Gamma \vdash C : r$. Il suffit de prendre $r = s$ et on montre facilement que $\Gamma \vdash C : s$ par confluence et par préservation du typage. ☺

En général, les conditions WEAK-SR$_{\preccurlyeq}$ et WEAK-SR$_{\succcurlyeq}$ du théorème ne sont pas toujours triviales à vérifier, on donnera dans la section 1.2 (lemme 1.2.36) des conditions suffisantes sur les paramètres et faciles à vérifier (elles sont même décidables dans le cas fini) pour prouver WEAK-SR$_{\preccurlyeq}$ et WEAK-SR$_{\succcurlyeq}$.

On démontrera également le théorème de renforcement (théorème 1.2.42) suivant qui sera au théorème 1.1.56 de Jiménez ce que le théorème 1.1.51 de Geuvers est au théorème 1.1.52 de van Benthem Jutting.

Théorème (Renforcement de certains WCTS)
Si $\models$ SR$_\beta$ *et si*

$$\Gamma \vdash A_1 : s_1 \wedge \Gamma, x : A_1 \vdash A_2 : s_2 \wedge \Gamma \vdash \forall x : A_1 . A_2 : s \quad \Rightarrow \quad \exists s_3, (s_1, s_2, s_3) \in \overline{\mathcal{R}}$$

alors on a $\models$ STRENGTHENING.

La notation $\overline{\mathcal{R}}$ désigne la clôture des règles par la relation de cumulativité (voir définition 1.2.14). La preuve est plus simple que celle de van Benthem Jutting (et *a fortiori* que celle de Jiménez) et elle capturera tous les WCTS qui nous intéresseront. Néanmoins, elle ne s'appliquera pas pas aux PTS non-fonctionnels. Nous donnerons également un critère sur les paramètres (lemme 1.2.44) pour vérifier la condition du théorème qui coïncideront à peu de choses près avec le critère utilisé pour prouver WEAK-SR$_{\preccurlyeq}$ et WEAK-SR$_{\succcurlyeq}$.

Dans le cas général, il semble que la question reste ouverte :

1.1.59 Question ouverte
Est-ce que tous les WCTS ont la propriété du renforcement ?

1.1.9 Morphismes

Les morphismes permettent de plonger la syntaxe d'un WCTS dans un autre WCTS moralement plus complet. Ils permettront de donner une interprétation sémantique aux systèmes qui se plongent dans les systèmes disposant de bonnes notions de modèle. C'est donc un outil de choix pour prouver des résultats de cohérence relative.

1.1.60 Définition (Morphisme)
Soient $\mathcal{P}$ et $\mathcal{P}'$ deux WCTS. On dit de $\varphi : \mathcal{S} \longrightarrow \mathcal{S}'$ qu'elle est un morphisme de WCTS si

1. Pour tout $(s_1, s_2) \in \mathcal{A}_\mathcal{P}$, $(\varphi(s_1), \varphi(s_2)) \in \mathcal{A}_{\mathcal{P}'}$,
2. Pour tout $(s_1, s_2, s_3) \in \mathcal{R}_\mathcal{P}$, $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) \in \mathcal{R}_{\mathcal{P}'}$.
3. Pour tout $(s_1, s_2) \in \mathcal{S}_\mathcal{P}$, $s_1 \preccurlyeq_\mathcal{P} s_2 \Rightarrow \varphi(s_1) \preccurlyeq_{\mathcal{P}'} \varphi(s_2)$.

Et on peut étendre φ à l'ensemble des expressions et des contextes de $\mathcal{P}$ de la manière suivante :

$$\begin{array}{rcl} \varphi(x) & = & x \\ \varphi(M\,N) & = & \varphi(M)\,\varphi(N) \\ \varphi(\lambda x : A.B) & = & \lambda x : \varphi(A).\varphi(B) \\ \varphi(\forall x : A.B) & = & \forall x : \varphi(A).\varphi(B) \end{array}$$

$$\begin{array}{rcl} \varphi(\langle\rangle) = \langle\rangle & & \\ \varphi(\Gamma, x : A) & = & \varphi(\Gamma), x : \varphi(A) \end{array}$$

On notera $\mathcal{P} \hookrightarrow_\varphi \mathcal{P}'$ pour signifier que φ est un morphisme entre $\mathcal{P}$ et $\mathcal{P}'$, on notera $\mathcal{P} \hookrightarrow \mathcal{P}'$ pour qu'il existe un tel morphisme.

Comme le fait remarquer Geuvers dans sa thèse [Geu93], les PTS forment une catégorie avec un objet initial (le PTS vide), un objet terminal (voir lemme 1.1.69), un produit et un coproduit. On remarquera également que $\mathcal{P} \subseteq \mathcal{P}'$ si et seulement si l'identité est un morphisme de WCTS.

Les morphismes ne transforment pas la structure des termes, on prouve donc aisément le lemme suivant.

1.1.61 Lemme
Si $\mathcal{P} \hookrightarrow_\varphi \mathcal{P}'$ alors, $A \rhd A' \Leftrightarrow \varphi(A) \rhd \varphi(A')$, $A \trianglerighteq A' \Leftrightarrow \varphi(A) \trianglerighteq \varphi(A')$ et $A \equiv A' \Leftrightarrow \varphi(A) \equiv \varphi(A')$.

Les morphismes préservent la relation de cumulativité :

1.1.62 Lemme
Si $\mathcal{P} \hookrightarrow_\varphi \mathcal{P}'$ alors,

$$A \preccurlyeq_\mathcal{P} B \qquad \Rightarrow \qquad \varphi(A) \preccurlyeq_{\mathcal{P}'} \varphi(B)$$

Démonstration Si $A \equiv B$, alors d'après le lemme 1.1.61, on a $\varphi(A) \equiv \varphi(B)$. Il suffit donc de montrer que

$$A \prec_{\mathcal{P},n} B \qquad \Rightarrow \qquad \varphi(A) \preccurlyeq_{\mathcal{P}'} \varphi(B)$$

et de procéder par induction sur n.

- Si on a $A \prec_{\mathcal{P},0} B$, alors cela signifie qu'il existe s_A et s_B telles que $A \equiv s_A$, $B \equiv s_B$ et $(s_A, s_B) \in \mathcal{C}_\mathcal{P}$. Comme φ est un morphisme, on a $\varphi(s_A) \preccurlyeq_{\mathcal{P}'} \varphi(s_B)$ et comme $\varphi(A) \equiv \varphi(s_a)$ et $\varphi(B) \equiv \varphi(s_B)$ (lemme 1.1.61), la compatibilité (lemme 1.1.7) donne $\varphi(A) \preccurlyeq \varphi(B)$.
- Si on a $A \prec_{\mathcal{P},n+1} B$, alors par définition deux cas sont possibles :
  - Soit $A \prec_{\mathcal{P},n} B$ et l'hypothèse de récurrence nous donne bien que $\varphi(A) \preccurlyeq \varphi(B)$.

– Soit il existe A', B' et C tels que $A \equiv \forall x : C.A'$, $B \equiv \forall x : C.B'$, $A' \prec_{\mathcal{P},n} B'$. Or par hypothèse de récurrence, on a $\varphi(A') \preccurlyeq \varphi(B')$ et on en conclut que $\varphi(\forall x : C.A') = \forall x : \varphi(C).\varphi(A') \preccurlyeq \forall x : \varphi(C).\varphi(B') = \varphi(\forall x : C.B')$. ☺

Par une induction facile sur la dérivation de $\Gamma \vdash_{\mathcal{P}} A : B$, on déduit le lemme suivant :

1.1.63 Lemme
Si $\mathcal{P} \hookrightarrow_{\varphi} \mathcal{P}'$ *alors,*

$$\Gamma \vdash_{\mathcal{P}} A : B \Rightarrow \varphi(\Gamma) \vdash_{\mathcal{P}'} \varphi(A) : \varphi(B)$$

Par ailleurs, comme les morphismes préservent de façon très fidèle la réduction des termes, on déduit immédiatement à partir des lemmes 1.1.61 et 1.1.63 le résultat suivant :

1.1.64 Lemme
Si $\mathcal{P} \hookrightarrow_{\varphi} \mathcal{P}'$ *et que* $\mathcal{P}'$ *est faiblement normalisant (resp. fortement normalisant) alors* $\mathcal{P}$ *est faiblement normalisant (resp. fortement normalisant).*

Nous allons maintenant donner une définition de la cohérence qui s'adapte à tous les systèmes, y compris ceux qui ne disposent pas d'encodage de l'absurdité logique (on trouve cette définition par exemple dans [BS00]) :

1.1.65 Définition (Sorte cohérente, Système cohérent)
Une sorte s d'un système $\mathcal{P}$ est dite Γ-cohérente (ou simplement cohérente dans le cas où $\Gamma = \langle\rangle$) s'il n'existe pas de terme M tel que $\Gamma, \alpha : s \vdash_{\mathcal{P}} M : \alpha$. Et un système est Γ-cohérent si toutes ses sortes sont Γ-cohérentes.

On remarque les sortes maximales pour la relation $\mathcal{A}$ sont toujours cohérentes car $\alpha : s$ n'est alors pas un contexte bien formé.

Dans CC, on peut montrer que la cohérence est équivalente à l'impossibilité de dériver l'absurdité logique :

1.1.66 Lemme
CC *est* Γ*-cohérent si et seulement s'il n'existe pas de terme* M *tel que* $\Gamma \vdash M : \forall X : \star.X$.

Démonstration Clairement s'il existe un tel M, alors on a $\Gamma, \alpha : \star \vdash M\,\alpha : \alpha$ et donc CC n'est pas Γ-cohérent. Réciproquement, si CC n'est pas Γ-cohérent on sait, par définition, qu'il existe N tel que $\Gamma, \alpha : \star \vdash N : \alpha$ (on notera que $\star$ est la seule sorte qui n'est pas maximale pour $\mathcal{A}$ dans CC). Donc d'après le lemme de substitution, on obtient que $\Gamma \vdash N[\forall X : \star.X/\alpha] : \forall X : \star.X$ et il suffit de prendre $M = N[\forall X : \star.X/\alpha]$ pour conclure. ☺

On dispose du lien suivant avec la normalisation :

1.1.67 Lemme
Si $\mathcal{P} \models$ WEAKLY-NORMALISING $\wedge\, \mathrm{SR}_{\beta}$, *alors* $\mathcal{P}$ *est cohérent.*

Démonstration Procédons par l'absurde, supposons que $\mathcal{P}$ ne soit pas cohérent. Alors il existerait un terme M et une sorte s tel que $\alpha : s \vdash M : \alpha$. Par préservation du typage, on a $\alpha : s \vdash \mathrm{NF}(M) : \alpha$. On peut donc utiliser le lemme d'inversion 1.1.32 pour inspecter les formes possibles de $\mathrm{NF}(M)$. On sait que $\mathrm{NF}(M)$ ne peut pas commencer par une abstraction car α n'est pas convertible en un produit. De

même, il ne peut pas commencer par un produit ou être une sorte car α n'est pas convertible en une sorte. Enfin, si $\mathrm{NF}(M)$ est une variable ce ne peut-être que α car c'est la seule variable du contexte, or le lemme d'inversion montre bien qu'il est impossible d'avoir $\alpha : s \vdash \alpha : \alpha$ (car on a pas $s \preccurlyeq^*_\Gamma \alpha$). On en déduit donc que $\mathrm{NF}(M)$ est nécessairement de la forme $(M_0\, M_1 \cdots M_n)$ pour un certain $n > 0$ avec M_0 qui n'est pas une application. Maintenant par n inversions successives, nous en déduisons que $\alpha : s \vdash M_0 : \forall x : A.B$ pour deux termes A et B. De plus comme $\mathrm{NF}(M)$ est en forme normale, M_0 ne peut pas commencer par une abstraction. De plus, par inversion il ne peut pas non plus être une sorte, ou un produit. C'est donc qu'il doit être une variable, or la seule variable disponible dans le contexte est α et on ne peut pas avoir $\alpha : s \vdash \alpha : \forall x : A.B$ car on ne peut pas avoir $s \preccurlyeq^*_\Gamma \forall x : A.B$. On en conclut qu'il ne peux exister de tel terme M et donc que $\mathcal{P}$ est cohérent. ☺

Comme indiqué dans l'introduction de cette sous-section, les morphismes sont un outil fort commode pour prouver des résultats de cohérence relative :

1.1.68 Lemme
Supposons $\mathcal{P} \hookrightarrow_\varphi \mathcal{P}'$.
- *Si $\varphi(s) \in \mathcal{S}_\mathcal{P}$ est $\varphi(\Gamma)$-cohérent, alors s est Γ-cohérente.*
- *Si $\mathcal{P}'$ est cohérent, alors $\mathcal{P}$ est cohérent.*

Démonstration
- Si s n'est pas Γ-cohérente, alors il existe un terme M tel que $\Gamma, \alpha : s \vdash_\mathcal{P} M : \alpha$ et donc d'après le lemme 1.1.63, on en déduit $\varphi(\Gamma), \alpha : \varphi(s) \vdash_\mathcal{P} \varphi(M) : \alpha$ et donc $\varphi(s)$ n'est pas $\varphi(\Gamma)$-cohérente.
- C'est une conséquence directe du cas précédent. ☺

1.1.10 La théorie des types naïve

L'un des premiers PTS que l'on est tenté de considérer est le PTS à une sorte, un axiome et une règle. On appellera ce système $\lambda\star$.

$$\begin{aligned} \mathcal{S}_{\lambda\star} &= \{\star\} \\ \mathcal{A}_{\lambda\star} &= \{(\star,\star)\} \\ \mathcal{R}_{\lambda\star} &= \{(\star,\star,\star)\} \\ \mathcal{C}_{\lambda\star} &= \emptyset \end{aligned}$$

Le système que l'on obtient correspond à la première version du système de type de Martin-Löf [Lö71]. Les types habitent tous la même sorte, $\star$, le type de tous les types ; on a donc $\vdash \star : \star$.

Tous les encodages que nous avons étudiés dans le calcul des constructions CC s'importent naturellement dans $\lambda\star$ et il suffit de remplacer toutes les occurrences de $\Box$ dans une dérivation de CC pour en faire une dérivation de $\lambda\star$.

Tout comme CC, le système identifie les propositions et les types de données ; ils habitent la même sorte. En particulier, le vrai, le faux, la conjonction, la disjonction, le connecteur existentiel sont également des encodages respectifs du type singleton, du type vide, du produit cartésien, du type somme,

et de la somme forte. Mais l'axiome $\vdash \star : \star$ a pour effet de ramener également les types au niveau des individus. Ce qui permet par exemple de programmer le prédicat

$$D = \lambda x : \mathsf{bool}\,.\,\mathsf{if}_\star\ x\ \top\ \bot : \mathsf{bool} \to \star$$

qui permet de distinguer le booléen true du booléen false. Le terme de preuve

$$\lambda h : \mathsf{true} =_{\mathsf{bool}} \mathsf{false}\,.h\ D\ (\lambda X : \star, h : X.h)$$

prouve $\vdash \mathsf{true} =_{\mathsf{bool}} \mathsf{false} \to \bot$. Cette propriété qui implique l'inexistence de modèles triviaux (dans le sens où les interprétations des types sont réduits à des ensembles d'au plus un élément) est indémontrable dans la logique du second-ordre et dans le calcul des constructions. C'est l'une des raisons principales qui a amené à l'introduction de type inductifs dans le calcul des constructions (comme nous le verrons plus avant, c'est grâce à l'élimination forte des types inductifs que l'on peut construire un prédicat analogue D pour le calcul des constructions inductives).

Le système que l'on obtient est donc très expressif et l'utiliser comme langage de programmation est à bien des égards très agréable, hélas Jean-Yves Girard prouva dans sa thèse d'État [Gir72] que ce système est incohérent : on dispose d'un terme π tel que $\vdash \pi : \bot$. Or par inversion des règles, on peut montrer qu'il n'existe aucune expression en forme normale de type $\bot$; on en déduit donc que π n'est pas (faiblement) normalisante. Et donc que $\lambda\star$ n'est pas faiblement normalisant. C'est regrettable car on aurait pu en déduire la normalisation de tous les PTS :

1.1.69 Lemme (Terminalité de $\lambda\star$)
Soit $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{R})$ un PTS. La fonction

$$\begin{array}{ccc} \mathcal{S} & \longrightarrow & \{\star\} \\ s & \longmapsto & \star \end{array}$$

est un morphisme de PTS entre $\mathcal{P}$ et $\lambda\star$.

On en déduit donc que

1.1.70 Lemme
Si $\lambda\star$ est faiblement normalisant (resp. fortement normalisant), alors tous les PTS sont faiblement normalisants (resp. fortement normalisants).

Du point du vue du logicien, ce système ne présent donc plus beaucoup d'intérêt, toutes les propositions sont démontrables. Néanmoins, du point du vue du programmeur, ce système permet de représenter toutes les fonctions calculables. En effet, Herman Geuvers et Joep Verkoelen [GV09] ont réussi à implementer dans $\lambda\star$ un "combinateur de bouclage" (*looping combinator* en anglais) permettant de programmer dans ce système toutes les fonctions récursives.

1.2 Propriétés liées à la cumulativité

Nous avons choisi de n'imposer aucune condition sur le paramètre $\mathcal{C}$ de la définition des WCTS. En pratique néanmoins la relation de cumulativité des exemples que nous étudierons satisfera un grand

nombre de bonnes propriétés. Dans cette sous-section, nous étudierons les répercussions de ces propriétés sur le système.

Nous nous intéresserons tout particulièrement aux conditions sur les paramètres faciles à vérifier (elles seront même très clairement décidables dans le cas fini). Nous prouverons les implications (certaines ont déjà été démontrées) de la figure 1.2.

Les propriétés DOWNSTABLE, MINLOC, et PRINCIPAL sont toutes les trois des conditions sur les paramètres. La condition DOWNSTABLE impliquera (si l'on admet la préservation du typage) que le WCTS type exactement les même termes que le CTS obtenu en prenant la clôture transitive de $\mathcal{C}$. La condition MINLOC plus forte que DOWNSTABLE généralise la condition de fonctionnalité vue précédemment (définition 1.1.9). Elle implique que si nous avons deux types A_1 et A_2 d'un même terme, alors il existe un type A de ce terme tel que $A \preccurlyeq A_1$ et $A \preccurlyeq A_2$. Et finalement, la propriété PRINCIPAL est la conjonction de MINLOC et de la bonne fondation de la relation de cumulativité. On montrera qu'elle implique l'existence d'un type principal, c'est-à-dire l'existence d'un plus petit type (au sens de $\preccurlyeq$) pour tout terme bien-typé.

1.2.1 Propriétés ne dépendant pas du typage

1.2.1 Lemme

$$A \prec_{n+1} B \quad \Leftrightarrow \quad A \prec_n B \quad \vee \quad \exists A', B', C, \begin{cases} A \trianglerighteq \forall x : C.A' \\ B \trianglerighteq \forall x : C.B' \\ A' \prec_n B' \end{cases}$$

Démonstration $\Rightarrow$: Supposons $A \equiv \forall x : C'.A''$, $B \equiv \forall x : C'.B''$ et $A'' \prec_n B''$. Par confluence, il existe C'', A', C''' et B' tels que $A \trianglerighteq \forall x : C''.A'$, $B \trianglerighteq \forall x : C'''.B'$ et $A' \equiv A''$, $B' \equiv B''$, $C' \equiv C'' \equiv C'''$. Comme $\prec_n$ est compatible avec $\equiv$ (lemme 1.1.7), nous obtenons $A' \prec_n B'$. Enfin, la confluence nous donne qu'il existe également C tel que $C'' \trianglerighteq C$ et $C''' \trianglerighteq C$. Et nous avons alors bien prouvé que $A \trianglerighteq \forall x : C.A'$, $B \trianglerighteq \forall x : C.B'$ et $A' \prec_n B'$.

$\Leftarrow$: La réciproque est immédiate car $\trianglerighteq \subseteq \equiv$. ☺

1.2.2 Lemme

$$C \prec_{n+1} \forall x : A.B \quad \Leftrightarrow \quad \exists A_0, B_0, \begin{cases} C \trianglerighteq \forall x : A_0.B_0 \\ A \trianglerighteq A_0 \\ B_0 \prec_n B \end{cases}$$

$$C \prec \forall x : A.B \quad \Leftrightarrow \quad \exists A_0, B_0, \begin{cases} C \trianglerighteq \forall x : A_0.B_0 \\ A \trianglerighteq A_0 \\ B_0 \prec B \end{cases}$$

$$C \preccurlyeq \forall x : A.B \quad \Leftrightarrow \quad \exists A_0, B_0, \begin{cases} C \trianglerighteq \forall x : A_0.B_0 \\ A \trianglerighteq A_0 \\ B_0 \preccurlyeq_n B \end{cases}$$

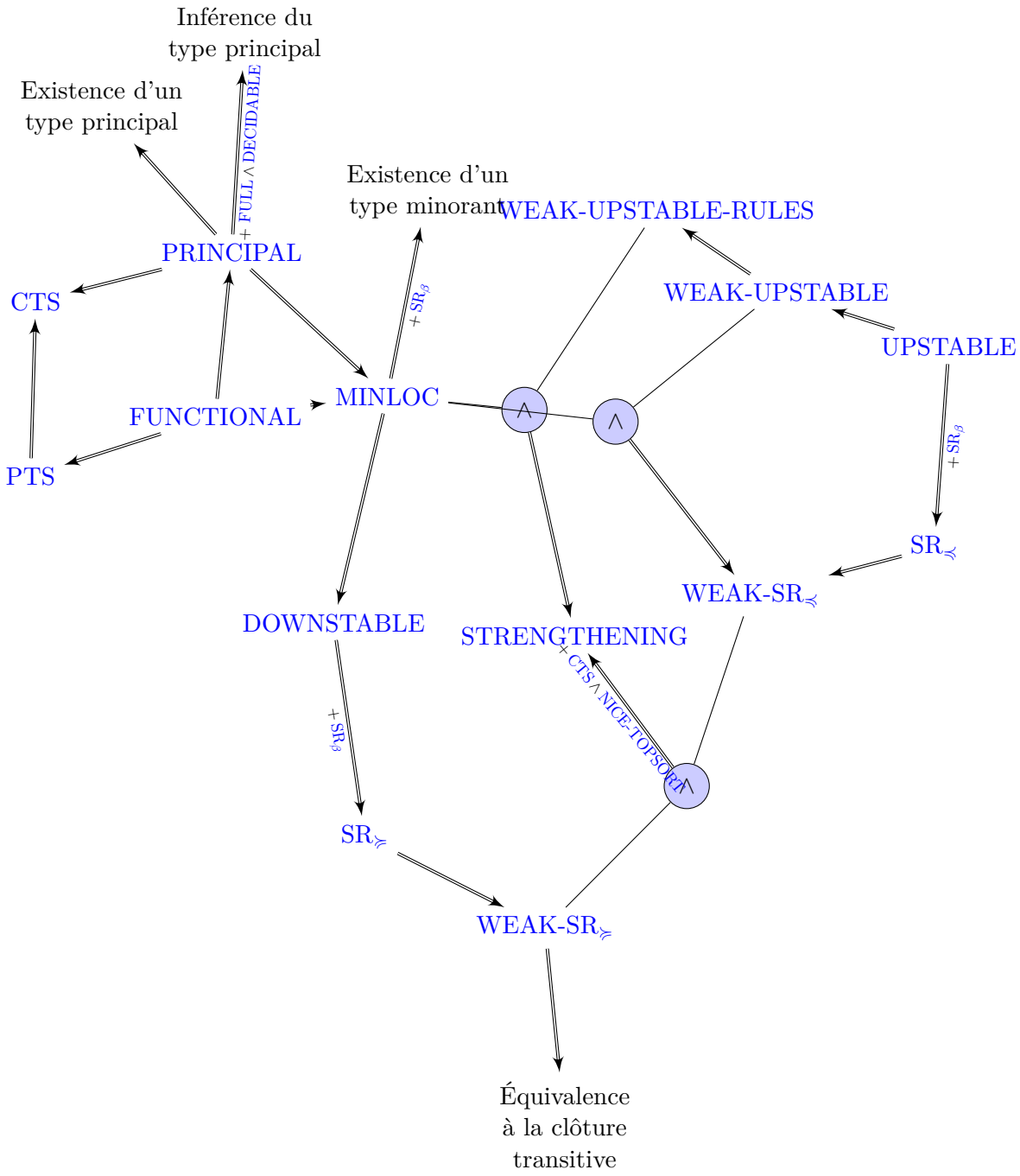


FIGURE 1.2 – Propriétés des WCTS

$$C \preccurlyeq^* \forall x : A.B \quad \Leftrightarrow \quad \exists A_0, B_0, \begin{cases} C \trianglerighteq \forall x : A_0.B_0 \\ A \trianglerighteq A_0 \\ B_0 \preccurlyeq^* B \end{cases}$$

$$\left.\begin{array}{lll} C & \preccurlyeq^* & \forall x : A_2.B_2 \\ C & \preccurlyeq^* & \forall x : A_2.B_2 \end{array}\right\} \quad \Leftrightarrow \quad \exists A_0, B_0, \begin{cases} A_1 \trianglerighteq A_0 \\ A_2 \trianglerighteq A_0 \\ C \trianglerighteq \forall x : A_0.B_0 \\ B_0 \preccurlyeq^* B_1 \\ B_0 \preccurlyeq^* B_2 \end{cases}$$

Démonstration Les réciproques sont immédiates, nous ne montrerons que les sens directs :

1. Par induction sur n.
   - Si on a $C \prec_1 \forall x : A.B$, le cas $C \prec_0 \forall x : A.B$ étant impossible d'après la confluence, le lemme 1.2.1 nous dit que l'on est nécessairement dans le cas où $C \trianglerighteq \forall x : A_0.B_0$, $\forall x : A.B \trianglerighteq \forall x : A_0.B'$ (en particulier $A \trianglerighteq A_0$), et $B_0 \prec_0 B'$ pour certains A_0, B_0 et B'. Or $B' \equiv B$ et d'après le lemme 1.1.7, on a bien $B_0 \prec_0 B$.
   - Si on a $C \prec_{n+2} \forall x : A.B$, d'après le lemme 1.2.1, deux cas sont possibles :
     - Soit $C \prec_{n+1} \forall x : A.B$, et on conclut par hypothèse d'induction.
     - Soit il existe A_0, B_0 et B' tels que $C \trianglerighteq \forall x : A_0.B_0$, $\forall x : A.B \trianglerighteq \forall x : A_0.B'$ (en particulier $A \trianglerighteq A_0$), et $B_0 \prec_{n+1} B'$. Or $B' \equiv B$ et d'après le lemme 1.1.7, on a bien $B_0 \prec_{n+1} B$.
2. Si $C \prec \forall x : A.B$, alors il existe un n tels que $C \prec_n \forall x : A.B$, or par confluence on a $n > 0$ et d'après le cas précédent, on en déduit qu'il existe A_0 et B_0 tels que $C \trianglerighteq \forall x : A_0.B_0$, $A \trianglerighteq A_0$, et $B_0 \prec_{n-1} B$. Et donc on a bien $B_0 \prec B$.
3. Si $C \equiv \forall x : A.B$, alors on peut conclure en prenant $A_0 = A$ et $B_0 = B$. Si $C \not\equiv \forall x : A.B$, on se ramène au cas précédent.
4. Par induction sur la longueur de la chaîne menant de C à $\forall x : A.B$:
   - Si $C = \forall x : A.B$, c'est évident.
   - Supposons que l'on ait $C_0 \preccurlyeq C_1 \preccurlyeq^* \forall x : A.B$ et que l'on ait par hypothèse d'induction A_1 et B_1 tels que $C_1 \trianglerighteq \forall x : A_1.B_1$, $A \trianglerighteq A_1$, et $B_1 \preccurlyeq^* B$. Alors d'après le lemme 1.1.7, $C_0 \preccurlyeq \forall x : A_1.B_1$ et on peut alors appliquer le cas précédent pour trouver A_0 et B_0 tels que $C_0 \trianglerighteq \forall x : A_0.B_0$, $A_1 \trianglerighteq A_0$, et $B_0 \preccurlyeq B_1$. Et on conclut bien que $B_0 \preccurlyeq^* B$.
5. C'est une conséquence de la confluence et du lemme 1.1.7. ☺

Symétriquement, on obtient :

1.2.3 Lemme

$$\forall x : A.B \prec_{n+1} C \quad \Leftrightarrow \quad \exists A_0, B_0, \begin{cases} C \trianglerighteq \forall x : A_0.B_0 \\ A \trianglerighteq A_0 \\ B \prec_n B_0 \end{cases}$$

$$\forall x : A.B \prec C \quad \Leftrightarrow \quad \exists A_0, B_0, \begin{cases} C \trianglerighteq \forall x : A_0.B_0 \\ A \trianglerighteq A_0 \\ B \prec B_0 \end{cases}$$

$$\forall x : A.B \preccurlyeq C \quad \Leftrightarrow \quad \exists A_0, B_0, \begin{cases} C \trianglerighteq \forall x : A_0.B_0 \\ A \trianglerighteq A_0 \\ B \preccurlyeq_n B_0 \end{cases}$$

$$\forall x : A.B \preccurlyeq C \quad \Leftrightarrow \quad \exists A_0, B_0, \begin{cases} C \trianglerighteq \forall x : A_0.B_0 \\ A \trianglerighteq A_0 \\ B \preccurlyeq^* B_0 \end{cases}$$

$$\left.\begin{array}{llll} \forall x : A_2.B_2 & \preccurlyeq^* & C \\ \forall x : A_2.B_2 & \preccurlyeq^* & C \end{array}\right\} \quad \Leftrightarrow \quad \exists A_0, B_0, \begin{cases} A_1 \trianglerighteq A_0 \\ A_2 \trianglerighteq A_0 \\ C \trianglerighteq \forall x : A_0.B_0 \\ B_1 \preccurlyeq^* B_0 \\ B_2 \preccurlyeq^* B_0 \end{cases}$$

Démonstration Les preuves sont similaires : il suffit de considérer la relation $\prec'$ générée par le paramètre $\mathcal{C}' = \{(s_1, s_2) | (s_2, s_1) \in \mathcal{C}\}$. ☺

1.2.4 Lemme

1. *Si $A \prec s$ (resp. $s \prec A$), alors il existe s_A telle que $A \equiv s_A$ et $(s_A, s) \in \mathcal{C}$ (resp. $(s, s_A) \in \mathcal{C}$) (et donc $s \prec s_A$).*
2. *Si $A \preccurlyeq s$ (resp. $s \preccurlyeq A$), alors il existe s_A telle que $A \equiv s_A$ et $s_A \preccurlyeq s$ (resp. $s \preccurlyeq s_A$).*

Démonstration On montre par récurrence sur n que si $A \prec_n s$, alors il existe s_A tel que $A \equiv s_A$ et $(s_A, s) \in \mathcal{C}$ (le cas symétrique est similaire). Les cas 1. et 2. en découlent alors naturellement.

- Si $n = 0$, alors $A \prec_0 s$ signifie par définition qu'il existe s_A tel que $A \equiv s_A$ et $(s_A, s) \in \mathcal{C}$.
- Supposons que $A \prec_{n+1} s$, alors par définition deux cas sont possibles :
  - Soit $A \prec_n s$ et on peut conclure en invoquant l'hypothèse de récurrence.
  - Soit s est convertible en un produit, ce qui est impossible. ☺

1.2.5 Lemme

Pour tout terme A, B, C, D :

$$A \prec_n B \wedge (B \prec_m C \vee A \prec_m D \vee A \prec_m C \vee D \ \prec_m B) \Rightarrow A \prec_{\min(n,m)} B$$

Démonstration On peut définir une famille de plus petits ensembles $(\mathbf{D}_n)_{n \in \mathbb{N}}$ de termes modulo $\equiv$ vérifiant pour tout entier $n \in \mathbb{N}$,

- Si $A \equiv B$, alors $A \in \mathbf{D}_n \Leftrightarrow B \in \mathbf{D}_n$,
- Pour tout $s \in \mathcal{S}$, $s \in \mathbf{D}_n$,
- $\forall x : A.B \in \mathbf{D}_{n+1} \Leftrightarrow B \in \mathbf{D}_n$.

On a alors par une induction immédiate sur n, que $\mathbf{D}_n \subseteq \mathbf{D}_{n+1}$.

On montre maintenant que :

1. $A \prec_n B \quad \Rightarrow \quad A \in \mathbf{D}_n \wedge B \in \mathbf{D}_n$
2. $A \in \mathbf{D}_n \wedge A \prec B \quad \Rightarrow \quad A \ \prec_n B$
3. $B \in \mathbf{D}_n \wedge A \prec B \quad \Rightarrow \quad A \ \prec_n B$

On procède par induction sur n :

- Si $n = 0$, alors
  1. Supposons $A \prec_0 B$. Par définition cela signifie qu'il existe deux sortes $(s_A, s_B) \in \mathcal{C}$ telles que $A \equiv s_A$ et $B \equiv s_B$. Et on a bien $s_A \in \mathbf{D}_0$ et $s_B \in \mathbf{D}_0$.
  2. Supposons $A \in \mathbf{D}_0$ et $A \prec B$. Par définition, il existe une sorte s_A telle que $A \equiv s_A$. Et d'après le lemme précédent on en déduit que $s_A \prec B$ et donc que $A \ \prec B$.
  3. Similaire au cas précédent.
- Supposons la propriété vraie à un rang n, on montrons la au rang $n+1$:
  1. Supposons $A \prec_{n+1} B$. Par définition, cela signifie que
     - Soit $A \prec_n B$ et on conclut en invoquant l'hypothèse de récurrence que $A \in \mathbf{D}_n$ et $B \in \mathbf{D}_n$. Et donc $A \in \mathbf{D}_{n+1}$ et $B \in \mathbf{D}_{n+1}$.
     - Soit il existe A', B', D, tels que $A \equiv \forall x : D.A'$ et $B \equiv \forall x : D.B'$ avec $A' \prec_n B'$. Dans ce cas, par hypothèse de récurrence, on a que $A' \in \mathbf{D}_n$ et $B' \in \mathbf{D}_{n+1}$. Et par définition de $\mathbf{D}$, on en déduit que $\forall x : D.A' \in \mathbf{D}_{n+1}$ et $\forall x : D.B' \in \mathbf{D}_{n+1}$, et donc que $A \in \mathbf{D}_{n+1}$ et $B \in \mathbf{D}_{n+1}$.
  2. Supposons $A \in \mathbf{D}_{n+1}$ et $A \prec_m B$, alors par définition de $\mathbf{D}$, on a que :
     - Soit $A \in \mathbf{D}_n$, et on obtient par hypothèse de récurrence que $A \prec_n B$ et donc $A \prec_{n+1} B$.
     - Soit il existe D et A' tel que $A \equiv \forall x : D.A'$ et $A' \in \mathbf{D}_n$. On a donc $\forall x : D.A' \prec B$ et d'après le lemme 1.2.3, cela signifie qu'il existe B' tel que $B \equiv \forall x : D.B'$ et $A' \prec B'$ et donc par hypothèse de récurrence que $A' \prec_n B'$ et donc $\forall x : D.A' \prec_{n+1} \forall x : D.B'$. On conclut alors bien que $A \prec_{n+1} B$.
  3. Similaire au cas précédent.

On peut maintenant démontrer l'énoncé du lemme. Supposons $A \prec_n B$ et montrons que $A \prec_m B$ (si $n \leq m$ il n'y a rien à démontrer) :

- Si $B \prec_m C \vee D \ \prec_m B$, alors $B \in \mathbf{D}_m$ et donc $A \prec_m B$.
- Si $A \prec_m D \vee A \prec_m C$, alors $a \in \mathbf{D}_m$ et donc $A \prec_m B$. ☺

1.2.6 Lemme

Si $\mathcal{C}$ est une relation transitive, alors $\prec$ est également transitive (et donc $\prec^\star = \prec$).

Démonstration Par induction sur n, on montre que $\prec_n$ est transitive.

- $\prec_0$ est transitif : Soit $A \prec_0 B$ et $B \prec_0 C$ par définition on sait qu'il existe s_A, s_B, s_C tels que $A \equiv s_A$, $B \equiv s_B$ et $C \equiv s_C$ tels que $(s_A, s_B) \in \mathcal{C}$ et $(s_B, s_C) \in \mathcal{C}$. On prouve donc par transitivité de $\mathcal{C}$ que $(s_A, s_C) \in \mathcal{C}$ et donc que $A \prec_0 C$.
- $\prec_{n+1}$ est transitif (sachant $\prec_n$ transitif) : Soit $A \prec_{n+1} B$ et $B \prec_{n+1} C$ par définition :

– Soit
$$\begin{cases} A \equiv \forall x : C_1.A' \\ B \equiv \forall x : C_1.B'_1 \\ A' \prec_n B'_1 \end{cases} \quad \text{et} \quad \begin{cases} B \equiv \forall x : C_2.B'_2 \\ C \equiv \forall x : C_2.C' \\ B'_2 \prec_n C' \end{cases}$$
et donc $\forall x : C_1.B'_1 \equiv \forall x : C_2.B'_2$. On en déduit $C_1 \equiv C_2$ et $B'_1 \equiv B'_2$. Le lemme 1.1.7 nous montre donc $C \equiv \forall x : C_1.C'$ et $A' \prec_n B'_2$ et par hypothèse de récurrence $A' \prec_n C'$. Enfin, par définition de $\prec_{n+1}$ on obtient $A \prec_{n+1} C$.
– Soit $A \prec_n B$ et $B \prec_n C$: on a $A \prec_n C$ par hypothèse d'induction.
– Soit $A \prec_{n+1} B$ et $B \prec_n C$ ou $A \prec_n B$ et $B \prec_{n+1} C$, alors d'après le lemme 1.2.5, on a $A \prec_n B$ et $B \prec_n C$ et on peut se reporter au cas précédent. ☺

On en déduit le corollaire suivant :

1.2.7 Lemme
Si $\mathcal{C}$ est transitive, alors $\preccurlyeq$ est une relation d'ordre vis-à-vis de $\equiv$, c'est-à-dire :
– *Réflexivité : $A \equiv B \Rightarrow A \preccurlyeq B$,*
– *Anti-symétrie : $A \preccurlyeq B \wedge B \preccurlyeq A \Rightarrow A \equiv B$,*
– *Transitivité : $A \preccurlyeq B \wedge B \preccurlyeq C \Rightarrow A \preccurlyeq C$.*

1.2.2 Clôture par transitivité

Dans cette sous-section, nous étudierons les propriétés des WCTS obtenus en prenant la clôture transitive de la relation de cumulativité.

1.2.8 Définition (Clôture par transitivité)
Soit $\mathcal{P}$ un WCTS, on définit $\mathcal{P}^*$ la clôture par transitivité par
$$\begin{cases} \mathcal{S}_{\mathcal{P}^*} = \mathcal{S}_{\mathcal{P}} \\ \mathcal{A}_{\mathcal{P}^*} = \mathcal{A}_{\mathcal{P}} \\ \mathcal{R}_{\mathcal{P}^*} = \mathcal{R}_{\mathcal{P}} \\ \mathcal{C}_{\mathcal{P}^*} = (\mathcal{C}_{\mathcal{P}})^* \end{cases}$$
où $(\mathcal{C}_{\mathcal{P}})^*$ désigne la clôture transitive de $\mathcal{C}_{\mathcal{P}}$.

Il faut noter que $\mathcal{P}^*$ et $\mathcal{P}$ ne typent pas nécessairement exactement les mêmes termes, si l'on prend par exemple PTS suivant :
$$\begin{cases} \mathcal{S}_{\mathcal{P}} = \{\star_1, \star_2, \star_3, \Box\} \\ \mathcal{A}_{\mathcal{P}} = \{(\star_1, \Box), (\star_3, \Box)\} \\ \mathcal{R}_{\mathcal{P}} = \{(\Box, \Box, \Box)\} \\ \mathcal{C}_{\mathcal{P}} = \{(\star_1, \star_2), (\star_2, \star_3)\} \end{cases}$$
On a $\beta : \forall \alpha : s.\star_1 \vdash_{\mathcal{P}^*} \beta : \forall \alpha : s.\star_3$ (il suffit d'invoquer la règle de cumulativité car on a bien $\forall \alpha : s.\star_1 \prec_{\mathcal{P}^*} \forall \alpha : s.\star_3$) et $\beta : \forall \alpha : s.\star_1 \not\vdash_{\mathcal{P}} \beta : \forall \alpha : s.\star_3$: on ne peut pas enchaîner deux règles de cumulativité car $\forall \alpha : s.\star_2$ n'est pas bien formé ; mais si l'on rajoute l'axiome $(\star_2, \Box)$ alors le séquent redevient dérivable.

Néanmoins, si la relation de cumulativité préserve la typabilité vers le bas, alors la règle est admissible.

1.2.9 Lemme
Si $\models$ WEAK-SR$_{\preccurlyeq}$, *alors la règle suivante est admissible :*

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'} \; B \preccurlyeq^* B'$$

Démonstration Il faut donc montrer que $\Gamma \vdash A : B'$ sachant que $B \preccurlyeq^* B'$ et $\Gamma \vdash A : B$ et $\Gamma \vdash B' : s$. Deux cas sont possibles :

- Soit $B' \equiv B$, et on prouve $\Gamma \ \vdash A : B'$ grâce à la règle de conversion.
- Soit $B \prec^* B'$ et on sait alors qu'il existe p termes tels que $B = B_1 \prec B_2 \prec \cdots \prec B_p = B'$. Ensuite d'après la propriété WEAK-SR$_{\preccurlyeq}$ et par induction sur p, on montre que $\mathrm{WF}_\Gamma(B_i)$ pour $1 \leq i \leq p$. Enfin, en enchaînant $p-1$ règles de cumulativité (lemme 1.1.28) on prouve $\Gamma \vdash A : B'$. ☺

La règle Cumulativité-Sortes peut toujours être étendue à la clôture transitive (car les sortes sont toujours des types bien-formés par définition),

1.2.10 Lemme
La règle suivante est admissible :

$$\frac{\Gamma \vdash A : C}{\Gamma \vdash A : s} \; C \preccurlyeq^* s$$

Démonstration Supposons $C \preccurlyeq^* s$, alors, d'après le lemme 1.2.4, il existe $n \geq 1$ sortes $C \preccurlyeq s_1 \preccurlyeq \cdots \preccurlyeq s_n = s$. Nous pouvons alors prouver $\Gamma \vdash A : s_n$ en enchaînant $n-1$ règles Cumulativité-Sortes :

$$\cfrac{\cfrac{\cfrac{\Gamma \vdash A : C}{\Gamma \vdash A : s_1} \; C \preccurlyeq s_1}{\Gamma \vdash A : s_2} \; s_1 \preccurlyeq s_2}{\begin{array}{c}\vdots \\ \Gamma \vdash A : s_{n-1}\end{array}} \; s_{n-2} \preccurlyeq s_{n-1}}{\Gamma \vdash A : s_n} \; s_{n-1} \preccurlyeq s_n$$

1.2.11 Lemme
La clôture transitive de $\prec_{\mathcal{P}}$ *est égale à* $\prec_{\mathcal{P}^*}$.

Démonstration
- Si $A_1 \prec_{\mathcal{P}} \cdots \prec_{\mathcal{P}} A_n$, alors $A_1 \prec_{\mathcal{P}^*} \cdots \prec_{\mathcal{P}^*} A_n$ car $\prec_{\mathcal{P}} \subseteq \prec_{\mathcal{P}*}$ or d'après le lemme 1.2.6, $\prec_{\mathcal{P}^*}$ est transitive. On a donc bien $A_1 \prec_{\mathcal{P}^*} A_n$.
- Réciproquement, montrons que $A \prec_{\mathcal{P}^*} B \Rightarrow A \prec^*_{\mathcal{P}} B$. Pour cela, montrons par induction sur n que $A \prec_{\mathcal{P}^*,n} B$ implique $A \prec^*_{\mathcal{P},n} B$:
  - Supposons $A \prec_{\mathcal{P}^*,0} B$, alors par définition, il existe s_A et s_B tels que $(s_A, s_B) \in \mathcal{C}^*$, $A \equiv s_A$ et $B \equiv s_B$. Il existe donc bien p sortes $s_1 = s_A, s_2, \ldots, s_p = s_B$ telles que $(s_i, s_{i+1}) \in \mathcal{C}$ (pour $1 \leq i < p$). On en déduit donc que $s_1 \prec_{\mathcal{P},0} \cdots \prec_{\mathcal{P},0} s_n$ et également que $s_A \prec^*_{\mathcal{P},0} s_B$. Et par compatibilité avec $\equiv$, on en déduit que $A \prec^*_{\mathcal{P},0} B$.
  - Supposons que $A \prec_{\mathcal{P}^*,n+1} B$, alors deux cas sont possibles :

– Soit $A \prec_{\mathcal{P}^*,n} B$ et on conclut par hypothèse d'induction.
– Soit $A \equiv \forall x : C.A'$ et $B \equiv \forall x : C.B'$ avec $A' \prec_{\mathcal{P}^*,n} B'$. Par hypothèse d'induction on aurait $A' \prec_{\mathcal{P},n}{}^* B'$, ce qui signifie qu'il existe p termes tels que $A' = A_1 \prec_{\mathcal{P},n} A_2 \prec_{\mathcal{P},n} \cdots \prec_{\mathcal{P},n} A_p = B'$. Or par définition, on aurait alors

$$A \equiv \forall x : C.A_1 \prec_{\mathcal{P},n+1} \forall x : C.A_2 \prec_{\mathcal{P},n+1} \cdots \prec_{\mathcal{P},n+1} \forall x : C.A_p \equiv B$$

et donc $A \prec^*_{\mathcal{P},n+1} B$. ☺

1.2.12 Lemme
La clôture transitive $\preccurlyeq^*_{\mathcal{P}}$ de $\preccurlyeq_{\mathcal{P}}$ est égale à $\preccurlyeq_{\mathcal{P}^*}$.

Démonstration On commence par montrer le sens direct de l'énoncé suivant (la réciproque est immédiate) : $A \preccurlyeq^*_{\mathcal{P}} B \Leftrightarrow A \equiv B \vee A \prec^*_{\mathcal{P}} B$.

Si $A \preccurlyeq^*_{\mathcal{P}} B$, alors par définition il existe $n \geq 1$ et

$$A_1 \equiv B_1 \prec_{\mathcal{P}} A_2 \equiv B_2 \prec_{\mathcal{P}} \cdots \prec_{\mathcal{P}} A_n \equiv B_n$$

tels que $A_1 = A$ et $B_n = B$. Si $n = 1$, alors on $A \equiv B$. Si $n > 1$, alors d'après le lemme 1.1.7, on a

$$A_1 \prec_{\mathcal{P}} B_2 \prec_{\mathcal{P}} B_3 \cdots \prec_{\mathcal{P}} B_n$$

et donc $A \prec^*_{\mathcal{P}} B$.

Supposons $A \preccurlyeq^* B$, alors soit :
– Soit $A \equiv B$ et dans ce cas on a bien $A \preccurlyeq_{\mathcal{P}^*} B$.
– Soit $A \prec^*_{\mathcal{P}} B$ et d'après le lemme 1.2.11, $A \prec_{\mathcal{P}^*} B$ et donc $A \preccurlyeq_{\mathcal{P}^*} B$. ☺

On déduit alors des lemmes 1.2.9, 1.2.10 et 1.2.12 le lemme ci-dessous.

1.2.13 Lemme
Si $\mathcal{P} \models$ WEAK-SR$_{\succcurlyeq}$,

$$\Gamma \vdash_{\mathcal{P}} A : B \Leftrightarrow \Gamma \vdash_{\mathcal{P}^*} A : B$$

.

1.2.3 Clôture par cumulativité

Dans les WCTS, les règles sont facilement redondantes. En effet, si dans WCTS $\mathcal{P}$ on a $(s, r) \in \mathcal{A}_{\mathcal{P}}$ alors pour toute sorte r' telle que $r \preccurlyeq^* r'$, on a $\vdash s : r'$. Ainsi le WCTS $\mathcal{P}'$ obtenu en ajoutant à $\mathcal{P}$ l'axiome (s, r') type exactement les même termes. Et le même genre de redondance peut apparaître au niveau des règles. C'est pourquoi on introduit la notion ci-dessous.

1.2.14 Définition (Clôture par cumulativité)
Soit $\mathcal{P}$ un WCTS, on définit la clôture par cumulativité $\overline{\mathcal{P}}$ de $\mathcal{P}$ de la façon suivante :

$$\begin{aligned}
\mathcal{S}_{\overline{\mathcal{P}}} &= \mathcal{S}_{\mathcal{P}} \\
\mathcal{A}_{\overline{\mathcal{P}}} &= \{(s, s') \in (\mathcal{S}_{\mathcal{P}})^2 | \exists s'', (s, s'') \in \mathcal{A}_{\mathcal{P}}, s'' \preccurlyeq^*_{\mathcal{P}} s'\} \\
\mathcal{R}_{\overline{\mathcal{P}}} &= \{(s_1, s_2, s_3) \in (\mathcal{S}_{\mathcal{P}})^3 | \exists (s'_1, s'_2, s'_3) \in \mathcal{R}_{\mathcal{P}}, s_1 \preccurlyeq^*_{\mathcal{P}} s'_1 \wedge s_2 \preccurlyeq^*_{\mathcal{P}} s'_2 \wedge s'_3 \preccurlyeq^*_{\mathcal{P}} s_3\} \\
\mathcal{C}_{\overline{\mathcal{P}}} &= \mathcal{C}_{\mathcal{P}}
\end{aligned}$$

Dans un souci de concision, on désignera également $\mathcal{A}_{\overline{\mathcal{P}}}$ (resp. $\mathcal{R}_{\overline{\mathcal{P}}}$) par $\overline{\mathcal{A}}$ (resp. $\overline{\mathcal{R}}$).

Le WCTS $\overline{\mathcal{P}}$ est obtenu en saturant les axiomes et les règles par les conséquences de la relation de cumulativité. Nous verrons que $\mathcal{P}$ et $\overline{\mathcal{P}}$ typent exactement les mêmes termes (bien que $\overline{\mathcal{P}}$ contienne des dérivations plus courtes que $\mathcal{P}$ car elles peuvent faire moins appel aux règles de cumulativité).

1.2.15 Lemme
La clôture par cumulativité est un opérateur de fermeture :

$$\overline{\mathcal{P}} \subseteq \overline{\mathcal{P}'} \Leftrightarrow \mathcal{P} \subseteq \overline{\mathcal{P}'}$$

On en déduit immédiatement les propriétés suivantes :

1. *Extensif :* $\mathcal{P} \subseteq \overline{\mathcal{P}}$,
2. *Croissant :* $\mathcal{P} \subseteq \mathcal{P}' \quad \Rightarrow \quad \overline{\mathcal{P}} \subseteq \overline{\mathcal{P}'}$,
3. *Idempotent :* $\overline{\mathcal{P}} = \overline{\overline{\mathcal{P}}}$.

Démonstration Il faut montrer que si $\mathcal{P} \subseteq \overline{\mathcal{P}'}$ (*), alors $\overline{\mathcal{P}} \subseteq \overline{\mathcal{P}'}$ (la réciproque étant directe car $\mathcal{P} \subseteq \overline{\mathcal{P}}$). On a $\mathcal{S}_{\overline{\mathcal{P}}} = \mathcal{S}_{\mathcal{P}} \subseteq \mathcal{S}_{\mathcal{P}'}$ et $\mathcal{C}_{\overline{\mathcal{P}}} = \mathcal{C}_{\mathcal{P}} \subseteq \mathcal{C}_{\overline{\mathcal{P}'}}$. Il nous reste à montrer que $\mathcal{A}_{\overline{\mathcal{P}}} \subseteq \mathcal{A}_{\overline{\mathcal{P}'}}$ et $\mathcal{R}_{\overline{\mathcal{P}}} \subseteq \mathcal{R}_{\overline{\mathcal{P}'}}$:

- Soit $(s, s') \in \mathcal{A}_{\overline{\mathcal{P}}}$, par définition, il existe s'' tel que $(s, s'') \in \mathcal{A}_{\mathcal{P}}$ et $s'' \preccurlyeq^*_{\mathcal{P}} s'$. Or d'après (*), on a $(s, s'') \in \mathcal{A}_{\overline{\mathcal{P}'}}$ et $s'' \preccurlyeq^*_{\overline{\mathcal{P}'}} s'$ et donc $(s, s') \in \mathcal{A}_{\overline{\overline{\mathcal{P}'}}} = \mathcal{A}_{\overline{\mathcal{P}'}}$.
- Soit $(s_1, s_2, s_3) \in \mathcal{R}_{\overline{\mathcal{P}}}$, par définition, il existe $(s'_1, s'_2, s'_3) \in \mathcal{R}_{\mathcal{P}}$ tels que $s_1 \preccurlyeq^*_{\mathcal{P}} s'_1 \wedge s_2 \preccurlyeq^*_{\mathcal{P}} s'_2 \wedge s'_3 \preccurlyeq^*_{\mathcal{P}} s_3$. Or d'après (*), on a $(s'_1, s'_2, s'_3) \in \mathcal{R}_{\overline{\mathcal{P}'}}$, $s_1 \preccurlyeq^*_{\overline{\mathcal{P}'}} s'_1$, $s_2 \preccurlyeq^*_{\overline{\mathcal{P}'}} s'_2$ et $s'_3 \preccurlyeq^*_{\overline{\mathcal{P}'}} s_3$. Donc $(s_1, s_2, s_3) \in \mathcal{R}_{\overline{\overline{\mathcal{P}'}}} = \mathcal{R}_{\overline{\mathcal{P}'}}$. ☺

1.2.16 Lemme
Les règles suivantes sont admissibles :

$$\frac{}{\vdash s : s'}\,(s, s') \in \overline{\mathcal{A}} \qquad\qquad \frac{\Gamma \vdash A : s_A \qquad \Gamma, x : A \vdash B : s_B}{\Gamma \vdash \forall x : A.B : s}\,(s_A, s_B, s) \in \overline{\mathcal{R}}$$

Démonstration

- Axiome : Si $(s, s') \in \mathcal{A}_{\overline{\mathcal{P}}}$ et par définition, il existe s'' tel que $(s, s'') \in \mathcal{A}$ et $s'' \preccurlyeq^* s'$. Et donc d'après le lemme 1.2.10, on a :

$$\frac{\dfrac{}{\vdash s : s''} \qquad s'' \preccurlyeq^* s'}{\vdash s : s'}$$

- Produit : Si $(s_A, s_B, s) \in \mathcal{R}_{\overline{\mathcal{P}}}$. Ensuite, par définition, il existe $(s'_A, s'_B, s') \in \mathcal{R}$ tels que $s_A \preccurlyeq^* s'_A$, $s_B \preccurlyeq^* s'_B$ et $s' \preccurlyeq^* s$. Et donc d'après le lemme 1.2.10, on a :

$$\frac{\dfrac{\dfrac{\Gamma \vdash A : s_A \qquad s_A \preccurlyeq^* s'_A}{\Gamma \vdash A : s'_A} \qquad \dfrac{\Gamma, x : A \vdash B : s_B \qquad s_B \preccurlyeq^* s'_B}{\Gamma, x : A \vdash B : s'_B}}{\Gamma \vdash \forall x : A.B : s'} \qquad s' \preccurlyeq^* s}{\Gamma \vdash \forall x : A.B : s}$$

Nous en déduisons alors le corollaire suivant :

1.2.17 Corollaire
La typabilité des termes de $\mathcal{P}$ est équivalente à celle de $\overline{\mathcal{P}}$:

$$\Gamma \vdash_{\mathcal{P}} A : B \Leftrightarrow \Gamma \vdash_{\overline{\mathcal{P}}} A : B$$

et donc

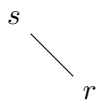
$$\overline{\mathcal{P}} = \overline{\mathcal{P}'} \Rightarrow (\Gamma \vdash_{\mathcal{P}} A : B \Leftrightarrow \Gamma \vdash_{\mathcal{P}'} A : B)$$

Ce corollaire nous fournit un bon moyen de comparer les WCTS, en effet si $\overline{\mathcal{P}} = \overline{\mathcal{P}'}$ nous considérerons que $\mathcal{P}$ et $\mathcal{P}'$ sont deux présentations équivalentes du même WCTS.

1.2.4 Stabilité vers le bas

Avant d'étudier les conditions sur les paramètres, nous allons présenter une représentation diagrammatique des contraintes. Elle servira –nous l'espérons– à faciliter la lecture de cette section mais elle ne remplacera pas les formulations explicites des conditions.

Nous représenterons par un trait simple la relation de cumulativité. Les diagrammes se liront de la gauche vers la droite ; les éléments gauches seront plus petits (au sens de $\preccurlyeq$) que ceux auxquels ils sont reliés à leur droite. Ainsi, le diagramme ci-dessous signifiera $s \preccurlyeq r$.



Nous représenterons les axiomes par une ligne entourée de deux points qui se lira également de la gauche vers la droite. Le diagramme ci-dessous indiquera donc le fait que $(s, r) \in \mathcal{A}$.

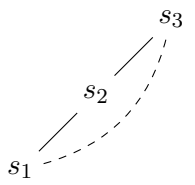
$$s \mathrel{-\!\!\div} r$$

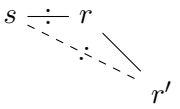
La relation $\mathcal{R}$ est une relation ternaire, nous la représenterons par une flèche qui part de la sorte du domaine, traverse la sorte du codomaine et arrive sur la sorte du produit. Ainsi, la condition $(s_1, s_2, s_3) \in \mathcal{R}$ sera représentée par :

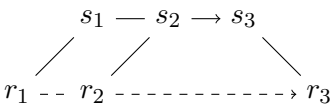
$$s_1 \mathrel{—} s_2 \longrightarrow s_3$$

Suivant la convention venue du domaine de la réécriture, nous représenterons les conditions «existentielles» par des pointillés.

Par exemples, les WCTS clôts par transitivité et par cumulativité (tels que $\mathcal{P}^* = \overline{\mathcal{P}} = \mathcal{P}$) doivent vérifier les trois diagrammes suivants :

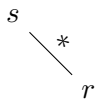






Comme nous exprimerons également des conditions pour les WCTS qui ne sont *a priori* ni clos par transitivité, ni clos par cumulativité. Il nous faudra représenter la clôture transitive $\preccurlyeq^*$ de la relation de cumulativité ainsi que l'appartenance à $\overline{\mathcal{A}}$ et à $\overline{\mathcal{R}}$.

Nous rajouterons une astérisque pour la clôture transitive et nous doublerons les lignes pour l'appartenance à $\overline{\mathcal{A}}$ et à $\overline{\mathcal{R}}$. Ainsi les diagrammes suivants représenteront respectivement $s \preccurlyeq^* r$, $(s,r) \in \overline{\mathcal{A}}$ et $(s_1, s_2, s_3) \in \overline{\mathcal{R}}$.



$s \mathrel{=\!\!\div} r$ $\qquad$ $s_1 = s_2 \Rightarrow s_3$

Enfin, nous représenterons la typabilité sous un contexte Γ par le diagramme suivant :

$$A \mathrel{{}_{\Gamma}\!\div} B$$

Qui signifiera que $\Gamma \vdash A : B$. On notera que dans le cas particulier où $\Gamma = \langle\rangle$ est vide et $A = s$ et $B = r$ sont deux sortes, le diagramme

$$s \mathrel{{}_{\langle\rangle}\!\div} r$$

signifie que $(s,r) \in \overline{\mathcal{A}}$ et non $(s,r) \in \mathcal{A}$.

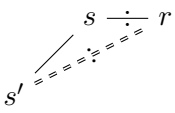
Sous hypothèse de préservation du typage, la définition suivante est une condition suffisante pour que la typabilité soit stable vers le bas par la cumulativité (lemme 1.2.21) : un sous-type d'un type bien formé est bien formé (modulo β-réduction). Elle entraînera l'équivalence à la clôture par transitivité.

1.2.18 Définition (Stabilité vers le bas)
On dira d'un WCTS qu'il est stable vers le bas s'il vérifie la condition suivante :

$$\left.\begin{array}{c}(s,r) \in \mathcal{A} \\ s' \preccurlyeq s\end{array}\right\} \Rightarrow (s',r) \in \overline{\mathcal{A}} \qquad \text{(DOWNSTABLE)}$$

On notera cette propriété DOWNSTABLE.

Nous représenterons graphiquement cette condition par :



Nous avons choisi la formulation de la condition la plus simple à vérifier, mais elle est admet d'autres formulations équivalentes.

1.2.19 Lemme
La condition DOWNSTABLE *est équivalente aux formulations suivantes :*

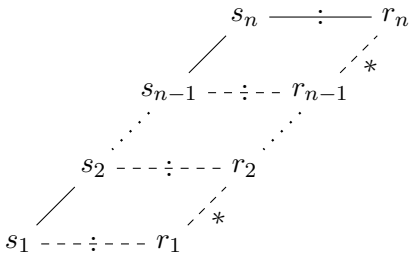
$$\left.\begin{array}{c}(s,r) \in \mathcal{A} \\ s' \preccurlyeq^* s\end{array}\right\} \Rightarrow (s',r) \in \overline{\mathcal{A}} \qquad (1)$$

$$\left.\begin{array}{c}(s,r)\in\overline{\mathcal{A}}\\ s'\preccurlyeq s\end{array}\right\}\Rightarrow(s',r)\in\overline{\mathcal{A}}\qquad(2)$$

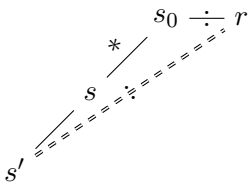
$$\left.\begin{array}{c}(s,r)\in\overline{\mathcal{A}}\\ s'\preccurlyeq^* s\end{array}\right\}\Rightarrow(s',r)\in\overline{\mathcal{A}}\qquad(3)$$

Démonstration On a trivialement $(k)\Rightarrow$ DOWNSTABLE pour $k=1,\dots,3$ car les prémisses des (k) sont plus générales que celle de DOWNSTABLE. Il reste à prouver les réciproques suivantes :

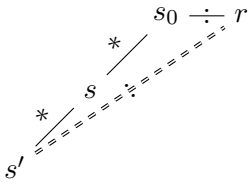
1. DOWNSTABLE $\Rightarrow$ (1) : Si $s\preccurlyeq^* s'$, alors il existe une chaîne $s=s_1\preccurlyeq\cdots\preccurlyeq s_n=s'$. On construit alors une chaîne $r_1\preccurlyeq^* r_2\preccurlyeq^*\cdots\preccurlyeq^* r_{n-1}\preccurlyeq^* r$ telle que $(s_k,r_k)\in\mathcal{A}$ pour $k=1,\dots,n$ en invoquant $n-1$ fois DOWNSTABLE :



2. DOWNSTABLE $\Rightarrow$ (2) : Si $(s,r)\in\overline{\mathcal{A}}$, cela signifie qu'il existe s_0 telle que $(s_0,r)\in\mathcal{A}$ et $s\preccurlyeq^* s_0$. On a donc $s'\preccurlyeq^* s_0$ et le point (1) montre que $(s',r)\in\overline{\mathcal{A}}$.



3. DOWNSTABLE $\Rightarrow$ (3) : Si $(s,r)\in\overline{\mathcal{A}}$, alors par définition il existe s_0 telle que $(s_0,r)\in\mathcal{A}$ et $s\preccurlyeq^* s_0$. De plus, si on a $s'\preccurlyeq^* s$, on en déduit alors $s'\preccurlyeq^* s_0$ et on peut conclurent en utilisant (1).



☺

On en déduit facilement que cette condition est stable par les deux opérations de clôture.

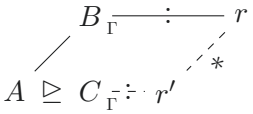
1.2.20 Lemme

$$\mathcal{P} \models \text{DOWNSTABLE} \quad \Leftrightarrow \quad \mathcal{P}^* \models \text{DOWNSTABLE}$$
$$\Leftrightarrow \quad \overline{\mathcal{P}} \models \text{DOWNSTABLE} \quad \Leftrightarrow \quad \overline{\mathcal{P}}^* \models \text{DOWNSTABLE}$$

Sous hypothèse de préservation du typage, la relation de cumulativité des WCTS stables vers le bas préserve le typage :

1.2.21 Lemme
Si $\models \text{DOWNSTABLE} \wedge \text{SR}_\beta$, *alors* $\models \text{SR}_{\succcurlyeq}$.

Démonstration On doit prouver que si $\Gamma \vdash B : r$ et $A \preccurlyeq B$ alors il existe C et r' tel que $A \trianglerighteq C$, $\Gamma \vdash C : r'$ et $r' \preccurlyeq^* r$ (et donc $\Gamma \vdash C : r$). Graphiquement,



Le cas $A \equiv B$ étant évident, on montre par induction sur n que si $A \prec_n B$ alors il existe r' et C tel que $A \trianglerighteq C$, $\Gamma \vdash C : r'$ et $r' \preccurlyeq^* r$.

- Si $A \prec_0 B$, cela signifie qu'il existe deux sortes s_A et s_B telles que $(s_A, s_B) \in \mathcal{C}$, $A \trianglerighteq s_A$, et $B \trianglerighteq s_B$. D'après la préservation du typage $\models \text{SR}_\beta$, $\Gamma \vdash s_B : r$. Donc d'après le lemme d'inversion (lemme 1.1.32), $(s_B, r) \in \overline{\mathcal{A}}$. La condition DOWNSTABLE nous prouve que $(s_A, r) \in \overline{\mathcal{A}}$. Il suffit alors de prendre $C = s_A$ pour conclure.
- Supposons la propriété vraie à un rang n et supposons que $\Gamma \vdash B : r$ et $A \prec_{n+1} B$. Alors, d'après le lemme 1.2.1,
  - Soit $A \prec_n B$ et on conclut directement grâce à l'hypothèse de récurrence.
  - Soit $A \trianglerighteq \forall x : D.A'$, $B \trianglerighteq \forall x : D.B'$ et $A' \prec_n B'$. D'après la préservation du typage, $\Gamma \vdash \forall x : D.B' : r$ et donc par inversion (lemme 1.1.32), on obtient que $\Gamma \vdash D : s_D$ et que $\Gamma, x : D \vdash B' : s_{B'}$ pour $(s_D, s_{B'}, r') \in \mathcal{R}$ avec $r' \preccurlyeq^* r$. Par hypothèse de récurrence, il existe C' et $s_{C'}$ tels que $A' \trianglerighteq C'$, $\Gamma, x : D \vdash C' : s_{C'}$ et $s_{C'} \preccurlyeq^* s_{B'}$. On a donc bien $(s_D, s_{C'}, r') \in \overline{\mathcal{R}}$ d'après le lemme 1.2.16, on a $\Gamma \vdash \forall x : D.C' : r'$. On conclut donc en prenant $C = \forall x : D.C'$.☺

On déduit alors le lemme ci-dessous à partir des lemmes 1.2.21 et 1.2.9.

1.2.22 Lemme
Si $\mathcal{P} \models \text{DOWNSTABLE} \wedge \text{SR}_\beta$, *alors la règle suivante est admissible :*

$$\frac{\Gamma \vdash_{\mathcal{P}} A : B \qquad \Gamma \vdash_{\mathcal{P}} B' : s}{\Gamma \vdash_{\mathcal{P}} A : B'} \; B \preccurlyeq^* B'$$

Et donc,

$$\Gamma \vdash_{\mathcal{P}} A : B \Leftrightarrow \Gamma \vdash_{\mathcal{P}^*} A : B$$

.

1.2.5 Propriété du minorant local

Dans cette sous-section, nous étudions la propriété du minorant local, donnée par la définition ci-dessous. Elle fait partie des propriétés parmi les plus importantes de CTS, puisqu'elle implique que si un terme M admet deux types T_1 et T_2, alors il existe un type T tel que $T \preccurlyeq^* T_1$ et $T \preccurlyeq^* T_2$ (voir lemme 1.2.31). En cela, elle constitue une généralisation de la propriété de fonctionnalité (FUNCTIONAL).

1.2.23 Définition (Minorant local)
On dit que $\mathcal{P}$ a la propriété du minorant local si

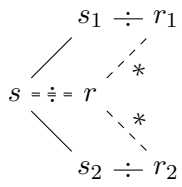
$$\left.\begin{array}{rcl} (s_1, r_1) & \in & \mathcal{A} \\ (s_2, r_2) & \in & \mathcal{A} \\ s \preccurlyeq s_1 & \wedge & s \preccurlyeq s_2 \end{array}\right\} \Rightarrow \exists r, (s, r) \in \overline{\mathcal{A}} \wedge r \preccurlyeq^* r_1 \wedge r \preccurlyeq^* r_2 \qquad \text{(MINLOC-AXIOMS)}$$

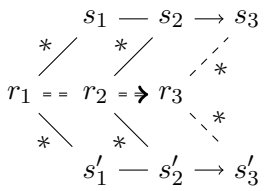
et

$$\left.\begin{array}{rcl} (s_1, s_2, s_3) & \in & \mathcal{R} \\ (s'_1, s'_2, s'_3) & \in & \mathcal{R} \\ r_1 \preccurlyeq^* s_1 & \wedge & r_1 \preccurlyeq^* s'_1 \\ r_2 \preccurlyeq^* s_2 & \wedge & r_2 \preccurlyeq^* s'_2 \end{array}\right\} \Rightarrow \exists r_3, (r_1, r_2, r_3) \in \overline{\mathcal{R}} \wedge r_3 \preccurlyeq^* s_3 \wedge r_3 \preccurlyeq^* s'_3 \qquad \text{(MINLOC-RULES)}$$

On notera la conjonction de ces deux propriétés MINLOC.

On représentera graphiquement cette définition de la façon suivante :





1.2.24 Lemme

$$\exists r, (s, r) \in \overline{\mathcal{A}} \wedge r \preccurlyeq^* r_1 \wedge r \preccurlyeq^* r_2 \quad \Leftrightarrow \quad \exists r, (s, r) \in \mathcal{A} \wedge r \preccurlyeq^* r_1 \wedge r \preccurlyeq^* r_2$$

Démonstration $\Rightarrow$: Si $(s, r) \in \overline{\mathcal{A}}$, alors par définition il existe $r' \preccurlyeq^* r$ tel que $(s, r') \in \mathcal{A}$. Et si $r \preccurlyeq^* r_1$ et $r \preccurlyeq^* r_2$, alors on a également $r' \preccurlyeq^* r_1$ et $r' \preccurlyeq^* r_2$. Et r' est un témoin valide pour prouver la conclusion.
$\Leftarrow$: Évident car $\mathcal{A} \subseteq \overline{\mathcal{A}}$. ☺

La propriété du minorant local est plus forte que la stabilité vers le bas :

1.2.25 Lemme
On a

MINLOC-AXIOMS $\Rightarrow$ DOWNSTABLE

et donc les WCTS ayant la propriété du minorant local sont stables vers le bas.

Démonstration On prouve DOWNSTABLE en prenant $s_1 = s_2$ et $r_1 = r_2$ dans MINLOC-AXIOMS.☺

1.2.26 Lemme
La condition MINLOC-AXIOMS *est équivalente aux formulations suivantes :*

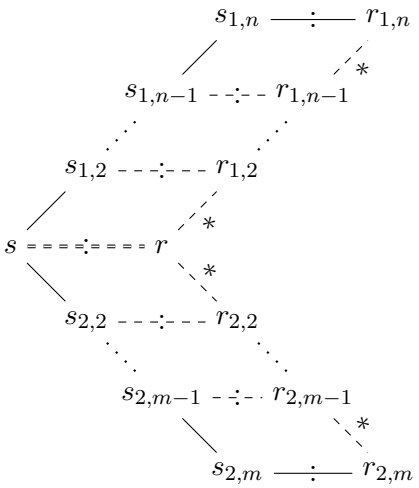
$$\left.\begin{array}{llll} (s_1, r_1) & \in & & \mathcal{A} \\ (s_2, r_2) & \in & & \mathcal{A} \\ s \preccurlyeq^* s_1 & \wedge & s \preccurlyeq^* s_2 & \end{array}\right\} \Rightarrow \exists r, (s, r) \in \overline{\mathcal{A}} \wedge r \preccurlyeq^* r_1 \wedge r \preccurlyeq^* r_2 \qquad (1)$$

$$\left.\begin{array}{llll} (s_1, r_1) & \in & & \overline{\mathcal{A}} \\ (s_2, r_2) & \in & & \overline{\mathcal{A}} \\ s \preccurlyeq s_1 & \wedge & s \preccurlyeq s_2 & \end{array}\right\} \Rightarrow \exists r, (s, r) \in \overline{\mathcal{A}} \wedge r \preccurlyeq^* r_1 \wedge r \preccurlyeq^* r_2 \qquad (2)$$

$$\left.\begin{array}{llll} (s_1, r_1) & \in & & \overline{\mathcal{A}} \\ (s_2, r_2) & \in & & \overline{\mathcal{A}} \\ s \preccurlyeq^* s_1 & \wedge & s \preccurlyeq^* s_2 & \end{array}\right\} \Rightarrow \exists r, (s, r) \in \overline{\mathcal{A}} \wedge r \preccurlyeq^* r_1 \wedge r \preccurlyeq^* r_2 \qquad (3)$$

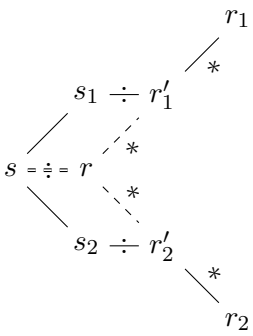
Démonstration On a trivialement $(k) \Rightarrow$ MINLOC-AXIOMS pour $k = 1, \ldots, 3$ car les prémisses des (k) sont plus générales que celle de MINLOC-AXIOMS.

– MINLOC-AXIOMS $\Rightarrow (1)$: Supposons que MINLOC-AXIOMS, $(s_1, r_1) \in \mathcal{A}$, $(s_2, r_2) \in \mathcal{A}$, $s \preccurlyeq^* s_1$ et $s \preccurlyeq^* s_2$. Alors, il existe $s = s_{1,1} \preccurlyeq s_{1,2} \preccurlyeq \cdots \preccurlyeq s_{1,n} = s_1$ et $s = s_{2,1} \preccurlyeq s_{2,2} \preccurlyeq \cdots \preccurlyeq s_{2,m} = s_2$. Posons $r_{1,n} = r_1$ et $r_{2,m} = r_2$. En invoquant $n - 2 + m - 2$ fois DOWNSTABLE (qui est vérifié d'après le lemme 1.2.25), on prouve l'existence de $r_{1,n-1}, \ldots, r_{1,2}, r_{2,2}, \ldots, r_{2,m-1}$ tels que $(s_{1,i}, r_{1,i}) \in \mathcal{A}$, $r_{1,i} \preccurlyeq^* r_{1,i+1}$ pour $2 \leq i \leq n-1$ et $(s_{2,i}, r_{2,i}) \in \mathcal{A}$, $r_{1,i} \preccurlyeq^* r_{1,i+1}$ pour $2 \leq i \leq m-1$:



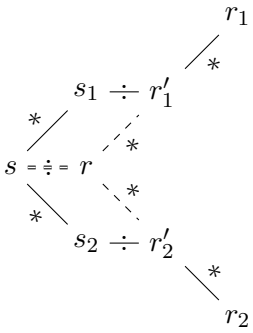
Ensuite, en invoquant MINLOC-AXIOMS, on trouve un r tel que $(s, r) \in \overline{\mathcal{A}}$, $r \preccurlyeq^* r_{1,2}$ et $r \preccurlyeq^* r_{2,2}$. On en déduit alors que $r \preccurlyeq^* r_1$ et $r \preccurlyeq^* r_2$.

– MINLOC-AXIOMS $\Rightarrow (2)$: Supposons MINLOC-AXIOMS, $(s_1, r_1) \in \overline{\mathcal{A}}$, $(s_2, r_2) \in \overline{\mathcal{A}}$, $s \preccurlyeq s_1$ et $s \preccurlyeq s_2$. Alors par définition, il existe r'_1 et r'_2 tels que $(s_1, r'_1) \in \mathcal{A}$ et $(s_2, r'_2) \in \mathcal{A}$ avec $r'_1 \preccurlyeq^* r_1$ et $r'_2 \preccurlyeq^* r_2$. Il est alors possible de trouver grâce à MINLOC-AXIOMS une sorte r telle que $(s, r) \in \overline{\mathcal{A}}$, $r \preccurlyeq^* r'_1$ et $r \preccurlyeq^* r'_2$:



Nous pouvons conclure car $r \preccurlyeq^* r'_1 \preccurlyeq^* r_1$ implique $r \preccurlyeq^* r_1$ et $r \preccurlyeq^* r'_2 \preccurlyeq^* r_2$ implique $r \preccurlyeq^* r_2$.

– MINLOC-AXIOMS $\Rightarrow$ (3) : Supposons MINLOC-AXIOMS, $(s_1, r_1) \in \overline{\mathcal{A}}$, $(s_2, r_2) \in \overline{\mathcal{A}}$, $s \preccurlyeq^* s_1$ et $s \preccurlyeq^* s_2$. Alors par définition, il existe r'_1 et r'_2 tels que $(s_1, r'_1) \in \mathcal{A}$ et $(s_2, r'_2) \in \mathcal{A}$ avec $r'_1 \preccurlyeq^* r_1$ et $r'_2 \preccurlyeq^* r_2$. On peut alors trouver grâce à (1) une sorte r telle que $(s, r) \in \overline{\mathcal{A}}$, $r \preccurlyeq^* r'_1$ et $r \preccurlyeq^* r'_2$:

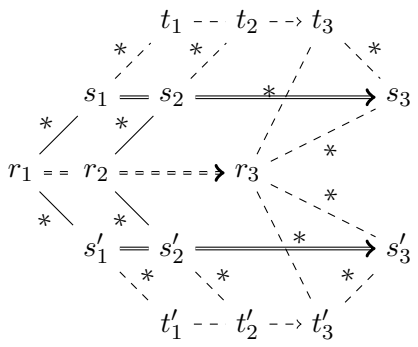


On conclut car $r \preccurlyeq^* r'_1 \preccurlyeq^* r_1$ implique $r \preccurlyeq^* r_1$ et $r \preccurlyeq^* r'_2 \preccurlyeq^* r_2$ implique $r \preccurlyeq^* r_2$.

1.2.27 Lemme
La condition MINLOC-RULES *est équivalente à :*

$$\left.\begin{array}{rcl} (s_1, s_2, s_3) & \in & \overline{\mathcal{R}} \\ (s'_1, s'_2, s'_3) & \in & \overline{\mathcal{R}} \\ r_1 \preccurlyeq^* s_1 & \wedge & r_1 \preccurlyeq^* s'_1 \\ r_2 \preccurlyeq^* s_2 & \wedge & r_2 \preccurlyeq^* s'_2 \end{array}\right\} \Rightarrow \exists r_3, (r_1, r_2, r_3) \in \overline{\mathcal{R}} \wedge r_3 \preccurlyeq^* s_3 \wedge r_3 \preccurlyeq^* s'_3 \qquad (*)$$

Démonstration On a trivialement $(*) \Rightarrow$ MINLOC-RULES car la prémisse de $(*)$ est plus générale que celle de MINLOC-RULES. On montre $(*) \Leftarrow$ MINLOC-RULES, supposons pour cela MINLOC-RULES. Soit $(s_1, s_2, s_3) \in \overline{\mathcal{R}}$, $(s'_1, s'_2, s'_3) \in \overline{\mathcal{R}}$, $r_1 \preccurlyeq^* s_1$, $r_1 \preccurlyeq^* s'_1$, $r_2 \preccurlyeq^* s_2$, $r_2 \preccurlyeq^* s'_2$. Par définition, il existe donc $(t_1, t_2, t_3) \in \mathcal{R}$ et $(t'_1, t'_2, t'_3) \in \mathcal{R}$ avec $s_1 \preccurlyeq^* t_1$, $s_2 \preccurlyeq^* t_2$, $t_3 \preccurlyeq^* s_3$, $s'_1 \preccurlyeq^* t'_1$, $s'_2 \preccurlyeq^* t'_2$ et $t'_3 \preccurlyeq^* s'_3$. Ensuite d'après MINLOC-RULES, il existe une sorte r_3 telle que $(r_1, r_2, r_3) \in \overline{\mathcal{R}}$, $r_3 \preccurlyeq^* t_3$ et $r_3 \preccurlyeq^* t'_3$. Or on conclut en remarquant que $r_3 \preccurlyeq^* t_3 \preccurlyeq^* s_3$ implique $r_3 \preccurlyeq^* s_3$ et que $r'_3 \preccurlyeq^* t'_3 \preccurlyeq^* s'_3$ implique $r_3 \preccurlyeq^* s'_3$:



☺

On déduit des lemmes 1.2.26 et 1.2.27 que :

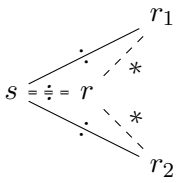
1.2.28 Lemme

$$\mathcal{P} \models \text{MINLOC} \quad \Leftrightarrow \quad \mathcal{P}^* \models \text{MINLOC} \quad \Leftrightarrow \quad \overline{\mathcal{P}} \models \text{MINLOC} \quad \Leftrightarrow \quad \overline{\mathcal{P}}^* \models \text{MINLOC}$$

1.2.29 Lemme
Si $\models$ MINLOC, *on a*

$$(s, r_1), (s, r_2) \in \mathcal{A} \Rightarrow \exists r, (s, r) \in \overline{\mathcal{A}} \wedge r \preccurlyeq^* r_1 \wedge r \preccurlyeq^* r_2$$

Ce qui donne graphiquement,



Démonstration Il suffit de prendre $s_1 = s_2 = s$ dans (1). ☺

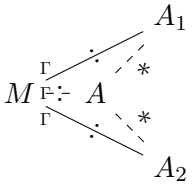
1.2.30 Lemme
Si $\mathcal{P} \models$ PTS, *alors*

$$\mathcal{P} \models \text{MINLOC} \qquad \Leftrightarrow \qquad \mathcal{P} \models \text{FUNCTIONAL}$$

Démonstration C'est immédiat une fois que l'on a constaté que $s \preccurlyeq^* r \Leftrightarrow s = r$ dans un PTS. ☺

On en déduit alors la généralisation du lemme 1.1.10 suivante :

1.2.31 Lemme (Existence d'un type minorant)
Si $\models$ MINLOC $\wedge$ SR$_\beta$ *et si* $\Gamma \vdash M : A_1$ *et* $\Gamma \vdash M : A_2$, *il existe* A *tel que* $\Gamma \vdash M : A$ *et* $A \preccurlyeq^* A_1$ *et* $A \preccurlyeq^* A_2$. *Graphiquement,*



Démonstration On procède par induction sur la structure de M,

- $M = x$: Par inversion (lemme 1.1.32), on sait que Γ est de la forme $\Gamma = \Gamma_1, x : A, \Gamma_2$ avec $A \preccurlyeq^* A_1$ et $A \preccurlyeq^* A_2$. Et on a bien $\Gamma \vdash x : A$.
- $M = s$: Par inversion, on sait qu'il existe r_1 telle que $(s, r_1) \in \mathcal{A}$, $(s, r_2) \in \mathcal{A}$ et $r_1 \preccurlyeq^* A_1$ et $r_2 \preccurlyeq^* A_2$ et comme on a $\models$ MINLOC-AXIOMS, on en déduit qu'il existe r telle que $(s, r) \in \mathcal{A}$, $r \preccurlyeq^* r_1$ et $r \preccurlyeq^* r_2$. Donc il suffit de prendre $A = r$.
- $M = \forall x : P.Q$: Par inversion, on sait qu'il existe $(s_1, r_1, t_1) \in \mathcal{R}$ et $(s_2, r_2, t_2) \in \mathcal{R}$ telles que :

$$\begin{array}{cc} \Gamma \vdash P : s_1 & \Gamma \vdash P : s_2 \\ \Gamma, x : P \vdash Q : r_1 & \Gamma, x : P \vdash Q : r_2 \\ t_1 \preccurlyeq^* A_1 & t_2 \preccurlyeq^* A_2 \end{array}$$

  Or par hypothèse d'induction, on sait qu'il existe S et R tels que $S \preccurlyeq^* s_1$, $S \preccurlyeq^* s_2$, $R \preccurlyeq^* r_1$, $R \preccurlyeq^* r_2$, $\Gamma \vdash P : S$ et $\Gamma, x : P \vdash Q : R$. On montre facilement que nécessairement, $S \trianglerighteq s$ et $R \trianglerighteq r$ pour deux sortes s et r. Or comme $\models$ SR$_\beta$, on a $\Gamma \vdash P : s$ et $\Gamma, x : P \vdash Q : r$. Enfin, puisque l'on a $\models$ MINLOC-RULES, on en déduit qu'il existe une sorte t telle que $(s, r, t) \in \overline{\mathcal{R}}$ avec $t \preccurlyeq^* t_1$ et $t \preccurlyeq^* t_2$ et donc que $\Gamma \vdash \forall x : P.Q : t$ (lemme 1.2.16). On conclut donc en prenant $A = t$.
- $M = U\,V$: Par inversion, on sait qu'il existe B_1, C_1, B_2, C_2 tels que

$$\begin{array}{cc} \Gamma \vdash U : \forall y : B_1.C_1 & \Gamma \vdash V : B_1 \\ \Gamma \vdash U : \forall y : B_2.C_2 & \Gamma \vdash V : B_2 \end{array}$$

  Par hypothèse d'induction, on obtient un D tel que $D \preccurlyeq^* \forall y : B_1.C_1$, $D \preccurlyeq^* \forall y : B_2.C_2$ et $\Gamma \vdash U : D$. D'après le lemme 1.2.2, il existe B et C tel que $D \trianglerighteq \forall x : B.C$, $B_1 \trianglerighteq B$, $B_2 \equiv B$, $C \preccurlyeq^* C_1$ et $C \preccurlyeq^* C_2$. Comme on a $\models$ SR$_\beta$, on a donc $\Gamma \vdash U : \forall x : B.C$ et $\Gamma \vdash V : B$. Et donc on a bien $\Gamma \vdash U\,V : B[V/x]$. Posons donc $A = B[V/x]$, alors le lemme 1.1.6 nous permet de conclure que $A \preccurlyeq^* A_1$ et $A \preccurlyeq^* A_2$.
- $M = \lambda x : D.N$: Par inversion, on sait qu'il existe B_1, s_1 et B_2, s_2 tels que :

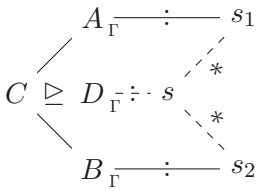
$$\begin{array}{cc} \Gamma, x : D \vdash N : B_1 & \Gamma \vdash \forall x : D.B_1 : s_1 \quad (1) \\ \Gamma, x : D \vdash N : B_2 & \Gamma \vdash \forall x : D.B_2 : s_2 \end{array}$$

  Par hypothèse d'induction, on en déduit qu'il existe B tel que $\Gamma, x : D \vdash N : B$, $B \preccurlyeq^* B_1$ et $B \preccurlyeq^* B_2$. Par inversion de (1), on obtient qu'il existe $(t, r, s) \in \mathcal{R}$ tel que $s \preccurlyeq^* s_1$ et $\Gamma \vdash D : t$, $\Gamma, x : D \vdash B_1 : r$. Or d'après le lemme 1.2.21, on obtient qu'il existe B' tel que $B \trianglerighteq B'$ et $\Gamma, x : D \vdash B' : r$ (2). On en conclut donc que $\Gamma \vdash \forall x : D.B' : t$ grâce à la règle Produit. Enfin d'après le lemme de réduction du type (lemme 1.1.47), on en déduit $\Gamma, x : D \vdash N : B'$ (3). On en conclut grâce à (3), (2) et la règle Abstraction que $\Gamma \vdash \lambda x : N.B' : \forall x : D.B'$ et on a bien $\forall x : D.B' \preccurlyeq \forall x : D.B_1$ et $\forall x : D.B' \preccurlyeq \forall x : D.B_2$. ☺

1.2.32 Lemme
Si $\models$ MINLOC $\wedge$ $\models$ SR$_\beta$,

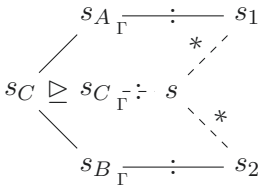
$$\begin{cases} C \preccurlyeq A \ \wedge \Gamma \vdash A : s_1 \\ C \preccurlyeq B \wedge \Gamma \vdash B : s_2 \end{cases}$$

alors il existe D *et* s *tels que* $C \trianglerighteq D$, *et* $\Gamma \vdash D : s$, $s \preccurlyeq^* s_1$ *et* $s \preccurlyeq^* s_2$. *Graphiquement,*



Démonstration On montre le résultat pour $C \prec_n A$ par induction sur n,

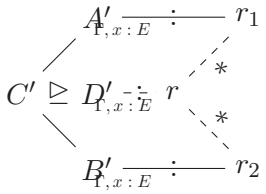
– Si $C \prec_0 A$, il existe par définition s_C et s_A tels que $C \equiv s_C$, $A \equiv s_A$ et $(s_C, s_A) \in \mathcal{C}$. D'après le lemme 1.2.5, on a également $C \prec_0 B$, $B \equiv s_B$ et $(s_C, s_B) \in \mathcal{C}$. Par confluence, on a $C \trianglerighteq s_C$, $A \trianglerighteq s_A$ et $B \trianglerighteq s_B$. Puis par préservation du typage ($\models \mathrm{SR}_\beta$), on a $\Gamma \vdash s_A : s_1$ et $\Gamma \vdash s_B : s_2$. Enfin, d'après MINLOC-AXIOMS, il existe s tel que (s_C, s) et $s \preccurlyeq^* s_A$ et $s \preccurlyeq^* s_B$.



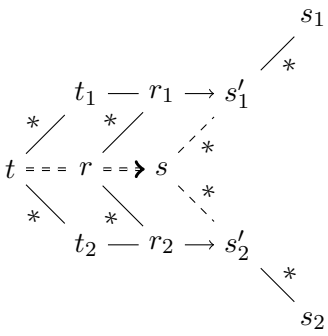
– Si $C \prec_{n+1} A$, d'après les lemmes 1.2.1 et 1.2.5, on peut se ramener à l'un des deux cas suivants :
  – Soit $C \prec_n A$ et $C \prec_n B$, et on conclut par hypothèse d'induction.
  – Soit il existe E, A', B' vérifiant $C \trianglerighteq \forall x : E.C'$ $A \trianglerighteq \forall x : E.A'$ $B \trianglerighteq \forall x : E.B'$ et tels que $C' \prec_n A'$ et $C' \prec_n B'$. Ensuite par préservation du typage ($\models \mathrm{SR}_\beta$) on en déduit que $\Gamma \vdash \forall x : E.A' : s_1$, $\Gamma \vdash \forall x : E.B' : s_2$. Et par inversion (lemme 1.1.32), on déduit l'existence de $(t_1, r_1, s_1') \in \mathcal{R}$ et de $(t_2, r_2, s_2') \in \mathcal{R}$ telles que :

$$\Gamma \vdash E : t_1 \tag{1}$$
$$\Gamma \vdash E : t_2 \tag{2}$$
$$\Gamma, x : E \vdash A' : r_1 \tag{3}$$
$$\Gamma, x : E \vdash B' : r_2 \tag{4}$$
$$s_1' \preccurlyeq^* s_1 \tag{5}$$
$$s_2' \preccurlyeq^* s_2 \tag{6}$$

D'après le lemme 1.2.31, de (1) et (2) on déduit qu'il existe une sorte t vérifiant $\Gamma \vdash E : t$ et vérifiant $t \preccurlyeq^* t_1$ et $t \preccurlyeq^* t_2$. Ensuite par hypothèse d'induction, de (3) et (4) on obtient l'existence de D' et r vérifiant $\Gamma, x : E \vdash D' : r$ tels que $C' \trianglerighteq D'$ et $r \preccurlyeq^* r_1$ et $r \preccurlyeq^* r_2$. On est alors dans la situation suivante :



Puis comme nous sommes dans un WCTS ayant la propriété du minorant local, on peut trouver une sorte s



Pour conclure, on pose $D = \forall x : E.D'$ et on a bien $\Gamma \vdash D : s$, $C \trianglerighteq D$, $s \preccurlyeq^* s_1$ et $s \preccurlyeq^* s_2$. ☺

1.2.6 Autres notions de stabilité

Nous présentons ici les autres notions de stabilité. Elles sont moins intéressantes que la stabilité vers le bas (à cause des règles de cumulativité qui orientent le système), mais elles pourront se révéler utiles.

1.2.33 Définition (Faible stabilité vers le haut et vers le bas, stabilité vers le haut)

1. On dira qu'un WCTS est faiblement stable vers le bas s'il vérifie

$$\left.\begin{array}{c}(s,r) \in \mathcal{A} \\ s' \preccurlyeq^* s\end{array}\right\} \Rightarrow \exists r', (s', r') \in \mathcal{A}$$

   Et on notera cette propriété WEAK-DOWNSTABLE.

2. On dira qu'un WCTS est stable vers le haut pour les axiomes s'il vérifie

$$\left.\begin{array}{c}(s,r) \in \overline{\mathcal{A}} \\ s \preccurlyeq^* s'\end{array}\right\} \Rightarrow \exists r' \succcurlyeq^* r, (s', r') \in \overline{\mathcal{A}}$$

   Et on notera cette propriété UPSTABLE-AXIOMS.

3. On dira qu'un WCTS est stable vers le haut pour les règles s'il vérifie

$$\left.\begin{array}{rcl}(s_1, s_2, s_3) & \in & \overline{\mathcal{R}} \\ s_1 & \preccurlyeq^* & s'_1 \\ s_2 & \preccurlyeq^* & s'_2\end{array}\right\} \quad \Rightarrow \quad \exists s'_3 \succcurlyeq^* s_3, (s'_1, s'_2, s'_3) \in \overline{\mathcal{R}}$$

   Et on notera cette propriété UPSTABLE-RULES.

4. On dira qu'un WCTS est stable vers le haut s'il vérifie UPSTABLE-AXIOMS et UPSTABLE-RULES. On notera UPSTABLE la conjonction de ces deux propriétés.

5. On dira qu'un WCTS est faiblement stable vers le haut pour les axiomes s'il vérifie

$$\left.\begin{array}{c}(s,r)\in\overline{\mathcal{A}}\\ s\preccurlyeq^* s'\end{array}\right\}\Rightarrow\exists r',(s',r')\in\mathcal{A}$$

   Et on notera cette propriété WEAK-UPSTABLE-AXIOMS.

6. On dira qu'un WCTS est faiblement stable vers le haut pour les règles s'il vérifie

$$\left.\begin{array}{rcl}(s_1,s_2,s_3)&\in&\overline{\mathcal{R}}\\ s_1&\preccurlyeq^*&s_1'\\ s_2&\preccurlyeq^*&s_2'\end{array}\right\}\quad\Rightarrow\quad\exists s_3',(s_1',s_2',s_3')\in\overline{\mathcal{R}}$$

   Et on notera cette propriété WEAK-UPSTABLE-RULES.

7. On dira qu'un WCTS est faiblement stable vers le haut s'il vérifie WEAK-UPSTABLE-AXIOMS et WEAK-UPSTABLE-RULES. On notera WEAK-UPSTABLE la conjonction de ces deux propriétés.

On dispose des implications ci-dessous.

1.2.34 Lemme

$$\begin{aligned}
\models \text{DOWNSTABLE} &\Rightarrow \models \text{WEAK-DOWNSTABLE} && (1)\\
\models \text{UPSTABLE} &\Rightarrow \models \text{WEAK-UPSTABLE} && (2)\\
\models \text{DOWNSTABLE}\wedge\text{SR}_\beta &\Rightarrow \models \text{SR}_{\succcurlyeq} && (3)\\
\models \text{UPSTABLE}\wedge\text{SR}_\beta &\Rightarrow \models \text{SR}_{\preccurlyeq} && (4)\\
\models \text{WEAK-DOWNSTABLE} &\not\Rightarrow \models \text{WEAK-SR}_{\succcurlyeq} && (5)\\
\models \text{WEAK-UPSTABLE} &\not\Rightarrow \models \text{WEAK-SR}_{\preccurlyeq} && (6)
\end{aligned}$$

Démonstration 1. Immédiat.

2. Immédiat.
3. Lemme 1.2.21.
4. Supposons $\models$ UPSTABLE $\wedge$ SR_β. Il suffit de montrer que si $\Gamma\vdash A:r$ et $A\preccurlyeq B$ alors il existe C et r' tel que $B\trianglerighteq C$, $\Gamma\vdash C:r'$ et $r\preccurlyeq^* r'$.

   Le cas $A\equiv B$ étant trivial, on montre par induction sur n que si $A\prec_n B$ alors il existe r' et C tel que $B\trianglerighteq C$, $\Gamma\vdash C:r'$ et $r\preccurlyeq^* r'$.
   - Si $A\prec_0 B$, cela signifie qu'il existe deux sortes s_A et s_B telles que $(s_A,s_B)\in\mathcal{C}$, $A\trianglerighteq s_A$, et $B\trianglerighteq s_B$. D'après la préservation du typage $\models$ SR_β, $\Gamma\vdash s_A:r$. Donc d'après le lemme d'inversion (lemme 1.1.32), $(s_A,r)\in\overline{\mathcal{A}}$. La condition UPSTABLE-AXIOMS nous prouve qu'il existe $r'\succcurlyeq^* r$ telle que $(s_B,r')\in\overline{\mathcal{A}}$. Il suffit alors de prendre $C=s_B$ pour conclure.
   - Supposons la propriété vraie à un rang n et supposons que $\Gamma\vdash A:r$ et $A\prec_{n+1}B$. Alors, d'après le lemme 1.2.1,
     - Soit $A\prec_n B$ et on conclut directement grâce à l'hypothèse de récurrence.

- Soit $A \trianglerighteq \forall x : D.A'$, $B \trianglerighteq \forall x : D.B'$ et $A' \prec_n B'$. D'après la préservation du typage, $\Gamma \vdash \forall x : D.A' : r$ et donc par inversion (lemme 1.1.32), on obtient qu'il existe s_D et $s_{A'}$ telles que $(s_D, s_{A'}, r) \in \overline{\mathcal{R}}$, $\Gamma \vdash D : s_D$ et que $\Gamma, x : D \vdash A' : s_{A'}$. Par hypothèse de récurrence, il existe C' et $s_{C'}$ tels que $B' \trianglerighteq C'$, $\Gamma, x : D \vdash C' : s_{C'}$ et $s_{A'} \preccurlyeq^* s_{C'}$. Donc la propriété UPSTABLE-RULES, prouve qu'il existe $r' \succcurlyeq^* r$ telle que $(s_D, s_{C'}, r') \in \overline{\mathcal{R}}$. Donc d'après le lemme 1.2.16, on a $\Gamma \vdash \forall x : D.C' : r'$. On conclut donc en prenant $C = \forall x : D.C'$. ☺

5. Soit $\mathcal{P}$ le CTS égal à

$$(\{\star_1, \Box_1, \star_2, \Box_2, \circ, \triangle\}, \{(\star_1, \Box_1), (\star_2, \Box_2), (\circ, \triangle)\}, \{(\triangle, \Box_2, \Box_2)\}, \{(\star_1, \star_2)\})$$

   On peut vérifier facilement que $\mathcal{P} \models$ WEAK-DOWNSTABLE, or on a $\vdash_{\mathcal{P}} \circ \to \star_2 : \Box_2$ et $\circ \to \star_1 \preccurlyeq \circ \to \star_2$ et pourtant il n'existe pas de sorte s telle que $\vdash_{\mathcal{P}} \circ \to \star_1 : s$.

6. Symétriquement, soit $\mathcal{P}$ le CTS égal à

$$(\{\star_1, \Box_1, \star_2, \Box_2, \circ, \triangle\}, \{(\star_1, \Box_1), (\star_2, \Box_2), (\circ, \triangle)\}, \{(\triangle, \Box_1, \Box_1)\}, \{(\star_1, \star_2)\})$$

   On peut vérifier facilement que $\mathcal{P} \models$ WEAK-UPSTABLE, or on a $\vdash_{\mathcal{P}} \circ \to \star_1 : \Box_2$ et $\circ \to \star_1 \preccurlyeq \circ \to \star_2$ et pourtant il n'existe pas de sorte s telle que $\vdash_{\mathcal{P}} \circ \to \star_2 : s$.

Les implications (5) et (6) nous montrent qu'on ne peut pas déduire la préservation de la typabilité en la vérifiant simplement sur les paramètres. On dispose néanmoins du lemme suivant qui se révélera utile dans la sous-section 1.2.8 :

1.2.35 Lemme

$$\models \text{SR}_\beta \wedge \text{MINLOC} \wedge \text{WEAK-UPSTABLE} \quad \Rightarrow \quad \models \text{WEAK-SR}_{\preccurlyeq}$$

Démonstration Supposons $\text{SR}_\beta \wedge \text{MINLOC} \wedge \text{WEAK-UPSTABLE}$ et $\Gamma \vdash A : s$.
Montrons que si $A \preccurlyeq B$ alors il existe C tel que $B \trianglerighteq C$ et $\text{WF}_\Gamma(C)$.

- Si $A \equiv B$, alors d'après le lemme 1.1.3, il existe C tel que $A \trianglerighteq C$ et $B \trianglerighteq C$. D'après la préservation du typage ($\models \text{SR}_\beta$), $\Gamma \vdash C : s$ et donc $\text{WF}_\Gamma(C)$.
- Montrons par induction sur n que si $A \prec_n B$, alors il existe C tel que $B \trianglerighteq C$ et $\text{WF}_\Gamma(C)$.
  - Si $A \preccurlyeq_0 B$, alors par définition, il existe s_A et s_B telle que $s_A \preccurlyeq s_B$, $A \trianglerighteq s_A$ et $B \trianglerighteq s_B$. D'après la préservation du typage, on a $\Gamma \vdash s_A : s$ et comme on a WEAK-UPSTABLE on sait qu'il existe r tel que $\Gamma \vdash s_B : r$. Donc on peut conclure en prenant $C = s_B$ et on a bien $B \trianglerighteq C$ et $\text{WF}_\Gamma(C)$.
  - Si $A \preccurlyeq_{n+1} B$, alors d'après le lemme 1.2.1, alors
    - Soit $A \preccurlyeq_n B$ et on conclut par hypothèse de récurrence,
    - Soit il existe A', B' et D tel que $A \trianglerighteq \forall x : D.A'$, $B \trianglerighteq \forall x : D.B'$ et $A' \preccurlyeq_n B'$. Par préservation du typage, on a $\Gamma \vdash \forall x : D.A' : s$ et par inversion (lemme 1.1.32), on en déduit qu'il existe $(s_1, s_2, s_3) \in \mathcal{R}$ telle que $s_3 \preccurlyeq^* s$, $\Gamma \vdash D : s_1$ (1) et $\Gamma, x : D \vdash A' : s_2$. On a donc $\text{WF}_{\Gamma, x:D}(A')$ et par hypothèse de récurrence, on sait qu'il existe C' tel que $B' \trianglerighteq C'$ et $\text{WF}_{\Gamma, x:D}(C')$.
      Montrons maintenant qu'il existe r_2 telle que $\Gamma, x : D \vdash C' : r_2$ (2). Comme $\text{WF}_{\Gamma, x:D}(C')$, deux cas sont possibles :
      - Soit il existe une sorte r_2 telle que $\Gamma, x : D \vdash C' : r_2$.

– Soit C' est une sorte $s_{C'}$, et comme $A' \preccurlyeq_n C'$, on en déduit qu'il existe une sorte $s_{A'}$ telle que $A' \trianglerighteq s_{A'}$. Et par préservation du typage, on a $\Gamma, x : D \vdash s_{A'} : s_2$, et d'après WEAK-UPSTABLE-AXIOMS, on en déduit qu'il existe une sorte r_2 telle que $\Gamma, x : D \vdash s_{C'} : r_2$.

Comme on a $\models$ DOWNSTABLE (car on a la propriété plus forte $\models$ MINLOC), le lemme 1.2.21 et $A' \preccurlyeq^* C'$, montrent qu'il existe $s_2' \preccurlyeq^* r_2$ et A'' tels que $A' \trianglerighteq A''$ et $\Gamma, x : D \vdash A'' : s_2'$. Or par préservation du typage, on a également $\Gamma, x : D \vdash A'' : s_2$ et $\models$ MINLOC et le lemme 1.2.31, prouvent qu'il existe t_2 tel que $\Gamma, x : D \vdash A'' : t_2$ et $t_2 \preccurlyeq^* s_2'$ et $t_2 \preccurlyeq^* s_2$. On a donc bien $(s_1, t_2, s_3) \in \overline{\mathcal{R}}$ et $t_2 \preccurlyeq^* r_2$. On peut donc appliquer la propriété WEAK-UPSTABLE-RULES pour prouver qu'il existe r telle que $(s_1, r_2, r) \in \overline{\mathcal{R}}$ (3). Et donc le lemme 1.2.16, (1), (2) et (3) prouve que $\Gamma \vdash \forall x : D.C' : r$. On conclut donc en prenant en $C = \forall x : D.C'$. ☺

On en déduit le critère suivant pour prouver WEAK-SR$_{\preccurlyeq}$ ∧ WEAK-SR$_{\succcurlyeq}$:

1.2.36 Lemme

$$\models \text{SR}_\beta \wedge \text{MINLOC} \wedge \text{WEAK-UPSTABLE} \quad \Rightarrow \quad \models \text{WEAK-SR}_{\preccurlyeq} \wedge \text{WEAK-SR}_{\succcurlyeq}$$

Démonstration On a $\models$ MINLOC $\Rightarrow\models$ DOWNSTABLE $\Rightarrow\models$ SR$_{\succcurlyeq}$ $\Rightarrow\models$ WEAK-SR$_{\succcurlyeq}$ et le lemme précédent prouve WEAK-SR$_{\preccurlyeq}$. ☺

On peut donc utiliser ce critère et le théorème 1.1.56 pour obtenir un critère de renforcement :

1.2.37 Lemme
Si $\models$ NICE-TOPSORT, *alors*

$$\models \text{CTS} \wedge \text{MINLOC} \wedge \text{WEAK-UPSTABLE} \quad \Rightarrow \quad \models \text{STRENGTHENING}$$

1.2.7 Sur l'anti-symétrie de la cumulativité

On remarquera que jusqu'à présent nous n'avons pas contraint la relation de cumulativité à être anti-symétrique. Dans un WCTS, si on a à la fois $s \preccurlyeq^* r$ et $r \preccurlyeq^* s$, alors s et r sont "observationnellement équivalents" dans le sens où tous les types qui habitent s habitent également r et réciproquement.

Soit $\sim$ la relation d'équivalence définie par $s \sim r$ si et seulement si $s \preccurlyeq^* r \wedge r \preccurlyeq^* s$. On notera $\tilde{\cdot} : \mathcal{S} \longrightarrow \mathcal{S}/\sim$ la fonction qui à une sorte s associe sa classe d'équivalence $\tilde{s}$. Soit $\mathcal{P}$ un WCTS, on définit le WCTS $\mathcal{P}/\sim$ par :

$$\begin{aligned}
\mathcal{S}_{\mathcal{P}/\sim} &= (\mathcal{S}_{\mathcal{P}})/\sim \\
\mathcal{A}_{\mathcal{P}/\sim} &= \{(\tilde{s}, \tilde{r}) | (s, r) \in \mathcal{A}_{\mathcal{P}}\} \\
\mathcal{R}_{\mathcal{P}/\sim} &= \{(\tilde{s_1}, \tilde{s_2}, \tilde{s_3}) | (s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}}\} \\
\mathcal{C}_{\mathcal{P}/\sim} &= \{(\tilde{s}, \tilde{r}) | (s, r) \in \mathcal{C}_{\mathcal{P}}\}
\end{aligned}$$

Clairement $\preccurlyeq^*_{\mathcal{P}/\sim}$ est une relation d'ordre stricte. La fonction $\tilde{\cdot}$ est par construction un morphisme $\mathcal{P} \hookrightarrow_{\tilde{\cdot}} \mathcal{P}/\sim$. On en déduit donc que :

1.2.38 Lemme
Si $\Gamma \vdash_{\mathcal{P}} M : A$, *alors* $\tilde{\Gamma} \vdash_{\mathcal{P}/\sim} \tilde{M} : \tilde{A}$.

Réciproquement, on a :

1.2.39 Lemme
Si $\mathcal{P} \models$ DOWNSTABLE $\vee$ UPSTABLE-AXIOMS, *alors*

$$(\tilde{s}, \tilde{r}) \in \mathcal{A}_{\mathcal{P}/\sim} \Rightarrow (s, r) \in \overline{\mathcal{A}_{\mathcal{P}}}$$

Démonstration Par définition $(\tilde{s}, \tilde{r}) \in \mathcal{A}_{\mathcal{P}/\sim}$ signifie qu'il existe $(s', r') \in \mathcal{A}_{\mathcal{P}}$ telles que $s \sim s'$ et $r \sim r'$. On a donc en particulier $r' \preccurlyeq^* r$, $s \preccurlyeq^* s'$ et $s' \preccurlyeq^* s$. D'après UPSTABLE-AXIOMS ou DOWNSTABLE (avec le lemme 1.2.19), on en déduit que $(s, r') \in \overline{\mathcal{A}_{\mathcal{P}}}$. Et comme $r' \preccurlyeq^* r$, on a également $(s, r) \in \overline{\mathcal{A}_{\mathcal{P}}}$. ☺

1.2.40 Lemme

$$(\tilde{s_1}, \tilde{s_2}, \tilde{s_3}) \in \mathcal{R}_{\mathcal{P}/\sim} \Rightarrow (s_1, s_2, s_3) \in \overline{\mathcal{R}_{\mathcal{P}}}$$

Démonstration Par définition $(\tilde{s_1}, \tilde{s_2}, \tilde{s_3}) \in \mathcal{R}_{\mathcal{P}/\sim}$ signifie qu'il existe $(s'_1, s'_2, s'_3) \in \mathcal{R}$ telles que $s_1 \sim s'_1 \wedge s_2 \sim s'_2 \wedge s_3 \sim s'_3$. Et donc en particulier, on a $s_1 \preccurlyeq^* s'_1 \wedge s_2 \preccurlyeq^* s'_2 \wedge s'_3 \preccurlyeq^* s_3$ et donc $(s_1, s_2, s_3) \in \overline{\mathcal{R}_{\mathcal{P}}}$.☺

Fixons une fonction $\mathbf{repr} : {}^{\mathcal{S}}/_{\sim} \longrightarrow \mathcal{S}$ qui fournit un représentant canonique de chaque classe, c'est-à-dire telle que pour toute sorte s, $\mathbf{repr}(\tilde{s}) \sim s$. On déduit alors des lemmes 1.2.39 et 1.2.40, le lemme suivant :

1.2.41 Lemme
Si $\models$ DOWNSTABLE $\vee$ UPSTABLE-AXIOMS, *alors* **repr** *est un morphisme :* ${}^{\mathcal{P}}/_{\sim} \hookrightarrow_{\mathbf{repr}} \overline{\mathcal{P}}$ *et*

$$\Gamma \vdash_{\mathcal{P}/\sim} M : A \qquad \Rightarrow \qquad \mathbf{repr}(\Gamma) \vdash_{\mathcal{P}} \mathbf{repr}(M) : \mathbf{repr}(A)$$

Il est important de remarquer que ne nous n'avons pas :

$$\tilde{\Gamma} \vdash_{\mathcal{P}/\sim} \tilde{M} : \tilde{A} \qquad \Rightarrow \Gamma \vdash_{\mathcal{P}} M : A$$

et qu'ainsi le choix de fixer en amont une notion de représentant canonique était important.

En effet, soit le CTS suivant :

$$\begin{aligned} \mathcal{S} &= \{\star, \star', \Box, \circ\} \\ \mathcal{A} &= \{(\star, \Box), (\star', \Box), (\Box, \circ)\} \\ \mathcal{R} &= \{(\circ, \circ, \circ)\} \\ \mathcal{C} &= \{(\star, \star'), (\star', \star)\} \end{aligned}$$

Nous avons $\star \sim \star'$ et

$$X : \tilde{\Box} \to \tilde{\Box}, h : X\, \tilde{\star} \vdash_{\mathcal{P}/\sim} h : X\, \tilde{\star'}$$

mais nous n'avons pas

$$X : \Box \to \Box, h : X\, \star \vdash_{\mathcal{P}/\sim} h : X\, \star'$$

Nous voyons donc que $\mathcal{P}$ et ${}^{\mathcal{P}}/_{\sim}$ ne typent pas exactement les mêmes termes. Il ne serait donc pas exclu que les relations de cumulativité, qui ne sont pas anti-symétriques, puissent avoir une utilité bien que nous n'en connaissions aucune à ce jour.

1.2.8 Un théorème de renforcement

Dans cette sous-section, nous allons prouver un théorème de renforcement suffisant pour tous les WCTS qui nous intéresseront.

1.2.42 Théorème (Renforcement)
Si $\models$ SR$_\beta$ *et si*

$$\Gamma \vdash A_1 : s_1 \quad \wedge \quad \Gamma, x : A_1 \vdash A_2 : s_2 \quad \wedge \quad \Gamma \vdash \forall x : A_1.A_2 : s \qquad \Rightarrow \qquad \exists s_3, (s_1, s_2, s_3) \in \overline{\mathcal{R}}$$

alors on a $\models$ STRENGTHENING.

Démonstration La preuve est une généralisation aux WCTS de celle que l'on trouve pour ECC dans la thèse de Luo [Luo90]. On commence par remarquer que pour prouver STRENGTHENING, il suffit de prouver l'assertion plus faible suivante.

$$\text{Si } \Gamma, x : A, \Delta \vdash M : B \text{ et } x \notin \mathcal{FV}(\Delta), \mathcal{FV}(M), \text{ alors il existe } B' \preccurlyeq^* B \text{ et } \Gamma, \Delta \vdash M : B'. \ (*)$$

En effet, supposons que $\Gamma, x : A, \Delta \vdash M : B$, $x \notin \mathcal{FV}(\Delta), \mathcal{FV}(M), \mathcal{FV}(B)$ et qu'il existe $B' \preccurlyeq^* B$ (1) tel que $\Gamma, \Delta \vdash M : B'$ (2). Alors d'après le lemme 1.2.22 pour prouver $\Gamma, \Delta \vdash M : B$ il suffit de prouver que $\mathrm{WF}_{\Gamma,\Delta}(B)$. Or $\Gamma, x : A, \Delta \vdash M : B$ implique $\mathrm{WF}_{\Gamma,x:A,\Delta}(B)$ (lemme 1.1.35). On distingue alors deux cas :

- Soit B est une sorte, (1), (2) et la règle Cumulativité-Sortes nous permet de conclure que $\Gamma, \Delta \vdash M : B$.
- Soit il existe s tel que $\Gamma, x : A, \Delta \vdash B : s$ et dans ce cas, nous pouvons invoquer $(*)$ pour en déduire que $\Gamma, \Delta \vdash B : s$ et conclure à l'aide de (1), (2) et la règle Cumulativité que $\Gamma, \Delta \vdash M : B$.

L'absence de condition sur les variables de B dans (*) est essentielle, c'est ce qui nous permettra de traiter le cas de la règle Cumulativité. On prouve donc $(*)$ par induction sur la dérivation de $\Gamma, x : A, \Delta \vdash M : B$. Supposons que $x \notin \mathcal{FV}(\Delta), \mathcal{FV}(M)$:

- Axiome : Ce cas est impossible car $\Gamma, x : A, \Delta$ est nécessairement non vide.
- Variable : Comme $x \notin \mathcal{FV}(M)$, le seul cas à traiter est le cas où Δ est de la forme $\Delta', y : B$ et $M = y$. Alors on a $\Gamma, x : A, \Delta' \vdash B : s$ et par hypothèse d'induction il existe un S' tel que $S' \preccurlyeq^* s$ et $\Gamma, \Delta' \vdash B : S'$. On en déduit qu'il existe une sorte s' tel que $S' \trianglerighteq s'$ (lemme 1.2.4). Et donc d'après le lemme 1.1.47 et la préservation du typage, on a $\Gamma, \Delta' \vdash B : s'$. Et on conclut que $\Gamma, \Delta \vdash M : B$ en utilisant la règle Variable (le B' qui convient est donc dans ce cas égal à B).
- Affaiblissement : On distingue les deux cas possibles :
  - Soit $\Delta = \langle\rangle$ et on a $\Gamma \vdash M : B$ et $\Gamma \vdash A : s$. Dans ce cas on obtient directement $\Gamma, \Delta \vdash M : B$ (le B' qui convient est donc dans ce cas égal à B).
  - Soit $\Delta = \Delta', y : C$ et on a $\Gamma, x : A, \Delta' \vdash M : B$ et $\Gamma, x : A, \Delta' \vdash C : s$. Alors par hypothèse de récurrence, on sait qu'il existe B' tel que $B' \preccurlyeq^* B$, et $\Gamma, \Delta' \vdash M : B'$ et $S' \preccurlyeq s$ telle que $\Gamma, \Delta' \vdash C : S'$. On en déduit qu'il existe une sorte s' telle que $S' \trianglerighteq s'$. Et donc d'après le lemme 1.1.47 et la préservation du typage, on a $\Gamma, \Delta' \vdash C : s'$. On conclut ainsi que $\Gamma, \Delta \vdash M : B'$ en utilisant la règle Affaiblissement.

– Abstraction : Dans ce cas M est de la forme $\lambda y : B_1.N$ et B de la forme $\forall y : B_1.B_2$ avec :

$$\Gamma, x : A, \Delta, y : B_1 \vdash N : B_2 \qquad \text{et} \qquad \Gamma, x : A, \Delta \vdash \forall y : B_1.B_2 : s \quad (1)$$

Par hypothèse d'induction, en en déduit qu'il existe $B_2' \preccurlyeq^* B_2$ tel que $\Gamma, \Delta, y : B_1 \vdash N : B_2'$ (2). On remarque qu'on ne peut pas appliquer l'hypothèse d'induction à la seconde prémisse car on ne peut pas prouver que $x \notin \mathcal{FV}(\forall y : B_1.B_2)$. Mais d'après (1) et (2), le fait que B_2 ne peut pas être une sorte maximale (par inversion de (1)), les lemmes 1.1.26 et 1.1.35, on montre facilement qu'il existe s_1, s_2 telles que $\Gamma, \Delta \vdash B_1 : s_1$ et $\Gamma, \Delta, y : B_1 \vdash B_2' : s_2$. Et donc, le lemme 1.1.24 nous donne $\Gamma, x : A, \Delta \vdash B_1 : s_1$ et $\Gamma, x : A, \Delta, y : B_1 \vdash B_2' : s_2$. Or, d'après l'hypothèse du théorème, on conclut qu'il existe s_3, tel que $(s_1, s_2, s_3) \in \overline{\mathcal{R}}$ et donc le lemme 1.2.16, nous permet de conclure que $\Gamma, \Delta \vdash \forall y : B_1.B_2 : s_3$. On peut donc conclure avec la règle Abstraction que $\Gamma, \Delta \vdash \lambda y : B_1.N : \forall y : B_1.B_2'$ et on a bien $\forall y : B_1.B_2' \preccurlyeq^* \forall y : B_1.B_2$.

– Application : Dans ce cas M est de la forme $M_1\, M_2$ et B de la forme $B_2[M_2/y]$ avec :

$$\Gamma, x : A, \Delta \vdash M_1 : \forall y : B_1.B_2 \quad \text{et} \quad \Gamma, x : A, \Delta \vdash M_2 : B_1$$

Par hypothèse d'induction, on en déduit qu'il existe $C \preccurlyeq^* \forall y : B_1.B_2$, $B_1'' \preccurlyeq^* B_1$ tels que $\Gamma, \Delta \vdash M_1 : C$ et $\Gamma, \Delta \vdash M_2 : B_1''$. D'après le lemme 1.2.2 et le lemme 1.1.3, Il existe B_1' et B_2' tel que $C \trianglerighteq \forall y : B_1'.B_2'$ tel que $B_1 \trianglerighteq B_1'$ et $B_1'' \trianglerighteq B_1'$. On a donc $\forall y : B_1'.B_2' \preccurlyeq^* \forall y : B_1.B_2$ et $B_1' \preccurlyeq^* B_1$. Grâce à la préservation du typage et grâce au lemme 1.1.47, on a $\Gamma, \Delta \vdash M_1 : \forall y : B_1'.B_2'$ et $\Gamma, \Delta \vdash M_2 : B_1'$. D'après la règle Application, on en déduit $\Gamma, \Delta \vdash M_1\, M_2 : B_2'[M_2/y]$ et on a bien $B_2'[M_2/y] \preccurlyeq^* B_2[M_2/y]$.

– Produit : Dans ce cas, M est de la forme $\forall y : M_1.M_2$, B est une sorte s_3 et on a

$$\Gamma, x : A, \Delta \vdash M_1 : s_1 \quad \text{et} \quad \Gamma, x : A, \Delta, y : M_1 \vdash M_2 : s_2$$

avec $(s_1, s_2, s_3) \in \mathcal{R}$. Par hypothèse d'induction on arrive à la conclusion qu'il existe $S_1' \preccurlyeq^* s_1$ et $S_2' \preccurlyeq^* s_2$ tels que $\Gamma, \Delta \vdash M_1 : S_1'$ et $\Gamma, \Delta, y : M_1 \vdash M_2 : S_2'$. On en déduit qu'il existe s_1' et s_2' tels que $S_1' \trianglerighteq s_1'$, $S_2' \trianglerighteq s_2$, $s_1' \preccurlyeq^* s_1$ et $s_2' \preccurlyeq^* s_2$. On a donc $(s_1', s_2', s_3) \in \overline{\mathcal{R}}$. Or, la préservation du typage et le lemme 1.1.47, nous donnent alors $\Gamma, \Delta \vdash M_1 : s_1'$ et $\Gamma, \Delta, y : M_1 \vdash M_2 : s_2'$. Ainsi le lemme 1.2.16, nous prouve $\Gamma, \Delta \vdash \forall y : M_1.M_2 : s_3$ et l'on prend $B' = B = s_3$.

– Cumulativité et Cumulativité-Sortes : Dans ce cas, on a $\Gamma, x : A, \Delta \vdash M : C$ avec $C \preccurlyeq B$ et par hypothèse d'induction, on sait qu'il existe $B' \preccurlyeq^* C$ tel que $\Gamma, \Delta \vdash M : B'$. Or ce B' convient car on a bien $B' \preccurlyeq^* B$. ☺

1.2.43 Lemme
Si $\models$ SR$_\beta \wedge$ MINLOC $\wedge$ WEAK-UPSTABLE-RULES, *alors*

$$\Gamma \vdash A_1 : s_1 \ \wedge\ \Gamma, x : A_1 \vdash A_2 : s_2 \ \wedge\ \Gamma \vdash \forall x : A_1.A_2 : s \quad \Rightarrow \quad \exists s_3, (s_1, s_2, s_3) \in \overline{\mathcal{R}}$$

Démonstration Supposons

$$\Gamma \vdash A_1 : s_1 \quad (1)$$
$$\Gamma, x : A_1 \vdash A_2 : s_2 \quad (2)$$
$$\Gamma \vdash \forall x : A_1.A_2 : s \quad (3)$$

Alors par inversion de (3) (lemme 1.1.32), on sait qu'il existe $(s'_1, s'_2, s'_3) \in \mathcal{R}$ telles que $s'_3 \preccurlyeq^* s_3$ et :

$$\Gamma \vdash A_1 : s'_1 \quad (1')$$
$$\Gamma, x : A_1 \vdash A_2 : s'_2 \quad (2')$$

Le lemme 1.2.31 appliqué à $(1), (1')$ et à $(2), (2')$ nous montre l'existence de r_1 et r_2 telles que

$$\begin{cases} r_1 \preccurlyeq^* s_1 & (3) \\ r_1 \preccurlyeq^* s'_1 \end{cases} \quad \text{et} \quad \begin{cases} r_2 \preccurlyeq^* s_2 & (4) \\ r_2 \preccurlyeq^* s'_2 \end{cases}$$

Et, puisque l'on a $\models$ MINLOC, on en déduit qu'il existe r_3 telle que $(r_1, r_2, r_3) \in \overline{\mathcal{R}}$ (5) (et $r_3 \preccurlyeq^* s'_3$). Et donc $\models$ WEAK-UPSTABLE-RULES, (3), (4) et (5) implique qu'il existe s_3 telle que $(s_1, s_2, s_3) \in \overline{\mathcal{R}}$ (6). ☺

On en déduit un critère pour appliquer le théorème 1.2.42 :

1.2.44 Lemme (Critère de renforcement)

$$\text{SR}_\beta \wedge \text{MINLOC} \wedge \text{WEAK-UPSTABLE-RULES} \quad \Rightarrow \quad \text{STRENGTHENING}$$

et

$$\text{CTS} \wedge \text{MINLOC} \wedge \text{WEAK-UPSTABLE-RULES} \quad \Rightarrow \quad \text{STRENGTHENING}$$

On peut alors montrer que théorème 1.2.42 n'est pas un cas particulier du théorème 1.1.56. En effet, le CTS défini par

$$\begin{aligned} \mathcal{S} &= \{\star, \star', \Box, \circ, \circledcirc\} \\ \mathcal{A} &= \{(\star, \Box), (\circ, \circledcirc)\} \\ \mathcal{R} &= \{(\circledcirc, \Box, \Box)\} \\ \mathcal{C} &= \{(\star, \star')\} \end{aligned}$$

ne vérifie pas la condition du théorème 1.1.56 car si nous avons $\text{WF}_{\langle\rangle}(\circ \to \star)$ nous n'avons pas en revanche $\text{WF}_{\langle\rangle}(\circ \to \star')$. Et pourtant il vérifie bien SR_β (car c'est un CTS, lemme 1.1.42), MINLOC et WEAK-UPSTABLE-RULES. Mais ceci est probablement un artefact dû au fait que la présentation de Jiménez ne comporte pas la règle Cumulativité-Sortes et que notre preuve l'utilise pour l'exemple ci-dessus. On remarquera également (pour les même raisons) que le critère sur les paramètres, proposé ci-dessus, est plus général que le lemme 1.2.37.

1.2.9 Types principaux

Dans cette sous-section, nous généralisons les théorèmes de Luo [Luo90] afin de prouver le Théorème 1.2.48.

1.2.45 Lemme
Si $\mathcal{C}$ est un ordre strict, alors $\prec$ est un ordre strict.

Démonstration
- transitivité : traité par le lemme 1.2.6.
- irréflexivité : Par induction sur n, on montre que $\prec_n$ est irréflexif :
  - $\prec_0$ est irréflexif :
    $A \prec_0 A$ implique $A \equiv s$ et $A \equiv s'$ tel que $(s, s') \in \mathcal{C}$ or la confluence implique que nécessairement $s = s'$, ce qui contredit l'irréflexivité de $\mathcal{C}$.
  - $\prec_{n+1}$ est irréflexif sachant que $\prec_n$ irréflexif :
    Si on a $A \prec_{n+1} A$, alors par définition :
    - Soit $A \prec_n A$: impossible car $\prec_n$ est irréflexif.
    - Soit il existe A', A'' et B tels que $A \equiv \forall x : B.A' \equiv \forall x : B.A''$ et $A' \prec_n A''$ or par confluence on obtient que $A' \equiv A''$ et donc $A' \prec_n A'$ ce qui est également impossible.
- anti-symétrie : Si $A \prec B$ et $B \prec A$ alors il existe n et m tels que $A \prec_n B$ et $B \prec_m A$ et d'après le lemme 1.2.5, $A \prec_{\min(n,m)} B$ et $B \prec_{\min(n,m)} A$. Il suffit donc de montrer que pour tout n, $\prec_n$ est anti-symétrique. On procède par induction sur n :
  - $\prec_0$ est anti-symétrique :
    $$A \prec_0 B \wedge B \prec_0 A \Rightarrow \left(\begin{cases} A \equiv s_A \\ B \equiv s_B \\ (s_A, s_B) \in \mathcal{C} \end{cases} \text{ et } \begin{cases} A \equiv s'_A \\ B \equiv s'_B \\ (s'_B, s'_A) \in \mathcal{C} \end{cases} \right)$$
    or par confluence on obtient que $s_A = s'_A$ et $s_B = s'_B$, ce qui contredit l'anti-symétrie de $\mathcal{C}$.
  - $\prec_{n+1}$ est anti-symétrique sachant que $\prec_n$ l'est : Par définition $A \prec_{n+1} B \wedge B \prec_{n+1} A$ implique que
    - Soit $A \prec_n B$ et le lemme 1.2.5 impliquent $B \prec_n A$ et on conclut par hypothèse d'induction.
    - Soit $B \prec_n A$ et on conclut de la même manière.
    - Soit
      $$\begin{cases} A \equiv \forall x : C_1.A_1 \\ B \equiv \forall x : C_1.B_1 \\ A_1 \prec_n B_1 \end{cases} \text{ et } \begin{cases} B \equiv \forall x : C_2.B_2 \\ A \equiv \forall x : C_2.A_2 \\ B_2 \prec_n A_2 \end{cases}$$
      or par confluence on obtient que $A_1 \equiv A_2$ et $B_1 \equiv B_2$ et donc $B_1 \prec_n A_1$, ce qui contredit l'anti-symétrie de $\prec_n$. ☺

1.2.46 Lemme
si $\mathcal{C}$ est un ordre strict bien fondé, alors $\prec$ l'est également.

Démonstration Montrons tout d'abord par induction sur n que $\prec_n$ est bien fondée.
- Le cas de base est trivial car $\prec_0 = \mathcal{C}$.
- Supposons $\mathcal{C}$ et $\prec_n$ bien fondées, montrons que $\prec_{n+1}$ l'est également. On procède par l'absurde : supposons qu'il existe une suite $A_0 \succ_{n+1} A_1 \succ_{n+1} \cdots$ infinie décroissante pour $\succ_{n+1}$. Deux cas sont alors possibles (d'après le lemme 1.2.5 et la définition de $\prec_{n+1}$) :
  - Soit on a $A_0 \succ_n A_1 \succ_n \cdots$ ce qui contredit la bonne fondation de $\prec_n$.
  - Soit il existe $A'_0, A'_1, \ldots$ et B tels que $A_0 \equiv \forall x : C.A'_0$, $A_1 \equiv \forall x : C.A'_1, \ldots$ et $A'_0 \succ_n A'_1 \succ_n \cdots$. Ce qui contredit également la bonne fondation de $\prec_n$.

Donc pour tout n, $\prec_n$ est bien fondée (*).

Supposons maintenant qu'il existe une suite $A_0 \succ A_1 \succ \cdots$ infinie décroissante pour $\succ$. Si $A_0 \succ A_1$, cela signifie qu'il existe un entier k tel que $A_0 \succ_k A_1$ et donc d'après le lemme 1.2.5, $A_0 \succ_k A_1 \succ_k \cdots$ ce qui contredit (*). ☺

1.2.47 Définition (CTS principal)
On appelle CTS principal, un CTS qui a la propriété du minorant local et dont la relation de cumulativité est bien fondée. On note cette propriété PRINCIPAL.

1.2.48 Théorème
Soit $\mathcal{P}$ un CTS principal. Soient M et Γ tels que $\Gamma \vdash M : A$, alors il existe C tel que $\Gamma \vdash M : C$ et pour tout B, $\Gamma \vdash M : B$ si et seulement si $C \preccurlyeq B$ et $\mathrm{WF}_\Gamma(B)$. Un tel C est appelé un type principal *de M (dans le contexte Γ).*

Démonstration Soit $\mathcal{T}_{\Gamma,M} = \{B | \Gamma \vdash M : B\}$ l'ensemble des types de M dans le contexte Γ. Et soit C un élément minimal de $\mathcal{T}_{\Gamma,M}$ pour $\prec$ (il existe car $\prec$ est bien fondée et $\mathcal{T}_{\Gamma,M}$ est non vide).

$\Rightarrow$: Soit B tel que $\Gamma \vdash M : B$. On a $\mathrm{WF}_\Gamma(B)$ et on sait qu'il existe D tel que $\Gamma \vdash M : D$ et $D \preccurlyeq B$ et $D \preccurlyeq C$ (lemme 1.2.31). Comme C est minimal, on a nécessairement $D \equiv C$ et donc $C \preccurlyeq B$.

$\Leftarrow$: Réciproquement, soit $C \preccurlyeq B$ tel que $\mathrm{WF}_\Gamma(B)$, alors d'après lemme 1.1.28, on a $\Gamma \vdash M : B$. ☺

1.2.49 Lemme (Unicité des types principaux modulo $\equiv$)
Dans un CTS principal, si A et B sont deux types principaux d'un même terme, alors nécessairement, $A \equiv B$.

Démonstration Comme A et B sont deux types principaux, on a $A \preccurlyeq B$ et $B \preccurlyeq A$. D'après le lemme 1.2.45, on en déduit que $A \equiv B$. ☺

1.2.10 Inférence du type principal

Dans cette sous-section, nous étudierons un algorithme d'inférence du type principal. L'existence d'un tel algorithme est essentielle si l'on souhaite utiliser les termes bien-typés comme des témoins *effectifs* de l'existence d'une dérivation.

On distingue les trois problèmes suivants :

1. La *vérification de type* : étant donnés Γ, M et A, a-t-on $\Gamma \vdash M : A$?
2. L'*inférence d'un type* : étant donnés Γ, M, existe-t-il A tel que $\Gamma \vdash M : A$?
3. L'*inférence d'un type principal* : étant donnés Γ et M, existe-t-il un type principal A de M dans le contexte Γ ?

Dans les assistants de preuves où les propositions sont représentées par des types (comme en Coq par exemple), la vérification de type permet à l'assistant de preuve de vérifier qu'un terme est bien une preuve correcte d'une proposition.

L'inférence de type est surtout utile dans les systèmes de programmation où l'existence d'une dérivation de typage implique que le programme considéré aura certaines bonnes propriétés opérationnelles : par exemple, la normalisation ou l'absence de programmes dégénérés qui se réduiraient sur un terme en forme normale et qui ne représentent pas une valeur.

Enfin, nous verrons que l'inférence du type principal est un moyen de résoudre le problème de vérification de type dans les CTS principaux.

Il nous faudra, avant tout, supposer la condition suivante d'effectivité du système que l'on cherchera à implémenter :

1.2.50 Définition (WCTS décidable)
On dira qu'un WCTS est décidable s'il est faiblement normalisant et si les tests d'appartenance aux ensembles $\mathcal{A}$, $\mathcal{R}$, et $\mathcal{C}$ sont décidables (on supposera qu'on dispose également d'une représentation concrète des sortes en machine). On notera cette propriété DECIDABLE.

On en déduit alors la décidabilité des relations $\equiv$ et $\preccurlyeq$:

1.2.51 Lemme (Décidabilité de $\equiv$)
Si $\models$ WEAKLY-NORMALISING *et si* $\mathrm{WF}_\Gamma(A)$ *et* $\mathrm{WF}_\Gamma(B)$*, alors le test* $A \equiv B$ *est décidable.*

Démonstration D'après le lemme 1.1.36, on sait que A et B sont faiblement normalisants. On peut alors utiliser une stratégie de réduction en appel par nom dont il est facile de montrer qu'elle terminera sur une forme normale. Et d'après le lemme 1.1.3, on en déduit qu'il suffit de comparer les formes normales pour décider si $A \equiv B$. ☺

1.2.52 Lemme (Décidabilité de $\preccurlyeq$)
Dans un WCTS décidable, si $\mathrm{WF}_\Gamma(A)$ *et* $\mathrm{WF}_\Gamma(B)$ *le test* $A \preccurlyeq^* B$ *est décidable.*

Démonstration D'après le lemme 1.1.36, on sait que A et B sont faiblement normalisants. On remarque que si A et B sont en forme normale, alors $A \preccurlyeq^* B$ si et seulement si

- Soit $A = B$,
- Soit A est de la forme $\forall x_1 : C_1, \ldots, x_n : C_n.s_A$ et B est de la forme $\forall x_1 : C_1, \ldots, x_n : C_n.s_B$ avec $s_A \preccurlyeq^* s_B$.

Donc, pour tester si $A \preccurlyeq^* B$ il suffit de normaliser A et de normaliser B puis de vérifier que soit les formes normales sont égales, soit elles sont toutes les deux une imbrication de n produits de forme respective $\forall x_1 : C_1, \ldots, x_n : C_n.s_A$ et $\forall x_1 : C_1, \ldots, x_n : C_n.s_B$ avec $s_A \preccurlyeq^* s_B$ (qui est décidable par hypothèse). ☺

1.2.53 Définition (princ-sorte$(\cdot)$ et princ-regle$(\cdot,\cdot)$)
Dans un CTS principal, on notera **princ-sorte**(s) la plus petite (au sens de $\preccurlyeq$) sorte r telle que $(s, r) \in \mathcal{A}$, en d'autres termes **princ-sorte**(s) est le type principal de s et on posera par convention **princ-sorte**$(s) = \lightning$ si une telle sorte n'existe pas (pour les sortes maximales par exemple). Et on notera **princ-regle**(s_1, s_2) la plus petite sorte s_3 telle que $(s_1, s_2, s_3) \in \overline{\mathcal{R}}$.

Nous avons clairement :

1.2.54 Lemme
Si $\mathcal{P}$ *est décidable,* **princ-sorte**$(\cdot)$ *et* **princ-regle**$(\cdot,\cdot)$ *s'implémentent effectivement.*

Dans CTS principal et décidable, considérons l'algorithme récursif ci-dessous qui infère le type principal de son argument (sous hypothèse, comme on le verra plus avant, que tous les produits sont autorisés) :

$$
\begin{aligned}
\mathbf{infer}_\Gamma(x) &= \begin{cases} \mathrm{NF}(A) \text{ si } x : A \in \Gamma \\ \lightning \text{ sinon} \end{cases} \\
\mathbf{infer}_\Gamma(s) &= \mathbf{princ\text{-}sorte}(s) \\
\mathbf{infer}_\Gamma(\lambda x : A.M) &= \begin{cases} \forall x : \mathrm{NF}(A).\, \mathbf{infer}_{\Gamma,x:A}(M) \text{ si } \mathbf{infer}_\Gamma(A) \in \mathcal{S} \\ \lightning \text{ sinon} \end{cases} \\
\mathbf{infer}_\Gamma(M\,N) &= \begin{cases} \mathrm{NF}(B[N/x]) \text{ si } \mathbf{infer}_\Gamma(M) = \forall x : A.B, \text{ et } \mathbf{infer}_\Gamma(N) \preccurlyeq A \\ \lightning \text{ sinon} \end{cases} \\
\mathbf{infer}_\Gamma(\forall x : A.B) &= \begin{cases} \mathbf{princ\text{-}regle}(s_1, s_2) \text{ si } s_1 = \mathbf{infer}_\Gamma(A) \text{ et } s_2 = \mathbf{infer}_{\Gamma,x:A}(B) \\ \lightning \text{ sinon} \end{cases}
\end{aligned}
$$

où $\lightning$ se comporte comme un résultat exceptionnel : si l'un des appels récursifs retourne $\lightning$ alors le résultat est également $\lightning$. Notons qu'il ne peut pas il y avoir une infinité d'appels récursifs car ceux-ci ont toujours lieu sur des termes strictement plus petits. Il suffira donc de montrer l'existence des formes normales dans chaque cas pour être convaincu de la terminaison de l'algorithme.

Par inspection des cas possibles, on a clairement :

1.2.55 Lemme
Si $\mathbf{infer}_\Gamma(M)$ *termine et* $\mathbf{infer}_\Gamma(M) \neq \lightning$, *alors* $\mathbf{infer}_\Gamma(M)$ *est en forme normale.*

Cet algorithme cherche à inférer le type principal de son argument par induction structurelle. Dans le cas, abstraction il vérifie que A est bien un type ; il s'assure pour cela que le type A est une sorte. On notera qu'il ne vérifie pas que le type est autorisé. C'est pourquoi, nous allons avoir besoin de la condition FULL, définie ci-dessous, afin de prouver la correction.

1.2.56 Définition (WCTS complet)
On dit qu'un WCTS est complet si pour tout couple de sortes s_1, s_2 il existe une sorte s_3 telle que $(s_1, s_2, s_3) \in \overline{\mathcal{R}}$. On note cette propriété FULL.

On montre aisément :

1.2.57 Lemme
Si $\models$ FULL, *alors* $\Gamma \vdash A : s_1$ *et* $\Gamma, x : A \vdash B : s_2$ *implique* $\mathrm{WF}_\Gamma(\forall x : A.B)$.

Le lemme ci-dessous énonce la correction de l'algorithme d'inférence : si $\mathbf{infer}_\Gamma(M)$ termine sans retourner d'erreur, alors $\mathbf{infer}_\Gamma(M)$ est un type principal de M dans le contexte Γ. De surcroît, le lemme précédent et le lemme 1.2.49 montrent qu'il est l'unique type principal en forme normale.

1.2.58 Lemme (Correction de l'inférence du type principal)
Supposons $\models$ DECIDABLE $\wedge$ PRINCIPAL $\wedge$ FULL, *et soit* Γ *tel que* $\mathrm{WF}(\Gamma)$ *et* M *un terme quelconque. Alors* $\mathbf{infer}_\Gamma(M)$ *termine. Et de plus,*

$$\mathbf{infer}_\Gamma(M) \neq \lightning \quad \Rightarrow \quad \Gamma \vdash M : \mathbf{infer}_\Gamma(M)$$

Démonstration On procède par induction sur la taille de M, on justifiera dans chaque cas l'existence des formes normales pour prouver la terminaison de l'algorithme :

- $M = s$: Comme on a $\mathbf{infer}_\Gamma(M) = \mathbf{princ\text{-}sorte}(s) \neq \lightning$, on a $(s, \mathbf{princ\text{-}sorte}(s)) \in \mathcal{A}$. Et d'après le lemme 1.1.31 et $\mathrm{WF}(\Gamma)$, on a $\Gamma \vdash s : \mathbf{princ\text{-}sorte}(s)$.
- $M = x$: Si $\mathbf{infer}_\Gamma(x) \neq \lightning$ cela signifie que Γ est de la forme $\Gamma_1, x : B, \Gamma_2$ et que $\mathbf{infer}_\Gamma(x) = \mathrm{NF}(B)$. Notons que $\mathrm{NF}(B)$ existe car $\mathrm{WF}(\Gamma) \Rightarrow \mathrm{WF}_{\Gamma_1}(B)$ (lemme 1.1.26) et le lemme 1.1.36 prouve que B est normalisant. D'après le lemme 1.1.33 et $\mathrm{WF}(\Gamma)$, on a $\Gamma \vdash x : B$ et par réduction du type (lemme 1.1.47), on a $\Gamma \vdash x : \mathrm{NF}(B)$.
- $M = M_1\, M_2$: Si $\mathbf{infer}_\Gamma(M) \neq \lightning$ cela signifie que $\mathbf{infer}_\Gamma(M_1) = \forall x : B_1.B_2 \neq \lightning$ et que $\mathbf{infer}_\Gamma(N) = C \neq \lightning$ avec $C \preccurlyeq B_1$. Par hypothèse de récurrence, on a $\Gamma \vdash M_1 : \forall x : B_1.B_2$ et $\Gamma \vdash M_2 : C$. D'après le lemme 1.1.34, on en conclut que $\mathrm{WF}_\Gamma(B_1)$. Et ainsi, d'après le lemme 1.1.28 et $C \preccurlyeq B_1$, on a $\Gamma \vdash M_2 : B_1$. On conclut alors en utilisant la règle Application que $\Gamma \vdash M_1\, M_2 : B_2[M_2/x]$. Par voix de conséquence, d'après le lemme 1.1.35, on en déduit que $\mathrm{WF}_\Gamma(B_2[M_2/x])$ et donc $\mathrm{NF}(B_2[M_2/x])$ existe (lemme 1.1.36). On a alors $\mathbf{infer}_\Gamma(M) = \mathrm{NF}(B_2[M_2/x])$. Par réduction du type, on a $\Gamma \vdash M_1\, M_2 : \mathrm{NF}(B_2[M_2/x])$.
- $M = \lambda x : B.N$: Si $\mathbf{infer}_\Gamma(M) \neq \lightning$ cela signifie qu'il existe C tel que $\mathbf{infer}_{\Gamma, x:B}(N) = C \neq \lightning$ et qu'il existe une sorte s telle que $\mathbf{infer}_\Gamma(B) = s$. Par hypothèse de récurrence, on $\Gamma \vdash B : \mathbf{infer}_\Gamma(B)$ et en en déduit $\mathrm{WF}(\Gamma, x : B)$ et l'existence de $\mathrm{NF}(B)$ (lemme 1.1.36). On a donc bien $\mathbf{infer}_\Gamma(M) = \forall x : \mathrm{NF}(B).C$. Par hypothèse de récurrence (permise car on a montré $\mathrm{WF}(\Gamma, x : B)$) on a $\Gamma, x : B \vdash N : C$ et comme on a $\models$ FULL, on a également $\mathrm{WF}_\Gamma(\forall x : B.C)$. Et donc $\Gamma \vdash \lambda x : B.N : \forall x : B.C$ d'après la règle Abstraction. Et par réduction du type, on a $\Gamma \vdash \lambda x : B.N : \forall x : \mathrm{NF}(B).C$.
- $M = \forall x : B_1.B_2$: Si $\mathbf{infer}_\Gamma(M) \neq \lightning$ cela signifie qu'il existe $(s_1, s_2, s_3) \in \overline{\mathcal{R}}$ avec s_3 minimale telles que $\mathbf{infer}_\Gamma(B_1) = s_1$ et $\mathbf{infer}_{\Gamma, x:B_1}(B_2) = s_2$. Nous avons également $\mathbf{infer}_\Gamma(M) = s_3$. Par hypothèse d'induction, on a $\Gamma \vdash B_1 : s_1$, on en déduit donc que $\mathrm{WF}(\Gamma, x : B_1)$. Et on peut alors appliquer une seconde fois l'hypothèse de récurrence pour prouver que $\Gamma, x : B_1 \vdash B_2 : \mathbf{infer}_{\Gamma, x:B_1}(B_2)$ et donc que $\Gamma, x : B_1 \vdash B_2 : s_2$. On en conclut alors $\Gamma \vdash \forall x : B_1.B_2 : s_3$ (lemme 1.2.16). ☺

L'hypothèse de FULL n'est pas nécessaire pour prouver la complétude de l'algorithme :

1.2.59 Lemme (Complétude de l'inférence du type principal)
Supposons $\mathcal{P} \models$ DECIDABLE $\wedge$ PRINCIPAL.
Si $\Gamma \vdash M : A$, *alors*

1. $\mathbf{infer}_\Gamma(M)$ *termine et* $\mathbf{infer}_\Gamma(M) \neq \lightning$,
2. $\Gamma \vdash M : \mathbf{infer}_\Gamma(M)$,
3. $\mathbf{infer}_\Gamma(M) \preccurlyeq A$.

Démonstration Supposons $\Gamma \vdash M : A$ (*). On procède par induction sur la taille de M :

- $M = s$: Par inversion de (*) (lemme 1.1.32), on sait qu'il existe $r \preccurlyeq A$, telle que $(s, r) \in \mathcal{A}$. On a donc bien $\mathbf{princ\text{-}sorte}(s) \neq \lightning$, on a $(s, \mathbf{princ\text{-}sorte}(s)) \in \mathcal{A}$. On en déduit donc 1., 2. et 3. immédiatement.

– $M = x$: Par inversion de (*), on sait que Γ est de la forme $\Gamma_1, x : B, \Gamma_2$ avec $B \preccurlyeq A$.

1. On a bien $\mathbf{infer}_\Gamma(M) = \mathrm{NF}(B) \neq \lightning$ ($\mathrm{NF}(B)$ existe d'après le lemme 1.1.36 et $\mathrm{WF}_{\Gamma_1}(B)$).
2. D'après le lemme 1.1.33 et $\mathrm{WF}(\Gamma)$ (par définition de WF), on a $\Gamma \vdash x : B$. Et par réduction du type (lemme 1.1.47), on en conclut que $\Gamma \vdash x : \mathrm{NF}(B)$.
3. On a $B \preccurlyeq A$ donc $\mathrm{NF}(B) \preccurlyeq A$.

– $M = M_1\, M_2$: Par inversion de (*), on sait qu'il existe B et C tels que $\Gamma \vdash M_1 : \forall x : B.C$, $\Gamma \vdash M_2 : B$ et $C[M_2/x] \preccurlyeq A$. Par hypothèse de récurrence, on en déduit que :

$$\begin{aligned} \mathbf{infer}_\Gamma(M_1) &\neq \lightning \\ \Gamma \vdash M_1 : \mathbf{infer}_\Gamma(M_1) &\qquad (1) \\ \mathbf{infer}_\Gamma(M_1) &\preccurlyeq \forall x : B.C \qquad (2) \end{aligned}$$

De même, on a :

$$\begin{aligned} \mathbf{infer}_\Gamma(M_2) &\neq \lightning \\ \Gamma \vdash M_2 : \mathbf{infer}_\Gamma(M_2) &\qquad (3) \\ \mathbf{infer}_\Gamma(M_2) &\preccurlyeq B \qquad (4). \end{aligned}$$

D'après le lemme 1.2.55, $\mathbf{infer}_\Gamma(M_1)$ est en forme normale donc le lemme 1.2.1 et (2) nous montrent que $\mathbf{infer}_\Gamma(M_1)$ est bien de la forme $\forall x : B'.C'$ avec $B \trianglerighteq B'$ et $C' \preccurlyeq C$. Et par compatibilité de la cumulativité (lemme 1.1.7), on en déduit que $\mathbf{infer}_\Gamma(M_2) \preccurlyeq B'$. Ensuite, par réduction du type, on prouve $\Gamma \vdash M_2 : B'$ (5). La règle Application, (1) et (5) montrent que $\Gamma \vdash M : C'[M_2/x]$ (6) et donc $\mathrm{WF}_\Gamma(C'[M_2/x])$ (lemme 1.1.35). On en déduit alors que $\mathrm{NF}(C'[M_2/x])$ existe.

1. Dans ce cas, on a $\mathbf{infer}_\Gamma(M) = \mathrm{NF}(C'[M_2/x]) \neq \lightning$.
2. Par réduction du type dans (6), on obtient $\Gamma \vdash M : \mathrm{NF}(C'[M_2/x])$.
3. Le lemme 1.1.35 appliqué à (*), prouve que $\mathrm{WF}_\Gamma(A)$ et lemme 1.1.6, montre que $\mathrm{NF}(C'[M_2/x]) \equiv C'[M_2/x] \preccurlyeq C[M_2/x] \preccurlyeq A$.

– $M = \lambda x : B.N$: Par inversion de (*), on en déduit qu'il existe C et s tels que $\Gamma, x : B \vdash N : C$ (1), $\Gamma \vdash \forall x : B.C : s$ (2), et $\forall x : B.C \preccurlyeq A$ (3). Ensuite par inversion de (2), on en déduit qu'il existe $(s_1, s_2, s_3) \in \mathcal{R}$ telles que $\Gamma \vdash B : s_1$ ($2'$) et $\Gamma, x : B \vdash C : s_2$ ($2''$) (et $s_3 \preccurlyeq s$). Par hypothèse de récurrence, on a $\mathbf{infer}_{\Gamma,x:B}(N) \neq \lightning$, $\mathbf{infer}_{\Gamma,x:B}(N) \preccurlyeq C$ (4), $\Gamma, x : B \vdash N : \mathbf{infer}_{\Gamma,x:B}(N)$ (5). Toujours par hypothèse de récurrence, on a $\mathbf{infer}_\Gamma(B) \neq \lightning$, $\Gamma \vdash B : \mathbf{infer}_\Gamma(B)$ (6) et $\mathbf{infer}_\Gamma(B) \preccurlyeq s_1$ (7).

1. On sait que $\mathbf{infer}_\Gamma(B)$ est en forme normale (lemme 1.2.55), on déduit donc de (7) (et de la confluence) que $\mathbf{infer}_\Gamma(B) \in \mathcal{S}$. Or (6) et $\models$ WEAKLY-NORMALISING, ce qui implique que $\mathrm{NF}(B)$ existe. On a alors $\mathbf{infer}_\Gamma(\lambda x : B.N) = \forall x : \mathrm{NF}(B).\, \mathbf{infer}_{\Gamma,x:B}(N) \neq \lightning$.

2. D'après ($2''$), (4) et le lemme 1.2.21, on sait qu'il existe D tel que $\mathbf{infer}_{\Gamma,x:B}(N) \trianglerighteq D$ et $\Gamma, x : B \vdash D : s_2$. Or $\mathbf{infer}_{\Gamma,x:B}(N)$ est en forme normale (lemme 1.2.55), on a donc $D = \mathbf{infer}_{\Gamma,x:B}(N)$. On en déduit donc (règle PRODUIT), que $\Gamma \vdash \forall x : B.\, \mathbf{infer}_{\Gamma,x:B}(N) : s_3$ (9). Et donc la règle ABSTRACTION, (6) et (9) prouvent que $\Gamma \vdash M : \forall x : B.\, \mathbf{infer}_{\Gamma,x:B}(N)$. Et par réduction du type, on en déduit que $\Gamma \vdash M : \forall x : \mathrm{NF}(B).\, \mathbf{infer}_{\Gamma,x:B}(N)$.
3. On a $\forall x : B.\, \mathbf{infer}_{\Gamma,x:B}(N) \preccurlyeq \forall x : B.C$ (d'après (4)) et on en conclut :

$$\forall x : \mathrm{NF}(B).\, \mathbf{infer}_{\Gamma,x:B}(N) \preccurlyeq A$$

– $M = \forall x : B_1.B_2$: Par inversion de (*), on en déduit qu'il existe $(s_1, s_2, s_3) \in \mathcal{R}$ telles que

$$\Gamma \vdash B_1 : s_1 \qquad \text{et} \qquad \Gamma, x : B_1 \vdash B_2 : s_2$$

avec $s_3 \preccurlyeq A$. Par hypothèse de récurrence, on a donc $\mathbf{infer}_\Gamma(B_1) \neq \lightning$, $\Gamma \vdash B_1 : \mathbf{infer}_\Gamma(B_1)$, $\mathbf{infer}_\Gamma(B_1) \preccurlyeq s_1$, $\mathbf{infer}_{\Gamma,x:B_1}(B_2) \neq \lightning$, $\Gamma, x : B_1 \vdash B_2 : \mathbf{infer}_{\Gamma,x:B_1}(B_2)$ et $\mathbf{infer}_{\Gamma,x:B_1}(B_2) \preccurlyeq s_2$. Comme **infer** ne retourne que des termes en forme normale (lemme 1.2.55), on a alors $\mathbf{infer}_\Gamma(B_1) = r_1$ et $\mathbf{infer}_\Gamma(B_2) = r_2$ pour deux sortes r_1 et r_2. Soit r_3 la plus petite sorte (au sens de $\preccurlyeq$), telle que $(r_1, r_2, r_3) \in \overline{\mathcal{R}}$ (cet ensemble est non-vide car $(r_1, r_2, s_3) \in \overline{\mathcal{R}}$).

1. $\mathbf{infer}_\Gamma(M) = \mathbf{princ\text{-}regle}(r_1, r_2) = r_3 \neq \lightning$.
2. La règle PRODUIT prouve bien que $\Gamma \vdash \forall x : B_1.B_2 : r_3$.
3. Enfin, par minimalité de r_3, on a bien $r_3 \preccurlyeq s_3 \preccurlyeq A$. ☺

Dans un CTS principal et décidable, considérons maintenant l'algorithme récursif suivant :

$$\begin{aligned} \mathbf{is\_wf}(\langle\rangle) &= \mathsf{true} \\ \mathbf{is\_wf}(\Gamma, x : A) &= \begin{cases} \mathsf{true} & \text{si } \mathbf{is\_wf}(\Gamma) = \mathsf{true} \\ & \text{et } \mathbf{infer}_\Gamma(A) \neq \lightning \\ & \text{et } \mathbf{infer}_\Gamma(A) = s \\ \mathsf{false} & \text{sinon} \end{cases} \end{aligned}$$

Clairement l'algorithme termine et il ne retourne pas d'erreur. Il nous permet de décider de la formation des contextes :

1.2.60 Lemme
Supposons $\models$ DECIDABLE $\wedge$ PRINCIPAL $\wedge$ FULL, *alors pour tout* Γ, *et*

$$\mathbf{is\_wf}(\Gamma) = \mathsf{true} \quad \Leftrightarrow \quad \mathrm{WF}(\Gamma).$$

Démonstration Par induction sur la longueur de Γ,
– Si $\Gamma = \langle\rangle$, alors $\mathbf{is\_wf}(\Gamma) = \mathsf{true}$ et on a bien $\mathrm{WF}(\Gamma)$.
– Si $\Gamma = \Delta, x : A$, par hypothèse de récurrence on sait que $\mathbf{is\_wf}(\Delta) = \mathsf{true} \Leftrightarrow \mathrm{WF}(\Delta)$. Deux cas sont alors à distinguer :
  – $\mathbf{is\_wf}(\Delta) = \mathsf{false}$: Dans ce cas on a $\mathbf{is\_wf}(\Gamma) = \mathsf{false}$ et on a bien $\neg\,\mathrm{WF}(\Gamma)$ (car $\neg\,\mathrm{WF}(\Delta)$).

- $\mathbf{is\_wf}(\Delta) = \mathsf{true}$: Dans ce cas, on sait que $\mathrm{WF}(\Delta)$, on peut alors appliquer le lemme 1.2.58, pour prouver que $\mathbf{infer}_\Delta(A)$ termine et que
  - Soit $\mathbf{infer}_\Delta(A) = \lightning$, et d'après la contraposée du lemme 1.2.59, on en déduit que $\Delta \not\vdash A : s$ pour tout s et donc que $\neg\,\mathrm{WF}(\Gamma)$. Or dans ce cas, on a bien $\mathbf{is\_wf}(\Gamma) = \mathsf{false}$.
  - Soit $\mathbf{infer}_\Delta(A) \neq \lightning$ et $\mathbf{infer}_\Delta(A)$ est un type principal de A. On conclut en distinguant les deux derniers cas :
    - Soit $\mathbf{infer}_\Delta(A) \in \mathcal{S}$, et dans ce cas on a bien $\mathrm{WF}(\Gamma)$ et $\mathbf{is\_wf}(\Gamma) = \mathsf{true}$.
    - Soit $\mathbf{infer}_\Delta(A) \notin \mathcal{S}$. Supposons que $\mathrm{WF}(\Gamma)$, on aurait alors $\Delta \vdash A : s$ pour une certaine sorte s, et donc $\mathbf{infer}_\Delta(A) \preccurlyeq s$. Or cela est impossible car $\mathbf{infer}_\Delta(A)$ est en forme normale et n'est pas une sorte. On en conclut donc que $\neg\,\mathrm{WF}(\Gamma)$ et on a bien $\mathbf{is\_wf}(\Gamma) = \mathsf{false}$. ☺

Enfin, l'algorithme suivant nous permet de résoudre le problème de la vérification des types :

$$\mathbf{of\_type}(\Gamma, M, A) \quad = \quad \begin{cases} \mathsf{true} & \text{si } \mathbf{is\_wf}(\Gamma) = \mathsf{true} \\ & \text{et } \mathbf{infer}_\Gamma(M) \neq \lightning \\ & \text{et } (\mathbf{infer}_\Gamma(A) \neq \lightning \text{ ou } A \in \mathcal{S}) \\ & \text{et } \mathbf{infer}_\Gamma(M) \preccurlyeq A \\ \mathsf{false} & \text{sinon} \end{cases}$$

On a alors :

1.2.61 Lemme
Supposons $\models$ DECIDABLE $\wedge$ PRINCIPAL $\wedge$ FULL. *Alors pour tout* Γ, M, *et* A, $\mathbf{of\_type}(\Gamma, M, A)$ *termine,* $\mathbf{of\_type}(\Gamma, M, A) \neq \lightning$ *et*

$$\mathbf{of\_type}(\Gamma, M, A) = \mathsf{true} \quad \Leftrightarrow \quad \Gamma \vdash M : A.$$

Démonstration L'algorithme **of_type** termine toujours car on montré que **is_wf** termine toujours sans erreur.

- Si $\mathbf{is\_wf}(\Gamma) = \mathsf{false}$, alors $\mathbf{of\_type}(\Gamma, M, A) = \mathsf{false}$ et on a $\neg\,\mathrm{WF}(\Gamma)$, on ne peut donc pas avoir $\Gamma \vdash M : A$.
- Si $\mathbf{is\_wf}(\Gamma) = \mathsf{true}$, alors on a $\mathrm{WF}(\Gamma)$ et on sait alors que $\mathbf{infer}_\Gamma$ termine.
  - Si $\mathbf{infer}_\Gamma(M) = \lightning$, alors $\mathbf{of\_type}(\Gamma, M, A) = \mathsf{false}$. Et par complétude de **infer**, on déduit bien qu'on ne peut pas avoir $\Gamma \vdash M : A$.
  - Si $\mathbf{infer}_\Gamma(A) = \lightning$ et $A \notin \mathcal{S}$, alors $\mathbf{of\_type}(\Gamma, M, A) = \mathsf{false}$. Et par complétude de **infer**, on montre qu'on ne peut pas avoir $\mathrm{WF}_\Gamma(A)$. Et donc d'après le lemme 1.1.35, on ne peut avoir $\Gamma \vdash M : A$.
  - Si $\mathbf{infer}_\Gamma(M) \neq \lightning$, et $(\mathbf{infer}_\Gamma(A) \neq \lightning$ ou $A \in \mathcal{S})$, alors par correction de **infer**, on a $\mathrm{WF}_\Gamma(A)$ et $\Gamma \vdash M : \mathbf{infer}_\Gamma(M)$ et donc $\mathrm{WF}_\Gamma(\mathbf{infer}_\Gamma(M))$. On peut alors décider si $\mathbf{infer}_\Gamma(M) \preccurlyeq A$ (lemme 1.2.52). Ainsi :
    - Si $\mathbf{infer}_\Gamma(M) \preccurlyeq A$, alors on a $\mathbf{of\_type}(\Gamma, M, A) = \mathsf{true}$. Et comme on a $\mathrm{WF}_\Gamma(A)$ et $\Gamma \vdash M : \mathbf{infer}_\Gamma(A)$, on peut utiliser la cumulativité (lemme 1.1.28) pour montrer que $\Gamma \vdash M : A$.
    - Si $\mathbf{infer}_\Gamma(M) \not\preccurlyeq A$, alors on ne peut pas avoir $\Gamma \vdash M : A$ car $\mathbf{infer}_\Gamma(M)$ est principal par correction de **infer**. ☺

Nous pouvons néanmoins remarquer qu'en l'absence de la propriété FULL, le lemme de complétude 1.2.59, nous donne une réduction du problème de l'inférence d'un type principal à celui de l'inférence de type. En effet, supposons que l'on dispose d'une procédure $\mathbf{well\text{-}typed}_\Gamma(M)$ qui termine toujours et qui retourne true ou false selon qu'il existe ou non un A tel que $\Gamma \vdash M : A$. Alors le lemme 1.2.59 nous garantit que la procédure $\mathbf{infer}'$ définie par

$$\mathbf{infer}'_\Gamma(M) = \begin{cases} \mathbf{infer}_\Gamma(M) & \text{si } \mathbf{well\text{-}typed}_\Gamma(M) = \mathsf{true} \\ \lightning & \text{sinon} \end{cases}$$

est correcte et complète et ceci même en l'absence de l'hypothèse FULL.

On notera que van Benthem Jutting [vBJ93] propose une implémentation de **well-typed** pour tous les PTS normalisants. Il utilise pour cela les outils qu'il a développés pour prouver son théorème de renforcement. Jiménez [Jim99] a par la suite également adapté cette démonstration pour étendre le résultat aux CTS qui satisfont les hypothèses WEAK-SR$_{\succcurlyeq}$ et WEAK-SR$_{\preccurlyeq}$ (auxquelles il faut rajouter NICE-TOPSORT pour importer son résultat puisqu'il l'a prouvé dans un système ne comportant pas la règle Cumulativité-Sortes).

En pratique, en l'absence de l'hypothèse FULL, l'utilisateur qui ne dispose pas d'une procédure **well-typed** peut employer l'algorithme ci-dessous. Soit $\mathcal{P} \models \text{DECIDABLE} \wedge \text{PRINCIPAL}$ et soit l'algorithme "semi-récursif" suivant :

$$\begin{aligned}
\mathbf{infer}^*_\Gamma(x) &= \begin{cases} \mathrm{NF}(A) \text{ si } x : A \in \Gamma \\ \lightning \text{ sinon} \end{cases} \\
\mathbf{infer}^*_\Gamma(s) &= \mathbf{princ\text{-}sorte}(s) \\
\mathbf{infer}^*_\Gamma(\lambda x : A.M) &= \begin{cases} \forall x : \mathrm{NF}(A).\, \mathbf{infer}^*_{\Gamma,x:A}(M) & \text{si } \mathbf{infer}^*_\Gamma(A) = s_1 \\ & \text{et } \mathbf{infer}^*_{\Gamma,x:A}(\mathbf{infer}^*_{\Gamma,x:A}(M)) = s_2 \\ & \text{et } \exists s_3, (s_1, s_2, s_3) \in \mathcal{R} \end{cases} \\
\mathbf{infer}^*_\Gamma(M\,N) &= \begin{cases} \mathrm{NF}(B[N/x]) \text{ si } \mathbf{infer}^*_\Gamma(M) = \forall x : A.B, \text{ et } \mathbf{infer}^*_\Gamma(N) \preccurlyeq A \\ \lightning \text{ sinon} \end{cases} \\
\mathbf{infer}^*_\Gamma(\forall x : A.B) &= \begin{cases} \mathbf{princ\text{-}regle}(s_1, s_2) \text{ si } s_1 = \mathbf{infer}^*_\Gamma(A) \text{ et } s_2 = \mathbf{infer}^*_{\Gamma,x:A}(B) \\ \lightning \text{ sinon} \end{cases}
\end{aligned}$$

La terminaison de $\mathbf{infer}^*$ n'est pas garantie, en effet, il n'y a pas d'argument direct de décroissance qui prouverait l'impossibilité d'obtenir une infinité d'appels récursifs lors du calcul d'$\mathbf{infer}^*$ à cause l'appel récursif "$\mathbf{infer}^*_{\Gamma,x:A}(\mathbf{infer}^*_{\Gamma,x:A}(M))$". Mais sous hypothèse de terminaison on peut prouver la correction et la complétude de l'algorithme ci-dessus sans avoir à utiliser l'hypothèse FULL :

1.2.62 Lemme (Correction de l'inférence du type principal)
Supposons $\mathcal{P} \models \text{DECIDABLE} \wedge \text{PRINCIPAL}$*, et soit* Γ *tel que* $\mathrm{WF}(\Gamma)$ *et* M *un terme quelconque. Supposons que le calcul de* $\mathbf{infer}^*_\Gamma(M)$ *termine, alors :*

$$\mathbf{infer}^*_\Gamma(M) \neq \lightning \quad \Rightarrow \quad \Gamma \vdash M : \mathbf{infer}^*_\Gamma(M)$$

Démonstration On procède par induction sur le nombre d'appels récursifs employés pour calculer $\mathbf{infer}^*_\Gamma(M)$, puis par analyse par cas de la structure de M. On ne traite que le cas $M = \lambda x : B.N$ (les autres cas sont similaires à ceux de la preuve du lemme 1.2.58).

– Supposons $M = \lambda x : B.N$. Si $\mathbf{infer}^*_\Gamma(M) \neq \lightning$ cela signifie qu'il existe C tel que $\mathbf{infer}^*_{\Gamma,x:B}(N) = C \neq \lightning$, qu'il existe une sorte s_1 telle que $\mathbf{infer}^*_\Gamma(B) = s_1$, une sorte s_2 telle que $\mathbf{infer}^*_{\Gamma,x:B}(C) = s_2$, et une sorte s_3 telle que $(s_1, s_2, s_3) \in \mathcal{R}$. Par hypothèse de récurrence, on $\Gamma \vdash B : \mathbf{infer}^*_\Gamma(B)$ et on en déduit $\mathrm{WF}(\Gamma, x : B)$ et l'existence de $\mathrm{NF}(B)$ (lemme 1.1.36). On a donc bien $\mathbf{infer}^*_\Gamma(M) = \forall x : \mathrm{NF}(B).C$. Par hypothèse de récurrence (autorisée car on a montré $\mathrm{WF}(\Gamma, x : B)$) on a :

$$\Gamma, x : B \vdash N : C \qquad \Gamma \vdash B : s_1 \qquad \Gamma, x : B \vdash C : s_2$$

On en déduit donc que $\Gamma \vdash \forall x : B.C : s_3$ et donc la règle Abstraction montre que $\Gamma \vdash \lambda x : B.N : \forall x : B.C$. Et par réduction du type (lemme 1.1.47), on a $\Gamma \vdash \lambda x : B.N : \forall x : \mathrm{NF}(B).C$. ☺

Ainsi, on prouve aisément la complétude :

1.2.63 Lemme (Complétude de l'inférence du type principal)
Supposons $\mathcal{P} \models$ DECIDABLE $\wedge$ PRINCIPAL.
Si $\Gamma \vdash M : A$ *et si* $\mathbf{infer}^*(M)$ *termine, alors* $\mathbf{infer}^*_\Gamma(M) = \mathbf{infer}_\Gamma(M)$,

1.3 Les systèmes de types purs prédicatifs

La concept d'imprédicativité a été introduit par Henri Poincaré afin d'évacuer les cercles vicieux dans les définitions mathématiques. Une définition d'un objet x est imprédicative si elle contient une quantification sur une collection qui contient un x. Le bon sens dictait alors que l'on évite les définitions imprédicatives pour éviter les antinomies.

C'est par ailleurs ce genre d'antinomies qui rendent la théorie des types naïve incohérente (voir Section 1.1.10). On cherchera donc à présent à formaliser cette notion d'imprédicativité pour écarter les PTS qui ressembleraient trop à la théorie des types naïve. La définition suivante introduit un ordre sur les sortes : $s <_\mathcal{P} r$ doit se penser comme "s a été définie avant r".

1.3.1 Définition (Relation de dépendance)
On définit la relation de dépendance $<_\mathcal{P} \subseteq \mathcal{S} \times \mathcal{S}$ d'un WCTS $\mathcal{P}$, comme étant la plus petite relation satisfaisant les règles de dépendance ci-dessous.

$$\frac{(s, s') \in \mathcal{A}}{s <_\mathcal{P} s'}\ \textsc{Axiome} \qquad \frac{(s, s', s'') \in \mathcal{R} \qquad r <_\mathcal{P} s}{r <_\mathcal{P} s''}\ \textsc{Prémisse} \qquad \frac{(s, s', s'') \in \mathcal{R} \qquad r <_\mathcal{P} s'}{r <_\mathcal{P} s''}\ \textsc{Conclusion}$$

$$\frac{s <_\mathcal{P} s' \qquad s' <_\mathcal{P} s''}{s <_\mathcal{P} s''}\ \textsc{Transitivité} \qquad \frac{s <_\mathcal{P} s' \qquad (s', s'') \in \mathcal{C}}{s <_\mathcal{P} s''}\ \textsc{Cumulativité}$$

On notera $\leq_{\mathcal{P}}$ sa clôture réflexive.

On construit cette relation comme étant l'intersection de tous les ordres satisfaisant ces règles (l'intersection de deux ordres qui satisfont ces règles est bien une relation qui les satisfait). Nous ne nous étonnerons pas que notre définition de prédicativité est ... imprédicative (nous n'y verrons donc rien d'*essentiel*).

1.3.2 Remarque
On notera que si $\mathcal{P}$ est fini alors $<_{\mathcal{P}}$ est décidable : il existe un nombre fini de relations qui vérifient les règles de dépendance, il suffit donc de toutes les tester.

1.3.3 Lemme
Si $s <_{\mathcal{P}} s'$ et $s' \preccurlyeq s''$, alors $s <_{\mathcal{P}} s''$.

Démonstration Si $s <_{\mathcal{P}} s'$ et $s' \preccurlyeq s''$, alors :
- soit $(s', s'') \in \mathcal{C}$ et donc $s <_{\mathcal{P}} s''$,
- soit $s' = s''$ et on a alors bien $s <_{\mathcal{P}} s''$. ☺

1.3.4 Lemme
Si $\mathcal{P} \hookrightarrow_{\varphi} \mathcal{P}'$, alors

$$s_1 <_{\mathcal{P}} s_2 \Rightarrow \varphi(s_1) <_{\mathcal{P}'} \varphi(s_2)$$

Démonstration Par induction sur les règles de dérivation de $s_1 <_{\mathcal{P}} s_2$:
- Axiome : Si on suppose $(s_1, s_2) \in \mathcal{A}$, alors $(\varphi(s_1), \varphi(s_2)) \in \mathcal{A}'$ et donc $\varphi(s_1) <_{\mathcal{P}'} \varphi(s_2)$ (d'après la règle Axiome).
- Prémisse : On a $(s_1, s_2, s_3) \in \mathcal{R}$, $s <_{\mathcal{P}} s_1$ et l'hypothèse de récurrence $\varphi(s) <_{\mathcal{P}'} \varphi(s_1)$ et doit montrer que $\varphi(s) <_{\mathcal{P}'} \varphi(s_3)$. C'est la règle Prémisse avec $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) \in \mathcal{R}'$, nous permet de conclure.
- Conclusion : similaire au cas précédent.
- Transitivité : Par hypothèse d'induction, $\varphi(s_1) <_{\mathcal{P}'} \varphi(s_2)$ et $\varphi(s_2) <_{\mathcal{P}'} \varphi(s_3)$ et donc par transitivité de $<_{\mathcal{P}'}$, on déduit $\varphi(s_1) <_{\mathcal{P}'} \varphi(s_3)$.
- Cumulativité : On a $\varphi(s_1) <_{\mathcal{P}'} \varphi(s_2)$ et $s_2 \prec_{\mathcal{P}} s_3$. Comme φ est un morphisme, $\varphi(s_2) \preccurlyeq_{\mathcal{P}'} \varphi(s_3)$. D'après le lemme précédent, on prouve $\varphi(s_1) <_{\mathcal{P}'} \varphi(s_3)$. ☺

On en déduit le lemme ci-dessous.

1.3.5 Lemme
Si $\mathcal{P} \subseteq \mathcal{P}'$, alors

$$s_1 <_{\mathcal{P}} s_2 \Rightarrow s_1 <_{\mathcal{P}'} s_2$$

.

Démonstration L'identité est un morphisme de $\mathcal{P}$ dans $\mathcal{P}'$. ☺

On propose la définition suivante :

1.3.6 Définition (Système de type pur prédicatif)
On dira d'un WCTS qu'il est prédicatif si sa relation de dépendance est un ordre strict bien fondé. On dira qu'il est imprédicatif dans le cas contraire. On notera cette propriété PREDICATIVE.

La condition de bonne fondation dans la définition précédente est évidemment superflue dans le cas des WCTS finis.

1.3.7 Lemme
$\lambda\star$ est un PTS imprédicatif.

Démonstration On a $(\star,\star) \in \mathcal{A}_{\lambda\star} \Rightarrow \star <_{\lambda\star} \star$ et donc $<_{\lambda\star}$ est réflexif. ☺

1.3.8 Lemme
Soit $\mathcal{P}$ et $\mathcal{P}'$ deux WCTS tel que $\mathcal{P} \hookrightarrow_{\varphi} \mathcal{P}'$ (ou moins généralement $\mathcal{P} \subseteq \mathcal{P}'$), alors si $\mathcal{P}'$ est prédicatif (resp. si $\mathcal{P}$ est imprédicatif), alors $\mathcal{P}$ est prédicatif (resp. $\mathcal{P}'$ est imprédicatif).

Démonstration Il s'agit d'une conséquence directe du lemme 1.3.4. ☺

1.3.9 Lemme
La face supérieure du cube de Barendregt ($\mathcal{F}$, $\mathcal{F}_\omega$, $\lambda\Pi^2$ et CC) est constituée de systèmes imprédicatifs.

Démonstration Le système $\mathcal{F}$ est imprédicatif car $(\star,\Box) \in \mathcal{A}_{\mathcal{F}} \Rightarrow \star <_{\mathcal{F}} \Box$ et donc $(\Box,\star,\star) \in \mathcal{R}_{\mathcal{F}} \Rightarrow \star <_{\mathcal{F}} \star$. Et comme $\mathcal{F}$ est inclus dans $\mathcal{F}_\omega$, $\lambda\Pi^2$ et CC ils sont également imprédicatifs. ☺

1.3.10 Lemme
La face inférieure (λ, $\lambda\omega$, $\lambda\Pi$ et $\lambda\Pi_\omega$) du cube de Barendregt est consitutée de systèmes prédicatifs.

Démonstration On a $<_\lambda = <_{\lambda\omega} = <_{\lambda\Pi} = <_{\lambda\Pi_\omega} = \{(\star,\Box)\}$. ☺

1.3.11 Définition (Rang dans les systèmes prédicatifs)
Soit $\mathcal{P}$ un WCTS prédicatif. Soit $s \in \mathcal{S}_{\mathcal{P}}$, on définit par induction bien fondée un ordinal $\mathrm{Rang}_{\mathcal{P}}(s)$ de la façon suivante :

$$\mathrm{Rang}_{\mathcal{P}}(s) = \begin{cases} 0 \text{ si } s \text{ est minimal pour } <_{\mathcal{P}} \\ \sup_{s' <_{\mathcal{P}} s}(\mathrm{Rang}_{\mathcal{P}}(s') + 1) \text{ sinon} \end{cases}$$

et note $\mathrm{Rang}(\mathcal{P}) = \sup_{s \in \mathcal{S}_{\mathcal{P}}} \mathrm{Rang}_{\mathcal{P}}(s)$.

1.3.12 Lemme
$s_1 <_{\mathcal{P}} s_2$ *implique* $\mathrm{Rang}_{\mathcal{P}}(s_1) < \mathrm{Rang}_{\mathcal{P}}(s_2)$.

Démonstration On a

$$\mathrm{Rang}(s_1) + 1 \in \{\mathrm{Rang}(s_2') + 1 | s_2' <_{\mathcal{P}} s_2\}$$

car $s_1 <_{\mathcal{P}} s_2$. Donc $\mathrm{Rang}(s_1) + 1 \leq \sup\{\mathrm{Rang}(s_2') + 1 | s_2' <_{\mathcal{P}} s_2\} = \mathrm{Rang}(s_2)$ et donc $\mathrm{Rang}(s_1) < \mathrm{Rang}(s_2)$. ☺

1.3.13 Lemme
$s_1 \preccurlyeq_{\mathcal{P}} s_2$ *implique* $\mathrm{Rang}_{\mathcal{P}}(s_1) \leq \mathrm{Rang}_{\mathcal{P}}(s_2)$.

Démonstration Supposons $s_1 \preccurlyeq_{\mathcal{P}} s_2$, alors on a :

$$\{\text{Rang}_{\mathcal{P}}(s) + 1 | s <_{\mathcal{P}} s_1\} \subseteq \{\text{Rang}_{\mathcal{P}}(s) + 1 | s <_{\mathcal{P}} s_2\}$$

car la cumulativité (lemme 1.3.3) pour $<_{\mathcal{P}}$ prouve $s <_{\mathcal{P}} s_1 \wedge s_1 \preccurlyeq_{\mathcal{P}} s_2 \Rightarrow s <_{\mathcal{P}} s_2$. Et donc :

$$\text{Rang}_{\mathcal{P}}(s_1) = \sup\{\text{Rang}_{\mathcal{P}}(s) + 1 | s <_{\mathcal{P}} s_1\} \leq \sup\{\text{Rang}_{\mathcal{P}}(s) + 1 | s <_{\mathcal{P}} s_2\} = \text{Rang}_{\mathcal{P}}(s_2)$$

1.3.14 Lemme
Soit $\mathcal{P}$ un WCTS prédicatif. Si $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}}$, alors $\text{Rang}_{\mathcal{P}}(s_1) \leq \text{Rang}_{\mathcal{P}}(s_3)$ *et* $\text{Rang}_{\mathcal{P}}(s_2) \leq \text{Rang}_{\mathcal{P}}(s_3)$.

Démonstration On a

$$\{\text{Rang}(s'_1) + 1 | s'_1 <_{\mathcal{P}} s_1\} \subseteq \{\text{Rang}(s'_3) + 1 | s'_3 <_{\mathcal{P}} s_3\}$$

car $s'_1 < s_1$ implique $s'_1 < s_3$ d'après la règle PRÉMISSE de $<_{\mathcal{P}}$. Donc

$$\text{Rang}(s_1) = \sup\{\text{Rang}(s'_1) + 1 | s'_1 < s_1\} \leq \sup\{\text{Rang}(s'_3) + 1 | s'_3 < s'_3\} = \text{Rang}(s_3)$$

et de façon symétrique on montre $\text{Rang}(s_2) \leq \text{Rang}(s_3)$. ☺

1.3.1 Théorie des types prédicative

Un moyen de contourner l'incohérence de $\star : \star$ consiste à stratifier la sorte $\star$ en une hiérarchie de sortes $\star_0, \star_1, \ldots$. Soit $\alpha \in \mathbf{Ord}$ un ordinal. On définit le système $\lambda\star_\alpha$ suivant :

$$\begin{array}{rcl} \mathcal{S}_{\lambda\star_\alpha} & = & \{\star_i | i < \alpha\} \\ \mathcal{A}_{\lambda\star_\alpha} & = & \{(\star_i, \star_j) | i < j < \alpha\} \\ \mathcal{R}_{\lambda\star_\alpha} & = & \{(\star_i, \star_j, \star_k) | i, j \leq k < \alpha\} \\ \mathcal{C}_{\lambda\star_\alpha} & = & \mathcal{A}_{\lambda\star_\alpha} \end{array}$$

On remarque que $\lambda\star_\alpha$ est clos par cumulativité : $\overline{\lambda\star_\alpha} = \lambda\star_\alpha$. Ce système que nous appelons "théorie des types prédicatives", en fait une présentation sous forme de CTS de la théorie des types de Martin-Löf avec univers [MLS84].

Le système admet plusieurs présentations équivalentes, dont le CTS $\lambda\star_\alpha{}'$:

$$\begin{array}{rcl} \mathcal{S}_{\lambda\star_\alpha{}'} & = & \{\star_i | i < \alpha\} \\ \mathcal{A}_{\lambda\star_\alpha{}'} & = & \{(\star_i, \star_{i+1}) | i + 1 < \alpha\} \\ \mathcal{R}_{\lambda\star_\alpha{}'} & = & \{(\star_i, \star_i, \star_i) | i < \alpha\} \\ \mathcal{C}_{\lambda\star_\alpha{}'} & = & \{(\star_i, \star_j) | i < j < \alpha\} \end{array}$$

qui a l'avantage de contenir moins de règles et moins d'axiomes mais aussi le désavantage de nécessiter des dérivations de type en général plus longues pour typer les mêmes termes que $\lambda\star_\alpha$. On a le lemme suivant :

1.3.15 Lemme
Le CTS $\lambda\star_\alpha{}'$ est le plus petit sous-CTS $\mathcal{P}$ de $\lambda\star_\alpha$ tel que $\overline{\mathcal{P}} = \lambda\star_\alpha$.

Démonstration On montre tout d'abord que $\overline{\lambda\star_\alpha{}'} = \lambda\star_\alpha$, puis on vérifie que c'est le plus petit sous-CTS de $\lambda\star_\alpha$ qui vérifie cette propriété :

- $\overline{\lambda\star_\alpha{}'} = \lambda\star_\alpha$:
  - $\subseteq$: D'après le lemme 1.2.15 et le fait que $\overline{\lambda\star_\alpha} = \lambda\star_\alpha$, il suffit de vérifier que $\lambda\star_\alpha{}' \subseteq \lambda\star_\alpha$, ce qui est immédiat.
  - $\supseteq$:
    - $\mathcal{S}_{\lambda\star_\alpha} = \mathcal{S}_{\lambda\star_\alpha{}'}$ et $\mathcal{C}_{\lambda\star_\alpha} = \mathcal{C}_{\lambda\star_\alpha{}'}$,
    - $\mathcal{A}_{\lambda\star_\alpha} \subseteq \mathcal{A}_{\overline{\lambda\star_\alpha{}'}}$: Soit $(\star_i, \star_j) \in \mathcal{A}_{\lambda\star_\alpha}$, montrons que $(\star_i, \star_j) \in \mathcal{A}_{\overline{\lambda\star_\alpha{}'}}$ pour cela il suffit de constater que nécessairement $i < j$, que $(\star_i, \star_{i+1}) \in \mathcal{A}_{\lambda\star_\alpha{}'}$ et que $\star_{i+1} \preccurlyeq_{\lambda\star_\alpha{}'} \star_j$.
    - $\mathcal{R}_{\lambda\star_\alpha} \subseteq \mathcal{R}_{\overline{\lambda\star_\alpha{}'}}$: Soit $(\star_i, \star_j, \star_k) \in \mathcal{R}_{\lambda\star_\alpha}$, montrons que $(\star_i, \star_j, \star_k) \in \mathcal{R}_{\overline{\lambda\star_\alpha{}'}}$ pour ça il suffit de constater que nécessairement $i, j \leq k$, que $(\star_{\max(i,j)}, \star_{\max(i,j)}, \star_{\max(i,j)}) \in \mathcal{R}_{\lambda\star_\alpha{}'}$ et que $\star_i, \star_j \preccurlyeq_{\lambda\star_\alpha{}'} \star_{\max(i,j)}$ et $\star_{\max(i,j)} \preccurlyeq_{\lambda\star_\alpha{}'} \star_k$.
- Soit $\mathcal{P} \subseteq \lambda\star_\alpha$ tel que $\overline{\mathcal{P}} = \lambda\star_\mathcal{P}$, montrons que $\lambda\star_\alpha{}' \subseteq \mathcal{P}$: il suffit de vérifier l'inclusion des axiomes et des règles :
  - Soit $i < \alpha$, on sait que $(\star_i, \star_{i+1}) \in \mathcal{A}_{\overline{\mathcal{P}}}$, or par définition de $\overline{\mathcal{P}}$, cela signifie qu'il existe $j < \alpha$ tels que $(\star_i, \star_j) \in \mathcal{A}_\mathcal{P}$ et que $\star_j \preccurlyeq \star_{i+1}$ et on obtient donc que $i < j \leq i+1$ c'est-à-dire $j = i+1$ et $(\star_i, \star_{i+1}) \in \mathcal{A}_\mathcal{P}$.
  - Soit $l < \alpha$, on sait que $(\star_l, \star_l, \star_l) \in \mathcal{R}_{\overline{\mathcal{P}}}$, or par définition de $\overline{\mathcal{P}}$, il existe $i, j \leq k < \alpha$ tels que $(\star_i, \star_j, \star_k) \in \mathcal{R}_\mathcal{P}$ et $\star_k \preccurlyeq \star_l$, $\star_l \preccurlyeq \star_i, \star_j$ et donc $k \leq l \leq i, j \leq k$. On en déduit alors que $k = l = i = j$ et donc on a bien $(\star_l, \star_l, \star_l) \in \mathcal{R}_\mathcal{P}$. ☺

Le système $\lambda\star_\alpha$ vérifie les propriétés suivantes :

1.3.16 Lemme
1. $\lambda\star_\alpha \models$ CTS,
2. $\lambda\star_\alpha \models \mathrm{SR}_\beta$,
3. $\lambda\star_\alpha \models$ MINLOC,
4. $\lambda\star_\alpha \models$ PRINCIPAL,
5. $\lambda\star_\alpha \models$ UPSTABLE,
6. $\lambda\star_\alpha \models$ FULL,
7. $\lambda\star_\alpha \models$ STRENGTHENING,

Démonstration
1. La relation $\mathcal{C}_{\lambda\star_\alpha}$ est une relation transitive.
2. C'est une conséquence directe du lemme 1.1.42.
3. La vérification est immédiate.
4. La relation $\mathcal{C}_{\lambda\star_\alpha}$ est un ordre strict bien-fondé.
5. La vérification est immédiate.
6. La vérification est immédiate.
7. C'est une conséquence directe du lemme 1.2.42. ☺

Le CTS $\lambda\star_\omega{}'$ est un sous-système du Calcul des Constructions Étendu (ECC) de Luo [Luo89]. ECC est une variante du calcul des constructions, avec univers, sommes fortes, et une sorte imprédicative. Luo a prouvé que ce système est fortement normalisant en adaptant la preuve des candidats de réducibilité de Girard aux types dépendants [GLT89]. On admettra donc le théorème suivant :

1.3.17 Théorème
$\lambda\star_\omega$ est fortement normalisant.

Dans $\lambda\star_\alpha$ la relation d'axiome, la relation de cumulativité et la relation de dépendance coïncident :

1.3.18 Lemme

$$<_{\lambda\star_\alpha} = \mathcal{A}_{\lambda\star_\alpha}$$

Démonstration

$\subseteq$: trivial car $\{(\star_i, \star_j) | i < j\} = \mathcal{A}_{\lambda\star_\alpha}$ satisfait les règles qui définissent $<_{\lambda\star_\alpha}$:
- Axiome : $\mathcal{A}_{\lambda\star_\alpha}$ contient $\mathcal{A}_{\lambda\star_\alpha}$.
- Prémisse : Soit $(\star_i, \star_j, \star_k) \in \mathcal{R}_{\lambda\star_\alpha}$ et $(\star_l, \star_i) \in \mathcal{A}_{\lambda\star_\alpha}$, alors on a par définitions, $i, j \leq k$ et $l < i$. On en conclut bien que $l < k$ et donc que $(\star_l, \star_k) \in \mathcal{A}_{\lambda\star_\alpha}$.
- Conclusion : similaire au cas précédent.
- Transitivité : $\mathcal{A}_{\lambda\star_\alpha}$ est bien une relation transitive.
- Cumulativité : Puisqu'on a $\mathcal{A}_{\lambda\star_\alpha} = \mathcal{C}_{\lambda\star_\alpha}$ la cumulativité est synonyme de transitivité.

$\supseteq$: La règle d'axiome nous montre bien que $(s, s') \in \mathcal{A}_{\lambda\star_\alpha}$ implique $s <_{\lambda\star_\alpha} s'$. ☺

1.3.19 Lemme

$$\mathrm{Rang}_{\lambda\star_\alpha}(\star_i) = i$$

Démonstration Par induction sur i,

$$\begin{aligned}
\mathrm{Rang}_{\lambda\star_\alpha}(\star_i) &= \sup_{\star_j <_{\lambda\star_\alpha} \star_i} (\mathrm{Rang}_{\lambda\star_\alpha}(\star_j) + 1) \\
&= \sup_{j<i} (\mathrm{Rang}_{\lambda\star_\alpha}(\star_j) + 1) \\
&\quad (\mathbf{HR}) \\
&= \sup_{j<i} (j+1) \\
&= i
\end{aligned}$$

☺

On en déduit donc que $\lambda\star_\alpha$ est un CTS prédicatif de rang α :

1.3.20 Lemme
On a $\lambda\star_\alpha \models$ PREDICATIVE *et* $\mathrm{Rang}(\lambda\star_\alpha) = \alpha$.

Terminalité faible de $\lambda\star_\alpha$ pour les WCTS prédicatifs Nous montrons ici que tous les WCTS prédicatifs se plongent dans $\lambda\star_\alpha$ pour un certain α : il existe un morphisme (*a priori* pas unique, c'est pourquoi on parle de terminalité *faible*) entre les WCTS prédicatifs et $\lambda\star_\alpha$.

1.3.21 Lemme
Soit $\mathcal{P}$ un WCTS prédicatif et soit $\alpha = \text{Rang}(\mathcal{P})$.

$$\begin{array}{rcl} \mathcal{S}_\mathcal{P} & \longrightarrow & \mathcal{S}_{\lambda\star_\alpha} \\ s & \longmapsto & \star_{\text{Rang}_\mathcal{P}(s)} \end{array}$$

est un morphisme.

Démonstration Soit φ cette fonction, montrons qu'elle est un morphisme :
- Préserve les axiomes : Si $(s_1, s_2) \in \mathcal{A}_\mathcal{P}$, on a $s_1 <_\mathcal{P} s_2$ et donc $\text{Rang}_\mathcal{P}\, s_1 < \text{Rang}_\mathcal{P}\, s_2$ et donc $(\varphi(s_1), \varphi(s_2)) \in \mathcal{A}_{\lambda\star_\alpha}$.
- Préserve la cumulativité : Soit $s_1, s_2 \in \mathcal{S}_\mathcal{P}$ tels que $s_1 \preccurlyeq_\mathcal{P} s_2$, le lemme 1.3.13 nous dit que $\text{Rang}_\mathcal{P}(s_1) \leq \text{Rang}_\mathcal{P}(s_2)$ et donc $\varphi(s_1) \preccurlyeq_{\lambda\star_\alpha} \varphi(s_2)$.
- Préserve les règles : $(s_1, s_2, s_3) \in \mathcal{R}_\mathcal{P}$, d'après le lemme 1.3.14, on a $\text{Rang}(s_1) \leq \text{Rang}(s_3)$ et $\text{Rang}(s_2) \leq \text{Rang}(s_3)$ et donc $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) \in \lambda\star_\alpha$. ☺

Réciproquement, on montre que les WCTS prédicatifs sont exactement les WCTS qui se plongent dans $\lambda\star_\alpha$.

1.3.22 Lemme
Un WCTS $\mathcal{P}$ est prédicatif si et seulement s'il existe un ordinal α et un morphisme φ tels que $\mathcal{P} \hookrightarrow_\varphi \lambda\star_\alpha$.

Démonstration C'est la conjonction du lemme 1.3.21 et du lemme 1.3.8. ☺

1.3.23 Théorème
Tous les WCTS prédicatifs sont fortement normalisants.

Démonstration Soit $\mathcal{P}$ un WCTS prédicatif et Γ, A et B tels que $\Gamma \vdash_\mathcal{P} A : B$, par compacité (lemme 1.1.39), il existe un sous-WCTS fini $\mathcal{P}'$ de $\mathcal{P}$ tel que $\Gamma \vdash_{\mathcal{P}'} A : B$. D'après le lemme 1.3.8, $\mathcal{P}'$ est prédicatif et il existe donc un morphisme $\varphi : \mathcal{P}' \longrightarrow \lambda\star_\omega$. Et puisque $\varphi(\Gamma) \vdash_{\lambda\star_\omega} \varphi(A) : \varphi(B)$, on obtient d'après le Théorème 1.3.17 que $\varphi(A)$ est une expression fortement normalisante et que, d'après le lemme 1.1.64, A l'est aussi. On a donc bien prouvé que $\mathcal{P}$ est fortement normalisant. ☺

En particulier, on peut généraliser le Théorème 1.3.17 :

1.3.24 Corollaire
Pour tout α, $\lambda\star_\alpha$ est fortement normalisant.

Démonstration $\lambda\star_\alpha$ est un CTS prédicatif. ☺

1.3.25 Lemme
On a $\lambda\star_\omega \models$ DECIDABLE.

Démonstration Le théorème 1.3.17 affirme que $\lambda\star_\omega$ est fortement normalisant et on implémente facilement les tests d'appartenance aux paramètres en comparant des entiers. ☺

De plus, d'après les lemmes 1.2.58 et 1.2.59, l'algorithme **infer** (sous-section 1.2.10) peut être utilisé directement pour résoudre le problème de l'inférence du type principal.

1.3.2 Système F stratifié

On appelle *Système $\mathcal{F}$ stratifié* le CTS défini par la spécification suivante :

$$\begin{aligned}
\mathcal{S}_{\mathcal{SF}^\alpha} &= \{\star_i, \Box_i | i < \alpha\} \\
\mathcal{A}_{\mathcal{SF}^\alpha} &= \{(\star_i, \Box_j) | i < j < \alpha\} \\
\mathcal{R}_{\mathcal{SF}^\alpha} &= \{(\star_i, \star_j, \star_k) | i, j \leq k < \alpha\} \cup \{(\Box_i, \star_j, \star_k) | 0 < i, j \leq k < \alpha\} \\
\mathcal{C}_{\mathcal{SF}^\alpha} &= \{(\star_i, \star_j) | i < j < \alpha\} \cup \{(\Box_i, \Box_j) | 0 < i < j < \alpha\}
\end{aligned}$$

Tout comme $\lambda\star_\alpha$, le système $\mathcal{SF}^\alpha$ admet une présentation plus succincte équivalente :

$$\begin{aligned}
\mathcal{S}_{\mathcal{SF}^{\alpha\prime}} &= \mathcal{S}_{\mathcal{SF}^\alpha} \\
\mathcal{A}_{\mathcal{SF}^{\alpha\prime}} &= \{(\star_i, \Box_{i+1}) | i + 1 < \alpha\} \\
\mathcal{R}_{\mathcal{SF}^{\alpha\prime}} &= \{(\star_i, \star_i, \star_i) | i < \alpha\} \cup \{(\Box_i, \star_i, \star_i) | 0 < i < \alpha\} \\
\mathcal{C}_{\mathcal{SF}^{\alpha\prime}} &= \mathcal{C}_{\mathcal{SF}^\alpha}
\end{aligned}$$

et on peut prouver que $\mathcal{SF}^{\alpha\prime}$ est le plus petit sous-CTS $\mathcal{P}$ de $\mathcal{SF}^\alpha$ tel que $\overline{\mathcal{P}} = \mathcal{SF}^\alpha$.

Ce système se plonge naturellement dans Système $\mathcal{F}$: la fonction

$$\begin{array}{rccc}
\psi & : & \mathcal{S}_{\mathcal{SF}^\alpha} & \longrightarrow & \mathcal{S}_{\mathcal{F}} \\
 & & \star_i & \mapsto & \star \\
 & & \Box_i & \mapsto & \Box
\end{array}$$

est trivialement un morphisme de $\mathcal{SF}^\alpha$ dans système $\mathcal{F}$. Il peut être vu comme la version "prédicative" de système $\mathcal{F}$, puisqu'il a été obtenu en stratifiant la sorte des types. On peut par ailleurs prouver que tous les WCTS prédicatifs qui se plongent dans $\mathcal{F}$ se plongent dans $\mathcal{SF}^\alpha$ pour un certain α :

1.3.26 Lemme
Soit $\mathcal{P}$ un WCTS prédicatif tel que $\mathcal{P} \hookrightarrow_\varphi \mathcal{F}$, alors il existe un morphisme χ tel que $\mathcal{P} \hookrightarrow_\chi \mathcal{SF}^{\mathrm{Rang}(\mathcal{P})}$ et $\varphi = \psi \circ \chi$.

Démonstration Soit χ la fonction définie par :

$$\begin{array}{rccc}
\chi & : & \mathcal{S}_{\mathcal{P}} & \longrightarrow & \mathcal{S}_{\mathcal{SF}^{\mathrm{Rang}(\mathcal{P})}} \\
 & & s & \mapsto & \begin{cases} \star_{\mathrm{Rang}(s)} & \text{si } \varphi(s) = \star \\ \Box_{\mathrm{Rang}(s)} & \text{si } \varphi(s) = \Box \end{cases}
\end{array}$$

Posons $\alpha = \mathrm{Rang}(\mathcal{P})$ et montrons que $\mathcal{P} \hookrightarrow_\chi \mathcal{SF}^\alpha$:

- Soit $(s, r) \in \mathcal{A}_\mathcal{P}$, on doit montrer que $(\chi(s), \chi(r)) \in \mathcal{A}_{\mathcal{SF}^\alpha}$. Or comme φ est un morphisme, on a $(\varphi(s), \varphi(r)) \in \mathcal{A}_\mathcal{F}$ et donc $(\varphi(s), \varphi(r)) = (\star, \Box)$. On en déduit donc que $(\chi(s), \chi(r)) = (\star_{\mathrm{Rang}(s)}, \Box_{\mathrm{Rang}(r)})$. Or par définition de $<_\mathcal{P}$, on a $s <_\mathcal{P} r$ et donc d'après le lemme 1.3.12, on en déduit $\mathrm{Rang}(s) < \mathrm{Rang}(r)$. On a donc bien $(\star_{\mathrm{Rang}(s)}, \Box_{\mathrm{Rang}(r)}) \in \mathcal{A}_{\mathcal{SF}^\alpha}$.
- Soit $(s_1, s_2, s_3) \in \mathcal{R}_\mathcal{P}$, on doit montrer que $(\chi(s_1), \chi(s_2), \chi(s_3)) \in \mathcal{R}_{\mathcal{SF}^\alpha}$. Or comme φ est un morphisme, on a $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) \in \mathcal{R}_\mathcal{F}$. Deux cas sont alors possibles :
  1. Soit $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) = (\star, \star, \star)$ et dans ce cas on a :

$$(\chi(s_1), \chi(s_2), \chi(s_3)) = (\star_{\mathrm{Rang}(s_1)}, \star_{\mathrm{Rang}(s_2)}, \star_{\mathrm{Rang}(s_3)})$$

2. Soit $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) = (\Box, \star, \star)$ et dans ce cas on a :

$$(\chi(s_1), \chi(s_2), \chi(s_3)) = (\Box_{\mathrm{Rang}(s_1)}, \star_{\mathrm{Rang}(s_2)}, \star_{\mathrm{Rang}(s_3)})$$

Donc pour prouver $(\chi(s_1), \chi(s_2), \chi(s_3)) \in \mathcal{R}_{\mathcal{SF}^\alpha}$ il suffit de montrer que $\mathrm{Rang}(s_1) \leq \mathrm{Rang}(s_3)$ et $\mathrm{Rang}(s_2) \leq \mathrm{Rang}(s_3)$. Or c'est exactement le lemme 1.3.14.

– Soit $(s, r) \in \mathcal{C}_{\mathcal{P}}$, on doit montrer que $\chi(s) \preccurlyeq_{\mathcal{SF}^\alpha} \chi(r)$. Or comme φ est un morphisme et comme $\mathcal{C}_{\mathcal{F}} = \emptyset$, on a $\varphi(s) = \varphi(r)$. Donc il suffit de montrer que $\mathrm{Rang}(s) \leq \mathrm{Rang}(r)$. On utilise pour ça le lemme 1.3.13. ☺

Ce système a été introduit par [Sta81] afin de contourner l'imprédicativité de Système $\mathcal{F}$: En effet, contrairement à système $\mathcal{F}$, l'absence d'imprédicativité permet de donner des modèles ensemblistes simples.

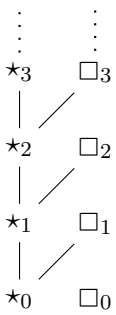
Le système $\mathcal{SF}^\alpha$ vérifie les propriétés suivantes :

1.3.27 Lemme

1. $\mathcal{SF}^\alpha \models$ CTS,
2. $\mathcal{SF}^\alpha \models \mathrm{SR}_\beta$,
3. $\mathcal{SF}^\alpha \models$ MINLOC,
4. $\mathcal{SF}^\alpha \models$ PRINCIPAL,
5. $\mathcal{SF}^\alpha \models$ UPSTABLE,
6. $\mathcal{SF}^\alpha \models$ STRENGTHENING,
7. $\mathcal{SF}^\alpha \models$ PREDICATIVE,
8. $\mathcal{SF}^\omega \models$ DECIDABLE,

Démonstration

1. La relation $\mathcal{C}_{\mathcal{SF}^\alpha}$ est une relation transitive.
2. C'est une conséquence directe du lemme 1.1.42.
3. La vérification est immédiate.
4. La relation $\mathcal{C}_{\mathcal{SF}^\alpha}$ est un ordre strict bien-fondé.
5. La vérification de UPSTABLE est immédiate.
6. C'est une conséquence directe du lemme 1.2.42.
7. La relation $<_{\mathcal{SF}^\alpha}$ est égale à la la relation $\{(\star_i, \star_j), (\star_i, \Box_j) | i < j < \alpha\}$ que l'on peut représenter par :



On a $\star_i <_{\mathcal{SF}^\alpha} \Box_j$ car $\mathcal{A}_{\mathcal{SF}^\alpha} \subseteq <_{\mathcal{SF}^\alpha}$ et $\star_i <_{\mathcal{SF}^\alpha} \star_j$ car $(\Box_j, \star_j, \star_j) \in \mathcal{R}_{\mathcal{SF}^\alpha}$ et $\star_i <_{\mathcal{SF}^\alpha} \Box_j$.

8. Tous les CTS prédicatifs sont fortement normalisables (théorème 1.3.23) et on implémente facilement les tests d'appartenance aux paramètres en comparant des entiers. ☺

Ce CTS ne vérifie néanmoins pas la condition FULL et donc l'algorithme **infer** de la sous-section 1.2.10 ne peut pas être utilisé directement pour résoudre le problème de l'inférence du type principal.

Or le fait que $\mathcal{SF}^\omega$ ne soit pas un CTS dépendant (c'est-à-dire nous n'avons pas la règle de la forme $(\star_i, \Box_j, \Box_k)$), nous permet de montrer facilement par inversion (lemme 1.1.32) qu'on ne trouve jamais de redex dans les types. On en déduit alors que l'ensemble des habitants de $\star_i$ est décrit par :

$$\begin{aligned}
\mathbf{Inhab}_\Gamma(\star_i) &= \{\alpha | \alpha : \star_j \in \Gamma \wedge j \leq i\ \} \\
&\cup \{A \to B | A \in \mathbf{Inhab}_\Gamma(\star_i) \wedge B \in \mathbf{Inhab}_{\Gamma,_:A}(\star_i)\} \\
&\cup \{\forall \alpha : \star_j . A | A \in \mathbf{Inhab}_{\Gamma,\alpha:\star_j}(\star_i) \wedge j < i\}
\end{aligned}$$

et donc que l'appartenance à $\mathbf{Inhab}_\Gamma(\star_i)$ est décidable : il suffit simplement de vérifier récursivement que le terme a la bonne forme. De la même façon, on se convainc que l'appartenance à ensemble $\bigcup_{i<\omega} \mathbf{Inhab}_\Gamma(\star_i)$ est décidable.

On en déduit alors sans difficulté que l'algorithme suivant est correct et complet pour le problème de l'inférence du type principal dans $\mathcal{SF}^\omega$.

$$\begin{aligned}
\mathbf{infer}_\Gamma(x) &= \begin{cases} A \text{ si } x : A \in \Gamma \\ \lightning \text{ sinon} \end{cases} \\
\mathbf{infer}_\Gamma(\star_i) &= \Box_{i+1} \\
\mathbf{infer}_\Gamma(\Box_i) &= \lightning \\
\mathbf{infer}_\Gamma(\lambda x : A.M) &= \begin{cases} \forall x : A.\, \mathbf{infer}_{\Gamma,x:A}(M) & \text{s'il existe } i \in \mathbb{N} \text{ et } j \in \mathbb{N} \text{ tels que} \\ & \quad A \in \mathbf{Inhab}_\Gamma(\star_i) \vee A = \star_i \\ & \quad \mathbf{infer}_{\Gamma,x:A}(M) \in \mathbf{Inhab}_\Gamma(\star_j) \\ \lightning & \text{sinon} \end{cases} \\
\mathbf{infer}_\Gamma(M\,N) &= \begin{cases} B[N/x] & \text{si } \mathbf{infer}_\Gamma(M) = \forall x : A.B, \\ & \text{et s'il existe } i \in \mathbb{N} \text{ tel que } \mathbf{infer}_\Gamma(N) = \star_i \\ B & \text{si } \mathbf{infer}_\Gamma(M) = \forall x : A.B \\ & \text{et s'il existe } i \in \mathbb{N} \text{ tel que } \mathbf{infer}_\Gamma(N) \in \mathbf{Inhab}_\Gamma(\star_i) \\ \lightning & \text{sinon} \end{cases} \\
\mathbf{infer}_\Gamma(\forall x : A.B) &= \begin{cases} \star_{\max(i,j)} & \text{si } s_i = \mathbf{infer}_\Gamma(A) \text{ avec } s \in \{\star, \Box\} \\ & \text{et } \star_j = \mathbf{infer}_{\Gamma,x:A}(B) \\ \lightning & \text{sinon} \end{cases}
\end{aligned}$$

Cet algorithme traite les abstractions et les applications en distinguant deux cas : les abstractions/applications de type (dans le cas où le domaine de l'abstraction est égal à $\star_i$ ou dans le cas où l'argument de l'application appartient à $\mathbf{Inhab}_\Gamma(\star_i)$).

Programmation. On peut représenter les entiers en suivant l'encodage habituel des entiers de Church paramétré par le niveau du type. Si n est un entier, on notera $\overline{n}^i$ l'expression

$$\lambda\alpha : \star_i, f : \alpha \to \alpha, x : \alpha. \overbrace{f\,(f\,\cdots(f}^{n \text{ fois}}\ x)\cdots)$$

et on notera nat_i son type principal, c'est-à-dire :

$$\mathsf{nat}_i = \forall\alpha : \star_i.(\alpha \to \alpha) \to \alpha \to \alpha$$

On a $\vdash \mathsf{nat}_i : \star_{i+1}$ et, pour tout entier n, la règle suivante est admissible :

$$\frac{}{\vdash \overline{n}^i : \mathsf{nat}_i}$$

On peut implémenter la fonction retournant le successeur des entiers :

$$\begin{array}{rcl} \mathsf{succ}_i & : & \mathsf{nat}_i \to \mathsf{nat}_i \\ \mathsf{succ}_i & = & \lambda n : \mathsf{nat}_i, \alpha : \star_i, f : \alpha \to \alpha, x : \alpha. f\,(n\,\alpha\,(f\,x)) \\ \mathsf{succ}_i\ \overline{n}^i & \trianglerighteq & \overline{n+1}^i \end{array}$$

Et on peut itérer cette fonction via l'encodage des entiers pour implémenter l'addition :

$$\begin{array}{rcl} \mathsf{plus}_i & : & \mathsf{nat}_{i+1} \to \mathsf{nat}_i \to \mathsf{nat}_i \\ \mathsf{plus}_i & = & \lambda n : \mathsf{nat}_{i+1}.n\ \mathsf{nat}_i\ \mathsf{succ}_i \\ \mathsf{plus}_i\ \overline{n}^{i+1}\,\overline{m}^i & \trianglerighteq & \overline{n+m}^i \end{array}$$

On note au passage que le niveau du type de l'argument sur lequel on effectue l'itération doit être incrémenté pour que le terme soit bien typé. On peut alors voir les entiers de type nat_i comme étant des entiers que l'on peut utiliser pour construire par itération d'une fonction des entiers de niveau strictement inférieur à i.

On utilise alors les fonctions $\mathsf{coercion}_i^j$ pour $j < i$ afin de plonger la représentation d'un entier vers un niveau inférieur :

$$\begin{array}{rcl} \mathsf{coercion}_i^j & : & \mathsf{nat}_i \to \mathsf{nat}_j \\ \mathsf{coercion}_i^j & = & \lambda n : \mathsf{nat}_i, \alpha : \star_j, f : \alpha \to \alpha, x : \alpha. n\,\alpha\,f\,x \\ \mathsf{coercion}_i^j\ \overline{n}^i & \trianglerighteq & \overline{n}^j \end{array}$$

On remarque que si on accepte "d'ouvrir la représentation des entiers" (en introduisant à gauche la quantification de type), on peut parvenir à implémenter certaines opérations avec un type plus souple. Comme par exemple pour l'addition :

$$\begin{array}{rcl} \mathsf{plus}_i & : & \mathsf{nat}_i \to \mathsf{nat}_i \to \mathsf{nat}_i \\ \mathsf{plus}_i & = & \lambda n\,m : \mathsf{nat}_i, \alpha : \star_i, f : \alpha \to \alpha, x : \alpha. n\,\alpha\,f\,(m\,\alpha\,f\,x) \\ \mathsf{plus}_i\ \overline{n}^i\,\overline{m}^i & \trianglerighteq & \overline{n+m}^i \end{array}$$

De même pour la multiplication :

$$\begin{array}{rcl} \mathsf{mult}_i & : & \mathsf{nat}_i \to \mathsf{nat}_i \to \mathsf{nat}_i \\ \mathsf{mult}_i & = & \lambda n\, m : \mathsf{nat}_i, \alpha : \star_i, f : \alpha \to \alpha, x : \alpha . n\, \alpha\, (m\, \alpha\, f)\, x \\ \mathsf{mult}_i\ \overline{n}^i\, \overline{m}^i & \trianglerighteq & \overline{n \times m}^i \end{array}$$

L'implémentation de l'exponentiation est un peu plus difficile, mais l'idée est la même : il faut repousser le plus tard possible l'utilisation de l'abstraction de type.

$$\begin{array}{rcl} \mathsf{exp}_i & : & \mathsf{nat}_i \to \mathsf{nat}_i \to \mathsf{nat}_i \\ \mathsf{exp}_i & = & \lambda n : \mathsf{nat}_i, m : \mathsf{nat}_i, \alpha : \star_i . \\ & & \quad m\, ((\alpha \to \alpha) \to \alpha \to \alpha) \\ & & \quad\ (\lambda k : (\alpha \to \alpha) \to \alpha \to \alpha, f : \alpha \to \alpha . n\, \alpha\, (k\, f)) \\ & & \quad\ (\lambda f : \alpha \to \alpha, x : \alpha . f\, x) \\ \mathsf{exp}_i\ \overline{n}^i\, \overline{m}^i & \trianglerighteq & \overline{n^m}^i \end{array}$$

La fonction qui permet de représenter les tours d'exponentielles de hauteur arbitraire peut être implémentée par le terme ci-dessus de type $\mathsf{nat}_{i+1} \to \mathsf{nat}_i \to \mathsf{nat}_i$.

$$\begin{array}{rcl} \mathsf{tower}_i & : & \mathsf{nat}_{i+1} \to \mathsf{nat}_i \to \mathsf{nat}_i \\ \mathsf{tower}_i & = & \lambda n : \mathsf{nat}_{i+1}, m : \mathsf{nat}_i . n\, (\mathsf{nat}_i)\, (\mathsf{exp}_i\ \overline{2}^i)\, m \\ \mathsf{tower}_i\ \overline{n}^{i+1}\, \overline{m}^i & \trianglerighteq & \overline{2^{2^{\cdot^{\cdot^{m}}}}}^i \quad \text{(la tour est de hauteur } n\text{)} \end{array}$$

On verra plus tard qu'on atteint ce qui semble être les limites de l'itération dans ce système, puisqu'on montrera qu'elle ne peut pas être implémentée avec un type de la forme $\mathsf{nat}_i \to \mathsf{nat}_i \to \mathsf{nat}_i$.

On peut également utiliser l'encodage standard pour encoder le produit cartésien à l'aide de la quantification du second-ordre. On notera donc $\sigma \times_k \tau$ le type $\forall \gamma : \star_k . (\sigma \to \tau \to \gamma) \to \gamma$. On notera que le type du produit doit être indexé par le niveau de la conclusion. On a alors $\alpha : \star_i, \beta : \star_j \vdash \alpha \times_k \beta : \star_{\max(k+1,i,j)}$:

$$\dfrac{\alpha : \star_i, \beta : \star_j \vdash \star_k : \square_{k+1} \qquad \dfrac{\Gamma \vdash \alpha : \star_i \qquad \Gamma \vdash \beta : \star_i \qquad \Gamma \vdash \gamma : \star_k}{\Gamma \vdash (\alpha \to \beta \to \gamma) \to \gamma : \star_{\max(k,i,j)}}}{\alpha : \star_i, \beta : \star_j \vdash \alpha \times_k \beta : \star_{\max(k+1,i,j)}}$$

Cette famille de types est alors munie des fonctions de construction et destructions suivantes :

$$\begin{array}{rcl} \mathsf{proj}_i^1 & : & \forall \alpha\, \beta : \star_i . \alpha \times_i \beta \to \alpha \\ \mathsf{proj}_i^1 & = & \lambda \alpha\, \beta : \star_i, c : \alpha \times_i \beta . c\, \alpha\, (\lambda x : \alpha, y : \beta . x) \\ \mathsf{proj}_i^2 & : & \forall \alpha\, \beta : \star_i . \alpha \times_i \beta \to \beta \\ \mathsf{proj}_i^2 & = & \lambda \alpha\, \beta : \star_i, c : \alpha \times_i \beta . c\, \alpha\, (\lambda x : \alpha, y : \beta . y) \\ \mathsf{pair}_i & : & \forall \alpha\, \beta : \star_i . \alpha \to \beta \to \alpha \times_i \beta \\ \mathsf{pair}_i & = & \lambda \alpha\, \beta : \star_i, x : \alpha, y : \beta, \gamma : \star_i, f : \alpha \to \beta \to \gamma . f\, x\, y \end{array}$$

On notera $\langle A, B\rangle_i$ le terme $\mathsf{pair}_i\ \sigma\,\tau\,A\,B$ où σ et τ seront les types principaux de A et B dans le contexte considéré (il peut toujours être inféré). De la même façon, on notera $(A)_i^k$ le terme $\mathsf{proj}_i^k\ \sigma\,\tau\,A$ quand $\sigma \times_i \tau$ sera le type de A. On montre alors facilement que les dérivations de typage suivantes sont admissibles :

$$\frac{\Gamma \vdash M : \tau \qquad \Gamma \vdash N : \sigma \qquad \Gamma \vdash \tau : \star_i \qquad \Gamma \vdash \sigma : \star_i}{\Gamma \vdash \langle M, N\rangle_i : \tau \times_i \sigma}$$

$$\frac{\Gamma \vdash M : \tau_1 \times_i \tau_2 \qquad \Gamma \vdash \tau_1 : \star_i \qquad \Gamma \vdash \tau_2 : \star_i}{\Gamma \vdash (M)_i^k : \tau_k}$$

Et on vérifie facilement les règles de réduction suivantes :

$$\begin{array}{rcl} (\langle A_1, A_2\rangle_i)_i^1 & \rhd & A_1 \\ (\langle A_1, A_2\rangle_i)_i^2 & \rhd & A_2 \end{array}$$

On peut en déduire par une technique standard une implémentation du prédécesseur sur les entiers :

$$\begin{array}{rcl} \mathsf{pred}_i & : & \mathsf{nat}_{i+1} \to \mathsf{nat}_i \\ \mathsf{pred}_i & = & \lambda n : \mathsf{nat}_{i+1}, \alpha : \star_i, f : \alpha \to \alpha, x : \alpha. \\ & & \Big(n \quad (\alpha \times_i \alpha) \\ & & \quad (\lambda c : \alpha \times_i \alpha.\langle f\,(c)_i^1, (c)_i^1\rangle) \\ & & \quad \langle x, x\rangle\Big)_i^2 \\ \mathsf{pred}_i\ \overline{n+1}^{i+1} & \trianglerighteq & \overline{n}^i \\ \mathsf{pred}_i\ \overline{0}^{i+1} & \trianglerighteq & \overline{0}^i \end{array}$$

Et, en suivant la même technique, on peut implémenter l'itérateur "rec" qui nous donne un schéma de récursion général :

$$\begin{array}{rcl} \mathsf{rec}_i & : & \forall\alpha.(\alpha \to \mathsf{nat}_i \to \alpha) \to \alpha \to \mathsf{nat}_{i+2} \to \alpha \\ \mathsf{rec}_i & = & \lambda\alpha : \star_i, f : \alpha \to \mathsf{nat}_i \to \alpha, x : \alpha, n : \mathsf{nat}_{i+2}\,. \\ & & \Big(n \quad (\alpha \times_{i+1} \mathsf{nat}_i) \\ & & \quad (\lambda c : \alpha \times_{i+1} \mathsf{nat}_i\,.\langle f\,(c)_{i+1}^1\,(c)_{i+1}^2, \mathsf{succ}_i\,(c)_{i+1}^2\rangle_{i+1}) \\ & & \quad \langle x, \overline{0}^i\rangle_{i+1}\Big)_{i+1}^1 \\ \mathsf{rec}_i\ \alpha\,f\,x\,\overline{0}^{i+2} & \trianglerighteq & x \\ \mathsf{rec}_i\ \alpha\,f\,x\,\overline{n+1}^{i+2} & \trianglerighteq & \overbrace{f\,(f\,\cdots\,(f}^{n+1 \text{ fois}}\ x\,\overline{0}^i)\,\cdots\,\overline{n-1}^i)\,\overline{n}^i \end{array}$$

On peut utiliser ce schéma de récurrence pour implémenter des schémas de produit, de somme et de "tétration" bornés :

$$\begin{array}{rcl}
\mathsf{sum}_i & : & (\mathsf{nat}_i \to \mathsf{nat}_i) \to \mathsf{nat}_{i+2} \to \mathsf{nat}_i \\
\mathsf{sum}_i & = & \lambda(f : \mathsf{nat}_i \to \mathsf{nat}_i)(n : \mathsf{nat}_{i+2}).\, \mathsf{rec}_i\ \mathsf{nat}_i\, (\lambda r\, n : \mathsf{nat}_i\, .\, \mathsf{plus}_i\ r\, (f\, n))\, \overline{0}^i\, n \\
\mathsf{sum}_i\ \overline{f}^i\, \overline{n}^{i+2} & \trianglerighteq & \overline{f(0) + \cdots + f(n-1)}^i
\end{array}$$

$$\begin{array}{rcl}
\mathsf{prod}_i & : & (\mathsf{nat}_i \to \mathsf{nat}_i) \to \mathsf{nat}_{i+2} \to \mathsf{nat}_i \\
\mathsf{prod}_i & = & \lambda(f : \mathsf{nat}_i \to \mathsf{nat}_i)(n : \mathsf{nat}_{i+2}).\, \mathsf{rec}_i\ \mathsf{nat}_i\, (\lambda r\, n : \mathsf{nat}_i\, .\, \mathsf{mult}_i\ r\, (f\, n))\, \overline{1}^i\, n \\
\mathsf{prod}_i\ \overline{f}^i\, \overline{n}^{i+2} & \trianglerighteq & \overline{f(0) \times \cdots \times f(n-1)}^i
\end{array}$$

$$\begin{array}{rcl}
\mathsf{tret}_i & : & (\mathsf{nat}_i \to \mathsf{nat}_i) \to \mathsf{nat}_{i+2} \to \mathsf{nat}_i \\
\mathsf{tret}_i & = & \lambda(f : \mathsf{nat}_i \to \mathsf{nat}_i)(n : \mathsf{nat}_{i+2}).\, \mathsf{rec}_i\ \mathsf{nat}_i\, (\lambda r\, n : \mathsf{nat}_i\, .\, \mathsf{exp}_i\ r\, (f\, n))\, \overline{1}^i\, n \\
\mathsf{tret}_i\ \overline{f}^i\, \overline{n}^{i+2} & \trianglerighteq & \overline{f(n-1)^{(f(n-2)^{\cdots^{f(0)}})}}^i
\end{array}$$

où $\overline{f}^i$ désigne un terme de type $\mathsf{nat}_i \to \mathsf{nat}_i$ représentant une fonction numérique $f : \mathbb{N} \longrightarrow \mathbb{N}$ (c'est-à-dire tel que pour tout n, $\overline{f}^i\, \overline{n}^i \trianglerighteq \overline{f(n)}^i$).

On ne peut néanmoins pas utiliser le schéma rec_i pour itérer sur des fonctions qui ne sont pas de la forme $\alpha \to \mathsf{nat}_i \to \alpha$. En particulier, on ne peut pas l'utiliser sur la fonction tower_i vue plus haut.

Daniel Leivant [Lei91] a prouvé que les fonctions représentables dans ce système sont exactement les fonctions super-élémentaires correspondant au quatrième étage ξ_4 de la hiérarchie de Grzegorczyk (voir [Ros] par exemple). Cet ensemble de fonctions est strictement inclus dans l'ensemble des fonctions primitives récursives qui est lui-même strictement plus petit que l'ensemble des fonctions représentables dans système $\mathcal{F}$. En particulier, la fonction d'Ackermann ack, connue pour ne pas être primitive récursive, est représentable dans système $\mathcal{F}$.

1.4 Les systèmes de types purs imprédicatifs

Nous avons vu que le critère de prédicativité implique la forte normalisation des CTS (théorème 1.3.23) et donc la cohérence. Ce critère permet en particulier d'écarter des systèmes tels que $\lambda\star$. Mais nous avons également étudié des systèmes imprédicatifs qui sont tout de même fortement normalisants ; comme par exemple toute la face supérieure du λ-cube. Dans cette section, nous chercherons donc à élargir le critère de prédicativité pour prendre en compte ces systèmes.

1.4.1 Définition (faiblement imprédicatif)
Soit $\mathcal{P}$ un WCTS, on dit qu'il est faiblement imprédicatif s'il existe un ensemble $S \subseteq \mathcal{S}_\mathcal{P}$ de sortes tel que :

1. S est un ensemble de sortes minimales pour $\mathcal{A}_\mathcal{P}$:

$$\text{pour toutes sortes } s, s' \in \mathcal{S}_\mathcal{P},\ s \in S \Rightarrow (s', s) \notin \mathcal{A}_\mathcal{P}$$

2. S est un ensemble stable vers le bas pour $\mathcal{C}_\mathcal{P}$:

$$\text{pour toutes sortes } s, s' \in \mathcal{S}_\mathcal{P},\ s \in S \wedge s' \preccurlyeq s \Rightarrow s' \in S$$

3. Si $(s_1, s_2, s_3) \in \mathcal{R}_\mathcal{P}$, alors $s_2 \in S \Leftrightarrow s_3 \in S$.
4. Le WCTS $\underline{\mathcal{P}}$, défini par les paramètres ci-dessous, est prédicatif.

$$\begin{array}{rcl} \mathcal{S}_{\underline{\mathcal{P}}} & = & \mathcal{S}_\mathcal{P} \\ \mathcal{A}_{\underline{\mathcal{P}}} & = & \mathcal{A}_\mathcal{P} \\ \mathcal{R}_{\underline{\mathcal{P}}} & = & \{(s_1, s_2, s_3) \in \mathcal{R}_\mathcal{P} \,|\, s_2 \notin S \wedge s_3 \ \notin S\} \\ \mathcal{C}_{\underline{\mathcal{P}}} & = & \mathcal{C}_\mathcal{P} \end{array}$$

On notera WEAKLY-IMPREDICATIVE cette propriété.

On remarque qu'il suffit de vérifier qu'il existe un plus petit ensemble, noté $\mathcal{S}^{\text{impr}}_\mathcal{P}$, de sortes qui vérifient les conditions 1.-4., il peut être construit en prenant l'intersection de tous les ensembles qui vérifient ces conditions. En effet, il est clair que que les conditions 1.-3. sont stables par intersection. Pour la condition 4. on remarque que si S_1 et S_2 sont deux ensembles de sortes vérifiant les condition 1.-4., alors les WCTS $\mathcal{P}_{S_1}$ et $\mathcal{P}_{S_2}$ contiennent tous les deux le WCTS $\mathcal{P}_{S_1 \cap S_2}$ et donc d'après le lemme 1.3.8, le fait que $\mathcal{P}_{S_1}$ et $\mathcal{P}_{S_2}$ soient prédicatifs (un seul aurait suffi) implique que $\mathcal{P}_{S_1 \cap S_2}$ l'est aussi.

On notera, dans la suite, $\underline{\mathcal{P}}$ le WCTS $\mathcal{P}_{\mathcal{S}^{\text{impr}}_\mathcal{P}}$ et on appellera les sortes $s \in \mathcal{S}^{\text{impr}}_\mathcal{P}$ des *sortes imprédicatives*.

1.4.2 Remarque

Dans le cas des WCTS finis, la "faible imprédicativité" est décidable. Soit $\mathcal{P}$ un WCTS fini, on peut construire de façon effective l'ensemble $\mathcal{S}^{impr}_\mathcal{P}$ vérifiant les conditions de la définition 1.4.1 : il y a en effet un nombre fini de parties de $\mathcal{S}$ et les conditions 1.-4. sont toutes décidables dans le cas fini (voir aussi remarque 1.3.2).

1.4.3 Lemme

Si $\mathcal{P}$ est prédicatif, alors il est faiblement imprédicatif.

Démonstration L'ensemble $\mathcal{S}^{\text{impr}}_\mathcal{P} = \emptyset$ convient. ☺

1.4.4 Lemme

Si $s \in \mathcal{S}^{impr}_\mathcal{P}$, alors $\text{Rang}_{\underline{\mathcal{P}}}(s) = 0$.

Démonstration On montre que

$$\forall s' s, s' <_{\underline{\mathcal{P}}} s \Rightarrow s \notin \mathcal{S}^{\text{impr}}_\mathcal{P}$$

par induction sur la structure des règles permettant de prouver $s' <_{\underline{\mathcal{P}}} s$:

- Axiome : Dans ce cas, on a $(s', s) \in \mathcal{A}$, on en déduit $s \notin \mathcal{S}^{\text{impr}}_\mathcal{P}$ par minimalité de $\mathcal{S}^{\text{impr}}_\mathcal{P}$ pour la relation $\mathcal{A}$.
- Prémisse / Conclusion : Dans ces cas, on aurait $(s'', s', s) \in \mathcal{R}_{\underline{\mathcal{P}}}$ ou $(s', s'', s) \in \mathcal{R}_{\underline{\mathcal{P}}}$ ce qui n'est pas possible puisque $s \in \mathcal{S}^{\text{impr}}_\mathcal{P}$.

- Cumulativité : Dans ce cas, il existe $s'' \preccurlyeq s$, tel que $s' <_{\underline{\mathcal{P}}} s''$. Or par stabilité de $\mathcal{S}^{\text{impr}}_{\mathcal{P}}$ par cumulativité, on a $s'' \in \mathcal{S}^{\text{impr}}_{\mathcal{P}}$ et donc par hypothèse d'induction on en déduit une contradiction.
- Transitivité : Dans ce cas, il existe $s'' <_{\underline{\mathcal{P}}} s$ et par hypothèse d'induction, on en déduit une contradiction. ☺

1.4.5 Lemme

Soient $\mathcal{P}$ et $\mathcal{P}'$ deux WCTS tels que $\mathcal{P} \hookrightarrow_{\varphi} \mathcal{P}'$ (ou moins généralement $\mathcal{P} \subseteq \mathcal{P}'$), alors si $\mathcal{P}'$ est faiblement imprédicatif, alors $\mathcal{P}$ est faiblement imprédicatif.

Démonstration Soit S l'ensemble des sortes qui s'envoient par φ vers des sortes imprédicatives :

$$S = \{s \in \mathcal{S} \,|\varphi(s) \in \mathcal{S}^{\text{impr}}_{\mathcal{P}'}\}$$

On vérifie alors facilement que :

- S est un ensemble de sortes minimales pour $\mathcal{A}_{\mathcal{P}}$: Supposons $(s, s') \in \mathcal{A}$ et $s' \in S$, alors on aurait $(\varphi(s), \varphi(s')) \in \mathcal{A}_{\mathcal{P}'}$ et $\varphi(s') \in \mathcal{S}^{\text{impr}}_{\mathcal{P}'}$ ce qui contredirait la minimalité des sortes de $\mathcal{S}^{\text{impr}}_{\mathcal{P}'}$ par la relation $\mathcal{A}_{\mathcal{P}'}$.
- S est stable vers le bas pour $\mathcal{C}$: Supposons $s \preccurlyeq_{\mathcal{P}} s'$ et $s' \in S$, alors on aurait $\varphi(s) \preccurlyeq_{\mathcal{P}'} \varphi(s')$ et $\varphi(s') \in \mathcal{S}^{\text{impr}}_{\mathcal{P}}$ et donc par stabilité vers le bas de $\mathcal{S}^{\text{impr}}_{\mathcal{P}'}$, on en déduirait que $\varphi(s) \in \mathcal{S}^{\text{impr}}_{\mathcal{P}'}$ et donc que $s \in S$.
- Si $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}}$, alors $s_2 \in S \Leftrightarrow s_3 \in S$: Supposons $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}}$, alors $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) \in \mathcal{R}_{\mathcal{P}'}$. De plus :
$$s_2 \in S \Leftrightarrow \varphi(s_2) \in \mathcal{S}^{\text{impr}}_{\mathcal{P}} \Leftrightarrow \varphi(s_3) \in \mathcal{S}^{\text{impr}}_{\mathcal{P}} \Leftrightarrow s_3 \in S$$
- P_S est prédicatif : Il suffit pour ça de remarquer que φ est un morphisme de P_S dans $\underline{\mathcal{P}'}$. En effet, si $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}_S}$, alors on a :
  - $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) \in \mathcal{R}_{\mathcal{P}'}$,
  - Et $s_2 \notin S$ et $s_3 \notin S$, c'est-à-dire $\varphi(s_2) \notin \mathcal{S}^{\text{impr}}_{\mathcal{P}'}$ et $\varphi(s_3) \notin \mathcal{S}^{\text{impr}}_{\mathcal{P}'}$.

  On en déduit donc que $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) \in \mathcal{R}_{\underline{\mathcal{P}'}}$. La préservation des axiomes et de la cumulativité ne posent pas de problème puisque $\mathcal{A}_{\mathcal{P}_S} = \mathcal{A}_{\underline{\mathcal{P}'}}$ et $\mathcal{R}_{\mathcal{P}_S} = \mathcal{R}_{\underline{\mathcal{P}'}}$. Enfin, le lemme 1.3.8 nous permet de conclure que $\mathcal{P}_S$ est prédicatif.

Le WCTS $\mathcal{P}$ est donc bien faiblement imprédicatif. ☺

Nous allons maintenant définir le Calcul des Construction CC_α avec α "univers" qui sera le WCTS terminal pour les WCTS faiblement imprédicatifs.

1.4.1 Le Calcul des Constructions avec univers

Soit $\alpha \in \mathbf{Ord}$ un ordinal. On définit le système CC_α suivant :

$$\begin{array}{rcl@{\quad}l@{\quad}l}
\mathcal{S}_{\mathrm{CC}_\alpha} & = & \{ & \star & \} \\
 & \cup & \{ & \star_i & \mid\ i < \alpha\} \\
\mathcal{A}_{\mathrm{CC}_\alpha} & = & \{ & (\star, \star_i) & \mid\ i < \alpha\} \\
 & \cup & \{ & (\star_i, \star_j) & \mid\ i < j < \alpha\} \\
\mathcal{R}_{\mathrm{CC}_\alpha} & = & \{ & (s, \star, \star) & \mid\ s \in \mathcal{S}_{\mathrm{CC}_\alpha}\} \\
 & \cup & \{ & (\star, \star_i, \star_j) & \mid\ i \leq j < \alpha\} \\
 & \cup & \{ & (\star_i, \star_j, \star_k) & \mid\ i, j \leq k < \alpha\ \} \\
\mathcal{C}_{\mathrm{CC}_\alpha} & = & \{ & (\star, \star_i) & \mid\ i < \alpha\} \\
 & \cup & \{ & (\star_i, \star_j) & \mid\ i < j < \alpha\}
\end{array}$$

Dans ce système, que l'on appellera *Calcul des Constructions avec univers*, les sortes $\star_0, \star_1, \ldots$ sont appelées *univers*.

1.4.6 Lemme
Soit $\mathcal{P}$ un WCTS faiblement imprédicatif et soit $\alpha = \mathrm{Rang}(\underline{\mathcal{P}})$. Alors la fonction

$$\begin{array}{rcl}
\mathcal{S}_\mathcal{P} & \longrightarrow & \mathcal{S}_{\mathrm{CC}_\alpha} \\
s & \longmapsto & \begin{cases} \star \ si & s \in \mathcal{S}_\mathcal{P}^{impr} \\ \star_{\mathrm{Rang}_{\underline{\mathcal{P}}}(s)} & sinon \end{cases}
\end{array}$$

est un morphisme de PTS de $\mathcal{P}$ dans CC_α.

Démonstration Soit φ cette fonction, montrons qu'elle est un morphisme :
- Préserve les axiomes : Supposons $(s_1, s_2) \in \mathcal{A}_\mathcal{P}$, nécessairement, $s_2 \notin \mathcal{S}_\mathcal{P}^{\mathrm{impr}}$ car s_2 n'est pas minimal pour $\mathcal{A}_\mathcal{P}$ et donc $\varphi(s_2) \neq \star$. On distingue alors deux cas :
  - Soit $s_1 \in \mathcal{S}_\mathcal{P}^{\mathrm{impr}}$, alors $\varphi(s_1) = \star$, et donc $(\varphi(s_1), \varphi(s_2)) \in \mathcal{A}_{\mathrm{CC}_\alpha}$.
  - Soit $s_1 \notin \mathcal{S}_\mathcal{P}^{\mathrm{impr}}$, dans ce cas $\varphi(s_1) = \star_{\mathrm{Rang}_{\underline{\mathcal{P}}}(s_1)}$ et $\varphi(s_2) = \star_{\mathrm{Rang}_{\underline{\mathcal{P}}}(s_2)}$. Or $(s_1, s_2) \in \mathcal{A}$ implique que $\mathrm{Rang}_{\underline{\mathcal{P}}}(s_1) < \mathrm{Rang}_{\underline{\mathcal{P}}}(s_2)$. On en déduit donc que $(\varphi(s_1), \varphi(s_2)) \in \mathcal{A}_{\mathrm{CC}_\alpha}$.
- Préserve les règles : Supposons $(s_1, s_2, s_3) \in \mathcal{R}_\mathcal{P}$,
  - Soit $s_2 \in \mathcal{S}_\mathcal{P}^{\mathrm{impr}}$, dans ce cas on a également $s_3 \in \mathcal{S}_\mathcal{P}^{\mathrm{impr}}$ car $\mathcal{P}$ est faiblement imprédicatif, et donc $\varphi(s_2) = \varphi(s_3) = \star$. On a alors bien $(\varphi(s_1), \star, \star) \in \mathcal{R}_{\mathrm{CC}_\alpha}$.
  - Soit $s_2 \notin \mathcal{S}_\mathcal{P}^{\mathrm{impr}}$, dans ce cas on a également $s_3 \notin \mathcal{S}_\mathcal{P}^{\mathrm{impr}}$. On a alors $(s_1, s_2, s_3) \in \mathcal{R}_{\underline{\mathcal{P}}}$ et donc $\mathrm{Rang}_{\underline{\mathcal{P}}}(s_1) \leq \mathrm{Rang}_{\underline{\mathcal{P}}}(s_3)$ et $\mathrm{Rang}_{\underline{\mathcal{P}}}(s_2) \leq \mathrm{Rang}_{\underline{\mathcal{P}}}(s_3)$ (lemme 1.3.14). De plus, par définition de φ, on a :
    $$\varphi(s_2) = \star_{\mathrm{Rang}_{\underline{\mathcal{P}}}(s_2)} \text{ et } \varphi(s_3) = \star_{\mathrm{Rang}_{\underline{\mathcal{P}}}(s_3)}$$
    On distingue alors deux cas :
    - $s_1 \in \mathcal{S}_\mathcal{P}^{\mathrm{impr}}$: Alors $\varphi(s_1) = \star$, et on a bien $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) \in \mathcal{R}_{\mathrm{CC}_\alpha}$.
    - $s_1 \notin \mathcal{S}_\mathcal{P}^{\mathrm{impr}}$: Alors $\varphi(s_1) = \star_{\mathrm{Rang}_{\underline{\mathcal{P}}}(s_1)}$ et donc $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) \in \mathcal{R}_{\mathrm{CC}_\alpha}$.
- Préserve la cumulativité : Supposons $s_1 \preccurlyeq_\mathcal{P} s_2$.
  - Soit $s_2 \in \mathcal{S}_\mathcal{P}^{\mathrm{impr}}$, et dans ce cas on a $s_1 = s_2$ et donc $\varphi(s_1) \preccurlyeq_{\mathrm{CC}_\alpha} \varphi(s_2)$.

– Soit $s_2 \notin \mathcal{S}_{\mathcal{P}}^{\text{impr}}$. Dans ce cas, on a $\varphi(s_2) = \star_{\text{Rang}_{\underline{\mathcal{P}}}(s_2)}$ et on peut distinguer deux cas :
  – $s_1 \in \mathcal{S}_{\mathcal{P}}^{\text{impr}}$: On a alors $\varphi(s_1) = \star$ et donc on a bien $\varphi(s_1) \preccurlyeq_{\text{CC}_\alpha} \varphi(s_2)$.
  – $s_2 \notin \mathcal{S}_{\mathcal{P}}^{\text{impr}}$: On a $\varphi(s_1) = \star_{\text{Rang}_{\underline{\mathcal{P}}}(s_1)}$ avec $\text{Rang}_{\underline{\mathcal{P}}}(s_1) \leq \text{Rang}_{\underline{\mathcal{P}}}(s_2)$ (lemme 1.3.13). On en déduit donc $\varphi(s_1) \preccurlyeq_{\text{CC}_\alpha} \varphi(s_2)$. ☺

Tout comme $\lambda\star_\alpha$, CC_α est clos par cumulativité et il est équivalent au système $\text{CC}_\alpha{}'$ défini par les paramètres suivants :

$$\begin{array}{rcl}
\mathcal{S}_{\text{CC}_\alpha{}'} & = & \{\star\} \cup \{\star_i | i < \alpha\} \\
\mathcal{A}_{\text{CC}_\alpha{}'} & = & \{(\star, \star_0)\} \cup \{(\star_i, \star_{i+1}) | i < \alpha\} \\
\mathcal{R}_{\text{CC}_\alpha{}'} & = & \{(s, \star, \star) | s \in \mathcal{S}_{\text{CC}_\alpha{}'}\} \\
 & \cup & \{(\star, \star_i, \star_i\} | i < \alpha\} \\
 & \cup & \{(\star_i, \star_i, \star_i) | i < \alpha\ \} \\
\mathcal{C}_{\text{CC}_\alpha{}'} & = & \{(\star, s) | s \in \mathcal{S}_{\lambda\star_\alpha}\} \cup \{(\star_i, \star_j) | i < j < \alpha\}
\end{array}$$

On vérifie facilement que $\overline{\text{CC}_\alpha{}'} = \text{CC}_\alpha$. Or, $\text{CC}_\alpha{}'$ est un sous-système direct du calcul des constructions étendu de Luo [Luo89], on admettra donc que :

1.4.7 Théorème
CC_ω *est fortement normalisant.*

On en déduit alors de façon similaire au cas des systèmes prédicatifs que :

1.4.8 Théorème
Tous les systèmes faiblement imprédicatifs sont fortement normalisants.

Démonstration Soit $\mathcal{P}$ un WCTS faiblement imprédicatif et Γ, A et B tels que $\Gamma \vdash_{\mathcal{P}} A : B$, par compacité (lemme 1.1.39), il existe un sous-WCTS fini $\mathcal{P}'$ de $\mathcal{P}$ tel que $\Gamma \vdash_{\mathcal{P}'} A : B$. D'après le lemme 1.4.5, $\mathcal{P}'$ est prédicatif et il existe donc un morphisme $\varphi : \mathcal{P}' \longrightarrow \lambda\star_\omega$. Et puisque $\varphi(\Gamma) \vdash_{\text{CC}_\omega} \varphi(A) : \varphi(B)$ et que, d'après le théorème 1.4.7, $\varphi(A)$ est une expression fortement normalisante et que, d'après le lemme 1.1.64, A l'est aussi. On a donc bien prouvé que $\mathcal{P}$ est fortement normalisant. ☺

Faible imprédicativité : un critère de terminaison puissant. Le théorème précédent indique que la faible imprédicativité est un critère de terminaison. Ce critère est de plus décidable dans le cas fini (remarque 1.4.2). La plupart des systèmes étudiés normalisants dans la littérature satisfont ce critère. On notera toutefois qu'il existe des systèmes fortement normalisants qui ne le satisfont pas, comme par exemple le système $\mathcal{F}$ "cyclique" défini par les paramètres suivants :

$$\begin{array}{rcl}
\mathcal{S} & = & \{\star, \Box\} \\
\mathcal{A} & = & \{(\star, \Box), (\Box, \star)\ \} \\
\mathcal{R} & = & \{(\star, \star, \star), (\Box, \star, \star)\} \\
\mathcal{C} & = & \emptyset
\end{array}$$

Ce système n'est pas faiblement imprédicatif car il n'est pas prédicatif et ne contient aucune sorte minimale pour la relation $\mathcal{A}$. On trouvera une preuve de la normalisation de ce système dans [MW96].

Propriétés métathéoriques du calcul des constructions avec univers. Le système CC_α vérifie les propriétés suivantes :

1.4.9 Lemme

1. $\mathrm{CC}_\alpha \models$ CTS,
2. $\mathrm{CC}_\alpha \models \mathrm{SR}_\beta$,
3. $\mathrm{CC}_\alpha \models$ MINLOC,
4. $\mathrm{CC}_\alpha \models$ PRINCIPAL,
5. $\mathrm{CC}_\alpha \models$ UPSTABLE,
6. $\mathrm{CC}_\alpha \models$ FULL,
7. $\mathrm{CC}_\alpha \models$ STRENGTHENING,
8. $\mathrm{CC}_\alpha \models$ WEAKLY-IMPREDICATIVE,
9. $\mathrm{CC}_\alpha \models$ DECIDABLE,

Démonstration

1. La relation $\mathcal{C}_{\mathrm{CC}\,\alpha}$ est une relation transitive.
2. C'est une conséquence directe du lemme 1.1.42.
3. La vérification est immédiate.
4. La relation $\mathcal{C}_{\mathrm{CC}\,\alpha}$ est un ordre strict bien-fondé.
5. La vérification est immédiate.
6. La vérification est immédiate.
7. C'est une conséquence directe du lemme 1.2.42.
8. Si on considère $S = \{\star\}$, alors :
   - $\star$ est bien une sorte minimale pour $\mathcal{A}$.
   - S est stable vers le bas : si on $s \preccurlyeq \star$, alors $s = \star$.
   - Soit $(s_1, s_2, s_3) \in \mathcal{R}$. on a bien $s_2 = \star \Leftrightarrow s_3 = \star$.
   - Enfin, le système $\mathcal{P}_S = \lambda\star_\alpha$ est bien prédicatif (lemme 1.3.20).
9. Le théorème 1.4.8 prouve la normalisation et on implémente facilement les tests d'appartenance aux paramètres en comparant des entiers. ☺

Chapitre 2

Extension aux types inductifs

Dans cette section, nous présenterons une extention des systèmes de types cumulatifs aux types inductifs. Notre présentation sera succincte et nous ne nous intéresserons pas aux propriétés sémantiques des types inductifs. Plus précisément, nous n'aborderons pas l'étude de la normalisation. Nous renvoyons le lecteur aux travaux de Jean-Pierre Jouanaud, Frédéric Blanqui et Mitsuhiro Okada pour des études plus complètes dans des cadres un peu moins généraux [Bla04, BJO02, BJO99].

La présentation des inductifs de ce travail sera volontairement la plus proche possible de celle des inductifs du système Coq ; du reste, nous encourageons le lecteur à consulter le Manuel de Référence [Coq04] pour plus de détails. Cela nous permettra de donner un cadre uniforme aux exemples des chapitres suivants tout en nous appuyant sur les bonnes propriétés du système de Coq. Ainsi tous les systèmes avec types inductifs que nous étudierons se plongeront naturellement dans Coq, ce qui fournira une garantie de leur normalisation à tout ceux qui sont prêts à admettre celle du système de types de Coq.

2.1 Syntaxe et définitions

2.1.1 Définition (Système de types cumulatifs avec inductif)
On appellera système de types cumulatifs avec inductif (CTSI en abrégé), la donnée $\mathcal{P}$ de :

- Un CTS $(\mathcal{S}_\mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{R}_\mathcal{P}, \mathcal{C}_\mathcal{P})$,
- D'un ensemble $\mathcal{I}_\mathcal{P}$ d'identifiants des inductifs,
- D'un ensemble $\mathcal{E}_\mathcal{P} \subseteq \mathcal{I}_\mathcal{P} \times \mathcal{S}_\mathcal{P}$ qui décrira l'ensemble des éliminations autorisées,
- À chaque identifiant d'inductif $I \in \mathcal{I}$, on associera :
  - Un entier p qui désignera le nombre de paramètres,
  - Une famille de p termes $Q_1, \ldots, Q_p$ désignant les types des paramètres,
  - Un terme A que l'on appelle l'arité de I,
  - Un entier k qui désignera le nombre de constructeurs,
  - Une famille de k identifiants $c_0, \ldots, c_k$ de constructeurs frais,
  - Un ensemble ordonné de termes $C_1, \ldots, C_k$ qui désigneront les types respectifs de chacun des constructeurs.

  Et on note cette association $\texttt{Ind}(x_1 : Q_1, \ldots, x_p : Q_p, I : A, c_0 : C_0, \ldots, c_k : C_k)$. On demandera de plus que les identifiants de constructeurs soient associés à au plus un identifiant d'inductif.

On étend de plus les termes par la grammaire suivante :

$$\begin{array}{rcll} A, B, ... & := & \cdots & \\ & | & I[\overrightarrow{P}] & \underline{\text{nom}} \\ & | & c[\overrightarrow{P}] & \underline{\text{constructeur}} \\ & | & \texttt{fix}\, f : A, \overrightarrow{x : B}.M & \underline{\text{point-fixe}} \\ & | & \texttt{case}_I(M,\ \overrightarrow{P},\ \lambda\overrightarrow{y : B}, i : C.T,\ \overrightarrow{\lambda\overrightarrow{z : E}.F}) & \underline{\text{analyse par cas}} \end{array}$$

où x parcourt un ensemble de variables $\mathcal{V}$, I l'ensemble des noms de types inductifs $\mathcal{I}$, c parcourt l'ensemble des identifiants de constructeurs. On notera simplement I et c pour désigner les identifiants $I[]$ et les constructeurs $c[]$ des inductifs sans paramètre.

Notation vectorielle. La notation vectorielle $\overrightarrow{A}$ désigne une famille finie $A_1, \ldots, A_n$ que l'on notera également $\overrightarrow{A}^n$ pour spécifier sa taille. On imbriquera parfois cette notation pour désigner les familles de familles de termes. Par exemple, les expressions de la forme $\texttt{case}_I(M, \overrightarrow{P}^p, \lambda\overrightarrow{y : B}^n, i : C.T, \overrightarrow{\lambda\overrightarrow{z : E}^m.F}^k)$ se déplient en :

$$\begin{array}{c} \texttt{case}_I \Big(M, P_1, \ldots, P_p, \lambda y_1 : B_1, \ldots y_n : B_n, i : C.T, \\ \lambda z_{1,1} : E_{1,1}, \ldots, z_{1,m_1} : E_{1,m_1}.F_1, \\ \vdots \\ \lambda z_{k,1} : E_{k,1}, \ldots, z_{k,m_k} : E_{k,m_k}.F_k\Big) \end{array}$$

Cette notation est également étendue aux substitutions ; ainsi, on notera $M[\overrightarrow{N}^n/\overrightarrow{x}^n]$ les substitutions successives $M[N_1/x_1, \ldots, N_n/x_n]$.

Nouvelles règles de réduction. Nous étendons la relation de réduction (et donc de conversion) aux règles suivantes :

- La règle de réduction qui permet de réduire `case` vers la branche correspondant au constructeur analysé :

$$\texttt{case}_I(c_j[\overrightarrow{P}^p]\,\overrightarrow{M}^{m_j}, \overrightarrow{Q}^p, \lambda\overrightarrow{y : B}^n, i : C.T, \overrightarrow{\lambda\overrightarrow{z : E}^m.F}^k) \rhd F_j[\overrightarrow{M}^{m_j}/\overrightarrow{z}^{m_j}]$$

- La règle de réduction du point-fixe :

$$((\texttt{fix}\, f : A, \overrightarrow{x : B}^{l+1}.M) \quad \overrightarrow{N}^{l+1}) \rhd M[\texttt{fix}\, f : A, \overrightarrow{x : B}^{l+1}.M/f, \overrightarrow{N}^{l+1}/\overrightarrow{x}^{l+1}]$$

Puis, nous définissons également deux notations qui permettent d'instancier l'arité et les constructeurs avec une valeur concrète de paramètres. Dans le cas où I est un inductif $\texttt{Ind}(x_1 : Q_1, \ldots, x_p : Q_p, I : A, c_0 : C_0, \ldots, c_k : C_k)$ on notera :

$$\begin{array}{rcl} \texttt{Arity}_I(\overrightarrow{P}^p) & = & A[\overrightarrow{P}^p/\overrightarrow{x}^p] \\ \texttt{Constr}_I^j(\overrightarrow{P}^p) & = & C_j[\overrightarrow{P}^p/\overrightarrow{x}^p] \end{array}$$

$$p > 0 \vee \Gamma = \langle\rangle \ \frac{(\Gamma \vdash P_i : Q_i[P_1/x_1, \cdots, P_{i-1}/x_{i-1}]\ )_{i=1\ldots p}}{\Gamma \vdash I[\overrightarrow{P}^p] : \mathtt{Arity}_I(\overrightarrow{P}^p)}\ \textsc{Inductif}$$

$$p > 0 \vee \Gamma = \langle\rangle \ \frac{(\Gamma \vdash P_i : Q_i[P_1/x_1, \cdots, P_{i-1}/x_{i-1}]\ )_{i=1\ldots p}}{\Gamma \vdash c_j[\overrightarrow{P}^p] : \mathtt{Constr}_I^j(\overrightarrow{P}^p)}\ \textsc{Constructeur}$$

$$(*)\ \frac{\begin{array}{c}\Gamma \vdash M : I[\overrightarrow{P}^p]\ \overrightarrow{G}^n \qquad \Gamma, \overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\ \overrightarrow{y}^n \vdash T : r \\ \left(\Gamma, \overrightarrow{z : E_j}^{m_j} \vdash F_j : T[\overrightarrow{D_j}^n / \overrightarrow{y}^n, c_j[\overrightarrow{P}^p]\ \overrightarrow{z}^{m_j}/i]\right)_{j=1\ldots k}\end{array}}{\Gamma \vdash \mathtt{case}_I(M, \overrightarrow{P}^p, \lambda\overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\ \overrightarrow{y}^n.T, \overrightarrow{\lambda\overrightarrow{z : E}^{m}.F}^{k}\ ) : T[\overrightarrow{G}^n / \overrightarrow{y}^n, M/i]}\ \textsc{Case}$$

Où (*) est la condition suivante :

$$\begin{array}{rcl}(I, r) & \in & \mathcal{E} \\ k \text{ est le nombre de constructeurs de } I & & \\ \mathtt{Arity}_I(\overrightarrow{P}^p) & = & \forall\overrightarrow{y : B}^n.s \\ \mathtt{Constr}_I^j(\overrightarrow{P}^p) & = & \forall\overrightarrow{z : E_j}^{m_j}.I[\overrightarrow{P}^p]\ D_j\end{array}$$

FIGURE 2.1 – Les règles des types inductifs.

$$f \text{ est gardé}\ \frac{\Gamma, f : A \vdash \lambda\overrightarrow{x : B}^{l+1}.M : A}{\Gamma \vdash \mathtt{fix}\, f : A, \overrightarrow{x : B}^{l+1}.M : A}\ \textsc{Point-Fixe}$$

FIGURE 2.2 – La règle de typage de l'opérateur de point-fixe.

Et on ajoute également les quatre règles de typage présentées dans les figures 2.1 et 2.2. La règle CASE à $k+2$ prémisses est particulièrement difficile à lire, cette règle est paramétrée par $\mathcal{E}$; ce paramètre permet donc de restreindre les utilisation de `case` selon l'inductif détruit et la sorte du type de retour de chacune des branches. Le lecteur qui n'est pas habitué au typage des inductifs pourra commencer par lire la sous-section 2.2 où nous détaillons quelques exemples. Les règles INDUCTIF et CONSTRUCTEUR ont p prémisses, il est de plus nécessaire de rajouter la contrainte $\Gamma = \langle\rangle$ dans le cas où $p = 0$ pour éviter la dérivation de séquent dont le contexte est mal-formé.

Remarque sur les lieurs dans la règle `case`. La syntaxe de la construction `case` peut paraître surprenante au premier abord car elle contient des lieurs explicites. Bien que nous ayons utilisé le symbole λ, la règle de typage CASE ne requiert pas que ces "pseudo-abstractions" soient typables par un produit autorisé. De la même façon, nous pouvons alors voir la notation $c[\cdot]$ comme une "pseudo-application" ne vérifiant pas que le produit $\forall \overrightarrow{x : Q}^p.A$ est autorisé, dans ce type, A désigne l'arité de l'inductif associée au constructeur.

Néanmoins, dans les systèmes qui vérifient la propriété FULL, on peut présenter les inductifs en remplaçant ces "pseudo-applications" et ces "pseudo-abstractions" par des applications et des abstractions puis, en remplaçant la règle CASE par la règle ci-dessous :

$$\frac{\Gamma \vdash M : I\ \overrightarrow{P}^p\ \overrightarrow{G}^n \qquad \Gamma \vdash T : \forall \overrightarrow{y : A}^n.I\ \overrightarrow{P}^p\ \overrightarrow{y}^n \to r \qquad \left(\Gamma \vdash F_j : \forall \overrightarrow{z : E_j}^{m_j}.T\ \overrightarrow{D_j}^n\ (c_j\ \overrightarrow{P}^p \overrightarrow{z}^{m_j})\right)_{j=1\ldots k}}{\Gamma \vdash \mathtt{case}_I(M, \overrightarrow{P}^p, T, \overrightarrow{F}^k) : T\ \overrightarrow{G}^n\ M}$$

Bonne formation des inductifs Les inductifs sont rajoutés au système incrémentalement, c'est pourquoi nous introduisons la définition suivante :

2.1.2 Définition (Ajout d'une déclaration d'inductif)
Soient $\mathcal{P}$ un CTSI et $I \notin \mathcal{I}_\mathcal{P}$, $c_0, \ldots, c_k$ k identifiants frais, et soient Q_0, …, Q_p, A, $C_0, \ldots, C_k$ $p+k+1$ termes de $\mathcal{P}$, on notera

$$\mathcal{P} + \mathtt{Ind}(x_1 : Q_1, \ldots, x_p : Q_p, I : A, c_0 : C_0, \ldots, c_k : C_k)$$

le CTSI $\mathcal{P}'$ obtenu en posant $\mathcal{I}_{\mathcal{P}'} = \mathcal{I}_\mathcal{P} \cup \{I\}$ et en associant dans $\mathcal{P}'$ à I la déclaration :

$$\mathtt{Ind}(x_1 : Q_1, \ldots, x_p : Q_p, I : A, c_0 : C_0, \ldots, c_k : C_k)$$

On dira qu'un CTSI est bien-formé s'il existe un façon d'ordonner les inductifs afin de les définir les uns après les autres de façon à ce que l'arité et les types des constructeurs soient bien typés et de la bonne forme.

2.1.3 Définition (CTSI bien-formé)
- Cas fini : Soit $\mathcal{P}$ un CTSI tel que $\mathcal{I}$ est fini. On définit le fait que $\mathcal{P}$ soit bien formé par récurrence sur la taille de $\mathcal{I}$:
  - Si $\mathcal{I} = \emptyset$, alors $\mathcal{P}$ est bien formé,

– S'il existe $I \in \mathcal{I}$ et $\mathcal{P}'$ bien formé tel que $\mathcal{P}$ se décompose en

$$\mathcal{P} = \mathcal{P}' + \texttt{Ind}(x_1 : Q_1, \ldots, x_p : Q_p, I : A, c_1 : C_1, \ldots, c_k : C_k)$$

et que

1. Si on note Γ_P le contexte $x_1 : Q_1, \ldots, x_p : Q_p$, alors A est un type bien formé de $\mathcal{P}'$ dans le contexte Γ_P (c'est-à-dire $\mathrm{WF}_{\Gamma_P}(A)$).
2. A de la forme $\forall \overrightarrow{y : B}^n.s$, on dira alors que A est une arité de sorte s et que I est un inductif de sorte s.
3. pour tout j, C_j est de la forme $C'_j[I[\overrightarrow{x}^p]/\alpha_I]$ avec I n'apparaissant pas dans C'_j. On demande également à ce que C'_j soit un type bien formé de $\mathcal{P}'$ et de sorte s dans le contexte $\Gamma_P, \alpha_I : A$, c'est-à-dire $\Gamma_P, \alpha_I : A \vdash_{\mathcal{P}'} C'_j : s$. On substitue ici les occurrences de $I[\cdot]$ par une variable de type α_I car l'inductif I n'est pas encore déclaré comme un inductif bien formé.
4. pour tout j, C_j aura la forme
$$\forall \overrightarrow{y : E_j}^{m_j}.I[\overrightarrow{x}^p]\,\overrightarrow{D_j}^n$$
où I ne peut apparaître à l'intérieur de E_j que comme une conclusion. Cette restriction s'appelle la condition de stricte positivité. En pratique, cette condition sera suffisante pour garantir la monotonicité de l'opérateur de type dont on prend implicitement le point fixe lorsque que l'on introduit un nouveau type inductif (voir par exemple [CP88, Wer94, Coq04]).

– Cas général : On dira que $\mathcal{P}$ est bien formé si pour tout $I \in \mathcal{I}$ il existe un ensemble fini $\mathcal{I}' \subseteq \mathcal{I}$ contenant I tel que $\mathcal{P}$ restreint à $\mathcal{I}'$ soit bien formé.

On remarque que notre présentation traite uniquement les inductifs définis dans un contexte vide. Nous pourrions l'étendre facilement pour prendre en compte les définitions des types inductifs dans un contexte non vide, mais cela alourdirait les notations qui sont déjà bien assez complexes. Cette présentation n'en est pas moins générale, puisqu'il est toujours possible d'ajouter le contexte dans les paramètres des types inductifs. De plus, notre cadre ne permet pas de prendre en compte les inductifs mutuellement définis ; cela pourrait aisément être fait au prix d'un nouvel alourdissement des notations (voir par exemple le manuel de Coq [Coq04] pour une présentation des inductifs mutuellement définis).

2.1.4 Convention

Par la suite, nous ferons systématiquement l'hypothèse que les CTSI que l'on considère sont bien formés.

Nous nous permettrons donc d'énoncer des lemmes sans expliciter le fait que nous nous plaçons dans un CTSI bien formé. Comme dans l'exemple suivant :

2.1.5 Lemme

Si $\texttt{Ind}(x_1 : Q_1, \ldots, x_p : Q_p, I : A, c_1 : C_1, \ldots, c_k : C_k)$*, alors*

1. $\mathrm{WF}_{x_1:Q_1,\ldots,x_p:Q_p}(A)$
2. *A est de la forme* $\forall \overrightarrow{y : B}^n.s$

3. *pour tout* j, $x_1 : Q_1, \ldots, x_p : Q_p \vdash C_j : s$,
4. *pour tout* j, C_j *est de la la forme*

$$\forall \overrightarrow{y : E_j}^{m_j}.I[\overrightarrow{x}^p]\,\overrightarrow{D_j}^n$$

où I *ne peut apparaître à l'intérieur de* E_j *que comme une conclusion.*

Démonstration Il s'agit d'une conséquence directe de la définition précédente, et aussi du fait qu'en ajoutant des éléments dans $\mathcal{I}$, on ne type pas moins de termes (on a également besoin d'invoquer le lemme de substitution pour prouver que $x_1 : Q_1, \ldots, x_p : Q_p \vdash_{\mathcal{P}'} C_j : s$ à partir de $x_1 : Q_1, \ldots, x_p : Q_p, \alpha_I : A \vdash_{\mathcal{P}'} C'_j[I[\overrightarrow{x}^p]/\alpha_I] : s$ mais sa preuve ne pose aucun problème). ☺

Condition de garde pour la construction `fix`. On dira que $\texttt{fix}\, f : A, \overrightarrow{x : B}^l, i : C.M$ vérifie la condition de garde si M satisfait un critère syntaxique assurant que :

- C est un type inductif complètement appliqué (c'est-à-dire de la forme $I[\overrightarrow{P}^p]\,\overrightarrow{D}^n$), et
- les appels récursifs à l'intérieur de M sont de la forme $f\,\overrightarrow{T}^k\,(y\,\overrightarrow{Q})$ où y est une variable introduite par une suite de destructions de x.

Moralement y est "structurellement plus petit que x", ce qui devrait assurer la terminaison de la fonction ainsi définie. On trouvera une formalisation plus rigoureuse de cette décroissance dans [Gim94] (bien qu'elle ne soit plus vraiment fidèle aux critères plus généraux implémentés aujourd'hui dans l'assistant de preuve Coq).

2.2 Explications informelles

Dans cette sous-section, nous donnerons quelques exemples pour illustrer la syntaxe et les règles des types inductifs. Le système sous-jacent ici choisi est le Calcul des Constructions Inductives. La seule sorte qui sera explicitement utilisée sera la sorte `Set` et nous reviendrons par la suite sur les spécificités de ce calcul.

L'inversion des listes. Les notations introduites ainsi que les règles que l'on expliquera plus loin sont complexes ; en guise d'illustration on utilisera dans cette sous-section la syntaxe de Coq pour illustrer la syntaxe des CTSI :

```
Inductive list (α : Set) : Set :=
  | nil : list α
  | cons : α → list α → list α.

Fixpoint rev (α : Set) (acc l : list α) : list α :=
  match l return list α with
    | nil  ⇒  acc
    | cons hd tl  ⇒  rev α (cons α hd acc) tl
  end.
```

L'inductif list est un inductif à $p = 1$ paramètre, $k = 2$ constructeurs et $n = 0$ argument, on énoncera sa bonne formation à l'aide du prédicat :

$$\mathtt{Ind}(\alpha : \mathtt{Set}, \mathsf{list} : \mathtt{Set}, \mathsf{nil} : \mathsf{list}[\alpha], \mathsf{cons} : \alpha \to \mathsf{list}[\alpha] \to \mathsf{list}[\alpha])$$

De plus, la fonction rev sera représentée par le point-fixe suivant :

$$\begin{aligned}\mathsf{rev} = \mathtt{fix}\ f : \forall \alpha : \mathtt{Set}\,.\,\mathsf{list}[\alpha] \to \mathsf{list}[\alpha] \to \mathsf{list}[\alpha], \alpha : \mathtt{Set}, acc : \mathsf{list}[\alpha], l : \mathsf{list}[\alpha].\\ \mathtt{case}_{\mathsf{list}}(l, \alpha, \_ : \mathsf{list}[\alpha].\,\mathsf{list}[\alpha], acc, \lambda\, hd : \alpha, tl : \mathsf{list}[\alpha].f\ \alpha\ (\mathsf{cons}[\alpha]\ hd\ acc)\ tl)\end{aligned}$$

Ainsi f représente le nom de la fonction utilisée pour les appels récursifs, f est donc du même type que le point fixe (ici $\forall \alpha : \mathtt{Set}\,.\,\mathsf{list}[\alpha] \to \mathsf{list}[\alpha] \to \mathsf{list}[\alpha]$), α et acc sont les deux arguments "non-structurels" de la fonction et le troisième argument l est celui sur lequel portera la décroissance.

Pour chaque inductif I (ici $I = \mathsf{list}$), la construction $\mathtt{case}_I$ a quatre familles de paramètres :

- Le terme que l'on cherche à détruire (ici l) qui doit être de type $I[\overrightarrow{P}^p]\ \overrightarrow{G}^n$ (ici $\mathsf{list}[\alpha]$).
- La valeur des paramètres $\overrightarrow{P}^p$ (ici α).
- Le type de retour T, qui peut dépendre des arguments, et qui peut aussi dépendre ou non de l'inductif détruit selon la règle de typage utilisée (nous sommes ici en présence d'un cas non dépendant $T = \lambda\_ : \mathsf{list}[\alpha].\,\mathsf{list}[\alpha]$ est une fonction constante).
- Les k branches de l'analyse par cas sont représentées par k fonctions dépendantes des arguments de chacune des constructions (ici $k = 2$, $F_1 = acc$ ne dépend d'aucun argument c'est le cas correspondant au constructeur nil et $F_2 = \lambda\, hd : \alpha, tl : \mathsf{list}[\alpha].f\ \alpha\ (\mathsf{cons}[\alpha]\ hd\ acc)\ tl$ dépend des deux arguments correspondant au constructeur cons).

Ce point-fixe satisfait la condition de garde (l'appel récursif se fait sur la queue de la liste passée en argument). La fonction retourne la liste qu'elle accepte en argument puis elle concatène l'accumulateur. Voici un exemple de réduction utilisant les deux nouvelles règles introduites :

$$\begin{aligned}\mathsf{rev}\ \alpha\ []\ [x; y] &\rhd \mathtt{case}_{\mathsf{list}}([x; y], \alpha, \mathsf{list}[\alpha], [], \lambda\, hd : \alpha, tl : \mathsf{list}[\alpha].\,\mathsf{rev}\ \alpha\ [hd]\ tl)\\ &\rhd \mathsf{rev}\ \alpha\ [x]\ [y]\\ &\rhd \mathtt{case}_{\mathsf{list}}([y], \alpha, \mathsf{list}[\alpha], [x], \lambda\, hd : \alpha, tl : \mathsf{list}[\alpha].\,\mathsf{rev}\ \alpha\ [hd; x]\ tl)\\ &\rhd \mathsf{rev}\ \alpha\ [y; x]\ []\\ &\rhd \mathtt{case}_{\mathsf{list}}([], \alpha, \mathsf{list}[\alpha], [y; x], \lambda\, hd : \alpha, tl : \mathsf{list}[\alpha].\,\mathsf{rev}\ \alpha\ [hd; x]\ tl)\\ &\rhd [y; x]\end{aligned}$$

où l'on note $[x_1; \ldots; x_n]$ le terme $\mathsf{cons}[\alpha]\, x_1\, (\mathsf{cons}[\alpha]\, x_2 \ldots (\mathsf{cons}[\alpha]\, x_n\ \mathsf{nil}[\alpha] \ldots))$ dans un contexte où les x_i sont de type α.

Si l'on note Γ le contexte

$$\Gamma = f : \forall \alpha : \mathtt{Set}\,.\,\mathsf{list}[\alpha] \to \mathsf{list}[\alpha] \to \mathsf{list}[\alpha], \alpha : \mathtt{Set}, acc : \mathsf{list}[\alpha], l : \mathsf{list}[\alpha]$$

alors nous prouvons le séquent typage suivant en utilisant la règle Case

$$\Gamma \vdash \mathtt{case}_{\mathsf{list}}(l, \alpha, \lambda\_ : \mathsf{list}[\alpha].\,\mathsf{list}[\alpha], acc, \lambda\, hd : \alpha, tl : \mathsf{list}[\alpha].f\ \alpha\ (\mathsf{cons}[\alpha]\ hd\ acc)\ tl)$$

en nous assurant que $(\mathsf{list}, \mathtt{Set}) \in \mathcal{E}$ puis en prouvant les quatre prémisses suivantes :

1. $\Gamma \vdash l : \mathsf{list}[\alpha]$
2. $\Gamma, \_ : \mathsf{list}[\alpha] \vdash \mathsf{list}[\alpha] : \mathtt{Set}$
3. $\Gamma \vdash acc : \mathsf{list}[\alpha]$
4. $\Gamma, hd : \alpha, tl : \mathsf{list}[\alpha] \vdash f\ \alpha\ (\mathsf{cons}[\alpha]\ hd\ acc)\ tl : \mathsf{list}[\alpha]$

Autres exemples de types inductifs. Parmi les exemples les plus simples de types inductifs, on trouve les énumérations comme par exemple le type des booléens. Dans notre syntaxe il est représenté par $\mathtt{Ind}(\mathsf{bool} : \mathtt{Set}, \mathsf{true} : \mathsf{bool}, \mathsf{false} : \mathsf{bool})$; c'est un type inductif avec zéro paramètre, zéro argument et deux constructeurs sans argument. Dans le système Coq il est défini par le script suivant :

```
Inductive bool : Set :=
  | true : bool
  | false : bool.
```

Viennent ensuite les types récursifs, qui correspondent aux types inductifs sans arguments, comme par exemple, le type des listes vu précédemment, ou bien encore, le type des entiers

$$\mathtt{Ind}(\mathsf{nat} : \mathtt{Set}, \mathsf{zero} : \mathsf{nat}, \mathsf{succ} : \mathsf{nat} \to \mathsf{nat})$$

représenté en Coq par :

```
Inductive nat : Set :=
  | zero : nat
  | succ : nat→ nat.
```

Un cas particulièrement dégénéré de type inductif est le type vide $\mathtt{Ind}(\mathsf{empty} : \mathtt{Set})$ représenté en Coq par :

```
Inductive empty : Set :=.
```

On peut alors utiliser `case` pour détruire n'importe quel habitant de empty et déduire un habitant de n'importe quel type. C'est pourquoi empty est aux types de données ce que l'absurdité est aux propositions. Ainsi le terme $\lambda\alpha : \mathtt{Set}, b : \mathsf{empty}\,.\,\mathtt{case}_{\mathsf{empty}}(b, \lambda\_ : \mathsf{empty}\,.\alpha)$ est bien de type $\forall\alpha : \mathtt{Set}, \mathsf{empty} \to \alpha$ et est implémenté en Coq par :

```
Definition Absurd (α : Set) (x : empty) : α :=
  match x return α with end.
```

On prouve le séquent de typage $\alpha : \mathtt{Set}, b : \mathsf{empty} \vdash \mathtt{case}_{\mathsf{empty}}(b, \lambda\_ : \mathsf{empty}\,.\alpha) : \alpha$ par la dérivation de typage suivante :

$$\frac{\alpha : \mathtt{Set}, b : \mathsf{empty} \vdash b : \mathsf{empty} \qquad \alpha : \mathtt{Set}, b : \mathsf{empty}, \_ : \mathsf{empty} \vdash \alpha : \mathtt{Set}}{\alpha : \mathtt{Set}, b : \mathsf{empty} \vdash \mathtt{case}_{\mathsf{empty}}(b, \lambda\_ : \mathsf{empty}) : \alpha}\ (\mathsf{empty}, \mathtt{Set}) \in \mathcal{E}$$

Voici notre premier exemple de type inductif avec un argument (et zéro paramètre) :

```
Inductive fin : nat→ Set :=
  | fin_zero : forall n:nat, fin n
  | fin_succ : forall n:nat, fin n→ fin (succ n).
```

Ce type inductif permet de représenter les entiers plus petits que son argument. Ainsi, fin_zero représente l'entier 0 et il est bien plus petit que tous les entiers. Le constructeur fin_succ retourne le représentant d'un entier plus petit que $n+1$ lorsqu'on lui passe en argument un entier plus petit que n. Il est représenté dans notre syntaxe par la déclaration suivante :

$$\texttt{Ind}(\mathsf{fin} : \mathsf{nat} \to \texttt{Set}, \mathsf{fin\_zero} : \forall n : \mathsf{nat}\,.\,\mathsf{vect}\ n, \mathsf{fin\_succ} : \forall n : \mathsf{nat}\,.\,\mathsf{fin}\ n \to \mathsf{fin}\,(\mathsf{succ}\ n))$$

À titre d'exemple, on peut programmer les deux injections réciproques suivantes :

```
Fixpoint fin_nat (n : nat) (x : fin n) : nat :=
  match x return nat with
    | fin_zero _  ⇒   zero
    | fin_succ k p  ⇒   succ (fin_nat k p)
  end.

Fixpoint nat_fin (n : nat) : fin n :=
 (* Elimination dependante : *)
  match n as m return fin m with
    | zero  ⇒   fin_zero zero
    | succ p  ⇒   fin_succ p (nat_fin p)
  end.
```

La première se traduit par :

$$\begin{array}{l}\texttt{fix}\, f : \forall n : \mathsf{nat}\,.\,\mathsf{fin}\ n \to \mathsf{nat}, n : \mathsf{nat}, x : \mathsf{fin}\ n.\\ \quad \mathsf{case}_{\mathsf{fin}\ n}\,(x, \lambda\_ : \mathsf{fin}\ n.\,\mathsf{nat},\\ \qquad \lambda\_ : \mathsf{nat}\,.\,\mathsf{zero},\\ \qquad \lambda k : \mathsf{nat}, p : \mathsf{fin}\ k.\,\mathsf{succ}\,(f\,k\,p))\end{array}$$

Et la seconde constitue notre premier exemple dans lequel le type des branches dépend de la valeur analysée :

$$\begin{array}{l}\texttt{fix}\, f : \forall n : \mathsf{nat}\,.\,\mathsf{fin}\ n, n : \mathsf{nat}\,.\\ \quad \mathsf{case}_{\mathsf{nat}}(n, \lambda x : \mathsf{nat}\,.\,\mathsf{fin}\ x,\\ \qquad \mathsf{fin\_zero}\ \mathsf{zero},\\ \qquad \lambda p : \mathsf{nat}\,.\,\mathsf{fin\_succ}\ p\ (f\,p))\end{array}$$

On prouve le séquent de typage

$$\Gamma \vdash \mathsf{case}_{\mathsf{nat}}(n, \lambda x : \mathsf{nat}\,.\,\mathsf{fin}\ x, \mathsf{fin\_zero}\ \mathsf{zero}, \lambda p : \mathsf{nat}\,.\,\mathsf{fin\_succ}\ p\ (f\,p))$$

où $\Gamma = f : \forall n : \mathsf{nat}\,.\,\mathsf{fin}\ n, n : \mathsf{nat}$ en vérifiant que $(\mathsf{nat}, \texttt{Set}) \in \mathcal{E}$ et les quatre prémisses suivantes :

1. $\Gamma \vdash n : \mathsf{nat}$,
2. $\Gamma, x : \mathsf{nat} \vdash \mathsf{fin}\ x : \texttt{Set}$,
3. $\Gamma \vdash \mathsf{fin\_zero}\ \mathsf{zero} : \mathsf{fin}\,(\mathsf{zero})$,
4. $\Gamma, p : \mathsf{nat} \vdash \mathsf{fin\_succ}\ p\,(f\,p) : \mathsf{fin}\,(\mathsf{succ}\ p)$.

Nous conseillons ici au lecteur de se rapporter à la règle Case pour bien comprendre en quoi ces quatre séquents sont des instances des prémisses de la règle.

Voici un autre inductif avec un argument (et un paramètre) qui est à fin ce que list est à nat :

```
Inductive vect (α : Set) : nat→ Set :=
  | nilV : vect zero
  | consV : α → forall (n : nat) (v : vect α n), vect α (succ n).
```

Il permet de représenter les listes de taille fixée, ainsi les habitants (en forme normale) du type vect n sont tous construits avec exactement n constructeurs consV. Les types dépendants permettent donc d'écrire dans le type de données des informations sur leur structure. On peut se servir de ces informations pour contraindre le domaine des fonctions aux endroits où elles sont bien définies. Ainsi, on peut par exemple écrire le type

$$\forall \alpha : \mathsf{Set}, n : \mathsf{nat} \,.\, \mathsf{vect}[\alpha]\,(\mathsf{succ}\ n) \to \mathsf{vect}[\alpha]\,n$$

que l'on peut donner à la fonction qui prend en paramètre un vecteur de taille strictement positive et retourne ce même vecteur, privé de son premier élément. Mais, pour implémenter cette fonction, nous avons tout d'abord besoin de comprendre ce qu'est l'élimination forte.

Élimination forte. On parle d'élimination forte lorsque l'on utilise un `case` pour construire un objet dont la sorte du type n'est pas minimale pour la relation $\mathcal{A}$. On peut par exemple décrire un constructeur de type qui dépend d'une valeur booléenne

$$\lambda b : \mathsf{bool} \,.\, \mathtt{case}_{\mathsf{bool}}(b, \lambda\_ : \mathsf{bool} \,.\, \mathtt{Set}, \mathsf{nat}, \mathsf{bool})$$

ce qui donne en Coq :

```
Definition NatOrBool (b : bool) : Set :=
  match b return Set with
    | true  ⇒  nat
    | false ⇒  bool
  end.
```

Si on note `Type` le type de `Set`, alors on vérifie le séquent

$$b : \mathsf{bool} \vdash \mathtt{case}_{\mathsf{bool}}(b, \lambda\_ : \mathsf{bool} \,.\, \mathtt{Set}, \mathsf{nat}, \mathsf{bool}) : \mathtt{Set}$$

en s'assurant que $(\mathsf{bool}, \mathtt{Type}) \in \mathcal{E}$ puis en vérifiant les quatre prémisses suivantes :

1. $b : \mathsf{bool} \vdash b : \mathsf{bool}$,
2. $b : \mathsf{bool}, \_ : \mathsf{bool} \vdash \mathtt{Set} : \mathtt{Type}$,
3. $b : \mathsf{bool} \vdash \mathsf{nat} : \mathtt{Set}$,
4. $b : \mathsf{bool} \vdash \mathsf{bool} : \mathtt{Set}$.

C'est bien parce que la sorte du type de retour est `Type`, et non une sorte minimale comme `Set`, que l'on a affaire à une élimination forte. On verra plus avant, qu'il est souvent nécessaire d'introduire des restrictions, sur la forme de l'inductif, pour contraindre ces éliminations, sous peine d'obtenir un système incohérent.

Ainsi `NatOrBool` : bool → `Set` retourne nat ou le type bool selon que son argument soit true ou false. On peut ensuite se servir de ce constructeur de type pour implémenter une fonction de type $\forall b : \mathsf{bool}\,.\texttt{NatOrBool}\,b$ comme par exemple la fonction

$$\lambda b : \mathsf{bool}\,.\,\mathsf{case}_{\mathsf{bool}}(b, \lambda x : \mathsf{bool}\,.\texttt{NatOrBool}\,x, \mathsf{zero}, \mathsf{true})$$

qui s'implémente en Coq de la façon suivante :

```
Definition f (b : bool) : NatOrBool b :=
  match b as x return NatOrBool x with
    | true  ⇒  zero
    | false ⇒  true
  end.
```

On prouve alors le séquent

$$b : \mathsf{bool} \vdash \mathsf{case}_{\mathsf{bool}}(b, \texttt{NatOrBool}, \mathsf{zero}, \mathsf{true}) : \texttt{NatOrBool}\,b$$

en vérifiant les quatre prémisses suivantes et $(\mathsf{nat}, \texttt{Set}) \in \mathcal{E}$:

1. $b : \mathsf{bool} \vdash b : \mathsf{bool}$
2. $b : \mathsf{bool}, x : \mathsf{bool} \vdash \texttt{NatOrBool}\,x : \texttt{Set}$
3. $b : \mathsf{bool} \vdash \mathsf{zero} : \texttt{NatOrBool}\ \mathsf{true}$
4. $b : \mathsf{bool} \vdash \mathsf{true} : \texttt{NatOrBool}\ \mathsf{false}$

Nous avons maintenant les outils pour implémenter la fonction vect_tail qui récupère la queue d'un vecteur :

```
Inductive unit : Set :=
  | I : unit.

Definition vect_tail
      (α : Set) (n : nat) (v : vect α (succ n)) : vect α n :=
  match v in vect _ m return
        match m return Set with
          | zero  ⇒  unit
          | succ p  ⇒  vect α p
        end
  with
    | nilV  ⇒  I
    | consV _ k tl  ⇒  tl
  end.
```

Nous voyons pour la première fois dans ce script la notation utilisée par Coq pour lier les arguments (ici m) dans le type de retour. Le type de retour du `case` le plus superficiel est lui-même défini par un `case` sur la valeur de l'argument m du type de v. Si cet argument est nul, il retourne un type dont on connaît un habitant (nous utilisons ici unit) et s'il est de la forme $\mathsf{succ}\,p$, il retourne le type $\mathsf{vect}[\alpha]\,p$.

Ainsi, dans le cas nilV, il nous suffit de retourner un habitant de unit, et dans le cas consV, on peut retourner tl.

Cette définition s'écrit de la manière suivante dans notre syntaxe :

$$\begin{array}{l}\lambda\alpha : \mathtt{Set}, n : \mathsf{nat}, v : \ \mathsf{vect}[\alpha]\,(\mathsf{succ}\ n).\\ \quad \mathsf{case}_{\mathsf{vect}}(v, \alpha, \lambda m : \mathsf{nat}, \_ : \mathsf{vect}[\alpha]\, m.\\ \qquad \mathsf{case}_{\mathsf{nat}}(m, \lambda\_ : \mathsf{nat}\,.\,\mathtt{Set}, \mathsf{unit}, \lambda p : \mathsf{nat}\,.\,\mathsf{vect}[\alpha]\, p),\\ \quad \mathtt{I},\\ \quad \lambda\_ : \alpha, k : \mathsf{nat}, tl : \mathsf{vect}[\alpha]\, k.\, tl)\end{array}$$

Si l'on note $\Gamma = \alpha : \mathtt{Set}, n : \mathsf{nat}, v : \mathsf{vect}[\alpha]\,(\mathsf{succ}\ n)$, alors pour vérifier le séquent :

$$\Gamma \vdash \left(\begin{array}{l}\mathsf{case}_{\mathsf{vect}}(v, \alpha, \lambda m : \mathsf{nat}, \_ : \mathsf{vect}[\alpha]\, m.\\ \quad \mathsf{case}_{\mathsf{nat}}(m, \lambda x : \mathsf{nat}\,.\,\mathtt{Set}, \mathsf{unit}, \lambda p : \mathsf{nat}\,.\,\mathsf{vect}[\alpha]\, p),\\ \mathtt{I},\\ \lambda\, hd : \alpha, k : \mathsf{nat}, tl : \mathsf{vect}[\alpha]\, k.\, tl)\end{array}\right) : \mathsf{vect}[\alpha]\, n$$

il suffit de vérifier les quatre prémisses suivantes :

1. $\Gamma \vdash v : \ \mathsf{vect}[\alpha]\,(\mathsf{succ}\ n)$,
2. $\Gamma, m : \mathsf{nat}, x : \mathsf{vect}[\alpha]\, m \vdash T : \mathtt{Set}$,
3. $\Gamma \vdash \mathtt{I} : T[\mathsf{zero}\,/m, \mathsf{nilV}[\alpha]/x]$,
4. $\Gamma, hd : \alpha, k : \mathsf{nat}, tl : \mathsf{vect}[\alpha]\, k \vdash tl : T[\mathsf{succ}\ k/m, \mathsf{consV}[\alpha]\ hd\ k\ tl\,/x]$.

et que $T[\mathsf{succ}\ n/m, v/x] \equiv \mathsf{vect}[\alpha]\, n$ où $T = \mathsf{case}_{\mathsf{nat}}(m, \lambda x : \mathsf{nat}\,.\,\mathtt{Set}, \mathsf{unit}, \lambda p : \mathsf{nat}\,.\,\mathsf{vect}[\alpha]\, p)$. Ces vérifications sont aisées une fois remarqué que :

$$\begin{array}{rcl} T[\mathsf{zero}\,/m, \mathsf{nilV}[\alpha]/x] & \rhd & \mathsf{unit} \\ T[\mathsf{succ}\ k/m, \mathsf{consV}[\alpha]\ hd\ k\ tl\,/x] & \rhd & \mathsf{vect}[\alpha]\, k \\ T[\mathsf{succ}\ n/m, v/x] \equiv \mathsf{vect}[\alpha]\, n & \rhd & \mathsf{vect}[\alpha]\, n \end{array}$$

Le lecteur qui souhaiterait s'exercer à la manipulation des règles d'élimination dépendante peut essayer de construire un terme de type $\forall\alpha : \mathtt{Set}, n : \mathsf{nat}\,.\,\mathsf{fin}\,(\mathsf{succ}\, n) \to \mathsf{vect}[\alpha]\, n \to \alpha$ qui retourne le k-ième élément d'un vecteur de taille n avec $k < n$.

Retour sur la condition de stricte positivité. La condition de stricte positivité esquissée plus haut permet d'évacuer certains inductifs problématiques. L'exemple emblématique de type inductif qui ne satisfait pas la condition de stricte positivité est l'encodage des termes du λ-calcul pur, par la technique nommé "syntaxe abstraite d'ordre supérieure" (Higher-Order Abstract Syntax en anglais, souvent abrégé en HOAS) [PE88]. Avec la syntaxe de Coq, cette représentation peut se traduire par le script ci-dessous.

```
Inductive lam : Set :=
  | abs : (lam→ lam)→ lam.

Definition apply (s t : lam) :=
```

```
  match s with
    | abs f  ⇒  f t
  end.

Definition δ := abs (fun x→ apply x x).
Definition ω := apply δ δ.
```

On remarque que si la définition de `lam` était acceptée, on aurait à disposition un terme ω tel que $\omega \triangleright \omega$. donc que le système ne serait pas normalisant.

2.3 Métathéorie

Dans cette sous section, nous énoncerons la façon dont la métathéorie des PTS peut s'étendre aux types inductifs. Mais, à l'heure de la rédaction, tous les détails n'ont pas encore été vérifiés. On admettra en particulier la confluence du calcul (ci-dessous), ainsi que le lemme 2.3.8.

2.3.1 Lemme (Propriété de Church-Rosser avec type inductifs)
Si $A \equiv A'$ alors il existe C tel que $A \trianglerighteq C$ et $A' \trianglerighteq C$.

Les propriétés métathéoriques suivantes s'étendent aux CTSI :

2.3.2 Lemme (Métathéorie des CTSI)
Les lemmes suivants restent vrais en présence d'inductifs :

- *Le lemme 1.1.23 (substitution) :*

$$\Gamma, x : A, \Delta \vdash M : B \quad \wedge \quad \Gamma \vdash N : A \qquad \Rightarrow \qquad \Gamma, \Delta[N/x] \vdash M[N/x] : B[N/x]$$

- *Le lemme 1.1.24 (affaiblissement généralisé) : Pour x frais,*

$$\Gamma, \Delta \vdash M : A \quad \wedge \quad \Gamma \vdash B : s \qquad \Rightarrow \qquad \Gamma, x : B, \Delta \vdash M : A$$

Démonstration Les deux énoncés se prouvent par induction. Les seuls cas intéressants de ces démonstrations sont les cas Variable, et Axiome et Affaiblissement. Tous les autres cas, y compris ceux introduits par les règles pour les inductifs se traitent simplement en propageant les hypothèse de récurrence, puis en réappliquant la règle en question. ☺

Pour le lemme d'inversion, on procède de la même façon : on remonte les affaiblissements et les règles de cumulativité jusqu'à arriver à la règle "active". L'énoncé du lemme d'inversion s'obtient donc en lisant à l'envers la règle d'inférence concernée :

2.3.3 Lemme (Inversion)
Le lemme 1.1.32 s'étend aux types inductifs de la façon suivante :

1. *Si $\Gamma \vdash c[\overrightarrow{P}^p] : Z$, alors*
   - WF(Γ),
   - *Pour tout $1 \leq i \leq p$, $\Gamma \vdash P_i : Q_i[P_1/x_1, \cdots, P_{i-1}/x_{i-1}]$.*
   - *Il existe un entier j tel que $c = c_j$, et $\mathtt{Constr}_I^j(\overrightarrow{P}^p) \preccurlyeq Z$,*

2. *Si* $\Gamma \vdash I[\overrightarrow{P}^p] : Z$, *alors*
   - $\mathrm{WF}(\Gamma)$,
   - *Pour tout* $1 \leq i \leq p$, $\Gamma \vdash P_i : Q_i[P_1/x_1, \cdots, P_{i-1}/x_{i-1}]$.
   - $\mathtt{Arity}_I(\overrightarrow{P}^p) \preccurlyeq Z$,
3. *Si* $\Gamma \vdash \mathtt{case}_I(M, \overrightarrow{P}^p, \lambda\overrightarrow{y : A}^n, i : I[\overrightarrow{P}^p]\,\overrightarrow{y}^n.T, \overrightarrow{F}^k) : Z$, *alors on a une famille de* n *termes* $\overrightarrow{G}^n$ *telle que :*
   - $(I, r) \in \mathcal{E}$,
   - I *est un inductif à* k *constructeurs,*
   - $\mathtt{Arity}_I(\overrightarrow{P}^p) = \forall\overrightarrow{y : B}^n.s$,
   - $\mathtt{Constr}_I^j(\overrightarrow{P}^p) = \forall\overrightarrow{z : E_j}^{m_j}.I[\overrightarrow{P}^p]\,D_j$,
   - $\Gamma \vdash M : I[\overrightarrow{P}^p]\,\overrightarrow{G}^n$,
   - $\Gamma, \overrightarrow{y : A}^n, i : I[\overrightarrow{P}^p]\,\overrightarrow{y}^n \vdash T : r$,
   - *Pour tout* $1 \leq i \leq k$ *:*
   $$\Gamma, \overrightarrow{z : E_j}^{m_j} \vdash F_j : T[\overrightarrow{D_j}^n/\overrightarrow{z}^n, c_j[\overrightarrow{P}^p]\,\overrightarrow{z}^{m_j}/i]$$
   - $T[\overrightarrow{G}^n.\overrightarrow{y}^m, M/i] \preccurlyeq Z$.
4. *Si* $\Gamma \vdash \mathtt{fix}\, f : A, \overrightarrow{x : B}^{l+1}.M : T$, *alors il existe une sorte* s *telle que*
   - $\Gamma \vdash A : s$,
   - $\Gamma, f : A \vdash \lambda\overrightarrow{x : B}^{l+1}.M : A$,
   - $A \preccurlyeq T$.

Démonstration La preuve s'esquisse de manière similaire à celle du lemme 1.1.32. ☺

2.3.4 Lemme
Si I *est inductif de sorte* s $\mathtt{Ind}(x_1 : Q_1, \ldots, x_p : Q_p, I : A, c_1 : C_1, \ldots, c_k : C_k)$ *(*s *est donc la conclusion de* A*), et si on dispose d'une suite* $P_1, \ldots, P_p$ *et d'un contexte* Γ *tels que pour tout* $1 \leq k \leq p$, $\Gamma \vdash P_k : Q_k$. *Alors,*
- $\mathrm{WF}_\Gamma(\mathtt{Arity}_I(\overrightarrow{P}^p))$
- $\Gamma \vdash \mathtt{Constr}_I^j(\overrightarrow{P}^p) : s$

Démonstration Si $p = 0$, il n'y rien à démontrer, on supposera donc que $p > 0$. D'après le lemme 2.1.5, on a :

1. $\mathrm{WF}_{x_1:Q_1,\ldots,x_p:Q_p}(A)$,
2. pour tout j, $x_1 : Q_1, \ldots, x_p : Q_p \vdash C_j : s$. ☺

Or d'après le lemme précédent $\Gamma \vdash P_1 : Q_1$ implique que $\mathrm{WF}(\Gamma)$ et donc d'après l'affaiblissement généralisée et on montre que $\mathrm{WF}_{\Gamma,x_1:Q_1,\ldots,x_p:Q_p}(A)$ et $\Gamma, x_1 : Q_1, \ldots, x_p : Q_p \vdash C_j : s$, puis par substitutions successives on en déduit que

$$\mathrm{WF}_\Gamma(A[\overrightarrow{P}^p/x]) \qquad \text{et} \qquad \Gamma \vdash C_j[\overrightarrow{P}^p/\overrightarrow{x}^p] : s$$

Or, par définition on a :

$$\mathtt{Arity}_I(\overrightarrow{P}^p) = A[\overrightarrow{P}^p/x] \qquad \text{et} \qquad \mathtt{Constr}_I^j(\overrightarrow{P}^p) = C_j[\overrightarrow{P}^p/\overrightarrow{x}^p]$$

On en déduit alors que :

2.3.5 Lemme
Le lemme 1.1.35 reste vrai dans les CTSI :

$$\Gamma \vdash M : A \qquad \Rightarrow \qquad \mathrm{WF}_\Gamma(A)$$

Démonstration Par induction sur la dérivation de $\Gamma \vdash M : A$. On traite les cas Point-Fixe, Case, Constructeur, et Inductif :

- Point-Fixe : On a M est de la forme $\mathtt{fix}\, f : A, \overrightarrow{x : B}^{l+1}.N$ et $\Gamma, f : A \vdash \lambda \overrightarrow{x : B}^{l+1}.N : A$, et donc $\mathrm{WF}(\Gamma, f : A)$, ce qui implique $\mathrm{WF}_\Gamma(A)$.
- Constructeur et Inductif : Conséquence directe du lemme précédent.
- Case : On a M de la forme

  $$\mathtt{case}_I(N, \overrightarrow{P}^p, \lambda \overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\, \overrightarrow{y}^n.T, \overrightarrow{\lambda \overrightarrow{z : E}^m.F}^k)$$

  et A de la forme $T[\overrightarrow{G}^n/\overrightarrow{y}^n, N/i]$ avec :
  - (1) : $\Gamma \vdash N : I[\overrightarrow{P}^p]\, \overrightarrow{G}^n$
  - (2) : $\Gamma, \overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\, \overrightarrow{y}^n \vdash T : r$

  Donc pour prouver que A est bien formé, on peut utiliser le lemme de substitution appliqué à (2). Pour pouvoir l'appliquer de façon successives, il faut montrer que

  $$(H_k) : \qquad \Gamma \vdash G_k : B_k[G_1/y_1, \ldots, G_{k-1}/y_{k-1}] \text{pour } 1 \leq k \leq n$$

  et enfin il faut montrer $\Gamma \vdash N : I[\overrightarrow{P}^p]\, \overrightarrow{G}^n$, ce qui ne pose pas de problème puisque c'est (1). Or par hypothèse d'induction, on obtient qu'il existe s' telle que $\Gamma \vdash I[\overrightarrow{P}^p]\, \overrightarrow{G}^n : s'$ (3). Or par n inversions successives à partir de (3), on peut déduire $H_n, \ldots, H_1$. ☺

Le lemme technique suivant sera utile pour démontrer la préservation du typage :

2.3.6 Lemme
Si $\Gamma \vdash c_j[\overrightarrow{P}^p]\, \overrightarrow{M}^{m_j} : I[\overrightarrow{P}^p]\, \overrightarrow{G}^n$ *et si*

$$\begin{aligned} \mathtt{Arity}_I(\overrightarrow{P}^p) &= \forall \overrightarrow{y : B}^n.s \\ \mathtt{Constr}_I^j(\overrightarrow{P}^p) &= \forall \overrightarrow{z : E_j}^{m_j}.I[\overrightarrow{P}^p]\, D_j \end{aligned}$$

alors

1. $\overrightarrow{G}^n \equiv \overrightarrow{D_j}^n[\overrightarrow{M}^{m_j}/\overrightarrow{z}^{m_j}]$,
2. $\Gamma \vdash M_l : E_j[M_1/z_1, \ldots, M_{l-1}/z_{l-1}]$ *pour* $1 \leq l \leq m_j$.

Démonstration Le cas général étant similaire, nous allons prouver le lemme dans le cas particulier $m_j = 2$. On a donc $\Gamma \vdash c_j[\overrightarrow{P}^p]\, M_1\, M_2 : I[\overrightarrow{P}^p]\, \overrightarrow{G}^n$. Par inversion, on obtient qu'il existe A et B, tels que :

- $(H_1) : \Gamma \vdash c_j[\overrightarrow{P}^p]\, M_1 : \forall z_2 : A.B$,

– $(H_2) : \Gamma \vdash M_2 : A$,
– $(H_3) : B[M_2/z_2] \preccurlyeq I[\overrightarrow{P}^p]\,\overrightarrow{G}^n$.

Par inversion (H_1), on obtient qu'il existe A' et B' tels que :

– $(H'_1) : \Gamma \vdash c_j[\overrightarrow{P}^p] : \forall z_1 : A'.B'$,
– $(H'_2) : \Gamma \vdash M_1 : A'$,
– $(H'_3) : B'[M_1/z_1] \preccurlyeq \forall y : A.B$.

Enfin, par inversion de (H'_1), on dispose de :

– $\mathrm{WF}(\Gamma)$,
– Pour tout $1 \leq i \leq p$, $\Gamma \vdash P_i : Q_i$.
– Il existe un entier j tel que $c = c_j$, et :

$$\forall z_1 : E_{j,1}, z_2 : E_{j,2}.I[\overrightarrow{P}^p]\,D_j \preccurlyeq \forall z_1 : A'.B'$$

et on en déduit que $E_{j,1} \equiv A'$ et

$$\forall z_2 : E_{j,2}.I[\overrightarrow{P}^p]\,\overrightarrow{D_j}^n \preccurlyeq B'$$

et donc que :

$$\forall z_2 : E_{j,2}[M_1/z_1].\left(I[\overrightarrow{P}^p]\,\overrightarrow{D_j}^n\right)[M_1/z_1] \preccurlyeq B'[M_1/z_1] \preccurlyeq \forall z_2 : A.B$$

ainsi nous obtenons :

$$E_{j,2}[M_1/z_1] \equiv A \quad \text{et} \quad \left(I[\overrightarrow{P}^p]\,\overrightarrow{D_j}^n\right)[M_1/z_1] \preccurlyeq B'[M_1/z_1] \preccurlyeq B$$

Comme $z_1, z_2 \notin \mathcal{FV}(\overrightarrow{P}^p)$, on a $\overrightarrow{P}^p[M_1/z_1, M_2/z_2] = \overrightarrow{P}^p$ et donc :

$$I[\overrightarrow{P}^p]\,\overrightarrow{D_j}^n[M_1/z_1, M_2/z_2] \preccurlyeq B[M_2/z_2] \preccurlyeq I[\overrightarrow{P}^p]\,\overrightarrow{G}^n$$

On a donc bien $\overrightarrow{D_j}^n[P/x, M/y] \equiv \overrightarrow{G}^n$. Il reste donc à prouver que $\Gamma \vdash M_1 : E_{j,1}$ et $\Gamma \vdash M_2[M_1/z_1] : E_{j,2}[M_1/z_1]$. Afin d'y parvenir, on peut invoquer le lemme 2.3.4 pour montrer que $\Gamma \vdash \mathtt{Constr}_I^j(\overrightarrow{P}^p) : s$, c'est-à-dire

$$\Gamma \vdash \forall z_1 : E_{j,1}, z_2 : E_{j,2}.I[\overrightarrow{P}^p]\,D_j : s$$

Or, on en déduit donc aisément par inversion que $\mathrm{WF}_\Gamma(E_{j,1})$ et $\mathrm{WF}_{\Gamma,z_1:E_{j,1}}(E_{j,2})$. On peut alors utiliser la cumulativité pour prouver que $\Gamma \vdash M_1 : E_{j,1}$ à partir de $A' \equiv E_{j,1}$ et (H'_2). Puis par substitution, dans $\mathrm{WF}_{\Gamma,z_1:E_{j,1}}(E_{j,2})$, on montre que $\mathrm{WF}_\Gamma(E_{j,2}[M_1/z_1])$. Nous sommes donc autorisé à utiliser les cumulativité avec $E_{j,2}[M_2/z_2] \equiv A$ et $\Gamma \vdash M_2 : A$ pour déduire $\Gamma \vdash M_2 : E_{j,2}[M_2/z_2]$. ☺

2.3.7 Lemme (Préservation du typage par réduction)
La préservation du typage des CTS s'étend aux CTSI :

$$\textit{Si } \Gamma \vdash M : A \textit{ et } M \rhd M', \textit{ alors } \Gamma \vdash M' : A.$$

Démonstration Les nouvelles règles de réduction introduites pour réduire `case` et `fix` nous contraignent à traiter deux nouveaux cas clefs :

– Supposons que la dernière règle appliquée dans la dérivation soit une application de la règle Case de la forme ci-dessous (il y a $k+2$ prémisses numérotés $(1), (2), \ldots, (k+2)$) :

$$\frac{\begin{array}{ccc}(1) & & (2)\\ \Gamma \vdash c_j[\overrightarrow{P'}^p]\,\overrightarrow{M}^{m_j} : I[\overrightarrow{P}^p]\,\overrightarrow{G}^n & & \Gamma, \overrightarrow{y : A}^n, i : I[\overrightarrow{P}^p]\,\overrightarrow{y}^n \vdash T : r\\ & (j+2) & \\ & \left(\Gamma, \overrightarrow{z : E_j}^{m_j} \vdash F_j : T[\overrightarrow{D_j}^n/\overrightarrow{y}^n, c_j[\overrightarrow{P}^p]\,\overrightarrow{z}^{m_j}/i]\right)_{j=1\ldots k} & \end{array}}{\Gamma \vdash \mathtt{case}_I(c_j[\overrightarrow{P'}^p]\,\overrightarrow{M}^{m_j}, \overrightarrow{P}^p, \lambda\overrightarrow{y : A}^n, i : I[\overrightarrow{P}^p]\,\overrightarrow{y}^n.T, \overrightarrow{\lambda\overrightarrow{z : E}. F}^k) : T[\overrightarrow{G}^n/\overrightarrow{y}^n, c_j[\overrightarrow{P}^p]\,\overrightarrow{M}^{m_j}/i]}\ (0)$$

Avec I un inductif à k constructeurs tel que $(I, r) \in \mathcal{I}$ et :

$$\begin{aligned}\mathtt{Arity}_I(\overrightarrow{P}^p) &= \forall\overrightarrow{y : B}^n.s\\ \mathtt{Constr}_I^j(\overrightarrow{P}^p) &= \forall\overrightarrow{z : E_j}^{m_j}.I[\overrightarrow{P}^p]\,D_j\end{aligned}$$

On doit alors montrer que

$$\Gamma \vdash F_j[\overrightarrow{M}^{m_j}/\overrightarrow{z}^{m_j}] : T[\overrightarrow{G}^n/\overrightarrow{y}^n, c_j[\overrightarrow{P}^p]\,\overrightarrow{M}^{m_j}/i] \qquad (*)$$

D'après le lemme précédent appliqué à (1), on a :
– $(H_1) : \overrightarrow{G}^n \equiv \overrightarrow{D_j}^n[\overrightarrow{M}^{m_j}/\overrightarrow{y}^{m_j}]$,
– $(H_2) : \Gamma \vdash M_l : E_j[M_1/y_1, \ldots, M_{l-1}/y_{l-1}]$ pour $1 \leq l \leq m_j$.

On déduit de $(i+2)$, du lemme de substitution (lemme 2.3.2) et de (H_2) :

$$\Gamma \vdash F_i[\overrightarrow{M}^{m_i}/\overrightarrow{z}^{m_i}] : \left(T[\overrightarrow{D_i}^n/\overrightarrow{y}^n, c_i[\overrightarrow{P}^p]\,\overrightarrow{z}^{m_i}/i]\right)[\overrightarrow{M}^{m_i}/\overrightarrow{z}^{m_i}]$$

or $\overrightarrow{z}^{m_i} \notin \mathcal{FV}(T)$, on a donc

$$\Gamma \vdash F_i[\overrightarrow{M}^{m_i}/\overrightarrow{z}^{m_i}] : T[\overrightarrow{D_i}^n[\overrightarrow{M}^{m_i}/\overrightarrow{z}^{m_i}]/\overrightarrow{y}^n, c_i[\overrightarrow{P}^p]\,\overrightarrow{M}^{m_i}/i] \qquad (*')$$

or on a d'après (H_1),

$$T[\overrightarrow{D_i}^n[\overrightarrow{M}^{m_i}/\overrightarrow{z}^{m_i}]/\overrightarrow{y}^n, c_i[\overrightarrow{P}^p]\,\overrightarrow{M}^{m_i}/i] \equiv T[\overrightarrow{G}^n/\overrightarrow{y}^n, c_j[\overrightarrow{P}^p]\,\overrightarrow{M}^{m_j}/i]$$

et le point 3. du lemme 2.3.2 appliqué à (0), nous donne $\mathrm{WF}_\Gamma(T[\overrightarrow{G}^n/\overrightarrow{y}^n, c_j[\overrightarrow{P}^p]\,\overrightarrow{M}^{m_j}/i])$. On peut donc utiliser la règle Cumulativité, pour déduire $(*)$ à partir de $(*')$.
– Supposons que la dernière règle appliquée dans la dérivation soit une application de la règle Application de la forme ci-dessous

$$\frac{\begin{array}{cc}(1) & (2)\\ \Gamma \vdash (\mathtt{fix}\, f : A, \overrightarrow{x : B}^l, i : C.M)\,\overrightarrow{N}^l : \forall i : P.Q & \Gamma \vdash N_{l+1} : P\end{array}}{\Gamma \vdash (\mathtt{fix}\, f : A, \overrightarrow{x : B}^l, i : C.M)\,\overrightarrow{N}^{l+1} : Q[N_{l+1}/i]}\ (0)$$

On a alors

$$(\texttt{fix}\, f : A, \overrightarrow{x : B}^{l}, y : C.M)\, \overrightarrow{N}^{l+1} \rhd M[\texttt{fix}\, f : A, \overrightarrow{x : B}^{l+1}.M/f, \overrightarrow{N}^{l+1}/\overrightarrow{x}^{l+1}]$$

et on doit montrer que

$$\Gamma \vdash M[\texttt{fix}\, f : A, \overrightarrow{x : B}^{l+1}.M/f, \overrightarrow{N}^{l+1}/\overrightarrow{x}^{l+1}] : Q[N_{l+1}/i]$$

Le cas général étant similaire, nous n'allons traiter que le cas $l = 1$:

$$\frac{\begin{array}{cc}(1) & (2)\\ \Gamma \vdash (\texttt{fix}\, f : A, x : B, i : C.M)\, N_1 : \forall i : P.Q & \Gamma \vdash N_2 : P\end{array}}{\begin{array}{c}\Gamma \vdash (\texttt{fix}\, f : A, x : B, i : C.M)\, N_1\, N_2 : Q[N_2/i]\\ (0)\end{array}}$$

On doit montrer que

$$\Gamma \vdash M[\texttt{fix}\, f : A, x : B, i : C.M/f, N_1/x, N_2/i] : Q[N_2/i] \qquad (*)$$

Par inversion de (1), obtient qu'il existe P' et Q' tels que :
– $(H_1) : \Gamma \vdash \texttt{fix}\, f : A, x : B, i : C.M : \forall x : P'.Q'$,
– $(H_2) : \Gamma \vdash N_1 : P'$,
– $(H_3) : Q'[N_1/x] \preccurlyeq \forall i : P.Q$
Par inversion de (H_1), on en déduit que
– $(H'_1) : \Gamma, f : A \vdash \lambda x : B, i : C.M : A$,
– $(H'_2) : A \preccurlyeq \forall x : P'.Q'$,
– (H'_3) : il existe une sorte t telle que $\Gamma \vdash A : t$.
Par inversion de (H'_1), on en déduit que
– (H''_1) : il existe A' et s, tels que $\Gamma, f : A \vdash \forall x : B.A' : s$,
– $(H''_2) : \Gamma, f : A, x : B \vdash \lambda i : C.M : A'$,
– $(H''_3) : \forall x : B.A' \preccurlyeq A$.
Par inversion de (H''_2), on en déduit que
– (H'''_1) : il existe A'' et r, tels que $\Gamma, f : A, x : B \vdash \forall i : C.A'' : r$,
– $(H'''_2) : \Gamma, f : A, x : B, i : C \vdash M : A''$,
– $(H'''_3) : \forall i : C.A'' \preccurlyeq A'$.
Posons $F = \texttt{fix}\, f : A, x : B, i : C.M$.
De (H'_2) et (H''_3) on déduit que $\forall x : P'.Q' \equiv \forall x : B.A'$ et donc que $P' \equiv B$ et $Q' \equiv A'$. À partir de (H''_2), on montre que $\mathrm{WF}_{\Gamma,f:A}(B)$ et d'après le lemme de substitution, on en déduit que $\mathrm{WF}_\Gamma(B[F/f])$ or $f \notin \mathcal{FV}(B)$ implique que $\mathrm{WF}_\Gamma(B)$. Et donc par conversion, on a $\Gamma \vdash N_1 : B$.
Par transitivité (H'_2) et (H''_3), montre que $\forall x : B.A' \preccurlyeq \forall x : P'.Q'$. On en déduit alors que $B \equiv P'$, $A' \preccurlyeq Q'$ et donc que $A'[F/f, N_1/x] \preccurlyeq Q'[F/f, N_1/x]$.
À partir de (H_1), on déduit que $f \notin Q'$ (car $f \notin \Gamma$ et $x \neq f$) et donc que $Q'[N_1/x] = Q'[F/f, N_1/x]$. On en déduit alors :

$$(\forall i : C.A'')[F/f, N_1/x] \preccurlyeq A'[F/f, N_1/x] \preccurlyeq \forall i : P.Q$$

On a donc $C[N_1/x] \equiv P$, $A''[F/f, N_1/x] \preccurlyeq Q$ et donc $A''[F/f, N_1/x, N_2/i] \preccurlyeq Q[N_2/i]$ (H_4). Or à partir de (H_1'''), on montre que $\mathrm{WF}_{\Gamma,f:A,x:B}(C)$ et donc par substitution on en déduit que $\mathrm{WF}_\Gamma(C[F/f, N_1/x])$ et par conversion de (2) on prouve que $\Gamma \vdash N_2 : C[F/f, N_1/x]$.
Ensuite à l'aide de la règle Point-Fixe appliquée à (H_1'), on montre $\Gamma \vdash F : A$ et comme on a prouvé $\Gamma \vdash N_1 : B$, on a $f \notin \mathcal{FV}(N_1) \cup \mathcal{FV}(B)$ et donc $\Gamma \vdash N_1[F/f] : B[F/f]$. On peut donc appliquer le lemme de substitution à (H_2''') et conclure que

$$\Gamma \vdash M[F/f, N_1/x, N_2/i] : A''[F/f, N_1/x, N_2/i]$$

Enfin on peut déduire de (0) que $\mathrm{WF}_\Gamma(Q[N_2/i])$ et donc (H_4) et la cumulativité nous permettent de prouver (*). ☺

Il est raisonnable de penser que les résultats de la section 1.2 s'étendent naturellement aux CTSI. On a admettra en particulier les résultats suivant :

2.3.8 Lemme

1. *Si* $\mathcal{P} \models$ MINLOC, *alors* $\Gamma \vdash M : A$ *et* $\Gamma \vdash M : B$ *implique qu'il existe* C *tel que* $\Gamma \vdash M : C$ *et* $C \preccurlyeq A$ *et* $C \preccurlyeq B$.
2. *Si* $\mathcal{P} \models$ MINLOC $\wedge$ WEAK-UPSTABLE-RULES, *alors* $\mathcal{P} \models$ STRENGTHENING.

2.4 Le λ-cube avec types inductifs

Dans cette section, nous présentons brièvement quelques exemples de systèmes avec types inductifs.

Système $\mathcal{T}$ Le premier d'entre eux est le système $\mathcal{T}$ de Gödel, c'est une extension du λ-calcul simplement type avec un produit cartésien de type et trois types de bases : les entiers, les booléens et un type à exactement un habitant. Dans sa présentation standard (voir [GLT89] par exemple), les types de données viennent chacun avec leur schéma de récursion. Ici, nous le présenterons comme le CTSI muni des paramètres suivants :

$$\begin{aligned}
\mathcal{S}_\mathcal{T} &= \{\star, \Box\} \\
\mathcal{A}_\mathcal{T} &= \{(\star, \Box)\} \\
\mathcal{R}_\mathcal{T} &= \{(\star, \star, \star)\} \\
\mathcal{C}_\mathcal{T} &= \emptyset \\
\mathcal{I}_\mathcal{T} &= \{\mathsf{nat}, \mathsf{bool}, \mathsf{prod}\} \\
\mathcal{E}_\mathcal{T} &= \{(I, \star) | I \in \mathcal{I}\}
\end{aligned}$$

et il contient un inductif pour représenter les entiers :

$$\mathtt{Ind}(\mathsf{nat} : \star, \mathsf{zero} : \mathsf{nat}, \mathsf{succ} : \mathsf{nat} \to \mathsf{nat})$$

un inductif pour représenter les booléens :

$$\mathtt{Ind}(\mathsf{bool} : \star, \mathsf{true} : \mathsf{bool}, \mathsf{false} : \mathsf{bool})$$

ainsi qu'un encodage du produit cartésien :

$$\texttt{Ind}(\alpha : \star, \beta : \star, \mathsf{prod} : \star, \mathsf{pair} : \alpha \to \beta \to \mathsf{prod}[\alpha, \beta])$$

Il partage donc les sortes, les axiomes et les règles de λ mais contient en plus deux types de base nat et bool ainsi qu'un constructeur de type prod pour représenter les produits cartésiens. On se permettra dans la suite de noter $\sigma \times \tau$ le terme $\mathsf{prod}[\sigma][\tau]$ tout en gardant en tête que le produit $(\times)$ n'est pas un terme du langage . On verra plus loin l'extension système $\mathcal{T}_\omega$, où on peut typer l'abstraction $\lambda\alpha : \star, \beta : \star.\, \mathsf{prod}[\alpha][\beta]$.

On peut définir les deux projections à l'aide de `case` :

$$\begin{array}{lcl} \mathsf{proj}^k_{\tau_1,\tau_2} & : & \tau_1 \times \tau_2 \to \tau_k \\ \mathsf{proj}^k_{\tau_1,\tau_2} & = & \lambda c : \tau_1 \times \tau_2.\, \mathsf{case}_{\mathsf{prod}}(c, \tau_1, \tau_2, \lambda\_ : \tau_1 \times \tau_2.\tau_k, \lambda x_1 : \tau_1, x_2 : \tau_2.x_k) \end{array}$$

Puis, on vérifie aisément qu'elle satisfait les règles de réduction attendues pour un produit :

$$\mathsf{proj}^k_{\tau_1,\tau_2}\,(\mathsf{pair}[\tau_1][\tau_2]\,A_1\,A_2) \rhd A_k$$

Nous pouvons programmer également dans ce système d'une famille de schémas de récurrence pour chaque type τ bien formé :

$$\begin{array}{lcl} \mathsf{rec}_\tau & : & (\tau \to \mathsf{nat} \to \tau) \to \tau \to \mathsf{nat} \to \tau \\ \mathsf{rec}_\tau & = & \texttt{fix}\, f : (\tau \to \mathsf{nat} \to \tau) \to \tau \to \mathsf{nat} \to \tau, \\ & & \qquad g : \tau \to \mathsf{nat} \to \tau, x : \tau, n : \mathsf{nat}\,. \\ & & \quad \mathsf{case}_{\mathsf{nat}}(n, \lambda\_ : \mathsf{nat}\,.\tau, \\ & & \quad x, \\ & & \quad \lambda p : \mathsf{nat}\,.g\,(f\,g\,x\,p)\,p) \end{array}$$

Nous notons $\overline{n}$ la représentation du n-ième entier :

$$\overbrace{\mathsf{succ}\,(\mathsf{succ}\,\cdots(\mathsf{succ}}^{n \text{ fois}}\ \mathsf{zero})\cdots)$$

Et on vérifie facilement que nous disposons bien des réductions suivantes :

$$\begin{array}{rcl} \mathsf{rec}_\tau\ f\ x\ \overline{0} & \rhd & x \\ \mathsf{rec}_\tau\ f\ x\ \overline{n+1} & \rhd & \overbrace{f\,(f\,\cdots\,(f}^{n+1 \text{ fois}}\ x\,\overline{0})\cdots\overline{n-1})\,\overline{n} \end{array}$$

Nous pouvons d'ores et déjà remarquer un avantage des représentations des types de données via les types inductifs par rapport aux encodages imprédicatifs : on dispose d'une implémentation des destructeurs en temps constant. Ainsi, nous pouvons programmer dans Système $\mathcal{T}$ la fonction prédécesseur de manière beaucoup plus naturelle que dans Système $\mathcal{F}$ (voir page 26) :

$$\begin{array}{lcl} \mathsf{pred} & : & \mathsf{nat} \to \mathsf{nat} \\ \mathsf{pred} & = & \lambda : \mathsf{nat}\,.\,\mathsf{case}_{\mathsf{nat}}(n, \lambda\_ : \mathsf{nat}\,.\,\mathsf{nat}, \overline{0}, \lambda p : \mathsf{nat}\,.p) \end{array}$$

Nous pouvons alors vérifier que deux étapes de réduction, une pour le λ, une pour le `case`, suffisent pour réduire ces fonctions appliquées à la représentation d'un entier :

$$\begin{aligned} \mathsf{pred}\ \overline{n+1} &\triangleright \overline{n} \\ \mathsf{pred}\ \overline{0} &\triangleright \overline{0} \end{aligned}$$

Nous avons déjà mentionné lors l'étude de Système $\mathcal{F}$ que la présence d'un schéma de récursion d'ordre supérieur permet d'implémenter une très grande classe de fonction (contenant les fonctions les fonctions primitives récursives, mais aussi la fonction d'Ackermann). Gödel a donné une caractérisation de cette classe en terme logique en prouvant que les fonctions représentables dans Système $\mathcal{T}$ sont les fonctions prouvablement totales dans l'arithmétique du premier ordre **PA**. Nous reviendrons sur ce résultat dans le chapitre 4.

Système $\mathcal{T}_\omega$. Nous pouvons étendre Système $\mathcal{T}$ en autorisant l'ordre supérieur dans les types. Nous définissons donc le système $\mathcal{T}_\omega$, par les paramètres suivants :

$$\begin{aligned} \mathcal{S}_\mathcal{T} &= \{\star, \Box\} \\ \mathcal{A}_\mathcal{T} &= \{(\star, \Box)\} \\ \mathcal{R}_\mathcal{T} &= \{(\star,\star,\star), (\Box,\Box,\Box)\} \\ \mathcal{C}_\mathcal{T} &= \emptyset \\ \mathcal{I}_\mathcal{T} &= \{\mathsf{nat}, \mathsf{bool}, \mathsf{prod}\} \\ \mathcal{E}_\mathcal{T} &= \{(I,\star) | I \in \mathcal{I}\} \end{aligned}$$

Nous obtenons donc un système qui est à $\mathcal{T}$ ce que $\lambda\omega$ est à λ. Et tout comme ce dernier, cette extension est essentiellement cosmétique, elle permet simplement d'internaliser les notations de type. Ainsi dans ce système, on peut définir des constructeurs de type comme par exemple :

$$\lambda\alpha : \star.(\alpha \to \mathsf{nat}) \times (\mathsf{nat} \to \alpha)$$

mais ces notations dans les types peuvent toujours être réduites. On montre aisément que la construction `case` ne peut pas apparaître dans un terme A si on a $\Gamma \vdash A : K : \Box$ (c'est une conséquence immédiate du fait que $(I, \Box) \notin \mathcal{E}$ pour $I = \mathsf{nat}, \mathsf{prod}, \mathsf{bool}$). Il suffit alors d'étendre la fonction d'élimination de l'ordre supérieur vue à la page 28 de la façon suivante :

$$\begin{aligned} \mathcal{E}^2(\mathsf{prod}[\sigma,\tau]) &= \mathsf{prod}[\mathcal{E}^2(\mathrm{NF}(\sigma)), \mathcal{E}^2(\mathrm{NF}(\tau))] \\ \mathcal{E}^2(\mathsf{nat}) &= \mathsf{nat} \\ \mathcal{E}^2(\mathsf{bool}) &= \mathsf{bool} \\ \mathcal{E}^1(\mathsf{zero}) &= \mathsf{zero} \\ \mathcal{E}^1(\mathsf{succ}) &= \mathsf{succ} \\ \mathcal{E}^1(\mathsf{pair}[\sigma,\tau]) &= \mathsf{pair}[\mathcal{E}^2(\mathrm{NF}(\sigma)), \mathcal{E}^2(\mathrm{NF}(\tau))] \end{aligned}$$

$$\begin{aligned}
&\mathcal{E}^1(\mathsf{case}_{\mathsf{nat}}(n, \lambda_ : \mathsf{nat}.T, F_1, \lambda p : \mathsf{nat}.F_2)) = \\
&\quad \mathsf{case}_{\mathsf{nat}}(\mathcal{E}^1(n), \lambda_ : \mathsf{nat}.\mathcal{E}^2(\mathrm{NF}(T)), \mathcal{E}^1(F_1), \lambda p : \mathsf{nat}.\mathcal{E}^1(F_2)) \\
&\mathcal{E}^1(\mathsf{case}_{\mathsf{bool}}(b, \lambda_ : \mathsf{bool}.T, F_1, F_2)) = \\
&\quad \mathsf{case}_{\mathsf{bool}}(\mathcal{E}^1(b), \lambda_ : \mathsf{bool}.\mathcal{E}^2(\mathrm{NF}(T)), \mathcal{E}^1(F_1), \mathcal{E}^1(F_2)) \\
&\mathcal{E}^1(\mathsf{case}_{\mathsf{prod}}(c, \sigma, \tau, \lambda_ : \mathsf{prod}[\sigma, \tau].T, \lambda x : \sigma, y : \tau.F)) = \\
&\quad \mathsf{case}_{\mathsf{prod}}(\mathcal{E}^1(c), \mathcal{E}^2(\mathrm{NF}(\sigma)), \mathcal{E}^2(\mathrm{NF}(\tau)), \\
&\qquad \lambda_ : \mathsf{prod}[\sigma, \tau].\mathcal{E}^2(\mathrm{NF}(T)), \lambda x : \mathcal{E}^2(\mathrm{NF}(\sigma)), y : \mathcal{E}^2(\mathrm{NF}(\tau)).\mathcal{E}^1(F))
\end{aligned}$$

On montre alors de la même façon que :

2.4.1 Lemme

- *Si* $\Gamma \vdash_{\mathcal{T}_\omega} A : K : \Box$, *alors* $\mathcal{E}(\Gamma) \vdash_{\mathcal{T}} \mathcal{E}^2(\mathrm{NF}(A)) : \star$.
- *Si* $\Gamma \vdash_{\mathcal{T}_\omega} M : A : \star$, *alors* $\mathcal{E}(\Gamma) \vdash_{\mathcal{T}} \mathcal{E}^1(M) : \mathcal{E}^2(\mathrm{NF}(A))$.

Pour en déduire le lemme suivant :

2.4.2 Lemme

Les fonctions représentables de $\mathcal{T}_\omega$ *sont exactement les fonctions représentables de* $\mathcal{T}$.

2.4.1 Extension aux inductifs avec élimination faible

On introduit maintenant la notion de CTSI *saturé*. Un CTSI est saturé s'il contient toutes les déclarations d'inductifs autorisées :

2.4.3 Définition (CTSI saturé)

On dira d'un CTSI qu'il est saturé si pour toute famille de termes $Q_1, \ldots, Q_n$, A, $C'_1, \ldots, C'_n$ qui vérifie les conditions de la définition 2.1.3, c'est-à-dire :

1. $\mathrm{WF}_{\Gamma_P}(A)$ où $\Gamma_P = x_1 : Q_1, \ldots, x_p : Q_p$,
2. A est de la forme $\forall \overrightarrow{y : B}^n.s$
3. $\Gamma_P, \alpha : A \vdash C'_j : s$
4. pour tout j, C_j est de la la forme
$$\forall \overrightarrow{y : E_j}^{m_j}.\alpha\, \overrightarrow{D_j}^n$$
où α ne peut apparaître à l'intérieur de E_j que comme une conclusion.

il existe un inductif I bien formé :

$$\mathtt{Ind}(x_1 : Q_1, \ldots, x_p : Q_p, I : A, c_1 : C_1, \ldots, c_k : C_k)$$

où les $C_i = C'[I[\overrightarrow{x}^p]/\alpha]$.

2.4.4 Définition (Extensions aux types inductifs avec élimination faible)

Étant donné un système $\mathcal{P}$ du λ-cube, on dira qu'un CTSI saturé $\mathcal{P}'$ est une extension de $\mathcal{P}$ aux types inductifs avec élimination faible, si :

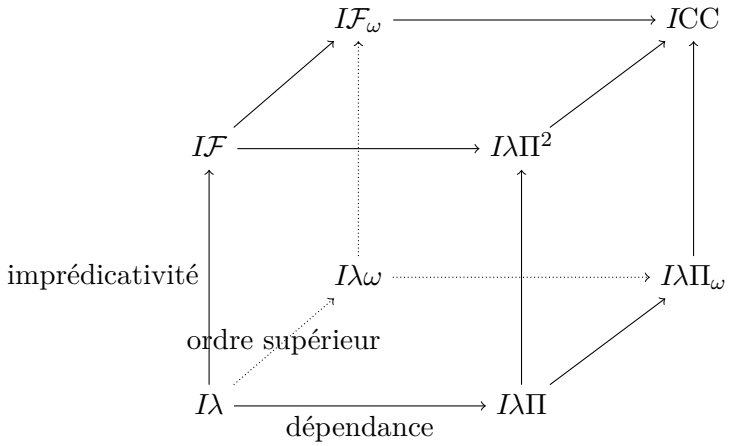
$$\begin{aligned}
\mathcal{S}_{\mathcal{P}'} &= \mathcal{S}_{\mathcal{P}} \\
\mathcal{A}_{\mathcal{P}'} &= \mathcal{A}_{\mathcal{P}} \\
\mathcal{R}_{\mathcal{P}'} &= \mathcal{R}_{\mathcal{P}} \\
\mathcal{C}_{\mathcal{P}'} &= \emptyset
\end{aligned}$$

et si de plus : $(I, r) \in \mathcal{E}$ si et seulement si I est un inductif de sorte $\star$ et $r = \star$.

2.4.5 Convention

Pour tout système $\mathcal{P}$ du λ-cube, on désignera par $I\mathcal{P}$ une extension quelconque de $\mathcal{P}$ aux types inductifs.

Nous pouvons maintenant étendre le λ-cube avec des inductifs :



Le λ-calcul simplement avec type inductif : $I\lambda$. C'est l'extension naturelle de système $\mathcal{T}$ dans laquelle nous disposons de beaucoup plus d'inductifs que simplement nat, bool et prod. On peut donc y encoder tous les types inductifs qui vérifient les conditions de stricte positivité et dont les constructeurs sont bien typables. Dans ce système, la seule arité bien formée de sorte minimale est la sorte $\star$. On ne peut donc pas construire d'inductif avec un ou plus arguments. On peut tout de même construire le produit cartésien, la somme disjointe, les listes, les arbres, les booléens, les entiers, ... Tous ces inductifs restent encodables par des opérations sur les entiers. À première vue, il pourrait sembler que nous ne gagnons pas en expressivité par rapport au système $\mathcal{T}$. Ce n'est pas le cas. On dispose en effet de type inductifs qui ne s'encodent pas dans les entiers, comme par exemple, le type des arbres à branchement infini dénombrable (parfois appelé le type des ordinaux de Brouwer) :

$$\begin{array}{ll} \texttt{Ind}\,(\,\mathsf{ord\_brouwer} : \texttt{Set}, & \mathsf{zero\_brouwer} : \mathsf{ord\_brouwer}, \\ & \mathsf{succ\_brouwer} : \mathsf{ord\_brouwer} \to \mathsf{ord\_brouwer}, \\ & \mathsf{sup\_brouwer} : (\mathsf{nat} \to \mathsf{ord\_brouwer}) \to \mathsf{ord\_brouwer}\,) \end{array}$$

Ce type inductif permet de décrire des ordinaux dénombrables dont certains plus grands que ϵ_ω. Bien que nous n'en ayons pas la preuve, il semble raisonnable de penser que le système $I\lambda$ permet de représenter strictement plus de fonctions que le système $\mathcal{T}$. Nous en rediscuterons après avoir décrit le théorème de représentation en présence d'inductif (page 257).

On notera qu'il est possible de définir dans $I\lambda$ des inductifs avec des argument qui ne sont pas des types. Comme par exemple l'inductif U défini par

$$\texttt{Ind}(n : \mathsf{nat}, \mathtt{U} : \star, c_{\mathtt{H}} : \mathtt{U}[n])$$

qui désigne une famille d'inductifs tous équivalent à l'inductif unit :

$$\texttt{Ind}(\texttt{unit} : \star, \texttt{tt} : \texttt{unit})$$

On peut donc alors construire le type suivant :

$$\vdash \forall n : \texttt{nat}.\texttt{U}[n] \to \texttt{U}[n] : \star$$

Notre λ-calcul simplement typé étendu avec des inductifs est devenu un système de types... dépendants. Mais, si cela est souhaitable, il est possible de restreindre ce système en interdisant les inductifs contenant des arguments dont le type est dans $\star$. Il est clair que cela ne permet de toute façon pas de représenter plus de fonctions.

En effet, on dira d'un paramètre qu'il est *informatif* si son type est typé par $\star$. Alors, l'absence de règle pour forme des types dépendants fait que ces paramètres ne pourront pas apparaître dans le type des constructeurs si ceux-ci ne contiennent pas d'occurrences de ces inductifs avec au moins un paramètre informatif (et encore moins dans l'arité puisqu'elle est toujours égale à $\star$). Soit donc H un inductif avec des paramètres informatifs qui ne contient pas d'occurrence d'un autre inductif ayant des paramètres informatifs dans sa déclaration (c'est-à-dire dans le type de ses paramètres, constructeurs et arité). Dans ce cas, H est équivalent au type inductif H' obtenu en enlevant ces paramètres informatifs de sa déclaration (c'est l'opération qui fait passer de U à unit). On peut alors remplacer dans une dérivation de typage toutes les utilisations de H par une utilisation de H' tout en préservant la typabilité (il faut bien sûr remplacer également les utilisations de ses constructeurs et ses case). En procédant récursivement, on peut alors remplacer tous les inductifs avec des paramètres informatifs par des inductifs qui n'en contiennent pas.

Ajout de l'ordre supérieur de type : $I\lambda\omega$. Il est clair que tout comme pour $\lambda\omega$, et $\mathcal{T}\omega$, l'ajout de l'ordre supérieur de type ne permet pas de représenter plus de fonctions. On prouve le lemme ci-dessous de façon analogue à $\mathcal{T}_\omega$, en étendant la fonction d'effacement de la manière suivante :

$$\begin{aligned}\mathcal{E}^2(I[\overrightarrow{P}^p]) &= I[\mathcal{E}^{1/2}(\overrightarrow{P}^p)]\\ \mathcal{E}^2(c[\overrightarrow{P}^p]) &= c[\mathcal{E}^{1/2}(\overrightarrow{P}^p)]\end{aligned}$$

$$\mathcal{E}^1(\,\texttt{case}_I(M, \overrightarrow{P}^p, \lambda x : I.T, \overrightarrow{y : E}^{\overrightarrow{m}}.F^{\overrightarrow{k}}\,)) =$$
$$\texttt{case}_{\mathcal{E}(I)}(\mathcal{E}^1(M), \mathcal{E}^{1/2}(\overrightarrow{P}^p), \lambda x : \mathcal{E}(I).\mathcal{E}^2(\mathrm{NF}(T)), \overrightarrow{y : \mathcal{E}^2(\mathrm{NF}(E))}^{\overrightarrow{m}}.\mathcal{E}^1(F)^{\overrightarrow{k}}\,)$$

où :

- $\mathcal{E}^{1/2}(\overrightarrow{P}^p) = P'_1, \ldots, P'_n$ avec $P'_k = \mathcal{E}^1(P)$ ou $P'_k = \mathcal{E}^2(\mathrm{NF}(P))$ selon que le k-ième argument de l'inductif correspondant est informatif ou non.
- $\mathcal{E}(I)$ désigne l'inductif :

$$\texttt{Ind}(x_1 : Q'_1, \ldots, x_p : Q'_p, I : \star, c_1 : \mathcal{E}^2 C_1, \ldots, c_k : \mathcal{E}^2 C_k)$$

si

$$\texttt{Ind}(x_1 : Q_1, \ldots, x_p : Q_p, I : \star, c_1 : C_1, \ldots, c_k : C_k)$$

et avec $Q'_k = \mathcal{E}^2(Q_1)$ ou $Q'_k = \star$ selon que le k-ième argument est informatif ou non.

On peut alors montrer aisément que :

2.4.6 Lemme

- *Si* $\Gamma \vdash_{I\lambda\omega} A : K : \Box$, *alors* $\mathcal{E}(\Gamma) \vdash_{I\lambda} \mathcal{E}^2(\mathrm{NF}(A)) : \star$.
- *Si* $\Gamma \vdash_{I\lambda\omega} M : A : \star$, *alors* $\mathcal{E}(\Gamma) \vdash_{I\lambda} \mathcal{E}^1(M) : \mathcal{E}^2(\mathrm{NF}(A))$.

Et en déduire que :

2.4.7 Lemme

Les fonctions représentables de $I\lambda\omega$ sont exactement les fonctions représentables de $I\lambda$.

Ajout des types dépendants En l'absence d'élimination forte, Christine Paulin [PM89b, PM89a] a montré que la fonction d'effacement des dépendance $\mathcal{E}^{\mathrm{dep}}$ s'étend bien aux inductifs.

En effet, soit $\mathcal{P} \in \{\lambda\Pi, \lambda\Pi_\omega, \lambda\Pi_\omega, \lambda\Pi^2\}$, nous définissons $\mathcal{E}^{\mathrm{dep}}(I\mathcal{P})$ par :

$$\begin{aligned}
\mathcal{E}^{\mathrm{dep}}(I\lambda\Pi) &= I\lambda \\
\mathcal{E}^{\mathrm{dep}}(I\lambda\Pi_\omega) &= I\lambda\omega \\
\mathcal{E}^{\mathrm{dep}}(I\lambda\Pi^2) &= I\mathcal{F} \\
\mathcal{E}^{\mathrm{dep}}(I\mathrm{CC}) &= I\mathcal{F}_\omega
\end{aligned}$$

Alors nous pouvons étendre la fonction $\mathcal{E}^{\mathrm{dep}}$ vue page 34 aux inductifs en posant :

$$\begin{aligned}
\mathcal{E}^{\mathrm{dep}}_\Gamma(I[\overrightarrow{P}^p]) &= \mathcal{E}^{\mathrm{dep}}(I)[\overrightarrow{\mathcal{E}^{\mathrm{dep}}_\Gamma(P)}^p] \\
\mathcal{E}^{\mathrm{dep}}_\Gamma(c[\overrightarrow{P}^p]) &= \mathcal{E}^{\mathrm{dep}}(c)[\overrightarrow{\mathcal{E}^{\mathrm{dep}}_\Gamma(P)}^p] \\
\mathcal{E}^{\mathrm{dep}}_\Gamma(\texttt{case}_I(M, \overrightarrow{P}^p, T, \overrightarrow{F}^k)) &= \texttt{case}_{\mathcal{E}^{\mathrm{dep}}(I)}(\mathcal{E}^{\mathrm{dep}}_\Gamma(M), \overrightarrow{\mathcal{E}^{\mathrm{dep}}_\Gamma(P)}^p, \mathcal{E}^{\mathrm{dep}}_\Gamma(T), \overrightarrow{\mathcal{E}^{\mathrm{dep}}_\Gamma(F)}^k)
\end{aligned}$$

Si $\texttt{Ind}(\overrightarrow{x : Q}^p, I : A, \overrightarrow{c : C}^k)$ est un inductif de $I\mathcal{P}$, alors $\mathcal{E}^{\mathrm{dep}}(I)$, $\mathcal{E}^{\mathrm{dep}}(c_j)$ désignent un inductif de $\mathcal{E}^{\mathrm{dep}}(I\mathcal{P})$ tel que

$$\texttt{Ind}(x_1 : \mathcal{E}^{\mathrm{dep}}_{\langle\rangle}(Q_1), \ldots, x_p : \mathcal{E}^{\mathrm{dep}}_{x_1:Q_1,\ldots,x_{p-1}:Q_{p-1}}(Q_p), \mathcal{E}^{\mathrm{dep}}_{x_1:Q_1,\ldots,x_p:Q_p}(A), \overrightarrow{\mathcal{E}^{\mathrm{dep}}(c) : \mathcal{E}^{\mathrm{dep}}_{x_1:Q_1,\ldots,x_p:Q_p}(C)}^k)$$

(son existence est assurée par la saturation).

Nous montrons que, tout comme dans le cas sans inductif, la face droite du cube est envoyée vers la face gauche :

2.4.8 Lemme

Si $\mathcal{P} \in \{\lambda\Pi, \lambda\Pi_\omega, \lambda\Pi_\omega, \lambda\Pi^2\}$, *on a :*

$$\Gamma \vdash_{I\mathcal{P}} A : B \text{ implique } \mathcal{E}^{dep}(\Gamma) \vdash_{\mathcal{E}^{dep}(I\mathcal{P})} \mathcal{E}^{dep}_\Gamma(A) : \mathcal{E}^{dep}_\Gamma(B).$$

De plus nous avons,

$$A \rhd B \quad \Rightarrow \quad \mathcal{E}^{\text{dep}}_{\Gamma}(A) \trianglerighteq \mathcal{E}^{\text{dep}}_{\Gamma}(B)$$

et $\mathcal{E}^{\text{dep}}(\mathsf{nat}) = \mathsf{nat}$ (modulo renommage), on en déduit donc que :

2.4.9 Lemme
Si $\mathcal{P} \in \{\lambda\Pi, \lambda\Pi_\omega, \lambda\Pi_\omega, \lambda\Pi^2\}$, alors les fonctions représentables dans $\mathcal{P}$ sont exactement les fonctions représentables de $\mathcal{E}^{dep}(I\mathcal{P})$.

Les encodages imprédicatifs des types inductifs. Les encodages imprédicatifs permettent d'encoder tous les types inductifs en l'absence d'élimination forte. Ce résultat a d'abord été démontré pour système $\mathcal{F}$ par Corrado Böhm et Alessandro Berarducci [BB85] puis étendu à système $\mathcal{F}_\omega$ par Christine Paulin et Frank Pfenning [PPM89].

Soit I un inductif dont la déclaration est : $\mathtt{Ind}(\overrightarrow{x : Q}^p, I : A, \overrightarrow{c : C}^k)$ Soit α une variable fraîche et $(C'_i)_{i=1\ldots k}$ la famille de termes telle que pour tout $1 \leq i \leq k$ on ait

$$C'_i[I[\overrightarrow{x}^p]/\alpha] = C_i \text{ et } I \text{ n'apparaît pas dans } C'_i$$

Alors on note $\underline{I}[\overrightarrow{x : Q}^p]$ *l'encodage imprédicatif* de I défini par le terme :

$$\underline{I}[\overrightarrow{x : Q}^p] = \forall\alpha : A.C'_1 \to \ldots \to C'_k \to \forall\overrightarrow{y : B}^n.\alpha\ \overrightarrow{y}^n$$

on retrouve bien des types isomorphes à ceux vus lors de l'étude de système $\mathcal{F}$:

$$\begin{aligned} \underline{\mathsf{nat}} &= \forall\alpha : \star.\alpha \to (\alpha \to \alpha) \to \alpha \\ \underline{\mathsf{prod}}[\alpha : \star, \beta : \star] &= \forall\gamma : \star.(\alpha \to \beta \to \gamma) \to \gamma \\ \underline{\mathsf{list}}[\alpha : \star] &= \forall\beta : \star.\beta \to (\alpha \to \beta \to \beta) \to \beta \end{aligned}$$

Où nat, prod et list_β sont les inductifs bien-formés de $I\mathcal{F}$ suivant :

$$\begin{array}{c} \mathtt{Ind}(\alpha : \star, \mathsf{list} : \star, \mathsf{nil} : \mathsf{list}[\alpha], \mathsf{cons} : \alpha \to \mathsf{list}[\alpha] \to \mathsf{list}[\alpha]) \\ \mathtt{Ind}(\mathsf{nat} : \star, \mathsf{zero} : \mathsf{nat}, \mathsf{succ} : \mathsf{nat} \to \mathsf{nat}) \qquad\qquad c \\ \mathtt{Ind}(\alpha : \star, \beta : \star, \mathsf{prod} : \star, \mathsf{pair} : \alpha \to \beta \to \mathsf{prod}) \end{array}$$

L'encodage fonctionne également pour système $\mathcal{F}_\omega$. Si on prend la définition inductive prod' du produit cartésien du produit sans paramètre suivante :

$$\mathtt{Ind}(\mathsf{prod}' : \star \to \star \to \star, \mathsf{pair}' : \forall\alpha\,\beta : \star, \alpha \to \beta \to \mathsf{prod}'\ \alpha\,\beta)$$

Elle est une définition inductive bien formée de $I\mathcal{F}_\omega$ et est interdite dans $I\mathcal{F}$ car son arité n'est pas un type valide de $\mathcal{F}$. Son encodage imprédicatif est alors :

$$\underline{\mathsf{prod}'} = \forall\gamma : \star \to \star \to \star, (\forall\alpha\,\beta : \star, \alpha \to \beta \to \gamma\,\alpha\,\beta) \to \forall\alpha\,\beta : \star.\gamma\,\alpha\,\beta$$

On peut alors montrer le lemme suivant :

2.4.10 Lemme
Soit $\mathcal{P} \in \{\mathcal{F}, \mathcal{F}_\omega\}$. Si $\texttt{Ind}(\overrightarrow{x : Q}^p, I : A, \overrightarrow{c : C}^k)$ *est un inductif bien formé de $I\mathcal{P}$, alors*

$$\overrightarrow{x : Q}^p \vdash_{I\mathcal{P}} \underline{I}[\overrightarrow{x : Q}^p] : A$$

De plus s'il n'est fait mention d'aucun autre inductif dans la déclaration de I, alors

$$\overrightarrow{x : Q}^p \vdash_{\mathcal{P}} \underline{I}[\overrightarrow{x : Q}^p] : A$$

Les auteurs de [PE88] donnent également un encodage $\underline{c}[\overrightarrow{x : Q}^p]$ des constructeurs puis décrivent comment construire des termes pour implémenter les schémas de récursion primitive (voir aussi la thèse de Christine Paulin [PM89b] pour plus détails). Si on admet que notre présentation des inductifs avec `case` et `fix` est équivalente à une présentation des inductifs avec récursion primitive, alors on peut remplacer les occurrences des inductifs et de leurs constructeurs par leurs encodages imprédicatifs et utiliser les schémas de récursion primitive pour remplacer les utilisation de `case` et `fix`. On en déduit alors le théorème :

2.4.11 Lemme
Soit $\mathcal{P} \in \{\mathcal{F}, \mathcal{F}_\omega\}$.
Les fonctions représentables de $I\mathcal{P}$ sont exactement les fonctions représentables de $\mathcal{P}$.

Pour conclure, on remarque que le λ-cube étendu aux types inductifs n'a rajouté qu'une seule nouvelle classe de fonctions représentables, les fonctions représentables dans Système $I\lambda$.

2.4.2 Élimination forte non restreinte dans une sorte imprédicative

Dans cette sous-section, nous allons étudier le danger qu'il peut il y avoir lorsque que l'on ne restreint pas l'élimination forte en présence d'une sorte imprédicative.

2.4.12 Définition (Extension aux types inductifs avec élimination forte non-restreinte)
Soit $\mathcal{P}$ un système du cube, on dira qu'un CTSI saturé $\mathcal{P}'$ est une extension de $\mathcal{P}$ aux types inductifs avec élimination forte non-restreinte, si :

$$\begin{aligned} \mathcal{S}_{\mathcal{P}'} &= \mathcal{S}_{\mathcal{P}} \\ \mathcal{A}_{\mathcal{P}'} &= \mathcal{A}_{\mathcal{P}} \\ \mathcal{R}_{\mathcal{P}'} &= \mathcal{R}_{\mathcal{P}} \\ \mathcal{C}_{\mathcal{P}'} &= \mathcal{C}_{\mathcal{P}} \end{aligned}$$

et si de plus : $(I, r) \in \mathcal{E}$ si et seulement si I est un inductif de sorte $\star$ et r est quelconque.

2.4.13 Convention
Pour tout système $\mathcal{P}$ du cube, on désignera par $I_{se}\mathcal{P}$ une extension quelconque de $\mathcal{P}$ aux types inductifs avec élimination forte.

Le type inductif suivant est bien-formé dans $I_{se}\mathcal{F}$, c'est un inductif sans paramètre et avec un argument (on a besoin de la règle $(\Box, \star, \star)$ pour vérifier que le type du constructeur habite bien $\star$).

$$\texttt{Ind}(\mathbf{star} : \star, \mathbf{close} : \star \to \mathbf{star})$$

Le constructeur de cet inductif permet de plonger tous les types (c'est-à-dire les habitants de $\star$) vers le type **star**. L'élimination forte, nous permet alors de typer la fonction réciproque suivante :

$$\begin{array}{rcl} \mathbf{open} & : & \mathbf{star} \to \star \\ \mathbf{open} & = & \lambda b : \mathbf{star}.\, \mathtt{case}_{\mathbf{star}}(b, \lambda\_ : \mathbf{star}.\star, \lambda\alpha : \star.\alpha) \\ \mathbf{open}(\mathbf{close}\,\alpha) & \trianglerighteq & \alpha \end{array}$$

C'est une élimination forte car le type de retour du case est $\star$, c'est-à-dire une type de sorte $\Box$ et par définition, nous avons bien $(\mathbf{star}, \Box) \in \mathcal{E}$. Nous allons maintenant voir que cet inductif permet de plonger le système incohérent $\lambda\star$ dans tous les systèmes qui contiennent $I_{\mathrm{se}}\mathcal{F}$.

On définit la transformation des termes et des contextes de $\lambda\star$ suivante :

$$\begin{array}{rcl} [\![\langle\rangle]\!] & = & \langle\rangle \\ [\![\Gamma, x : A]\!] & = & [\![\Gamma]\!], x : \mathbf{open}\,[\![A]\!] \end{array}$$

$$\begin{array}{rcl} [\![x]\!] & = & x \\ [\![\star]\!] & = & \mathbf{close}\,\mathbf{star} \\ [\![\forall x : A.B]\!] & = & \mathbf{close}\,(\forall x : \mathbf{open}\,[\![A]\!].\,\mathbf{open}\,[\![B]\!]) \\ [\![\lambda x : A.M]\!] & = & \lambda x : \mathbf{open}\,[\![A]\!].[\![M]\!] \\ [\![M\,N]\!] & = & [\![M]\!]\,[\![N]\!] \end{array}$$

Cette transformation se comporte bien vis-à-vis de la substitution et donc de la conversion :

2.4.14 Lemme
- $[\![B[N/x]]\!] = [\![B]\!][[\![N]\!]/x]$,
- $B \equiv B' \quad \Rightarrow \quad [\![B]\!] \equiv [\![B']\!]$.

Démonstration
- Immédiat par induction sur la structure de B.
- On montre que
$$[\![(\lambda x : A.M)\,N]\!] \equiv [\![M[N/x]]\!]$$
En effet, d'après le cas précédent, on a
$$\begin{array}{rcl} [\![(\lambda x : A.M)\,N]\!] & = & (\lambda x : \mathbf{open}\,[\![A]\!].[\![M]\!])\,[\![N]\!] \\ & \twoheadrightarrow & [\![M]\!][[\![N]\!]/x] \\ & = & [\![M[N/x]]\!] \end{array}$$
☺

Cette transformation permet de plonger les termes typables de $\lambda\star$ dans $I_{\mathrm{se}}\mathcal{F}$:

2.4.15 Lemme
Si $\Gamma \vdash_{\lambda\star} M : A$, *alors* $[\![\Gamma]\!] \vdash_{I_{\mathrm{se}}\mathcal{F}} [\![M]\!] : \mathbf{open}\,[\![A]\!]$.

Démonstration On procède par induction sur la dérivation de $\Gamma \vdash_{\lambda\star} M : A$ et on désignera par **HI** les hypothèses d'induction. Dans cette démonstration les séquents $\vdash$ désigneront la typabilité $\vdash_{I_{\mathrm{se}}\mathcal{F}}$.

– Axiome $\star : \star$:

$$\dfrac{\dfrac{\vdash \mathbf{close} : \star \to \mathbf{star} \qquad \vdash \mathbf{star} : \star}{\vdash [\![\star]\!] : \mathbf{star}}}{\vdash [\![\star]\!] : \mathbf{open}\,[\![\star]\!]}$$

On doit montrer que $\vdash \mathbf{close\ star} : \mathbf{open}\,(\mathbf{close\ star})$, ce qui peut se faire à l'aide de la règle de conversion, puisque l'on a $\mathbf{open}\,(\mathbf{close\ star}) \trianglerighteq \mathbf{star}$, $\vdash \mathbf{open}\,(\mathbf{close\ star}) : \star$ et $\vdash \mathbf{close\ star} : \mathbf{star}$.

– Affaiblissement :

$$\dfrac{\dfrac{\mathbf{HI}}{[\![\Gamma]\!] \vdash [\![M]\!] : \mathbf{open}\,[\![B]\!]} \qquad \dfrac{[\![\Gamma]\!] \vdash \mathbf{open} : \mathbf{star} \to \star \qquad \dfrac{\dfrac{\mathbf{HI}}{[\![\Gamma]\!] \vdash [\![A]\!] : \mathbf{open}\,[\![\star]\!]}}{[\![\Gamma]\!] \vdash [\![A]\!] : \mathbf{star}}}{[\![\Gamma]\!] \vdash \mathbf{open}\,[\![A]\!] : \star}}{[\![\Gamma]\!], x : \mathbf{open}\,[\![A]\!] \vdash [\![M]\!] : \mathbf{open}\,[\![B]\!]}$$

– Variable :

$$\dfrac{\dfrac{[\![\Gamma]\!] \vdash \mathbf{open} : \mathbf{star} \to \star \qquad \dfrac{\mathbf{HI}}{[\![\Gamma]\!] \vdash [\![A]\!] : \mathbf{star}}}{[\![\Gamma]\!] \vdash \mathbf{open}\,[\![A]\!] : \star}}{[\![\Gamma]\!], x : \mathbf{open}\,[\![A]\!] \vdash x : \mathbf{open}\,[\![A]\!]}$$

– Application :

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{HI}}{[\![\Gamma]\!] \vdash [\![M]\!] : \mathbf{open}\,[\![\forall x : A.B]\!]}}{[\![\Gamma]\!] \vdash [\![M]\!] : \forall x : \mathbf{open}\,[\![A]\!].\,\mathbf{open}\,[\![B]\!]} \qquad \dfrac{\mathbf{HI}}{[\![\Gamma]\!] \vdash [\![N]\!] : \mathbf{open}\,[\![A]\!]}}{[\![\Gamma]\!] \vdash [\![M]\!]\,[\![N]\!] : (\mathbf{open}\,[\![B]\!])[[\![N]\!]/x]}}{[\![\Gamma]\!] \vdash [\![M]\!]\,[\![N]\!] : \mathbf{open}\,[\![B[N/x]]\!]}\ \text{Lemme 2.4.14}$$

– Abstraction :

$$\dfrac{\dfrac{\mathbf{HI}}{[\![\Gamma]\!], x : \mathbf{open}\,[\![A]\!] \vdash [\![M]\!] : \mathbf{open}\,[\![B]\!]} \qquad \dfrac{H}{[\![\Gamma]\!] \vdash \forall x : \mathbf{open}\,[\![A]\!].\,\mathbf{open}\,[\![B]\!] : \star}}{[\![\Gamma]\!] \vdash [\![\lambda x : A.M]\!] : \mathbf{open}\,[\![\forall x : A.B]\!]}$$

On peut dériver (H), par exemple, en dérivant le séquent :

$$\dfrac{[\![\Gamma]\!] \vdash \mathbf{open} : \mathbf{star} \to \star \qquad \dfrac{\dfrac{\mathbf{HI}}{[\![\Gamma]\!] \vdash \mathbf{close}\,(\forall x : \mathbf{open}\,[\![A]\!].\,\mathbf{open}\,[\![B]\!]) : \mathbf{open}\,(\mathbf{close\ star})}}{[\![\Gamma]\!] \vdash \mathbf{close}\,(\forall x : \mathbf{open}\,[\![A]\!].\,\mathbf{open}\,[\![B]\!]) : \mathbf{star}}}{[\![\Gamma]\!] \vdash \mathbf{open}\,(\mathbf{close}\,(\forall x : \mathbf{open}\,[\![A]\!].\,\mathbf{open}\,[\![B]\!])) : \mathbf{star}}$$

puis en invoquant la préservation du typage : $\mathbf{open}\,(\mathbf{close}\,(\forall x : \mathbf{open}\,[\![A]\!].\,\mathbf{open}\,[\![B]\!])) \rhd \forall x : \mathbf{open}\,[\![A]\!].\,\mathbf{open}\,[\![B]\!]$.

– Produit : À l'aide des hypothèse d'induction on prouve les deux séquents ci-dessous.

$$\frac{\dfrac{\textbf{HI}}{[\![\Gamma]\!], x : [\![A]\!] \vdash [\![B]\!] : \mathbf{open}\,[\![\star]\!]}}{[\![\Gamma]\!], x : [\![A]\!] \vdash [\![B]\!] : \mathbf{star}} \qquad \frac{\dfrac{\textbf{HI}}{[\![\Gamma]\!] \vdash [\![A]\!] : \mathbf{open}\,[\![\star]\!]}}{[\![\Gamma]\!] \vdash [\![A]\!] : \mathbf{star}}$$

On en déduit facilement $[\![\Gamma]\!] \vdash \mathbf{close}\,(\forall x : \mathbf{open}\,[\![A]\!].\,\mathbf{open}\,[\![B]\!]) : \mathbf{star}$ en appliquant **close** et **open**.

– Conversion : Par hypothèse d'induction, on a $[\![\Gamma]\!] \vdash [\![M]\!] : \mathbf{open}\,[\![B]\!]$ et $[\![\Gamma]\!] \vdash [\![B']\!] : \mathbf{open}\,[\![\star]\!]$. On en déduit par conversion que $[\![\Gamma]\!] \vdash \mathbf{open}\,[\![B']\!] : \star$. Et en enfin d'après le lemme précédent, on a $\mathbf{open}\,[\![B]\!] \equiv \mathbf{open}\,[\![B']\!]$ et on peut donc montrer que $[\![\Gamma]\!] \vdash [\![M]\!] : \mathbf{open}\,[\![B']\!]$. ☺

On en déduit alors le théorème suivant :

2.4.16 Théorème (Incohérence de $I_{se}\mathcal{F}$)
Soit $\Gamma \vdash_{I_{se}\mathcal{F}} \tau : \star$, alors il existe M tel que alors $\Gamma \vdash_{I_{se}\mathcal{F}} M : \tau$.

Démonstration On sait qu'il existe π tel que $\vdash_{\lambda\star} \pi : \forall\alpha : \star.\alpha$ (voir section 1.1.10). Il suffit alors de prendre $M = [\![\pi]\!]\,(\mathbf{close}\ \tau)$. Or, on a bien $\vdash [\![\pi]\!] : \mathbf{open}\,[\![\forall\alpha : \star.\alpha]\!]$, c'est-à-dire par réduction du type que $\vdash [\![\pi]\!] : \forall\alpha : \mathbf{star}.\,\mathbf{open}\ \alpha$. Enfin, on bien $\Gamma \vdash \mathbf{close}\ \tau : \mathbf{star}$ et donc $\Gamma \vdash M : \mathbf{open}\,(\mathbf{close}\ \tau)$ et par réduction $\Gamma \vdash M : \tau$. ☺

L'élimination forte non-restreinte pour les inductifs ayant pour sorte une sorte imprédicative implique donc l'incohérence du système. La solution adoptée par Coq pour pallier ce problème est d'autoriser l'élimination forte uniquement pour détruire des "petits inductifs".

Un petit inductif est un inductif dont les arguments des constructeurs sont typables par une sorte minimale pour la relation $\mathcal{A}$:

2.4.17 Définition (Petit inductif)
Un petit inductif est un inductif

$$\texttt{Ind}(x_1 : Q_1, \ldots, x_p : Q_p, I : A, c_1 : C_1, \ldots, c_k : C_k)$$

dont les types des constructeurs $C_1, \ldots, C_k$ sont de la forme $C_j = \forall \overrightarrow{y : E_j}^{m_j}.I[\overrightarrow{x}^p]\,\overrightarrow{D_j}^n$ et tel que pour tout j et tout $1 \leq i \leq m_j$, il existe une sorte s minimale pour la relation $\mathcal{A}$ vérifiant : $x_1 : Q_1, \ldots, x_p : Q_p, y_1 : E_{j,1}, \ldots, y_i : E_{j,i} \vdash E_{j,i} : s$

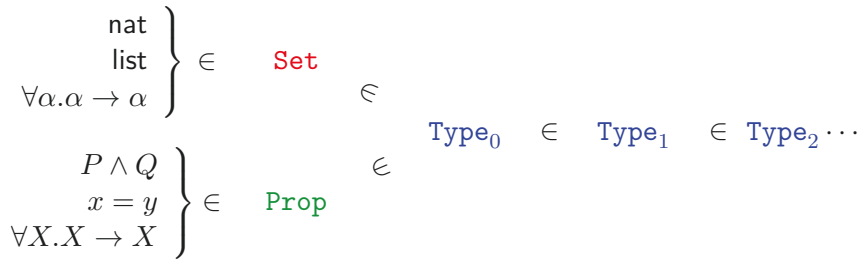
À titre d'exemple, l'inductif **star** vu précédemment n'est pas un petit inductif puisque l'argument $\star$ du constructeur **close** est de type $\Box$ (qui n'est pas minimal pour $\mathcal{A}$).

Nous allons maintenant voir comment cette définition est utilisée dans le Calcul des Constructions Inductives, c'est-à-dire le calcul sur lequel est basé l'assistant de preuve Coq.

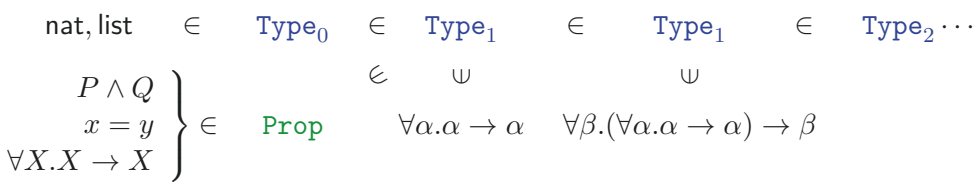
2.5 Les variantes du Calcul des Constructions Inductives

Le Calcul des Constructions Inductives avec univers (CCI) est le calcul sous-jacent à l'assistant de preuve Coq. Dans cette section, nous allons introduire trois variantes de ce calcul. Elles correspondent à trois organisations différentes de la hiérarchie des sortes :

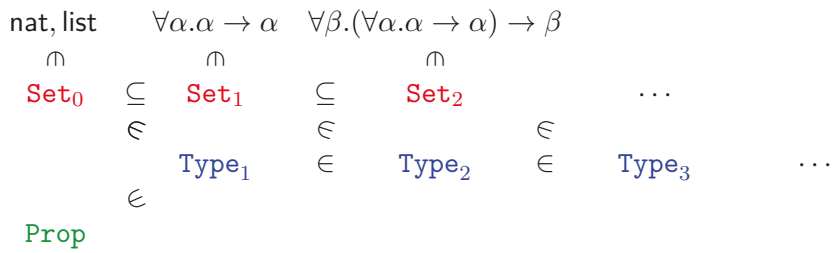
1. Le Calcul des Constructions Inductives avec `Set` imprédicatif que l'on notera CIC$^-$:



2. Le Calcul des Constructions Inductives (sans `Set`) que l'on notera CIC :



3. Le Calcul des Constructions Inductives avec `Set` prédicatif que l'on notera CIC$^+$:



Le système CIC$^-$ correspond au calcul implémenté dans le système Coq avant la version 8.0 (c'est-à-dire avant 2006). Le système CIC est celui qui est actuellement implémenté par Coq et enfin CIC$^+$ est une variante originale introduite dans ce travail.

Ces trois systèmes sont faiblement imprédicatifs, si on ignore les inductifs, ils se plongent dans le calcul des constructions CC_ω que nous avons étudié précédemment. Ils satisfont tous les trois les propriétés liées à la cumulativité suivantes (le dernier point est une conséquence du lemme 2.3.8 que nous avons admis) :

2.5.1 Lemme
Si $\mathcal{P}\in\{\mathrm{CIC}^-,\mathrm{CIC},\mathrm{CIC}^+\}$, *alors*

1. $\mathcal{P}\models \mathrm{CTS}$,
2. $\mathcal{P}\models \mathrm{SR}_\beta$,

3. $\mathcal{P} \models$ MINLOC,
4. $\mathcal{P} \models$ PRINCIPAL,
5. $\mathcal{P} \models$ UPSTABLE,
6. $\mathcal{P} \models$ FULL,
7. $\mathcal{P} \models$ WEAKLY-IMPREDICATIVE,
8. $\mathcal{P} \models$ STRENGTHENING,

Ils ont en commun la présence d'une sorte notée `Prop` dont les habitants sont des types interprétés comme des formules logiques.

La sorte des propositions. La sorte `Prop` permet de confiner tout le contenu logique, ce qui est rendu possible essentiellement grâce à son caractère imprédicatif qui autorise la quantification sur tous les objets du système. Ainsi, si τ est un type bien formé de sorte s, c'est-à-dire tel que $\Gamma \vdash \tau : s$, et si P est une proposition avec une variable libre x de type τ, c'est-à-dire telle que $\Gamma, x : \tau \vdash P :$ `Prop`, alors la quantification $\forall x : \tau.P$ est une proposition valide dans toutes les variantes du calcul des constructions : on a toujours $\Gamma \vdash \forall x : \tau.P :$ `Prop`. L'imprédicativité permet, comme nous avons pu le voir auparavant, d'encoder les différents connecteurs logiques. En pratique dans le calcul des constructions inductives, on choisit plus souvent d'encoder ces derniers par des types inductifs de sorte `Prop`. À titre d'exemple, on peut donner l'encodage de la disjonction :

```
Inductive or (P Q : Prop) : Prop :=
  | left : P → or P Q
  | right : Q → or P Q.
```

Cet inductif dispose de deux constructeurs pour construire une preuve de $P \vee Q$, l'un à partir d'une preuve de P et l'autre à partir d'une preuve de Q. Il peut-être représenté dans notre syntaxe par la déclaration suivante :

$$\mathsf{Ind}(P : \mathtt{Prop}, Q : \mathtt{Prop}, \mathsf{or} : \mathtt{Prop}, \mathsf{left} : P \to \mathsf{or}\ P\ Q, \mathsf{right} : Q \to \mathsf{or}\ P\ Q)$$

Et on pourra par exemple utiliser la construction `case` pour prouver que $\forall P : \mathtt{Prop}, \mathsf{or}\ P\ P \to P$:

$$\lambda P : \mathtt{Prop}, h : \mathsf{or}\ P\ P.\ \mathtt{case}_{\mathsf{or}}(P, P, h, \lambda\_ : \mathsf{or}\ P\ P.P, \lambda x : P.x, \lambda x : P.x)$$

Ce qui donnerait dans la syntaxe de Coq le terme suivant :

```
fun (P : Prop) (h : or P P) ⇒
  match h return P with
    | left x ⇒ x
    | right x ⇒ x
  end
```

De la même façon, on encode le vrai logique, noté True, le faux logique False, la conjonction, l'égalité de Leibniz, et le connecteur existentiel par les inductifs suivants :

```
Inductive True : Prop :=
  | tauto : True.

Inductive False : Prop :=.

Inductive and (P Q : Prop) : Prop :=
  | conj : P → Q → and P Q.

Inductive eq (X : Type) (x : X) : X→ Prop :=
  | eq_refl : eq X x x.

Inductive ex (X : Type) (P : X→ Prop) :=
  | ex_intro : forall x:X, P x→ ex X P.
```

On utilisera dans les suite les notations habituelles pour alléger ces encodages :

Terme	Notation
True	$\top$
False	$\bot$
$P \to$ False	$\neg P$
and $P\,Q$	$P \wedge Q$
or $P\,Q$	$P \vee Q$
eq $\tau\,x\,y$	$x =_\tau y$
ex $X\,(\lambda x : X.P)$	$\exists x : X.P$

Un des principaux attraits du Calcul des Constructions est sa compatibilité avec des axiomes classiques. Le premier que l'on peut vouloir considérer est le tiers-exclu. Cet axiome peut s'énoncer dans le calcul des constructions par le terme suivant :

$$\texttt{EM} = \forall X : \texttt{Prop}.\neg X \vee X$$

Or, Thierry Coquand a prouvé [Coq89] que si on dispose d'un schéma d'élimination dépendante pour la disjonction (ce qui est le cas lorsqu'on encode la disjonction par un type inductif) alors le tiers-exclu EM implique l'équivalence des preuves PI. L'équivalence des preuves s'énonce par la proposition suivante :

$$\mathsf{PI} = \forall P : \texttt{Prop}, h_1\,h_2 : P, h_1 =_P h_2$$

Elle affirme que deux preuves d'une même proposition sont indistinguables. La compatibilité du système avec cet axiome a pour corollaire qu'il est impossible d'écrire dans le système une fonction non constante de type $\top \vee \top \to$ bool. En effet, si on dispose d'une telle fonction f vérifiant $f\,(\mathsf{left\ tauto}) = \mathsf{true}$ et $f\,(\mathsf{right\ tauto}) = \mathsf{false}$, alors on montrerait facilement que $\mathsf{PI} \Rightarrow \mathsf{true} = \mathsf{false}$ et donc que le système est incompatible avec PI et donc *a fortiori* avec EM. C'est pourquoi il est important de restreindre les utilisations de case qui détruisent un inductif de sorte Prop pour fabriquer un objet dont le type n'est pas une proposition.

Ainsi le système Coq rejette la définition suivante :

```
Definition f (h : True ∨ True) : bool :=
  match h return bool with
    | left _  ⇒  true
    | right _  ⇒  false
  end.
```

L'interpréteur de Coq retourne le message suivant :

```
Error: Incorrect elimination of ‘‘h’’ in the inductive type ‘‘or’’:
the return type has sort ‘‘Set’’ while it should be ‘‘Prop’’.
Elimination of an inductive object of sort Prop
is not allowed on a predicate in sort Set
because proofs can be eliminated only to build proofs.
```

Néanmoins, on ne veut pas interdire toutes les éliminations des inductifs propositionnels vers autre chose que des propositions sinon nous n'aurions aucune interaction entre les termes calculatoires et les termes propositionnels. On veut en effet pouvoir utiliser les propositions pour spécifier des pré-conditions qui doivent être satisfaites pour appliquer certaines fonctions. On peut par exemple vouloir implémenter la division entière par une fonction du type :

$$\mathsf{nat} \to \forall y : \mathsf{nat}\,.\, y \neq_{\mathsf{nat}} 0 \to \mathsf{nat}$$

L'implémentation de la division aura alors besoin d'être capable de construire un objet de type nat dans la branche sensée être impossible : celle où y vaut 0. Pour ça il faut pouvoir détruire une preuve de $\bot$ pour construire un objet de type quelconque. Ce qui explique que le programme suivant soit accepté :

```
Definition absurd (X : Type) (h : False) : X :=
 match h return X with end.
```

Enfin, on mentionne qu'il peut-être également utile de détruire des égalités pour pouvoir réécrire dans les types dépendants. Ceci est utile par exemple pour construire un terme de type :

$$\forall \alpha : \mathsf{Set}, n : \mathsf{nat}\,.\, \mathsf{vect}[\alpha]\,(n - n)$$

Par exemple, on peut implémenter un tel terme en Coq par le terme suivant :

```
fun (α : Set) (n : nat)  ⇒
  match π n as _ = x return vect α x with
    eq_refl     ⇒    nilV
  end.
```

où $\pi : \forall n : \mathsf{nat}\,.0 = n - n$ est une preuve que $n - n = 0$.

Il y a un donc un critère sur la forme des inductifs pour décider si celui-ci peut-être détruit pour fabriquer autre chose qu'un habitant d'une proposition. Ce critère doit être suffisamment général pour que l'on puisse détruire les égalités et l'absurdité logique tout en restant compatible avec l'équivalence des preuves.

Le critère retenu par Coq est le suivant :

– Il faut que cet inductif ait au plus un constructeur,

– Et que tous les arguments de ce constructeur soient propositionnels.
Ce que l'on formalise par la définition ci-dessous :

2.5.2 Définition (Petit inductif logique)
Un petit inductif logique est un inductif

$$\texttt{Ind}(x_1 : Q_1, \ldots, x_p : Q_p, I : A, c_1 : C_1, \ldots, c_k : C_k)$$

de sorte Prop, avec $k \leq 1$ et dont le type de l'éventuel constructeur C est de la forme : $C = \forall \overrightarrow{y : E}^m . I[\overrightarrow{x}^p] \overrightarrow{D}^n$ et tel que pour tout j et tout $1 \leq i \leq m$, on ait : $x_1 : Q_1, \ldots, x_p : Q_p, y_1 : E_1, \ldots, y_i : E_i \vdash E_i : \texttt{Prop}$

Les inductifs true, false, and et eq sont des petits inductifs logiques tandis que or et ex n'en sont pas (l'un parce qu'il a deux constructeurs et l'autre parce que ex_intro a un argument de type X et que X n'est pas une proposition).

Le Calcul des Constructions Inductives avec Set imprédicatif. Nous pouvons maintenant présenter la première variante du système de types de Coq. Il est défini dans notre cadre par la spécification suivante :

$$\begin{array}{rcll}
\mathcal{S}_{\mathrm{CIC}^-} & = & \{ \quad \texttt{Prop}, \texttt{Set}, \texttt{Type}_i & \mid \ i < \omega \} \\
\mathcal{A}_{\mathrm{CIC}^-} & = & \{ \quad (\texttt{Prop}, \texttt{Type}_0) & \} \\
 & \cup & \{ \quad (\texttt{Set}, \texttt{Type}_0) & \} \\
 & \cup & \{ \quad (\texttt{Type}_i, \texttt{Type}_{i+1}) & \mid \ i < \omega \} \\
\mathcal{R}_{\mathrm{CIC}^-} & = & \{ \quad (s, \texttt{Prop}, \texttt{Prop}) & \mid \ s \in \mathcal{S}_{\mathrm{CIC}^-} \} \\
 & \cup & \{ \quad (s, \texttt{Set}, \texttt{Set}) & \mid \ s \in \mathcal{S}_{\mathrm{CIC}^-} \} \\
 & \cup & \{ \quad (\texttt{Prop}, \texttt{Type}_i, \texttt{Type}_i\} & \mid \ i < \omega \} \\
 & \cup & \{ \quad (\texttt{Set}, \texttt{Type}_i, \texttt{Type}_i\} & \mid \ i < \omega \} \\
 & \cup & \{ \quad (\texttt{Type}_i, \texttt{Type}_j, \texttt{Type}_j) & \mid \ i \leq j < \omega \ \} \\
\mathcal{C}_{\mathrm{CIC}^-} & = & \{ \quad (\texttt{Type}_i, \texttt{Type}_j) & \mid \ i < j < \omega \}
\end{array}$$

et $(I, r) \in \mathcal{E}_{\mathrm{CIC}^-}$ si et seulement si :
– Soit I est un inductif de sorte $\texttt{Type}_i$,
– Soit I est un inductif de sorte Prop et $r = \texttt{Prop}$,
– Soit I est un inductif de sorte Set et $r = \texttt{Prop}$ ou $r =$ Set,
– Soit I est un petit inductif de sorte Set.
– Soit I est un petit inductif logique.

Les sortes Prop et Set sont toutes les deux imprédicatives. La sorte Set est utilisée pour typer les types de données, la sorte Prop les propositions et la hiérarchie de sortes Type permet de donner un type aux sortes. Si l'on omet la présence des inductifs, CIC^- se plonge dans le Calcul des Constructions avec univers CC_ω vu précédemment par le morphisme :

$$\begin{array}{rcl}
\texttt{Prop} & \mapsto & \star \\
\texttt{Set} & \mapsto & \star \\
\texttt{Type}_i & \mapsto & \star_i
\end{array}$$

Dans ce calcul, la sorte `Prop` se distingue de la sorte `Set` uniquement de par les éliminations autorisées pour détruire les inductifs. Comme nous l'avons vu, ces restrictions garantissent la compatibilité avec l'équivalence des preuves et comme nous le verrons plus avant la possibilité d'effacer le contenu logique afin d'extraire le contenu calculatoire des termes.

Le passage de CIC$^-$ à CIC. Dans le système CIC$^-$ la sorte `Set` est imprédicative. Elle est utilisée pour contenir les types de données. Or, il a été démontré par Herman Geuvers [Geu01] en adaptant la preuve du paradoxe de Hurkens [Hur95] que l'imprédicativité de `Set` était incompatible avec l'existence d'une rétraction de `Prop` vers bool. Une rétraction est la donnée de deux fonction $f : \texttt{Prop} \to \mathsf{bool}$ et $g : \mathsf{bool} \to \texttt{Prop}$ qui satisfait :

$$\forall P : \texttt{Prop}\,.g\,(f\,P) \leftrightarrow P$$

Herman Geuvers montre dans [Geu01] que l'on peut dériver dans CIC$^-$ le séquent suivant :

$$\vdash \forall f : \texttt{Prop} \to \mathsf{bool}, g : \mathsf{bool} \to \texttt{Prop}, (\forall P : \texttt{Prop}\,.g\,(f\,P) \leftrightarrow P) \to \bot$$

Si on admet la cohérence de CIC$^-$, cela montre que l'on ne peut pas construire une telle rétraction dans un contexte vide. Mais cette découverte a pu déplaire aux utilisateurs de Coq qui utilisaient le système pour formaliser des preuves classiques. En effet, la conjonction du tiers-exclu `EM` et d'un axiome de description Descr implique l'existence d'une rétraction de `Prop` dans bool. L'axiome de description s'énonce de la façon suivante :

$$\mathsf{Descr} = \forall A : \texttt{Set}, P : A \to \texttt{Prop}\,.(\exists! x : A.P\,x) \to \{x : A | P\,x\}$$

Il contient deux notations de Coq que nous n'avons pas eu l'occasion de voir jusqu'à présent :

- Le connecteur d'existence unique :

$$\exists! x : A.P\,x = \exists x : A.P\,x \wedge \forall x\,y : A.P\,x \to P\,y \to x =_A y$$

  Il exprime le fait qu'il existe un unique objet x de type A qui satisfait le prédicat P.
- Le type "sous-ensemble" : $\{x : A | P\,x\}$ désigne le type des sous-ensembles des éléments de A qui vérifient la propriété P. Il est encodé par l'inductif suivant :

```
Inductive sig (A:Set) (P:A→ Prop) : Set :=
  exist : forall x:A, P x→ sig P.
```

  Ainsi $\{x : A |\ P\,x\}$ est une notation pour l'inductif `sig` $A\,P$. Il peut-être vu comme un couple dépendant contenant un objet x de type A et une preuve de $P\,x$.

Cet axiome est très utile en logique classique puisqu'il donne la possibilité d'obtenir un nom pour un objet dont on a prouvé l'existence unique. Or il est possible de s'en servir pour implémenter une rétraction de `Prop` dans bool de la façon suivante :

$$\vdash \lambda P : \texttt{Prop}\,.\pi_1\,(\mathsf{Descr}\ \mathsf{bool}\,(\lambda b : \mathsf{bool}\,.b = \mathsf{true} \leftrightarrow P)\,H) : \texttt{Prop} \to \mathsf{bool}$$

où π_1 est la première projection de `sig` et où H est une preuve utilisant l'axiome `EM` de :

$$\exists! b : \mathsf{bool}\,.b = \mathsf{true} \leftrightarrow P$$

Cette preuve H consiste à exhiber le témoin true lorsque P est vraie et false dans le cas contraire, et l'unicité vient du fait que l'on ne peut pas avoir à la fois P et $\neg P$. Et on montre facilement que cette fonction implémente une rétraction par rapport à la fonction $\lambda b : \mathsf{bool}\,.b =_{\mathsf{bool}} \mathsf{true}$.

L'équipe de Coq a donc cherché à corriger le système pour rétablir la compatibilité avec ces deux axiomes. La solution qui a été adoptée fut de supprimer la sorte `Set` et d'utiliser la hiérarchie des univers pour contenir les types de données. La constante `Set` a été conservée mais elle n'est aujourd'hui rien de plus qu'un synonyme pour $\texttt{Type}_0$. C'est ainsi que le système CIC s'est imposé pour devenir le système de types de Coq. Ce système se formalise dans notre cadre par les paramètres suivants :

$$\begin{array}{rcl}
\mathcal{S}_{\mathrm{CIC}} & = & \{ \quad \texttt{Prop}, \texttt{Type}_i \quad | \quad i < \omega \} \\
\mathcal{A}_{\mathrm{CIC}} & = & \{ \quad (\texttt{Prop}, \texttt{Type}_1) \quad \} \\
 & \cup & \{ \quad (\texttt{Type}_i, \texttt{Type}_{i+1}) \quad | \quad i < \omega \} \\
\mathcal{R}_{\mathrm{CIC}} & = & \{ \quad (s, \texttt{Prop}, \texttt{Prop}) \quad | \quad s \in \mathcal{S}_{\mathrm{CIC}} \} \\
 & \cup & \{ \quad (\texttt{Prop}, \texttt{Type}_i, \texttt{Type}_i \} \quad | \quad i < \omega \} \\
 & \cup & \{ \quad (\texttt{Type}_i, \texttt{Type}_j, \texttt{Type}_j) \quad | \quad i \leq j < \omega \ \} \\
\mathcal{C}_{\mathrm{CIC}} & = & \{ \quad (\texttt{Type}_i, \texttt{Type}_j) \quad | \quad i < j < \omega \}
\end{array}$$

et $(I, r) \in \mathcal{E}_{\mathrm{CIC}}$ si et seulement si :

- Soit I est un inductif de sorte $\texttt{Type}_i$,
- Soit I est un inductif de sorte `Prop` et $r = \texttt{Prop}$,
- Soit I est un petit inductif logique.

Ce système est clairement inclus dans CIC^- ; il est obtenu en enlevant des paramètres tout ce qui concerne la sorte `Set`. On notera que l'identifiant "`Set`" est toujours utilisé comme un synonyme de $\texttt{Type}_0$ [1].

Nous verrons dans le paragraphe suivant que cette solution présente toutefois des désavantages qui nous ont conduits à proposer une nouvelle version du calcul de constructions inductives qui contient une hiérarchie de sortes `Set` pour confiner les types de données tout en restant compatible avec ces axiomes classiques.

L'extraction de Coq. Le fait que les programmes ne puissent pas dépendre des preuves permet au système Coq de fournir un mécanisme qui extrait le contenu calculatoire des définitions en fabriquant un programme non-typé qui calcule les mêmes valeurs que le terme dont il est extrait. Ainsi, si l'utilisateur fait confiance à ce mécanisme d'extraction il peut obtenir un programme efficace implémentant les fonctions qu'il a spécifiées dans Coq.

L'extraction constitue une des caractéristiques principales de l'assistant de preuve Coq. L'exemple le plus notable de son utilisation est le compilateur certifié du projet CompCert de Xavier Leroy et al. [Ler06]. Il s'agit d'un compilateur d'un sous-ensemble large du langage C vers plusieurs langages d'assemblage réalistes (ARM, x86 et PowerPC). En plus de l'implémentation effective du compilateur, le développement Coq contient une formalisation complète des sémantiques des langages sources et cibles ainsi qu'une preuve que la fonction de compilation les préserve. L'extraction fournit donc un programme efficace sensé générer le même code que celui qui aurait été généré par le terme Coq réduit par l'interprète de Coq.

1. Dans le système Coq, les indices des univers sont inférés par le système. Le mot-clé `Set` de coq n'est donc pas entièrement superflu puisqu'il permet à l'utilisateur de désigner explicitement le premier niveau d'univers.

Le système implémente une fonction d'extraction $\xi(\cdot)$ qui construit un programme non-typé à partir d'un terme bien typé de CIC. Elle procède en effaçant les sous-termes "non-informatifs", c'est-à-dire sans contenu calculatoire. On peut trouver la description de la fonction d'extraction pour CIC et CIC$^-$ dans [Let02, Let04, Let08].

Essentiellement, le critère pour décider si un sous-terme M de type T dans un contexte Γ est informatif consiste à inspecter la forme normale de son type. Ainsi M n'est pas informatif :

– Si T est une arité, c'est-à-dire convertible en un terme de la forme

$$T \equiv \forall x_1 : A_1, \ldots, x_n : A_n.s$$

– Ou si T est une proposition, c'est-à-dire tel que

$$\Gamma \vdash T : \texttt{Prop}$$

L'idée étant que les termes dont le type est une arité sont des constructeurs de type et doivent donc disparaître (les programmes extraits sont non-typés). De plus, les termes dont les types sont des propositions sont également effacés car les restrictions imposées sur l'élimination des inductifs garantissent que le contenu calculatoire des termes ne peut pas dépendre du contenu logique.

Dans les détails, l'implémentation de la fonction d'extraction est loin d'être triviale. Elle effectue des optimisations pour limiter la taille des programmes extraits, elle dispose d'heuristiques pour recouvrer des informations de typage lorsque le langage cible est typé (comme c'est le cas de *Haskell* et *Objective Caml*), et enfin elle doit gérer les multiples extensions de Coq. Bien que cette partie du code de Coq soit critique, peu de choses assurent que l'implémentation actuelle soit exempte de bug. Si une certification complète de l'extraction semble être une tâche ardue, nous pensons qu'à long terme elle peut être tout à fait envisageable. Des travaux ont par ailleurs déjà été entrepris dans cette direction. Stéphane Glondu [Glo09] a réussi le tour de force d'implémenter en Coq une fonction d'extraction pour CIC et à formaliser les preuves des propriétés suivantes (elles avaient été démontrées sur papier dans la thèse de Pierre Letouzey [Let04]) :

1. Si le programme $\xi(M)$ se réduit en $\xi(M')$, alors M se réduit en M' en au moins autant d'étapes.
2. Si un terme M se réduit en un terme M', alors le programme $\xi(M)$ peut se réduire vers le programme $\xi(M')$.

Le premier point prouve la normalisation du programme extrait à partir de la normalisation de CIC. Tandis que le second permet de garantir que la forme normale du programme extrait d'un terme M est identique au programme extrait de la forme normale de M.

Si cette approche de la correction est satisfaisante pour étudier le comportement de l'extraction sur les termes clos, elle n'est pas suffisamment souple pour garantir la correction de programmes s'interfaçant avec d'autres programmes qui ne sont pas obtenus par extraction. C'est en particulier problématique pour étudier les programmes manipulant des entrées-sorties ou effectuant d'autres appels systèmes. Nous pensons que pour prouver dans un cadre souple la correction de la fonction d'extraction nous avons en fait besoin d'une théorie de la réalisabilité. Une telle théorie de la réalisabilité fournirait des outils pour énoncer et prouver la correction des programmes extraits, mais permettrait également de spécifier les interfaces avec des programmes non-certifiés qui seraient représentés par des axiomes du système. On trouve dans la littérature sur l'extraction dans le calcul des constructions essentiellement deux notions de réalisabilité :

– Dans la thèse de Christine Paulin [PM89b], on trouve une construction de réalisabilité pour pour le calcul des constructions avec `Set` et `Prop` imprédicatifs. Elle permet de prouver la correction d'un programme extrait typable dans $\mathcal{F}_\omega$.
– Dans la thèse de Pierre Letouzey [Let04], on trouve une notion de réalisabilité plus complexe grâce à laquelle il parvient à prouver la correction de son extraction non-typée pour CIC.

Le Calcul des Constructions avec `Set` prédicatif Dans ce paragraphe, nous proposons un nouveau système qui présente un compromis entre CIC et CIC$^-$. Le système obtenu peut être présenté par les paramètres suivants :

$$\begin{array}{rcll}
\mathcal{S}_{\mathrm{CIC}^+} & = & \{ \quad \mathtt{Prop}, \mathtt{Set}_i, \mathtt{Type}_i & \mid \ i < \omega\} \\
\mathcal{A}_{\mathrm{CIC}^+} & = & \{ \quad (\mathtt{Prop}, \mathtt{Type}_1) & \} \\
 & \cup & \{ \quad (\mathtt{Set}_i, \mathtt{Type}_1) & \} \\
 & \cup & \{ \quad (\mathtt{Type}_i, \mathtt{Type}_{i+1}) & \mid \ i < \omega\} \\
\mathcal{R}_{\mathrm{CIC}^+} & = & \{ \quad (s, \mathtt{Prop}, \mathtt{Prop}) & \mid \ s \in \mathcal{S}_{\mathrm{CIC}^+}\} \\
 & \cup & \{ \quad (\mathtt{Prop}, \mathtt{Type}_i, \mathtt{Type}_i)\} & \mid \ i < \omega\} \\
 & \cup & \{ \quad (\mathtt{Prop}, \mathtt{Set}_i, \mathtt{Set}_i)\} & \mid \ i < \omega\} \\
 & \cup & \{ \quad (\mathtt{Set}_i, \mathtt{Set}_j, \mathtt{Set}_j) & \mid \ i \leq j < \omega \ \} \\
 & \cup & \{ \quad (\mathtt{Type}_i, \mathtt{Set}_j, \mathtt{Set}_j) & \mid \ i \leq j < \omega \ \} \\
 & \cup & \{ \quad (\mathtt{Set}_i, \mathtt{Type}_j, \mathtt{Type}_j) & \mid \ i \leq j < \omega \ \} \\
 & \cup & \{ \quad (\mathtt{Type}_i, \mathtt{Type}_j, \mathtt{Type}_j) & \mid \ i \leq j < \omega \ \} \\
\mathcal{C}_{\mathrm{CIC}^+} & = & \{ \quad (\mathtt{Set}_i, \mathtt{Set}_j) & \mid \ i < j < \omega\} \\
 & \cup & \{ \quad (\mathtt{Type}_i, \mathtt{Type}_j) & \mid \ i < j < \omega\}
\end{array}$$

et $(I, r) \in \mathcal{E}_{\mathrm{CIC}^+}$ si et seulement si :
– Soit I est un inductif de sorte $\mathtt{Set}_i$,
– Soit I est un inductif de sorte `Prop` et $r = \mathtt{Prop}$,
– Soit I est un petit inductif logique.

Il contient toujours la sorte `Prop` imprédicative pour représenter les propositions, mais il contient également une hiérarchie de sortes $\mathtt{Set}_0$, $\mathtt{Set}_1$, ... permettant de confiner les types de données. Les encodages inductifs des types de données usuels (nat, bool, list$[\alpha]$, ...) habitent la sorte $\mathtt{Set}_0$ tandis que les types polymorphes ($\forall \alpha : \mathtt{Set}_i, \alpha \to \alpha$, $\forall \alpha : \mathtt{Set}_i$, list$[\alpha]$, ...) habiterons la sorte $\mathtt{Set}_{i+1}$ (pour i fixé). On notera que l'on dispose de relations de cumulativité entre les sortes $\mathtt{Set}_i$ et $\mathtt{Set}_{i+1}$ mais que le système reste construit de façon à ce que les sortes restent minimales pour la relation $\mathcal{A}_{\mathrm{CIC}^+}$.

Ce système fournit une définition plus naturelle de ce que signifie "être informatif" :

2.5.3 Définition (Type informatif et Programme dans CIC$^+$)
1. On appellera type informatif (relativement à un contexte Γ) tout terme T pour lequel il existe i tel que $\Gamma \vdash_{\mathrm{CIC}^+} T : \mathtt{Set}_i$.
2. On appellera programme (relativement à un contexte Γ) tout terme M tel que $\Gamma \vdash_{\mathrm{CIC}^+} M : T$ et T est un type informatif.

Le système CIC$^+$ est donc construit avec l'idée que l'on doit pouvoir reconnaître par le typage les sous-termes qui ne doivent pas être effacés par la fonction d'extraction. Ainsi dans ce système, les habitants des habitants de `Type` sont :

– Soit des arités, c'est-à-dire des termes de la forme :

$$\forall x_1 : A_1, \ldots, x_n : A_n.s$$

– Soit le type des "constructeurs d'arité", c'est-à-dire des termes du langage qui calculent sur les arités. Un exemple de tel programme serait le terme :

$$\texttt{relation}_i = \lambda X : \texttt{Set}_i \,.X \to X \to \texttt{Prop}$$

Il fournit la signature d'une relation dont le support est le type informatif qu'il prend en paramètre.

Les éliminations correspondent à des branchements dans le programme. C'est pourquoi on interdit les inductifs de sorte `Type`. Ainsi, on a purgé la hiérarchie `Type` de tout contenu calculatoire.

Nous avons introduit le système CIC$^+$ dans l'article [KL12] co-écrit avec Chantal Keller afin de développer une théorie de la paramétricité pour le système Coq. Nous verrons plus loin que le fait que les sortes `Set` soit minimales pour la relation $\mathcal{A}$ est un ingrédient essentiel pour construire des relations logiques représentées par des prédicats à valeur dans `Prop`. Mais nous pensons que ce système pourrait par ailleurs avoir de bonnes propriétés vis-à-vis de l'extraction et l'effacement des dépendances. Une implémentation de l'extraction non-typée pourrait alors procéder en effaçant tous les sous-termes qui ne sont pas des programmes.

Chapitre 3

Des programmes à la logique

Dans ce chapitre nous décrivons une méthode systématique pour construire une logique à partir d'un système de type pur cumulatif. Les formules de cette logique expriment des propriétés concernant les programmes. Nous verrons les raisons pour lesquelles cette logique constitue le cadre adéquat pour développer à la fois une théorie de la réalisabilité et une théorie de la paramétricité. De plus, nous constaterons que la seconde se définit à partir de la première.

Une des ambitions principales de la théorie de la démonstration est de donner une interprétation calculatoire aux démonstrations mathématiques ; c'est-à-dire, étant donné un système de preuve, chercher à observer les preuves comme des objets "calculant". Dans ce chapitre, nous adopterons une approche opposée : notre point de départ sera un langage de programmation et nous verrons comment construire une logique dont les preuves s'interprèteront comme des programmes du langage de départ. Nous intéresserons tout particulièrement à la question suivante : "Étant donné un langage de programmation, peut-on construire une logique dont les réalisateurs des formules sont les programmes ?" Afin d'y répondre, nous serons amenés à répliquer la structure du langage de programmation au niveau des démonstrations tout en rajoutant le minimum de règles pour en faire une logique utilisable. Ainsi, le pouvoir expressif de la logique sera directement conditionné par les constructions disponibles dans le langage de programmation. À titre d'exemple, la possibilité de construire des couples de programmes induira la présence d'une conjonction dans la logique.

3.1 Logique engendrée par un langage de programmation

On décrit dans cette section la logique que l'on utilisera pour raisonner à propos des expressions d'un CTS arbitraire $\mathcal{P}$. Elle sera elle-même décrite comme un CTS que l'on notera $\mathcal{P}^2$.

3.1.1 Définition (Logique engendrée par un CTS)
Étant donné un CTS $\mathcal{P}$, on définit $\mathcal{P}^2$ par les équations suivantes :

$$\begin{array}{rcl}
\mathcal{S}_{\mathcal{P}^2} & = & \mathcal{S}_{\mathcal{P}} \cup \{ \lceil s \rceil \mid s \in \mathcal{S}_{\mathcal{P}} \} \\
\mathcal{A}_{\mathcal{P}^2} & = & \mathcal{A}_{\mathcal{P}} \cup \{ (\lceil s_1 \rceil, \lceil s_2 \rceil) \mid (s_1, s_2) \in \mathcal{A}_{\mathcal{P}} \} \\
\mathcal{R}_{\mathcal{P}^2} & = & \mathcal{R}_{\mathcal{P}} \cup \{ (\lceil s_1 \rceil, \lceil s_2 \rceil, \lceil s_3 \rceil), (s_1, \lceil s_3 \rceil, \lceil s_3 \rceil) \mid (s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}} \} \\
 & & \cup \{ (s_1, \lceil s_2 \rceil, \lceil s_2 \rceil) \mid (s_1, s_2) \in \mathcal{A}_{\mathcal{P}} \} \\
\mathcal{C}_{\mathcal{P}^2} & = & \mathcal{C}_{\mathcal{P}} \cup \{ (\lceil s_1 \rceil, \lceil s_2 \rceil) \mid (s_1, s_2) \in \mathcal{C}_{\mathcal{P}} \}
\end{array}$$

Puisque l'on verra $\mathcal{P}$ comme un langage de programmation et $\mathcal{P}^2$ comme une logique pour raisonner sur les programmes, nous adopterons les conventions suivantes : on appellera

- *sorte de premier niveau* les sortes $s \in \mathcal{S}_\mathcal{P}$ et on appellera *sorte de second niveau* les sortes $\lceil s \rceil \in \mathcal{S}_{\mathcal{P}^2} \setminus \mathcal{S}_\mathcal{P}$.
- *type* les habitants d'une sorte de premier niveau et *programmes* les habitants d'un type.
- *propositions* les habitants d'une sorte de second niveau et *preuves* les habitants d'une proposition.

De plus, on dira que les types et les programmes sont des *termes de premier niveau* et les preuves et les propositions des *termes de second niveau*.

Explications de la construction. On dispose de deux morphismes qui vont de $\mathcal{P}$ dans $\mathcal{P}^2$:

- L'identité : on a bien $\mathcal{P} \subseteq \mathcal{P}^2$,
- Le soulèvement : défini par le morphisme ci-dessous.

$$\begin{array}{rcl} \lceil\cdot\rceil : \mathcal{P} & \longrightarrow & \mathcal{P}^2 \\ s & \longmapsto & \lceil s \rceil \end{array}$$

  Toutes les sortes, axiomes et règles ont été dupliqués pour fabriquer le second niveau, il est donc clair que $\lceil\cdot\rceil$ est un morphisme.

Le CTS $\mathcal{P}^2$ peut être perçu comme deux copies du système $\mathcal{P}$ auxquelles nous avons ajouté deux familles de règles :

- Pour chaque règle $(s_1, s_2, s_3) \in \mathcal{R}_\mathcal{P}$ du système de départ, on rajoute dans $\mathcal{R}_{\mathcal{P}^2}$ la règle $(s_1, \lceil s_3 \rceil, \lceil s_3 \rceil)$. Cette règle permet de quantifier sur les programmes dont le type est de sorte s_1 dans les formules de sorte s_3.
- Pour chaque axiome $(s_1, s_2) \in \mathcal{A}_\mathcal{P}$, on rajoute dans $\mathcal{R}_{\mathcal{P}^2}$ la règle $(s_1, \lceil s_2 \rceil, \lceil s_2 \rceil)$. Elle autorise la construction des arités de sorte $\lceil s_1 \rceil$, c'est-à-dire des termes de la forme $\forall x_1 : A_1, \ldots, x_n : A_n.\lceil s_1 \rceil$ où les A_i habitent s_1. Ces arités nous permettront de construire des prédicats sur les programmes.

La logique du second-ordre (avec individus d'ordre supérieur). Un premier exemple de logique générée par cette construction est $\mathcal{F}^2$, la logique engendrée par système $\mathcal{F}$.

3.1.2 Exemple

Le PTS $\mathcal{F}^2$ est spécifié par les équations suivantes :

$$\begin{array}{lcll} \mathcal{S}_{\mathcal{F}^2} & = & \{ & \star, \Box, \lceil\star\rceil, \lceil\Box\rceil \quad\} \\ \mathcal{A}_{\mathcal{F}^2} & = & \{ & (\star, \Box), (\lceil\star\rceil, \lceil\Box\rceil) \quad\} \\ \mathcal{R}_{\mathcal{F}^2} & = & \{ & (\star,\star,\star), (\Box,\star,\star), (\lceil\star\rceil,\lceil\star\rceil,\lceil\star\rceil), (\lceil\Box\rceil,\lceil\star\rceil,\lceil\star\rceil) \\ & & & (\star,\lceil\Box\rceil,\lceil\Box\rceil), (\star,\lceil\star\rceil,\lceil\star\rceil), (\Box,\lceil\star\rceil,\lceil\star\rceil) \quad\} \\ \mathcal{C}_{\mathcal{F}^2} & = & \emptyset & \end{array}$$

Ce système est une présentation de la logique du second ordre dont les individus sont des termes de système $\mathcal{F}$. Elle contient en plus de système $\mathcal{F}$:

- Une sorte $\lceil\star\rceil$ pour typer les propositions,
- Une règle $(\lceil\star\rceil, \lceil\star\rceil, \lceil\star\rceil)$ pour construire l'implication entre deux propositions :

$$X : \lceil\star\rceil, Y : \lceil\star\rceil \vdash X \to Y : \lceil\star\rceil$$

– Une règle $(\star, \lceil\Box\rceil, \lceil\Box\rceil)$ pour construire le type des prédicats :

$$\alpha : \star \vdash \alpha \to \lceil\star\rceil : \lceil\Box\rceil$$

– Une règle $(\star, \lceil\star\rceil, \lceil\star\rceil)$ pour quantifier sur les programmes :

$$\alpha : \star, X : \alpha \to \lceil\star\rceil \vdash \forall \boldsymbol{x : \alpha}. X\, x : \lceil\star\rceil$$

– La règle d'imprédicativité dans les propositions pour quantifier sur tous les prédicats :

$$\alpha : \star \vdash \forall \boldsymbol{X : \alpha \to \lceil\star\rceil}, x : \alpha. X\, x : \lceil\star\rceil$$

– Et enfin la règle $(\Box, \lceil\star\rceil, \lceil\star\rceil)$ pour quantifier sur les types :

$$\vdash \forall \boldsymbol{\alpha : \star}, X : \alpha \to \lceil\star\rceil, x : \alpha. X\, x : \lceil\star\rceil$$

La logique $\mathcal{F}^2$ est une logique du second-ordre dans la mesure où l'on peut quantifier à la fois sur les individus et sur les "ensembles d'individus".

Cette logique se plonge dans CIC^- par le morphisme suivant :

$$\begin{array}{rcl} \mathcal{F}^2 & \longrightarrow & \mathrm{CIC}^- \\ \star & \longmapsto & \texttt{Set} \\ \lceil\star\rceil & \longmapsto & \texttt{Prop} \\ \Box & \longmapsto & \texttt{Type} \\ \lceil\Box\rceil & \longmapsto & \texttt{Type} \end{array}$$

Dans le paragraphe suivant nous allons voir les pré-réquis que doivent satisfaire les logiques engendrées pour être des logiques "raisonnables", c'est-à-dire capables de représenter les preuves d'arithmétique.

3.2 Systèmes de notations pour l'arithmétique

Le système $\mathcal{F}^2$ est suffisamment expressif pour représenter des formules et des preuves de l'arithmétique. Il contient l'arithmétique du second ordre (AF_2) introduite par Daniel Leivant [Lei83, Lei90] et Jean-Louis Krivine [Kri93]. Dans ces systèmes, les individus, c'est-à-dire l'ensemble des termes du premier ordre, sont décrits par des *expressions arithmétiques*. Les expressions arithmétiques sont générées à partir de variables et de symboles de fonctions d'arité fixée (par exemple $\mathrm{plus}(x, \mathrm{succ}(y))$ est une expression arithmétique). Les expressions arithmétiques ne présentent pas de notion de réduction et AF_2 ne contient pas d'équivalent de la règle de conversion. Les calculs dans AF_2 sont représentés par réécriture des propositions via des axiomes (comme par exemple $\forall x\, y. \mathrm{plus}(x, \mathrm{succ}(y)) = \mathrm{succ}(\mathrm{plus}(x, y))$).

Nous décrivons dans cette section, un système de notations pour l'arithmétique qui offre les connecteurs logiques nécessaires pour exprimer et prouver les propriétés arithmétiques. Elle contient un type pour représenter les entiers, et des termes pour représenter l'entier zéro, la fonction successeur, ainsi qu'un récurseur. Enfin, on disposera également d'atomes et de connecteurs logiques.

$$\frac{}{\vdash \bot : \mathsf{prop}} \qquad \frac{\Gamma \vdash P : \mathsf{prop} \qquad \Gamma \vdash Q : \mathsf{prop}}{\Gamma \vdash P \wedge Q : \mathsf{prop}}$$

$$\frac{\Gamma \vdash M : \mathsf{nat}}{\Gamma \vdash M \in \mathbb{N} : \mathsf{prop}} \qquad \frac{\Gamma, x : \tau \vdash P : \mathsf{prop}}{\Gamma \vdash \exists x : \tau . P : \mathsf{prop}}$$

$$\frac{}{\Gamma \vdash \mathsf{nat} : \mathsf{set}} \qquad \frac{}{\Gamma \vdash \overline{0} : \mathsf{nat}} \qquad \frac{}{\Gamma \vdash \mathsf{S} : \mathsf{nat} \to \mathsf{nat}} \qquad \frac{\Gamma \vdash \sigma : \mathsf{set} \qquad \Gamma \vdash \tau : \mathsf{set}}{\Gamma \vdash \sigma \times \tau : \mathsf{set}}$$

$$\frac{\Gamma \vdash \tau : \mathsf{set}}{\Gamma \vdash (\!|\tau|\!) : \mathsf{set}} \qquad \frac{\Gamma \vdash \tau : \mathsf{set}}{\Gamma \vdash \mathsf{open}[\tau] : (\!|\tau|\!) \to \tau} \qquad \frac{\Gamma \vdash \tau : \mathsf{set}}{\Gamma \vdash \mathsf{close}[\tau] : \tau \to (\!|\tau|\!)}$$

$$\frac{\Gamma \vdash \tau : \mathsf{set}}{\Gamma \vdash \mathsf{rec}[\tau] : (\tau \to \mathsf{nat} \to \tau) \to \tau \to \mathsf{nat} \to \tau}$$

$$\frac{\Gamma \vdash \sigma : \mathsf{set} \qquad \Gamma \vdash \tau : \mathsf{set}}{\Gamma \vdash \mathsf{pair}[\sigma, \tau] : \sigma \to \tau \to \sigma \times \tau}$$

$$\frac{\Gamma \vdash \sigma : \mathsf{set} \qquad \Gamma \vdash \tau : \mathsf{set}}{\Gamma \vdash \mathsf{proj}^1[\sigma, \tau] : \sigma \times \tau \to \sigma} \qquad \frac{\Gamma \vdash \sigma : \mathsf{set} \qquad \Gamma \vdash \tau : \mathsf{set}}{\Gamma \vdash \mathsf{proj}^2[\sigma, \tau] : \sigma \times \tau \to \tau}$$

$$\frac{\Gamma \vdash \tau : \mathsf{set} \qquad \Gamma \vdash M : \tau \qquad \Gamma \vdash N : \tau}{\Gamma \vdash M =_\tau N : \mathsf{prop}}$$

FIGURE 3.1 – Règles des systèmes de notations pour l'arithmétique (typage).

$$\frac{\Gamma \vdash \pi : \bot \qquad \Gamma \vdash P : \mathsf{prop}}{\Gamma \vdash \mathsf{elim}_{\bot}[P, \pi] : P} \qquad \frac{\Gamma \vdash \tau : \mathsf{set} \qquad \Gamma \vdash M : \tau}{\Gamma \vdash \mathsf{intro}_{=}[\tau, M] : M =_{\tau} M}$$

$$\frac{\Gamma \vdash \tau : \mathsf{set} \qquad \Gamma \vdash \pi : M =_{\tau} N \qquad \Gamma \vdash \varrho : P[M/x] \qquad \Gamma, x : \tau \vdash P : \mathsf{prop}}{\Gamma \vdash \mathsf{elim}_{=}[\tau, x, P, \pi, \varrho] : P[N/x]}$$

$$\frac{\Gamma \vdash \pi_1 : P \qquad \Gamma \vdash \pi_2 : Q \qquad \Gamma \vdash P : \mathsf{prop} \qquad \Gamma \vdash Q : \mathsf{prop}}{\Gamma \vdash \mathsf{intro}_{\wedge}[P, Q, \pi_1, \pi_2] : P \wedge Q}$$

$$\frac{\Gamma \vdash \pi : P \wedge Q \quad \Gamma \vdash P : \mathsf{prop} \quad \Gamma \vdash Q : \mathsf{prop}}{\Gamma \vdash \mathsf{elim\text{-}gauche}_{\wedge}[P, Q, \pi] : P} \qquad \frac{\Gamma \vdash \pi : P \wedge Q \quad \Gamma \vdash P : \mathsf{prop} \quad \Gamma \vdash Q : \mathsf{prop}}{\Gamma \vdash \mathsf{elim\text{-}droite}_{\wedge}[P, Q, \pi] : Q}$$

$$\frac{\Gamma \vdash M : \tau \quad \Gamma \vdash \pi : P[M/x] \quad \Gamma, x : \mathsf{nat} \vdash P : \mathsf{prop}}{\Gamma \vdash \mathsf{intro}_{\exists}[\tau, x, P, M, \pi] : \exists x : \tau.P}$$

$$\frac{\Gamma \vdash \pi : \exists x : \tau.P \qquad \Gamma \vdash Q : \mathsf{prop} \qquad \Gamma, x : \tau, h : P \vdash \varrho : Q}{\Gamma \vdash \mathsf{elim}_{\exists}[\tau, x, P, h, Q, \pi, \varrho] : Q}$$

$$\frac{}{\Gamma \vdash \mathsf{intro\text{-}zero}_{\mathbb{N}} : \overline{0} \in \mathbb{N}} \qquad \frac{\Gamma \vdash M : \mathsf{nat} \qquad \Gamma \vdash \pi : M \in \mathbb{N}}{\Gamma \vdash \mathsf{intro\text{-}succ}_{\mathbb{N}}[M, \pi] : (\mathsf{S}\, M) \in \mathbb{N}}$$

$$\frac{\Gamma, x : \mathsf{nat} \vdash P : \mathsf{prop} \quad \Gamma \vdash \pi : N \in \mathbb{N} \quad \Gamma, x : \mathsf{nat}, h : P \vdash \pi_1 : P[\mathsf{S}\, x/x] \quad \Gamma \vdash \pi_2 : P[\overline{0}/x]}{\Gamma \vdash \mathsf{induction}_{\mathbb{N}}[x, h, P, N, \pi, \pi_1, \pi_2] : P[N/x]}$$

$$\frac{\Gamma \vdash \tau : \mathsf{set}}{\Gamma \vdash \mathsf{open\text{-}close}[\tau] : \forall x : \tau.x =_{\tau} \mathsf{open}[\tau]\,(\mathsf{close}[\tau]\, x)}$$

$$\frac{\Gamma \vdash \tau_1 : \mathsf{set} \qquad \Gamma \vdash \tau_2 : \mathsf{set}}{\Gamma \vdash \mathsf{pair\text{-}proj}^k[\tau_1, \tau_2] : \forall x_1 : \tau_1, x_2 : \tau_2.x_k =_{\sigma} \mathsf{proj}^k[\tau_1, \tau_2]\,(\mathsf{pair}[\sigma, \tau]\, x_1\, x_2)}\ k = 1, 2$$

$$\frac{\Gamma \vdash \tau : \mathsf{set}}{\Gamma \vdash \mathsf{rec\text{-}succ}[\tau] : \forall f : \tau \to \mathsf{nat} \to \tau, x : \tau, n : \mathsf{nat}.n \in \mathbb{N} \to \mathsf{rec}[\tau]\, f\, x\, (\mathsf{S}\, n) =_{\tau} f\, (\mathsf{rec}[\tau]\, f\, x\, n)\, n}$$

$$\frac{\Gamma \vdash \tau : \mathsf{set}}{\Gamma \vdash \mathsf{rec\text{-}zero}[\tau] : \forall f : \tau \to \mathsf{nat} \to \tau, x : \tau.\mathsf{rec}[\tau]\, f\, x\, \overline{0} =_{\tau} x}$$

FIGURE 3.2 – Règles des systèmes de notations pour l'arithmétique (logique).

On appelle termes avec n trous, la donnée d'un terme $E[X_1, \ldots, X_n]$ dépendant de n "méta-variables". On distingue ces méta-variables par le fait que leur substitution ne prévient pas la capture des variables liées (par exemple, le terme avec trou $E[M] = \lambda x : A.M$, peut être instancié en $E[x] = \lambda x : A.x$). Dans un souci de lisibilité, nous prenons la liberté d'introduire des notations pour les termes avec trous et on indiquera le nombre de trous et leurs emplacements par le symbole "_".

3.2.1 Définition (Système de notations pour l'arithmétique)
Étant donné un CTSI $\mathcal{P}$ contenant deux sortes set et prop tel que

$$\{(\mathsf{set}, \mathsf{set}, \mathsf{set}), (\mathsf{prop}, \mathsf{prop}, \mathsf{prop}), (\mathsf{set}, \mathsf{prop}, \mathsf{prop})\} \subseteq \overline{\mathcal{R}_{\mathcal{P}}}$$

on appelle système de notations pour l'arithmétique la donnée $\mathbf{Syst}_{\mathcal{P}}(\mathsf{set}, \mathsf{prop})$ de :

1. Trois constructeurs de type [1] :

   $$\mathsf{nat},\ \_ \times \_,\ (\!|\_|\!)$$

   Ainsi que leurs primitives :

   $$\mathsf{S},\ \overline{0},\ \mathsf{proj}^1[\_],\ \mathsf{proj}^2[\_],\ \mathsf{pair}[\_],\ \mathsf{rec}[\_],\ \mathsf{open}[\_],\ \mathsf{close}[\_]$$

2. Cinq connecteurs logiques [2] :

   $$\bot,\ \_ \wedge \_,\ \_ =_{\_} \_,\ \_ \in \mathbb{N},\ \exists x : \tau.\_$$

   Ainsi que leurs règles d'introduction et d'élimination :

   $$\mathsf{elim}_{\bot}[\_],\ \mathsf{intro}_{\wedge}[\_,\_,\_,\_],\ \mathsf{elim\text{-}gauche}_{\wedge}[\_,\_,\_],\ \mathsf{elim\text{-}droite}_{\wedge}[\_,\_,\_],\ \mathsf{intro}_{\exists}[\_,\_,\_,\_],\ \mathsf{elim}_{\exists}[\_,\_,\_,\_,\_,\_],$$
   $$\mathsf{intro}_{=}[\_,\_],\ \mathsf{elim}_{=}[\_,\_,\_,\_,\_],\ \mathsf{intro\text{-}zero}_{\mathbb{N}},\ \mathsf{intro\text{-}succ}_{\mathbb{N}}[\_],\ \mathsf{induction}_{\mathbb{N}}[\_,\_,\_,\_,\_,\_]$$

3. Les règles de réécriture pour le produit, le récurseur et d'un opérateur "d'encapsulation" $(\!|\_|\!)$:

   $$\mathsf{pair\text{-}proj}^1[\_],\ \mathsf{pair\text{-}proj}^2[\_],\ \mathsf{rec\text{-}zero}[\_],\ \mathsf{rec\text{-}succ}[\_],\ \mathsf{open\text{-}close}[\_]$$

Ces notations sont telles que les règles des figures 3.1 et 3.2 sont admissibles. On notera $\mathcal{P} \models$ NOTATION pour signifier que $\mathcal{P}$ est muni d'un système de notations pour l'arithmétique.

1. On désigne ici informellement par constructeur de type les termes dont le type est une arité de sorte de premier niveau.
2. Les constructeur de type désignent les termes dont le type est une arité de sorte de second niveau.

3.2.2 Définition (Notations pour l'arithmétique)
Dans un CTSI muni d'un système de notations pour l'arithmétique, on adoptera les notations suivantes :

$$\begin{aligned}
\mathsf{plus} &= \lambda n\, m : \mathsf{nat.rec}[\mathsf{nat}]\,(\lambda x\, y : \mathsf{nat.S}\, x)\, m\, n \\
N + M &= \mathsf{plus}\, N\, M \\
\mathsf{mult} &= \lambda n\, m : \mathsf{nat.rec}[\mathsf{nat}]\,(\lambda x\, y : \mathsf{nat.plus}\, m\, x)\, \overline{0}\, n \\
N * M &= \mathsf{mult}\, N\, M \\
\mathsf{pred} &= \lambda n : \mathsf{nat.rec}[\mathsf{nat}]\,(\lambda x\, y : \mathsf{nat}.x)\, \overline{0}\, n \\
\mathsf{minus} &= \lambda n\, m : \mathsf{nat.rec}[\mathsf{nat}]\,(\lambda x\, y : \mathsf{nat.pred}x)\, n\, m \\
N - M &= \mathsf{minus}\, N\, M \\
\forall x \in \mathbb{N}.P &= \forall x : \mathsf{nat}.x \in \mathbb{N} \Rightarrow P \\
\exists x \in \mathbb{N}.P &= \exists x : \mathsf{nat}.x \in \mathbb{N} \wedge P \\
\bot_{\mathrm{w}} &= \forall x : \mathsf{nat}.x =_{\mathsf{nat}} \overline{0} \\
\neg_{\mathrm{w}}\, P &= P \to \bot_{\mathrm{w}} \\
P \vee Q &= \exists x \in \mathbb{N}.\overline{0} =_{\mathsf{nat}} x \to P \quad \wedge \quad \exists y : \mathsf{nat.S}\, y =_{\mathsf{nat}} x \to Q \\
P \leftrightarrow Q &= (P \to Q) \wedge (Q \to P)
\end{aligned}$$

On notera également $\overline{n}$ le terme $\mathsf{S}\,(\mathsf{S} \cdots \overline{0})$ avec n occurrences de S.

On désignera également par $\forall x_1 \cdots x_n \in \mathbb{N}.P$ et $\exists x_1 \cdots x_n \in \mathbb{N}.P$ les termes $\forall x_1 \in \mathbb{N}.\cdots \forall x_n \in \mathbb{N}.P$ et $\exists x_1 \in \mathbb{N}.\cdots \exists x_n \in \mathbb{N}.P$.

On adoptera la convention suivante lorsque que nous ne nous intéresserons pas à la forme précise des termes de preuves :

3.2.3 Convention
On notera dans la suite $\Gamma \vdash P$ le fait qu'il existe un terme π tel que $\Gamma \vdash \pi : P$. On utilisera cette notation pour toutes les relations de typage $\_ \vdash \_ : \_$. On utilisera cette notation lorsque que π sera un terme de preuve dont l'existence nous importera plus que sa forme. On se permettra de plus de décrire la construction de ces preuves dans un style informel.

On utilise par exemple cette convention dans le lemme suivant :

Étant donnée une proposition $\Gamma \vdash P :$ prop typable dans un contexte Γ, on notera $\Gamma \vdash P$ le fait qu'il existe un terme (de preuve) π tel que $\Gamma \vdash \pi : P$.

3.2.4 Lemme
Si $\models$ NOTATION*, alors on peut démontrer que la relation $\cdot =_\alpha \cdot$ est une relation d'équivalence :*

1. $\alpha : \mathsf{set} \vdash \forall x : \alpha.x =_\alpha x$,
2. $\alpha : \mathsf{set} \vdash \forall x\, y : \alpha.x =_\alpha y \to y =_\alpha x$,
3. $\alpha : \mathsf{set} \vdash \forall x\, y\, z : \alpha.x =_\alpha y \to y =_\alpha z \to x =_\alpha z$

Démonstration Les termes suivants sont des preuves respectives de ces énoncés :

1. $\lambda x : \alpha.\mathsf{intro}_=[\alpha, x]$

2. $\lambda x\, y : \alpha, h : x =_\alpha y.\mathsf{elim}_=[\alpha, z, z =_\alpha x, h, \mathsf{intro}_=[\alpha, x]]$
3. $\lambda x\, y\, z : \alpha, h_1 : x =_\alpha y, h_2 : y =_\alpha z.\mathsf{elim}_=[\alpha, z, x =_\alpha z, h_2, h_1]$ ☺

Le récurseur permet d'implémenter les opérations arithmétiques usuelles et de prouver que ces opérations satisfont les axiomes de l'arithmétique :

3.2.5 Lemme
Si $\models$ NOTATION, *alors*

$$\begin{array}{lll}
\vdash & \forall x \in \mathbb{N}. & x =_{\mathsf{nat}} \overline{0} + x \\
\vdash & \forall x\, y \in \mathbb{N}. & \mathsf{S}\,(x + y) =_{\mathsf{nat}} (\mathsf{S}\,x) + y \\
\vdash & \forall x \in \mathbb{N}. & \overline{0} =_{\mathsf{nat}} \overline{0} * x \\
\vdash & \forall x\, y \in \mathbb{N}. & x * y + y =_{\mathsf{nat}} (\mathsf{S}x) * y \\
\vdash & & \overline{0} =_{\mathsf{nat}} \mathsf{pred}\,\overline{0} \\
\vdash & \forall x \in \mathbb{N}. & x =_{\mathsf{nat}} \mathsf{pred}\,(\mathsf{S}\,x) \\
\vdash & \forall x \in \mathbb{N}. & x =_{\mathsf{nat}} x - \overline{0} \\
\vdash & \forall x\, y \in \mathbb{N}. & \mathsf{pred}\,(x - y) =_{\mathsf{nat}} x - (\mathsf{S}\,y)
\end{array}$$

Démonstration Les points impairs se prouvent directement en appliquant rec-zero[nat] et les points pairs en appliquant rec-succ[nat]. ☺

On peut par ailleurs montrer que tous les entiers clos satisfont le principe d'induction :

3.2.6 Lemme
Supposons $\mathcal{P} \models$ NOTATION.
Pour tout entier n, *on a* $\vdash \overline{n} \in \mathbb{N}$.

Démonstration On procède par récurrence sur n. Pour le cas de base, on utilise la règle $\mathsf{intro\text{-}zero}_{\mathbb{N}}$ et pour l'hérédité on utilise $\mathsf{intro\text{-}succ}_{\mathbb{N}}[\_]$. On remarque donc que la preuve construite dépend de la valeur de n (elle utilisera n fois la règle $\mathsf{intro\text{-}succ}_{\mathbb{N}}[\_]$). ☺

3.2.7 Lemme
Si $\models$ NOTATION, *alors*

$$\vdash \forall n \in \mathbb{N}.\overline{0} =_{\mathsf{nat}} n \vee \exists y : \mathsf{nat}.\mathsf{S}\,y =_{\mathsf{nat}} n$$

Démonstration Cette dérivation est tautologique : lorsque l'on déplie la notation $\_ \vee \_$ on obtient :

$$\vdash \forall n \in \mathbb{N}.\exists x \in \mathbb{N}.\overline{0} =_{\mathsf{nat}} x \to \overline{0} =_{\mathsf{nat}} n \quad \wedge \quad (\exists y : \mathsf{nat}.\mathsf{S}\,y =_{\mathsf{nat}} x) \to (\exists y : \mathsf{nat}.\mathsf{S}\,y =_{\mathsf{nat}} n)$$

Il suffit donc de prendre $x = n$ pour conclure. ☺

Nous pouvons démontrer les schémas d'introduction et d'élimination du connecteur $\_ \vee \_$:

3.2.8 Lemme
Si $\models$ NOTATION, *alors*

$$X : \mathsf{prop}, Y : \mathsf{prop} \vdash (X \to X \vee Y) \wedge (Y \to X \vee Y)$$

Démonstration On prouve $X \to X \vee Y$ en prenant $x = \overline{0}$ comme témoin du connecteur existentiel sous la notation et on prouve $Y \to X \vee Y$ en prenant comme témoin $x = \mathsf{S}\,\overline{0}$ pour prouver que $\neg_{\mathrm{w}}\,\overline{0} =_{\mathsf{nat}} x$.☺

3.2.9 Lemme
Si $\models$ NOTATION, *alors*

$$X : \mathsf{prop}, Y : \mathsf{prop}, Z : \mathsf{prop} \vdash (X \vee Y) \to (X \to Z) \to (Y \to Z) \to Z$$

Démonstration On procède en utilisant le principe d'induction sous la notation $X \vee Y$. Dans le cas de base, on utilise la réflexivité de l'égalité pour dériver X et pour prouver l'hérédité, on démontre Y en fournissant n comme témoin pour dériver $\exists y : \mathsf{nat}.(\mathsf{S}\,y) =_{\mathsf{nat}} (\mathsf{S}\,n)$ (l'hypothèse d'induction n'est pas utilisée). ☺

Bien que l'on peut représenter tous les énoncés de l'arithmétique à l'aide du système de notations présentés ci-dessus, les règles d'introduction et d'élimination ne sont pas suffisamment complètes pour représenter toutes les preuves de l'arithmétique intuitionniste. En effet, les deux axiomes ci-dessous seront, en général, indémontrables dans les systèmes que l'on considérera dans cette section. On verra néanmoins plus loin qu'il seront *réalisables*.

3.2.10 Définition
On nomme les formules suivantes :

$$\begin{aligned} \mathsf{succ\_inj} &= \forall x\, y : \mathsf{nat}.\mathsf{S}\,x =_{\mathsf{nat}} \mathsf{S}\,y \Rightarrow x =_{\mathsf{nat}} y \\ \mathsf{not\_cyclic} &= \forall x : \mathsf{nat}.\mathsf{S}\,x =_{\mathsf{nat}} \overline{0} \Rightarrow \bot_{\mathrm{w}} \end{aligned}$$

et on notera **HA** le contexte contenant succ_inj et not_cyclic.

Finalement, on admettra [3] que toutes les preuves de l'arithmétique de Heyting (la logique du premier ordre intuitionniste munie des axiomes de l'arithmétique de Peano) se représenteront dans les CTSI munis d'un système de notations pour l'arithmétique (dans un contexte où les axiomes **HA** sont démontrables).

Arithmétique dans $\mathcal{F}^2$. Dans cette thèse nous introduirons uniquement deux systèmes de notations, l'un dans $\mathcal{F}^2$ qui nous permettra d'étudier l'arithmétique sur les programmes utilisant les codages imprédicatifs des entiers de Church dans $\mathcal{F}$. Puis, plus loin nous verrons un système de notations utilisant les inductifs pour représenter les types de donnée.

Dans $\mathcal{F}^2$, l'utilisateur dispose de toute la puissance de système $\mathcal{F}$ pour écrire les termes du premier ordre. Ainsi, les propositions de $\mathcal{F}^2$ expriment des propriétés sur les termes de système $\mathcal{F}$ modulo réduction et non plus uniquement sur les expressions arithmétiques. Le système $\mathcal{F}^2$ est exactement celui décrit par Philip Wadler dans [Wad07] dans lequel il présente une version avec des individus typés de l'arithmétique du second ordre en s'inspirant de la logique pour la paramétricité de Gordon Plotkin et Martin Abadi [PA93] et de celle de Izumi Takeuti [Tak98].

3. Pour s'en convaincre, il suffit de plonger l'arithmétique de Heyting en utilisant les notations vue ci-dessous. Les quantificateurs $\forall x$ et $\exists x$ sont alors interprétés par les notations $\forall x \in \mathbb{N}$ et $\exists x \in \mathbb{N}$ et on remarque que tous les axiomes de l'arithmétique de Heyting sont démontrables dans le contexte **HA**.

$\mathbf{Syst}_{\mathcal{F}^2}(\star, \lceil\star\rceil) = \{$

$$
\begin{array}{rcl}
\mathsf{nat} & := & \forall\alpha:\star,(\alpha\to\alpha)\to\alpha\to\alpha \\
\mathsf{S} & := & \lambda n:\mathsf{nat},\alpha:\star,f:\alpha\to\alpha,x:\alpha.n\,\alpha\,f\,(f\,x) \\
\overline{0} & := & \lambda\alpha:\star,f:\alpha\to\alpha,x:\alpha.x \\
\sigma\times\tau & := & \forall\alpha:\star.(\sigma\to\tau\to\alpha)\to\alpha \\
\mathsf{pair}[\sigma,\tau] & := & \lambda x:\sigma,y:\tau,\alpha:\star,f:\sigma\to\tau\to\alpha.f\,x\,y \\
\mathsf{proj}^1[\sigma,\tau] & := & \lambda c:\sigma\times\tau.c\,\sigma\,(\lambda x:\sigma,y:\tau.x) \\
\mathsf{proj}^2[\sigma,\tau] & := & \lambda c:\sigma\times\tau.c\,\tau\,(\lambda x:\sigma,y:\tau.y) \\
(\!|\tau|\!) & := & \forall\alpha:\star.(\tau\to\alpha)\to\alpha \\
\mathsf{close}[\tau] & := & \lambda x:\tau,\alpha:\star,f:\tau\to\alpha.f\,x \\
\mathsf{open}[\tau] & := & \lambda c:(\!|\tau|\!).c\,\tau\,(\lambda x:\tau.x) \\
\mathsf{rec}[\tau] & := & (\mathsf{rec}\ \tau) \qquad \text{(voir page 27)} \\
\bot & := & \forall X:\mathsf{prop}.X \\
A\wedge B & := & \forall X:\lceil\star\rceil.(A\to B\to X)\to X \\
M=_\tau N & := & \forall X:\tau\to\lceil\star\rceil.X\,M\to X\,N \\
M\in\mathbb{N} & := & \forall X:\mathsf{nat}\to\lceil\star\rceil.(\forall y:\mathsf{nat}.X\,y\to X\,(\mathsf{S}\,y))\to X\,\overline{0}\to X\,M \\
\exists x:\tau.P & := & \forall X:\lceil\star\rceil.(\forall x:\tau.P\to X)\to X
\end{array}
$$

$$
\begin{array}{rcl}
\mathsf{elim}_\bot[P,\pi] & := & \pi\,P \\
\mathsf{intro}_\wedge[P,Q,\pi_1,\pi_2] & := & \lambda X:\lceil\star\rceil,f:P\to Q\to X.f\,\pi_1\,\pi_2 \\
\mathsf{elim\text{-}gauche}_\wedge[P,Q,\pi] & := & \pi\,P\,(\lambda x:P,y:Q.x) \\
\mathsf{elim\text{-}droite}_\wedge[P,Q,\pi] & := & \pi\,Q\,(\lambda x:P,y:Q.y) \\
\mathsf{intro}_\exists[\tau,x,P,M,\pi] & := & \lambda X:\lceil\star\rceil,f:(\forall x:\tau,P\to X).f\,M\,\pi \\
\mathsf{elim}_\exists[\tau,x,P,h,Q,\pi,\varrho] & := & \pi\,Q\,(\lambda x:\tau,h:P.\varrho) \\
\mathsf{intro\text{-}zero}_\mathbb{N} & := & \lambda X:\mathsf{nat}\to\lceil\star\rceil,f:(\forall y:\mathsf{nat}.X\,y\to X\,(\mathsf{S}\,y)),x:X\,\overline{0}.x \\
\mathsf{intro\text{-}succ}_\mathbb{N}[M,\pi] & := & \lambda X:\mathsf{nat}\to\lceil\star\rceil,f:(\forall y:\mathsf{nat}.X\,y\to X\,(\mathsf{S}\,y)),x:X\,\overline{0}.f\,x\,(\pi\,X\,f\,x) \\
\mathsf{induction}_\mathbb{N}[x,P,\pi,N,\pi_1,\pi_2] & := & \pi\,(\lambda x:\mathsf{nat}.P)\,(\lambda x:\mathsf{nat},h:P.\pi_1)\,\pi_2 \\
\mathsf{intro}_=[\tau] & := & \lambda x:\tau,X:\tau\to\lceil\star\rceil,h:X\,x.h \\
\mathsf{elim}_=[\tau,x,P,\pi,\varrho] & := & \pi\,(\lambda x:\tau.P)\,\varrho \\
\mathsf{open\text{-}close}[\tau] & := & \lambda x:\tau,X:\tau\to\mathsf{prop},h:X\,x.h \\
\mathsf{pair\text{-}proj}^1[\sigma,\tau] & := & \lambda x:\sigma,y:\tau,X:\sigma\to\mathsf{prop},h:X\,x.h \\
\mathsf{pair\text{-}proj}^2[\sigma,\tau] & := & \lambda x:\sigma,y:\tau,X:\tau\to\mathsf{prop},h:X\,y.h \\
\mathsf{rec\text{-}succ}[\tau] & := & \text{(voir lemme 3.2.11)} \\
\mathsf{rec\text{-}zero}[\tau] & := & \text{(voir lemme 3.2.11)}
\end{array}
$$

$\}$

FIGURE 3.3 – Système de notations pour l'arithmétique dans $\mathcal{F}^2$

On trouvera sur la figure 3.3 la description d'un système de notations pour l'arithmétique dans $\mathcal{F}^2$. Ces notations utilisent essentiellement les encodages imprédicatifs pour représenter les différents connecteurs.

La notation $M \in \mathbb{N}$ signifie que M vérifie le plus petit prédicat satisfait par $\overline{0}$ et stable par successeur. Cette notation est encodée par :

$$\forall X : \mathsf{nat} \to \lceil\star\rceil.(\forall y : \mathsf{nat}.X\, y \to X\,(\mathsf{S}\, y)) \to X\,\overline{0} \to X\, M$$

On montre facilement que tous ces termes satisfont bien les règles des figures 3.1 et 3.2, à l'exception des termes $\mathsf{rec\text{-}succ}[]$ et $\mathsf{rec\text{-}zero}[]$ que nous ne donnons pas explicitement en raison de leur taille importante (voir lemme 3.2.11).

Le lemme suivant nous permet d'affirmer que $\mathsf{rec\text{-}succ}[\cdot]$ et $\mathsf{rec\text{-}zero}[\cdot]$ sont dérivables :

3.2.11 Lemme
Il existe deux preuves π_1 et π_2 telles que les deux séquents ci-dessous sont dérivables dans F^2.

1. $\vdash_{\mathcal{F}^2} \pi_1 : \forall\alpha : \star, f : \alpha \to \mathsf{nat} \to \alpha, x : \alpha.\mathsf{rec}[\alpha]\, f\, x\, \overline{0} =_\alpha x$
2. $\vdash_{\mathcal{F}^2} \pi_2 : \forall\alpha : \star, f : \alpha \to \mathsf{nat} \to \alpha, x : \alpha, n : \mathsf{nat}.n \in \mathbb{N} \to \mathsf{rec}[\alpha]\, f\, x\, (\mathsf{S}\, n) =_\alpha f\, (\mathsf{rec}[\alpha]\, f\, x\, n)\, n$

Démonstration On notera $\mathbf{rec}'$ le terme ci-dessous :

$$\begin{array}{rcl}\mathbf{rec}' & : & \forall\alpha.(\alpha \to \mathsf{nat} \to \alpha) \to \alpha \to \mathsf{nat} \to \alpha \times \mathsf{nat}. \\ & & \quad \lambda\alpha : \star, f : \alpha \to \mathsf{nat} \to \alpha, x : \alpha. \\ & & \qquad n \quad (\alpha \times \mathsf{nat}) \\ & & \qquad\quad (\lambda c : \alpha \times \mathsf{nat}.\langle f\, (c)^1\, (c)^2, \mathsf{succ}\, (c)^2\rangle) \\ & & \qquad\quad \langle x, \overline{0}\rangle\end{array}$$

où l'on rappelle que $\langle M, N\rangle$ est une notation pour $(\mathsf{pair}[\alpha, \mathsf{nat}]\, M\, N)$ et $(M)^1$ et $(M)^2$ deux notations pour respectivement $(\mathsf{proj}^1[\alpha, \mathsf{nat}]\, M)$ et $(\mathsf{proj}^2[\alpha, \mathsf{nat}]\, M)$. On remarque que $\mathsf{rec}[\alpha] \equiv \lambda f : \alpha \to \mathsf{nat} \to \alpha, x : \alpha.(\mathbf{rec}'\, \alpha\, f\, x)^1$. Pour montrer les points 1 et 2 il suffit de dériver les deux séquents suivants :

$$\vdash \forall\alpha : \star, f : \alpha \to \mathsf{nat} \to \alpha, x : \alpha.\ (\mathbf{rec}'\, \alpha\, f\, x\, \overline{0}) =_{\alpha\times\mathsf{nat}} \langle x, \overline{0}\rangle \qquad (1)$$

$$\vdash \forall\alpha : \star, f : \alpha \to \mathsf{nat} \to \alpha, x : \alpha, n : \mathsf{nat}.n \in \mathbb{N} \to\ (\mathbf{rec}'\, \alpha\, f\, x\, (\mathsf{S}\, n)) =_{\alpha,\mathsf{nat}} \langle f\, (\mathbf{rec}'\, \alpha\, f\, x\, n)^1\, n, \mathsf{S}\, n\rangle \qquad (2)$$

Le séquent (1) se montre facilement en remarquant que :

$$\mathbf{rec}'\, \alpha\, f\, x\, \overline{0} \equiv \langle x, \overline{0}\rangle$$

et par la réflexivité de $\cdot =_{\alpha,\mathsf{nat}} \cdot$ permet de conclure.

Pour dériver le séquent (2) on procède par induction sur n grâce à l'hypothèse $n \in \mathbb{N}$. Dans le cas de base il faut prouver :

$$\alpha : \star, f : \alpha \to \mathsf{nat} \to \alpha, x : \alpha \vdash (\mathbf{rec}'\, \alpha\, f\, x\, (\mathsf{S}\, \overline{0})) =_{\alpha,\mathsf{nat}} \langle f\, (\mathbf{rec}'\, \alpha\, f\, x\, \overline{0})^1\, \overline{0}, \mathsf{S}\, \overline{0}\rangle$$

or on a :

$$\mathbf{rec}'\, \alpha\, f\, x\, (\mathsf{S}\, \overline{0})) \equiv \langle f\, (\mathbf{rec}'\alpha\, f\, x\, \overline{0})^1\, \overline{0}, \mathsf{S}\, \overline{0}\rangle$$

et on peut conclure grâce à la réflexivité de $\cdot =_{\alpha,\mathsf{nat}} \cdot$. Et pour l'hérédité il faut prouver :

$$\alpha : \star, f : \alpha \to \mathsf{nat} \to \alpha, x : \alpha, n : \mathsf{nat}, h : (\mathbf{rec}'\,\alpha\,f\,x\,(\mathsf{S}\,n)) =_{\alpha,\mathsf{nat}} \langle f\,(\mathbf{rec}'\alpha\,f\,x\,n)^1\,n, \mathsf{S}\,n\rangle$$
$$\vdash (\mathbf{rec}'\,\alpha\,f\,x\,(\mathsf{S}\,(\mathsf{S}\,n))) =_{\alpha,\mathsf{nat}} \langle f\,(\mathbf{rec}'\alpha\,f\,x\,(\mathsf{S}\,n))^1\,(\mathsf{S}\,n), \mathsf{S}\,(\mathsf{S}\,n)\rangle$$

or on a :

$$\mathbf{rec}'\,\alpha\,f\,x\,(\mathsf{S}\,(\mathsf{S}\,n)) \equiv \langle f\,(\mathbf{rec}'\,\alpha\,f\,x\,(\mathsf{S}\,n))^1\,(\mathsf{S}\,n), \mathsf{S}\,(\mathbf{rec}'\,\alpha\,f\,x\,(\mathsf{S}\,n))^2\rangle$$

Or, grâce à l'hypothèse de récurrence h on montre que :

$$\alpha : \star, f : \alpha \to \mathsf{nat} \to \alpha, x : \alpha, n : \mathsf{nat}, h : (\mathbf{rec}'\,\alpha\,f\,x\,(\mathsf{S}\,n)) =_{\alpha,\mathsf{nat}} \langle f\,(\mathbf{rec}'\,\alpha\,f\,x\,n)^1\,n, \mathsf{S}\,n\rangle$$
$$\vdash (\mathbf{rec}'\,\alpha\,f\,x\,(\mathsf{S}\,n))^2 =_{\mathsf{nat}} (\mathsf{S}\,n)$$

On se ramène donc à démontrer

$$\alpha : \star, f : \alpha \to \mathsf{nat} \to \alpha, x : \alpha, n : \mathsf{nat}, h : (\mathbf{rec}'\,\alpha\,f\,x\,(\mathsf{S}\,n)) =_{\alpha,\mathsf{nat}} \langle f\,(\mathsf{rec}[\alpha]\,f\,x\,n)^1\,n, \mathsf{S}\,n\rangle$$
$$\vdash \langle f\,(\mathsf{rec}[\alpha]\,f\,x\,(\mathsf{S}\,n))^1\,(\mathsf{S}\,n), \mathsf{S}\,(\mathsf{S}\,n)\rangle \doteq \langle f\,(\mathsf{rec}[\alpha]\,f\,x\,(\mathsf{S}\,n))^1\,(\mathsf{S}\,n), \mathsf{S}\,(\mathsf{S}\,n)\rangle$$

et on conclut par réflexivité. ☺

On en déduit donc bien que :

3.2.12 Lemme
$\mathcal{F} \models$ NOTATION.

3.3 Petites annotations dans les logiques engendrées

Dans cette section, nous allons voir qu'il est possible d'annoter les variables des termes de $\mathcal{P}^2$ dans le but de reconnaître facilement le niveau des sous-termes ; ce qui s'avérera essentiel pour définir la fonction de projection et les transformations de réalisabilité sans que ces fonctions dépendent du contexte dans lequel leur argument est bien typé. Ces annotations viendront nécessairement avec de nouvelles relations de réduction (que l'on notera $\twoheadrightarrow^+$, $\triangleright^+$ et $\equiv^+$)) qui préserveront les annotations ainsi qu'une nouvelle relation de dérivabilité, notée $\vdash^+$, qui s'assurera que les termes sont bien annotés en plus d'être bien typés. On conclura cette section avec la preuve que tout séquent bien typé $\Gamma \vdash M : A$ pourra être annoté de façon unique en un séquent $\Gamma^+ \vdash^+ M^+ : A^+$.

Soit $\mathcal{P}$ un CTS, nous allons voir qu'il est possible d'annoter les termes de $\mathcal{P}^2$ avec des informations sur le *niveau des variables*, c'est-à-dire avec l'entier 1 ou 2 selon le type de la variable qui sera dans le premier cas de sorte s et dans le second de sorte $\lceil s \rceil$ (avec $s \in \mathcal{S}_{\mathcal{P}}$). La grammaire des termes de $\mathcal{P}^2$ devient alors :

$$\begin{array}{rcl} A, B, \ldots & := & (A\ B) \\ & | & \lambda x^k : A.B \\ & | & \forall x^k : A.B \\ & | & x^k \\ & | & s \end{array}$$

avec $k \in \{1,2\}$ et $s \in \mathcal{S}_{\mathcal{P}^2}$. On notera également les variables dans les contexte :

$$\Gamma := \langle\rangle \quad | \quad \Gamma, x^k : A$$

On définit le niveau $\mathrm{lvl}(s)$ d'une sorte s par :

$$\begin{aligned} \mathrm{lvl}(s) &= 1 \text{ si } s \in \mathcal{S}_{\mathcal{P}} \\ \mathrm{lvl}(s) &= 2 \text{ si } s \in \mathcal{S}_{\mathcal{P}^2} \setminus \mathcal{S}_{\mathcal{P}} \end{aligned}$$

On peut étendre la notation $\mathrm{lvl}(\cdot)$ à tous les termes de la façon suivante :

$$\begin{aligned} \mathrm{lvl}(x^k) &= k \\ \mathrm{lvl}((M\,N)) &= \mathrm{lvl}(M) \\ \mathrm{lvl}(\lambda x^k : A.M) &= \mathrm{lvl}(M) \\ \mathrm{lvl}(\forall x^k : A.B) &= \mathrm{lvl}(B) \end{aligned}$$

Les relations $\mathcal{A}$ et $\mathcal{C}$ préservent le niveau des sortes :

3.3.1 Lemme
Si $(s,r) \in \mathcal{A}_{\mathcal{P}^2}$ *ou* $(s,r) \in \mathcal{C}_{\mathcal{P}^2}$, *alors* $\mathrm{lvl}(s) = \mathrm{lvl}(r)$.

Démonstration La preuve est immédiate d'après la définition de $\mathcal{P}^2$. ☺

On prouve alors :

3.3.2 Lemme

$$\mathrm{lvl}(M[N/x^{\mathrm{lvl}(N)}]) = \mathrm{lvl}(M)$$

Démonstration Par induction sur la structure de M :
- Si $M = s$ ou $M = y^k$ avec $y^k \neq x^{\mathrm{lvl}(N)}$, on a $M[N/x^{\mathrm{lvl}(N)}] = M$ et donc c'est immédiat.
- Si $M = x^{\mathrm{lvl}(N)}$, on a $\mathrm{lvl}(M) = \mathrm{lvl}(N) = \mathrm{lvl}(M[N/x^{\mathrm{lvl}(N)}])$.
- Si $M = (M_1\,M_2)$, on a $\mathrm{lvl}(M[N/x^{\mathrm{lvl}(N)}]) = \mathrm{lvl}(M_1[N/x^{\mathrm{lvl}(N)}])$. Or par hypothèse d'induction, on a $\mathrm{lvl}(M_1[N/x^{\mathrm{lvl}(N)}]) = \mathrm{lvl}(M_1) = \mathrm{lvl}(M)$.
- Si $M = \lambda x : M_1.M_2$ ou $M = \forall x : M_1.M_2$, on a $\mathrm{lvl}(M[N/x^{\mathrm{lvl}(N)}]) = \mathrm{lvl}(M_2[N/x^{\mathrm{lvl}(N)}])$. Or par hypothèse d'induction, on a $\mathrm{lvl}(M_2[N/x^{\mathrm{lvl}(N)}]) = \mathrm{lvl}(M_2) = \mathrm{lvl}(M)$. ☺

Nous allons maintenant définir la relation de réduction $\twoheadrightarrow^+$ en une étape entre les termes annotés. Cette relation est définie de façon analogue à la relation $\twoheadrightarrow$ à ceci près qu'elle s'assure que les annotations sont préservées. Elle est donc générée par les règles suivantes :

$$\frac{}{(\lambda x^{\mathrm{lvl}(C)} : A.B)\,C \twoheadrightarrow^+ B[C/x^{\mathrm{lvl}(C)}]}$$

$$\frac{M \twoheadrightarrow^+ M'}{(M\,N) \twoheadrightarrow^+ (M'\,N)} \qquad \frac{N \twoheadrightarrow^+ N'}{(M\,N) \twoheadrightarrow^+ (M\,N')}$$

$$\frac{A \twoheadrightarrow^+ A'}{\lambda x^k : A.M \twoheadrightarrow^+ \lambda x^k : A'.M} \qquad \frac{M \twoheadrightarrow^+ M'}{\lambda x^k : A.M \twoheadrightarrow^+ \lambda x^k : A.M'}$$

$$\frac{A \twoheadrightarrow^+ A'}{\forall x^k : A.B \twoheadrightarrow^+ \forall x^k : A'.B} \qquad \frac{B \twoheadrightarrow^+ B'}{\forall x^k : A.B \twoheadrightarrow^+ \forall x^k : A.B'}$$

On notera $\triangleright^+$ la clôture transitive de $\twoheadrightarrow^+$, $\trianglerighteq^+$ la clôture réflexive de $\triangleright^+$ et $\equiv^+$ la clôture symétrique et transitive de $\twoheadrightarrow^+$.

Le fait que ces réductions ne réduisent pas les rédexs qui ne respectent pas les niveaux nous permet de montrer le lemme suivant :

3.3.3 Lemme
Si $M \equiv^+ N$, *alors* $\mathrm{lvl}(M) = \mathrm{lvl}(N)$.

Démonstration Il suffit de montrer que si $M \twoheadrightarrow^+ N$, alors $\mathrm{lvl}(M) = \mathrm{lvl}(N)$. On procède par induction sur les règles définissant $M \twoheadrightarrow^+ N$:

- Si M est de la forme $((\lambda x^{\mathrm{lvl}(M_2)} : A.M_1)\, M_2)$ et $N = M_1[M_2/x^{\mathrm{lvl}(M_2)}]$. On a $\mathrm{lvl}(M) = \mathrm{lvl}(M_1)$ et d'après le lemme 3.3.2, on a $\mathrm{lvl}(N) = \mathrm{lvl}(M_1)$.
- Si M est de la forme $(M_1\, M_2)$ et N de la forme $(N_1\, M_2)$ avec $M_1 \twoheadrightarrow^+ N_1$, alors par hypothèse d'induction on a $\mathrm{lvl}(M) = \mathrm{lvl}(M_1) = \mathrm{lvl}(N_1) = \mathrm{lvl}(N)$.
- Si M est de la forme $(M_1\, M_2)$ et N de la forme $(M_1\, N_2)$ avec $M_2 \twoheadrightarrow^+ N_2$, on a $\mathrm{lvl}(M) = \mathrm{lvl}(M_1) = \mathrm{lvl}(N)$.
- Si M est de la forme $\lambda x : M_1.M_2$ (resp. $\forall x : M_1.M_2$) et N de la forme $\lambda x : N_1.M_2$ (resp. $\forall x : N_1.M_2$). avec $M_1 \twoheadrightarrow^+ N_1$, on a $\mathrm{lvl}(M) = \mathrm{lvl}(M_2) = \mathrm{lvl}(N)$.
- Si M est de la forme $\lambda x : M_1.M_2$ (resp. $\forall x : M_1.M_2$) et N de la forme $\lambda x : M_1.N_2$ (resp. $\forall x : M_1.N_2$). avec $M_2 \twoheadrightarrow^+ N_2$, on a par hypothèse d'induction $\mathrm{lvl}(M) = \mathrm{lvl}(M_2) = \mathrm{lvl}(N_2) = \mathrm{lvl}(N)$.☺

3.3.4 Lemme
Soit $k = \mathrm{lvl}(N)$.

1. *Si* $M \twoheadrightarrow^+ M'$, *alors* $M[N/x^k] \twoheadrightarrow^+ M'[N/x^k]$.
2. *Si* $M \triangleright^+ M'$, *alors* $M[N/x^k] \triangleright^+ M'[N/x^k]$.
3. *Si* $M \equiv^+ M'$, *alors* $M[N/x^k] \equiv^+ M'[N/x^k]$.

Démonstration Les points 2 et 3 se déduisent immédiatement du premier point. On prouve celui-ci par induction sur les règles qui définissent $M \twoheadrightarrow^+ M'$. On ne traite que le cas clef : si M est de la forme $(\lambda y^{k'} : A.M_1)\, M_2$ et $M' = M_1[M_2/y^{k'}]$ avec $y^{k'} \neq x^k$ et $y^{k'} \notin \mathcal{FV}(N)$, on a alors :

$$M[N/x^k] = (\lambda y^{k'} : A[N/x^k].M_1[N/x^k])\, M_2[N/x^k] \twoheadrightarrow M_1[N/x^k, M_2[N/x^k]/y^{k'}]$$

Or comme $y^{k'} \notin \mathcal{FV}(N)$, on a bien $M_1[N/x^k, M_2[N/x^k]/y^{k'}] = (M_1[M_2/y^{k'}])[N/x^k]$. ☺

De la même façon les différentes relations de cumulativité sont ensuite étendues aux termes avec annotations :

$$\begin{aligned} A \prec_0^+ B &\Leftrightarrow A \equiv^+ s \wedge B \equiv^+ s' \wedge (s, s') \in \mathcal{C}_{\mathcal{P}^2} \\ A \prec_{n+1}^+ B &\Leftrightarrow A \prec_n^+ B \vee \left(A \equiv^+ \forall x^k : C.A' \wedge B \equiv^+ \forall x^k : C.B' \wedge A' \prec_n^+ B'\right) \\ A \prec^+ B &\Leftrightarrow \exists n \in \mathbb{N}, A \prec_n^+ B \\ A \preccurlyeq^+ B &\Leftrightarrow A \equiv^+ B \vee A \prec^+ B \end{aligned}$$

On notera que la cumulativité impose que les annotations entre produits comparables soient les mêmes :

3.3.5 Lemme
Si $\forall x^k : A.B \preccurlyeq^+ \forall x^{k'} : A'.B'$*, alors* $k = k'$*.*

Tout comme la version sans annotation, la cumulativité est stable par substitution :

3.3.6 Lemme
Si $A \preccurlyeq^+ B$*, alors* $A[M/x^k] \preccurlyeq^+ B[M/x^k]$*.*

Démonstration La preuve est similaire à celle du lemme 1.1.6. ☺

Et tout comme la conversion, la cumulativité préserve les niveaux :

3.3.7 Lemme
Si $M \preccurlyeq^+ N$*, alors* $\mathrm{lvl}(M) = \mathrm{lvl}(N)$*.*

Démonstration D'après le lemme 3.3.3, il suffit de montrer que :

$$\text{Si } M \prec_n N, \text{ alors } \mathrm{lvl}(M) = \mathrm{lvl}(N).$$

On procède par induction sur n :
- $M \prec_0^+ N$ signifie qu'il existe $s_m \equiv^+ M$ et $s_n \equiv^+ N$ tels que $(s_m, s_n) \in \mathcal{C}_{\mathcal{P}^2}$. Or par construction de $\mathcal{P}^2$, on a $\mathrm{lvl}(s_m) = \mathrm{lvl}(s_n)$. Et d'après le lemme 3.3.3, on en déduit $\mathrm{lvl}(M) = \mathrm{lvl}(N)$.
- Si $M \prec_{n+1} N$, par définition deux cas sont possibles :
  - Soit $M \prec_n^+ N$, et on conclut par hypothèse de récurrence.
  - Soit il existe M', N' et C tels que $M \equiv^+ \forall x^k : C.M'$, $N \equiv^+ \forall x^k : C.N'$ et $M' \prec_n^+ N'$. Or par hypothèse de récurrence, on a $\mathrm{lvl}(M') = \mathrm{lvl}(N')$. Et d'après le lemme 3.3.3, on a $\mathrm{lvl}(M) = \mathrm{lvl}(\forall x^k : C.M') = \mathrm{lvl}(M') = \mathrm{lvl}(N') = \mathrm{lvl}(\forall x^k : C.N') = \mathrm{lvl}(N)$. ☺

La figure 3.4 montre comment on adapte les règles de typage afin de s'assurer que les termes sont bien annotés.

Le lemme suivant montre que les annotations des variables sont bien positionnée avec le niveau de leur type.

3.3.8 Lemme
Si $\Gamma, x^k : A, \Delta \vdash^+ M : T$*, alors* $k = \mathrm{lvl}(A)$ *et il existe* s *tel que* $\Gamma \vdash^+ A : s$*.*

$$\frac{}{\vdash^+ s_1 : s_2}\ (s_1, s_2) \in \mathcal{A}_{\mathcal{P}^2}$$
AXIOME

$$\frac{\Gamma \vdash^+ A : s}{\Gamma, x^k : A \vdash^+ x^k : A}\ x^k \notin \Gamma \qquad k = \mathrm{lvl}(C)\ \frac{\Gamma \vdash^+ A : B \qquad \Gamma \vdash^+ C : s}{\Gamma, x^k : C \vdash^+ A : B}\ x \notin \Gamma$$
VARIABLE — AFFAIBLISSEMENT

$$k = \mathrm{lvl}(A)\ \frac{\Gamma \vdash^+ A : s_1 \qquad \Gamma, x^k : A \vdash^+ B : s_2}{\Gamma \vdash^+ \forall x^k : A.B : s_3}\ (s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}^2}$$
PRODUIT

$$\frac{\Gamma, x^k : A \vdash^+ B : C \qquad \Gamma \vdash^+ \forall x^k : A.C : s}{\Gamma \vdash^+ \lambda x^k : A.B \quad : \quad \forall x^k : A.C} \qquad \frac{\Gamma \vdash^+ M : \forall x^k : A.B \qquad \Gamma \vdash^+ N : A}{\Gamma \vdash^+ (M\ N) : B[N/x^k]}$$
ABSTRACTION — APPLICATION

$$\frac{\Gamma \vdash^+ A : B \qquad \Gamma \vdash^+ B' : s \qquad B \preccurlyeq^+ B'}{\Gamma \vdash^+ A : B'} \qquad \frac{\Gamma \vdash^+ A : C \qquad C \preccurlyeq^+ s}{\Gamma \vdash^+ A : s}$$
CUMULATIVITÉ — CUMULATIVITÉ-SORTES

FIGURE 3.4 – Règles de typage munies de petites annotations pour les logiques engendrées.

Démonstration On procède par induction sur la dérivation de $\Gamma, x^k : A, \Delta \vdash^+ M : T$. Le cas Axiome est impossible car le contexte n'est pas vide. Les cas Produit, Abstraction, Application, Cumulativité et Cumulativité-Sortes se traitent simplement en utilisant l'hypothèse d'induction relative à l'une des prémisses qui ne modifient pas le contexte (il y en a toujours une). Il nous reste à traiter les cas Variable et Affaiblissement. Pour chacune de ces deux règles, on distingue deux cas :

- Soit $\Delta = \langle\rangle$ et on d'après la condition de bord que $k = \mathrm{lvl}(A)$ et l'une des prémisses nous donne $\Gamma \vdash^+ A : s$.
- Soit $\Delta = \Delta', y^{k'} : B$ avec $\Gamma, x^k : A, \Delta' \vdash^+ B : s$ et on peut conclure par hypothèse d'induction. ☺

On définit ensuite l'oubli $|\cdot|$ qui envoie les termes annotés vers les termes tout en préservant leur structure :

$$\begin{aligned} |(A\ B)| &= (|A|\ |B|) \\ |\lambda x^k : A.B| &= \lambda x : |A|.|B| \\ |\forall x^k : A.B| &= \forall x : |A|.|B| \\ |x^k| &= x \\ |s| &= s \end{aligned}$$

$$\begin{aligned} |\langle\rangle| &= \langle\rangle \\ |\Gamma, x :^k A| &= |\Gamma|, x : |A| \end{aligned}$$

On montre alors facilement le lemme ci-dessous :

3.3.9 Lemme
Si $A \preccurlyeq^+ B$, *alors* $|A| \preccurlyeq |B|$.

Et par une induction immédiate on montre :

3.3.10 Lemme
Si $\Gamma \vdash^+ M : A$, *alors* $|\Gamma| \vdash |M| : |A|$

Nous cherchons maintenant à prouver la réciproque (voir le lemme 3.3.23) qui nous montrera que l'on peut toujours annoter les termes bien typés. Pour cela nous allons avoir besoin de développer la métathéorie du système $\vdash^+$ avec annotation. Nous démontrerons donc le minimum de propriété métathéorique nécessaire pour prouver l'équivalence de $\vdash$ et $\vdash^+$ car une fois l'équivalence les autres propriétés en découleront. On remarquera que hormis les vérifications directement liées aux annotations les démonstrations seront similaires à leurs versions sans annotations.

Le lemme de substitution reste vrai en présence d'annotations.

3.3.11 Lemme (Substitution avec annotations)
Si $\Gamma, x^k : A, \Delta \vdash^+ B : C$ *et* $\Gamma \vdash^+ D : A$, *alors* $\Gamma, \Delta[D/x^k] \vdash^+ B[D/x^k] : C[D/x^k]$.

Démonstration Tout comme pour le lemme 1.1.23, il suffit de procéder par une induction routinière sur la structure de la dérivation de $\Gamma, x^k : A, \Delta \vdash^+ B : C$. Les cas Cumulativité et Cumulativité-Sortes sont traités en appliquant l'hypothèse d'induction et le lemme 3.3.6. Ici, on traite donc simplement le cas Variable. Pour cela on distingue deux cas :

– Soit $\Delta = \langle\rangle$. Dans ce cas, $B = x^k$ et $C = A$ et on a donc $\Gamma, \Delta[D/x^k] = \Gamma$, $B[D/x^k] = D$ et $C[D/x^k] = A[D/x^k] = A$ (car $x^k \notin A$). On doit donc montrer que $\Gamma \vdash^+ D : A$ que l'on a dans nos hypothèses.
– Soit $\Delta = \Delta', y^{k'} : C$ avec $B = y^{k'}$, $y^{k'} \neq x^k$ et $\Gamma, x^k : A, \Delta' \vdash^+ C : s$. Alors par hypothèse d'induction, on a $\Gamma, \Delta'[D/x^k] \vdash^+ C[D/x^k] : s$, et donc la règle VARIABLE montre que $\Gamma, \Delta'[D/x^k], y^{k'} : C[D/x^k] \vdash^+ y^{k'} : C[D/x^k]$. On conclut donc en remarquant que $y^{k'} = B[D/x^k]$. ☺

Le lemme suivant nous sera utile pour démontrer le lemme d'inversion.

3.3.12 Lemme (Affaiblissement généralisé dans les systèmes avec petites annotations)
La règle suivante est admissible :

$$k = \mathrm{lvl}(B)\ \frac{\Gamma, \Delta \vdash^+ M : A \qquad \Gamma \vdash^+ B : s}{\Gamma, x^k : B, \Delta \vdash^+ M : A}\ x \notin \Gamma, \Delta$$

Démonstration La preuve s'obtient par la même induction routinière que la preuve du lemme 1.1.24 sur la structure de la dérivation de $\Gamma, \Delta \vdash^+ M : A$ en supposant $\Gamma \vdash^+ B : s$ et $x \notin \Gamma, \Delta$. ☺

On peut alors prouver le lemme ci-dessous :

3.3.13 Lemme (Inversion de $\vdash^+$)

1. $\Gamma \vdash^+ s : A \Rightarrow \exists r \preccurlyeq^+ A, (s, r) \in \mathcal{A}$,
2. $\Gamma \vdash^+ x^k : A \Rightarrow \exists B \preccurlyeq^+ A, (x^k : B) \in \Gamma$,
3. $\Gamma \vdash^+ \lambda x^k : A.M : C \Rightarrow \exists (B, s), \begin{cases} \Gamma \vdash^+ \forall x^k : A.B : s \\ \Gamma, x^k : A \vdash^+ M : B \\ \forall x^k : A.B \preccurlyeq^+ C \end{cases}$,
4. $\Gamma \vdash^+ \forall x^k : A.B : C \Rightarrow \exists (s_1, s_2, s_3) \in \mathcal{R}, \begin{cases} \Gamma \vdash^+ A : s_1 \\ \Gamma, x^k : A \vdash^+ B : s_2 \\ s_3 \preccurlyeq^+ C \end{cases}$,
5. $\Gamma \vdash^+ (MN) : C \Rightarrow \exists (A, k, B), \begin{cases} \Gamma \vdash^+ M : \forall x^k : A.B \\ \Gamma \vdash^+ N : A \\ B[N/x^k] \preccurlyeq^+ C \end{cases}$.

Démonstration La preuve est similaire à celle du lemme 1.1.32. ☺

3.3.14 Lemme
Si $\Gamma \vdash^+ M : \forall x^k : A.B$, alors il existe $(s_1, s_2, s_3) \in \mathcal{R}$ telles que $\Gamma \vdash^+ A : s_1$ et $\Gamma, x^k : A \vdash^+ B : s_2$ (et donc $k = \mathrm{lvl}(A)$)

Démonstration La preuve est similaire à celle du lemme 1.1.34, le fait que $k = \mathrm{lvl}(A)$ est justifié par le lemme 3.3.8. ☺

3.3.15 Lemme
Si $\Gamma \vdash^+ M : A$, alors il existe s telle que $A = s$ ou $\Gamma \vdash^+ A : s$.

Démonstration La preuve est analogue au lemme 1.1.35. Dans les cas Axiome, Produit et Cumulativité-Sortes, on a bien A qui est une sorte. Dans les cas Abstraction, Variable et Cumulativité, une prémisse donne directement $\Gamma \vdash A : s$. Dans le cas Affaiblissement, il suffit d'appliquer l'hypothèse d'induction. Il nous faut encore traiter le cas de Application : Ici, M est de la forme $M_1\, M_2$ avec $\Gamma \vdash^+ M_1 : \forall x^k : A_1.A_2$ et $\Gamma \vdash^+ M_2 : A_1$ et $A = A_2[M_2/x^k]$. D'après le lemme précédent, il existe une sorte s telle que $\Gamma, x^k : A_1 \vdash^+ A_2 : s$ et par substitution, on obtient bien que $\Gamma \vdash^+ A_2[M_2/x^k] : s$. ☺

On peut alors prouver le lemme suivant :

3.3.16 Lemme
Si $\Gamma \vdash^+ M : T$, *alors* $\mathrm{lvl}(M) = \mathrm{lvl}(T)$.

Démonstration On procède par induction sur la dérivation de $\Gamma \vdash^+ M : T$:

- Axiome : Dans ce cas, $M = s$ et $T = r$ avec $(s, r) \in \mathcal{A}_{\mathcal{P}^2}$ et donc par construction de $\mathcal{P}^2$, on a $\mathrm{lvl}(s) = \mathrm{lvl}(r)$.
- Variable : On a $M = x^k$, $\Gamma = \Gamma', x^k : T$ et $\Gamma' \vdash^+ T : s$ avec $s \in \mathcal{S}_{\mathcal{P}^2}$ et $k = \mathrm{lvl}(s)$. Par hypothèse d'induction, on a $\mathrm{lvl}(T) = \mathrm{lvl}(s) = k = \mathrm{lvl}(M)$.
- Affaiblissement : On a Γ est de la forme $\Gamma', x^k : A$ et $\Gamma' \vdash^+ M : T$ et par hypothèse d'induction on obtient directement $\mathrm{lvl}(M) = \mathrm{lvl}(T)$.
- Abstraction : On a $M = \lambda x : A.N$ et $T = \forall x : A.B$ avec $\Gamma, x : A \vdash^+ N : B$. Donc par hypothèse d'induction, on obtient que $\mathrm{lvl}(N) = \mathrm{lvl}(B)$. Or on a bien $\mathrm{lvl}(M) = \mathrm{lvl}(N) = \mathrm{lvl}(B) = \mathrm{lvl}(T)$.
- Application : On a $M = (M_1\, M_2)$ et $T = B[M_2/x^k]$ avec $\Gamma \vdash^+ M_1 : \forall x^k : A.B$ et $\Gamma \vdash^+ M_2 : A$. Et d'après le lemme précédent, on en déduit que $k = \mathrm{lvl}(A)$, puis par hypothèse d'induction, on a $\mathrm{lvl}(M_1) = \mathrm{lvl}(\forall x^k : A.B) = \mathrm{lvl}(B)$ et $\mathrm{lvl}(M_2) = \mathrm{lvl}(A)$. On peut donc conclure d'après le lemme 3.3.2 que $\mathrm{lvl}(B[M_2/x^k]) = \mathrm{lvl}(B)$.
- Produit : On a $M = \forall x^k : A.B$ et $T = s_3$ avec $\Gamma \vdash^+ A : s_1$ et $\Gamma, x : A \vdash^+ B : s_2$ et $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}^2}$. Or par hypothèse d'induction on a $\mathrm{lvl}(B) = \mathrm{lvl}(s_2)$. Or par construction de $\mathcal{R}_{\mathcal{P}^2}$, on a $\mathrm{lvl}(s_2) = \mathrm{lvl}(s_3)$. On en conclut donc bien que $\mathrm{lvl}(M) = \mathrm{lvl}(B) = \mathrm{lvl}(s_2) = \mathrm{lvl}(s_3) = \mathrm{lvl}(T)$.
- Cumulativité, Cumulativité-Sortes : Dans ces deux cas on a $T' \preccurlyeq^+ T$ et $\Gamma \vdash^+ M : T'$ et donc par hypothèse d'induction on obtient que $\mathrm{lvl}(M) = \mathrm{lvl}(T')$. Enfin, d'après le lemme 3.3.5, on en conclut que $\mathrm{lvl}(M) = \mathrm{lvl}(T)$. ☺

3.3.17 Lemme (Préservation du typage avec annotations)
Si $\Gamma \vdash^+ M : A$ *et* $M \trianglerighteq^+ M'$, *alors* $\Gamma \vdash^+ M' : A$.

Démonstration La démonstration s'adapte sans problème à la préservation du typage. À titre d'illustration, nous traitons le cas clef. Supposons que la dernière règle appliquée dans la dérivation soit :

$$\dfrac{\overset{(1)}{\Gamma \vdash^+ \lambda x^k : A.M : \forall x^{k'} : A'.B'} \qquad \overset{(2)}{\Gamma \vdash^+ N : A'}}{\underset{(0)}{\Gamma \vdash^+ (\lambda x^k : A.M)\, N : B'[N/x^{k'}]}}$$

avec $k = \text{lvl}(N)$.

On cherche maintenant à montrer $\Gamma \vdash^+ M[N/x^k] : B'[N/x^{k'}]$ (*). Si l'on applique le lemme d'inversion (lemme 3.3.13) sur (1), on obtient qu'il existe B et s tels que $\Gamma, x^k : A \vdash^+ M : B$ (2), $\Gamma \vdash^+ \forall x^k : A.B : s$ (3) et $\forall x^k : A.B \preccurlyeq^+ \forall x^{k'} : A'.B'$ (4).

D'après le lemme 3.3.5, on en déduit que $k = k'$ et donc on déduit de (4) que $A \equiv^+ A'$ et que $B[N/x^k] \preccurlyeq^+ B'[N/x^k]$. Or d'après le lemme d'inversion 3.3.13, (3) implique qu'il existe s_A telle que $\Gamma \vdash^+ A : s_A$ (5) et par cumulativité on en déduit $\Gamma \vdash^+ N : A$ (6). Grâce à (6), (2) et au lemme de substitution (lemme 3.3.11), on a $\Gamma \vdash^+ M[N/x^k] : B[N/x^k]$.

Enfin, par inversion de (3), on obtient que $\Gamma, x^k : A \vdash^+ B : s_B$ pour une sorte sorte $s_B \in \mathcal{S}_{\mathcal{P}^2}$ et par substitution et (6), on a $\Gamma \vdash^+ B[N/x^k] : s_B$ et on en déduit (*) par cumulativité. ☺

Le lemme ci-dessous affirme l'unicité des annotations des termes bien typés, il est essentiel pour démontrer le lemme 3.3.18 puisqu'il permettra de démontrer l'unicité des annotations obtenues à partir des hypothèses induction associées aux différentes prémisses de chaque cas.

3.3.18 Lemme
Si $\Gamma \vdash^+ M : A$, $\Gamma \vdash^+ N : B$ *et* $|M| = |N|$, *alors* $M = N$.

Démonstration Par induction sur la structure de M :

- $M = s$: dans ce cas, on a nécessairement $N = s$, on n'a donc rien à démontrer.
- $M = x^k$: on a nécessairement $N = x^{k'}$. Par inversion de $\Gamma \vdash^+ M : A$ et de $\Gamma \vdash^+ N : B$, on sait qu'il existe C tel que $(x^k : C) \in \Gamma$ tel que $C \preccurlyeq^+ A$ et $C \preccurlyeq^+ B$. D'après le lemme 3.3.16, on obtient que $k = \text{lvl}(M) = \text{lvl}(A)$ et $k' = \text{lvl}(N) = \text{lvl}(B)$, or d'après le lemme 3.3.7, on en déduit que $\text{lvl}(C) = \text{lvl}(A) = \text{lvl}(B) = k = k'$. On a donc bien $M = N$.
- $M = M_1\, M_2$: dans ce cas, N est de la forme $N_1\, N_2$ avec $|M_1| = |N_1|$ et $|M_2| = |N_2|$. Par inversion de $\Gamma \vdash^+ M : A$, on obtient qu'il existe T_1 et T_2 tels que $\Gamma \vdash^+ M_1 : T_1$ et $\Gamma \vdash^+ M_2 : T_2$. De même par inversion de $\Gamma \vdash^+ N : B$, on obtient qu'il existe S_1 et S_2 tels que $\Gamma \vdash^+ N_1 : S_1$ et $\Gamma \vdash^+ M_2 : S_2$. Par hypothèse d'induction, on obtient alors que $M_1 = N_1$ et $M_2 = N_2$ et donc que $M = N$.
- $M = \lambda x^k : M_1.M_2$: dans ce cas, N est de la forme $\lambda x^{k'} : N_1.N_2$ avec $|M_1| = |N_1|$ et $|M_2| = |N_2|$. Par inversion de $\Gamma \vdash^+ M : A$, on obtient qu'il existe C et s tels que $\Gamma, x^k : M_1 \vdash^+ M_2 : C$ (1) et $\Gamma \vdash^+ \forall x^k : M_1.C : s$ (2). Par inversion de (2), on montre qu'il existe s' telle que $\Gamma \vdash^+ M_1 : s'$ (3). De même, par inversion de $\Gamma \vdash^+ N : B$, on obtient qu'il existe D et r tels que $\Gamma, x^{k'} : N_1 \vdash^+ N_2 : D$ (1') et $\Gamma \vdash^+ \forall x^{k'} : N_1.D : r$ (2'), puis par inversion de (2') on montre qu'il existe r' telle que $\Gamma \vdash^+ N_1 : r'$ (3'). D'après le lemme 3.3.8, on a $k' = \text{lvl}(N_1)$ et $k = \text{lvl}(M_1)$. Par hypothèse d'induction avec (3) et (3'), on montre donc bien que $N_1 = M_1$ et donc que $k = k'$. On a donc bien $\Gamma, x^k : M_1 = \Gamma, x^{k'} : N_1$, on peut donc appliquer une nouvelle fois l'hypothèse d'induction sur (1) et (1') pour montrer que $M_1 = N_2$. Et donc $M = N$.
- $M = \forall x^k : M_1.M_2$: dans ce cas, N est de la forme $\forall x^{k'} : N_1.N_2$ avec $|M_1| = |N_1|$ et $|M_2| = |N_2|$. Par inversion de $\Gamma \vdash^+ M : A$, on obtient qu'il existe s_1 et s_2 telles que $\Gamma \vdash^+ M_1 : s_1$ et $\Gamma, x^k : M_1 \vdash^+ M_2 : s_2$ avec $k = \text{lvl}(M_1)$. Et par inversion de $\Gamma \vdash^+ N : B$, on obtient qu'il existe r_1 et r_2 telles que $\Gamma \vdash^+ N_1 : r_1$ et $\Gamma, x^{k'} : N_1 \vdash^+ N_2 : r_2$ avec $k' = \text{lvl}(N_1)$. Par hypothèse d'induction on en déduit que $M_1 = N_1$ et donc que $k = k'$. Et donc $\Gamma, x^k : M_1 = \Gamma, x^{k'} : N_1$, on peut donc appliquer une nouvelle fois l'hypothèse d'induction pour montrer que $M_1 = N_2$. Et donc $M = N$. ☺

Nous avons un résultat analogue pour les annotations des contextes :

3.3.19 Lemme
Si $\Gamma \vdash^+ A : B$ *et* $\Gamma' \vdash^+ A' : B'$ *avec* $|\Gamma| = |\Gamma'|$, *alors* $\Gamma = \Gamma'$.

Démonstration Par récurrence sur la taille de Γ :
- Si $\Gamma = \langle\rangle$, alors comme $|\Gamma| = |\Gamma'|$, on a nécessairement $\Gamma' = \langle\rangle$ et donc $\Gamma = \Gamma'$.
- Si $\Gamma = \Delta, x^k : A$ et $\Gamma' = \Delta', x^{k'} : A'$, alors d'après le lemme 3.3.8, on a $\Delta \vdash A : s$, $\Delta' \vdash A' : s'$, $k = \text{lvl}(A)$, $k' = \text{lvl}(A')$. Ensuite $|\Gamma| = |\Gamma'|$ implique $|A| = |A'|$ et $|\Delta| = |\Delta'|$. Et donc par hypothèse de récurrence, on obtient $\Delta = \Delta'$. Et donc d'après le lemme précédent, on obtient $A = A'$. On en déduit finalement que $k = \text{lvl}(A) = \text{lvl}(A') = k'$ et donc $\Gamma = \Gamma'$. ☺

Le lemme suivant affirme que nous pouvons recouvrer les annotations des termes bien typés après réduction :

3.3.20 Lemme
Si $\Gamma \vdash^+ M : T$ *et* $|M| \rhd N^-$, *alors il existe* N *tel que* $|N| = N^-$ *et* $M \rhd^+ N$.

Démonstration D'après la préservation du typage (lemme 3.3.17), il suffit de démontrer la proposition pour $|M| \twoheadrightarrow N^-$. On procède par induction sur les règles définissant la réduction.
- Si M est de la forme $(\lambda x^k : A.M_1)\, M_2$ et $N^- = |M_1|[|M_2|/x]$:
  Par inversion de $\Gamma \vdash^+ M : T$, on obtient que $\Gamma \vdash^+ \lambda x^k : A.M_1 : \forall x^{k'} : A'.B'$ (1) et $\Gamma \vdash^+ M_2 : A'$ (2) pour deux termes A' et B'. Par inversion de (1), on en déduit qu'il existe B et s tels que $\Gamma, x^k : A \vdash^+ M_1 : B$, $\Gamma \vdash^+ \forall x^k : A.B : s$ et $\forall x^k : A.B \preccurlyeq^+ \forall x^{k'} : A'.B'$. On en déduit (lemme 3.3.5), que $k = k'$ et que $A \equiv^+ A'$. Et d'après le lemme 3.3.8, on a $k = \text{lvl}(A) = \text{lvl}(A')$. Enfin, le lemme 3.3.16 et (2) prouvent que $\text{lvl}(M_2) = \text{lvl}(A') = k$. On peut alors prendre $N = M_1[M_2/x^k]$, et on a bien $M \twoheadrightarrow N$ et $|N| = N^-$.
- Si M est de la forme $(M_1\, M_2)$ avec $|M_1| \twoheadrightarrow N_1^-$ et $N^- = (N_1^-\, |M_2|)$ (resp. $|M_2| \twoheadrightarrow N_2^-$ et $N^- = (|M_1|\, N_2^-)$) :
  Par inversion de $\Gamma \vdash^+ M : T$, on obtient qu'il existe T' tel que $\Gamma \vdash^+ M_1 : T'$ (resp. $\Gamma \vdash^+ M_2 : T'$) et donc par hypothèse d'induction on sait qu'il existe N_1 (resp. N_2) tel que $M_1 \twoheadrightarrow N_1$ et $|N_1| = N_1^-$ (resp. $M_2 \twoheadrightarrow N_2$ et $|N_2| = N_2^-$). Il suffit alors de prendre $N = (N_1\, M_2)$ (resp. $N = (M_1\, N_2)$).
- Si M est de la forme $\forall x^k : M_1.M_2$ avec $|M_1| \twoheadrightarrow N_1^-$ et $N^- = \forall x : N_1^-.|M_2|$ (resp. $|M_2| \twoheadrightarrow N_2^-$ et $N^- = \forall x : |M_1|.N_2^-$) :
  Par inversion de $\Gamma \vdash^+ M : T$, on obtient qu'il existe T' tel que $\Gamma \vdash^+ M_1 : T'$ (resp. $\Gamma, x : M_1 \vdash^+ M_2 : T'$) et donc par hypothèse d'induction on sait qu'il existe N_1 (resp. N_2) tel que $M_1 \twoheadrightarrow N_1$ et $|N_1| = N_1^-$ (resp. $M_2 \twoheadrightarrow N_2$ et $|N_2| = N_2^-$). Il suffit alors de prendre $N = \forall x^k : N_1.M_2$ (resp. $N = \forall x^k : M_1.N_2$). ☺

Le lemme suivant affirme si deux termes bien typés dont on a effacé les annotations sont convertibles, alors ils sont également convertibles lorsque que l'on conserve les annotations.

3.3.21 Lemme
Si $\Gamma \vdash^+ M : T$ *et* $\Gamma \vdash^+ M' : T'$ *et si* $|M| \equiv |M'|$, *alors* $M \equiv^+ M'$.

Démonstration Par confluence (lemme 1.1.3), on sait qu'il existe C^- tel que $|M| \trianglerighteq C^-$ et $|M'| \trianglerighteq C^-$. Donc d'après le lemme précédent, on sait qu'il existe C_1 tel que $M \trianglerighteq C_1$ et C_2 tel que $M \trianglerighteq C_2$ avec $|C_1| = |C_2| = C^-$. Par préservation du typage, on en déduit que $\Gamma \vdash^+ C_1 : T$ et $\Gamma \vdash^+ C_2 : T'$ et donc, d'après le lemme 3.3.18, on a $C_1 = C_2$. On en déduit donc que $M \equiv^+ M'$. ☺

Le résultat précédent s'étend naturellement à la relation de cumulativité :

3.3.22 Lemme
Si $\Gamma \vdash^+ M : T$ *et* $\Gamma \vdash^+ M' : T'$ *et si* $|M| \preccurlyeq |M'|$, *alors* $M \preccurlyeq^+ M'$.

Démonstration Le cas $|M| \equiv |M'|$ est traité par le lemme précédent. On montre donc :

$$\text{Si } \Gamma \vdash^+ M : T \text{ et } \Gamma \vdash^+ M' : T' \text{ et si } |M| \prec_n |M'|, \text{ alors } M \prec_n^+ M'.$$

On procède par induction sur n :
- Si $|M| \prec_0 |M'|$: Alors il existe s et s' telles que $|M| \trianglerighteq s$ et $|M'| \trianglerighteq s'$ avec $(s, s') \in \mathcal{C}_{\mathcal{P}^2}$. D'après le lemme 3.3.20, on $M \trianglerighteq^+ s$ et $M' \trianglerighteq^+ s'$ et donc $M \prec_0^+ M'$.
- Si $|M| \prec_{n+1} |M'|$, on distingue deux cas (lemme 1.2.1) :
  - Soit on a directement $|M| \prec_n |M'|$, et on peut conclure par hypothèse de récurrence.
  - Soit $|M| \trianglerighteq \forall x : C^-.N^-$ et $|M'| \trianglerighteq \forall x : C^-.N'^-$ avec $N^- \prec_n N'^-$. D'après le lemme 3.3.20, on sait qu'il existe k, C et N tels que $|\forall x^k : C.N| = \forall x : C^-.N^-$ avec $M \trianglerighteq^+ \forall x^k : C.N$ (0). Par préservation du typage on obtient donc $\Gamma \vdash^+ \forall x^k : C.N : T$ (1). De même, on obtient qu'il existe k', C' et N' tels que $|\forall x^{k'} : C'.N'| = \forall x : C^-.N'^-$ avec $M' \trianglerighteq^+ \forall x^{k'} : C'.N'$ (0′) et par préservation du typage on obtient $\Gamma \vdash^+ \forall x^{k'} : C'.N' : T'$ (1′). Par inversion de (1) et (1′), on en déduit qu'il existe s_1, s_2, s_1', s_2' telles que $\Gamma \vdash^+ C : s_1$ (2), $\Gamma \vdash^+ C' : s_1'$ (2′), $\Gamma, x^k : C \vdash N : s_2$ (3) et $\Gamma, x^{k'} : C' \vdash^+ N' : s_2'$ (3′). D'après le lemme 3.3.18 appliqué à (2) et (2′), on prouve que $C = C'$. D'après le lemme 3.3.8 appliqué à (3) et (3′), on en déduit que $k = \text{lvl}(C) = \text{lvl}(C') = k'$. On a donc bien $\Gamma, x^k : C \vdash N : s_2$ et $\Gamma, x^k : C \vdash^+ N' : s_2'$, on peut alors appliquer l'hypothèse d'induction montrer que $N \prec_n^+ N'$. Et donc à l'aide de (0) et (0′), on conclut que $M \prec_{n+1}^+ M'$. ☺

Nous avons à présent démontré tous les outils qui vont nous permettre de démontrer que l'on peut toujours annotés de façon unique les séquents dérivables.

3.3.23 Lemme (Equivalence de $\vdash^+$ et $\vdash$)
Si $\Gamma \vdash M : T$, *alors il existe* Γ^+, M^+ *et* T^+ *tels que* $|\Gamma^+| = \Gamma$, $|M^+| = M$, $|T^+| = T$ *et* $\Gamma^+ \vdash^+ M^+ : T^+$.

Démonstration
- Axiome : Dans ce cas, $\Gamma = \langle\rangle$, $M = s$ et $T = r$ avec $(s, r) \in \mathcal{A}_{\mathcal{P}^2}$. On peut alors prendre $\Gamma^+ = \langle\rangle$, $M^+ = s$ et $T^+ = r$ et on a bien $\Gamma^+ \vdash^+ M^+ : T^+$.
- Variable : On a $M = x$, $\Gamma = \Gamma', x : T$ et $\Gamma' \vdash T : s$ avec $s \in \mathcal{S}_{\mathcal{P}^2}$ et $k = \text{lvl}(s)$. Par hypothèse d'induction, on obtient l'existence de Γ'^+ et T^+ tels que $\Gamma'^+ \vdash^+ T^+ : s$, $|\Gamma'^+| = \Gamma'$ et $|T^+| = T$. On pose donc $\Gamma^+ = \Gamma'^+, x^{\text{lvl}(T^+)} : T^+$, $M^+ = x^{\text{lvl}(T^+)}$ et on a bien $|\Gamma^+| = \Gamma$, $|M^+| = M$, $|T^+| = T$ et $\Gamma^+ \vdash^+ M^+ : T^+$.
- Affaiblissement : On a Γ est de la forme $\Gamma', x : A$, $\Gamma' \vdash M : T$ et $\Gamma' \vdash A : s$. Or par hypothèse d'induction on obtient qu'il existe Γ'^+, M^+ et T^+ tels que $\Gamma'^+ \vdash^+ M^+ : T^+$, $|\Gamma'^+| = \Gamma'$, $|M^+| = M$ et $|T^+| = T$. De même, on a l'existence de Γ''^+, A^+ tels que $\Gamma''^+ \vdash^+ A^+ : s$, $|\Gamma''^+| = \Gamma'$, $|A^+| = A$. D'après le lemme 3.3.19, on obtient que $\Gamma'^+ = \Gamma''^+$. On peut alors poser $\Gamma^+ = \Gamma'^+, x^{\text{lvl}(A^+)} : A^+$, et on a bien $|\Gamma^+| = \Gamma$, $|M^+| = M$, $|T^+| = T$ et $\Gamma^+ \vdash^+ M^+ : T^+$.

– Abstraction : On a $M = \lambda x : A.N$ et $T = \forall x : A.B$ avec $\Gamma, x : A \vdash N : B$ et $\Gamma \vdash \forall x : A.B : s$. Par hypothèse d'induction, on montre l'existence de Γ^+, Γ'^+, k, A^+, N^+, B^+, et de k',A'^+, B'^+ tels que :

$$|\Gamma^+| = |\Gamma'^+| = \Gamma \qquad |A^+| = |A'^+| = A \qquad |B^+| = |B'^+| = B \qquad |N^+| = N$$

avec $\Gamma^+, x^k : A^+ \vdash^+ N^+ : B^+$ et $\Gamma'^+ \vdash^+ \forall x^{k'} : A'^+.B'^+ : s$. Le lemme 3.3.8 montre que $k = \text{lvl}(A^+)$ et $\Gamma^+ \vdash^+ A^+ : r$ pour une sorte r. Le lemme 3.3.19 montre donc que $\Gamma^+ = \Gamma'^+$. Par inversion de $\Gamma^+ \vdash^+ \forall x^{k'} : A'^+.B'^+ : s$, on montre qu'il existe s_1 et s_2 telles que $\Gamma^+ \vdash^+ A'^+ : s_1$ et $\Gamma^+, x^{k'} : A'^+ \vdash^+ B'^+ : s_2$. Le lemme 3.3.8 montre alors que $k' = \text{lvl}(A'^+)$, puis le lemme 3.3.18, montre successivement que $A'^+ = A^+$ (et donc $k' = \text{lvl}(A^+) = k$) puis que $B'^+ = B^+$. On peut alors montrer $\Gamma^+ \vdash^+ M^+ : T^+$ grâce à la règle Abstraction.

– Application : On a $M = (M_1\, M_2)$ et $T = B[M_2/x]$ avec $\Gamma \vdash M_1 : \forall x : A.B$ et $\Gamma \vdash M_2 : A$. Par hypothèse d'induction, on obtient qu'il existe Γ^+, M_1^+, k, A^+ et B^+ tels que $\Gamma^+ \vdash^+ M_1^+ : \forall x^k : A^+.B^+$ et $\Gamma'^+ \vdash^+ M_2^+ : A'^+$. D'après le lemme 3.3.19, on a $\Gamma^+ = \Gamma'^+$. Puis d'après le lemme 3.3.14, on obtient qu'il existe s telle que $\Gamma^+ \vdash^+ A^+ : s$. Et d'après le lemme 3.3.15, on sait qu'il existe r tel que $\Gamma^+ \vdash^+ A'^+ : r$ et donc d'après le lemme 3.3.18, on montre que $A'^+ = A^+$. On peut donc prendre $T^+ = B^+[M_2^+/x^k]$ et utiliser la règle Application pour conclure $\Gamma^+ \vdash^+ M^+ : T^+$.

– Produit : On a $M = \forall x : A.B$ et $T = s_3$ avec $\Gamma \vdash A : s_1$ et $\Gamma, x : A \vdash B : s_2$ avec $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}^2}$. Par hypothèse d'induction, il existe Γ^+, A^+ tels que $\Gamma^+ \vdash^+ A^+ : s_1$, $|\Gamma^+| = \Gamma$ et $|A^+| = A$ et il existe Γ'^+, k, A'^+ et B^+ tels que $\Gamma'^+, x^k : A'^+ \vdash^+ B^+ : s_2$. D'après le lemme 3.3.8, on en déduit que $\Gamma'^+ \vdash^+ A'^+ : r$ pour une sorte r avec $k = \text{lvl}(A')$. On peut donc utiliser le lemme 3.3.19 pour montrer que $\Gamma^+ = \Gamma'^+$ et le lemme 3.3.18 pour montrer que $A^+ = A'^+$. On en déduit donc que $k = \text{lvl}(A') = \text{lvl}(A)$. Si on pose $M^+ = \forall x^k : A^+.B^+$ et $T^+ = s_3$, alors on peut donc invoquer la règle Produit pour montrer $\Gamma^+ \vdash^+ M^+ : T^+$.

– Cumulativité : On a $\Gamma \vdash M : T'$ et $\Gamma \vdash T : s$ avec $T' \preccurlyeq T$ et donc par hypothèse d'induction on obtient l'existence de Γ^+, M^+ et T'^+ tels que $\Gamma^+ \vdash^+ M^+ : T'^+$, $|\Gamma^+| = \Gamma$, $|M^+| = M$ et $|T'^+| = T'$. De même, on obtient qu'il existe Γ'^+, T^+ tels que $\Gamma'^+ \vdash^+ T^+ : s$. D'après le lemme 3.3.19, on a $\Gamma^+ = \Gamma'^+$. Et d'après le lemme 3.3.22, on a $T'^+ \preccurlyeq^+ T^+$. On peut donc utiliser la règle Cumulativité, pour prouver $\Gamma^+ \vdash^+ M^+ : T^+$.

– Cumulativité-Sortes : On a $\Gamma \vdash M : T'$ et $T = s_3$ avec $T' \preccurlyeq T$. Par hypothèse d'induction on obtient l'existence de Γ^+, M^+ et T'^+ tels que $\Gamma^+ \vdash^+ M^+ : T'^+$, $|\Gamma^+| = \Gamma$, $|M^+| = M$ et $|T'^+| = T'$. Posons $T^+ = s_3$, alors d'après le lemme 3.3.22, on a $T'^+ \preccurlyeq^+ T^+$. On peut donc utiliser la règle Cumulativité-Sortes, pour prouver $\Gamma^+ \vdash^+ M^+ : T^+$. ☺

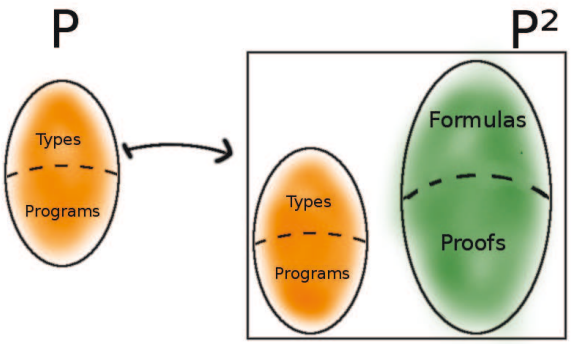
Maintenant que nous avons démontré que les annotations peuvent toujours être ajouté aux termes bien typés et que donc les systèmes avec ou sans annotations sont équivalents, nous considérerons que tous les termes sont annotés bien que l'on n'écrira explicitement les annotations que lorsqu'elles seront utiles (en particulier en l'absence d'hypothèse affirmant que les termes considérés sont bien typés). De plus nous n'utiliserons dans la suite le symbole $\vdash$ à la place de $\vdash^+$.

3.4 La structure de la logique engendrée

Nous allons à présent étudier la structure des termes typables dans le système $\mathcal{P}^2$ ainsi que les propriétés satisfaite par $\mathcal{P}^2$ en fonction de celle satisfaite par $\mathcal{P}$.

Nous avons vu dans la section précédente que les termes de $\mathcal{P}^2$ pouvait être divisé en deux catégories : les termes de premier et de second niveau.

Le premier lemme que nous démontrons dans cette section affirme que l'ensemble des termes de premier niveau typables dans $\mathcal{P}^2$ le sont également dans $\mathcal{P}$. Ainsi $\mathcal{P}^2$ contient une "copie fidèle" de $\mathcal{P}$.



Ensuite, nous verrons que les termes de second niveau contiennent également une copie de $\mathcal{P}$ qui est l'image de $\mathcal{P}$ par la fonction de soulèvement $\lceil\cdot\rceil$ vue précédemment. Enfin, tous les termes de second niveau s'envoient vers les termes de premier niveau (et donc vers $\mathcal{P}$) par une fonction de projection $\lfloor\cdot\rfloor$ qui est un inverse à gauche de la fonction de soulèvement.

On propose la définition suivante :

3.4.1 Définition (Filtrage des termes de niveau 2)
On définit la fonction filter$(\cdot)$ qui enlève des contextes les variables de second niveau :

$$\begin{aligned} \text{filter}(\langle\rangle) &= \langle\rangle \\ \text{filter}(\Gamma, x^1 : A) &= \text{filter}(\Gamma), x : A \\ \text{filter}(\Gamma, x^2 : A) &= \text{filter}(\Gamma) \end{aligned}$$

On peut alors énoncer le lemme qui affirme que les termes de premier niveau sont typables dans $\mathcal{P}$:

3.4.2 Lemme (separation)
Si $\Gamma \vdash_{\mathcal{P}^2} M : T$ *et* $\text{lvl}(M) = 1$, *alors* $\text{filter}(\Gamma) \vdash_{\mathcal{P}} M : T$.

Démonstration Par induction sur la structure de la preuve de $\Gamma \vdash_{\mathcal{P}^2} M : T$.

- Axiome : On a $\Gamma = \langle\rangle$, $M = s$ et $T = r$ avec $(s, r) \in \mathcal{A}_{\mathcal{P}^2}$. Par construction de $\mathcal{P}^2$, on a bien $\text{lvl}(s) = \text{lvl}(r) = 1$ et donc $(s, r) \in \mathcal{A}_{\mathcal{P}}$. On a donc bien $\text{filter}(\Gamma) \vdash_{\mathcal{P}} M : T$.
- Variable : On a Γ de la forme $\Gamma', x^k : T$ et $M = x^k$ avec $\Gamma' \vdash_{\mathcal{P}^2} T : s$. On a $\text{lvl}(M) = \text{lvl}(T) = 1$ d'après le lemme 3.3.16 et donc $\text{filter}(\Gamma) = \text{filter}(\Gamma'), x : T$. On en déduit donc par hypothèse d'induction on a $\text{filter}(\Gamma') \vdash_{\mathcal{P}} T : s$. On peut donc utiliser la règle Variable pour montrer que $\text{filter}(\Gamma) \vdash_{\mathcal{P}} M : T$.
- Affaiblissement : On a Γ de la forme $\Gamma', x^k : A$ avec $\Gamma' \vdash_{\mathcal{P}^2} M : T$ et $\Gamma' \vdash_{\mathcal{P}^2} A : s$. Par hypothèse d'induction, on a $\text{filter}(\Gamma') \vdash_{\mathcal{P}} M : T$. On distingue alors deux cas :
  - Soit $k = 2$ et on a $\text{filter}(\Gamma) = \text{filter}(\Gamma')$ et on a bien $\text{filter}(\Gamma) \vdash_{\mathcal{P}} M : T$.

  - Soit $k = 1$ et on a $\text{filter}(\Gamma) = \text{filter}(\Gamma'), x : A$. D'après le lemme 3.3.8, on montre que $k = \text{lvl}(A) = 1$. D'après l'hypothèse d'induction, on obtient que $\text{filter}(\Gamma') \vdash_{\mathcal{P}} A : s$. Et donc d'après la règle Affaiblissement, on montre bien que $\text{filter}(\Gamma) \vdash_{\mathcal{P}} M : T$.
- Produit : On a M de la forme $\forall x^k : A_1.A_2$ et $T = s_3$ tels que $\Gamma \vdash_{\mathcal{P}^2} A_1 : s_1, \Gamma, x^k : A_1 \vdash_{\mathcal{P}^2} A_2 : s_2$ avec $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}^2}$. Comme on a $\text{lvl}(M) = 1$, cela signifie que $\text{lvl}(A_2) = \text{lvl}(M) = 1$. Et d'après le lemme 3.3.16, $\text{lvl}(A_2) = \text{lvl}(s_2)$. Or par construction de $\mathcal{R}_{\mathcal{P}^2}$, si on a $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}^2}$ et $\text{lvl}(s_2) = 1$, alors $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}}$ et $\text{lvl}(s_1) = \text{lvl}(s_2) = \text{lvl}(s_3) = 1$. D'après le lemme 3.3.16, on a donc $\text{lvl}(A_1) = \text{lvl}(s_1) = 1$ et d'après le lemme 3.3.8, on en déduit $k = \text{lvl}(A_1) = 1$. On a alors que $\text{filter}(\Gamma, x^k : A_1) = \text{filter}(\Gamma), x : A_1$. Et par hypothèse d'induction, on a $\text{filter}(\Gamma) \vdash_{\mathcal{P}} A : s_1$ et $\text{filter}(\Gamma), x : A_1 \vdash_{\mathcal{P}} A_2 : s_2$. On peut alors utiliser la règle Produit pour montrer $\text{filter}(\Gamma) \vdash_{\mathcal{P}} M : T$.
- Application : On a M de la forme $M_1\, M_2$ et T de la forme $B[M_2/x]$ et il existe A tel que $\Gamma \vdash_{\mathcal{P}^2} M_1 : \forall x : A.B$ et $\Gamma \vdash_{\mathcal{P}^2} M_2 : A$. Comme on a $\text{lvl}(M) = 1$, on en déduit que $\text{lvl}(M_1) = \text{lvl}(M) = 1$ et donc, d'après le lemme 3.3.16, que $\text{lvl}(B) = 1$. D'après le lemme 3.3.14, on sait qu'il existe $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}^2}$ telles que $\Gamma \vdash_{\mathcal{P}^2} A : s_1$, $\Gamma, x : A \vdash_{\mathcal{P}^2} B : s_2$. D'après le lemme 3.3.16, on en déduit que $\text{lvl}(s_2) = \text{lvl}(B) = 1$. Or par construction de $\mathcal{R}_{\mathcal{P}^2}$, on en déduit que $\text{lvl}(s_1) = 1$ et donc que, d'après le lemme 3.3.16, $\text{lvl}(s_1) = \text{lvl}(A) = \text{lvl}(M_2) = 1$. On peut donc appliquer l'hypothèse d'induction pour montrer que $\text{filter}(\Gamma) \vdash_{\mathcal{P}} M_1 : \forall x : A.B$ et $\text{filter}(\Gamma) \vdash_{\mathcal{P}} M_2 : A$. Il suffit alors d'appliquer la règle Application, pour prouver que $\text{filter}(\Gamma) \vdash_{\mathcal{P}} (M_1\, M_2) : B[M_2/x]$.
- Abstraction : On a M de la forme $\lambda x^k : A.N$ et T de la forme $\forall x : A.B$ avec $\Gamma, x^k : A \vdash_{\mathcal{P}^2} N : B$ et $\Gamma \vdash_{\mathcal{P}^2} \forall x : A.B : s$. Comme $\text{lvl}(M) = 1$, on a $\text{lvl}(N) = \text{lvl}(M) = 1$. Et donc d'après le lemme 3.3.16, on en déduit $\text{lvl}(B) = \text{lvl}(N) = 1$. Puis, par inversion de $\Gamma \vdash_{\mathcal{P}^2} \forall x : A.B : s$ et par construction de $\mathcal{R}_{\mathcal{P}^2}$, on montre que $\text{lvl}(A) = 1$ et d'après le lemme 3.3.8, on en déduit que $k = \text{lvl}(A) = 1$. On a donc $\text{filter}(\Gamma, x^k : A) = \text{filter}(\Gamma), x : A$ et par hypothèse d'induction, on a $\text{filter}(\Gamma), x : A \vdash_{\mathcal{P}} N : B$ et $\text{filter}(\Gamma) \vdash_{\mathcal{P}} \forall x : A.B : s$. On peut donc appliquer la règle Abstraction pour montrer $\Gamma \vdash_{\mathcal{P}} M : T$.
- Cumulativité : On a $\Gamma \vdash_{\mathcal{P}^2} M : T'$ et $T' \preccurlyeq T$ avec $\Gamma \vdash_{\mathcal{P}^2} T : s$. On a $\text{lvl}(M) = 1$ par hypothèse, on peut donc appliquer le lemme 3.3.16 pour montrer que $\text{lvl}(T') = \text{lvl}(M) = 1$. Et d'après le lemme 3.3.22, on en déduit que $\text{lvl}(T) = 1$. On peut donc appliquer les hypothèse d'induction pour montrer que $\text{filter}(\Gamma) \vdash_{\mathcal{P}} M : T'$ et $\text{filter}(\Gamma) \vdash_{\mathcal{P}} T : s$. On conclut grâce à la règle Cumulativité que $\text{filter}(\Gamma) \vdash_{\mathcal{P}} M : T$.
- Cumulativité-Sortes : On a ici $\Gamma \vdash_{\mathcal{P}^2} M : T'$ et $T = s$ avec $T' \preccurlyeq s$. Par hypothèse on a $\text{lvl}(M) = 1$ et on peut donc appliquer l'hypothèse d'induction pour montrer que $\text{filter}(\Gamma) \vdash_{\mathcal{P}} M : T'$ et on conclut grâce à la règle Cumulativité-Sortes que $\text{filter}(\Gamma) \vdash_{\mathcal{P}} M : T$. ☺

Soulèvement et projection. Nous avons déjà introduit la fonction de *soulèvement* définie par le morphisme $s \mapsto \lceil s \rceil$ de $\mathcal{P}$ dans $\mathcal{P}^2$. Par exemple, dans $\mathcal{F}^2$, la fonction $\lceil \cdot \rceil$ envoie les types de $\mathcal{F}$ vers des formules du second ordre (sans premier ordre). Ainsi $\lceil \forall \alpha : \star.\alpha \to \alpha \rceil$ s'envoie vers $\forall X : \lceil \star \rceil.X \to X$ (on a ici renommé la variable pour insister sur le fait qu'on est passé d'une variable de type à une variable propositionnelle). Le soulèvement envoie les termes de premier niveau (c'est-à-dire les termes de $\mathcal{P}$) vers des termes de second niveau ne contenant aucun sous-terme de premier niveau.

3.4.3 Lemme
Si $\Gamma \vdash_{\mathcal{P}} M : T$, *alors* $\lceil \Gamma \rceil \vdash_{\mathcal{P}^2} \lceil M \rceil : \lceil T \rceil$ *et* $\text{lvl}(\lceil M \rceil) = 2$.

Démonstration Le fait que $\lceil\cdot\rceil$ soit un morphisme montre que l'on a bien $\lceil\Gamma\rceil \vdash_{\mathcal{P}^2} \lceil M\rceil : \lceil T\rceil$. Nous montrons alors facilement que $\mathrm{lvl}(\lceil M\rceil) = 2$ par induction sur la dérivation de $\Gamma \vdash_{\mathcal{P}} M : T$. ☺

On étendra donc $\lceil\cdot\rceil$ aux termes annotés de la façon suivante :

$$\begin{array}{lcl}
\lceil x^1\rceil & = & x^2 \\
\lceil s\rceil & = & \lceil s\rceil \\
\lceil \forall x^1 : A.B\rceil & = & \forall x^2 : \lceil A\rceil.\lceil B\rceil \\
\lceil \lambda x^1 : A.B\rceil & = & \lambda x^2 : \lceil A\rceil.\lceil B\rceil \\
\lceil (A\,B)\rceil & = & \lceil A\rceil\,\lceil B\rceil \\
 & & \\
\lceil\langle\rangle\rceil & = & \langle\rangle \\
\lceil\Gamma, x^1 : A\rceil & = & \lceil\Gamma\rceil, x^2 : \lceil A\rceil.
\end{array}$$

On définit maintenant la projection $\lfloor\cdot\rfloor$ qui sera un inverse à gauche de la projection : $\lfloor\lceil A\rceil\rfloor = A$. Son rôle est d'effacer, dans les termes de second niveau, les quantifications, les abstractions et les applications des termes de premier niveau. C'est donc les quantifications sur les individus qui sont effacées.

3.4.4 Définition (Projection)
La fonction de projection $\lfloor\cdot\rfloor$ est défini par les équations suivantes :

$$\begin{array}{lcl}
\lfloor x^2\rfloor & = & x^1 \\
\lfloor\lceil s\rceil\rfloor & = & s \\
\lfloor \forall x^1 : A.B\rfloor & = & \lfloor B\rfloor \\
\lfloor \forall x^2 : A.B\rfloor & = & \forall x^1 : \lfloor A\rfloor.\lfloor B\rfloor \\
\lfloor \lambda x^1 : A.B\rfloor & = & \lfloor B\rfloor \\
\lfloor \lambda x^2 : A.B\rfloor & = & \lambda x^1 : \lfloor A\rfloor.\lfloor B\rfloor \\
\lfloor (A\,B)\rfloor & = & \lfloor A\rfloor \text{ si } \mathrm{lvl}(B) = 1 \\
\lfloor (A\,B)\rfloor & = & \lfloor A\rfloor\,\lfloor B\rfloor \text{ sinon} \\
 & & \\
\lfloor\langle\rangle\rfloor & = & \langle\rangle \\
\lfloor\Gamma, x^1 : A\rfloor & = & \lfloor\Gamma\rfloor \\
\lfloor\Gamma, x^2 : A\rfloor & = & \lfloor\Gamma\rfloor, x : \lfloor A\rfloor.
\end{array}$$

On notera que $\lfloor A\rfloor$ n'est bien défini que lorsque $\mathrm{lvl}(A) = 2$.

Voici quelques exemples de projections de propositions dans $\mathcal{F}^2$:

$$\lfloor \mathsf{True} \rfloor = \lfloor \forall X : \lceil \star \rceil . X \to X \rfloor = \forall X : \star, X \to X = \mathsf{unit}$$

$$\begin{aligned}
\lfloor \forall \alpha : \star, x : \alpha . x =_\alpha x \rfloor &= \lfloor \forall \alpha : \star, x : \alpha, X : \alpha \to \lceil \star \rceil . X\, x \to X\, x \rfloor \\
&= \lfloor \forall X : \alpha \to \lceil \star \rceil . X\, x \to X\, x \rfloor \\
&= \forall X : \star . \lfloor X\, x \to X\, x \rfloor \\
&= \forall X : \star . X \to X \\
&= \mathsf{unit}
\end{aligned}$$

$$\begin{aligned}
\lfloor t \in \mathbb{N} \rfloor &= \lfloor \forall X : \mathsf{nat} \to \lceil \star \rceil . (\forall y : \mathsf{nat} . X\, y \to X\, (\mathsf{S}\, y)) \to X\, \overline{0} \to X\, t \rfloor \\
&= \forall X : \star . \lfloor (\forall y : \mathsf{nat} . X\, y \to X\, (\mathsf{S}\, y)) \to X\, \overline{0} \to X\, t \rfloor \\
&= \forall X : \star . \lfloor \forall y : \mathsf{nat} . X\, y \to X\, (\mathsf{S}\, y) \rfloor \to \lfloor X\, \overline{0} \to X\, t \rfloor \\
&= \forall X : \star . \lfloor X\, y \to X\, (\mathsf{S}\, y) \rfloor \to X \to X \\
&= \forall X : \star . (X \to X) \to X \to X \\
&= \mathsf{nat}
\end{aligned}$$

La projection se comporte de la façon suivante vis-à-vis de la substitution :

3.4.5 Lemme
Soit M tel que $\mathrm{lvl}(M) = 2$ *et N tel que* $\mathrm{lvl}(N) = k$, *alors :*

$$\lfloor M[N/x^k] \rfloor = \begin{cases} \lfloor M \rfloor & \text{si } k = 1 \\ \lfloor M \rfloor [\lfloor N \rfloor / x^1] & \text{si } k = 2 \end{cases}$$

Démonstration Par induction sur la structure de M,
- $M = y^2$ (avec $y^2 \neq x^k$) ou $M = s$: On a bien $\lfloor M[N/x^k] \rfloor = \lfloor M \rfloor = \lfloor M \rfloor [\lfloor N \rfloor / x^1]$.
- $M = x^k$: Dans ce cas, on a $k = \mathrm{lvl}(M) = 2$, $\lfloor M \rfloor = x^1$ et on a :

$$\lfloor x^2[N/x^2] \rfloor = \lfloor N \rfloor = x^1[\lfloor N \rfloor / x^1] = \lfloor M \rfloor [\lfloor N \rfloor / x^1]$$

- $M = M_1 M_2$, (resp. $M = \forall y : M_2 . M_1$, resp. $M = \lambda y : M_2 . M_1$) : On a $\mathrm{lvl}(M) = \mathrm{lvl}(M_1) = 2$, on a donc par hypothèse d'induction :

$$\lfloor M_1[N/x^k] \rfloor = \begin{cases} \lfloor M_1 \rfloor & \text{si } k = 1 \\ \lfloor M_1 \rfloor [\lfloor N \rfloor / x^1] & \text{si } k = 2 \end{cases}$$

  On distingue deux cas :
  - $\mathrm{lvl}(M_2) = 1$: Dans ce cas, $\lfloor M \rfloor = \lfloor M_1 \rfloor$, $\mathrm{lvl}(M_2[N/x^k]) = \mathrm{lvl}(M_2) = 1$ (d'après le lemme 3.3.2) et donc $\lfloor M[N/x^k] \rfloor = \lfloor M_1[N/x^k] \rfloor$. Et on conclut par hypothèse d'induction.
  - $\mathrm{lvl}(M_2) = 2$: Dans ce cas, $\lfloor M \rfloor = \lfloor M_1 \rfloor \lfloor M_2 \rfloor$ (resp. $\lfloor \forall y^2 : M_2 . M_1 \rfloor = \forall y^1 : \lfloor M_2 \rfloor . \lfloor M_1 \rfloor$, resp. $\lfloor \lambda y^2 : M_2 . M_1 \rfloor = \lambda y^1 : \lfloor M_2 \rfloor . \lfloor M_1 \rfloor$). De plus, d'après le lemme 3.3.2, $\mathrm{lvl}(M_2[N/x^k]) =$

$\mathrm{lvl}(M_2) = 2$. On en déduit donc que $\lfloor M[N/x^k] \rfloor$ est égal à $(\lfloor M_1[N/x^k] \rfloor \lfloor M_2[N/x^k] \rfloor)$ (resp. $\forall y^1 : \lfloor M_2[N/x^k] \rfloor . \lfloor M_1[N/x^k] \rfloor$, resp. $\lambda y^1 : \lfloor M_2[N/x^k] \rfloor . \lfloor M_1[N/x^k] \rfloor$). On peut donc appliquer l'hypothèse d'induction pour montrer que :

$$\lfloor M_2[N/x^k] \rfloor = \begin{cases} \lfloor M_2 \rfloor & \text{si } k = 1 \\ \lfloor M_2 \rfloor [\lfloor N \rfloor / x^1] & \text{si } k = 2 \end{cases}$$

On a donc selon le cas considéré :
– $k = 1$:

$$\begin{array}{rcl} \Big(\lfloor M_1[N/x^1] \rfloor \lfloor M_2[N/x^k] \rfloor\Big) & = & (\lfloor M_1 \rfloor \lfloor M_2 \rfloor) \\ \forall y^1 : \lfloor M_2[N/x^1] \rfloor . \lfloor M_1[N/x^1] \rfloor & = & \forall y^1 : \lfloor M_2 \rfloor . \lfloor M_1 \rfloor \\ \lambda y^1 : \lfloor M_2[N/x^1] \rfloor . \lfloor M_1[N/x^1] \rfloor & = & \lambda y^1 : \lfloor M_2 \rfloor . \lfloor M_1 \rfloor \end{array}$$

– $k = 2$:

$$\begin{array}{rcl} \Big(\lfloor M_1[N/x^2] \rfloor \lfloor M_2[N/^2x] \rfloor\Big) & = & \Big(\lfloor M_1 \rfloor [N/x^2] \lfloor M_2 \rfloor [N/x^2]\Big) \\ \forall y^1 : \lfloor M_2[N/x^2] \rfloor . \lfloor M_1[N/x^2] \rfloor & = & \forall y^1 : \lfloor M_2 \rfloor [N/x^2] . \lfloor M_1 \rfloor [N/x^2] \\ \lambda y^1 : \lfloor M_2[N/x^2] \rfloor . \lfloor M_1[N/x^2] \rfloor & = & \lambda y^1 : \lfloor M_2 \rfloor [N/x^2] . \lfloor M_1 \rfloor [N/x^2] \end{array}$$

On en conclut bien :

$$\lfloor M[N/x^k] \rfloor = \begin{cases} \lfloor M \rfloor & \text{si } k = 1 \\ \lfloor M \rfloor [\lfloor N \rfloor / x^1] & \text{si } k = 2 \end{cases}$$

☺

Dans le système $\mathcal{P}^2$, on peut classer les rédexs en trois catégories :
– Les rédexs correspondant à la réduction des abstractions de premier niveau dont le type de l'abstraction est formé avec une règle de la forme $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}}$.
– Les rédexs correspondant à la réduction des abstractions de second niveau dont le type de l'abstraction est formé avec une règle de la forme $(\lceil s_1 \rceil, \lceil s_2 \rceil, \lceil s_3 \rceil) \in \mathcal{R}_{\mathcal{P}^2}$ (avec s_1, s_2, s_3 trois sortes de premier niveau appartenant à $\mathcal{S}_{\mathcal{P}}$).
– Les rédexs correspondant à la réduction des abstractions de second niveau dont le type de l'abstraction est formé avec une règle de la forme $(s_1, \lceil s_2 \rceil, \lceil s_3 \rceil) \in \mathcal{R}_{\mathcal{P}^2}$ (avec s_1, s_2, s_3 trois sortes de premier niveau appartenant à $\mathcal{S}_{\mathcal{P}}$).

On définit donc trois réductions $\twoheadrightarrow^1$, $\twoheadrightarrow^2$ et $\twoheadrightarrow^{3/2}$ pour ces classes respectives. Les règles correspondant à la réduction des rédexs sont les suivantes :

$$\frac{\mathrm{lvl}(B) = 1}{(\lambda x^{\mathrm{lvl}(C)} : A.B)\, C \twoheadrightarrow^1 B[C/x^{\mathrm{lvl}(C)}]}\ \textsc{Redex} \qquad \frac{\mathrm{lvl}(B) = \mathrm{lvl}(C) = 2}{(\lambda x^{\mathrm{lvl}(C)} : A.B)\, C \twoheadrightarrow^2 B[C/x^{\mathrm{lvl}(C)}]}\ \textsc{Redex}$$

$$\frac{\mathrm{lvl}(B) = 2 \text{ et } \mathrm{lvl}(C) = 1}{(\lambda x^{\mathrm{lvl}(C)} : A.B)\, C \twoheadrightarrow^{3/2} B[C/x^{\mathrm{lvl}(C)}]}\ \textsc{Redex}$$

Ces trois relations satisfont également les règles de congruence suivantes (pour $i = 1, 2, 3/2$) :

$$\frac{M \twoheadrightarrow^i M'}{(M\,N) \twoheadrightarrow^i (M'\,N)}\ \textsc{Application-Gauche} \qquad \frac{N \twoheadrightarrow^i N'}{(M\,N) \twoheadrightarrow^i (M\,N')}\ \textsc{Application-Droite} \qquad \frac{A \twoheadrightarrow^i A'}{\lambda x^k : A.M \twoheadrightarrow^i \lambda x^k : A'.M}\ \textsc{Abstraction-Gauche}$$

$$\frac{M \twoheadrightarrow^i M'}{\lambda x^k : A.M \twoheadrightarrow^i \lambda x^k : A.M'}\ \textsc{Abstraction-Droite} \qquad \frac{A \twoheadrightarrow^i A'}{\forall x^k : A.B \twoheadrightarrow^i \forall x^k : A'.B}\ \textsc{Produit-Gauche} \qquad \frac{B \twoheadrightarrow^i B'}{\forall x^k : A.B \twoheadrightarrow^i \forall x^k : A.B'}\ \textsc{Produit-Droite}$$

3.4.6 Lemme
Si $M \twoheadrightarrow M'$, alors $M \twoheadrightarrow^1 M'$ ou $M \twoheadrightarrow^{3/2} M'$ ou $M \twoheadrightarrow^2 M'$.

Démonstration On procède par une induction immédiate sur les règles définissant $M \twoheadrightarrow M'$. Dans le cas Redex, on choisit la relation qui convient par inspection des niveaux de la fonction et de l'argument. ☺

Le lemme ci-dessous montre que la projection efface les rédexs de $\twoheadrightarrow^1$ et $\twoheadrightarrow^{3/2}$ et préserve les rédexs de $\twoheadrightarrow^2$.

3.4.7 Lemme
Supposons $\mathrm{lvl}(M) = 2$.

1. *Si $M \twoheadrightarrow^1 M'$, alors $\lfloor M \rfloor = \lfloor M' \rfloor$.*
2. *Si $M \twoheadrightarrow^{3/2} M'$, alors $\lfloor M \rfloor = \lfloor M' \rfloor$.*
3. *Si $M \twoheadrightarrow^2 M'$, alors $\lfloor M \rfloor \twoheadrightarrow \lfloor M' \rfloor$.*
4. *Si $M \rhd M'$, alors $\lfloor M \rfloor \trianglerighteq \lfloor M' \rfloor$.*
5. *Si $M \equiv M'$, alors $\lfloor M \rfloor \equiv \lfloor M' \rfloor$.*
6. *Si $M \prec_n M'$, alors $\lfloor M \rfloor \prec_n \lfloor M' \rfloor$.*
7. *Si $M \preccurlyeq M'$, alors $\lfloor M \rfloor \preccurlyeq \lfloor M' \rfloor$.*

Démonstration 1. Par induction sur la structure des règles définissant $\twoheadrightarrow^1$. Le cas Redex est impossible car $\mathrm{lvl}(M) = 2$. Cela signifie que le rédex réduit se trouve dans une partie nécessairement effacée par la projection. Pour prouver Application-Gauche, Abstraction-Droite, Produit-Droite se traitent en invoquant l'hypothèse d'induction (car on a $\mathrm{lvl}(M_1\,M_2) = \mathrm{lvl}(M_1)$, $\mathrm{lvl}(\lambda x : A.N) = \mathrm{lvl}(\forall x : A.N) = \mathrm{lvl}(N)$). Pour chacun des cas Application-Droite, Abstraction-Gauche, Produit-Gauche, on distingue deux cas selon le niveau du sous-terme qui se fait potentiellement effacer par la projection (l'argument pour les applications, le domaine pour l'abstraction et le produit). Dans le cas où ce sous-terme est de niveau 2, on peut conclure par hypothèse d'induction et dans le cas où il est de niveau 1, ce dernier est effacé par la projection et il n'y a plus rien à démontrer.

2. On procède par induction sur les règles définissant $\twoheadrightarrow^{3/2}$. Le cas Redex est ici possible. En effet, si M est de la forme $(\lambda x^{\mathrm{lvl}(C)} : A.B)\, C$ et M' est de la forme $B[C/x^{\mathrm{lvl}(C)}]$ avec $\mathrm{lvl}(B) = 2$ et $\mathrm{lvl}(C) = 1$, alors $\lfloor M \rfloor = \lfloor \lambda x^{\mathrm{lvl}(C)} : A.B \rfloor = \lfloor B \rfloor$ et $\lfloor M' \rfloor = \lfloor B[C/x^{\mathrm{lvl}(C)}] \rfloor = \lfloor B \rfloor$ (lemme 3.4.5). On a donc bien dans ce cas, $\lfloor M \rfloor = \lfloor M' \rfloor$. Enfin, les autres cas se traitent de façon analogue à $\twoheadrightarrow^1$.
3. On procède par induction sur les règles définissant $\twoheadrightarrow^2$. Dans le cas Redex, on a M est de la forme $(\lambda x^{\mathrm{lvl}(C)} : A.B)\, C$ et M' est de la forme $B[C/x^{\mathrm{lvl}(C)}]$ avec $\mathrm{lvl}(B) = \mathrm{lvl}(C) = 2$. On a donc d'après le lemme 3.4.5 :

   $$\lfloor M \rfloor = \lfloor (\lambda x^2 : A.B)\, C \rfloor = (\lambda x^1 : \lfloor A \rfloor.\lfloor B \rfloor)\, \lfloor C \rfloor \twoheadrightarrow \lfloor B \rfloor[\lfloor C \rfloor / x^1] = \lfloor B[C/x^2] \rfloor = \lfloor M' \rfloor$$

   Les autres cas se traitent de façon analogue à $\twoheadrightarrow^1$.
4. C'est une conséquence directe du lemme 3.4.6 et de 1, 2, et 3
5. C'est une conséquence directe du cas précédent.
6. On procède par induction sur n :
   - Si $M \prec_0 M'$ alors il existe $(s, s') \in \mathcal{C}_{\mathcal{P}^2}$ telles que $M \equiv s$ et $M' \equiv s'$. D'après le cas précédent, on en déduit que $\lfloor M \rfloor \equiv \lfloor s \rfloor$ et $\lfloor M' \rfloor \equiv \lfloor s' \rfloor$. Or par construction de $\mathcal{P}^2$, $(s, s') \in \mathcal{C}_{\mathcal{P}^2}$ et $\mathrm{lvl}(s) = 2$ implique $(\lfloor s \rfloor, \lfloor s' \rfloor) \in \mathcal{C}_{\mathcal{P}^2}$. On en déduit donc bien que $\lfloor M \rfloor \preccurlyeq \lfloor M' \rfloor$.
   - Si $M \prec_{n+1} M'$, alors on distingue deux cas :
     - Soit on a directement $M \prec_n M'$, et on peut conclure en appliquant l'hypothèse de récurrence.
     - Soit $M \equiv \forall x^k : C.N$ et $M' \equiv \forall x^k : C.N'$ avec $N \preccurlyeq_n N'$. Montrons $M \preccurlyeq_{n+1} M'$. Or $2 = \mathrm{lvl}(M) = \mathrm{lvl}(\forall x : C.N) = \mathrm{lvl}(N)$, on peut donc appliquer l'hypothèse de récurrence pour montrer que $\lfloor N \rfloor \preccurlyeq_n \lfloor N' \rfloor$ (1).
       - $k = 1$: on a $\lfloor M \rfloor = \lfloor N \rfloor$ et $\lfloor M' \rfloor = \lfloor N' \rfloor$, et donc, d'après (1), on a $\lfloor M \rfloor \preccurlyeq_n \lfloor M' \rfloor$. On en conclut alors $\lfloor M \rfloor \preccurlyeq_{n+1} \lfloor M' \rfloor$.
       - $k = 2$: on a $\lfloor M \rfloor = \forall x^1 : \lfloor C \rfloor.\lfloor N \rfloor$ et $\lfloor M' \rfloor = \forall x^1 : \lfloor C \rfloor.\lfloor N' \rfloor$. D'après (1), on obtient bien :

         $$\lfloor M \rfloor = \forall x^1 : \lfloor C \rfloor.N \preccurlyeq_{n+1} \forall x^1 : \lfloor C \rfloor.N' = \lfloor M' \rfloor$$
7. Conséquence directe des deux cas précédents. ☺

Enfin, voici le lemme de correction de la projection :

3.4.8 Lemme
Si $\Gamma \vdash_{\mathcal{P}^2} M : T$ *et* $\mathrm{lvl}(M) = 2$, *alors* $\lfloor \Gamma \rfloor \vdash_{\mathcal{P}} \lfloor M \rfloor : \lfloor T \rfloor$.

Démonstration Par induction sur la structure de $\Gamma \vdash_{\mathcal{P}^2} M : T$:
- Axiome : Dans ce cas, $\Gamma = \langle\rangle$, $M = s$ et $T = r$ avec $(s, r) \in \mathcal{A}_{\mathcal{P}^2}$. Puisque $\mathrm{lvl}(s) = 2$, on sait qu'il existe $(s', r') \in \mathcal{A}_{\mathcal{P}}$ telles que $\lceil s' \rceil = s$ et $\lceil r' \rceil = r$. On a donc bien $\vdash_{\mathcal{P}} \lfloor M \rfloor : \lfloor T \rfloor$.
- Variable : On a Γ de la forme $\Gamma', x^2 : T$ et $M = x^2$ avec $\Gamma' \vdash_{\mathcal{P}^2} T : s$. On a $\mathrm{lvl}(T) = 2$, on peut donc appliquer l'hypothèse d'induction pour montrer que $\lfloor \Gamma' \rfloor \vdash_{\mathcal{P}} \lfloor T \rfloor : \lfloor s \rfloor$ et utiliser la règle Variable pour montrer que $\lfloor \Gamma \rfloor \vdash \lfloor M \rfloor : \lfloor T \rfloor$.
- Affaiblissement : On a Γ de la forme $\Gamma', x^k : A$ tel que $\Gamma' \vdash M : T$ et $\Gamma' \vdash A : s$. Par hypothèse d'induction on a $\lfloor \Gamma' \rfloor \vdash \lfloor M \rfloor : \lfloor T \rfloor$. On distingue alors deux cas :
  - $k = 1$: on a $\lfloor \Gamma \rfloor = \lfloor \Gamma' \rfloor$ et on a $\lfloor \Gamma \rfloor \vdash \lfloor M \rfloor : \lfloor T \rfloor$.

  - $k = 2$: on a $\lfloor\Gamma\rfloor = \lfloor\Gamma'\rfloor, x : \lfloor A\rfloor$ et on a $\mathrm{lvl}(A) = 2$. On peut donc appliquer l'hypothèse d'induction, pour prouver que $\lfloor\Gamma'\rfloor \vdash \lfloor T\rfloor : \lfloor s\rfloor$ et $\lfloor\Gamma'\rfloor \vdash \lfloor M\rfloor : \lfloor T\rfloor$. Enfin, la règle Affaiblissement, nous permet de déduire $\lfloor\Gamma\rfloor \vdash \lfloor M\rfloor : \lfloor T\rfloor$.
- Abstraction : Dans ce cas, on a $M = \lambda x^k : A.N$ et $T = \forall x^k : A.B$ avec $\Gamma, x^k : A \vdash N : B$ et $\Gamma \vdash \forall x^k : A.B : s$. On a $\mathrm{lvl}(M) = \mathrm{lvl}(N) = \mathrm{lvl}(B) = \mathrm{lvl}(T) = 2$, on peut donc appliquer l'hypothèse d'induction pour montrer que $\lfloor\Gamma, x^k : A\rfloor \vdash \lfloor N\rfloor : \lfloor B\rfloor$ (1) et $\lfloor\Gamma\rfloor \vdash \lfloor\forall x^k : A.B\rfloor : \lfloor s\rfloor$ (2). On distingue alors deux cas :
  - $k = 1$: Dans ce cas, on a $\lfloor\Gamma, x^1 : A\rfloor = \lfloor\Gamma\rfloor$, $\lfloor M\rfloor = \lfloor N\rfloor$ et $\lfloor T\rfloor = \lfloor B\rfloor$. On a donc bien d'après (1) que $\lfloor\Gamma\rfloor \vdash \lfloor M\rfloor : \lfloor T\rfloor$.
  - $k = 2$: Dans ce cas, on a $\lfloor\Gamma, x^2 : A\rfloor = \lfloor\Gamma\rfloor, x : \lfloor A\rfloor$, $\lfloor M\rfloor = \lambda x : \lfloor A\rfloor.\lfloor N\rfloor$ et $\lfloor T\rfloor = \forall x : \lfloor A\rfloor.\lfloor B\rfloor$ et d'après la règle Abstraction, (1) et (2) on montre bien que $\lfloor\Gamma\rfloor \vdash \lfloor M\rfloor : \lfloor T\rfloor$.
- Application : Dans ce cas, on a $M = M_1\ M_2$ et T de la forme $B[M_2/x]$ avec $\Gamma \vdash M_1 : \forall x^k : A.B$ et $\Gamma \vdash M_2 : A$. On a $\mathrm{lvl}(M) = \mathrm{lvl}(M_1) = 2$. On a donc par hypothèse d'induction que $\lfloor\Gamma\rfloor \vdash \lfloor M_1\rfloor : \lfloor\forall x^k : A.B\rfloor$ (1). On distingue alors deux cas :
  - $k = 1$: Dans ce cas, on a $\lfloor\forall x^1 : A.B\rfloor = \lfloor B\rfloor$ et $\lfloor M\rfloor = \lfloor M_1\rfloor$. De plus d'après le lemme 3.4.5, $\lfloor T\rfloor = \lfloor B\rfloor$. On a donc bien d'après (1) que $\lfloor\Gamma\rfloor \vdash \lfloor M\rfloor : \lfloor T\rfloor$.
  - $k = 2$: Dans ce cas, on a $\mathrm{lvl}(A) = \mathrm{lvl}(M_2) = 2$ (lemme 3.3.16) et par hypothèse d'induction on obtient $\lfloor\Gamma\rfloor \vdash \lfloor M_2\rfloor : \lfloor A\rfloor$. De plus, $\lfloor\forall x^2 : A.B\rfloor = \forall x : \lfloor A\rfloor.\lfloor B\rfloor$, $\lfloor M\rfloor = (\lfloor M_1\rfloor\ \lfloor M_2\rfloor)$ et, d'après le lemme 3.4.5, $\lfloor T\rfloor = \lfloor B\rfloor[\lfloor M_2\rfloor/x]$. Et donc d'après la règle Application et (1), on a bien $\lfloor\Gamma\rfloor \vdash \lfloor M\rfloor : \lfloor T\rfloor$.
- Produit : Dans ce cas, M est de la forme $\forall x^k : A.B$ et $T = s_3$ avec $\Gamma \vdash A : s_1$ et $\Gamma, x^k : A \vdash B : s_2$ pour $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}^2}$. On a par hypothèse, $\mathrm{lvl}(M) = \mathrm{lvl}(B) = 2$, on peut donc appliquer l'hypothèse d'induction on a $\lfloor\Gamma, x^k : A\rfloor \vdash \lfloor B\rfloor : \lfloor s_2\rfloor$ (1). On distingue alors deux cas :
  - $k = 1$: Dans ce cas, on a $\lfloor M\rfloor = \lfloor\forall x^1 : A.B\rfloor = \lfloor B\rfloor$ et $\lfloor\Gamma, x^1 : A\rfloor = \lfloor\Gamma\rfloor$. On a donc bien d'après (1) que $\lfloor\Gamma\rfloor \vdash \lfloor M\rfloor : \lfloor T\rfloor$ car $s_2 = s_3$.
  - $k = 2$: Dans ce cas, on a $\mathrm{lvl}(A) = \mathrm{lvl}(s_1) = 2$ et $\lfloor\Gamma, x^2 : A\rfloor = \lfloor\Gamma\rfloor, x : \lfloor A\rfloor$. Or par hypothèse d'induction on obtient $\lfloor\Gamma\rfloor \vdash \lfloor A\rfloor : \lfloor s_1\rfloor$ (2). Et par construction de $\mathcal{R}_{\mathcal{P}^2}$, on a $(\lfloor s_1\rfloor, \lfloor s_2\rfloor, \lfloor s_3\rfloor) \in \mathcal{R}_{\mathcal{P}}$ et donc d'après la règle Produit, (2) et (1), on montre bien $\lfloor\Gamma\rfloor \vdash \lfloor M\rfloor : \lfloor T\rfloor$.
- Cumulativité : On a $\Gamma \vdash M : T'$, $T' \preccurlyeq T$ et $\Gamma \vdash T : s$. On a par hypothèse $\mathrm{lvl}(M) = 2$, on en déduit donc, d'après le lemme 3.3.16, que $\mathrm{lvl}(T') = 2$ puis d'après le lemme 3.3.22 que $\mathrm{lvl}(T) = 2$. On peut donc appliquer l'hypothèse d'induction pour montrer que $\lfloor\Gamma\rfloor \vdash \lfloor M\rfloor : \lfloor T'\rfloor$ et $\lfloor\Gamma\rfloor \vdash \lfloor T\rfloor : \lfloor s\rfloor$. Or d'après, le lemme 3.4.7, on a $\lfloor T'\rfloor \preccurlyeq \lfloor T\rfloor$ et donc en appliquant la règle Cumulativité, on prouve bien $\lfloor\Gamma\rfloor \vdash \lfloor M\rfloor : \lfloor T\rfloor$.
- Cumulativité-Sortes : On a $\Gamma \vdash M : T'$ et $T' \preccurlyeq s = T$. D'après les lemmes 3.3.16 et 3.3.7, on a $\mathrm{lvl}(M) = \mathrm{lvl}(T') = \mathrm{lvl}(s) = 2$. Par hypothèse d'induction, on en déduit que $\lfloor\Gamma\rfloor \vdash \lfloor M\rfloor : \lfloor T'\rfloor$ et d'après le lemme 3.4.7, on a $\lfloor T'\rfloor \preccurlyeq \lfloor s\rfloor = \lfloor T\rfloor$. On peut donc invoquer la règle Cumulativité-Sortes, pour conclure $\lfloor\Gamma\rfloor \vdash \lfloor M\rfloor : \lfloor T\rfloor$. ☺

Une première application de la projection consiste à ramener la cohérence de la logique engendrée $\mathcal{P}^2$ à celle du système de départ $\mathcal{P}$.

3.4.9 Lemme

- *Si s est* $\mathrm{filter}(\Gamma)$*-cohérente, alors s est Γ-cohérente.*
- *Si s est $\lfloor\Gamma\rfloor$-cohérente, alors $\lceil s\rceil$ est Γ-cohérente.*

– *Si $\mathcal{P}$ est cohérent, alors $\mathcal{P}^2$ est cohérent.*

Démonstration – C'est une conséquence du lemme 3.4.2. Si s n'est pas Γ-cohérente, cela signifie que qu'il existe un terme M tel que $\Gamma, \alpha^1 : s \vdash M : \alpha^1$. D'après le lemme 3.3.16, on a $\text{lvl}(M) = 1$ et donc d'après lemme 3.4.2, on obtient $\text{filter}(\Gamma), \alpha : s \vdash M : \alpha$ et donc s n'est pas $\text{filter}(\Gamma)$-cohérente (et *a fortiori* Γ-cohérente puisque $\text{filter}(\Gamma) \subseteq \Gamma$).
– C'est une conséquence du lemme 3.4.8. En effet, si $\lceil s \rceil$ n'est pas Γ-cohérente cela signifie qu'il existe un terme M tel que $\Gamma, \alpha^2 : \lceil s \rceil \vdash_{\mathcal{P}^2} M : \alpha^2$. D'après le lemme 3.3.16, on a $\text{lvl}(M) = 2$ et donc d'après le lemme 3.4.8, on en déduit que $\lfloor \Gamma \rfloor, \alpha : s \vdash_{\mathcal{P}} \lfloor M \rfloor : \alpha$. Et donc s n'est pas $\lfloor \Gamma \rfloor$-cohérente.
– Si $\mathcal{P}^2$ n'est pas cohérent cela signifie qu'il existe une sorte s telle que s ne soit pas cohérente. On distingue alors deux cas : soit $\text{lvl}(s) = 1$ et on peut appliquer le premier tiret, soit $\text{lvl}(s) = 2$ et on peut appliquer le second. ☺

Nous allons à présent voir les propriétés préservées par la construction qui fait passer de $\mathcal{P}$ à $\mathcal{P}^2$. Nous verrons plus tard (page 186) sur des exemples que cette dernière ne préserve pas les conditions de stabilité.

3.4.10 Lemme
On dispose des implications suivantes :

$\mathcal{P}$	$\models$	PTS	$\Rightarrow$	$\mathcal{P}^2$	$\models$	PTS	(1)
$\mathcal{P}$	$\models$	HOMOGENEOUS	$\Rightarrow$	$\mathcal{P}^2$	$\models$	HOMOGENEOUS	(2)
$\mathcal{P}$	$\models$	NICE-TOPSORT	$\Rightarrow$	$\mathcal{P}^2$	$\models$	NICE-TOPSORT	(3)
$\mathcal{P}$	$\models$	FUNCTIONAL	$\Rightarrow$	$\mathcal{P}^2$	$\models$	FUNCTIONAL	(4)
$\mathcal{P}$	$\models$	PREDICATIVE	$\Rightarrow$	$\mathcal{P}^2$	$\models$	PREDICATIVE	(5)
$\mathcal{P}$	$\models$	WEAKLY-IMPREDICATIVE	$\Rightarrow$	$\mathcal{P}^2$	$\models$	WEAKLY-IMPREDICATIVE	(6)

Démonstration 1. On a bien : $\mathcal{C}_{\mathcal{P}} = \emptyset \quad \Rightarrow \quad \mathcal{C}_{\mathcal{P}^2} = \emptyset$.

2. Supposons $\mathcal{P} \models$ HOMOGENEOUS et soit $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}^2}$. On distingue quatre cas :
   – $\text{lvl}(s_1) = 1$ et $\text{lvl}(s_2) = 1$: Alors par définition de $\mathcal{P}^2$, on a $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}}$ et puisque l'on a $\mathcal{P} \models$ HOMOGENEOUS, on en déduit bien $s_2 = s_3$.
   – $\text{lvl}(s_1) = 2$ et $\text{lvl}(s_2) = 2$: Par définition de $\mathcal{P}^2$, on a l'existence de $(s'_1, s'_2, s'_3) \in \mathcal{R}$ telles que $s_i = \lceil s'_i \rceil$ pour $i = 1, 2, 3$. Or, $\mathcal{P} \models$ HOMOGENEOUS implique que $s'_2 = s'_3$ et donc $\lceil s'_2 \rceil = \lceil s'_3 \rceil$.
   – $\text{lvl}(s_1) = 2$ et $\text{lvl}(s_2) = 1$: Ce cas est impossible par définition de $\mathcal{P}^2$ (on n'a aucune règle de la forme $(\lceil r_1 \rceil, r_2, r_3) \in \mathcal{R}_{\mathcal{P}^2}$ avec $r_1, r_2 \in \mathcal{S}_{\mathcal{P}}$).
   – $\text{lvl}(s_1) = 1$ et $\text{lvl}(s_2) = 2$: Alors par définition de $\mathcal{P}^2$, on a $s_2 = s_3$.

3. Supposons $\mathcal{P} \models$ NICE-TOPSORT et soit $(t, s) \in \mathcal{C}_{\mathcal{P}^2}$. Montrons qu'il existe r telle que $(s, r) \in \mathcal{A}_{\mathcal{P}^2}$. Pour cela, on distingue deux cas :
   – Soit $(t, s) \in \mathcal{C}_{\mathcal{P}}$ et, d'après $\mathcal{P} \models$ NICE-TOPSORT, il existe r telle que $(s, r) \in \mathcal{A}_{\mathcal{P}} \subseteq \mathcal{A}_{\mathcal{P}^2}$.
   – Soit il existe $(t', s') \in \mathcal{C}_{\mathcal{P}}$ avec $t = \lceil t' \rceil$ et $s = \lceil s' \rceil$. Et d'après $\mathcal{P} \models$ NICE-TOPSORT, il existe r' telle que $(s', r') \in \mathcal{A}_{\mathcal{P}} \subseteq \mathcal{A}_{\mathcal{P}^2}$. Et donc par construction de $\mathcal{P}^2$, on a bien $(\lceil s' \rceil, \lceil r' \rceil) \in \mathcal{A}_{\mathcal{P}^2}$. Il suffit donc de prendre $r = \lceil r' \rceil$.
   On a donc bien montré que $\mathcal{P}^2 \models$ NICE-TOPSORT.

4. Supposons $\mathcal{P} \models$ FUNCTIONAL. Montrons que $\mathcal{P}^2 \models$ FUNCTIONAL :
   - Soit $(s_1, s_2) \in \mathcal{A}_{\mathcal{P}^2}$ et $(s_1, r_2) \in \mathcal{A}_{\mathcal{P}^2}$. Alors par construction de $\mathcal{P}^2$, on distingue deux cas :
     - $(s_1, s_2) \in \mathcal{A}_{\mathcal{P}}$: Dans ce cas on a également, $(s_1, r_2) \in \mathcal{A}_{\mathcal{P}}$ (car par construction de $\mathcal{P}^2$, la relation $\mathcal{A}_{\mathcal{P}^2}$ préserve les niveaux). Or, $\mathcal{P} \models$ FUNCTIONAL implique bien que $s_2 = r_2$.
     - Soit il existe $(s'_1, s'_2) \in \mathcal{A}_{\mathcal{P}}$ telles que $s_1 = \lceil s'_1 \rceil$ et $s_2 = \lceil s'_2 \rceil$. Dans ce cas, on a également l'existence de r'_2 tel que $r_2 = \lceil r'_2 \rceil$ et $(s_1, r_2) \in \mathcal{A}_{\mathcal{P}}$. Or, puisque l'on a $\mathcal{P} \models$ FUNCTIONAL, on déduit que $r'_2 = s'_2$ et donc que $r_2 = s_2$
   - Soit $(s_1, s_2, s_3) \in \mathcal{R}$ et $(s_1, s_2, r_3) \in \mathcal{R}$. On distingue quatre cas :
     - $\text{lvl}(s_1) = 1$ et $\text{lvl}(s_2) = 1$: Alors par définition de $\mathcal{P}^2$, on a $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}}$ et $(s_1, s_2, r_3) \in \mathcal{R}_{\mathcal{P}}$ et puisque l'on a $\mathcal{P} \models$ FUNCTIONAL, on en déduit bien $s_2 = r_3$.
     - $\text{lvl}(s_1) = 2$ et $\text{lvl}(s_2) = 2$: Par définition de $\mathcal{P}^2$, on a l'existence de $(s'_1, s'_2, s'_3) \in \mathcal{R}$ et de $(s'_1, s'_2, r'_3) \in \mathcal{R}$ telles que $s_i = \lceil s'_i \rceil$ pour $i = 1, 2, 3$ et $r_3 = \lceil r'_3 \rceil$. Or, $\mathcal{P} \models$ FUNCTIONAL implique que $s'_3 = r'_3$ et donc $r_3 = s_3$.
     - $\text{lvl}(s_1) = 2$ et $\text{lvl}(s_2) = 1$: Ce cas est impossible par définition de $\mathcal{P}^2$.
     - $\text{lvl}(s_1) = 1$ et $\text{lvl}(s_2) = 2$: Alors par définition de $\mathcal{P}^2$, on a $s_2 = s_3$ et $s_2 = r_3$. On en déduit donc bien que $s_3 = r_3$.
5. Pour prouver qu'un CTS est prédicatif, il suffit d'exhiber une relation bien fondée R qui satisfait les règles de la définition 1.3.1. Par construction, on aura $<_{\mathcal{P}^2} \subseteq R$ et donc R bien fondé implique $<_{\mathcal{P}^2}$ bien fondé (et donc $\mathcal{P}^2$ est un système prédicatif). On définit $s \, R \, r$ si l'une des conditions suivantes est vérifiées :
   - (COND1) : $(s <_{\mathcal{P}} r \wedge \text{lvl}(s) = \text{lvl}(r) = 1)$,
   - (COND2) : $(\text{lvl}(s) = \text{lvl}(r) = 2 \wedge s = \lceil s' \rceil \wedge r = \lceil r' \rceil \wedge s' <_{\mathcal{P}} r')$,
   - (COND3) : $(\text{lvl}(s) = 1 \wedge \text{lvl}(r) = 2)$.

   et vérifions que R satisfait bien les règles de la définition 1.3.1 :
   - Axiome : Soit $(s, r) \in \mathcal{A}_{\mathcal{P}^2}$, alors il existe $(s', r') \in \mathcal{A}_{\mathcal{P}}$ tel que l'on ait soit $s = s'$ et $r = r'$ (1), soit $s = \lceil s' \rceil$ et $r = \lceil r' \rceil$ (2). Par définition de $<_{\mathcal{P}}$, on a $s' <_{\mathcal{P}} r'$. On en déduit donc bien $s \, R \, r$ grâce à (COND1) dans le cas (1) et grâce à (COND2) dans le cas (2).
   - Prémisse : Soit $(s, r, t) \in \mathcal{R}_{\mathcal{P}^2}$ et soit $u \, R \, s$. Montrons que $u \, R \, t$. On distingue pour cela quatre cas selon la définition de $\mathcal{R}_{\mathcal{P}^2}$:
     - $(s, r, t) \in \mathcal{R}_{\mathcal{P}}$: Dans ce cas, $\text{lvl}(s) = 1$ et donc $u \, R \, s$ signifie que $\text{lvl}(u) = 1$ et $u <_{\mathcal{P}} s$. Et donc, par définition de $<_{\mathcal{P}}$, on obtient bien $u <_{\mathcal{P}} t$ et donc $u \, R \, t$ d'après (COND1).
     - $(s', r', t') \in \mathcal{R}_{\mathcal{P}}, \lceil s' \rceil = s, \lceil r' \rceil = r, \lceil t' \rceil = t$: Dans ce cas, $\text{lvl}(s) = 2$ et donc $u \, R \, s$ signifie :
       - Soit $\text{lvl}(u) = 2$ et il existe u' tel que $\lceil u' \rceil = u$ et avec $u' <_{\mathcal{P}} s'$. Or, par définition de $<_{\mathcal{P}}$, on a bien $u' <_{\mathcal{P}} t'$ et donc $u \, R \, t$ d'après (COND2).
       - Soit $\text{lvl}(u) = 1$ et on a alors bien $u \, R \, t$ d'après (COND3).
     - $(s, r', t') \in \mathcal{R}_{\mathcal{P}}, \lceil t' \rceil = r = t$ ou $(s, r') \in \mathcal{A}_{\mathcal{P}}, \lceil r' \rceil = t = r$: Dans ce cas $\text{lvl}(s) = 1$ et donc $u \, R \, s$ signifie que $\text{lvl}(u) = 1$. Or $\text{lvl}(t) = 2$. On a donc bien $u \, R \, t$.
   - Conclusion : Soit $(s, r, t) \in \mathcal{R}_{\mathcal{P}^2}$ et soit $u \, R \, r$. Montrons que $u \, R \, t$. Tout comme dans le cas précédent, on distingue pour cela quatre cas selon la définition de $\mathcal{R}_{\mathcal{P}^2}$. Les cas se traitent de manière similaire.
   - Transitivité : Il suffit de vérifier que la relation R est bien transitive.
   - Cumulativité : Pour traiter ce cas, il suffit de montrer que si $s_i \, R \, r_j$ et $(r_j, r_k) \in \mathcal{C}$, alors $s_i \, R \, r_k$. Il suffit pour cela de distinguer les trois significations possibles de $s_i \, R \, r_j$. Dans les cas

(COND1) et (COND2) on se ramène la propriété de cumulativité de $<_{\mathcal{P}}$. Enfin, dans le cas (COND3), on utilise le fait que la relation de cumulativité $\mathcal{C}_{\mathcal{P}^2}$ préserve le niveau des sortes (pour tout $(s,r) \in \mathcal{C}_{\mathcal{P}^2}$, $\mathrm{lvl}(s) = \mathrm{lvl}(r)$).

6. Soit $S \subseteq \mathcal{S}_{\mathcal{P}}$ un ensemble qui vérifie les propriétés de la définition 1.4.1 relative à la faible imprédicativité de $\mathcal{P}$. Alors on pose $S^2 = \{s, \lceil s \rceil | s \in S\}$. Montrons que S^2 est un témoin de la faible imprédicativité de $\mathcal{P}^2$. On vérifie facilement que S^2 satisfait les propriétés suivantes :
   - S^2 est un ensemble de sortes minimales pour $\mathcal{A}_{\mathcal{P}^2}$,
   - S^2 est un ensemble stable vers le bas pour $\mathcal{C}_{\mathcal{P}^2}$,
   - Si $(s,r,t) \in \mathcal{R}_{\mathcal{P}^2}$, alors $r \in S \Leftrightarrow t \in S$.

   Il nous reste à démontrer que le système $\underline{\mathcal{P}^2}$, défini par les paramètres ci-dessous, est prédicatif.

$$\begin{array}{rcl} \mathcal{S}_{\underline{\mathcal{P}^2}} & = & \mathcal{S}_{\mathcal{P}^2} \\ \mathcal{A}_{\underline{\mathcal{P}^2}} & = & \mathcal{A}_{\mathcal{P}^2} \\ \mathcal{R}_{\underline{\mathcal{P}^2}} & = & \{(s_1,s_2,s_3) \in \mathcal{R}_{\mathcal{P}^2} \,|s_2 \notin S^2 \wedge s_3 \ \notin S^2\} \\ \mathcal{C}_{\underline{\mathcal{P}^2}} & = & \mathcal{C}_{\mathcal{P}^2} \end{array}$$

   Pour cela il suffit de remarquer que $(\underline{\mathcal{P}})^2 = \underline{\mathcal{P}^2}$. En effet, on a :

$$\begin{array}{rcl} \mathcal{S}_{(\underline{\mathcal{P}})^2} & = & \{s, \lceil s \rceil | s \in \mathcal{S}_{\underline{\mathcal{P}}}\} \\ & = & \{s, \lceil s \rceil | s \in \mathcal{S}_{\mathcal{P}}\} \\ & = & \mathcal{S}_{\mathcal{P}^2} \\ & = & \mathcal{S}_{\underline{\mathcal{P}^2}} \end{array}$$

   et :

$$\begin{array}{rcl} \mathcal{A}_{(\underline{\mathcal{P}})^2} & = & \{(s,r),(\lceil s \rceil, \lceil r \rceil) | (s,r) \in \mathcal{A}_{\underline{\mathcal{P}}}\} \\ & = & \{(s,r),(\lceil s \rceil, \lceil r \rceil) | (s,r) \in \mathcal{A}_{\mathcal{P}}\} \\ & = & \mathcal{A}_{\mathcal{P}^2} \\ & = & \mathcal{A}_{\underline{\mathcal{P}^2}} \end{array} \qquad \begin{array}{rcl} \mathcal{C}_{(\underline{\mathcal{P}})^2} & = & \{(s,r),(\lceil s \rceil, \lceil r \rceil) | (s,r) \in \mathcal{C}_{\underline{\mathcal{P}}}\} \\ & = & \{(s,r),(\lceil s \rceil, \lceil r \rceil) | (s,r) \in \mathcal{C}_{\mathcal{P}}\} \\ & = & \mathcal{C}_{\mathcal{P}^2} \\ & = & \mathcal{C}_{\underline{\mathcal{P}^2}} \end{array}$$

   Il faut encore démontrer que $\mathcal{R}_{(\underline{\mathcal{P}})^2} = \mathcal{R}_{\underline{\mathcal{P}^2}}$.
   - $\mathcal{R}_{(\underline{\mathcal{P}})^2} \subseteq \mathcal{R}_{\underline{\mathcal{P}^2}}$: Soit $(s,r,t) \in \mathcal{R}_{(\underline{\mathcal{P}})^2}$, par définition deux cas sont possibles :
     - Soit $r = t$ et il existe t' tel que $\lceil t' \rceil = t$ et $(s,t') \in \mathcal{A}_{\underline{\mathcal{P}}} = \mathcal{A}_{\mathcal{P}}$. On a donc $(s,t,t) \in \mathcal{R}_{\mathcal{P}^2}$ or comme S^2 est un ensemble de sortes minimales pour $\mathcal{A}_{\mathcal{P}^2}$ on en déduit que $t \notin S^2$ et donc que $(s,t,t) \in \mathcal{R}_{\underline{\mathcal{P}^2}}$.
   - Soit il existe $(s', r', t') \in \mathcal{R}_{\underline{\mathcal{P}}}$ tel que l'on est dans l'une des situations suivantes :
     - (CAS1) : $s' = s$, $r' = r$, $t' = t$,
     - (CAS2) : $\lceil s' \rceil = s$, $\lceil r' \rceil = r$, $\lceil t' \rceil = t$,
     - (CAS3) : $s' = s$, $\lceil t' \rceil = t = r$.

Les équivalences suivantes sont des conséquences immédiates des définitions :

$$\begin{aligned}
& (s', r', t') \in \mathcal{R}_{\underline{\mathcal{P}}} \\
\Leftrightarrow\ & (s', r', t') \in \mathcal{R}_{\mathcal{P}} \wedge r' \notin S \wedge t' \notin S \\
\Leftrightarrow\ & (s', r', t') \in \mathcal{R}_{\mathcal{P}^2} \wedge r' \notin S^2 \wedge t' \notin S^2 \wedge \mathrm{lvl}(s') = \mathrm{lvl}(r') = \mathrm{lvl}(t') = 1 \\
\Leftrightarrow\ & (s', r', t') \in \mathcal{R}_{\underline{\mathcal{P}^2}} \wedge \mathrm{lvl}(s') = \mathrm{lvl}(r') = \mathrm{lvl}(t') = 1 \qquad (1) \\
\Leftrightarrow\ & (\lceil s' \rceil, \lceil r' \rceil, \lceil t' \rceil) \in \mathcal{R}_{\mathcal{P}^2} \wedge \lceil r' \rceil \notin S^2 \wedge \lceil t' \rceil \notin S^2 \wedge \mathrm{lvl}(s') = \mathrm{lvl}(r') = \mathrm{lvl}(t') = 1 \\
\Leftrightarrow\ & (\lceil s' \rceil, \lceil r' \rceil, \lceil t' \rceil) \in \mathcal{R}_{\underline{\mathcal{P}^2}} \wedge \mathrm{lvl}(s') = \mathrm{lvl}(r') = \mathrm{lvl}(t') = 1 \qquad (2)
\end{aligned}$$

Ainsi on montre $(s, r, t) \in \mathcal{R}_{\underline{\mathcal{P}^2}}$ dans le (CAS1) et le (CAS2) grâce au sens direct des équivalences (1) et (2).

$$\begin{aligned}
& (s', r', t') \in \mathcal{R}_{\underline{\mathcal{P}}} \\
\Leftrightarrow\ & (s', r', t') \in \mathcal{R}_{\mathcal{P}} \wedge r' \notin S \wedge t' \notin S \\
\Rightarrow\ & (s', \lceil t' \rceil, \lceil t' \rceil) \in \mathcal{R}_{\mathcal{P}^2} \wedge \lceil t' \rceil \notin S^2 \\
\Leftrightarrow\ & (s', \lceil t' \rceil, \lceil t' \rceil) \in \mathcal{R}_{\underline{\mathcal{P}^2}} \qquad (3)
\end{aligned}$$

– $\mathcal{R}_{(\underline{\mathcal{P}})^2} \supseteq \mathcal{R}_{\underline{\mathcal{P}^2}}$: Soit $(s, r, t) \in \mathcal{R}_{\underline{\mathcal{P}^2}}$. Par définition de $\mathcal{R}_{\underline{\mathcal{P}^2}}$, cela signifie que $(s, r, t) \in \mathcal{R}_{\mathcal{P}^2}$ et $r \notin S^2 \wedge t \notin S^2$. Puis, par définition de $\mathcal{R}_{\mathcal{P}^2}$, quatre cas sont possibles :
  – Soit $(s, r, t) \in \mathcal{R}_{\mathcal{P}}$. Dans ce cas, on a $s, r, t \in \mathcal{S}_{\mathcal{P}}$ et, par définition de S^2, $r \notin S$ et $t \notin S$. On en déduit que $(s, r, t) \in \mathcal{R}_{\underline{\mathcal{P}}} \subseteq \mathcal{R}_{(\underline{\mathcal{P}})^2}$.
  – Soit il existe $(s', r', t') \in \mathcal{R}_{\mathcal{P}}$ avec $\lceil s' \rceil = s$, $\lceil r' \rceil = r$, et $\lceil t' \rceil = t$. Or, par définition de S^2, on a $r' \notin S$ et $t' \notin S$, on en déduit donc que $(s', r', t') \in \mathcal{R}_{\underline{\mathcal{P}}}$ et donc $(\lceil s' \rceil, \lceil r' \rceil, \lceil t' \rceil) \in \mathcal{R}_{(\underline{\mathcal{P}})^2}$.
  – Soit il existe r' et t' telles que $(s, r', t') \in \mathcal{R}_{\mathcal{P}}$ avec $\lceil t' \rceil = r = t$. Or on a $t = \lceil t' \rceil \notin S^2$ et donc $t' \notin S$. Or S vérifie bien que $t' \in S \Leftrightarrow r' \in S$. On en déduit donc que $r' \notin S$ et donc que $(s, r', t') \in \mathcal{R}_{\underline{\mathcal{P}}}$ puis que $(s, t, t) \in \mathcal{R}_{(\underline{\mathcal{P}})^2}$.
  – Soit il existe r' telle que $(s, r') \in \mathcal{A}_{\mathcal{P}}$ avec $\lceil r' \rceil = t = r$. On $\mathcal{A}_{\mathcal{P}} = \mathcal{A}_{\underline{\mathcal{P}}}$, on en déduit donc que $(s, \lceil r' \rceil, \lceil r' \rceil) \in \mathcal{R}_{(\underline{\mathcal{P}})^2}$.

On conclut enfin en appliquant le cas précédent pour montrer, à partir de la prédicativité de $\underline{\mathcal{P}}$, que $(\underline{\mathcal{P}})^2$ est prédicatif.

☺

De la forte normalisation de $\mathcal{P}$ à la forte normalisation de $\mathcal{P}^2$. Dans ce paragraphe, nous allons démontrer que la construction $\mathcal{P} \mapsto \mathcal{P}^2$ préserve la forte normalisation des systèmes. En pratique, la preuve de cette propriété ne s'avère pas très utile puisque l'on a vu précédemment que la construction préserve la propriété WEAKLY-IMPREDICATIVE ; cette dernière est satisfaite par la grande majorité des systèmes étudiés et implique la forte normalisation. Néanmoins, nous pensons que la preuve directe que l'on propose ici offre une perspective intéressante sur la structure du système $\mathcal{P}^2$. Elle est basée sur l'observation selon laquelle, si un terme A est dans $\mathcal{P}^2$ et s'il n'est pas normalisable alors on est dans l'une des situations suivantes :

– Soit il existe un sous-terme de A qui est de premier niveau et qui n'est pas normalisable.

– Soit A est un terme de second niveau et $\lfloor A \rfloor$ n'est pas normalisable.

Or, d'après le lemme de séparation (lemme 3.4.2), à la fois le sous-terme et $\lfloor A \rfloor$ sont typables dans $\mathcal{P}$. Donc chacune de ces situations contredit la propriété de forte normalisation de $\mathcal{P}$.

3.4.11 Théorème
Si $\mathcal{P} \models$ NORMALISING, *alors* $\mathcal{P}^2 \models$ NORMALISING.

Démonstration On commence par remarquer les faits suivants :
Soit A un terme bien typé (il existe Γ et T tel que $\Gamma \vdash A : T$).

1. Si $A \twoheadrightarrow^2 A'$ ou $A \twoheadrightarrow^{3/2} A'$, alors A est un terme de second niveau (dans le cas contraire, le lemme 3.4.2 montre que A est typable dans $\mathcal{P}$, il ne peut donc contenir de sous-terme de niveau 2) .
2. Si $A \twoheadrightarrow^2 A'$, alors $\lfloor A \rfloor \twoheadrightarrow \lfloor A' \rfloor$; car la projection conserve les rédexs réduits par $\twoheadrightarrow^2$ (lemme 3.4.7).
3. Si A est un terme de second niveau, alors
$$A(\twoheadrightarrow^1 \cup \twoheadrightarrow^{3/2})A' \text{ implique } \lfloor A \rfloor = \lfloor A' \rfloor$$
car la projection efface tous les rédexs réduits par $\twoheadrightarrow^1$ et par $\twoheadrightarrow^{3/2}$ (lemme 3.4.7).
4. Si $A \twoheadrightarrow^{3/2} A'$, alors le nombre de rédex que l'on peut réduire grâce à $\twoheadrightarrow^{3/2}$ (on appelle ces rédexs des *rédexs d'interaction*) est réduit de exactement 1 dans A'.
En effet, un rédex d'interaction est l'application d'un terme de second niveau à un terme de premier niveau. L'argument qui peut être éventuellement dupliqué ne peut donc pas contenir de rédexs d'interaction (puisque ce dernier est un terme de premier niveau). De plus, pour la même raison, l'argument ne peut être une abstraction qui engendrerait un rédex d'interaction puisqu'il est de premier niveau. On en conclut donc bien que $\twoheadrightarrow^{3/2}$ ne peut créer ni dupliquer des rédexs d'interaction.
5. Le nombre de rédexs d'interaction est invariant par $\twoheadrightarrow^1$ puisque les rédexs d'interaction sont des termes de second niveau.

Montrons, la contraposée du lemme : supposons qu'il existe
$$A \twoheadrightarrow A_1 \twoheadrightarrow A_2 \twoheadrightarrow ... \twoheadrightarrow A_n \twoheadrightarrow ...$$
une suite infinie de termes obtenue par réductions successives de A avec $\Gamma \vdash_{\mathcal{P}^2} A : T$. Si $\mathrm{lvl}(A) = 1$, d'après le lemme 3.4.2, $\mathrm{filter}(\Gamma) \vdash_{\mathcal{P}} A : T$ et donc la suite infinie contredit la forte normalisation de $\mathcal{P}$. Traitons à présent le cas où $\mathrm{lvl}(A) = 2$.

Alors nous sommes dans l'une des deux situations suivantes :

– Soit il est possible d'extraire une sous-séquence $(A_{n_i})_{i\in\mathbb{N}}$ telle que $A_{n_i} \left((\twoheadrightarrow^1 \cup \twoheadrightarrow^{3/2})^* \circ \twoheadrightarrow^2\right) A_{n_{i+1}}$ (où $\cdot \mapsto \cdot^*$ désigne la clôture transitive de la relation et $\circ$ la composition des relations) pour tout $i \in \mathbb{N}$;
– Soit il existe N tel que pour tout $n \geq N$, $A_n(\twoheadrightarrow^1 \cup \twoheadrightarrow^{3/2})A_{n+1}$ ou plus prosaïquement : $\twoheadrightarrow^2$ n'est pas utilisé dans la suite à partir du rang N.

Dans le premier cas, puisque $A(\twoheadrightarrow^1 \cup \twoheadrightarrow^{3/2})^* \circ \twoheadrightarrow^2 A'$ implique $\lfloor A \rfloor \twoheadrightarrow \lfloor A' \rfloor$, on peut construire une suite infinie $(\lfloor A_{n_i} \rfloor)_{i\in\mathbb{N}}$ de termes en relation pour $\twoheadrightarrow^1$.

Dans le second cas, puisque $\twoheadrightarrow^{3/2}$ fait strictement décroître le nombre de rédexs d'intéraction et que la réduction $\twoheadrightarrow^1$ ne change pas ce dernier, on montre qu'il existe un entier $M \geq N$ tel que pour tout $n \geq M$, $A_n \twoheadrightarrow^1 A_{n+1}$.

On définit, par induction sur la structure d'un terme M de second niveau, la liste $\mathrm{SS}(M)$ des sous-termes "superficiels" de premier niveau de M. Puisque M est un terme de second niveau, il n'y a donc que les cas suivants à traiter :

$$\begin{aligned}
\mathrm{SS}(x^2) &= [] \\
\mathrm{SS}(\lceil s \rceil) &= [] \\
\mathrm{SS}(M\,N) &= \begin{cases} \mathrm{SS}(M) +\!\!+ \mathrm{SS}(N) & \text{si } \mathrm{lvl}(N) = 2 \\ N :: \mathrm{SS}(M) & \text{si } \mathrm{lvl}(N) = 1 \end{cases} \\
\mathrm{SS}(\lambda x : M.N) &= \mathrm{SS}(\forall x : M.N) = \begin{cases} \mathrm{SS}(M) +\!\!+ \mathrm{SS}(N) & \text{si } \mathrm{lvl}(M) = 2 \\ M :: \mathrm{SS}(N) & \text{si } \mathrm{lvl}(M) = 1 \end{cases}
\end{aligned}$$

où $+\!\!+$, $::$, et $[]$ désignent respectivement la concaténation des listes, l'ajout d'un élément en tête et la liste vide.

Soit $[t_1, \ldots, t_k]$ la liste générée par $\mathrm{SS}(M)$, on montre facilement que :

1. (PROP1) : Si $M \twoheadrightarrow^1 M'$ il existe $1 \leq i \leq k$, $\mathrm{SS}(M') = [t_1, \ldots, t_{i-1}, t'_i, t_{i+1} \ldots t_k]$ avec $t_i \twoheadrightarrow t'_i$ (on rappelle que pour les termes de premier niveau $t_i \twoheadrightarrow t'_i$ est équivalent à $t_i \twoheadrightarrow^1 t'_i$). La preuve se fait facilement par induction sur les règles définissant $M \twoheadrightarrow^1 M'$.
2. (PROP2) : Pour tout $1 \leq i \leq k$, il existe T_i tel que $\mathrm{filter}(\Gamma) \vdash_{\mathcal{P}} t_i : T_i$ (par induction sur la dérivation de $\Gamma \vdash_{\mathcal{P}^2} A : T$).

Soit $[t_1^M, \ldots, t_k^M] = \mathrm{SS}(A_M)$. Or, puisque $A_M \twoheadrightarrow A_{M+1} \twoheadrightarrow \cdots$, on sait, grâce à (PROP1), qu'il existe k suites $t_1^M, t^{M+1}, \ldots, t_2^M, t^{M+1}, \ldots, \ldots, t_k^M, t_k^{M+1}, \ldots$ et une suite $i_M, i_{M+1}, \ldots$ d'entiers entre $1 \leq i_n \leq k$ telles que $t_i^n = t_i^{n+1}$ si $i \neq i_n$ et $t_{i_n}^n \twoheadrightarrow t_{i_n}^{n+1}$ pour tout $1 \leq i \leq k$ et tout $n \geq M$. De plus, (PROP2) nous montre que t_i^n est typable dans $\mathrm{filter}(\Gamma)$ pour tout $1 \leq i \leq k$ et tout $n \geq M$. Or, par un argument de cardinalité immédiat on peut extraire une sous-suite constante de $i_M, i_{M+1}, \ldots$, c'est-à-dire, il existe $1 \leq r \leq n$ et une fonction strictement croissante $\varphi : \mathbb{N} \longrightarrow \mathbb{N}$ telle que $\varphi(n) \geq M$ et $i_{\varphi(n)} = r$. On a alors bien $t_r^{\varphi(n)} \twoheadrightarrow t_r^{\varphi(n+1)}$ pour tout n, ce qui contredit la forte normalisation de $\mathcal{P}$.☺

Comportement de $\cdot \mapsto \cdot^2$ vis-à-vis de la clôture par cumulativité. La construction qui fait passer de $\mathcal{P}$ à $\mathcal{P}^2$ n'a néanmoins pas de bonne propriété vis-à-vis de la clôture par cumulativité. En effet, la propriété suivante n'est pas vérifiée :

$$\text{Si } \overline{\mathcal{P}} = \overline{\mathcal{P}'}, \text{ alors } \overline{\mathcal{P}^2} = \overline{\mathcal{P}'^2} \qquad (*)$$

Or elle implique (en prenant $\mathcal{P}' = \overline{\mathcal{P}}$) que $\overline{(\overline{\mathcal{P}}^2)} = \overline{\mathcal{P}^2}$. Nous allons voir un exemple de système $\mathcal{P}$ qui ne vérifie pas $\overline{(\overline{\mathcal{P}}^2)} = \overline{\mathcal{P}^2}$ (ce qui prouvera donc que (*) n'est pas démontrable). Le système que nous allons introduire n'admet aucun séquent typable puisqu'il ne contient pas d'axiome. Il est défini par les paramètres suivants :

$$\mathcal{S}_{\mathcal{P}} = \{\star, \star'\} \quad \mathcal{A}_{\mathcal{P}} = \emptyset \quad \mathcal{R}_{\mathcal{P}} = \{(\star, \star, \star)\} \quad \mathcal{C}_{\mathcal{P}} = \{(\star, \star')\}$$

Nous pouvons calculer sa clôture $\overline{\mathcal{P}}$ par cumulativité ainsi que le système $\mathcal{P}^2$:

$$\begin{array}{rcl} \mathcal{S}_{\overline{\mathcal{P}}} & = & \{\star, \star' \} \\ \mathcal{A}_{\overline{\mathcal{P}}} & = & \emptyset \\ \mathcal{R}_{\overline{\mathcal{P}}} & = & \{(\star,\star,\star),(\star,\star,\star')\} \\ \mathcal{C}_{\overline{\mathcal{P}}} & = & \{(\star,\star')\} \end{array} \qquad \begin{array}{rcl} \mathcal{S}_{\mathcal{P}^2} & = & \{\star, \lceil\star\rceil, \star', \lceil\star'\rceil \} \\ \mathcal{A}_{\mathcal{P}^2} & = & \emptyset \\ \mathcal{R}_{\mathcal{P}^2} & = & \{(\star,\star,\star),(\lceil\star\rceil,\lceil\star\rceil,\lceil\star\rceil),(\star,\lceil\star\rceil,\lceil\star\rceil)\} \\ \mathcal{C}_{\mathcal{P}^2} & = & \{(\star,\star'),(\lceil\star\rceil,\lceil\star'\rceil)\} \end{array}$$

On peut calculer maintenant la clôture par cumulativité de $\mathcal{P}^2$ (on ne calcule que le paramètre $\mathcal{R}$ les autres étant égaux à ceux de $\mathcal{P}^2$) :

$$\mathcal{R}_{\overline{\mathcal{P}^2}} = \left\{ \begin{array}{lcl} (\star,\star,\star) & , & (\star,\star,\star'), \\ (\lceil\star\rceil,\lceil\star\rceil,\lceil\star\rceil) & , & (\lceil\star\rceil,\lceil\star\rceil,\lceil\star'\rceil), \\ (\star,\lceil\star\rceil,\lceil\star\rceil) & , & (\star,\lceil\star\rceil,\lceil\star'\rceil) \end{array} \right\}$$

De même, on calcule $\left(\overline{\mathcal{P}}\right)^2$, puis sa clôture par cumulativité $\overline{\left(\overline{\mathcal{P}}\right)^2}$ (les autres paramètres sont également égaux à ceux de $\mathcal{P}^2$) :

$$\mathcal{R}_{\left(\overline{\mathcal{P}}\right)^2} = \left\{ \begin{array}{lcl} (\star,\star,\star) & , & (\star,\star,\star'), \\ (\lceil\star\rceil,\lceil\star\rceil,\lceil\star\rceil) & , & (\lceil\star\rceil,\lceil\star\rceil,\lceil\star'\rceil), \\ (\star,\lceil\star\rceil,\lceil\star\rceil) & , & (\star,\lceil\star'\rceil,\lceil\star'\rceil) \end{array} \right\}$$

et :

$$\mathcal{R}_{\overline{\left(\overline{\mathcal{P}}\right)^2}} = \left\{ \begin{array}{lcl} (\star,\star,\star) & , & (\star,\star,\star'), \\ (\lceil\star\rceil,\lceil\star\rceil,\lceil\star\rceil) & , & (\lceil\star\rceil,\lceil\star\rceil,\lceil\star'\rceil), \\ (\star,\lceil\star\rceil,\lceil\star\rceil) & , & (\star,\lceil\star'\rceil,\lceil\star'\rceil), \\ & & (\star,\lceil\star\rceil,\lceil\star'\rceil) \end{array} \right\}$$

On en déduit donc bien que $\overline{(\overline{\mathcal{P}^2})} \neq \overline{\mathcal{P}^2}$.

Préservation des propriétés de stabilité. La construction $\cdot \mapsto \cdot^2$ ne préserve pas les propriétés de stabilité. Ainsi dans ce paragraphe nous allons construire trois systèmes $\mathcal{P}$ qui vérifient $\mathcal{P} \models$ MINLOC et $\mathcal{P}^2 \not\models$ MINLOC. Ces trois contre-exemples illustrent trois manières différentes de générer des systèmes violant la propriété MINLOC. Le premier ne disposera d'aucune règle (*i.e.* $\mathcal{R}_\mathcal{P} = \emptyset$), le second d'aucun axiome (*i.e.* $\mathcal{A}_\mathcal{P} = \emptyset$) et enfin le dernier disposera d'exactement une règle et un axiome.

Le premier contre-exemple exploite le fait que la construction fabrique une règle de la forme $(s, \lceil t\rceil, \lceil t\rceil) \in \mathcal{R}_{\mathcal{P}^2}$ à partir d'une règle de la forme $(s, r, t) \in \mathcal{R}_\mathcal{P}$. On remarque donc que des sortes qui sont en "troisième position" d'une règle dans $\mathcal{P}$ (ici t) apparaîtront en "seconde position" d'une règle dans $\mathcal{P}^2$. Cette situation peut donc engendrer de nouvelles contraintes à vérifier pour satisfaire MINLOC-RULES qui n'ont aucune raison d'être vérifiée *a priori* :

$$\begin{array}{rcl} \mathcal{S}_\mathcal{P} & = & \{\star_1,\star_2,\star_2',\star_3,\star_3',\star\ \} \\ \mathcal{A}_\mathcal{P} & = & \emptyset \\ \mathcal{R}_\mathcal{P} & = & \{(\star_1,\star_2,\star_3),(\star_1,\star_2',\star_3')\} \\ \mathcal{C}_\mathcal{P} & = & \{(\star,\star_3),(\star,\star_3')\} \end{array} \qquad \begin{array}{rcl} \mathcal{S}_{\mathcal{P}^2} & = & \{\ \star_1,\star_2,\star_2',\star_3,\star_3',\star, \\ & & \ \lceil\star_1\rceil,\lceil\star_2\rceil,\lceil\star_2'\rceil,\lceil\star_3\rceil,\lceil\star_3'\rceil,\lceil\star\rceil\ \} \\ \mathcal{A}_{\mathcal{P}^2} & = & \emptyset \\ \mathcal{R}_{\mathcal{P}^2} & = & \{(\star_1,\star_2,\star_3),(\lceil\star_1\rceil,\lceil\star_2\rceil,\lceil\star_3\rceil),(\star_1,\lceil\star_3\rceil,\lceil\star_3\rceil)\} \\ & \cup & \{(\star_1,\star_2',\star_3'),(\lceil\star_1\rceil,\lceil\star_2'\rceil,\lceil\star_3'\rceil),(\star_1,\lceil\star_3'\rceil,\lceil\star_3'\rceil)\} \\ \mathcal{C}_{\mathcal{P}^2} & = & \{(\star,\star_3),(\star,\star_3'),(\lceil\star\rceil,\lceil\star_3\rceil),(\lceil\star\rceil,\lceil\star_3'\rceil)\} \end{array}$$

Le système $\mathcal{P}$ satisfait trivialement MINLOC-AXIOMS car $\mathcal{A}_\mathcal{P} = \emptyset$ et MINLOC-RULES est vérifiée car si

$(s_1, s_2, s_3) \in \mathcal{R}_\mathcal{P}$ et $(t_1, t_2, t_3) \in \mathcal{R}_\mathcal{P}$ avec $r_1 \preccurlyeq s_1$, $r_1 \preccurlyeq t_1$, $r_2 \preccurlyeq s_2$ et $r_2 \preccurlyeq t_2$ alors nécessairement $r_1 = s_1 = t_1 = \star_1$ et $r_2 = s_2 = \star_2$ ou $r_2 = s_2 = \star'_2$ et donc on a bien $(r_1, r_2, s_3) \in \overline{\mathcal{R}_\mathcal{P}}$.

Le système $\mathcal{P}^2$ réfute bien la propriété MINLOC : on a $(\star_1, \lceil\star_3\rceil, \lceil\star_3\rceil), (\star_1, \lceil\star'_3\rceil, \lceil\star'_3\rceil) \in \mathcal{R}_{\mathcal{P}^2}$ et $\lceil\star\rceil \preccurlyeq \lceil\star_3\rceil$ et $\lceil\star\rceil \preccurlyeq \lceil\star'_3\rceil$ or il n'existe pas de sorte s telle que $(\star_1, \lceil\star\rceil, s) \in \overline{\mathcal{R}_{\mathcal{P}^2}}$ ($\lceil\star\rceil$ est la seule sorte vérifiant $\lceil\star\rceil \preccurlyeq \lceil\star_3\rceil$ et $\lceil\star\rceil \preccurlyeq \lceil\star'_3\rceil$ et on n'a pas $(\star_1, \lceil\star\rceil, \lceil\star\rceil) \in \mathcal{R}_{\mathcal{P}^2}$.

On pourrait pallier ce problème en imposant la condition HOMOGENEOUS, mais ce n'est pas la seule situation où la propriété MINLOC est mise en défaut : le système ci-dessous réfute lui aussi la préservation de MINLOC par la construction $\cdot \mapsto \cdot^2$.

$$\begin{aligned}
\mathcal{S}_\mathcal{P} &= \{\star, \square_1, \square_2, \square, \square'\} \\
\mathcal{A}_\mathcal{P} &= \{(\star, \square_1), (\star, \square_2), (\star, \square)\} \\
\mathcal{R}_\mathcal{P} &= \emptyset \\
\mathcal{C}_\mathcal{P} &= \{(\square, \square_1), (\square, \square_2), (\square', \square_1), (\square', \square_2)\}
\end{aligned}$$

Nous avons bien $\mathcal{P} \models$ MINLOC. En effet, MINLOC-RULES est trivialement vérifiée puisque $\mathcal{R}_\mathcal{P}$ est vide et vérifie facilement MINLOC-AXIOMS : soit $(s_1, r_1), (s_2, r_2) \in \mathcal{A}_\mathcal{P}$ avec $s \preccurlyeq s_1$ et $s \preccurlyeq s_2$, il existe r telle que $(s, r) \in \overline{\mathcal{A}_\mathcal{P}}$ ($(\star, \square) \in \mathcal{A}_\mathcal{P}$). La construction $\cdot \mapsto \cdot^2$ va générer des règles dans $\mathcal{R}_{\mathcal{P}^2}$ à partir des axiomes $\mathcal{A}_\mathcal{P}$ et ces dernières mettrons en défaut la propriété MINLOC :

$$\begin{aligned}
\mathcal{S}_{\mathcal{P}^2} &= \{\star, \square_1, \square_2, \square, \square'\} \\
&\cup \{\lceil\star_1\rceil, \lceil\star_2\rceil, \lceil\star\rceil, \lceil\square_1\rceil, \lceil\square_2\rceil, \lceil\square\rceil, \lceil\square'\rceil\} \\
\mathcal{A}_{\mathcal{P}^2} &= \{(\star, \square_1), (\star, \square_2), (\star, \square)\} \\
&\cup \{(\lceil\star\rceil, \lceil\square_1\rceil), (\lceil\star\rceil, \lceil\square_2\rceil), (\lceil\star\rceil, \lceil\square\rceil)\} \\
\mathcal{R}_{\mathcal{P}^2} &= \{(\star, \lceil\square_1\rceil, \lceil\square_1\rceil), (\star, \lceil\square_2\rceil, \lceil\square_2\rceil), (\star, \lceil\square\rceil, \lceil\square\rceil)\} \\
\mathcal{C}_{\mathcal{P}^2} &= \{(\square, \square_1), (\square, \square_2), (\square', \square_1), (\square', \square_2)\} \\
&\cup \{(\lceil\square\rceil, \lceil\square_1\rceil), (\lceil\square\rceil, \lceil\square_2\rceil), (\lceil\square'\rceil, \lceil\square_1\rceil), (\lceil\square'\rceil, \lceil\square_2\rceil)\}
\end{aligned}$$

En effet, nous avons bien $\mathcal{P}^2 \not\models$ MINLOC puisque $(\star, \lceil\square_1\rceil, \lceil\square_1\rceil), (\star, \lceil\square_2\rceil, \lceil\square_2\rceil) \in \mathcal{R}_{\mathcal{P}^2}$ avec $\lceil\square'\rceil \preccurlyeq \lceil\square_1\rceil$ et $\lceil\square'\rceil \preccurlyeq \lceil\square_2\rceil$, mais nous n'avons pas $(\star, \lceil\square'\rceil, \lceil\square'\rceil) \in \overline{\mathcal{R}_{\mathcal{P}^2}}$.

On pourrait vouloir chercher une condition raisonnable à imposer sur les axiomes pour que la construction préserve la propriété MINLOC, mais comme nous allons le voir dans le dernier contre-exemple, il faudrait que cette dernière porte également sur les règles. En effet, le système ci-dessous montre qu'une mauvaise "interaction" entre les axiomes et les règles peut également donner lieu à un système qui ne préserve pas MINLOC :

$$\begin{aligned}
\mathcal{S}_\mathcal{P} &= \{\star, \square_1, \square_2, \square\} \\
\mathcal{A}_\mathcal{P} &= \{(\star, \square_1)\} \\
\mathcal{R}_\mathcal{P} &= \{(\star, \square_2, \square_2)\} \\
\mathcal{C}_\mathcal{P} &= \{(\square, \square_1), (\square, \square_2)\}
\end{aligned}$$

Nous avons bien $\mathcal{P} \models$ MINLOC :

– MINLOC-AXIOMS : Trivial car $(\star, \Box_1)$ est le seul axiome et $\star$ est minimal pour $\mathcal{C}_\mathcal{P}$.
– MINLOC-RULES : Il n'y a qu'une seule règle, il suffit donc de vérifier que $s \preccurlyeq \star$ et $r \preccurlyeq \Box_2$ alors il existe r' telle que $r' \preccurlyeq \Box_2$ et $(s, r, r') \in \overline{\mathcal{R}_\mathcal{P}}$. Or la relation de cumulativité impose que $s = \star$ et $r = \Box$ ou $r = \Box_2$ et dans les deux cas, on a $r' = \Box_2$ qui convient : $(\star, \Box_2, \Box_2) \in \mathcal{R}_\mathcal{P} \subseteq \overline{\mathcal{R}_\mathcal{P}}$ et $(\star, \Box, \Box_2) \in \overline{\mathcal{R}_\mathcal{P}}$.

Calculons maintenant $\mathcal{P}^2$:

$$\begin{aligned}
\mathcal{S}_{\mathcal{P}^2} &= \{\star, \Box_1, \Box_2, \Box\} \quad \cup \quad \{\lceil\star\rceil, \lceil\Box_1\rceil, \lceil\Box_2\rceil, \lceil\Box\rceil\} \\
\mathcal{A}_\mathcal{P} &= \{(\star, \Box_1), (\lceil\star\rceil, \lceil\Box_1\rceil)\} \\
\mathcal{R}_{\mathcal{P}^2} &= \{(\star, \Box_2, \Box_2),\ (\lceil\star\rceil, \lceil\Box_2\rceil, \lceil\Box_2\rceil), (\star, \lceil\Box_2\rceil, \lceil\Box_2\rceil), (\star, \lceil\Box_1\rceil, \lceil\Box_1\rceil)\} \\
\mathcal{C}_{\mathcal{P}^2} &= \{(\Box, \Box_2), (\Box, \Box_1), (\lceil\Box\rceil, \lceil\Box_2\rceil), (\lceil\Box\rceil, \lceil\Box_1\rceil)\}
\end{aligned}$$

Nous avons bien $\mathcal{P}^2 \not\models$ MINLOC puisque $(\star, \lceil\Box_1\rceil, \lceil\Box_1\rceil), (\star, \lceil\Box_2\rceil, \lceil\Box_2\rceil) \in \mathcal{R}_{\mathcal{P}^2}$ avec $\lceil\Box\rceil \preccurlyeq \lceil\Box_1\rceil$, $\lceil\Box\rceil \preccurlyeq \lceil\Box_2\rceil$ et il n'existe pas de sorte s telle que $(\star, \lceil\Box\rceil, s) \in \overline{\mathcal{R}_{\mathcal{P}^2}}$ avec $s \preccurlyeq \lceil\Box_1\rceil$ et $s \preccurlyeq \lceil\Box_2\rceil$.

On remarque donc que les conditions à imposer sur le système de départ pour que la construction préserve la propriété MINLOC sont particulièrement ad-hoc et n'aident pas vraiment à la compréhension des liens entre notre construction et la relation de cumulativité. Nous pensons que ceci est dû au fait que la classe des CTS contient beaucoup de systèmes "pathologiques". C'est pourquoi dans la section suivante nous allons introduire une nouvelle classe de CTS qui contiendra suffisamment de systèmes pour être digne d'intérêt et dont les éléments satisferont toutes les propriétés de stabilité. De plus cette classe sera stable par la construction $\cdot \mapsto \cdot^2$.

3.5 Système stratifié engendré

Dans cette section, nous allons introduire la classe des *systèmes stratifiés* engendrés par un PTS fonctionnel et homogène $\mathcal{P}$. Une stratification $\Omega_\mathcal{P}$ de $\mathcal{P}$ sera obtenue en dupliquant chaque sorte $s \in \mathcal{S}_\mathcal{P}$ un certain nombre de fois que l'on notera ϵ_s. Ainsi, les sortes de $\Omega_\mathcal{P}$ consisteront en la donnée, pour chaque sorte s de $\mathcal{P}$, de

$$s_0, s_1, \ldots, s_i, \ldots$$

avec $i < \epsilon_s$ (en pratique on aura toujours $\epsilon_s \leq \omega$). On dira alors que i est la *hauteur* de s_i et que s_i est *engendrée* par s.

3.5.1 Convention

Dans un contexte où nous étudierons les sortes d'un système stratifié, nous n'utiliserons uniquement les indices sous les sortes pour indiquer la hauteur de ces dernières et nous utiliserons pas les indices pour distinguer les "méta-variables". Ainsi si $s \in \mathcal{S}_\mathcal{P}$ et $s_0 \in \mathcal{S}_{\Omega_\mathcal{P}}$ alors s_0 désigne la sorte de hauteur 0 engendrée par s (ce n'est donc pas n'importe quelle sorte de $\mathcal{S}_{\Omega_\mathcal{P}}$).

La relation de cumulativité est générée à partir de l'ordre associé à ϵ_s, ainsi on aura $s_i \preccurlyeq_{\Omega_\mathcal{P}} r_j$ si et seulement si $s = r$ et $i \leq j$. Les axiomes de $\Omega_\mathcal{P}$ seront engendrés à partir de chacun des axiomes $(s, r) \in \mathcal{A}_\mathcal{P}$ et d'une fonction notée $\xi_{s,r} : \epsilon_s \longrightarrow \epsilon_r + 1$ qui donnera la hauteur du type de s_i : $(s_i, r_{\xi_{s,r}(i)}) \in \mathcal{A}_{\Omega_\mathcal{P}}$ à condition que $\xi_{s,r}(i) < \epsilon_r$; le "+1" permet donc de spécifier qu'une sorte cesse d'être typable à partir d'une certaine hauteur. Enfin, les règles de $\Omega_\mathcal{P}$ sont générées à partir des règles

$(s,r,r) \in \mathcal{R}_\mathcal{P}$ et d'une fonction notée $\psi_{s,r} : \epsilon_s \longrightarrow \epsilon_r + 1$ qui donne, en fonction d'une hauteur i pour s, la hauteur $\psi_{s,r}(i)$ pour r à partir de laquelle les produits de la forme (s_i, r_j, r_j) sont autorisés pour $\psi_{s,r}(i) \leq j < \epsilon_r$; donc si $\psi_{s,r}(i) = \epsilon_r$ alors la règle $(s_i, , r_j, r_j)$ n'est pas autorisée.

La donnée des familles $(\epsilon_s)_{s\in\mathcal{S}_\mathcal{P}}$, $(\xi_{s,r})_{(s,r)\in\mathcal{A}_\mathcal{P}}$ et $(\psi_{s,r})_{(s,r,r)\in\mathcal{R}_\mathcal{P}}$ constituent donc les paramètres de la stratification $\Omega_\mathcal{P}$ de $\mathcal{P}$. Ce que l'on résume par la définition suivante :

3.5.2 Définition (Système stratifié engendré)
Soit $\mathcal{P}$ un PTS tel que $\mathcal{P} \models$ HOMOGENEOUS $\wedge$ FUNCTIONAL et soit la donnée de :

- Une famille $(\epsilon_s)_{s\in\mathcal{S}_\mathcal{P}}$ d'ordinaux
- Une famille $(\xi_{s,r})_{(s,r)\in\mathcal{A}_\mathcal{P}}$ de fonctions croissantes $\xi_s : \epsilon_s \longrightarrow \epsilon_r + 1$.
- Une famille $(\psi_{s,r})_{(s,r,r)\in\mathcal{R}_\mathcal{P}}$ de fonctions croissantes $\psi_{s,r} : \epsilon_s \longrightarrow \epsilon_r + 1$.

Notons $\Omega_\mathcal{P} = (\epsilon, \xi, \psi)$. On définit alors le système stratifié engendré par $\Omega_\mathcal{P}$ comme étant le CTS, que l'on désignera également par $\Omega_\mathcal{P}$, défini par les équations suivantes :

$$\begin{array}{rcl{l}{|}{c}{l}{c}{l}}
\mathcal{S}_{\Omega_\mathcal{P}} & = & \{ & s_i & s \in \mathcal{S}_\mathcal{P} & \wedge & i < \epsilon_s & \} \\
\mathcal{A}_{\Omega_\mathcal{P}} & = & \{ & (s_i, r_{\xi_{s,r}(i)}) & (s,r) \in \mathcal{A}_\mathcal{P} & \wedge & i < \epsilon_s \wedge \xi_{s,r}(i) < \epsilon_r & \} \\
\mathcal{R}_{\Omega_\mathcal{P}} & = & \{ & (s_i, r_j, r_j) & (s,r,r) \in \mathcal{R}_\mathcal{P} & \wedge & i < \epsilon_s \wedge \psi_{s,r}(i) \leq j \leq \epsilon_r & \} \\
\mathcal{C}_{\Omega_\mathcal{P}} & = & \{ & (s_i, s_j) & s \in \mathcal{S}_\mathcal{P} & \wedge & i < j < \epsilon_s & \}
\end{array}$$

On dira d'un CTS $\mathcal{P}'$ tel que $\overline{\Omega_\mathcal{P}} = \overline{\mathcal{P}'}$ qu'il peut être présenté comme une stratification de $\mathcal{P}$ selon $\Omega_\mathcal{P}$.

La plupart des systèmes que nous avons étudiés jusqu'à présent peuvent être représentés comme une stratification de PTS :

3.5.3 Exemple

- *Soit $\mathcal{P}$ un PTS fonctionnel et homogène, alors $\mathcal{P}$ est une stratification de $\mathcal{P}$ selon :*

$$\begin{array}{rcl}
\epsilon_s & = & 1 \text{ pour } s \in \mathcal{S}_\mathcal{P} \\
\xi_{s,r}(0) & = & 0 \text{ pour } (s,r) \in \mathcal{A}_\mathcal{P} \\
\psi_{s,r}(0) & = & 0 \text{ pour } (s,r,r) \in \mathcal{R}_\mathcal{P}
\end{array}$$

- *Le système $\lambda\star_\omega$ (page 93) est une stratification de $\lambda\star$ (page 52) selon*

$$\begin{array}{rcl}
\epsilon_\star & = & \omega \\
\xi_{\star,\star}(i) & = & i+1 \\
\psi_{\star,\star}(i) & = & i
\end{array}$$

- *Le système $\mathcal{SF}^\alpha$ (page 97) est une stratification de $\mathcal{F}$ selon*

$$\begin{array}{rcl@{\qquad}rcl}
\epsilon_\star & = & \alpha & \xi_{\star,\Box}(i) & = & i+1 \\
\epsilon_\Box & = & \alpha & \psi_{\star,\star}(i) & = & i \\
 & & & \psi_{\Box,\star}(i) & = & i
\end{array}$$

- *Le système $\mathcal{F}_n$ défini comme une stratification de $\mathcal{F}_\omega$ selon*

$$\begin{array}{rcl@{\qquad}rcl}
\epsilon_\star & = & 1 & \xi_{\star,\Box}(i) & = & 0 \\
\epsilon_\Box & = & n+1 & \psi_{\star,\star}(0) & = & 0 \\
 & & & \psi_{\Box,\Box}(i) & = & i+1 \\
 & & & \psi_{\Box,\star}(i) & = & 0
\end{array}$$

Le système $\mathcal{F}_0$ correspond à système $\mathcal{F}$, le système $\mathcal{F}_1$ autorise les quantifications sur les constructeurs de types comme par exemple :

$$\vdash_{\mathcal{F}_1} \forall c : \star \to \star \to \star, \alpha : \star, \beta : \star . c\,\alpha\,\beta : \star$$

(on note ici $\star$ à la place de $\star_0$ car il n'y a pas d'ambiguïté puisque $\epsilon_\star = 1$).
Le système $\mathcal{F}_2$ autorise les quantifications sur les constructeurs de constructeurs de types comme par exemple :

$$\vdash_{\mathcal{F}_2} \forall c : (\star \to \star) \to \star, \alpha : \star . c\,(\lambda\beta : \star . \beta)\,\alpha : \star$$

Le système $\mathcal{F}_3$ autorise les quantifications sur les constructeurs de constructeurs de constructeurs de types, etc...
Ainsi $\mathcal{F}_\omega$ peut se comprendre comme la limite des $\mathcal{F}_n$ puisqu'il n'admet aucune limite dans l'ordre des constructeurs de type.

– *Soit $\lambda\star_{\mathtt{Prop},\mathtt{Set}}$ le PTS fonctionnel et homogène défini par les équations suivantes :*

$$\begin{aligned}
\mathcal{S}_{\lambda\star_{\mathtt{Prop},\mathtt{Set}}} &= \{\mathtt{Set}, \mathtt{Prop}, \mathtt{Type}\} \\
\mathcal{A}_{\lambda\star_{\mathtt{Prop},\mathtt{Set}}} &= \{(\mathtt{Set}, \mathtt{Type}), (\mathtt{Prop}, \mathtt{Type}), (\mathtt{Type}, \mathtt{Type})\} \\
\mathcal{R}_{\lambda\star_{\mathtt{Prop},\mathtt{Set}}} &= \{(s, r, r) | (s, r) \in \mathcal{S} \times \mathcal{S}\}
\end{aligned}$$

Il est obtenu en ajoutant à $\lambda\star$ les sortes `Set` *et* `Prop`*. On a alors :*
– *Le système* CIC$^-$ *(page 139) est une stratification de $\lambda\star_{\mathtt{Prop},\mathtt{Set}}$ selon*

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl}
\epsilon_{\mathtt{Type}} &=& \omega & \xi_{\mathtt{Type},\mathtt{Type}}(i) &=& i+1 & \psi_{s,\mathtt{Prop}}(i) &=& 0 \\
\epsilon_{\mathtt{Prop}} &=& 1 & \xi_{\mathtt{Prop},\mathtt{Type}}(0) &=& 1 & \psi_{s,\mathtt{Set}}(i) &=& 0 \\
\epsilon_{\mathtt{Set}} &=& 1 & \xi_{\mathtt{Set},\mathtt{Type}}(0) &=& 1 & \psi_{\mathtt{Type},\mathtt{Type}}(i) &=& i \\
 & & & & & & \psi_{\mathtt{Prop},\mathtt{Type}}(0) &=& 0 \\
 & & & & & & \psi_{\mathtt{Set},\mathtt{Type}}(0) &=& 0
\end{array}$$

– *Le système* CIC *(page 139) est une stratification de $\lambda\star_{\mathtt{Prop},\mathtt{Set}}$ selon*

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl}
\epsilon_{\mathtt{Type}} &=& \omega & \xi_{\mathtt{Type},\mathtt{Type}}(i) &=& i+1 & \psi_{\mathtt{Type},\mathtt{Type}}(i) &=& i \\
\epsilon_{\mathtt{Prop}} &=& 1 & \xi_{\mathtt{Prop},\mathtt{Type}}(0) &=& 1 & \psi_{\mathtt{Type},\mathtt{Prop}}(i) &=& 0 \\
\epsilon_{\mathtt{Set}} &=& 0 & \xi_{\mathtt{Set},\mathtt{Type}} &=& \emptyset & \psi_{\mathtt{Prop},\mathtt{Type}}(0) &=& 0 \\
 & & & & & & \psi_{\mathtt{Prop},\mathtt{Prop}} &=& 0 \\
 & & & & & & \psi_{\mathtt{Set},s} &=& \emptyset
\end{array}$$

où $\emptyset$ désigne la fonction de domaine vide.
– *Le système* CIC$^+$ *(page 139) est une stratification de $\lambda\star_{\mathtt{Prop},\mathtt{Set}}$ selon*

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl}
\epsilon_{\mathtt{Type}} &=& \omega & \xi_{\mathtt{Type},\mathtt{Type}}(i) &=& i+1 & \psi_{s,\mathtt{Prop}}(i) &=& 0 \\
\epsilon_{\mathtt{Prop}} &=& 1 & \xi_{\mathtt{Set},\mathtt{Type}}(i) &=& i+1 & \psi_{\mathtt{Prop},\mathtt{Set}}(0) &=& 0 \\
\epsilon_{\mathtt{Set}} &=& \omega & \xi_{\mathtt{Prop},\mathtt{Type}}(0) &=& 1 & \psi_{\mathtt{Set},\mathtt{Set}}(i) &=& i \\
 & & & & & & \psi_{\mathtt{Type},\mathtt{Set}}(i) &=& i \\
 & & & & & & \psi_{\mathtt{Type},\mathtt{Type}}(i) &=& i \\
 & & & & & & \psi_{\mathtt{Prop},\mathtt{Type}}(0) &=& 0 \\
 & & & & & & \psi_{\mathtt{Set},\mathtt{Type}}(i) &=& i
\end{array}$$

Les systèmes stratifiés satisfont les propriétés suivantes :

3.5.4 Lemme

1. $\Omega_{\mathcal{P}} \models$ HOMOGENEOUS,
2. $\Omega_{\mathcal{P}} \models$ MINLOC,
3. $\Omega_{\mathcal{P}} \models$ PRINCIPAL,
4. *Si pour tout* $i < \epsilon_r$, *pour tout* $(s,r) \in \mathcal{A}_{\mathcal{P}}, \xi_{s,r}(i) < \epsilon_r$ *et pour tout* $(s,r,r) \in \mathcal{R}_{\mathcal{P}}, \psi_{s,r}(i) < \epsilon_r$, *alors* $\Omega_{\mathcal{P}} \models$ UPSTABLE $\wedge$ STRENGTHENING

Démonstration 1. Par définition, les règles de $\Omega_{\mathcal{P}}$ sont toutes de la forme (s,r,r).

2. Il faut montrer $\Omega_{\mathcal{P}} \models$ MINLOC-AXIOMS et $\Omega_{\mathcal{P}} \models$ MINLOC-RULES :
   - $\Omega_{\mathcal{P}} \models$ MINLOC-AXIOMS : Soit $(s_i, r_{\xi_{s,r}(i)}) \in \mathcal{A}_{\Omega_{\mathcal{P}}}$, $(s_j, r_{\xi_{s,r}(j)}) \in \mathcal{A}_{\Omega_{\mathcal{P}}}$, et $k < \epsilon_s$ tel que $k \leq i$ et $k \leq j$. Il faut montrer qu'il existe $k' < \epsilon_r$ tel que $(s_k, r_{k'}) \in \mathcal{A}_{\Omega_{\mathcal{P}}}$, $k' \leq \xi_{s,r}(i)$ et $k' \leq \xi_{s,r}(j)$). Or par croissance de $\xi_{s,r}$, $k' = \xi_{s,r}(k)$ convient.
   - $\Omega_{\mathcal{P}} \models$ MINLOC-RULES : Soit $(s_i, r_{i'}, r_{i'}) \in \mathcal{R}_{\Omega_{\mathcal{P}}}$, $(s_j, r_{j'}, r_{j'}) \in \mathcal{R}_{\Omega_{\mathcal{P}}}$ et $k < \epsilon_s$ et $k' < \epsilon_r$ avec $k \leq i$, $k \leq j$ et $k' \leq i'$ et $k' \leq j'$. Il faut montrer qu'il existe $k'' < \epsilon_r$ tel que $(s_k, r_{k'}, r_{k''}) \in \overline{\mathcal{R}_{\Omega_{\mathcal{P}}}}$ avec $k'' \leq i'$ et $k'' \leq j'$. Par définition de $\mathcal{R}_{\Omega_{\mathcal{P}}}$, on a $\psi_{s,r}(i) \leq i' < \epsilon_r$ et $\psi_{s,r}(j) \leq j' < \epsilon_r$. Posons, $k'' = \max(k', \psi_{s,r}(k))$. On a alors bien $k'' \leq i'$ et $k'' \leq j'$ car :

$$\begin{cases} k' & \leq & i' \\ k' & \leq & j' \\ \psi_{s,r}(k) & \leq & \psi_{s,r}(i) & \leq & i' \\ \psi_{s,r}(k) & \leq & \psi_{s,r}(j) & \leq & j' \end{cases}$$

   Enfin, on a $(s_k, r_{k'}, r_{k''}) \in \overline{\mathcal{R}_{\Omega_{\mathcal{P}}}}$ car $(s_k, r_{k''}, r_{k''}) \in \mathcal{R}_{\Omega_{\mathcal{P}}}$ et $k' \leq k'' < \epsilon_r$.

3. Pour prouver que $\Omega_{\mathcal{P}} \models$ PRINCIPAL, il faut montrer que la relation de cumulativité $\mathcal{C}_{\Omega_{\mathcal{P}}}$ est bien fondée (ce qui est évident car chacune des composantes connexes de la relation est isomorphe à l'un des ordinal ϵ_s pour un $s \in \mathcal{S}_{\mathcal{P}}$) et vérifier que $\Omega_{\mathcal{P}} \models$ MINLOC. Ce qui est traité par le cas précédent.

4. La vérification de UPSTABLE est aisée :
   - UPSTABLE-AXIOMS : Soit $(s_i, r_j) \in \overline{\mathcal{A}_{\Omega_{\mathcal{P}}}}$ et $s_i \preccurlyeq s_{i'}$. Par définition de $\Omega_{\mathcal{P}}$, cela signifie que $i \leq i'$ et que $\xi_{s,r}(i) \leq j < \epsilon_r$. Alors par croissance de $\xi_{s,r}$, on a $\xi_{s,r}(i) \leq \xi_{s,r}(i')$. Et par hypothèse on a $\xi_{s,r}(i') < \epsilon_r$. Et donc, si on pose $j' = \max(j, \xi_{s,r}(i'))$, nous obtenons bien $r_{j'} \succcurlyeq r_j$, $\xi_{s,r}(i') \leq j' < \epsilon_r$ et donc $(s_{i'}, r_{j'}) \in \overline{\mathcal{A}_{\Omega_{\mathcal{P}}}}$.
   - UPSTABLE-RULES : Soit $(s_i, r_j, r_k) \in \overline{\mathcal{R}_{\Omega_{\mathcal{P}}}}$ avec $s_i \preccurlyeq s_{i'}$ et $r_j \preccurlyeq r_{j'}$; c'est-à-dire $i \leq i' < \epsilon_s$ et $j \leq j' < \epsilon_r$ et $k < \epsilon_r$. Il faut montrer qu'il existe $k \leq k'$ tel que $(s_{i'}, r_{j'}, r_{k'}) \in \overline{\mathcal{R}_{\Omega_{\mathcal{P}}}}$: Posons $k' = \max(j', k, \psi_{s,r}(i'))$. Par hypothèse, on a $\psi_{s,r}(i') < \epsilon_r$, on en déduit donc que $k' < \epsilon_r$. De plus, par définition de $\mathcal{R}_{\Omega_{\mathcal{P}}}$, on a $(s_{i'}, r_x, r_x) \in \mathcal{R}_{\Omega_{\mathcal{P}}}$ pour tout $\epsilon_r > x \geq \psi_{s,r}(i')$ on en déduit donc que $(s_{i'}, r_{k'}, r_{k'}) \in \mathcal{R}_{\Omega_{\mathcal{P}}}$. Or $k' \geq j'$, on a donc $(s_{i'}, r_{j'}, r_{k'}) \in \overline{\mathcal{R}_{\Omega_{\mathcal{P}}}}$.

   Enfin STRENGTHENING est une conséquence directe de UPSTABLE, des points précédents et du lemme 1.2.37. ☺

On dispose du critère ci-dessous pour tester la prédicativité des systèmes stratifié.

3.5.5 Lemme
S'il existe pour chaque sorte $s \in \mathcal{S}_\mathcal{P}$, un ordinal δ_s tel que :
- *Pour tout $(s,r) \in \mathcal{A}_\mathcal{P}$, $\delta_r + \xi_{s,r}(i) > \delta_s + i$.*
- *Pour tout $(s,r,r) \in \mathcal{R}_\mathcal{P}$, $\delta_r + \psi_{s,r}(i) \geq \delta_s + i$.*

alors $\Omega_\mathcal{P} \models$ PREDICATIVE.

Démonstration Pour prouver qu'un CTS est prédicatif, il suffit d'exhiber une relation bien fondée R qui satisfait les règles de la définition 1.3.1. Par construction, on aura $<_{\Omega_\mathcal{P}} \subseteq R$ et donc R bien fondée implique $<_{\Omega_\mathcal{P}}$ bien fondée (et donc $\Omega_\mathcal{P}$ est un système prédicatif). Posons $s_i \, R \, r_j \Leftrightarrow \delta_s + i < \delta_r + j$ et vérifions que R satisfait bien les règles de la définition 1.3.1 :
- Axiome : Il faut montrer que l'on a pour tout $(s_i, r_{\xi_{s,r}(i)}) \in \mathcal{A}_{\Omega_\mathcal{P}}$, $s_i \, R \, r_{\xi_{s,r}(i)}$ c'est-à-dire $\delta_s + i < \delta_r + \xi_{s,r}(i)$. Ce qui est bien une des hypothèses du lemme.
- Prémisse : Soit $(s_i, r_j, r_j) \in \mathcal{R}_{\Omega_\mathcal{P}}$ avec $\psi_{s,r}(i) \leq j < \epsilon_r$. Supposons $t_k \, R \, s_i$ (c-à-d $\delta_t + k < \delta_s + i$). Montrons que $t_k \, R \, r_j$. On montre montre grâce à la seconde hypothèse du lemme que $\delta_t + k < \delta_s + i \leq \delta_r + \psi_{s,r}(i) \leq \delta_r + j$ et donc on a bien $t_k \, R \, r_j$.
- Conclusion : Ce cas est tautologique dans les systèmes vérifiant HOMOGENEOUS (il faut vérifier que si $(s_i, r_j, r_j) \in \mathcal{R}_{\Omega_\mathcal{P}}$, alors $t_k \, R \, r_j$ implique $t_k \, R \, r_j$).
- Transitivité : La relation R est bien transitive.
- Cumulativité : Pour traiter ce cas, il faut montrer que si $s_i \, R \, r_j$ et $(r_j, r_k) \in \mathcal{C}$, alors $s_i \, R \, r_k$. On remarque que $\mathcal{C}_{\Omega_\mathcal{P}} \subseteq R$ et donc la transitivité de R, nous permet bien de conclure $s_i \, R \, r_k$. ☺

À titre d'exemple on prouve que la théorie des types prédicative $\lambda\star_\omega$, le système stratifié $\mathcal{SF}^\omega$, le λ-calcul simplement typé λ sont prédicatifs :
- $\lambda\star_\omega \models$ PREDICATIVE : On prend $\delta\delta_\star = 0$ et on a bien $\xi_{\star,\star}(i) = i + 1 > i$ et $\psi_{\star,\star}(i) = i \geq i$.
- $\mathcal{SF}^\omega \models$ PREDICATIVE : On prend $\delta_\Box = \delta_\star = 0$. On a bien $\xi_{\star,\Box}(i) = i + 1 > i$ et $\psi_{\star,\star}(i) = \psi_{\Box,\star}(i) = i \geq i$.
- $\lambda \models$ PREDICATIVE : Pour montrer que le système λ est prédicatif, on peut prendre $\delta_\Box = 1$ et $\delta_\star = 0$. Et on vérifie bien que $\delta_\star + 0 = 0 + 0 < \xi_{\star,\Box}(0) + \delta_\Box = 0 + 1$ et que $\delta_\star + 0 \leq \psi_{\star,\star}(0) + \delta_\star = 0 + 0$.

Ce critère s'étend en un critère pour tester la faible imprédicativité de la façon suivante :

3.5.6 Lemme
Soit $S \subseteq \mathcal{S}_\mathcal{P}$ un ensemble de sortes minimales pour $\mathcal{A}_\mathcal{P}$ et un ordinal δ_s pour chaque sorte $s \in \mathcal{S}_\mathcal{P}$. Supposons de plus que :
- *Pour tout $(s,r) \in \mathcal{A}_\mathcal{P}$, $\delta_r + \xi_{s,r}(i) > \delta_s + i$.*
- *Pour tout $(s,r,r) \in \mathcal{R}_\mathcal{P}$ avec $r \notin S$, $\delta_r + \psi_{s,r}(i) \geq \delta_s + i$.*

Alors $\Omega_\mathcal{P} \models$ WEAKLY-IMPREDICATIVE.

Démonstration On pose $S' = \{s_i | s \in S, i < \epsilon_s\}$.

1. L'ensemble S' est un ensemble de sortes minimales pour $\mathcal{A}_{\Omega_\mathcal{P}}$: En effet si $(s_i, r_j) \in \mathcal{A}_{\Omega_\mathcal{P}}$, alors $(s,r) \in \mathcal{A}_\mathcal{P}$. Or par hypothèse, si $r \in S$, alors r est minimale pour $\mathcal{A}_\mathcal{P}$ et donc on en déduit que $r_j \notin S'$.
2. L'ensemble S' est stable vers le bas pour $\mathcal{C}_{\Omega_\mathcal{P}}$: Si $s_i \in S'$ et $s_j \preccurlyeq s_i$ (c-à-d $j \leq i$), alors on a bien $s_j \in S'$.
3. Si $(s,r,t) \in \mathcal{R}_{\Omega_\mathcal{P}}$ alors $r \in S' \Leftrightarrow t \in S'$: $\Omega_\mathcal{P}$ vérifie HOMOGENEOUS, on a donc $r = t$.

4. Le système $\underline{\mathcal{P}}$, défini par les paramètres ci-dessous, est un système prédicatif.

$$\begin{array}{rcl} \mathcal{S}_{\underline{\Omega_\mathcal{P}}} & = & \mathcal{S}_{\Omega_\mathcal{P}} \\ \mathcal{A}_{\underline{\Omega_\mathcal{P}}} & = & \mathcal{A}_{\Omega_\mathcal{P}} \\ \mathcal{R}_{\underline{\Omega_\mathcal{P}}} & = & \{(s_1, s_2, s_3) \in \mathcal{R}_{\Omega_\mathcal{P}} \,|s_2 \notin S' \wedge s_3 \notin S'\} \\ \mathcal{C}_{\underline{\Omega_\mathcal{P}}} & = & \mathcal{C}_{\Omega_\mathcal{P}} \end{array}$$

Il suffit de remarquer qu'il peut être présenté comme une stratification $\Omega_{\underline{\mathcal{P}}}$ de $\underline{\mathcal{P}}$ et qu'il vérifie bien les conditions du lemme 3.5.5. ☺

On peut se servir du précédent critère pour démontrer que $\mathcal{F}$, CIC$^-$, CIC, et CIC$^+$ sont faiblement imprédicatifs :

1. $\mathcal{F} \models$ WEAKLY-IMPREDICATIVE : On prend $S = \{\star\}$, $\delta_\star = 0$ et $\delta_\Box = 1$. On a bien $\delta_\star + 0 = 0 + 0 < \xi_{\star,\Box}(0) + \delta_\Box = 0 + 1$ et il n'y aucune condition à vérifier sur ψ car toutes les règles de $\mathcal{F}$ sont de la forme $(s, \star, \star)$.
2. $\mathcal{P} \models$ WEAKLY-IMPREDICATIVE pour $\mathcal{P} = \text{CIC}^-, \text{CIC}, \text{CIC}^+$: On prends $S = \{$ `Prop`, `Set`$\}$ et $\delta_s = 0$ pour tout $s \in \mathcal{S}_\mathcal{P}$. On vérifie bien :
   – $\xi_{s,r}(i) > i$,
   – si $r \neq$ `Prop`, `Set`, alors $\psi_{s,r}(i) \geq i$ pour tout $i < \epsilon_s$.
   On remarquera que dans les cas de CIC et CIC$^+$ on peut également utiliser le critère en prenant $S = \{$ `Prop` $\}$.

La transformation $\cdot \mapsto \cdot^2$ des systèmes stratifiés. Le lemme ci-dessous montre que l'image par la construction $\cdot \mapsto \cdot^2$ d'un système stratifié engendré par un PTS $\mathcal{P}$ peut être présentée comme un système stratifié engendré par $\mathcal{P}^2$. Ce lemme nous garantira que les systèmes auxquelles nous appliquerons la transformation $\cdot \mapsto \cdot^2$ auront toutes les bonnes propriétés des systèmes stratifiés.

3.5.7 Lemme
Nous avons $(\Omega_\mathcal{P})^2 = \Omega_{\mathcal{P}^2}$ où $\Omega_{\mathcal{P}^2}$ est le système stratifie définit par $\mathcal{P}^2$ en étendant la spécification de $\Omega_\mathcal{P}$ aux sortes de second-niveau de la façon suivante :

$$\begin{array}{rcl} \epsilon_{\lceil s \rceil} & = & \epsilon_s \\ \xi_{\lceil s \rceil, \lceil r \rceil}(i) & = & \xi_{s,r}(i) \\ \psi_{\lceil s \rceil, \lceil r \rceil}(i) & = & \psi_{s,r}(i) \\ \psi_{s, \lceil r \rceil}(i) & = & \begin{cases} \min(\psi_{s,r}(i), \xi_{s,r}(i)) & si\ (s,r,r) \in \mathcal{R} \wedge (s,r) \in \mathcal{A} \\ \psi_{s,r}(i) & si\ (s,r,r) \in \mathcal{R} \wedge (s,r) \notin \mathcal{A} \\ \xi_{s,r}(i) & si\ (s,r,r) \notin \mathcal{R} \wedge (s,r) \in \mathcal{A} \end{cases} \end{array}$$

et où on identifie $\lceil s_i \rceil$ et $\lceil s \rceil_i$.

Démonstration On montre l'égalité des paramètres, les vérifications pour $\mathcal{S}$, $\mathcal{C}$ et $\mathcal{A}$ sont aisées :

– $\mathcal{S}_{\Omega_{\mathcal{P}^2}} = \mathcal{S}_{(\Omega_\mathcal{P})^2}$:

$$\begin{array}{rcl}
\mathcal{S}_{\Omega_{\mathcal{P}^2}} & = & \{ \quad s_i \quad | \quad s \in \mathcal{S}_{\mathcal{P}^2} \wedge i < \epsilon_s \quad \} \\
& = & \{ \quad s_i, \lceil s \rceil_i \quad | \quad s \in \mathcal{S}_\mathcal{P} \wedge i < \epsilon_s \quad \} \\
& = & \{ \quad s_i \quad | \quad s \in \mathcal{S}_\mathcal{P} \wedge i < \epsilon_s \quad \} \cup \{ \quad \lceil s_i \rceil \quad | \quad s \in \mathcal{S}_\mathcal{P} \wedge i < \epsilon_s \quad \} \\
& = & \{ \quad s \quad | \quad s \in \mathcal{S}_{\Omega_\mathcal{P}} \quad \} \cup \{ \quad \lceil s \rceil \quad | \quad s \in \mathcal{S}_{\Omega_\mathcal{P}} \quad \} \\
& = & \mathcal{S}_{(\Omega_\mathcal{P})^2}
\end{array}$$

– $\mathcal{C}_{\Omega_{\mathcal{P}^2}} = \mathcal{C}_{(\Omega_\mathcal{P})^2}$:

$$\begin{array}{rcl}
\mathcal{C}_{\Omega_{\mathcal{P}^2}} & = & \{ \quad (s_i, s_j) \quad | \quad s \in \mathcal{S}_{\mathcal{P}^2} \wedge i < j < \epsilon_s \quad \} \\
& = & \{ \quad (s_i, s_j),\ (\lceil s \rceil_i, \lceil s \rceil_j) \quad | \quad s \in \mathcal{S}_\mathcal{P} \wedge i < j < \epsilon_s \quad \} \\
& = & \{ \quad (s, r),\ (\lceil s \rceil, \lceil r \rceil) \quad | \quad (s, r) \in \mathcal{C}_{\Omega_\mathcal{P}} \quad \} \\
& = & \mathcal{C}_{(\Omega_\mathcal{P})^2}
\end{array}$$

– $\mathcal{A}_{\Omega_{\mathcal{P}^2}} = \mathcal{A}_{(\Omega_\mathcal{P})^2}$:

$$\begin{array}{rcl}
\mathcal{A}_{\Omega_{\mathcal{P}^2}} & = & \{ \quad (s_i, r_{\xi_{s,r}(i)}) \quad | \quad (s, r) \in \mathcal{A}_{\mathcal{P}^2} \wedge i < \epsilon_s \wedge \xi_{s,r}(i) < \epsilon_r \quad \} \\
& = & \{ \quad (s_i, r_{\xi_{s,r}(i)}), (\lceil s \rceil_i, \lceil r \rceil_{\xi_{s,r}(i)}) \quad | \quad (s, r) \in \mathcal{A}_\mathcal{P} \wedge i < \epsilon_s \wedge \xi_{s,r}(i) < \epsilon_r \quad \} \\
& = & \mathcal{A}_{(\Omega_\mathcal{P})^2}
\end{array}$$

– $\mathcal{R}_{\Omega_{\mathcal{P}^2}} = \mathcal{R}_{(\Omega_\mathcal{P})^2}$:

$$\begin{array}{rcl}
& & \mathcal{R}_{\Omega_{\mathcal{P}^2}} \\
& = & \{ \quad (s_i, r_j, r_j) \quad | \quad (s, r, r) \in \mathcal{R}_{\mathcal{P}^2}, i < \epsilon_s, \psi_{s,r}(i) \leq j < \epsilon_r \quad \} \\
& = & \left\{ \begin{array}{c} (s_i, r_j, r_j), \\ (\lceil s \rceil_i, \lceil r \rceil_j, \lceil r \rceil_j) \end{array} \quad | \quad (s, r, r) \in \mathcal{R}_\mathcal{P}, i < \epsilon_s, \psi_{s,r}(i) = \psi_{\lceil s \rceil, \lceil r \rceil}(i) \leq j < \epsilon_r \quad \right\} \\
& \cup & \{ \quad (s_i, \lceil r \rceil_j, \lceil r \rceil_j) \quad | \quad (s, r) \in \mathcal{A}_\mathcal{P}, i < \epsilon_s, \psi_{s,\lceil r \rceil}(i) \leq j < \epsilon_r \quad \} \\
& \cup & \{ \quad (s_i, \lceil r \rceil_j, \lceil r \rceil_j) \quad | \quad (s, r, r) \in \mathcal{R}_\mathcal{P}, i < \epsilon_s, \psi_{s,\lceil r \rceil}(i) \leq j < \epsilon_r \quad \}
\end{array}$$

Posons

$$\begin{array}{rcl}
\mathcal{E}_1 & = & \{ \quad (s_i, \lceil r \rceil_j, \lceil r \rceil_j) \quad | \quad (s, r) \in \mathcal{A}_\mathcal{P}, i < \epsilon_s, \psi_{s,\lceil r \rceil}(i) \leq j < \epsilon_r \quad \} \\
& \cup & \{ \quad (s_i, \lceil r \rceil_j, \lceil r \rceil_j) \quad | \quad (s, r, r) \in \mathcal{R}_\mathcal{P}, i < \epsilon_s, \psi_{s,\lceil r \rceil}(i) \leq j < \epsilon_r \quad \}
\end{array}$$

et

$$\begin{array}{rcl}
\mathcal{E}_2 & = & \{ \quad (s_i, \lceil r \rceil_j, \lceil r \rceil_j) \quad | \quad (s, r) \in \mathcal{A}_\mathcal{P}, i < \epsilon_s, \xi_{s,r}(i) \leq j < \epsilon_r \quad \} \\
& \cup & \{ \quad (s_i, \lceil r \rceil_j, \lceil r \rceil_j) \quad | \quad (s, r, r) \in \mathcal{R}_\mathcal{P}, i < \epsilon_s, \psi_{s,r}(i) \leq j < \epsilon_r \quad \}
\end{array}$$

On a clairement $\mathcal{E}_2 \subseteq \mathcal{E}_1$ puisque que l'on a toujours $\psi_{s,\lceil r \rceil}(i) \leq \xi_{s,r}(i)$ si $(s, r) \in \mathcal{A}_\mathcal{P}$ et $\psi_{s,\lceil r \rceil}(i) \leq \psi_{s,r}(i)$ si $(s, r, r) \in \mathcal{R}_\mathcal{P}$. Montrons que $\mathcal{E}_1 \subseteq \mathcal{E}_2$.

– Supposons $(s, r) \in \mathcal{A}_\mathcal{P}$ avec $i < \epsilon_s$ et $\psi_{s,\lceil r \rceil}(i) \leq j < \epsilon_r$.
  Montrons que $(s_i, \lceil r \rceil_j, \lceil r \rceil_j) \in \mathcal{E}_2$. On distingue deux cas :
  – $(s, r, r) \in \mathcal{R}_\mathcal{P}$: On a $\psi_{s,\lceil r \rceil}(i) = \min(\psi_{s,r}(i), \xi_{s,r}(i))$. Deux cas sont alors possibles :
    – $\xi_{s,r}(i) \leq \psi_{s,r}(i)$: On a alors $\xi_{s,r}(i) \leq j$ et donc $(s_i, \lceil r \rceil_j, \lceil r \rceil_j) \in \mathcal{E}_2$.
    – $\psi_{s,r}(i) < \xi_{s,r}(i)$: On a alors $\psi_{s,r}(i) \leq j$ et donc $(s_i, \lceil r \rceil_j, \lceil r \rceil_j) \in \mathcal{E}_2$.
  – $(s, r, r) \notin \mathcal{R}_\mathcal{P}$: On a $\psi_{s,\lceil r \rceil}(i) = \xi_{s,r}(i)$ et donc $\xi_{s,r}(i) \leq j$ et $(s_i, \lceil r \rceil_j, \lceil r \rceil_j) \in \mathcal{E}_2$.

– Supposons $(s,r,r) \in \mathcal{R}_{\mathcal{P}}$ avec $i < \epsilon_s$ et $\psi_{s,\lceil r \rceil}(i) \leq j < \epsilon_r$. On peut également supposer que $(s,r) \notin \mathcal{A}_{\mathcal{P}}$ (car le contraire à été étudié dans le cas précédent). Dans ce cas, $\psi_{s,\lceil r \rceil}(i) = \psi_{s,r}(i)$ et on a donc $\psi_{s,r}(i) \leq j$ et $(s_i, \lceil r \rceil_j, \lceil r \rceil_j) \in \mathcal{E}_2$.

Nous en déduisons donc que $\mathcal{E}_2 = \mathcal{E}_1$, d'où :

$$
\begin{array}{lll}
 & \mathcal{R}_{\Omega_{\mathcal{P}^2}} & \\
= & \left\{ \begin{array}{c} (s_i, r_j, r_j), \\ (\lceil s \rceil_i, \lceil r \rceil_j, \lceil r \rceil_j), \\ (s_i, \lceil r \rceil_j, \lceil r \rceil_j) \end{array} \right. & \left| \quad (s,r,r) \in \mathcal{R}_{\mathcal{P}} \wedge \psi_{s,r}(i) \leq j < \epsilon_r \wedge i < \epsilon_s \right. \left. \right\} \\
\cup & \{ \quad (s_i, \lceil r \rceil_j, \lceil r \rceil_j) & \left| \quad (s,r) \in \mathcal{A}_{\mathcal{P}} \wedge \xi_{s,r}(i) \leq j < \epsilon_r \wedge i < \epsilon_s \right. \} \\
= & \left\{ \begin{array}{c} (s,r,t), \\ (\lceil s \rceil, \lceil r \rceil, \lceil t \rceil), \\ (s, \lceil t \rceil, \lceil t \rceil) \end{array} \right. & \left| \quad (s,r,t) \in \mathcal{R}_{\Omega_{\mathcal{P}}} \right. \left. \right\} \\
\cup & \{ \quad (s, \lceil r \rceil, \lceil r \rceil) & \left| \quad (s,r) \in \mathcal{A}_{\Omega_{\mathcal{P}}} \right. \} \\
= & \mathcal{R}_{(\Omega_{\mathcal{P}})^2} &
\end{array}
$$

☺

Chapitre 4

Réalisabilité

Le concept de réalisabilité a été introduit par Kleene en 1945 dans un article [Kle45] qui depuis fait école. L'idée de mettre en relation les programmes et les formules, afin d'étudier leur contenu calculatoire, a été largement utilisée par les théoriciens de la démonstration. Un des premiers exemples d'application de la réalisabilité est de fournir des outils pour prouver que certaines formules sont indépendantes de certains systèmes d'axiomes. Par exemple, elle peut être utilisée pour prouver l'indépendance du tiers-exclu dans des systèmes intuitionnistes [Kle71, Tro98]. Elle permet également de démontrer d'autres résultats méta-mathématiques comme le fait que les systèmes intuitionnistes satisfont la *propriété de l'existence* [1] (voir [Har56, Tro98]). Nous renvoyons le lecteur intéressé par les aspects historiques du développement de la réalisabilité au compte rendu historique de Jap Van Oosten [Van02].

Dans l'article originel, Kleene représente les programmes comme des entiers grâce à une énumération à la Gödel des fonctions récursives. Cette technique a été étendue afin de prendre en compte d'autres notions de programme comme les algèbres de combinateurs [Sta73, Tro98] ou encore les termes du Système $\mathcal{T}$ de Gödel grâce à une variante de la réalisabilité –dite *modifiée*– due à Kreisel [Kre59, Tro98]. Dans notre travail, nous généralisons cette dernière approche en utilisant un système de type pur arbitraire (un CTSI) comme langage pour les programmes.

Cette approche a déjà été suivie auparavant par Daniel Leivant [Lei90], ainsi que par Jean-Louis Krivine et Michel Parigot [KP90, Kri93] dans le but de prouver le théorème de représentation de Girard [Gir72] : les fonctions définissables dans $\mathcal{F}$ sont exactement les fonctions prouvablement totales dans l'arithmétique du second ordre.

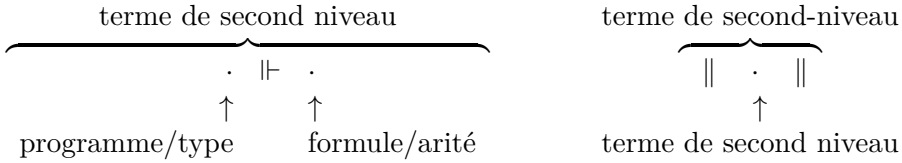
4.1 Définitions

Dans cette section, nous allons définir deux termes de second niveau :

- La relation de réalisabilité. Elle admet deux paramètres, correspondant aux termes mis en relation ; l'un est un terme de premier niveau et l'autre un terme de second niveau. On note la relation de réalisabilité $P \Vdash F$ entre deux termes et on dit alors que "P réalise F".
- La transformation de réalisabilité. Elle n'admet qu'un paramètre et n'est bien définie que sur les termes de second niveau. On la note $\|A\|$ et on l'étend également aux contextes. La transformation

1. Si $\forall x \exists y.\varphi(x,y)$ est dérivable, alors il existe un programme f qui fournit des témoins, c-à-d. tel que $\forall x, \varphi(x, f(x))$

de réalisabilité fournira, étant donnée une preuve π d'une formule F dans un contexte Γ (c'est-à-dire $\Gamma \vdash \pi : F$ avec $\mathrm{lvl}(F) = \mathrm{lvl}(\pi) = 2$), une nouvelle preuve $\|\pi\|$ de la formule $\lfloor\pi\rfloor \Vdash F$ dans le contexte $\|\Gamma\|$. En d'autres termes, $\|\pi\|$ est une preuve que la projection $\lfloor\pi\rfloor$ est un programme qui réalise la formule F.



relation de réalisabilité **transformation de réalisabilité**

Pour définir les relations de réalisabilité, nous avons besoin d'un générateur de noms frais. Étant donnée une variable x, on notera x_R une nouvelle variable fraîche explicitement associée à x. Les variables x_R et y_R seront syntaxiquement égales si et seulement si x et y sont syntaxiquement égales. De plus on étendra la notation A_R à tous les termes de second niveau en remplaçant les occurrences libres des variables x^2 de second niveau par x^2_R. Et on étend cette notation aux contextes en l'appliquant aux assignations de second niveau. Plus formellement :

$$\begin{aligned}
(X^2)_R &= X^2_R \\
s_R &= s \\
(\lambda x^1 : A.M)_R &= \lambda x^1 : A.M_R \\
(\forall x^1 : A.M)_R &= \forall x^1 : A.M_R \\
(\lambda x^2 : A.B)_R &= \lambda x^2_R : A_R.B_R \\
(\forall x^2 : A.B)_R &= \forall x^2_R : A_R.B_R
\end{aligned}$$

$$\begin{aligned}
(M\,N)_R &= \begin{cases} M_R\,N_R & \text{si } \mathrm{lvl}(M) = \mathrm{lvl}(N) = 2 \\ M_R\,N & \text{si } \mathrm{lvl}(M) = 2 \text{ et } \mathrm{lvl}(N) = 1 \end{cases} \\
\langle\rangle_R &= \langle\rangle \\
(\Gamma, x^1 : A)_R &= \Gamma_R, x^1 : A \\
(\Gamma, x^2 : A)_R &= \Gamma_R, x^2_R : A_R
\end{aligned}$$

4.1.1 Lemme
Supposons $\Gamma \vdash_{\mathcal{P}^2} M : T$.
- *Si* $\mathrm{lvl}(M) = \mathrm{lvl}(T) = 1$, *alors* $\Gamma_R \vdash_{\mathcal{P}^2} M : T$,
- *Si* $\mathrm{lvl}(M) = \mathrm{lvl}(T) = 2$, *alors* $\Gamma_R \vdash_{\mathcal{P}^2} M_R : T_R$.

Démonstration La preuve de ce lemme s'obtient facilement par induction sur la dérivation de $\Gamma \vdash_{\mathcal{P}^2} M : T$. C'est essentiellement un lemme d'α-conversion, la dérivation de typage est utile uniquement pour s'assurer que les niveaux sont bien respectés afin de pouvoir appliquer les hypothèses d'induction (par exemple, pour montrer que si $M = \lambda x^1 : A.N$ alors $\mathrm{lvl}(A) = 1$). ☺

Nous pouvons maintenant définir la relation de réalisabilité. Puisque dans les systèmes de types purs les termes et types habitent la même catégorie syntaxique, il faut définir simultanément la relation $t \Vdash P$ qui affirme que t est un réalisateur de P, et la transformation $\|\pi\|$ qui envoie les preuves π d'une formule P vers une preuve de la formule $\lfloor\pi\rfloor \Vdash P$. Nous allons à présent décrire pas à pas cette définition, et la définition 4.1.2 récapitulera tous les cas de notre discussion. Ces transformations de termes sont définies par induction, sur la structure de P pour la relation $t \Vdash P$ et sur la structure de π, pour la transformation $\|\pi\|$. Nous verrons que, même si la définition peut sembler ardue, la plupart des cas sont contraints par le théorème d'adéquation 4.1.8 qui impose que si M a le type T, alors $\|M\|$ doit avoir le type $\lfloor M\rfloor \Vdash T$.

Commençons par étudier la transformation de $t \Vdash \forall x : P.Q$ avec $\mathrm{lvl}(P) = \mathrm{lvl}(Q) = 2$. Nous voulons signifier qu'un programme t réalise $P \to Q$ s'il envoie les réalisateurs de P vers les réalisateur de Q. La formule qui semble naturellement exprimer cela est donc :

$$t \Vdash \forall x : P.Q \quad = \quad \forall x : \lfloor P\rfloor, x_R : x \Vdash P.(t\,x) \Vdash Q$$

Comme nous le constatons, la transformation a dupliqué le produit : nous sommes passé de $\forall x : P.Q$ (un produit) à $\forall x : \lfloor P\rfloor, x_R : x \Vdash P.(t\,x) \Vdash Q$ (deux produits). Nous utilisons ici le "générateur de noms frais" $\cdot \mapsto \cdot_R$ afin d'obtenir un identifiant, pour le nouveau produit généré. De plus, nous souhaitons que la réalisabilité se comporte uniformément vis-à-vis des quantificateurs de premier niveau. Ainsi, pour traiter le cas $t \Vdash \forall x : A.Q$ avec $\mathrm{lvl}(A) = 1$ et $\mathrm{lvl}(Q) = 2$, nous posons :

$$t \Vdash \forall x : A.Q \quad = \quad \forall x : A.t \Vdash Q$$

Puisque nous avons en tête de démontrer que $\vdash \pi : P$ implique $\|\pi\| : \lfloor\pi\rfloor \Vdash P$, nous allons devoir retrouver cette duplication au niveau des abstractions. Ainsi, une fois la transformation définie pour le produit, il semble inévitable de définir la transformation des abstractions de la façon suivante (on a toujours $\mathrm{lvl}(P) = \mathrm{lvl}(Q) = 2$ et $\mathrm{lvl}(A) = 1$) :

$$\|\lambda x : P.Q\| = \lambda x : \lfloor P\rfloor, x_R : x \Vdash P.\|Q\|$$

et

$$\|\lambda x : A.Q\| = \lambda x : A.\|Q\|$$

Ainsi, si $\|\pi\|$ prouve que $\lfloor\pi\rfloor \Vdash Q$, alors $\|\lambda x : P.\pi\|$ prouve que

$$\forall x : \lfloor P\rfloor, x_R : x \Vdash P.\lfloor\|\lambda x : P.\pi\|\rfloor\, x \Vdash Q$$

(car $\lfloor\|\lambda x : P.\pi\|\rfloor\, x = (\lambda x : \lfloor P\rfloor.\lfloor\|\pi\|\rfloor)\, x \equiv \lfloor\|\pi\|\rfloor$) et $\|\lambda x : A.\pi\|$ prouve que

$$\forall x : A.\lfloor\|\lambda x : A.\pi\|\rfloor \Vdash Q$$

(car $\lfloor\|\lambda x : A.\pi\|\rfloor = \lfloor\|\pi\|\rfloor$). De plus, si on constate cette duplication sur les abstractions, on comprend bien que l'on doit nécessairement la retrouver dans les applications et dans les contextes. On pose donc pour les contextes :

$$\begin{array}{rcl} \|\langle\rangle\| & = & \langle\rangle \\ \|\Gamma, x : A\| & = & \|\Gamma\|, x : A \quad \text{si } \mathrm{lvl}(A) = 1 \\ \|\Gamma, x : P\| & = & \|\Gamma\|, x : \lfloor P\rfloor, x_R : x \Vdash P \quad \text{si } \mathrm{lvl}(P) = 2 \end{array}$$

et pour les applications :

$$\|(M\,N)\| = \begin{cases} \|M\|\,N & \text{si } \mathrm{lvl}(N) = 1 \\ \|M\|\,\lfloor N\rfloor\,\|N\| & \text{si } \mathrm{lvl}(N) = 2 \end{cases}$$

De la même façon, nous n'avons guère le choix pour définir $\|x\|$. En effet, considérons la traduction du séquent : $x : A \vdash x : A$ avec $\mathrm{lvl}(A) = 2$. Le contexte $x : A$ se traduit en $x : \lfloor A\rfloor, x_R : x \Vdash A$ et doit construire une preuve $\|x\|$ de $\lfloor x\rfloor \Vdash A$, c'est-à-dire de $x \Vdash A$ puisque $\lfloor x\rfloor = x$. On voit donc bien que le seul choix "raisonnable" est de poser :

$$\|x\| \quad = \quad x_R$$

Nous allons chercher comment traduire $t \Vdash x$ et $t \Vdash \lceil s\rceil$. Ce sont probablement les deux cas les plus difficiles à appréhender. En effet, nous n'avons *a priori* aucune information sur la forme des réalisateurs d'une variable libre ; il est donc raisonnable de chercher à les construire à partir des informations du contexte. Une solution qui convient est de poser :

$$t \Vdash x \quad = \quad (x_R\,t) \text{ et } t \Vdash \lceil s\rceil \quad = \quad t \to \lceil s\rceil$$

Voyons ce qu'elle donne si l'on calcule le contexte transformé suivant :

$$\|x : \lceil s\rceil, h : x\| = x : s, x_R : x \to \lceil s\rceil, h : x, h_R : x_R\,x$$

Ainsi, le prédicat 0-aire $x : \lceil s\rceil$ se transforme en un prédicat unaire $x_R : x \to \lceil s\rceil$ qui décrit moralement l'ensemble des réalisateurs de x (avant transformation). Dans le cas où le prédicat de départ n'est pas d'arité nulle, on voit que les définitions précédemment vues "incrémentent son arité". Si x est de type $A \to \lceil s\rceil$, alors x_R sera, dans le contexte transformé, de type $x \Vdash A \to \lceil s\rceil$ c'est-à-dire de type $A \to x \to \lceil s\rceil$ ou $\forall y : \lfloor A\rfloor, y \Vdash A \to (x\,y) \to \lceil s\rceil$ selon que A soit de niveau 1 ou 2.

Pareillement, si M est de type $A \to \lceil s\rceil$ et N est de type A, alors $\|M\,N\|$ est de type $\lfloor M\,N\rfloor \Vdash (A \to \lceil s\rceil)$ c'est-à-dire de type $A \to \lfloor M\,N\rfloor \to \lceil s\rceil$ ou $\forall x : \lfloor A\rfloor, x \Vdash A \to \lfloor M\,N\rfloor \to \lceil s\rceil$ selon que $\mathrm{lvl}(A)$ es égale à 1 ou 2. On en déduit que $\|M\,N\|$ est un prédicat d'arité $\lfloor M\,N\rfloor \Vdash (A \to \lceil s\rceil)$. C'est pourquoi nous pouvons utiliser ce prédicat pour donner la signification de $t \Vdash M\,N$; on pose donc :

$$t \Vdash (M\,N) \quad = \quad (\|M\,N\|\,t)$$

Enfin, le dernier cas restant à traiter est le calcul de $\|\forall x : A.B\|$. Si on a $\forall x : A.B$ de type $\lceil s\rceil$, alors on devrait avoir $\|\forall x : A.B\|$ de type $\lfloor \forall x : A.B\rfloor \Vdash \lceil s\rceil = \lfloor \forall x : A.B\rfloor \to \lceil s\rceil$. On observe que $\|\forall x : A.B\|$ est un prédicat unaire et d'après le cas précédent, il doit être vérifié par les réalisateurs de $\forall x : A.B$. C'est pourquoi on pose :

$$\|\forall x : A.B\| = \lambda f : \lfloor \forall x : A.B\rfloor.f \Vdash \forall x : A.B$$

Dès lors, nous avons suffisamment défini de cas pour regarder ce que donne la transformation sur le prédicat $x \in \mathbb{N}$ de $\mathcal{F}^2$:

$$\begin{aligned}
r \Vdash x \in \mathbb{N} &= r \Vdash \left(\forall X : \mathsf{nat} \to \lceil\star\rceil.\, (\forall y : \mathsf{nat}. X\, y \to X\, (\mathsf{S}\, y)) \to X\, \overline{0} \to X\, x\right) \\
&= \forall X : \star, X_R : \mathsf{nat} \to X \to \lceil\star\rceil.(r\, X) \Vdash \left((\forall y : \mathsf{nat}. X\, y \to X\, (\mathsf{S}\, y)) \to X\, \overline{0} \to X\, x\right) \\
&= \forall X : \star, X_R : \mathsf{nat} \to X \to \lceil\star\rceil, f : X \to X, f_R : f \Vdash (\forall y : \mathsf{nat}. X\, y \to X\, (\mathsf{S}\, y))\,. \\
&\qquad (r\, X\, f) \Vdash \left(X\, \overline{0} \to X\, x\right) \\
&= \forall X : \star, X_R : \mathsf{nat} \to X \to \lceil\star\rceil, f : X \to X, f_R : f \Vdash (\forall y : \mathsf{nat}. X\, y \to X\, (\mathsf{S}\, y))\,, \\
&\qquad z : X, z_R : X_R\, \overline{0}\, z.(r\, X\, f\, z) \Vdash X\, x \\
&= \forall X : \star, X_R : \mathsf{nat} \to X \to \lceil\star\rceil, f : X \to X, f_R : f \Vdash (\forall y : \mathsf{nat}. X\, y \to X\, (\mathsf{S}\, y))\,, \\
&\qquad z : X, z_R : X_R\, \overline{0}\, z. X_R\, x\, (r\, X\, f\, z) \\
&= \forall X : \star, X_R : \mathsf{nat} \to X \to \lceil\star\rceil, f : X \to X, \\
&\qquad f_R : (\forall y : \mathsf{nat}, z : X, z_R : X_R\, y\, z. X_R\, (\mathsf{S}\, y)\, (f\, z))\,, \\
&\qquad z : X, z_R : X_R\, \overline{0}\, z. X_R\, x\, (r\, X\, f\, z) \\
&= \forall \alpha : \star, X : \mathsf{nat} \to \alpha \to \lceil\star\rceil, f : \alpha \to \alpha. \\
&\qquad (\forall y : \mathsf{nat}, z : \alpha. X\, y\, z \to X\, (\mathsf{S}\, y)\, (f\, z)) \to \forall z : \alpha. X_R\, \overline{0}\, z \to X\, x\, (r\, \alpha\, f\, z)
\end{aligned}$$

Afin de rendre la formule plus lisible, la dernière égalité est obtenue en renommant les variables liées et en utilisant la notation "$\cdot \to \cdot$" pour les produits non-dépendants. Par la suite, lorsque nous "déplierons" les formules de réalisabilité, nous renommerons souvent les variables "à la volée", privilégiant ainsi la lecture de la formule (au détriment du calcul de la forme dépliée).

Ainsi, on peut paraphraser le prédicat $r \Vdash x \in \mathbb{N}$ de la façon suivante :

> Pour tout type α, tout prédicat binaire X d'arité $\mathsf{nat} \to \alpha \to \lceil\star\rceil$, pour toute fonction f de type $f : \alpha \to \alpha$, et pour tout z de type α, si les deux conditions suivantes sont vérifiées :
>
> 1. Le prédicat X est stable par S à gauche, et f à droite (c'est-à-dire si $X\, y\, z$ est vérifiée alors $X\, (\mathsf{S}\, y)\, (f\, z)$ est vérifiée),
> 2. Le prédicat X est satisfait par $\overline{0}$ et z (c'est-à-dire on a $X\, \overline{0}\, z$)
>
> alors le prédicat est X est satisfait par x et $r\, \alpha\, f\, z$.

On voit que si l'on prend $\alpha = \mathsf{nat}$, $f = \mathsf{S}$, $z = \overline{0}$ et $X = \lambda a\, b : \mathsf{nat}.a =_{\mathsf{nat}} b$, alors les conditions 1 et 2 sont vérifiées. On peut en déduire que $x =_{\mathsf{nat}} r\, \mathsf{nat}\, \mathsf{S}\, \overline{0}$. Par ailleurs, on verra plus loin que si l'on dispose d'un axiome d'extensionalité, alors on peut montrer $r\, \mathsf{nat}\, \mathsf{S}\, \overline{0} =_{\mathsf{nat}} r$ et déduire que $r \Vdash x \in \mathbb{N}$ implique $r =_{\mathsf{nat}} x$ ou, en d'autres termes, que les réalisateurs de $x \in \mathbb{N}$ sont (extensionellement) égaux à x.

On peut récapituler par la définition suivante :

4.1.2 Définition (Relation de réalisabilité)
On définit simultanément la relation de réalisabilité, notée $\cdot \Vdash \cdot$, et la transformation de réalisabilité,

notée $\|\cdot\|$, par les équations suivantes.

$$\begin{array}{lcl}
C \Vdash \lceil s \rceil & = & C \to \lceil s \rceil \\
C \Vdash \forall x^1 : A.B & = & \forall x^1 : A.C \Vdash B \\
C \Vdash \forall x^2 : A.B & = & \forall x^1 : \lfloor A \rfloor, x^2_R : x^1 \Vdash A.(C\, x^1) \Vdash B \\
C \Vdash F & = & \|F\|\, C \text{ sinon}
\end{array}$$

$$\begin{array}{lcl}
\|x^2\| & = & x^2_R \\
\|\lambda x^1 : A.B\| & = & \lambda x^1 : A.\|B\| \\
\|\lambda x^2 : A.B\| & = & \lambda x^1 : \lfloor A \rfloor, x^2_R : x^1 \Vdash A.\|B\| \\
\|(A\, B)\| & = & \begin{cases} (\|A\|\, B) & \text{si } \mathrm{lvl}(B) = 1 \\ (\|A\|\, \lfloor B \rfloor\, \|B\|) & \text{si } \mathrm{lvl}(B) = 2 \end{cases} \\
\|T\| & = & \lambda z^1 : \lfloor T \rfloor . z^1 \Vdash T \text{ sinon} \qquad (z^1 \notin \mathcal{FV}(T) \cup \mathcal{FV}(\lfloor T \rfloor))
\end{array}$$

$$\begin{array}{lcl}
\|\langle\rangle\| & = & \langle\rangle \\
\|\Gamma, x^1 : A\| & = & \|\Gamma\|, x^1 : A \\
\|\Gamma, x^2 : A\| & = & \|\Gamma\|, x^1 : \lfloor A \rfloor, x^2_R : x^1 \Vdash A
\end{array}$$

Cette transformation "conserve la structure du second niveau" dans le sens suivant :

4.1.3 Lemme (Projection de la réalisabilité)
Soit t tel que $\mathrm{lvl}(t) = 1$ et P tel que $\mathrm{lvl}(P) = 2$, on a $\lfloor t \Vdash P \rfloor = \lfloor P_R \rfloor$ et pour tout contexte $\lfloor \|\Gamma\| \rfloor = \lfloor \Gamma_R \rfloor$.

Démonstration Par une induction immédiate sur les formes possibles de P. ☺

Le lemme ci-dessous décrit comment la substitution des termes de premier niveau se comporte au travers des transformations de réalisabilité. On notera la condition "d'hygiène" qui impose que la variable substituée n'apparaît avec l'annotation de second niveau que dans le terme à transformer. Ce qui est en pratique vérifié lorsque l'on s'assure que les termes ne contiennent pas deux occurrences de la même variable annotées différemment.

4.1.4 Lemme (Substitution des termes de premier niveau)
Soit A un terme de second niveau tel que $x^2 \notin \mathcal{FV}(A)$ et B et M deux termes de premier niveau :

1. $\|A\|[M/x^1] = \|A[M/x^1]\|$,
2. $(B \Vdash A)\,[M/x^1] = (B[M/x^1]) \Vdash (A[M/x^1])$,

Démonstration On procède par induction sur la structure de A. La démonstration consiste essentiellement à réécrire les substitutions à l'aide des définitions, des hypothèses d'inductions et du lemme 3.4.5 pour réécrire sous les projections. ☺

La substitution des termes de second niveau nécessite quant à elle une ou plusieurs étapes de réduction. Par exemple le terme

$$B \Vdash x^2[\lceil s \rceil / x^2] = B \Vdash \lceil s \rceil = B \to \lceil s \rceil$$

n'est pas égal au terme

$$(B \Vdash x^2)[s/x^1, \|\lceil s \rceil\|/x^2_R] = x^2_R\, B[s/x^1, \|\lceil s \rceil\|/x^2_R] = \|\lceil s \rceil\|\, B(\text{si } x^1 \notin \mathcal{FV}(B))$$

Il lui est néanmoins convertible ; en effet, on a $\|\lceil s \rceil\|\, B = (\lambda z : s.z \to \lceil s \rceil)\, B \twoheadrightarrow B \to \lceil s \rceil$.

4.1.5 Lemme (Substitution des termes de second niveau)
Soient A un terme de second niveau tel que $x^1 \notin \mathcal{FV}(A)$, B un terme de premier niveau et M un terme de second niveau. Alors,

1. $\|A\|[\lfloor M\rfloor/x^1, \|M\|/x^2_R] \trianglerighteq \|A[M/x^2]\|$,
2. $(B \Vdash A)\,[\lfloor M\rfloor/x^1, \|M\|/x^2_R] \trianglerighteq (B[\lfloor M\rfloor/x^1]) \Vdash (A[M/x^2])$.

Démonstration On procède par induction sur la structure de A. Le cas de base $A = x^2$ se traite par une égalité dans les cas où M est une variable, une application ou une abstraction, et par une réduction dans les autres cas. Ce cas est le seul qui puisse éventuellement introduire une étape de réduction. Les autres cas consistent essentiellement à propager les hypothèses de récurrence en récrivant les substitutions à l'aide de la condition "d'hygiène" $x^1 \notin \mathcal{FV}(A)$ en pratique toujours vérifiée et du lemme 3.4.5 de substitution de la projection. ☺

Le lemme suivant énonce les propriétés de la transformation de réalisation vis-à-vis des réductions annotées. Ces résultats sont faux en l'absence de contraintes de niveaux lors des réductions. On rappelle toutefois que si le terme A et A' sont bien typés, alors les relations annotées et les relations non-annotées sont équivalentes (lemmes 3.3.20, 3.3.21, 3.3.22).

4.1.6 Lemme

1. *Si $A \twoheadrightarrow^+ A'$, alors*
   (a) *$\|A\| \rhd^+ \|A'\|$ et*
   (b) *si $B \rhd^+ B'$, alors $B \Vdash A \rhd^+ B' \Vdash A'$.*
2. *Si $A \rhd^+ A'$, alors $\|A\| \rhd^+ \|A'\|$.*
3. *Si $A \rhd^+ A'$ et $B \rhd^+ B'$, alors $B \Vdash A \rhd^+ B' \Vdash A'$.*
4. *Si $A \equiv^+ A'$, alors $\|A\| \equiv^+ \|A'\|$.*
5. *Si $A \equiv^+ A'$ et $B \equiv^+ B'$, alors $B \Vdash A \equiv^+ B' \Vdash A'$.*
6. *Si $A \prec^+ A'$ et $B \Vdash A \prec^+ B \Vdash A'$.*
7. *Si $A \preccurlyeq^+ A'$ et $B \Vdash A \preccurlyeq^+ B \Vdash A'$.*

Démonstration Pour alléger les notations dans cette preuve, on omettra les exposants $\cdot^+$ indiquant que les relations de réduction sont annotées.

1. On procède par induction sur les règles définissant $A \twoheadrightarrow A'$:
   – Si A est de la forme $(\lambda x^1 : A_1.A_2)\,A_3$ et A' de la forme $A_2[A_3/x^1]$ avec $\mathrm{lvl}(A_3) = 1$.
   (a) On a : $\|A\| = (\lambda x^1 : A_1.\|A_2\|)\,A_3 \twoheadrightarrow \|A_2\|[A_3/x^1]$. Or d'après le lemme 4.1.4 on a $\|A_2\|[A_3/x^1] = \|A_2[A_3/x^1]\|$. On a donc bien $\|A\| \rhd \|A'\|$.
   (b) Supposons $B \rhd B'$. On a $B \Vdash A = \|A\|\,B$ et donc d'après le cas précédent, on a $\|A\|\,B \rhd \|A'\|\,B'$. On distingue les formes possibles de A' :
   – Soit A' est une sorte ou un produit : Dans ce cas $\|A'\| = \lambda z : \lfloor A'\rfloor.z \Vdash A'$ avec $z \notin \mathcal{FV}(A')$ et donc $B \Vdash A \rhd \|A'\|\,B' = (\lambda z : \lfloor A'\rfloor.z \Vdash A')\,B' \rhd B' \Vdash A'$.
   – Soit A' est une application, une variable ou une abstraction. On a $B' \Vdash A' = \|A'\|\,B'$ et donc $B \Vdash A \rhd \|A'\|\,B' = B' \Vdash A'$.
   – Si A est de la forme $(\lambda x^2 : A_1.A_2)\,A_3$ et A' de la forme $A_2[A_3/x^2]$ avec $\mathrm{lvl}(A_3) = 2$.

(a) On a :

$$\|A\| = (\lambda x^1 : \lfloor A_1 \rfloor, x_R^2 : x^1 \Vdash A_1.\|A_2\|)\ \lfloor A_3 \rfloor\ \|A_3\| \twoheadrightarrow \|A_2\|[\lfloor A_3 \rfloor / x^1, \|A_3\| / x_R^2]$$

Or d'après le lemme 4.1.5 on a $\|A_2\|[\lfloor A_3 \rfloor / x^1, \|A_3\| / x_R^2] \trianglerighteq \|A_2[A_3/x^2]\|$. On a donc bien $\|A\| \rhd \|A'\|$.

(b) Similaire au cas (b) précédent.

– Les autres cas de l'induction s'effectuent facilement en distinguant selon les niveaux afin de déplier les définitions de réalisabilité, et à l'aide des lemmes 4.1.4, 4.1.5 et 3.4.7.

2,3,4,5. Ce sont des conséquences immédiates du cas 1

6. On montre par récurrence sur n que $A \prec_n A'$ implique pour tout terme B de premier niveau $B \Vdash A \prec B \Vdash A'$.
   – Si $A \prec_0 A'$ alors il existe $(s_A, s'_A) \in \mathcal{C}_\mathcal{P}$ telles que $A \equiv \lceil s_A \rceil$ et $A' \equiv \lceil s'_A \rceil$. D'après le cas 5, on en déduit que $B \Vdash A \equiv B \to \lceil s_A \rceil$ et $B \Vdash A' \equiv B \to \lceil s'_A \rceil$. Or on a $B \to \lceil s_A \rceil \ \prec B \to \lceil s'_A \rceil$ et donc d'après le lemme 1.1.7 on en conclut $B \Vdash A \prec B \Vdash A'$.
   – Supposons $A \prec_{n+1} A'$, alors par définition :
     – Soit on a $A \prec_n A'$ et on conclut par hypothèse de récurrence.
     – Soit il existe D, D' et C tels que $A \equiv \forall x : C.D$ et $A' \equiv \forall x : C.D'$ avec $D \prec_n D'$. Par hypothèse de récurrence, on en déduit pour tout B de premier niveau, $B \Vdash D \prec B \Vdash D'$ (H). Soit B un terme de premier niveau. D'après le cas 5, on a $B \Vdash A \equiv B \Vdash \forall x : C.D$ et $B \Vdash A' \equiv B \Vdash \forall x : C.D'$. On distingue deux cas selon le niveau de C :
       – $\mathrm{lvl}(C) = 1$: Dans ce cas, $B \Vdash \forall x : C.D = \forall x : C.B \Vdash D$ et $B \Vdash \forall x : C.D' = \forall x : C.B \Vdash D'$ on a donc bien d'après (H) :

$$B \Vdash \forall x : C.D = \forall x : C.B \Vdash D \prec \forall x : C.B \Vdash D' = B \Vdash \forall x : C.D'$$

       D'après le lemme 1.1.7, on en déduit bien $B \Vdash A \prec B \Vdash A'$.
       – $\mathrm{lvl}(C) = 2$: Ici, $B \Vdash \forall x : C.D = \forall x : \lfloor C \rfloor, x_R : x \Vdash C.B\,x \Vdash D$ et $B \Vdash \forall x : C.D' = \forall x : \lfloor C \rfloor, x_R : x \Vdash C.B\,x \Vdash D'$ Or on peut appliquer (H) avec $B\,x$ pour obtenir $B\,x \Vdash D \ \prec B\,x \Vdash D'$ et donc :

$$\forall x : \lfloor C \rfloor, x_R : x \Vdash C.B\,x \Vdash D \prec \forall x : \lfloor C \rfloor, x_R : x \Vdash C.B\,x \Vdash D'$$

       C'est-à-dire $B \Vdash \forall x : C.D \prec B \Vdash \forall x : C.D'$ et donc d'après le lemme 1.1.7, $B \Vdash A \prec B \Vdash A'$.

7. Conséquence immédiate de 5 et 6. ☺

4.1.7 Lemme

On a $\mathrm{filter}(\Gamma) \subseteq \|\Gamma\|$ *et* $\lfloor \Gamma \rfloor \subseteq \|\Gamma\|$.

Démonstration Par induction sur la longueur de Γ.

– $\Gamma = \langle\rangle$: Dans ce cas, $\mathrm{filter}(\Gamma) = \|\Gamma\| = \lfloor \Gamma \rfloor = \langle\rangle$.
– $\Gamma = \Gamma', x^1 : A$: Dans ce cas, on a $\mathrm{filter}(\Gamma) = \mathrm{filter}(\Gamma'), x : A$, $\lfloor \Gamma \rfloor = \lfloor \Gamma' \rfloor$ et $\|\Gamma\| = \|\Gamma'\|, x : A$. Or par hypothèse d'induction on a $\mathrm{filter}(\Gamma') \subseteq \|\Gamma'\|$ et $\lfloor \Gamma' \rfloor \subseteq \|\Gamma'\|$. On en déduit donc bien $\mathrm{filter}(\Gamma) \subseteq \|\Gamma\|$ et $\lfloor \Gamma \rfloor \subseteq \|\Gamma\|$.

– $\Gamma = \Gamma', x^2 : A$: Dans ce cas, on a $\text{filter}(\Gamma) = \text{filter}(\Gamma')$, $\lfloor\Gamma\rfloor = \lfloor\Gamma'\rfloor, x : \lfloor A\rfloor$ et $\|\Gamma\| = \|\Gamma'\|, x : \lfloor A\rfloor, x_R : x \Vdash A$. Par hypothèse d'induction on a $\text{filter}(\Gamma') \subseteq \|\Gamma'\|$ et $\lfloor\Gamma'\rfloor \subseteq \|\Gamma'\|$. On en déduit bien $\text{filter}(\Gamma) \subseteq \|\Gamma\|$ et $\lfloor\Gamma\rfloor \subseteq \|\Gamma\|$. ☺

Nous avons à présent développé tous les lemmes nécessaire pour prouver le théorème d'adéquation ci-dessous. La preuve consiste en une longue vérification que la transformation de termes typées demeure typée. L'induction procède par induction sur la structure des dérivations. Or, comme les construction $\|\cdot\|$ et de $\cdot \Vdash \cdot$ sont mutuellement définies, il nous sera nécessaire prouver simultanément que ces deux transformations préserve le typage (ce qui explique en grande partie la longueur de la démonstration. La démonstration ne pose pas de problème particulier, une fois l'énoncé du théorème établi, chaque cas consiste en une simple vérification que le séquent transformé est dérivable.

4.1.8 Théorème (Théorème d'adéquation de la réalisabilité)
Soit $\mathcal{P}$ un CTS tel que $\mathcal{P}^2 \models$ STRENGTHENING $\wedge$ WEAK-DOWNSTABLE.
Si $\Gamma \vdash M : T$, alors

$$\begin{cases} \text{lvl}(M) = \text{lvl}(T) = 1 & \Rightarrow & \|\Gamma\| \vdash M : T & (1) \\ \Gamma \vdash T : \lceil s\rceil & \Rightarrow & \|\Gamma\| \vdash \|M\| : \lfloor M\rfloor \Vdash T & (2) \\ T \equiv \lceil s\rceil & \Rightarrow & \|\Gamma\|, x : \lfloor M\rfloor \vdash x \Vdash M : \lceil s\rceil & (3) \end{cases}$$

Démonstration Par induction sur la structure de la dérivation de $\Gamma \vdash M : T$, on notera $\mathbf{HI}_1$, $\mathbf{HI}_2$ et $\mathbf{HI}_3$ les hypothèses d'induction associées à chacune des implications. On traite séparément le cas où $\text{lvl}(M) = \text{lvl}(T) = 1$ et le cas où $\text{lvl}(M) = \text{lvl}(T) = 2$.

Supposons $\text{lvl}(M) = \text{lvl}(T) = 1$.

Les implications (2) et (3) sont triviales car les prémisses sont impossibles si M et T sont des termes de premier niveau. Pour montrer (1) dans chacun des cas de l'induction, on remarque qu'il suffit de prouver $\text{WF}(\|\Gamma\|)$ (0). En effet, on utilise le fait que, d'après le lemme 3.4.2, $\Gamma \vdash M : T$ implique que l'on a $\text{filter}(\Gamma) \vdash M : T$ $(*)$. Or par définition de $\text{filter}(\cdot)$ et de $\|\cdot\|$ on a $\text{filter}(\Delta) \subseteq \|\Delta\|$ (lemme précédent). D'après le lemme 1.1.27 appliqué à $(*)$ et (0), on obtient $\|\Gamma\| \vdash M : T$.
Montrons maintenant que dans chacun des cas de l'induction, nous avons $\text{WF}(\|\Gamma\|)$. Le cas Axiome est immédiat car dans ce cas $\|\Gamma\| = \|\langle\rangle\| = \langle\rangle$. Les cas Application, Abstraction, Produit, Cumulativité, et Cumulativité-Sortes s'obtiennent directement en appliquant l'hypothèse d'induction puisque ces règles ne modifient pas le contexte.

Il reste donc à traiter les cas Variable et Affaiblissement :

– Variable : Dans ce cas, on a $\Gamma = \Delta, y : T$, avec $M = y$ et $\Delta \vdash T : t$. On a ici $\|\Gamma\| = \|\Delta\|, y : T$ (on est dans le cas où $\text{lvl}(T) = 1$). Or par l'hypothèse d'induction $\mathbf{HI}_1$ associée à $\Delta \vdash T : t$, on a $\|\Delta\| \vdash T : t$ et on en déduit bien $\text{WF}(\|\Delta\|, y : T)$.
– Affaiblissement : Dans ce cas, on a $\Gamma = \Delta, y : A$ avec $\Delta \vdash M : T$ et $\Delta \vdash A : t$. Il faut ici distinguer deux cas :
  – $\text{lvl}(A) = \text{lvl}(t) = 1$: On a dans ce cas, $\|\Gamma\| = \|\Delta\|, y : A$ et on peut appliquer l'hypothèse d'induction $\mathbf{HI}_1$ associée à $\Delta \vdash A : t$ pour obtenir que $\|\Delta\| \vdash A : t$ et donc que $\text{WF}(\|\Delta\|, y : A)$.

– $\mathrm{lvl}(A) = \mathrm{lvl}(t) = 2$: Ici, $\|\Gamma\| = \|\Delta\|, y : \lfloor A \rfloor, y_R : y \Vdash A$. On peut donc appliquer l'hypothèse d'induction $\mathbf{HI}_3$ associée à $\Delta \vdash A : t$ pour montrer que $\|\Delta\|, y : \lfloor A \rfloor \vdash y \Vdash A : t$ et en déduire que $\mathrm{WF}(\|\Delta\|, y : \lfloor A \rfloor, y_R : y \Vdash A)$.

Supposons $\mathrm{lvl}(M) = \mathrm{lvl}(T) = 2$.

Dans ce cas, l'implication (1) est triviale car sa prémisse est impossible si M et T sont des termes de second niveau.

– Axiome : Il existe $(\lceil r \rceil, \lceil t \rceil) \in \mathcal{A}_{\mathcal{P}}$ (H0) avec $M = \lceil r \rceil$ et $T = \lceil t \rceil$. Par construction de $\mathcal{P}^2$, nous en déduisons : $(r, t) \in \mathcal{A}_{\mathcal{P}^2}$ (H1) et $(r, \lceil t \rceil, \lceil t \rceil) \in \mathcal{R}_{\mathcal{P}^2}$ (H2).
  – Implication (2) : Supposons que $\vdash \lceil t \rceil : \lceil s \rceil$, alors d'après le lemme d'inversion on montre qu'il existe $s' \preccurlyeq s$ telle que $(\lceil t \rceil, \lceil s' \rceil) \in \mathcal{A}_{\mathcal{P}^2}$ (H3). On en déduit donc par construction de $\mathcal{P}^2$ que :
    – (H4) : $(t, s') \in \mathcal{A}_{\mathcal{P}^2}$,
    – (H5) : $(t, \lceil s' \rceil, \lceil s' \rceil) \in \mathcal{R}_{\mathcal{P}^2}$.

    On a :
    $$\|M\| = \lambda x : \lfloor \lceil r \rceil \rfloor . x \to \lceil r \rceil = \lambda x : r . x \to \lceil r \rceil$$
    et :
    $$\lfloor M \rfloor \Vdash T = r \to \lceil t \rceil$$
    On montre bien $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash T$ à l'aide des dérivations suivantes :
    $$\dfrac{\dfrac{\dfrac{\dfrac{(\mathrm{H1})\ \dfrac{}{\vdash r : t}}{x : r \vdash x : r} \quad \dfrac{\quad}{x : r,\ \_ : x \vdash \lceil r \rceil : \lceil t \rceil}\ (\mathrm{H1}) \text{ et } (\mathrm{H0})}{x : r \vdash x \to \lceil r \rceil : \lceil t \rceil}\ (\mathrm{H2}) \qquad \vdash r \to \lceil t \rceil : \lceil s' \rceil}{\vdash \|M\| : \lfloor M \rfloor \Vdash T}$$
    Et de :
    $$\dfrac{(\mathrm{H1})\ \dfrac{}{\vdash r : t} \qquad \dfrac{(\mathrm{H3})\ \dfrac{}{\vdash \lceil t \rceil : \lceil s' \rceil} \quad (\mathrm{H1})\ \dfrac{}{\vdash r : t}}{\_ : r \vdash \lceil t \rceil : \lceil s' \rceil}}{(\mathrm{H5})\ \vdash r \to \lceil t \rceil : \lceil s' \rceil}$$
  – Implication (3) : Si $T \equiv \lceil s \rceil$, on a donc $t = s$. On doit alors montrer que $x : \lfloor M \rfloor \vdash x \Vdash M : \lceil t \rceil$. Nous avons ici $\lfloor M \rfloor = r$ et $x \Vdash M = x \to \lceil r \rceil$.
    $$(\mathrm{H2})\ \dfrac{\dfrac{}{x : r \vdash x : r} \qquad \dfrac{(\mathrm{H1})\ \dfrac{}{\vdash r : t} \quad (\mathrm{H0})\ \dfrac{}{\vdash \lceil r \rceil : \lceil t \rceil}}{x : r \vdash \lceil r \rceil : \lceil t \rceil}}{x : \lfloor M \rfloor \vdash x \Vdash M : \lceil t \rceil}$$

– Variable : On a $\Gamma = \Delta, x : T$ et $M = x$ avec $\Delta \vdash T : \lceil t \rceil$ pour un certain t. Par hypothèse d'induction, on a $\|\Delta\|, x : \lfloor T \rfloor \vdash x \Vdash T : \lceil t \rceil$ (H) et on en déduit $\mathrm{WF}(\|\Delta\|)$ (H′).
  – Implication (2) : Supposons $\Gamma \vdash T : \lceil s \rceil$ (on n'utilisera pas cette hypothèse). On a $\|\Gamma\| = \|\Delta\|, x : \lfloor T \rfloor, x_R : x \Vdash T$ avec $\|M\| = x_R$ et $\lfloor M \rfloor = x$. On peut donc utiliser la règle Variable avec (H) pour montrer $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash T$.

– Implication (3) : Supposons $T \equiv \lceil s \rceil$. D'après le lemme 3.4.8, on a $\lfloor \Delta \rfloor \vdash \lfloor T \rfloor : t$. Or, d'après le lemme précédent on a $\lfloor \Delta \rfloor \subseteq \|\Delta\|$ et, d'après le lemme 1.1.27 et (H′), on en déduit $\|\Delta\| \vdash \lfloor T \rfloor : t$ (H″).
De plus, $T \equiv \lceil s \rceil$ implique, par confluence, $T \trianglerighteq \lceil s \rceil$ et d'après le lemme 3.4.7, $\lfloor T \rfloor \trianglerighteq s$.
On a $y \Vdash M = (x_R\ y)$, on doit donc montrer $\|\Delta\|, x : T, x_R : x \Vdash T, y : x \vdash (x_R\ y) : \lceil s \rceil$ et on procède en suivant l'arbre de dérivation suivant :

$$\dfrac{\dfrac{(i)}{\|\Delta, x : T\|, y : x \vdash x_R : x \to \lceil s \rceil} \quad \dfrac{\dfrac{(ii)}{\|\Delta, x : T\| \vdash x : s}}{\|\Delta, x : T\|, y : x \vdash y : x}}{\|\Delta, x : T\|, y : x \vdash x_R\, y : \lceil s \rceil}$$

où le séquent (i) $\|\Delta, x : T\|, y : x \vdash x_R : x \to \lceil s \rceil$ est obtenu par :

$$\dfrac{\dfrac{(iii)}{\|\Delta, x : T\| \vdash x_R : x \to \lceil s \rceil} \quad \dfrac{(ii)}{\|\Delta, x : T\| \vdash x : s}}{\|\Delta, x : T\|, y : x \vdash x_R : x \to \lceil s \rceil}\ (i)$$

et le séquent (ii) $\|\Delta, x : T\| \vdash x : s$ par :

$$\dfrac{\dfrac{\dfrac{\dfrac{(\mathrm{H}'')}{\|\Delta\| \vdash \lfloor T \rfloor : t'}}{\|\Delta\|, x : \lfloor T \rfloor \vdash x : \lfloor T \rfloor}}{\|\Delta\|, x : \lfloor T \rfloor \vdash x : s}\ \text{lemme 1.1.47} \quad \dfrac{(\mathrm{H})}{\|\Delta\|, x : \lfloor T \rfloor \vdash x \Vdash T : \lceil t \rceil}}{\|\Delta, x : T\| \vdash x : s}\ (ii)$$

et le séquent (iii) $\|\Delta, x : T\| \vdash x_R : x \to \lceil s \rceil$ par :

$$\dfrac{\dfrac{(\mathrm{H})}{\|\Delta\|, x : \lfloor T \rfloor \vdash x \Vdash T : \lceil t \rceil}}{\|\Delta\|, x : \lfloor T \rfloor, x_R : x \Vdash T \vdash x_R : x \Vdash T} \quad \dfrac{\dfrac{\dfrac{(\mathrm{H})}{\|\Delta\|, x : \lfloor T \rfloor \vdash x \Vdash T : \lceil t \rceil}}{\|\Delta\|, x : \lfloor T \rfloor \vdash x \to \lceil s \rceil : \lceil t \rceil}\ \text{Red. Sujet}}{\|\Delta\|, x : \lfloor T \rfloor, x_R : x \Vdash T \vdash x \to \lceil s \rceil : \lceil t \rceil}\ \text{Aff.} + (\mathrm{H})}{\|\Delta, x : T\| \vdash x_R : x \to \lceil s \rceil}\ (\mathrm{H})\ (iii)$$

– Affaiblissement : On $\Gamma = \Delta, y : A$ avec $\Delta \vdash M : T$ et $\Delta \vdash A : t$. En appliquant $\mathbf{HI}_1$ et $\mathbf{HI}_2$ on montre les faits suivants :

$$\begin{array}{lcll} \mathrm{lvl}(t) = 1 & \Rightarrow & \|\Delta\| \vdash A : t & (H1) \\ \mathrm{lvl}(t) = 2 & \Rightarrow & \|\Delta\|, y : \lfloor A \rfloor \vdash y \Vdash A : t & (H2) \end{array}$$

– Implication (2) : Supposons $\Gamma \vdash T : \lceil s \rceil$. Nous avons $y \notin \mathcal{FV}(T)$ et d'après $\mathcal{P} \Vdash$ STRENGTHENING, on en déduit que $\Delta \vdash T : \lceil s \rceil$. Et donc par hypothèse d'induction $\mathbf{HI}_2$: $\|\Delta\| \vdash \|M\| : \lfloor M \rfloor \Vdash T$ (*). On distingue deux cas :

- $\mathrm{lvl}(t) = 1$: On a $\|\Gamma\| = \|\Delta\|, y : A$. On peut appliquer la règle Affaiblissement avec (*) et (H1), pour obtenir $\|\Delta\|, y : A \vdash \|M\| : \lfloor M \rfloor \Vdash T$.
- $\mathrm{lvl}(t) = 2$: Ici, $\|\Gamma\| = \|\Delta\|, y : \lfloor A \rfloor, y_R : y \Vdash A$. On peut donc appliquer successivement deux fois la règle Affaiblissement, à l'aide de (*) et (H2), pour montrer $\|\Delta\|, y : \lfloor A \rfloor, y_R : y \Vdash A \vdash \|M\| : \lfloor M \rfloor \Vdash T$.

– Implication (3) : Supposons $T \equiv \lceil s \rceil$. On doit montrer que $\|\Gamma\|, x : \lfloor M \rfloor \vdash x \Vdash M : \lceil s \rceil$. Par hypothèse d'induction $\mathbf{HI}_3$, nous avons $\|\Delta\|, x : \lfloor M \rfloor \vdash x \Vdash M : \lceil s \rceil$ (*). Tout comme pour montrer l'implication précédente, on distingue deux cas :
  - $\mathrm{lvl}(t) = 1$, dans ce cas on a $\|\Gamma\| = \|\Delta\|, x : A$. On peut appliquer le lemme 1.1.24 avec (*) et (H1) pour montrer $\|\Delta\|, y : A, x : \lfloor M \rfloor \vdash x \Vdash M : \lceil s \rceil$.
  - $\mathrm{lvl}(t) = 2$, dans ce cas on a $\|\Gamma\| = \|\Delta\|, y : \lfloor A \rfloor, y_R : y \Vdash A$ et donc nous pouvons appliquer deux fois le lemme 1.1.24 à l'aide de (*) et (H2) pour montrer $\|\Delta\|, y : \lfloor A \rfloor, y_R : y \Vdash A, x : \lfloor M \rfloor \vdash x \Vdash M : \lceil s \rceil$.

– Produit : On a $M = \forall x : M_1.M_2$ et $T = \lceil t \rceil$ avec $\Gamma \vdash M_1 : t_1$ et $\Gamma, x : M_1 \vdash M_2 : \lceil t_2 \rceil$ avec $\mathrm{lvl}(M_2) = 2$ et $(t_1, \lceil t_2 \rceil, \lceil t \rceil) \in \mathcal{R}_{\mathcal{P}^2}$ (H0).
  - Implication (2) : Supposons $\Gamma \vdash T : \lceil s \rceil$. On en déduit par inversion, qu'il existe $s' \preccurlyeq s$ telle que $(\lceil t \rceil, \lceil s' \rceil) \in \mathcal{A}_{\mathcal{P}^2}$ (H1) et par construction de $\mathcal{P}^2$ que $(t, s') \in \mathcal{A}_{\mathcal{P}^2}$ et $(t, \lceil s' \rceil, \lceil s' \rceil) \in \mathcal{R}_{\mathcal{P}^2}$ (H2). On distingue deux cas :
    - $\mathrm{lvl}(t_1) = 1$: Par construction de $\mathcal{P}^2$, on en déduit que $t_2 = t$ (en effet, si $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}^2}$ avec $\mathrm{lvl}(s_1) = 1$ et $\mathrm{lvl}(s_2) = \mathrm{lvl}(s_3)$, alors $s_2 = s_3$). On a
    $$\|M\| \quad = \quad \lambda f : \lfloor M \rfloor . f \Vdash M \quad = \quad \lambda f : \lfloor M_2 \rfloor . \forall x : M_1 . f \Vdash M_2$$
    or par hypothèse d'induction $\mathbf{HI}_3$ on a $\|\Gamma\|, x : M_1, f : \lfloor M_2 \rfloor \vdash f \Vdash M_2 : \lceil t_2 \rceil$ et on a $x \notin \lfloor M_2 \rfloor$ car $\mathrm{lvl}(x) = 1$ et x est effacé par la projection, on peut donc utiliser le lemme 1.1.49 (on utilise donc ici STRENGTHENING) pour montrer $\|\Gamma\|, f : \lfloor M_2 \rfloor, x : M_1 \vdash f \Vdash M_2 : \lceil t_2 \rceil$. D'après le lemme 3.4.8, $\lfloor \Gamma \rfloor \vdash \lfloor M_2 \rfloor : t_2$ et donc d'après le lemme précédent et le lemme 1.1.27, on en déduit $\|\Gamma\| \vdash \lfloor M_2 \rfloor : t_2$. Or d'après $\mathbf{HI}_1$, nous avons $\|\Gamma\| \vdash M_1 : t_1$ et donc $\|\Gamma\|, f : \lfloor M_2 \rfloor \vdash M_1 : t_1$ (d'après la règle Affaiblissement). D'après (H0) et la règle Produit, $\|\Gamma\|, f : \lfloor M_2 \rfloor \vdash \forall x : M_1.f \Vdash M_2 : \lceil t \rceil$. On a $\|\Gamma\| \vdash \lfloor M_2 \rfloor \to \lceil t \rceil : \lceil s' \rceil$ grâce à la règle Produit et (H2). La règle Abstraction nous permet bien de conclure $\|\Gamma\| \vdash \lambda f : \lfloor M_2 \rfloor . \forall x : M_1 . f \Vdash M_2 : \lfloor M_2 \rfloor \to \lceil t \rceil$ or $\lfloor M_2 \rfloor \to \lceil t \rceil = \lfloor M \rfloor \Vdash T$. Ce que l'on peut résumer par l'arbre de dérivation suivant :
    $$\dfrac{\dfrac{\|\Gamma\| \vdash M_1 : t_1 \quad \|\Gamma\| \vdash \lfloor M_2 \rfloor : t_2}{\text{(H0)}\ \dfrac{\|\Gamma\|, f : \lfloor M_2 \rfloor \vdash M_1 : t_1 \quad \|\Gamma\|, f : \lfloor M_2 \rfloor, x : M_1 \vdash f \Vdash M_2 : \lceil t_2 \rceil}{\text{(H2)}\ \dfrac{\|\Gamma\|, f : \lfloor M_2 \rfloor \vdash \forall x : M_1.f \Vdash M_2 : \lceil t \rceil \qquad \|\Gamma\| \vdash \lfloor M_2 \rfloor \to \lceil t \rceil : \lceil s' \rceil}{\|\Gamma\| \vdash \lambda f : \lfloor M_2 \rfloor . \forall x : M_1 . f \Vdash M_2 : \lfloor M_2 \rfloor \to \lceil t \rceil}}}$$
    - $\mathrm{lvl}(t_1) = 2$: Il existe donc t_1' telle que $t_1 = \lceil t_1' \rceil$ et on a :
    $$\begin{aligned} \|M\| &= \lambda f : \lfloor M \rfloor . f \Vdash M \\ &= \lambda f : \forall x : \lfloor M_1 \rfloor . \lfloor M_2 \rfloor . \forall x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1 . (f\, x) \Vdash M_2 \end{aligned}$$
    De plus, par construction de $\mathcal{P}^2$, $(\lceil t_1' \rceil, \lceil t_2 \rceil, \lceil t \rceil) \in \mathcal{R}_{\mathcal{P}^2}$ implique que $(t_1', t_2, t) \in \mathcal{R}_{\mathcal{P}}$ et donc que $(t_1', \lceil t \rceil, \lceil t \rceil) \in \mathcal{R}_{\mathcal{P}^2}$.

D'après le lemme 3.4.8 appliqué à $\Gamma \vdash \forall x : M_1.M_2 : \lceil t \rceil$, on a $\lfloor \Gamma \rfloor \vdash \lfloor \forall x : M_1.M_2 \rfloor : t$ et donc, puisque $\lfloor \Gamma \rfloor \subseteq \|\Gamma\|$, on en déduit que $\|\Gamma\| \vdash \lfloor \forall x : M_1.M_2 \rfloor : t$. Nous avons de plus $\lfloor \forall x : M_1.M_2 \rfloor = \forall x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor$ car $\mathrm{lvl}(M_1) = \mathrm{lvl}(t_1) = 2$. De même, on montre que $\|\Gamma\| \vdash \lfloor M_1 \rfloor : t'_1$.
Par hypothèse d'induction $\mathbf{HI}_3$ associée à $\Gamma \vdash M_1 : t_1$ on a $\|\Gamma\|, x : \lfloor M_1 \rfloor \vdash x \Vdash M_1 : \lceil t'_1 \rceil$. Puis d'après le lemme 1.1.24, on en déduit que :

$$\|\Gamma\|, f : \forall x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor, x : \lfloor M_1 \rfloor \vdash x \Vdash M_1 : \lceil t'_1 \rceil$$

De même, par hypothèse d'induction $\mathbf{HI}_3$ associée à $\Gamma, x : M_1 \vdash M_2 : \lceil t_2 \rceil$ on a $\|\Gamma\|, x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1, y : \lfloor M_2 \rfloor \vdash y \Vdash M_2 : \lceil t_2 \rceil$.
Et d'après le lemme 1.1.24, on en déduit que :

$$\|\Gamma\|, f : \forall x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor, x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1, y : \lfloor M_2 \rfloor \vdash y \Vdash M_2 : \lceil t_2 \rceil$$

Et donc, d'après le lemme de substitution (lemme 1.1.23), on obtient

$$\|\Gamma\|, f : \forall x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor, x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1 \vdash (y \Vdash M_2)[f\, x/y] : \lceil t_2 \rceil$$

Puis d'après le lemme 4.1.4, on obtient ($f \notin \mathcal{FV}(M_2)$) :

$$\|\Gamma\|, f : \forall x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor, x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1 \vdash f\, x \Vdash M_2 : \lceil t_2 \rceil$$

On peut donc utiliser la règle Produit pour dériver (on a bien $(\lceil t'_1 \rceil, \lceil t_2 \rceil, \lceil t \rceil) \in \mathcal{R}_{\mathcal{P}^2}$) :

$$\|\Gamma\|, f : \forall x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor, x : \lfloor M_1 \rfloor \vdash \forall x_R : x \Vdash M_1.f\, x \Vdash M_2 : \lceil t \rceil$$

On peut alors utiliser une nouvelle fois la règle Produit avec $(t'_1, \lceil t \rceil, \lceil t \rceil) \in \mathcal{R}_{\mathcal{P}^2}$ pour prouver :

$$\|\Gamma\|, f : \forall x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor \vdash \forall x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1.f\, x \Vdash M_2 : \lceil t \rceil$$

Avant de conclure ce cas, il est important de remarquer que le produit $\forall x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor \to \lceil t \rceil$ est un produit autorisé dans le contexte $\|\Gamma\|$ car $(t, \lceil s' \rceil, \lceil s' \rceil) \in \mathcal{R}_{\mathcal{P}^2}$. On applique la règle Abstraction pour montrer :

$$\|\Gamma\| \vdash \lambda f : \forall x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor.\forall x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1.f\, x \Vdash M_2 : \lfloor M \rfloor \to \lceil t \rceil$$

Or $\lfloor M \rfloor \Vdash T = \lfloor M \rfloor \to \lceil t \rceil$, on a donc bien montré $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash T$.

– Implication (3) : Supposons $T \equiv \lceil s \rceil$ (on a donc $s = t$). On distingue deux cas :
  – $\mathrm{lvl}(t_1) = 1$: Dans ce cas on a $\lfloor M \rfloor = \lfloor M_2 \rfloor$ et $y \Vdash M = \forall x : M_1.y \Vdash M_2$. On doit montrer que $\|\Gamma\|, y : \lfloor M_2 \rfloor \vdash \forall x : M_1.y \Vdash M_2 : \lceil s \rceil$. D'après le lemme 3.4.8, $\lfloor \Gamma \rfloor \vdash \lfloor M_2 \rfloor : t$ et puisque $\lfloor \Gamma \rfloor \subseteq \|\Gamma\|$ et d'après le lemme 1.1.27, on en déduit que $\|\Gamma\| \vdash \lfloor M_2 \rfloor : t$. Or par hypothèse d'induction $\mathbf{HI}_1$ associée à $\Gamma \vdash M_1 : t_1$, on a $\|\Gamma\| \vdash M_1 : t_1$ et par hypothèse d'induction $\mathbf{HI}_3$ associée à $\Gamma, x : M_1 \vdash M_2 : \lceil t_2 \rceil$ on a $\|\Gamma\|, x : M_1, y : \lfloor M_2 \rfloor \vdash y \Vdash M_2 : \lceil t_2 \rceil$.

Or $x \notin \lfloor M_2 \rfloor$: les occurrences de x dans M_2 sont des occurrences de premier niveau et sont effacées par la projection. Et donc d'après le lemme 1.1.49, on obtient $\|\Gamma\|, y : \lfloor M_2 \rfloor, x : M_1 \vdash y \Vdash M_2 : \lceil t_2 \rceil$. D'après le lemme d'affaiblissement 1.1.24, on obtient $\|\Gamma\|, y : \lfloor M_2 \rfloor \vdash M_1 : t_1$. On peut alors utiliser la règle PRODUIT avec $(t_1, \lceil t_2 \rceil, \lceil s \rceil) \in \mathcal{R}_{\mathcal{P}^2}$ pour dériver $\|\Gamma\|, y : \lfloor M_2 \rfloor \vdash \forall x : M_1 . y \Vdash M_2 : \lceil s \rceil$.

- $\mathrm{lvl}(t_1) = 2$: On sait donc qu'il existe une sorte t_1' telle que $\lceil t_1' \rceil = t_1$ et, par construction de $\mathcal{P}^2$, $(\lceil t_1' \rceil, \lceil t_2 \rceil, \lceil t \rceil) \in \mathcal{R}_{\mathcal{P}^2}$ implique que $(t_1', t_2, t) \in \mathcal{R}_{\mathcal{P}}$ et donc que $(t_1', \lceil t \rceil, \lceil t \rceil) \in \mathcal{R}_{\mathcal{P}^2}$.
On a ici $\lfloor M \rfloor = \forall x : \lfloor M_1 \rfloor . \lfloor M_2 \rfloor$ et $y \Vdash M = \forall x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1 . (y\, x) \Vdash M_2$.
On doit donc montrer :

$$\|\Gamma\|, y : \forall x : \lfloor M_1 \rfloor . \lfloor M_2 \rfloor \vdash \forall x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1 . (y\, x) \Vdash M_2 : \lceil s \rceil$$

Tout comme dans le cas précédent, on prouve à l'aide des lemmes 3.4.8 et 1.1.27 que $\|\Gamma\| \vdash \lfloor M \rfloor : t$ et $\|\Gamma\| \vdash \lfloor M_1 \rfloor : t_1'$. Par la suite, les hypothèses d'induction $\mathbf{HI}_3$ associée à $\Gamma \vdash M_1 : t_1$ ainsi qu'à $\Gamma, x : M_1 \vdash M_2 : \lceil t_2 \rceil$ nous donnent respectivement $\|\Gamma\|, x : \lfloor M_1 \rfloor \vdash x \Vdash M_1 : \lceil t_1' \rceil$ et $\|\Gamma\|, x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1, y : \lfloor M_2 \rfloor \vdash y \Vdash M_2 : \lceil t_2 \rceil$.
Or, d'après le lemme 1.1.24, on en déduit que :

$$\|\Gamma\|, f : \lfloor M \rfloor, x : \lfloor M_1 \rfloor \vdash x \Vdash M_1 : \lceil t_1' \rceil$$

et

$$\|\Gamma\|, f : \lfloor M \rfloor, x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1, y : \lfloor M_2 \rfloor \vdash y \Vdash M_2 : \lceil t_2 \rceil$$

Or $\|\Gamma\|, f : \lfloor M \rfloor, x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1 \vdash f\, x : \lfloor M_2 \rfloor$ et on a $(y \Vdash M_2)[f\, x/y] = f\, x \Vdash M_2$ d'après le lemme 4.1.4 et $y \notin M_2$.
On peut donc appliquer le lemme de substitution pour dériver :

$$\|\Gamma\|, f : \lfloor M \rfloor, x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1 \vdash (f\, x) \Vdash M_2 : \lceil t_2 \rceil$$

Puis, en appliquant deux fois la règle produit avec $(\lceil t_1' \rceil, \lceil t_3 \rceil, \lceil t \rceil) \in \mathcal{R}_{\mathcal{P}^2}$ puis $(t_1, \lceil t \rceil, \lceil t \rceil) \in \mathcal{R}_{\mathcal{P}^2}$ on conclut :

$$\|\Gamma\|, f : \lfloor M \rfloor \vdash \forall x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1 . (f\, x) \Vdash M_2 : \lceil t_2 \rceil$$

- APPLICATION : On a $M = (M_1\, M_2)$ avec $\Gamma \vdash M_1 : \forall x : A.B$ et $\Gamma \vdash M_2 : A$ et $T = B[M_2/x]$.
  - Implication (2) : Supposons $\Gamma \vdash T : \lceil s \rceil$. On rappelle que l'on est dans le cas où $\mathrm{lvl}(M) = \mathrm{lvl}(M_1) = \mathrm{lvl}(\forall x : A.B) = 2$. D'après le lemme 1.1.35, on sait qu'il existe r telle que $\Gamma \vdash M_1 : \forall x : A.B : \lceil r \rceil$. On peut donc appliquer l'hypothèse d'induction $\mathbf{HI}_2$ associée à $\Gamma \vdash M_1 : \forall x : A.B$ pour dériver $\|\Gamma\| \vdash \|M_1\| : \lfloor M_1 \rfloor \Vdash \forall x : A.B$. On distingue alors deux cas :
    - $\mathrm{lvl}(M_2) = \mathrm{lvl}(A) = 1$: Dans ce cas, $\lfloor M_1 \rfloor \Vdash \forall x : A.B = \forall x : A . \lfloor M_1 \rfloor \Vdash B$ et on applique l'hypothèse d'induction $\mathbf{HI}_1$ associée à $\Gamma \vdash M_2 : A$ pour dériver $\|\Gamma\| \vdash M_2 : A$. On utilise donc la règle APPLICATION pour dériver $\|\Gamma\| \vdash \|M_1\|\, M_2 : (\lfloor M_1 \rfloor \Vdash B)[M_2/x]$. Or d'après le lemme 4.1.4, on a $(\lfloor M_1 \rfloor \Vdash B)[M_2/x] = (\lfloor M_1 \rfloor [M_2/x]) \Vdash$

$(B[M_2/x]) = \lfloor M_1 \rfloor \Vdash B[M_2/x]$ car $x \notin \lfloor M_1 \rfloor$. Or dans ce cas, $\|M\| = \|M_1\| M_2$ et $\lfloor M \rfloor \Vdash B[M_2/x] = \lfloor M_1 \rfloor \Vdash B[M_2/x]$. On a donc bien montré $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash (B[M_2/x])$.

- $\mathrm{lvl}(M_2) = \mathrm{lvl}(A) = 2$: D'après le lemme 1.1.34 appliqué à $\Gamma \vdash M_1 : \forall x : A.B$, on a l'existence d'une sorte t telle que $\Gamma \vdash A : \lceil t \rceil$. On utilise donc l'hypothèse d'induction $\mathbf{HI}_2$ associée à $\Gamma \vdash M_2 : A$ afin de dériver $\|\Gamma\| \vdash \|M_2\| : \lfloor M_2 \rfloor \Vdash A$.
  Ici $(\lfloor M_1 \rfloor \Vdash \forall x : A.B) = (\forall x : \lfloor A \rfloor, x_R : x \Vdash A.\lfloor M_1 \rfloor\, x \Vdash B)$, on en déduit donc que $\|\Gamma\| \vdash \|M_1\| \lfloor M_2 \rfloor \|M_2\| : (\lfloor M_1 \rfloor\, x \Vdash B)[\lfloor M_2 \rfloor / x, \|M_2\| / x_R]$. Or d'après le lemme 4.1.5, on a :

$$(\lfloor M_1 \rfloor\, x \Vdash B)[\lfloor M_2 \rfloor / x, \|M_2\| / x_R] \trianglerighteq (\lfloor M_1 \rfloor\, x)[\lfloor M_2 \rfloor / x] \Vdash B[M_2/x]$$

  et on a $(\lfloor M_1 \rfloor\, x)[\lfloor M_2 \rfloor / x] = (\lfloor M_1 \rfloor\, \lfloor M_2 \rfloor)$ var $x \notin \mathcal{FV}(\lfloor M_1 \rfloor)$. Et donc d'après le lemme 1.1.47 (on utilise donc ici $\models \mathrm{SR}_\beta$), on en conclut que :

$$\|\Gamma\| \vdash (\|M_1\| \lfloor M_2 \rfloor \|M_2\|) : (\lfloor M_1 \rfloor\, \lfloor M_2 \rfloor) \Vdash B[M_2/x]$$

- Implication (3) : Supposons $T \equiv \lceil s \rceil$ (on a donc $s = t$). D'après l'hypothèse d'induction $\mathbf{HI}_2$ associée à $\Gamma \vdash M_1 : \forall x : A.B$, on a $\|\Gamma\| \vdash \|M_1\| : \lfloor M_1 \rfloor \Vdash \forall x : A.B$. On distingue alors deux cas :
  - $\mathrm{lvl}(M_2) = \mathrm{lvl}(A) = 1$: Dans ce cas, on peut utiliser l'hypothèse d'induction $\mathbf{HI}_1$ associée à $\Gamma \vdash M_2 : A$ pour démontrer $\|\Gamma\| \vdash M_2 : A$. Or dans ce cas $\lfloor M_1 \rfloor \Vdash \forall x : A.B = \forall x : A.\lfloor M_1 \rfloor \Vdash B$ et $(\lfloor M_1 \rfloor \Vdash B)[M_2/x] = \lfloor M_1 \rfloor \Vdash B[M_2/x]$ d'après le lemme 4.1.4 et $x \notin \mathcal{FV}(\lfloor M_1 \rfloor)$. On peut donc utiliser la règle Application pour dériver :
$$\|\Gamma\| \vdash (\|M_1\| M_2) : \lfloor M_1 \rfloor \Vdash B[M_2/x]$$
  - $\mathrm{lvl}(M_2) = \mathrm{lvl}(A) = 2$:
    D'après le lemme 3.4.8, on a $\lfloor \Gamma \rfloor \vdash \lfloor M_2 \rfloor : \lfloor A \rfloor$ et puisque $\lfloor \Gamma \rfloor \subseteq \|\Gamma\|$, d'après lemme 1.1.27, on a $\|\Gamma\| \vdash \lfloor M_2 \rfloor : \lfloor A \rfloor$.
    Puis d'après l'hypothèse d'induction $\mathbf{HI}_2$ associée à $\Gamma \vdash M_2 : A$ (on a $\Gamma \vdash A : \lceil r \rceil$ pour une sorte r d'après le lemme 3.4.8 appliqué à $\Gamma \vdash M_1 : \forall x : A.B$), on en déduit $\|\Gamma\| \vdash \|M_2\| : \lfloor M_2 \rfloor \Vdash A$
    Or ici $\lfloor M_1 \rfloor \Vdash \forall x : A.B = \forall x : \lfloor A \rfloor, x_R : x \Vdash A.(\lfloor M_1 \rfloor\, x) \Vdash B$ et donc en appliquant deux fois la règle Application, on dérive le séquent :

$$\|\Gamma\| \vdash \|M_1\| \lfloor M_2 \rfloor \|M_2\| : ((\lfloor M_1 \rfloor\, x) \Vdash B)[\lfloor M_1 \rfloor / x, \lfloor M_2 \rfloor / x_R]$$

    Or d'après le lemme 4.1.5, on a

$$((\lfloor M_1 \rfloor\, x) \Vdash B)[\lfloor M_1 \rfloor / x, \lfloor M_2 \rfloor / x_R] \trianglerighteq (\lfloor M_1 \rfloor\, \lfloor M_2 \rfloor) \Vdash (B[M_2/x])$$

    car $x \notin \lfloor M_1 \rfloor$. Et donc d'après le lemme 1.1.47, on en déduit :

$$\|\Gamma\| \vdash \|M_1\| \lfloor M_2 \rfloor \|M_2\| : (\lfloor M_1 \rfloor\, \lfloor M_2 \rfloor) \Vdash (B[M_2/x])$$

- Abstraction : On a $M = \lambda x : M_1.M_2$ et $T = \forall x : M_1.B$ avec $\Gamma, x : M_1 \vdash M_2 : B$ et $\Gamma \vdash T : \lceil t \rceil$.
  - Implication (2) : Supposons $\Gamma \vdash T : \lceil s \rceil$. On doit montrer $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash T$. Par inversion de $\Gamma \vdash T : \lceil t \rceil$ on en déduit qu'il existe r telle que $\Gamma, x : M_1 \vdash B : \lceil r \rceil$. Il est donc possible d'utiliser l'hypothèse d'induction $\mathbf{HI}_2$ associée à $\Gamma, x : M_1 \vdash M_2 : B$ pour en déduire $\|\Gamma, x : M_1\| \vdash \|M_2\| : \lfloor M_2 \rfloor \Vdash B$ (H1). De plus d'après l'hypothèse d'induction $\mathbf{HI}_3$ associée à $\Gamma \vdash T : \lceil t \rceil$, on a $\|\Gamma\|, f : \lfloor T \rfloor \vdash f \Vdash T : \lceil t \rceil$. Or, d'après le lemme 3.4.8, on a $\lfloor \Gamma \rfloor \vdash \lfloor M \rfloor : \lfloor T \rfloor$ et, puisque $\lfloor \Gamma \rfloor \subseteq \|\Gamma\|$ et d'après le lemme 1.1.27, on en déduit $\|\Gamma\| \vdash \lfloor M \rfloor : \lfloor T \rfloor$. Et donc, le lemme de substitution nous permet de dériver le séquent : $\|\Gamma\| \vdash (f \Vdash T)[\lfloor M \rfloor / f] : \lceil t \rceil$ or comme $f \notin \mathcal{FV}(T)$ on déduit à l'aide du lemme 4.1.4 que $\|\Gamma\| \vdash \lfloor M \rfloor \Vdash T : \lceil t \rceil$ (H2). On distingue deux cas :
    - $\mathrm{lvl}(M_1) = 1$: Dans ce cas :
      - $\|\Gamma, x : M_1\| = \|\Gamma\|, x : M_1$
      - $\|M\| = \lambda x : M_1.\|M_2\|$
      - $\lfloor M \rfloor = \lfloor M_2 \rfloor$
      - $\lfloor M \rfloor \Vdash T = \forall x : M_1.\lfloor M_2 \rfloor \Vdash B$

      et donc la règle Abstraction appliquée à (H1) et à (H2) nous permet de conclure que $\|\Gamma\| \vdash \lambda x : M_1.\|M_2\| : \forall x : M_1.\lfloor M_2 \rfloor \Vdash B$, c'est-à-dire $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash T$.
    - $\mathrm{lvl}(M_1) = 2$: Dans ce cas :
      - $\|\Gamma, x : M_1\| = \|\Gamma\|, x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1$
      - $\|M\| = \lambda x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1.\|M_2\|$
      - $\lfloor M \rfloor = \lambda x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor$
      - $\lfloor M \rfloor \Vdash T = \forall x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1.((\lambda x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor)\, x) \Vdash B$

      Or par inversion de (H2), on obtient qu'il existe t' telle que :

      $$\|\Gamma\|, x : \ \lfloor M_1 \rfloor \vdash \forall x_R : x \Vdash M_1.((\lambda x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor)\, x) \Vdash B : \lceil t' \rceil$$

      Or le lemme 4.1.6 prouve que $((\lambda x : \lfloor M_1 \rfloor.\lfloor M_2 \rfloor)\, x) \Vdash B \trianglerighteq \lfloor M_2 \rfloor \Vdash B$ et donc d'après SR_β, on en déduit que :

      $$\|\Gamma\|, x : \ \lfloor M_1 \rfloor \vdash \forall x_R : x \Vdash M_1.\lfloor M_2 \rfloor \Vdash B : \lceil t' \rceil \qquad \text{(H3)}$$

      On peut alors appliquer la règle Abstraction à (H1) et à (H3) pour dériver le séquent :

      $$\|\Gamma\|, x : \lfloor M_1 \rfloor \vdash \lambda x_R : x \Vdash M_1.\|M_2\| : \forall x_R : x \Vdash M_1.\lfloor M_2 \rfloor \Vdash B$$

      Puis en utilisant une seconde fois la règle Abstraction avec le séquent précédent (H2), on en déduit :

      $$\|\Gamma\| \vdash \lambda x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1.\|M_2\| \quad : \quad \forall x : \lfloor M_1 \rfloor, x_R : x \Vdash M_1.\lfloor M_2 \rfloor \Vdash B$$

      c'est-à-dire $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash T$.
  - Implication (3) : $T \equiv \lceil s \rceil$ est impossible : l'implication est donc triviale.
- Cumulativité : $T' \preccurlyeq T$ avec $\Gamma \vdash M : T'$ et $\Gamma \vdash T : \lceil t \rceil$.

- Implication (2) : Supposons que $\Gamma \vdash T : \lceil s \rceil$, on veut montrer qu'il existe s' telle que $\Gamma \vdash T' : \lceil s' \rceil$ afin de pouvoir appliquer l'hypothèse d'induction $\mathbf{HI}_2$ associée à $\Gamma \vdash M : T'$. D'après le lemme 1.1.35 appliqué à $\Gamma \vdash M : T'$ on a $\mathrm{WF}_\Gamma(T')$ c'est-à-dire soit il existe s' telle que $\Gamma \vdash T' : \lceil s' \rceil$ soit T' est une sorte r'. Dans ce dernier cas, on en déduit que T est convertible en une sorte r avec $(r', r) \in \mathcal{C}_{\mathcal{P}^2}$. Et par préservation du typage SR_β, on en déduit que $\Gamma \vdash r : \lceil s \rceil$. On peut appliquer ici l'hypothèse WEAK-DOWNSTABLE pour conclure qu'il existe une sorte s' telle que $\Gamma \vdash T' : \lceil s' \rceil$.
  On peut donc bien appliquer l'hypothèse d'induction $\mathbf{HI}_2$ associée à $\Gamma \vdash M : T'$ pour en conclure que $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash T'$ (H1). Or, d'après le lemme 4.1.6, on a $\lfloor M \rfloor \Vdash T' \preccurlyeq \lfloor M \rfloor \Vdash T$. Et donc l'hypothèse d'induction $\mathbf{HI}_3$ associée à $\Gamma \vdash T : \lceil t \rceil$, nous donne $\|\Gamma\|, x : \lfloor T \rfloor \vdash x \Vdash T : \lceil t \rceil$. Et on a $\|\Gamma\| \vdash \lfloor M \rfloor : \lfloor T \rfloor$ en utilisant le lemme 3.4.8 ainsi que le fait que $\lfloor \Gamma \rfloor \subseteq \|\Gamma\|$. Donc, d'après le lemme de substitution, on en déduit que $\|\Gamma\| \vdash \lfloor M \rfloor \Vdash T : \lceil t \rceil$ (H2) car $(x \Vdash T)[\lfloor M \rfloor / x] = \lfloor M \rfloor \Vdash T$ d'après le lemme 4.1.4 et le fait que $x \notin T$. On peut alors conclure que $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash T$ grâce à la règle Conversion appliquée à (H1) et (H2).
- Implication (3) : Supposons $T \equiv \lceil s \rceil$, alors on a également $T' \equiv \lceil s' \rceil$ pour une sorte s' vérifiant $(s', s) \in \mathcal{C}_{\mathcal{P}}$. On peut donc appliquer l'hypothèse d'induction $\mathbf{HI}_3$ associée à $\Gamma \vdash M : T'$. On en déduit donc que $\|\Gamma\|, x : \lfloor M \rfloor \vdash x \Vdash M : \lceil s' \rceil$ et on peut utiliser la règle Cumulativité-Sortes pour dériver $\|\Gamma\|, x : \lfloor M \rfloor \vdash x \Vdash M : \lceil s \rceil$ car on a bien $\lceil s' \rceil \preccurlyeq \lceil s \rceil$.

- Cumulativité-Sortes : $T' \preccurlyeq T = \lceil t \rceil$ avec $\Gamma \vdash M : T'$.
  - Implication (2) : Supposons $\Gamma \vdash T : \lceil s \rceil$. En utilisant un raisonnement identique au cas de la preuve de l'implication (2) pour traiter la règle Cumulativité, on montre que $\Gamma \vdash T' : \lceil s' \rceil$ pour une certaine sorte s'. On peut donc utiliser l'hypothèse d'induction $\mathbf{HI}_2$ associée à $\Gamma \vdash M : T'$ afin de dériver le séquent $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash T'$ (H1).
    Montrons maintenant que $\lfloor M \rfloor \Vdash T = \lfloor M \rfloor \to \lceil t \rceil$ est typable dans le contexte $\|\Gamma\|$. Par inversion de $\Gamma \vdash \lceil t \rceil : \lceil s \rceil$ on en déduit qu'il existe $s'' \preccurlyeq s$ telle que $(\lceil t \rceil, \lceil s'' \rceil) \in \mathcal{A}_{\mathcal{P}^2}$. Par construction de $\mathcal{P}^2$, on a $(t, s'') \in \mathcal{A}_{\mathcal{P}}$ et donc $(t, \lceil s'' \rceil, \lceil s'' \rceil) \in \mathcal{R}_{\mathcal{P}^2}$. Or on a $\|\Gamma\| \vdash \lfloor M \rfloor : t$ d'après le lemme 3.4.8 et $\lfloor \Gamma \rfloor \subseteq \|\Gamma\|$. On en déduit donc que $\|\Gamma\| \vdash \lfloor M \rfloor \to \lceil t \rceil : \lceil s'' \rceil$ (H2). On peut donc utiliser la règle Conversion avec (H1) et (H2), pour dériver $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \to \lceil t \rceil$ (on a bien $\lfloor M \rfloor \Vdash T' \preccurlyeq \lfloor M \rfloor \Vdash T$ d'après le lemme 4.1.6).
  - Implication (3) : Identique à la preuve pour Cumulativité. ☺

4.2 Le cas des systèmes utilisant les notations de $\mathcal{F}^2$

Dans cette section, nous allons illustrer des conséquences de la théorie de la réalisabilité dans les systèmes qui contiennent $\mathcal{F}^2$ et qui utilisent les notations pour l'arithmétique de $\mathcal{F}^2$.

Nous insisterons tout particulièrement sur les spécificités de $\mathcal{F}^2$. En particulier, le choix de représenter les types de données en utilisant des encodages imprédicatifs qui induisent un changement de polarité, nous poussera à considérer de nouveaux axiomes afin d'obtenir des propriétés attendues des types de données positifs.

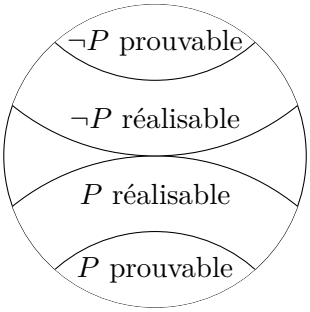
Nous introduirons pour cela des définitions et des lemmes qui ne sont pas spécifiques à $\mathcal{F}^2$ et

qui s'appliqueront également aux systèmes utilisant les inductifs pour représenter les données. Nous prendrons donc garde à bien préciser pour chacune des notions introduites si elle concerne $\mathcal{F}^2$, un système contenant $\mathcal{F}^2$ ou un système quelconque pouvant vérifier certaines propriétés.

4.2.1 Définition (Formule réalisable)
Soit une formule P typable dans un contexte Γ : $\Gamma \vdash_{\mathcal{P}^2} P : \lceil s \rceil$, on dira de P qu'elle est réalisée dans un contexte Δ lorsque qu'il existe un programme r (de type $\lfloor P \rfloor$) tel que $\Delta \vdash r \Vdash P$.

Le théorème 4.1.8 implique que les formules prouvables dans un contexte Γ sont réalisées par la projection de leur preuve dans le contexte $\|\Gamma\|$. Dans un contexte vide (ou dans un contexte auto-réalisé, voir plus loin), on dira simplement que les formules prouvables sont réalisables. Ainsi, il est possible de comprendre la réalisabilité comme une extension de la notion de prouvabilité. De plus, dans les systèmes disposant d'une notation $\bot$ pour l'absurdité logique et qui sont capables de prouver que cette dernière n'est pas réalisée (c-à-d. que $\forall x : \lfloor \bot \rfloor . ((x \Vdash \bot) \to \bot)$), on montre facilement que l'on ne peut pas avoir P et $P \to \bot$ simultanément réalisées (en effet, si $t \Vdash P$ et $f \Vdash P \to \bot$, c-à-d. en dépliant $\forall x : \lfloor P \rfloor, x \Vdash P \to (f\, x) \Vdash \bot$, alors on a $f\, t \Vdash \bot$).

Ainsi, on peut organiser l'ensemble des formules bien typées selon le diagramme suivant :



4.2.1 Formule auto-réalisée

Il existe néanmoins des formules où les deux notions coïncident. Par exemple, dans $\mathcal{F}^2$, les notations $\bot$, $\_ \in \mathbb{N}$, $x =_\tau y$ sont réalisables dans tous les contextes où ils sont vérifiés. Pour préciser cette idée, nous introduisons les deux définitions suivantes :

4.2.2 Définition (Formule autoréalisable, réalisateur canonique)
Une formule F est auto-réalisable dans un contexte Γ s'il existe un terme ϑ_F et un réalisateur canonique t_F tels que :

$$\Gamma \vdash_{\mathcal{P}} \vartheta_F : F \to t_F \Vdash F$$

On dira de plus que le t_F est uniforme si $\mathcal{FV}(t_F) = \emptyset$, et on dira alors que F est uniformément auto-réalisable.

Dualement, on définit les formules "honnêtes" comme étant les formules dont l'existence d'un réalisateur est une condition suffisante pour qu'elles soient prouvables ("les réalisateurs ne mentent pas").

4.2.3 Définition (Formule honnête)
Une formule F est <u>honnête</u> dans un contexte Γ s'il existe un terme π_F tel que :

$$\Gamma \vdash \pi_F : \forall x : \lfloor F \rfloor, x \Vdash F \to F$$

Dans $\mathcal{F}^2$, on a le lemme suivant :

4.2.4 Lemme
On a :

1. *l'absurdité est honnête :* $\vdash_{\mathcal{F}^2} \forall x : \lfloor \bot \rfloor . x \Vdash \bot \quad \to \quad \bot$
2. *les égalités sont honnêtes :* $\vdash_{\mathcal{F}^2} \forall \alpha : \star, x\, y \ : \alpha, r : \lfloor x =_\alpha y \rfloor, r \Vdash x =_\alpha y \to x =_\alpha y$
3. *les égalités sont uniformément auto-réalisées :* $\vdash_{\mathcal{F}^2} \forall \alpha : \star, x\, y \ : \alpha . x =_\alpha y \to \mathsf{id} \Vdash x =_\alpha y$
4. *le prédicat d'induction est honnête :* $\vdash_{\mathcal{F}^2} \forall x\, r : \mathsf{nat}.r \Vdash x \in \mathbb{N} \to x \ \in \mathbb{N}$
5. *le prédicat d'induction est auto-réalisé (non-uniformément) :* $\vdash_{\mathcal{F}^2} \forall x \in \mathbb{N} . x \Vdash x \in \mathbb{N}$

où on rappelle que $\mathsf{id} = \lambda \alpha : \star, x : \alpha . x$.

Démonstration On montre dans un style informel comment construire la preuve dans chacun des cas :

1. On a $x \Vdash \bot = x \Vdash (\forall X : \lceil \star \rceil . X) = \forall \gamma : \star, X : \gamma \to \lceil \star \rceil . X\,(x\,\gamma)$ et donc en posant $\gamma = \forall \alpha : \star . \alpha$ (ou n'importe quel autre type bien formé) et $X = \lambda z : \gamma . Y$ on démontre bien $\forall Y : \lceil \star \rceil . Y$, c'est-à-dire $\bot$.
2. De la même façon, $r \Vdash x =_\alpha y$ se déplie de la façon suivant :

$$\begin{aligned} r \Vdash x =_\alpha y &= r \Vdash (\forall X : \alpha \to \lceil \star \rceil . X\, x \to X\, y) \\ &= \forall X : \star, X_R : \alpha \to X \to \lceil \star \rceil . (r\, X) \Vdash (X\, x \to X\, y) \\ &= \forall X : \star, X_R : \alpha \to X \to \lceil \star \rceil, z : X, z_R : X_R\, x\, z . (r\, X\, z) \Vdash X\, y \\ &= \forall X : \star, X_R : \alpha \to X \to \lceil \star \rceil, z : X, z_R : X_R\, x\, z . X\, y\, (r\, X\, z) \end{aligned}$$

   On peut renommer les variables liées et utiliser la notation $\cdot \to \cdot$ pour les produits non-dépendants. On obtient :

$$\forall \gamma : \star, X : \alpha \to \gamma \to \lceil \star \rceil, z : \gamma . X\, x\, z \to X\, y\, (r\, \gamma\, z)$$

   On peut conclure en instanciant γ en un type habité comme par exemple $\forall \alpha : \star . \alpha \to \alpha$ et z en un habitant, ici l'identité, et de prendre $X = \lambda x : \alpha, z : \gamma . Y\, x$ pour prouver $\forall Y : \alpha \to \lceil \star \rceil . Y\, x \to Y\, y$, c'est-à-dire $x =_\alpha y$.
3. Réciproquement, $(\lambda \alpha : \star, x : \alpha . x) \Vdash x =_\alpha y$ se déplie en

$$\forall \gamma : \star, X : \alpha \to \gamma \to \lceil \star \rceil, z : \gamma . X\, x\, z \to X\, y\, ((\lambda \alpha : \star, x : \alpha . x)\, \gamma\, z)$$

   ce qui peut se réduire en

$$\forall \gamma : \star, X : \alpha \to \gamma \to \lceil \star \rceil, z : \gamma . X\, x\, z \to X\, y\, z$$

   puis, grâce à $x =_\alpha y$, on peut récrire cette formule en la tautologie suivante :

$$\forall \gamma : \star, X : \alpha \to \gamma \to \lceil \star \rceil, z : \gamma . X\, x\, z \to X\, x\, z$$

4. On dérive $\forall x\, r : \mathsf{nat}.r \Vdash x \in \mathbb{N} \to x \ \in \mathbb{N}$ de manière similaire aux cas précédents. En effet, $r \Vdash x \in \mathbb{N}$ se déplie en (voir page 200) :

$$\begin{array}{l}\forall \alpha : \star, X : \mathsf{nat} \to \alpha \to \lceil\star\rceil, f : \alpha \to \alpha. \\ \quad (\forall x : \mathsf{nat}, z : \alpha. X\, x\, z \to X\, (\mathsf{S}\, x)\, (f\, z)) \to \\ \quad\quad \forall z : \alpha. X\, \overline{0}\, z \to X\, x\, (r\, \alpha\, f\, z)\end{array}$$

On obtient donc le résultat en instanciant X par $\lambda y : \mathsf{nat}, z : \alpha. Y\, y$ pour dériver $\forall Y : \mathsf{nat} \to \lceil\star\rceil.(\forall y : \mathsf{nat}. Y\, y \to Y\, (\mathsf{S}\, y)) \to Y\, \overline{0} \to Y\, x$ c'est-à-dire $x \in \mathbb{N}$.

5. En utilisant le principe d'induction $x \in \mathbb{N}$ on se ramène à démontrer :
   – $\overline{0} \Vdash \overline{0} \in \mathbb{N}$: La formule $\overline{0} \Vdash \overline{0} \in \mathbb{N}$ se déplie en

$$\begin{array}{l}\forall \alpha : \star, X : \mathsf{nat} \to \alpha \to \lceil\star\rceil, f : \alpha \to \alpha. \\ \quad (\forall x : \mathsf{nat}, z : \alpha. X\, x\, z \to X\, (\mathsf{S}\, x)\, (f\, z)) \to \\ \quad\quad \forall z : \alpha. X\, \overline{0}\, z \to X\, \overline{0}\, (\overline{0}\, \alpha\, f\, z)\end{array}$$

   puis se réduit en la tautologie suivante :

$$\begin{array}{l}\forall \alpha : \star, X : \mathsf{nat} \to \alpha \to \lceil\star\rceil, f : \alpha \to \alpha. \\ \quad (\forall x : \mathsf{nat}, z : \alpha. X\, x\, z \to X\, (\mathsf{S}\, x)\, (f\, z)) \to \\ \quad\quad \forall z : \alpha. X\, \overline{0}\, z \to X\, \overline{0}\, z\end{array}$$

   – $(n \Vdash n \in \mathbb{N}) \to (\mathsf{S}\, n) \Vdash (\mathsf{S}\, n) \in \mathbb{N}$: La formule $(\mathsf{S}\, n) \Vdash (\mathsf{S}\, n) \in \mathbb{N}$ se déplie en

$$\begin{array}{l}\forall \alpha : \star, X : \mathsf{nat} \to \alpha \to \lceil\star\rceil, f : \alpha \to \alpha. \\ \quad (\forall x : \mathsf{nat}, z : \alpha. X\, x\, z \to X\, (\mathsf{S}\, x)\, (f\, z)) \to \\ \quad\quad \forall z : \alpha. X\, \overline{0}\, z \to X\, (\mathsf{S}\, n)\, (\mathsf{S}\, n\, \alpha\, f\, z)\end{array}$$

   ce qui est convertible en

$$\begin{array}{l}\forall \alpha : \star, X : \mathsf{nat} \to \alpha \to \lceil\star\rceil, f : \alpha \to \alpha. \\ \quad (\forall x : \mathsf{nat}, z : \alpha. X\, x\, z \to X\, (\mathsf{S}\, x)\, (f\, z)) \to \\ \quad\quad \forall z : \alpha. X\, \overline{0}\, z \to X\, (\mathsf{S}\, n)\, (f\, (n\, \alpha\, f\, z))\end{array}$$

   Or on peut dériver cette formule dans $\mathcal{F}^2$ à partir de $n \Vdash n \in \mathbb{N}$ en se ramenant à prouver $X\, n\, (n\, \alpha\, f\, z))$ grâce à la prémisse $(\forall x : \mathsf{nat}, z : \alpha. X\, z\, x \to X\, (\mathsf{S}\, x)\, (f\, z))$ instanciée avec $x = n$ et $z = (n\, \alpha\, f\, z)$. ☺

On notera que l'absurdité logique n'est pas uniformément auto-réalisée car le type $\lfloor\bot\rfloor$ n'a pas d'habitant clos. Ainsi, nous n'aurons pas de critère syntaxique pour montrer que les négations sont auto-réalisées (par exemple, la formule $\forall x : \mathsf{nat}. \mathsf{S}\, x =_{\mathsf{nat}} \overline{0} \to \bot$ ne sera pas auto-réalisée[2]). Une façon de pallier ce problème pourrait être d'ajouter dans le contexte un "démon" de type $\lfloor\bot\rfloor$ qui permettrait de montrer que $\bot$ est auto-réalisé. Mais cette solution a le désavantage de polluer l'espace des programmes. On choisit plutôt d'utiliser ici une version plus faible de l'absurdité ($\bot_{\mathrm{w}} = \forall x : \mathsf{nat}. x =_{\mathsf{nat}} \overline{0}$) qui a l'avantage d'être auto-réalisée et presque aussi dévastatrice que l'absurdité.

2. On notera toute fois que le fait que $\bot$ apparaisse comme conclusion d'une formule n'est pas une condition suffisante pour ne pas être auto-réalisée. Par exemple, la formule $\bot \to \bot$ –comme toutes les formules prouvables– est auto-réalisée.

4.2.5 Définition (Système de notations pour la réalisabilité)
Enfin, on appellera système de notations pour la réalisabilité tout système de la forme $\mathcal{P}^2$ tel que :
- $\lfloor \mathsf{prop} \rfloor = \mathsf{set}$,
- $\lfloor N \in \mathbb{N} \rfloor = \mathsf{nat}$,
- $\lfloor P \wedge Q \rfloor = \lfloor P \rfloor \times \lfloor Q \rfloor$,
- $\lfloor \exists x : A.P \rfloor = (\!|\lfloor P \rfloor|\!)$

c'est-à-dire dans lequel les réalisateurs des formules de type prop sont des programmes dont le type vit dans set, les réalisateurs du prédicat d'induction sont les entiers, les réalisateurs de la conjonction sont des pairs de réalisateurs et les réalisateurs du connecteur existentiel sont marqués par l'opérateur $(\!|\cdot|\!)$.
On notera cette propriété $\mathcal{P}^2 \models$ NOTATION-REALIZABILITY.

On vérifie alors bien que $\mathcal{F}^2$ est muni d'un système de notations pour la réalisabilité :

4.2.6 Lemme

$$\mathcal{F}^2 \models \text{NOTATION-REALIZABILITY}$$

Dans de tels systèmes, le fait que $(\mathsf{set}, \mathsf{prop}, \mathsf{prop}) \in \overline{\mathcal{R}}$ permet de quantifier sur les réalisateurs dans les formules. Et l'implication $P \to Q$ entre deux formules est bien réalisée par des programmes de type $\lfloor P \rfloor \to \lfloor Q \rfloor$ qui habitent set.

On verra par la suite que la propriété d'être auto-réalisé est essentiellement désirée pour les formules en "position négative" (ainsi on cherchera surtout à montrer que les formules des contextes sont auto-réalisées) tandis que le fait d'être honnête est plutôt désiré pour les formules en "position positive" (par exemple dans le théorème 4.2.30, on demande à ce que la conclusion du séquent soit honnête).

On ne s'étonne donc pas d'observer le comportement suivant de ces concepts vis-à-vis de l'implication :

4.2.7 Lemme
Supposons $\mathcal{P}^2 \models$ NOTATION-REALIZABILITY.
Supposons de plus que l'on ait $\Gamma \vdash_{\mathcal{P}^2} P : \mathsf{prop}$ *et* $\Gamma \vdash_{\mathcal{P}^2} Q : \mathsf{prop}$.
Si P *est auto-réalisé dans* Γ *et* Q *est honnête dans* Γ, *alors* $P \to Q$ *est honnête dans* Γ *et* $Q \to P$ *est auto-réalisé dans* Γ.
De plus, si P *est uniformément auto-réalisé, alors* $Q \to P$ *est uniformément auto-réalisé.*

Démonstration On utilise un style informel et on donne les termes construits entre parenthèses :
- $\Gamma \vdash_{\mathcal{P}^2} \pi_{P\to Q} : \forall f : \lfloor P \to Q \rfloor.(f \Vdash (P \to Q)) \to P \to Q$:
  Supposons $f \Vdash P \to Q$ (H_1) pour un certain f de type $\lfloor P \to Q \rfloor$ et supposons P (H_2). Par hypothèse, on sait qu'il existe t_P tel que $t_P \Vdash P$ (terme : $\vartheta_P\, H_2$). On peut donc utiliser (H_1) pour montrer que $(f\, t_P) \Vdash Q$ (terme : $H_1\, t_P\, (\vartheta_P\, H_2)$) et d'après l'hypothèse d'induction on en déduit que Q (terme : $\pi_Q\, (f\, t_P)\, (H_1\, t_P\, (\vartheta_P\, H_2))$). On a donc construit :

$$\pi_{P\to Q} = \lambda f : \lfloor P \to Q \rfloor, H_1 : f \Vdash P \to Q, H_2 : P.\pi_Q\, (f\, t_P)\, (H_1\, t_P\, (\vartheta_P\, H_2))$$

- il existe $t_{Q\to P}$ tel que $\Gamma \vdash_{\mathcal{P}^2} \vartheta_{Q\to P} : (Q \to P) \to t_{Q\to P} \Vdash Q \to P$:
  On prend $t_{Q\to P} = \lambda x : \lfloor Q \rfloor.t_P$. Supposons $Q \to P$ (H_1), on doit montrer que $t_{Q\to P} \Vdash (Q \to P)$ c'est-à-dire $\forall x : \lfloor Q \rfloor.x \Vdash Q \to t_{Q\to P}\, x \Vdash P$ ou encore par conversion que : $\forall x : \lfloor Q \rfloor.x \Vdash Q \to t_P \Vdash$

P. Soit donc x de type $\lfloor Q \rfloor$ tel que $x \Vdash Q$ (H_2). On déduit Q grâce à π_Q (terme : $\pi_Q\, x\, H_2$) et donc $Q \to P$ nous permet de déduire P (terme : $H_1\,(\pi_Q\, x\, H_2)$) puis ϑ_P nous permet d'en déduire que $t_P \Vdash P$ (terme : $\vartheta_P\,(H_1\,(\pi_Q\, x\, H_2))$). On a donc construit :

$$\vartheta_{Q\to P} = \lambda H_1 : Q \to P, x : \lfloor Q \rfloor, H_2 : x \Vdash Q.\vartheta_P\,(H_1\,(\pi_Q\, x\, H_2))$$

Enfin, dans le cas où P est uniformément auto-réalisé, on a $\mathcal{FV}(t_P) = \emptyset$ et donc $\mathcal{FV}(t_{Q\to P}) = \emptyset$ également et donc $Q \to P$ est bien uniformément auto-réalisé dans Γ. ☺

Le lemme suivant montre que si une formule P est honnête (resp. uniformément auto-réalisée) alors $\forall x : A.P$ est honnête (resp. uniformément auto-réalisée). Mais ce résultat ne s'étend pas facilement aux formules non-uniformément auto-réalisées car les réalisateurs canoniques peuvent dépendre de x.

Néanmoins, lorsque que le quantificateur est relativisé et que le contexte permet de démontrer que $r \Vdash x \in \mathbb{N} \Rightarrow r =_{\mathsf{nat}} x$, alors P auto-réalisé implique $\forall x \in \mathbb{N}.P$ auto-réalisé.

Notons $\mathsf{datatype}$ la formule :

$$\mathsf{datatype} = \forall x\, r : \mathsf{nat}.r \Vdash x \in \mathbb{N} \to r =_{\mathsf{nat}} x$$

4.2.8 Lemme
Supposons $\mathcal{P}^2 \models$ NOTATION-REALIZABILITY.
Supposons que l'on ait $\Gamma \vdash A : s$ *et* $\Gamma, x : A \vdash_{\mathcal{P}^2} P : \lceil r \rceil$ *avec* $(s, \mathsf{prop}, \mathsf{prop}) \in \mathcal{R}_{\mathcal{P}^2}$.
Alors,

- *Si* P *est honnête dans* $\Gamma, x : A$, *alors* $\forall x : A.P$ *est honnête dans* Γ.
- *Si* P *est honnête dans* $\Gamma, x : \mathsf{nat}$, *alors* $\forall x \in \mathbb{N}.P$ *est honnête dans* Γ.
- *Si* P *est uniformément auto-réalisée dans* $\Gamma, x : A$, *alors* $\forall x : A.P$ *est uniformément auto-réalisée dans* Γ.
- *Si* $\Gamma \vdash \mathsf{datatype}$ *et si* $x \in \mathbb{N}$ *est honnête dans* $\Gamma, x : \mathsf{nat}$, *alors si* P *est auto-réalisée dans* $\Gamma, x : A$, *alors* $\forall x \in \mathbb{N}.P$ *est auto-réalisée dans* Γ.

Démonstration
- On a $r \Vdash \forall x : A.P = \forall x : A.r \Vdash P$, le terme $\pi_{\forall x:A.P} = \lambda r : \lfloor \forall x : A.P \rfloor, h : r \Vdash \forall x : A.P.\pi_P\,(h\, x)$ montre donc bien que $\forall r : \lfloor \forall x : A.P \rfloor.(r \Vdash \forall x : A.P) \to \forall x : A.P$.
- On a $\forall x \in \mathbb{N}.P = \forall x : \mathsf{nat}.x \in \mathbb{N} \to P$, on peut donc utiliser le cas précédent et lemme 4.2.7 pour conclure.
- On a $\Gamma, x : A \vdash_{\mathcal{P}^2} \vartheta_P : P \to t_P \Vdash P$ avec $\mathcal{FV}(t_P) = \emptyset$. On pose $t_{\forall x:A.P} = t_P$ (ce réalisateur est bien uniforme car $\mathcal{FV}(t_{\forall x:A.P}) = \mathcal{FV}(t_P) = \emptyset$) et on veut montrer la formule $(\forall x : A.P) \to t_P \Vdash \forall x : A.P$ c'est-à-dire $(\forall x : A.P) \to \forall x : A.t_P \Vdash P$. Le terme $\vartheta_{\forall x:A.P} = \lambda h : (\forall x : A.P), x : A.\vartheta_P\,(h\, x)$ convient bien pour cela.
- On a $\Gamma, x : A \vdash_{\mathcal{P}^2} \vartheta_P : P \to t_P \Vdash P$. On pose $t_{\forall x\in\mathbb{N}} = \lambda x : \mathsf{nat}.t_P$ et on veut montrer dans le contexte Γ que $(\forall x \in \mathbb{N}.P) \to t_{\forall x\in\mathbb{N}.P} \Vdash \forall x \in \mathbb{N}.P$ c'est-à-dire $(\forall x \in \mathbb{N}.P) \to \forall x : \mathsf{nat}, r : \mathsf{nat}, r \Vdash x \in \mathbb{N}.(t_{\forall x\in\mathbb{N}.P}\, r) \Vdash P$.
  Dans un style informel :
  Supposons $\forall x \in \mathbb{N}.P$ (h_1), soit x et r de type nat tels que $r \Vdash x \in \mathbb{N}$ (h_2). Soit ϱ une preuve de $r \Vdash x \in \mathbb{N} \to r =_{\mathsf{nat}} x$ (on en dispose d'une par hypothèse). Alors, en appliquant ϱ à h_2, on en déduit $r =_{\mathsf{nat}} x$. On peut utiliser cette égalité pour se ramener à montrer $(t_{\forall x\in\mathbb{N}.P}\, x) \Vdash P$. On a $(t_{\forall x\in\mathbb{N}.P}\, x) \trianglerighteq t_P$, on doit donc montrer $t_P \Vdash P$. Or, $x \in \mathbb{N}$ est fidèle, il existe donc une preuve

$\pi_{\mathbb{N}}$ de la formule $\forall r : \mathsf{nat}, r \Vdash x \in \mathbb{N} \to x \in \mathbb{N}$ que l'on peut utiliser pour en déduire $x \in \mathbb{N}$ en l'appliquant à r et h_2. Puis, on obtient une preuve de P en utilisant h_1 (terme : $h_1\,(\pi_{\mathbb{N}}\,r\,h_2)$). Enfin, on conclut que $t_P \Vdash P$ grâce à ϑ_P appliqué à la preuve de P. On a donc construit le terme :

$$\begin{aligned}\vartheta_{\forall x\in\mathbb{N}.P} \quad &= \quad \lambda h_1 : \forall x \in \mathbb{N}.P, x\,r : \mathsf{nat}, h_2 : r \Vdash x \in N. \\ &\qquad \mathsf{elim}_{=}[\mathsf{nat}, r, (t_{\forall x\in\mathbb{N}.P}\,r) \Vdash P, \varrho\,h_2, \vartheta_P\,(h_1\,(\pi_{\mathbb{N}}\,r\,h_2))]\end{aligned}$$ ☺

Les deux lemmes suivants montrent que la conjonction de formules honnêtes (resp. auto-réalisées) est honnête (resp. auto-réalisée) à condition que les réalisateurs de la conjonction se comportent bien.

4.2.9 Lemme
Supposons $\mathcal{P}^2 \models$ NOTATION-REALIZABILITY.
Si on a :

$$\Gamma \vdash_{\mathcal{P}^2} \forall c : \lfloor P\rfloor \times \lfloor Q\rfloor . c \Vdash P \wedge Q \to (\mathsf{proj}^1[\lfloor P\rfloor, \lfloor Q\rfloor]\,c) \Vdash P \wedge (\mathsf{proj}^2[\lfloor P\rfloor, \lfloor Q\rfloor]\,c) \Vdash Q$$

et P et Q sont honnêtes dans Γ, alors $P \wedge Q$ est honnête dans Γ.

Démonstration Informellement. On doit montrer la formule suivante dans le contexte Γ :

$$\forall r : \lfloor P\rfloor \times \lfloor Q\rfloor, r \Vdash P \wedge Q \to P \wedge Q$$

En utilisant l'hypothèse, on montre que $(\mathsf{proj}^1[\lfloor P\rfloor, \lfloor Q\rfloor]\,r) \Vdash P$ et par honnêteté on en déduit P. De même pour Q et on peut donc utiliser $\mathsf{intro}_\wedge[\_]$ pour conclure $P \wedge Q$. ☺

4.2.10 Lemme
Supposons $\mathcal{P}^2 \models$ NOTATION-REALIZABILITY.
Si on a :

$$\Gamma \vdash_{\mathcal{P}^2} \forall x : \lfloor P\rfloor . x \Vdash P \to \forall y : \lfloor Q\rfloor . y \Vdash Q \to (\mathsf{pair}[\lfloor P\rfloor, \lfloor Q\rfloor]\,x\,y) \Vdash P \wedge Q$$

et P et Q sont (uniformément) auto-réalisées dans Γ, alors $P \wedge Q$ est (uniformément) auto-réalisée dans Γ.

Démonstration Soit $t_{P\wedge Q} = \mathsf{pair}[\lfloor P\rfloor, \lfloor Q\rfloor]\,t_P\,t_Q$.
Montrons que $\Gamma \vdash P \wedge Q \to t_{P\wedge Q} \Vdash P \wedge Q$. Supposons $P \wedge Q$, en utilisant $\mathsf{elim\text{-}gauche}_\wedge[\_]$ et $\mathsf{elim\text{-}droite}_\wedge[\_]$, on peut déduire P et déduire Q. Puis ϑ_P et ϑ_Q nous permettent de construire des preuves de $t_P \Vdash P$ et $t_Q \Vdash Q$, puis l'hypothèse nous permet d'en déduire $t_{P\wedge Q} \Vdash P \wedge Q$.

Enfin, on a bien $\mathcal{FV}(t_P) = \mathcal{FV}(t_Q) = \emptyset$ implique $\mathcal{FV}(t_{P\wedge Q}) = \emptyset$ et donc si t_P et t_Q sont uniformes alors $t_{P\wedge Q}$ l'est aussi. ☺

Le lemme suivant montre que, dans $\mathcal{F}^2$, les hypothèse des deux lemmes précédents sont vérifiées :

4.2.11 Lemme
Si $\Gamma, x : \lfloor P\rfloor \vdash_{\mathcal{F}^2} x \Vdash P : \mathsf{prop}$ *et* $\Gamma, x : \lfloor Q\rfloor \vdash x \Vdash Q : \mathsf{prop}$, *alors*

$$\Gamma \vdash_{\mathcal{F}^2} \forall x : \lfloor P\rfloor . x \Vdash P \to \forall y : \lfloor Q\rfloor . y \Vdash Q \to (\mathsf{pair}[\lfloor P\rfloor, \lfloor Q\rfloor]\,x\,y) \Vdash P \wedge Q$$

et

$$\Gamma \vdash_{\mathcal{F}^2} \forall c : \lfloor P\rfloor \times \lfloor Q\rfloor . c \Vdash P \wedge Q \to (\mathsf{proj}^1[\lfloor P\rfloor, \lfloor Q\rfloor]\,c) \Vdash P \wedge (\mathsf{proj}^2[\lfloor P\rfloor, \lfloor Q\rfloor]\,c) \Vdash Q$$

Démonstration On montre les deux formules suivantes sous le contexte Γ :

- $\forall x : \lfloor P \rfloor . x \Vdash P \to \forall y : \lfloor Q \rfloor . y \Vdash Q \to (\mathsf{pair}[\lfloor P \rfloor, \lfloor Q \rfloor]\, x\, y) \Vdash P \wedge Q$:
  Supposons $x \Vdash P$ (H1) et $y \Vdash Q$ (H2). Il faut montrer $(\mathsf{pair}[\lfloor P \rfloor, \lfloor Q \rfloor]\, x\, y) \Vdash P \wedge Q$; cette formule se déplie en :
  $$\begin{array}{l}\forall \gamma : \star, X : \gamma \to \lceil \star \rceil, f : \lfloor P \rfloor \to \lfloor Q \rfloor \to \gamma. \\ \quad (\forall x : \lfloor P \rfloor, x \Vdash P \to \forall y : \lfloor Q \rfloor . y \Vdash Q \to X\,(f\, x\, y)) \to \\ \quad\quad X\,((\mathsf{pair}[\lfloor P \rfloor, \lfloor Q \rfloor]\, x\, y)\gamma\, f)\end{array}$$
  Or la conclusion $X\,((\mathsf{pair}[\lfloor P \rfloor, \lfloor Q \rfloor]\, x\, y)\gamma\, f)$ se réduit en $X\,(f\, x\, y)$. On peut utiliser la prémisse $\forall x : \lfloor P \rfloor, x \Vdash P \to \forall y : \lfloor Q \rfloor . y \Vdash Q \to X\,(f\, x\, y)$ et (H1) et (H2) pour démontrer $X\,(f\, x\, y)$ et conclure.
- $\forall c : \lfloor P \rfloor \times \lfloor Q \rfloor . c \Vdash P \wedge Q \to (\mathsf{proj}^1[\lfloor P \rfloor, \lfloor Q \rfloor]\, c) \Vdash P \wedge (\mathsf{proj}^1[\lfloor P \rfloor, \lfloor Q \rfloor]\, c) \Vdash Q$:
  L'hypothèse $c \Vdash P \wedge Q$ se déplie en
  $$\begin{array}{l}\forall \gamma : \star, X : \gamma \to \lceil \star \rceil, f : \lfloor P \rfloor \to \lfloor Q \rfloor \to \gamma. \\ \quad (\forall x : \lfloor P \rfloor, x \Vdash P \to \forall y : \lfloor Q \rfloor . y \Vdash Q \to X\,(f\, x\, y)) \to \\ \quad\quad X\,(c\,\gamma\, f)\end{array}$$
  Il suffit donc de l'instancier en prenant $\gamma = \lfloor P \rfloor$ et $f = \lambda x : \lfloor P \rfloor, y : \lfloor Q \rfloor . x$ pour démontrer $(\mathsf{proj}^1[\lfloor P \rfloor, \lfloor Q \rfloor]\, c) \Vdash P$ car $\mathsf{proj}^1[\lfloor P \rfloor, \lfloor Q \rfloor]\, c$ est convertible en $c\, \lfloor P \rfloor\, (\lambda x : \lfloor P \rfloor, y : \lfloor Q \rfloor . x)$. De même on prouve $(\mathsf{proj}^2[\lfloor P \rfloor, \lfloor Q \rfloor]\, c) \Vdash Q$ en prenant $\gamma = \lfloor Q \rfloor$ et $f = \lambda x : \lfloor P \rfloor, y : \lfloor Q \rfloor . y$. ☺

Le lemme précédent montre que, dans $\mathcal{F}^2$, la formule suivante est prouvable :

$$\forall x : \lfloor P \rfloor, y : \lfloor Q \rfloor . x \Vdash P \wedge y \Vdash Q \leftrightarrow (\mathsf{pair}[\lfloor P \rfloor, \lfloor Q \rfloor]\, x\, y) \Vdash P \wedge Q$$

On notera toutefois qu'il ne permet pas de démontrer :

$$\Gamma \vdash \forall c : \lfloor P \rfloor \times \lfloor Q \rfloor . (\mathsf{proj}^1[\lfloor P \rfloor, \lfloor Q \rfloor]\, c) \Vdash P \wedge (\mathsf{proj}^2[\lfloor P \rfloor, \lfloor Q \rfloor]\, c) \Vdash Q \leftrightarrow c \Vdash P \wedge Q$$

Dans les cas où cette propriété est désirée, on peut vouloir admettre l'axiome d'extensionalité des paires :

$$\forall \alpha\, \beta : \star, c : \alpha \times \beta . \mathsf{pair}[\alpha, \beta]\, (\mathsf{proj}^1[\alpha, \beta]\, c)\, (\mathsf{proj}^2[\alpha, \beta]\, c) =_{\alpha \times \beta} c$$

(c'est une formule auto-réalisée et honnête, on verra qu'elle est donc compatible avec la réalisabilité).

Afin de généraliser les pré-réquis nécessaires pour que la réalisabilité soit exploitable dans un système $\mathcal{P}^2$, on introduit la définition ci-dessous. Elle impose que l'absurdité, l'égalité, et le prédicat d'induction soient honnêtes, que les égalités soient uniformément auto-réalisés et que le prédicat d'induction soit auto-réalisé. De plus, elle impose que les réalisateurs de la conjonction et du connecteur existentiel se comportent bien vis-à-vis de la réalisabilité (dans la prochaine sous-section, nous reviendrons plus en détails sur les réalisateurs du connecteur existentiel).

4.2.12 Définition (Adéquat pour la réalisabilité)
On dit de $\mathcal{P}^2$ qu'il est adéquat pour la réalisabilité si $\mathcal{P}^2 \models$ NOTATION-REALIZABILITY et si :

1. l'absurdité est honnête : $\vdash_{\mathcal{P}^2} \forall x : \lfloor \bot \rfloor . x \Vdash \bot \quad \to \quad \bot$
2. les égalités sont honnêtes : $\vdash_{\mathcal{P}^2} \forall \alpha : \star, x\, y\ : \alpha, r : \lfloor x =_\alpha y \rfloor, r \Vdash x =_\alpha y \to x =_\alpha y$

3. les égalités sont uniformément auto-réalisées : il existe un terme clos tt tel que
$$\vdash_{\mathcal{P}^2} \forall \alpha : \star, x\, y\ : \alpha.x =_\alpha y \to \mathsf{tt} \Vdash x =_\alpha y$$
4. le prédicat d'induction est honnête : $\vdash_{\mathcal{P}^2} \forall x\, r : \mathsf{nat}.r \Vdash x \in \mathbb{N} \to x\ \in \mathbb{N}$
5. le prédicat d'induction est auto-réalisé : $\vdash_{\mathcal{P}^2} \forall x \in \mathbb{N}.x \Vdash x \in \mathbb{N}$

et si les réalisateurs de la conjonction et du connecteur existentiel vérifient les conditions suivantes :
- Si $\Gamma \vdash P : \mathsf{prop}$ et $\Gamma \vdash Q : \mathsf{prop}$, alors
  - $\Gamma \vdash_{\mathcal{P}^2} \forall c : \lfloor P \rfloor \times \lfloor Q \rfloor.c \Vdash P \wedge Q \to (\mathsf{proj}^1[\lfloor P \rfloor, \lfloor Q \rfloor]\, c) \Vdash P \wedge (\mathsf{proj}^2[\lfloor P \rfloor, \lfloor Q \rfloor]\, c) \Vdash Q$,
  - $\Gamma \vdash_{\mathcal{P}^2} \forall x : \lfloor P \rfloor.x \Vdash P \to \forall y : \lfloor Q \rfloor.y \Vdash Q \to (\mathsf{pair}[\lfloor P \rfloor, \lfloor Q \rfloor]\, x\, y) \Vdash P \wedge Q$,
- Si $\Gamma, x : \tau \vdash P : \mathsf{prop}$, alors
  - $\Gamma \vdash_{\mathcal{P}^2} \forall r : (\!|\lfloor P \rfloor|\!).r \Vdash (\exists x : \tau.P) \quad \to \quad \exists x : \tau.(\mathsf{open}[\lfloor P \rfloor]\, r) \Vdash P$
  - $\Gamma \vdash_{\mathcal{P}^2} \forall x : \tau, r : \lfloor P \rfloor.r \Vdash P \quad \to \quad (\mathsf{close}[\lfloor P \rfloor]\, r) \Vdash \exists x : \tau.P$

enfin on demande également que la condition $\mathcal{P}^2 \models$ STRENGTHENING $\wedge$ WEAK-DOWNSTABLE du théorème 4.1.8 soit vérifiée.
On note $\mathcal{P}^2 \models$ REALIZABILITY.

Nous avons déjà prouvé (lemmes 4.2.4 et 4.2.11) que $\mathcal{F}^2$ vérifie toutes ces conditions à l'exception des deux dernières. Nous reviendrons dans la sous-section 4.2.3 sur ces propriétés concernant les réalisateurs du connecteur existentiel, puis nous montrerons que $\mathcal{F}^2 \models$ REALIZABILITY (lemme 4.2.29).

Il manque néanmoins une "bonne propriété" aux systèmes qui vérifient REALIZABILITY ; il s'agit du fait que la formule $\mathsf{datatype} = \forall x\, r : \mathsf{nat}.r \Vdash x \in \mathbb{N} \to r =_{\mathsf{nat}} x$ soit dérivable. Nous n'avons pas incorporé cette propriété à la définition car $\mathcal{F}^2$ ne la vérifie pas.

Dans les systèmes qui satisfont REALIZABILITY, si on peut démontrer $\mathsf{datatype}$, alors les réalisateurs de $x \in \mathbb{N}$ sont exactement les objets de type nat égaux à x et qui vérifient le principe d'induction :

4.2.13 Lemme
Si $\mathcal{P}^2 \models$ REALIZABILITY *et si* $\Gamma \vdash_{\mathcal{P}^2} \mathsf{datatype}$ *alors* $\Gamma \vdash_{\mathcal{P}^2} \forall x\, r : \mathsf{nat}.r \Vdash x \in \mathbb{N} \Leftrightarrow r =_{\mathsf{nat}} x \wedge x \in \mathbb{N}$.

Démonstration
- $\Rightarrow$: $r =_{\mathsf{nat}} x$ est donné par $\mathsf{datatype}$ et $x \in \mathbb{N}$ par le fait que $x \in \mathbb{N}$ est honnête.
- $\Leftarrow$: Grâce à l'égalité $r =_{\mathsf{nat}} x$ on se ramène à démontrer que $\forall x \in \mathbb{N}.x \Vdash x \in \mathbb{N}$ ce qui est une des conditions de REALIZABILITY. ☺

En combinant les lemmes précédents (4.2.4, 4.2.7, 4.2.8, 4.2.9, 4.2.10) on obtient le critère syntaxique suivant :

4.2.14 Lemme (Grammaire pour les formules honnêtes et auto-réalisées)
Supposons $\mathcal{P} \models$ REALIZABILITY. *Soit les trois grammaires mutuellement définies suivantes :*
$$\begin{array}{rcl} H, H_1, H_2 & := & \bot \\ & | & t \in \mathbb{N} \\ & | & t_1 =_\tau t_2 \\ & | & A \to H \\ & | & H_1 \wedge H_2 \\ & | & \forall x^1 : A.H \end{array}$$

$$\begin{array}{rcll} A, A_1, A_2 & := & U & \\ & | & t \in \mathbb{N} & \\ & | & H \to A & \\ & | & A_1 \wedge A_2 & \\ & | & \forall x \in \mathbb{N}.A & (*) \end{array}$$

$$\begin{array}{rcl} U, U_1, U_2 & := & t_1 =_\tau t_2 \\ & | & H \to U \\ & | & U_1 \wedge U_2 \\ & | & \forall x^1 : A.U \end{array}$$

Alors :
- *on a* $\Gamma \vdash_{\mathcal{P}^2} H : \mathsf{prop}$ *implique* H *honnête dans* Γ.
- *on a* $\Gamma \vdash_{\mathcal{P}^2} A : \mathsf{prop}$ *implique* A *auto-réalisée dans* Γ.
- *on a* $\Gamma \vdash_{\mathcal{P}^2} U : \mathsf{prop}$ *implique* U *uniformément auto-réalisée dans* Γ.

où la règle de production (*) est autorisée à condition que $\Gamma \vdash \mathsf{datatype}$.

On verra plus tard (lemme 4.2.27) que, dans $\mathcal{F}^2$, il suffit que Γ implique un principe d'extensionalité sur les entiers pour vérifier la dernière condition.

4.2.15 Exemple
- $\bot_{\mathrm{w}}$ *:* $\forall x : \mathsf{nat}.x =_{\mathsf{nat}} \overline{0}$ *: honnête et uniformément auto-réalisée.*
- $\mathsf{succ\_inj}$ *:* $\forall x\, x : \mathsf{nat}.\mathsf{S}\, x =_{\mathsf{nat}} \mathsf{S}\, y \to x =_{\mathsf{nat}} y$ *: idem.*
- $\mathsf{not\_cyclic}$ *:* $\forall x : \mathsf{nat}.\mathsf{S}\, x =_{\mathsf{nat}} \overline{0} \to \bot_{\mathrm{w}}$ *: idem.*
- $\mathsf{ext\_nat}$ *:* $\forall x\, y : \mathsf{nat}.(\forall \alpha : \star, f : \alpha \to \alpha, z : \alpha.x\, \alpha\, f\, z =_\alpha y\, \alpha\, f\, z) \to x =_{\mathsf{nat}} y$ *: idem.*
- *Toutes les formules prouvables dans un contexte vide sont trivialement honnêtes et uniformément réalisées.*
- $\forall x : \mathsf{nat}.x \in \mathbb{N}$ *: honnête.*

La notion de contexte auto-réalisé est importante puisque elle nous donne une condition suffisante pour obtenir des réalisateurs canoniques pour les axiomes.

4.2.16 Définition (Contexte auto-réalisé, contexte uniformément auto-réalisé)
On définit les contextes auto-réalisés Γ par induction sur la structure des contextes :
- Le contexte vide est auto-réalisé.
- Si $\mathrm{lvl}(A) = 1$ et Γ auto-réalisé alors $\Gamma, x : A$ est auto-réalisé.
- Si $\mathrm{lvl}(A) = 2$, Γ auto-réalisé et si A est auto-réalisée dans Γ, alors $\Gamma, x : A$ est auto-réalisé.

On dit d'un contexte de la forme Γ, Δ qu'il est auto-réalisé uniformément dans Δ s'il satisfait la définition inductive (sur Δ) suivante :
- Le contexte $\Gamma, \langle\rangle$ est auto-réalisé uniformément dans $\langle\rangle$ si Γ est auto-réalisé.
- Si $\mathrm{lvl}(A) = 1$ et Γ, Δ auto-réalisé uniformément dans Δ alors $\Gamma, \Delta, x : A$ est auto-réalisé uniformément dans $\Delta, x : A$.
- Si $\mathrm{lvl}(A) = 2$, Γ, Δ auto-réalisé uniformément dans Δ et si A est uniformement auto-réalisée dans Γ, Δ, alors $\Gamma, \Delta, x : A$ est auto-réalisé uniformémént dans $\Delta, x : A$.

Enfin, on dira simplement qu'un contexte Γ est uniformément auto-réalisés si $\langle\rangle, \Gamma$ est auto-réalisé uniformément dans Γ.

4.2.17 Exemple

- *Le contexte* **HA** *(voir 157) est uniformément auto-réalisé.*
- *Les contextes de "spécification", c-à-d. formés à partir de variables du premier ordre et d'égalités quantifiées, sont uniformémement auto-réalisables. Comme par exemple :*

$$\Delta = f : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}, h_1 : \forall x : \mathsf{nat}. f\, x\, \overline{0} =_{\mathsf{nat}} x, h_2 : \forall x\, y : \mathsf{nat}. f\, x\, (\mathsf{S}\, y) =_{\mathsf{nat}} \mathsf{S}\, (f\, x\, y)$$

- *Le contexte suivant :*

$$\Gamma = x : \mathsf{nat}, h_1 : x \in \mathbb{N}$$

  est auto-réalisé pas uniformément (le réalisateur canonique de $x \in \mathbb{N}$ est x qui n'est pas clos). Cela signifiera que le réalisateur produit par le théorème 4.2.22 (voir plus loin) pourra contenir la variable x.
- *La concaténation Γ, Δ des deux contextes précédents est auto-réalisée uniformément dans Δ. Ainsi, x peut apparaître dans les réalisateurs produit par le théorème 4.2.22 mais pas f.*

Les contextes auto-réalisés ont l'avantage d'être compatibles avec la réalisabilité dans le sens suivant (ce sera le théorème 4.2.22) :

$$\text{Si } \Gamma \vdash_{\mathcal{P}^2} \pi : P \text{ alors il existe un programme } p \text{ tel que } \Gamma \vdash_{\mathcal{P}^2} p \Vdash P.$$

à condition que Γ et P ne contiennent pas de variable libre de second niveau, ce que l'on appellera "être clos au second niveau" :

4.2.18 Définition (Clos au second niveau)
Un terme P est clos au second niveau si $x \in \mathcal{FV}(P)$ implique $\mathrm{lvl}(x) = 1$ et un contexte Γ est clos au second niveau si pour tout $x : A \in \Gamma$, A est clos au second niveau.

Un exemple de contexte qui n'est pas clos au second niveau est le contexte de $\mathcal{F}^2$ suivant : $X : \mathsf{nat} \to \mathsf{prop}, h : X\, \overline{3}$ car X est une variable de second niveau qui apparaît dans "$X\, \overline{3}$". La théorie que l'on développe ici ne permet pas de bien gérer les contextes qui ne sont pas clos au second niveau.

Pour prouver le théorème 4.2.22, il nous suffit de substituer dans le séquent $\|\Gamma\| \vdash \|\pi\| : \lfloor \pi \rfloor \Vdash P$ obtenu par le théorème 4.1.8, pour chaque $x^2 : F \in \Gamma$ les occurrences de x par le réalisateur canonique t_F et les occurrences de x_R par la preuve $(\vartheta_F\, x)$ (où x désigne alors une preuve de F).

Pour ce faire, nous introduisons la définition suivante :

4.2.19 Définition (Substitution par les réalisateurs canoniques)
Soit Γ un contexte auto-réalisé, on définit, par induction sur la structure[3] de Γ, la fonction $\mathbf{subst}_\Gamma$ des termes de $\mathcal{P}^2$ dans les termes de $\mathcal{P}^2$ de la façon suivante :

$$\begin{aligned}
\mathbf{subst}_{\langle\rangle}(M) &= M \\
\mathbf{subst}_{x:A,\Gamma}(M) &= \mathbf{subst}_\Gamma(M) \text{ si } \mathrm{lvl}(A) = 1 \\
\mathbf{subst}_{x:A,\Gamma}(M) &= \mathbf{subst}_\Gamma(M)[t_A/x, (\vartheta_A\, x)/x_R] \text{ si } \mathrm{lvl}(A) = 2
\end{aligned}$$

avec dans le dernier cas $\Gamma \vdash \vartheta_A : A \to t_A \Vdash A$ (ϑ_A et t_A existent car le contexte est auto-réalisé).

3. On notera que c'est à dessein que nous définissons cette fonction en commençant par la première variable du contexte.

On montre alors le lemme suivant :

4.2.20 Lemme
Soit Δ, Γ *un contexte auto-réalisé clos au second niveau.*
Si $\Delta, \|\Gamma\| \vdash M : T$, *alors* $\Delta, \Gamma \vdash \mathbf{subst}_\Gamma(M) : \mathbf{subst}_\Gamma(T)$.

Démonstration Par récurrence sur la longueur de Γ :
- Si Γ est vide, l'implication est triviale.
- Si Γ est de la forme $x : A, \Gamma'$, on distingue deux cas :
  - $\mathrm{lvl}(x) = \mathrm{lvl}(A) = 1$: Dans ce cas $\|\Gamma\| = x : A, \|\Gamma'\|$ et $\mathbf{subst}_\Gamma(M) = \mathbf{subst}_{\Gamma'}(M)$, on peut donc conclure directement grâce à l'hypothèse de récurrence (on a fait diminuer Γ et grandir Δ).
  - $\mathrm{lvl}(x) = \mathrm{lvl}(A) = 2$: Dans ce cas $\|\Gamma\| = x : \lfloor A \rfloor, x_R : x \Vdash A, \|\Gamma'\|$. Comme A est auto-réalisée dans Δ, on a
  $$\Delta \vdash \vartheta_A : A \to t_A \Vdash A$$
  pour une preuve ϑ_A et un programme t_A. On peut donc utiliser les lemmes 1.1.23 et 4.1.4, pour en déduire $\Delta, x_R : t_A \Vdash A, \|\Gamma'\|[t_A/x] \vdash M[t_A/x] : T[t_A/x]$. D'après le lemme 1.1.24, on en déduit que :
  $$\Delta, x : A, x_R : t_A \Vdash A, \|\Gamma'\|[t_A/x] \vdash M[t_A/x] : T[t_A/x]$$
  Or $\Delta \vdash \vartheta_A\, x : t_A \Vdash A$, on peut donc utiliser le lemme de substitution (lemme 1.1.23) afin de démontrer que :
  $$\Delta, x : A, \|\Gamma'\|[t_A/x, (\vartheta_A\, x)/x_R] \vdash M[t_A/x, (\vartheta_A\, x)/x_R] : T[t_A/x, (\vartheta_A\, x)/x_R]$$
  Or comme Γ' est clos au second niveau, on a $x, x_R \notin \mathcal{FV}\, \|\Gamma'\|$ et donc $\|\Gamma'\|[t_A/x, (\vartheta_A\, x)/x_R] = \|\Gamma'\|$. On a donc montré que :
  $$\Delta, x : A, \|\Gamma'\| \vdash M[t_A/x, (\vartheta_A\, x)/x_R] : T[t_A/x, (\vartheta_A\, x)/x_R]$$
  On peut donc appliquer l'hypothèse d'induction pour en déduire que :
  $$\Delta, x : A, \Gamma' \vdash \mathbf{subst}_{\Gamma'}(M[t_A/x, (\vartheta_A\, x)/x_R]) : \mathbf{subst}_{\Gamma'}(T[t_A/x, (\vartheta_A\, x)/x_R])$$
  or $\mathbf{subst}_{\Gamma'}(M[t_A/x, (\vartheta_A\, x)/x_R]) = \mathbf{subst}_\Gamma(M)$ $\mathbf{subst}_{\Gamma'}(T[t_A/x, (\vartheta_A\, x)/x_R]) = \mathbf{subst}_\Gamma(T)$ on a donc démontré :
  $$\Delta, \Gamma \vdash \mathbf{subst}_\Gamma(M) : \mathbf{subst}_\Gamma(T)$$

4.2.21 Lemme
Soit Γ *un contexte autoréalisé, tel que* $\Gamma \vdash P : \lceil s \rceil$. *Si* P *est clos au second niveau et* $\mathrm{lvl}(t) = 1$:
$$\mathbf{subst}_\Gamma(t \Vdash P) = \mathbf{subst}_\Gamma(t) \Vdash P$$

Démonstration (Esquisse) Supposons $x^2 : A \in \Gamma$. On montre que :
$$\|P\|[t_A/x, \vartheta_A\, x/x_R] = \|P\|$$

En inspectant la définition de $\|\cdot\|$ on se convainc que le seul moyen pour que $x^1 \in \mathcal{FV}(\|P\|)$ ou $x^2_R \in \mathcal{FV}(\|P\|)$ est d'avoir $x^2 \in \mathcal{FV}(P)$. Comme P est close au second niveau on en déduit que $x^1, x^2_R \notin \mathcal{FV}(\|P\|)$ et donc que la substitution ne modifie pas le terme.

Enfin, on peut montrer pour tout P clos au second niveau, on a

$$(t \Vdash P)[t_A/x, \vartheta_A\, x/x_R] = t[t_A/x] \Vdash P$$

en procédant par induction sur la structure de P. ☺

4.2.22 Théorème (Adéquation pour les contextes auto-réalisés)
Soit $\mathcal{P}$ un CTS tel que $\mathcal{P}^2 \models$ STRENGTHENING $\wedge$ WEAK-DOWNSTABLE.
Et soient Γ un contexte auto-réalisé et P un terme tous deux clos au second niveau.
Si $\Gamma \vdash_{\mathcal{P}^2} P$ alors il existe p tel que $\Gamma \vdash_{\mathcal{P}^2} p \Vdash P$.

Démonstration Supposons que l'on ait $\Gamma \vdash_{\mathcal{P}^2} \pi : P$. Alors d'après le théorème 4.1.8, on a $\|\Gamma\| \vdash_{\mathcal{P}^2} \|\pi\| : \lfloor\pi\rfloor \Vdash P$. D'après le lemme 4.2.20, on en déduit que

$$\Gamma \vdash_{\mathcal{P}^2} \mathbf{subst}_\Gamma(\|\pi\|) : \mathbf{subst}_\Gamma\,(\lfloor\pi\rfloor \Vdash P)$$

puis le lemme 4.2.21 nous montre que $\mathbf{subst}_\Gamma\,(\lfloor\pi\rfloor \Vdash P) = \mathbf{subst}_\Gamma(\lfloor\pi\rfloor) \Vdash P$. On en déduit que $p = \mathbf{subst}_\Gamma(\lfloor\pi\rfloor)$ convient. ☺

Enfin, on remarque que l'on peut généraliser sans problème ce théorème aux contextes uniformément auto-réalisés de la manière suivante. Cette généralisation est utile lorsque que l'on souhaite s'assurer que certaines variables du premier niveau n'apparaissent pas dans le réalisateur fourni par le théorème précédent.

4.2.23 Théorème (Adéquation pour les contextes uniformément auto-réalisés)
Soit $\mathcal{P}$ un CTS tel que $\mathcal{P}^2 \models$ STRENGTHENING $\wedge$ WEAK-DOWNSTABLE.
Soient Γ, Δ un contexte auto-réalisé uniformément dans Δ et P un terme tous deux clos au second niveau.

Si $\Gamma, \Delta \vdash_{\mathcal{P}^2} P$ alors il existe p tel que $\Gamma, \Delta \vdash_{\mathcal{P}^2} p \Vdash P$ et $\mathrm{filter}(\Gamma) \vdash_{\mathcal{P}} p : \lfloor P\rfloor$.

Démonstration (Esquisse) Soit $x : F \in \Delta$, alors le réalisateur t_F est uniforme (c-à-d. $\mathcal{FV}(t_F) = \emptyset$) et donc il ne dépend d'aucune variable (du premier niveau) de Δ. Or, seuls les réalisateurs canoniques de Δ peuvent introduire des variables de Δ lors du calcul de $\mathbf{subst}_\Gamma(\lfloor\pi\rfloor)$. On en déduit donc que si $x \in \mathcal{FV}(\mathbf{subst}_\Gamma(\lfloor\pi\rfloor))$, alors $x \notin \Delta$. Or, puisque que l'on a $\mathcal{P}^2 \models$ STRENGTHENING, $\Gamma, \Delta \vdash p : \lfloor P\rfloor$ implique $\Gamma \vdash p : \lfloor P\rfloor$ et enfin le lemme 3.4.2 nous permet de déduire $\mathrm{filter}(\Gamma) \vdash_{\mathcal{P}} p : \lfloor P\rfloor$. ☺

4.2.2 Type de donnée dans $\mathcal{F}^2$

Dans $\mathcal{F}$ les entiers sont représentés par les habitants du type $\mathsf{nat} = \forall\alpha : \star.(\alpha \to \alpha) \to \alpha \to \alpha$. On peut montrer aisément que les seuls termes clos qui habitent ce type sont exactement les entiers de Church. Mais la logique $\mathcal{F}^2$ n'impose pas que tous les habitants de nat soient générés par la syntaxe. En effet, on dérive facilement le séquent $\vdash_{\mathcal{F}^2} \overline{n} \in \mathbb{N}$ pour tout entier n fixé, mais nous allons voir que l'on ne peut pas démontrer la formule $\forall x : \mathsf{nat}.x \in \mathbb{N}$.

Le prédicat $x \in \mathbb{N}$ signifie que x satisfait un principe d'induction, c'est-à-dire qu'il est satisfait par tous les prédicats qui contiennent $\overline{0}$ et qui sont stables par successeurs. Ainsi, on peut comprendre $x \in \mathbb{N}$ comme étant la "façon interne" que le système a de voir les entiers obtenus à partir de $\overline{0}$ et par itérations successives de la fonction S.

Tout comme pour l'encodage du produit (et comme on le verra également un peu plus avant pour l'encodage du connecteur existentiel), l'encodage des entiers de $\mathcal{F}$ n'est pas aussi souple que ce que l'on pourrait souhaiter. En effet, on ne peut pas démontrer la propriété $\mathsf{ext\_nat}$ d'extensionalité suivante :

$$\mathsf{ext\_nat} = \forall x\, y : \mathsf{nat}.x \overset{\text{ext}}{=}_{\mathsf{nat}} y \to x =_{\mathsf{nat}} y$$

où la notation $x \overset{\text{ext}}{=}_{\mathsf{nat}} y$ est définie par :

$$x \overset{\text{ext}}{=}_{\mathsf{nat}} y = \forall \alpha : \star, f : \alpha \to \alpha, z : \alpha.x\, \alpha\, f\, z =_\alpha y\, \alpha\, f\, z$$

La formule $x \overset{\text{ext}}{=}_{\mathsf{nat}} y$ signifie que les entiers x et y fournissent toujours le même résultat lorsqu'ils sont utilisés comme des itérateurs. Bien sûr, on peut montrer que tous les entiers de Church satisfont ce principe car

$$\vdash_{\mathcal{F}^2} \overline{n} \overset{\text{ext}}{=}_{\mathsf{nat}} \overline{m} \to \overline{n} =_{\mathsf{nat}} \overline{m}$$

pour tout entier n et m (il suffit pour cela de remarquer que $\overline{n}\,\mathsf{nat}\,\mathsf{S}\,\overline{0} \trianglerighteq \overline{n}$ pour tout entier n). Plus généralement, on peut montrer

$$\vdash_{\mathcal{F}^2} \forall x \in N.x\,\mathsf{nat}\,\mathsf{S}\,\overline{0} =_{\mathsf{nat}} x$$

en se ramenant à démontrer, grâce au principe d'induction, que :

- $(\overline{0}\,\mathsf{nat}\,\mathsf{S}\,\overline{0}) =_{\mathsf{nat}} \overline{0}$: Trivial par réflexivité car $\overline{0}\,\mathsf{nat}\,\mathsf{S}\,\overline{0} \trianglerighteq \overline{0}$.
- $\forall y : \mathsf{nat}.(y\,\mathsf{nat}\,\mathsf{S}\,\overline{0}) =_{\mathsf{nat}} y \to (\mathsf{S}\,y\,\mathsf{nat}\,\mathsf{S}\,\overline{0}) =_{\mathsf{nat}} \mathsf{S}\,y$: on utilise l'hypothèse pour transformer le but en $(\mathsf{S}\,y\,\mathsf{nat}\,\mathsf{S}\,\overline{0}) =_{\mathsf{nat}} (\mathsf{S}\,(y\,\mathsf{nat}\,\mathsf{S}\,\overline{0}))$ et on conclut par réflexivité car $\mathsf{S}\,y\,\mathsf{nat}\,\mathsf{S}\,\overline{0} \trianglerighteq \mathsf{S}\,(y\,\mathsf{nat}\,\mathsf{S}\,\overline{0})$.

On en déduit alors facilement le lemme suivant :

4.2.24 Lemme
On a $\vdash_{\mathcal{F}^2} \mathsf{ext\_nat}$*, c'est-à-dire :*

$$\vdash_{\mathcal{F}^2} \forall x \in \mathbb{N}.\forall y \in \mathbb{N}.x \overset{ext}{=}_{\mathsf{nat}} y \to x =_{\mathsf{nat}} y$$

Le prédicat $x \in \mathbb{N}$ est parfois appelé un "type de donnée" pour la réalisabilité dans le sens où l'on peut prouver que les réalisateurs de $x \in \mathbb{N}$ sont extensionellement égaux à x.

4.2.25 Lemme (Type de donnée dans $\mathcal{F}^2$)
Il existe une preuve π *du séquent :*

$$\vdash_{\mathcal{F}^2} \pi : \forall x\, y : \mathsf{nat}.y \Vdash x \in \mathbb{N} \to x \overset{ext}{=}_{\mathsf{nat}} y$$

Démonstration Supposons $y \Vdash x \in \mathbb{N}$ (H) pour deux variables x et y de type nat et soit α de type $\star$, f de type $\alpha \to \alpha$ et z de type α.

Montrons $x\,\alpha\,f\,z =_\alpha y\,\alpha\,f\,z$.
La proposition $y \Vdash x \in \mathbb{N}$ se déplie en :

$$\begin{array}{l}\forall \alpha : \star, X : \mathsf{nat} \to \alpha \to \lceil\star\rceil, f : \alpha \to \alpha.\\ \quad (\forall x : \mathsf{nat}, z' : \alpha. X\,x\,z' \to X\,(\mathsf{S}\,x)\,(f\,z')) \to\\ \quad\quad \forall z : \alpha. X\,\overline{0}\,z \to X\,x\,(y\,\alpha\,f\,z)\end{array}$$

On peut utiliser (H) et donc instancier X en $\lambda x : \mathsf{nat}, y : \alpha. x\,\alpha\,f\,z =_\alpha y$ pour se ramener à démontrer :

- $\forall x : \mathsf{nat}, z' : \alpha. x\,\alpha\,f\,z =_\alpha z' \to (\mathsf{S}\,x\,\alpha\,f\,z) =_\alpha (f\,z')$: On peut donc utiliser $x\,\alpha\,f\,z =_\alpha z'$ pour réécrire le but $(\mathsf{S}\,x\,\alpha\,f\,z) =_\alpha (f\,(x\,\alpha\,f\,z))$. Or le terme $\mathsf{S}\,x\,\alpha\,f\,z \trianglerighteq f\,(x\,\alpha\,f\,z)$ on peut donc utiliser la réflexivité pour conclure.
- $\overline{0}\,\alpha\,f\,z =_\alpha z$: Or on a $\overline{0}\,\alpha\,f\,z \trianglerighteq z$ et on peut donc conclure par réflexivité. ☺

Une des conséquences du lemme précédent est l'indépendance de $\forall x : \mathsf{nat}. x \in \mathbb{N}$ dans $\mathcal{F}^2$:

4.2.26 Lemme
Soit Γ *un contexte auto-réalisé clos au second niveau.*
Si $\Gamma \vdash_{\mathcal{F}^2} \forall x : \mathsf{nat}. x \in \mathbb{N}$, *alors* $\Gamma \vdash_{\mathcal{F}^2} \bot_{\mathrm{w}}$.

Démonstration Soit π une preuve de $\Gamma \vdash \forall x : \mathsf{nat}. x \in \mathbb{N}$. D'après le théorème 4.2.22, on peut prouver dans le contexte Γ que $t \Vdash \forall x : \mathsf{nat}. x \in \mathbb{N}$ pour un programme t c'est-à-dire que $\forall x : \mathsf{nat}. t \Vdash x \in \mathbb{N}$ et d'après le lemme précédent on en déduit que $\forall x : \mathsf{nat}. t \overset{\mathrm{ext}}{=}_{\mathsf{nat}} x$.
Soit x de type nat. On peut utiliser π pour démontrer que $x \in \mathbb{N}$ et $\lfloor\pi\rfloor \in \mathbb{N}$ et donc d'après le lemme 4.2.24 on en déduit que $\lfloor\pi\rfloor =_{\mathsf{nat}} x$. On a donc montré que pour tout x de type nat, on a $t =_{\mathsf{nat}} x$ et donc $\overline{0} =_{\mathsf{nat}} x$. On a donc bien dérivé $\bot_{\mathrm{w}}$. ☺

On déduit que si $\Gamma \nvdash \bot_{\mathrm{w}}$ (c'est-à-dire il existe un modèle de Γ où nat est interprété par un ensemble à au moins deux éléments), alors $\forall x : \mathsf{nat}. x \in \mathbb{N}$ n'est pas démontrable dans le contexte Γ.

4.2.27 Lemme
Si $\Gamma \vdash_{\mathcal{F}^2} \mathsf{ext\_nat}$ *alors* $\Gamma \vdash_{\mathcal{F}^2} \forall x : \mathsf{nat}, y : \mathsf{nat}. y \Vdash x \in \mathbb{N} \Leftrightarrow (y \in \mathbb{N} \wedge x =_{\mathsf{nat}} y)$

Démonstration C'est une conséquence directe des lemmes 4.2.4, 4.2.25 et 4.2.24. ☺

4.2.3 Réalisateurs du connecteur existentiel

Dans $\mathcal{F}^2$, les réalisateurs sont peu marqués par le passage au travers du connecteur existentiel. En effet, le type des réalisateurs de $\exists x : \mathsf{nat}. P$ est égal à

$$\lfloor\exists x : \alpha. P\rfloor = \lfloor\forall X : \lceil\star\rceil. (\forall x : \alpha. P \to X) \to X)\rfloor = \forall \gamma : \star. (\lfloor P\rfloor \to \gamma) \to \gamma = (\!|\lfloor P\rfloor|\!)$$

où la notation $(\!|\tau|\!)$ est utilisée pour désigner le type $\forall \gamma : \star. (\tau \to \gamma) \to \gamma$. Ce constructeur de type est un opérateur monadique qui encapsule les objets de type τ. On dispose de deux opérations, l'une pour injecter les objets de type τ dans $(\!|\tau|\!)$ et l'autre pour récupérer les valeurs encapsulées :

$$\begin{array}{lcl}\mathsf{close}[\alpha] & : & \alpha \to (\!|\alpha|\!)\\ \mathsf{close}[\alpha] & = & \lambda x : \alpha, \gamma : \star, f : \alpha \to \gamma. f x\\ \mathsf{open}[\alpha] & : & (\!|\alpha|\!) \to \alpha\\ \mathsf{open}[\alpha] & = & \lambda f : (\!|\alpha|\!). f\,\alpha\,(\lambda x : \alpha. x)\end{array}$$

On a de plus $\mathsf{open}[\tau]\,(\mathsf{close}[\tau]\,t) \trianglerighteq t$ on peut démontrer dans $\mathcal{F}^2$ par réflexivité que $\mathsf{open}[\tau]$ est une rétraction de $\mathsf{close}[\tau]$:

$$\forall \alpha : \star, x : \alpha.\mathsf{open}[\alpha]\,(\mathsf{close}[\alpha]\,x) =_\alpha x$$

Néanmoins, on ne peut pas prouver dans $\mathcal{F}^2$ que c'est une bijection (c'est une nouvelle fois un problème d'extensionalité).

On peut montrer que $\mathcal{F}^2$ satisfait les deux dernières conditions nécessaires pour vérifier REALIZABILITY :

4.2.28 Lemme
Supposons $\Gamma, x : \tau \vdash P : \mathsf{prop}$ *et* $\Gamma, x : \tau, r : \lfloor P \rfloor \vdash_{\mathcal{F}^2} \pi : r \Vdash P : \mathsf{prop}$ *, alors*

$$\Gamma \vdash_{\mathcal{F}^2} \forall x : \tau, r : \lfloor P \rfloor . r \Vdash P \quad \to \quad (\mathsf{close}[\lfloor P \rfloor]\, r) \Vdash \exists x : \tau . P$$

et

$$\Gamma \vdash_{\mathcal{F}^2} \forall r : (\!|\lfloor P \rfloor|\!) . r \Vdash (\exists x : \tau . P) \quad \to \quad \exists x : \tau . (\mathsf{open}[\lfloor P \rfloor]\, r) \Vdash P$$

Démonstration
– La formule $\mathsf{close}[r] \Vdash \exists x : \tau . P$ se déplie en :

$$\begin{array}{l} \forall \gamma : \star, X : \gamma \to \lceil \star \rceil, f : \lfloor P \rfloor \to \gamma . \\ \quad (\forall x : \tau, y : \lfloor P \rfloor . y \Vdash P \to X\,(f\,y)) \to \\ \quad\quad X\,(\mathsf{close}[\lfloor P \rfloor]\, r\, \gamma\, f) \end{array}$$

Or $\mathsf{close}[\lfloor P \rfloor]\, r\, \gamma\, f \trianglerighteq f\, r$. On se ramène donc à démontrer :

$$\begin{array}{l} \forall \gamma : \star, X : \gamma \to \lceil \star \rceil, f : \lfloor P \rfloor \to \gamma . \\ \quad (\forall x : \tau, y : \lfloor P \rfloor . y \Vdash P \to X\,(f\,y)) \to \\ \quad\quad X\,(f\, r) \end{array}$$

Pour cela il suffit donc d'appliquer la prémisse $\forall x : \tau, y : \lfloor P \rfloor . y \Vdash P \to X\,(f\,y)$ en utilisant l'hypothèse $r \Vdash P$.

– La formule $r \Vdash \exists x : \tau . P$ se déplie en

$$\begin{array}{l} \forall \gamma : \star, X : \gamma \to \lceil \star \rceil, f : \lfloor P \rfloor \to \gamma . \\ \quad (\forall x : \tau, y : \lfloor P \rfloor . y \Vdash P \to X\,(f\,y)) \to \\ \quad\quad X\,(r\, \gamma\, f) \end{array}$$

on peut donc l'utiliser en prenant $\gamma = \lfloor P \rfloor$, $X = \lambda z : \lfloor P \rfloor . \exists x : \tau . z \Vdash P$, $f = \lambda x : \lfloor P \rfloor . x$ puis en prouvant

$$\forall x : \tau, y : \lfloor P \rfloor . y \Vdash P \to (\lambda z : \lfloor P \rfloor . \exists x : \tau . z \Vdash P)\,((\lambda x : \lfloor P \rfloor . x)\, y)$$

qui se réduit en la tautologie $\forall x : \tau, y : \lfloor P \rfloor . y \Vdash P \to \exists x : \tau . y \Vdash P$. On déduit alors que $\exists x : \tau . r\, \lfloor P \rfloor\, (\lambda x : \lfloor P \rfloor . x) \Vdash P$ qui est bien convertible en $\exists x : \tau . (\mathsf{open}[\tau]\, r) \Vdash P$. ☺

On en déduit bien :

4.2.29 Lemme
Pour tout $\mathcal{P}^2$ *tel que* $\mathcal{F}^2 \subseteq \mathcal{P}^2$ *et* $\mathcal{P}^2 \models$ STRENGTHENING $\wedge$ WEAK-DOWNSTABLE *:*

$$\mathcal{P}^2 \models \text{REALIZABILITY}$$

On peut alors utiliser la réalisabilité pour démontrer une variante de la propriété de l'existence (pour les formules auto-réalisées) :

4.2.30 Théorème (Propriété de l'existence pour les formules auto-réalisées)
Supposons $\mathcal{P}^2 \models$ REALIZABILITY.
Soit Γ, Δ *un contexte auto-réalisé uniformément dans* Δ.
Si

$$\Gamma, \Delta \vdash_{\mathcal{P}^2} \mathsf{datatype}$$

et si

$$\Gamma, \Delta \vdash_{\mathcal{P}^2} \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}. \exists y_1 \in \mathbb{N}, \ldots, y_m \in \mathbb{N}. P$$

où P *est une formule honnête, alors il existe* m *termes clos* t_1, ..., t_m *tels que :*

$$\Gamma, \Delta \vdash_{\mathcal{P}^2} \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}. \left(\begin{array}{c} (t_1\, x_1 \cdots x_n) \in \mathbb{N} \\ \wedge \cdots \wedge \\ (t_m\, x_1 \cdots x_n) \in \mathbb{N} \\ \wedge \\ P[t_1\, x_1 \cdots x_n / y_1, \ldots, t_m\, x_1 \cdots x_n / y_m] \end{array} \right)$$

avec

$$\mathrm{filter}(\Gamma) \vdash_{\mathcal{P}} t_k : \overbrace{\mathsf{nat} \to \cdots \mathsf{nat}}^{n \text{ fois}} \to \mathsf{nat}$$

pour $1 \leq k \leq m$.

Démonstration On montre le cas $n = m = 1$, le cas général étant similaire. Supposons $\Gamma, \Delta \vdash_{\mathcal{P}^2} \forall x \in \mathbb{N}. \exists y \in \mathbb{N}. P$ d'après le théorème 4.2.23 on a $\Gamma, \Delta \vdash_{\mathcal{P}^2} p \Vdash \forall x \in \mathbb{N}. \exists y \in \mathbb{N}. P$ pour un programme p tel que $\mathrm{filter}(\Gamma) \vdash_{\mathcal{P}} p : \mathsf{nat} \to (\!|\mathsf{nat} \times \lfloor P \rfloor|\!)$. On en déduit d'après REALIZABILITY que (on rappelle que $\exists y \in \mathbb{N}. P$ est égal à $\exists y : \mathsf{nat}. y \in \mathbb{N} \wedge P$) :

$$\Gamma, \Delta \vdash_{\mathcal{P}^2} \forall x\, r_x : \mathsf{nat}. r_x \Vdash x \in \mathbb{N} \to \exists y : \mathsf{nat}. \left(\begin{array}{c} \mathsf{proj}^1[\mathsf{nat}, \lfloor P \rfloor]\, (\mathsf{open}[\mathsf{nat} \times \lfloor P \rfloor]\, (p\, r_x)) \Vdash y \in \mathbb{N} \\ \wedge \\ \mathsf{proj}^2[\mathsf{nat}, \lfloor P \rfloor]\, (\mathsf{open}[\mathsf{nat} \times \lfloor P \rfloor]\, (p\, r_x)) \Vdash P \end{array} \right)$$

On pose $t = \lambda z : \mathsf{nat}. \mathsf{proj}^1[\mathsf{nat}, \lfloor P \rfloor]\, (\mathsf{open}[\mathsf{nat} \times \lfloor P \rfloor]\, (p\, z))$. Puisque P est honnête, on en déduit que

$$\Gamma \vdash_{\mathcal{P}^2} \forall x\, r_x : \mathsf{nat}. r_x \Vdash x \in \mathbb{N} \to \exists y : \mathsf{nat}. ((t\, r_x) \Vdash y \in \mathbb{N}) \wedge P$$

Enfin, on utilise datatype pour déduire que :

$$\Gamma \vdash_{\mathcal{P}^2} \forall x : \mathsf{nat}. x \in \mathbb{N} \to (t\, x) \in \mathbb{N} \wedge P[f\, x / y]$$

4.2.4 Fonctions prouvablement inductives

On trouvera tous les résultats de cette sous-section dans l'article [Wad07], dans lequel Philip Wadler a formalisé les résultats de Jean-Yves Girard, John Reynolds, Daniel Leivant et Jean-Louis Krivine dans $\mathcal{F}^2$.

4.2.31 Définition (Prouvablement inductif)
Supposons :

$$\Gamma \vdash_{\mathcal{P}^2} t : \overbrace{\mathsf{nat} \to \cdots \to \mathsf{nat}}^{n \text{ fois}} \to \mathsf{nat}$$

On dit alors de t qu'il est prouvablement inductif dans un contexte Γ si

$$\Gamma \vdash_{\mathcal{P}^2} \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}.(f\, x_1 \cdots x_n) \in \mathbb{N}$$

On dira qu'un terme est prouvablement inductif s'il est prouvablement inductif dans le contexte vide.

Afin de prouver que tous les termes clos de $\mathcal{P}$ sont prouvablement inductifs dans $\mathcal{P}^2$, nous avons besoin d'introduire un "avant-goût" de la théorie de la paramétricité (développée dans le prochain chapitre). Nous noterons donc $t_1 \sim_{\mathsf{nat}} t_2$ le terme de $\mathcal{P}^2$ suivant :

$$t_1 \sim_{\mathsf{nat}} t_2 = t_2 \Vdash t_1 \Vdash \lceil \mathsf{nat} \rceil$$

Dans $\mathcal{F}^2$, ce dernier se déplie en :

$$\begin{aligned}
t \sim_{\mathsf{nat}} t' &= t' \Vdash t \Vdash \lceil \mathsf{nat} \rceil \\
&= t' \Vdash t \Vdash \lceil \forall \alpha : \star.(\alpha \to \alpha) \to \alpha \to \alpha \rceil \\
&= t' \Vdash t \Vdash \forall X : \lceil \star \rceil.(X \to X) \to X \to X \\
&= t' \Vdash \forall \gamma : \star, X : \gamma \to \lceil \star \rceil, f : \gamma \to \gamma. \\
&\qquad (\forall x : \gamma.X\, x \to X\, (f\, x)) \to \forall z : \gamma.X\, z \to X\, (t\, \gamma\, f\, z) \\
&= \forall \gamma\, \gamma' : \star, X : \gamma \to \gamma' \to \lceil \star \rceil, f : \gamma \to \gamma, f' : \gamma' \to \gamma'. \\
&\qquad (\forall x : \gamma, x' : \gamma'.X\, x\, x' \to X\, (f\, x)\, (f'\, x')) \to \\
&\qquad\quad \forall z : \gamma, z' : \gamma'.X\, z\, z' \to X\, (t\, \gamma\, f\, z)(t'\, \gamma'\, f'\, z')
\end{aligned}$$

4.2.32 Lemme
Les objets en relation paramétrique sont extensionnellement égaux :

$$\vdash_{\mathcal{F}^2} \forall n\, m : \mathsf{nat}.n \sim_{\mathsf{nat}} m \quad \to \quad n \overset{ext}{=}_{\mathsf{nat}} m$$

Démonstration On peut utiliser l'hypothèse $n \sim_{\mathsf{nat}} m$ en prenant le cas particulier où $\gamma' = \gamma$, $X = \lambda x\, x' : \gamma.x =_\gamma x'$ et où $f' = f$. On obtient alors

$$\begin{aligned}
&\forall \gamma : \star, f : \gamma \to \gamma.(\forall x : \gamma, x' : \gamma.x =_\gamma x' \to (f\, x) =_\gamma (f\, x')) \to \\
&\quad \forall z : \gamma, z' : \gamma.z =_\gamma z' \to (n\, \gamma\, f\, z) =_\gamma (m\, \gamma\, f\, z')
\end{aligned}$$

Ce qui est bien équivalent à :

$$\forall \gamma : \star, f : \gamma \to \gamma, z : \gamma.(n\, \gamma\, f\, z) =_\gamma (m\, \gamma\, f\, z)$$

c'est-à-dire $n \overset{\text{ext}}{=}_{\mathsf{nat}} m$. ☺

4.2.33 Lemme

$$\vdash_{\mathcal{F}^2} \forall n : \mathsf{nat}.n \sim_{\mathsf{nat}} n \quad \rightarrow \quad n \overset{ext}{=}_{\mathsf{nat}} n\,\mathsf{nat}\,\mathsf{S}\,\overline{0}$$

Démonstration Supposons $n \sim_{\mathsf{nat}} n$ (H) et soit γ un type, f de type $\gamma \to \gamma$ et z de type γ. La preuve se construit en utilisant l'hypothèse (H) dans le cas particulier où $\gamma' = \mathsf{nat}$, $X = \lambda x : \gamma, n : \mathsf{nat}.x =_\gamma n\,\gamma\,f\,z$, $f' = \mathsf{S}$ et $z' = \overline{0}$. Les deux prémisses à prouver pour utiliser (H) sont alors :
- $\forall x : \gamma, x' : \mathsf{nat}.x =_\gamma (x'\,\gamma\,f\,z) \to (f\,x) =_\gamma (\mathsf{S}\,x'\,\gamma\,f\,z)$
- $z =_\alpha \overline{0}\,\gamma\,f\,z$

Or par réduction de S et $\overline{0}$, on obtient les formules suivantes :
- $\forall x : \gamma, x' : \mathsf{nat}.x =_\gamma (x'\,\gamma\,f\,z) \to (f\,x) =_\gamma (f\,(x'\,\gamma\,f\,z))$
- $z =_\alpha z$

Ces deux prémisses sont des tautologies, ce qui nous permet de conclure :

$$(n\,\gamma\,f\,z) =_\gamma (n\,\mathsf{nat}\,\mathsf{S}\,\overline{0}\,\gamma\,f\,z)$$

On a donc bien montré $n \overset{\text{ext}}{=}_{\mathsf{nat}} n\,\mathsf{nat}\,\mathsf{S}\,\overline{0}$. ☺

4.2.34 Lemme (Paramétrique implique inductif)
Si $\Gamma \vdash \mathsf{ext\_nat}$, *alors* $\Gamma \vdash \forall n : \mathsf{nat}.n \sim_{\mathsf{nat}} n \to n \in \mathbb{N}$.

Démonstration On remarque tout d'abord que $n \sim_{\mathsf{nat}} n$ implique $n \Vdash \lceil \mathsf{nat} \rceil$: il suffit pour cela d'instancier toutes les variables dupliquées x et x' dans $n \sim_{\mathsf{nat}} n$ par x, on obtient alors bien $n \Vdash \lceil \mathsf{nat} \rceil$. Or $n \Vdash \lceil \mathsf{nat} \rceil$ se déplie en :

$$\forall \gamma : \star, X : \gamma \to \lceil \star \rceil, f : \gamma \to \gamma.(\forall x : \gamma.X\,x \to X\,(f\,x)) \to \forall z : \gamma.X\,z \to X\,(n\,\gamma\,f\,z)$$

En prenant le cas particulier, $\gamma = \mathsf{nat}$, $X = \lambda x : \mathsf{nat}.x \in \mathbb{N}$, $f = \mathsf{S}$ et $z = \overline{0}$ on obtient :

$$(\forall x : \mathsf{nat}.x \in \mathbb{N} \to (\mathsf{S}\,x) \in \mathbb{N}) \to \overline{0} \in \mathbb{N} \to (n\,\mathsf{nat}\,\mathsf{S}\,\overline{0}) \in \mathbb{N}$$

Or nous avons déjà vu que ces deux prémisses sont prouvables (par $\mathsf{intro\text{-}succ}_{\mathbb{N}}[]$ et $\mathsf{intro\text{-}zero}_{\mathbb{N}}$). On en déduit donc bien $(n\,\mathsf{nat}\,\mathsf{S}\,\overline{0}) \in \mathbb{N}$. Or, d'après le lemme précédent, on a $n \overset{\text{ext}}{=}_{\mathsf{nat}} n\,\mathsf{nat}\,\mathsf{S}\,\overline{0}$, on en déduit bien de $\mathsf{ext\_nat}$ que $n \in \mathbb{N}$. ☺

Réciproquement,

4.2.35 Lemme (Inductif implique paramétrique)

$$\vdash_{\mathcal{F}^2} \forall n \in \mathbb{N}.n \sim_{\mathsf{nat}} n$$

Démonstration On utilise le principe d'induction. Il suffit donc de montrer que $\overline{0} \sim_{\mathsf{nat}} \overline{0}$ (1) et que $\forall x : \mathsf{nat}.x \sim_{\mathsf{nat}} x \to \mathsf{S}\,x \sim_{\mathsf{nat}} \mathsf{S}\,x$ (2). On peut pour cela utiliser le théorème 4.1.8. En effet, on a $\vdash_{\mathcal{F}} \overline{0} : \mathsf{nat}$ et $\vdash_{\mathcal{F}} \mathsf{S} : \mathsf{nat} \to \mathsf{nat}$. On en déduit donc que $\vdash_{\mathcal{F}^2} \lceil \overline{0} \rceil : \lceil \mathsf{nat} \rceil$ et que $\vdash_{\mathcal{F}^2} \lceil \mathsf{S} \rceil : \lceil \mathsf{nat} \rceil \to \lceil \mathsf{nat} \rceil$. En utilisant le théorème 4.1.8, on obtient :

$$\vdash_{\mathcal{F}^2} \|\lceil \overline{0} \rceil\| : \lfloor \lceil \overline{0} \rceil \rfloor \Vdash \lceil \mathsf{nat} \rceil$$

et

$$\vdash_{\mathcal{F}^2} \|\lceil \mathsf{S} \rceil\| : \lfloor \lceil \mathsf{S} \rceil \rfloor \Vdash \lceil \mathsf{nat} \rceil \to \lceil \mathsf{nat} \rceil$$

En appliquant ce même théorème une seconde fois, on obtient que

$$\vdash_{\mathcal{F}^2} \|\|\lceil \overline{0} \rceil\|\| : \lfloor \|\lceil \overline{0} \rceil\| \rfloor \Vdash \lfloor \lceil \overline{0} \rceil \rfloor \Vdash \lceil \mathsf{nat} \rceil$$

et

$$\vdash_{\mathcal{F}^2} \|\|\lceil \mathsf{S} \rceil\|\| : \lfloor \|\lceil \mathsf{S} \rceil\| \rfloor \Vdash \lfloor \lceil \mathsf{S} \rceil \rfloor \Vdash \lceil \mathsf{nat} \rceil \to \lceil \mathsf{nat} \rceil$$

Or d'après le lemme 4.1.3, on sait que :

- $\lfloor \|\lceil \overline{0} \rceil\| \rfloor$, $\lfloor \lceil \overline{0} \rceil \rfloor$ sont tous deux α-convertibles en $\overline{0}$,
- $\lfloor \|\lceil \mathsf{S} \rceil\| \rfloor$, $\lfloor \lceil \mathsf{S} \rceil \rfloor$ sont tous deux α-convertibles en S.

On en déduit alors

$$\vdash_{\mathcal{F}^2} \overline{0} \Vdash \overline{0} \Vdash \lceil \mathsf{nat} \rceil$$

c'est-à-dire (1) et

$$\vdash_{\mathcal{F}^2} \mathsf{S} \Vdash \mathsf{S} \Vdash \lceil \mathsf{nat} \rceil \to \lceil \mathsf{nat} \rceil$$

Ce dernier se déplie en :

$$\vdash_{\mathcal{F}^2} \forall x\, x' : \mathsf{nat}.x \sim_{\mathsf{nat}} x' \to \mathsf{S}\, x \sim_{\mathsf{nat}} \mathsf{S}\, x'$$

Et en prenant le cas particulier où $x' = x$, on en déduit bien (2). ☺

4.2.36 Lemme
Supposons $\mathcal{F}^2 \subseteq \mathcal{P}^2$. Si

$$\vdash_{\mathcal{P}^2} t : \overbrace{\mathsf{nat} \to \cdots \to \mathsf{nat}}^{n \text{ fois}} \to \mathsf{nat}$$

alors t est prouvablement inductif dans $\mathcal{P}^2$ sous le contexte ext_nat.

Démonstration Si $\vdash_{\mathcal{P}^2} t : \overbrace{\mathsf{nat} \to \cdots \to \mathsf{nat}}^{n \text{ fois}} \to \mathsf{nat}$ alors d'après le lemme 3.4.2 on a $\vdash_{\mathcal{P}} t : \mathsf{nat} \to \cdots \to \mathsf{nat} \to \mathsf{nat}$. On peut alors appliquer la fonction de soulèvement pour obtenir :

$$\vdash_{\mathcal{P}^2} \lceil t \rceil : \lceil \mathsf{nat} \rceil \to \cdots \to \lceil \mathsf{nat} \rceil \to \lceil \mathsf{nat} \rceil$$

puis le théorème d'adéquation 4.1.8 pour en déduire :

$$\vdash_{\mathcal{P}^2} \|\lceil t \rceil\| : \lfloor \lceil t \rceil \rfloor \Vdash (\lceil \mathsf{nat} \rceil \to \cdots \to \lceil \mathsf{nat} \rceil)$$

puis en l'appliquant une second fois :

$$\vdash_{\mathcal{P}^2} \|\|\lceil t \rceil\|\| : \lfloor \|\lceil t \rceil\| \rfloor \Vdash \lfloor \lceil t \rceil \rfloor \Vdash (\lceil \mathsf{nat} \rceil \to \cdots \to \lceil \mathsf{nat} \rceil \to \lceil \mathsf{nat} \rceil)$$

Or d'après le lemme 4.1.3, on obtient que $\lfloor \|\lceil t \rceil\| \rfloor$ et $\lfloor \lceil t \rceil \rfloor$ sont α-convertibles à t. On en déduit donc que $t \Vdash t \Vdash (\lceil \mathsf{nat} \rceil \to \cdots \to \lceil \mathsf{nat} \rceil)$. Si l'on déplie cette dernière formule on obtient :

$$\forall x_1\, x_1' : \mathsf{nat}.x_1 \sim_{\mathsf{nat}} x_1' \to \cdots \to \forall x_n\, x_n' : \mathsf{nat}.x_n \sim_{\mathsf{nat}} x_n' \to t\, x_1 \cdots x_n \sim_{\mathsf{nat}} t\, x_1' \cdots x_n'$$

En prenant les cas particuliers où $x_k' = x_k$, on obtient :

$$\forall x_1 : \mathsf{nat}.x_1 \sim_{\mathsf{nat}} x_1 \to \cdots \to \forall x_n : \mathsf{nat}.x_n \sim_{\mathsf{nat}} x_n \to t\, x_1 \cdots x_n \sim_{\mathsf{nat}} t\, x_1 \cdots x_n$$

Puis d'après les lemmes 4.2.35 et 4.2.34, on en déduit bien :

$$\mathsf{ext\_nat} \vdash \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}.(t\, x_1 \cdots x_n) \in \mathbb{N}$$

4.2.5 Programmation par preuve

Les résultats précédents ont été utilisés par Jean-Louis Krivine, Michel Parigot et Daniel Leivant pour prouver le théorème de représentation ci-dessous.

On notera que dans la présentation de Krivine, la notion de modèle est introduite très tôt. Il construit un modèle de réalisabilité dans lequel les individus sont interprétés par des programmes (plus précisément des éléments d'une algèbre combinatoire) et les réalisateurs des axiomes sont exhibés à l'intérieur du modèle et non comme des objets décrits par la syntaxe. Cette approche est plus souple à deux égards :

- Elle permet d'étendre la syntaxe des programmes dans le modèle, ainsi il est possible d'y exhiber des réalisateurs qui ne peuvent pas être construits par extraction des preuves. C'est en suivant cette démarche que Jean-Louis Krivine [Kri09] est parvenu à étendre la théorie de la réalisabilité à la logique classique.
- Elle permet de ne considérer que des modèles qui satisfont $\forall x : \mathsf{nat}.x \in \mathbb{N}$ et donc l'extensionalité. Ce qui simplifie grandement les énoncés des théorèmes.

Nous n'avons pas eu vraiment la possibilité de suivre cette approche puisque nous ne disposons pas de notion générale de modèle pour nos systèmes. C'est pourquoi nous cherchons à repousser le plus loin possible l'utilisation des modèles. Nous verrons néanmoins que certaines hypothèses des théorèmes que nous présenterons nécessiteront l'emploi de techniques "sémantiques" pour être prouvées. Un exemple d'une telle hypothèse est la condition de non-dégénérescence ; elle affirme que si $t_1 =_{\mathsf{nat}} t_2$ est prouvable sous un contexte Γ pour deux programmes clos, alors les programmes sont convertibles $t_1 \equiv t_2$.

Dans notre cadre donc, le théorème de représentation de Krivine s'énonce ainsi :

4.2.37 Théorème (Krivine & Parigot [KP90] et Leivant [Lei83])
Soit $\mathcal{P}$ tel que $\mathcal{F}^2 \subseteq \mathcal{P}^2$.
Soit Γ un contexte uniformément auto-réalisé clos au second niveau et tel que $\Gamma \vdash_{\mathcal{P}^2} \mathsf{ext\_nat}$.
On a $\Gamma \vdash_{\mathcal{P}^2} \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}.(t\, x_1 \ldots x_n) \in \mathbb{N}$, si et seulement il existe un terme p tel que

$$\vdash_{\mathcal{P}} p : \mathsf{nat} \to \ldots \to \mathsf{nat} \quad \textit{et} \quad \Gamma \vdash_{\mathcal{P}^2} \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}.(t\, x_1 \ldots x_n) =_{\mathsf{nat}} (p\, x_1 \ldots x_n)$$

Démonstration
- $\Rightarrow$: Supposons $\Gamma \vdash_{\mathcal{P}^2} \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}.(t\, x_1 \ldots x_n) \in \mathbb{N}$ alors d'après le théorème 4.2.23, on a $\Gamma \vdash_{\mathcal{P}^2} p \Vdash \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}.(t\, x_1 \ldots x_n) \in \mathbb{N}$ pour un programme p tel $\vdash p : \mathsf{nat} \to \cdots \to \mathsf{nat}$. Enfin, en dépliant, puis en utilisant le lemme 4.2.27, on obtient que $\Gamma \vdash_{\mathcal{P}^2} \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}.(p\, x_1 \ldots x_n) =_{\mathsf{nat}} (t\, x_1 \ldots x_n)$.
- $\Leftarrow$: Supposons $\vdash_{\mathcal{P}} p : \mathsf{nat} \to \ldots \mathsf{nat}$ (1) et $\Gamma \vdash_{\mathcal{P}^2} \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}.(t\, x_1 \ldots x_n) =_{\mathsf{nat}} (p\, x_1 \ldots x_n)$ (2). D'après le lemme 4.2.36 appliqué à (1), on montre que p est prouvablement inductif, c'est-à-dire que :

  $$\Gamma \vdash_{\mathcal{P}^2} \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}.(p\, x_1 \ldots x_n) \in \mathbb{N}$$

  On en déduit bien à l'aide de (2) que :

  $$\Gamma \vdash_{\mathcal{P}^2} \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}.(t\, x_1 \ldots x_n) \in \mathbb{N}$$

☺

Ce théorème fournit un nouveau paradigme de programmation : la programmation par preuve. L'idée de cette méthodologie introduite par Daniel Leivant et Jean-Louis Krivine, consiste à demander au programmeur de spécifier son programme dans un contexte uniformément auto-réalisé. Ce dernier doit ensuite prouver que le programme qu'il a spécifié est prouvablement inductif. S'il y parvient, le théorème précédent assure que l'on pourra extraire de la preuve une implémentation concrète de son programme. Afin d'illustrer cette méthodologie, nous allons voir comment l'utiliser pour trouver une implémentation de l'addition dans le contexte suivant :

$$\begin{array}{rcl}\Gamma & = & f : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}, \\ & & \mathsf{Ax}_1 : \forall x : \mathsf{nat}.x =_{\mathsf{nat}} f\, x\, \overline{0}, \\ & & \mathsf{Ax}_2 : \forall x : \mathsf{nat}, y : \mathsf{nat}.\mathsf{S}\,(f\, x\, y) =_{\mathsf{nat}} f\, x\, (\mathsf{S}\, y)\end{array}$$

On peut alors trouver une preuve π telle que :

$$\Gamma \vdash_{\mathcal{F}^2} \pi : \forall x \in \mathbb{N}, y \in \mathbb{N}.(f\, x\, y) \in \mathbb{N}$$

Décrivons une preuve possible dans un style informel.
Supposons que l'on dispose de x et y de type nat vérifiant $x \in \mathbb{N}$ (H_x) et $y \in \mathbb{N}$ (H_y) et montrons que $(f\, x\, y) \in \mathbb{N}$. Soit donc X un prédicat de type $\mathsf{nat} \to \mathsf{prop}$, stable par successeur (H_f) et vérifié par $\overline{0}$ (H_z), montrons que $(f\, x\, y)$ satisfait $X : X\,(f\, x\, y)$. Pour cela on peut utiliser le principe d'induction H_y pour se ramener à démontrer :

- $\forall z : \mathsf{nat}.X\,(f\, x\, z) \to X\,(f\, x\,(\mathsf{S}\, z))$: Soit z de type nat et supposons $X\,(f\, x\, z)$ (H). On doit montrer $X\,(f\, x\,(\mathsf{S}\, z))$. Pour cela on peut utiliser l'axiome Ax_2 pour réécrire $(f\, x\,(\mathsf{S}\, z))$ en $(\mathsf{S}\,(f\, x\, z))$ puis utiliser la stabilité par successeur (H_f) et pour conclure avec (H).
- $X\,(f\, x\, \overline{0})$: On peut utiliser l'axiome Ax_1 pour réécrire $(f\, x\, \overline{0})$ en x. On se ramène donc à montrer $X\, x$ et pour cela on utilise le principe d'induction H_x avec H_f et H_z.

Le terme de preuve sous-jacent à cette preuve informelle est le suivant :

$$\begin{array}{rcl}\pi & = & \lambda x : \mathsf{nat}, H_x : x \in \mathbb{N}, y : \mathsf{nat}, H_y : y \in \mathbb{N}, X\ : \mathsf{nat} \to \lceil\star\rceil, \\ & & \quad H_f : \forall z : \mathsf{nat}.X\, z \to X\,(\mathsf{S}\, z), H_z : X\, \overline{0}. \\ & & \quad H_y(\lambda z : \mathsf{nat}.X\,(f\, x\, z)) \\ & & \qquad (\lambda z : \mathsf{nat}, H : X\,(f\, x\, z).\mathsf{Ax}_2\, x\, z\, X\,(H_f\,(f\, x\, z)\, H)) \\ & & \qquad (\mathsf{Ax}_1\, x\, X\,(H_x\, X\, H_f\, H_z))\end{array}$$

On peut calculer le programme extrait de cette preuve en calculant la projection de π :

$$\begin{array}{rcl}\lfloor\pi\rfloor & = & \lambda H_x : \mathsf{nat}, H_y : \mathsf{nat}, X : \star, \\ & & \quad H_f : X \to X, H_z : X. \\ & & \quad H_y X \\ & & \qquad (\lambda H : X.\mathsf{Ax}_2\, X\,(H_f\, H)) \\ & & \qquad (\mathsf{Ax}_1\, X\,(H_x\, X\, H_f\, H_z))\end{array}$$

La projection contient deux occurrences libres : Ax_1 et Ax_2 que l'on peut instancier par des réalisateurs canoniques (ici, comme il s'agit d'égalités, les réalisateurs canoniques sont des identités). C'est le rôle du théorème 4.2.22 qui construit le programme $\lfloor\pi\rfloor'$ suivant :

$$\begin{aligned}
\lfloor\pi\rfloor' &= \lfloor\pi\rfloor[\lambda\alpha:\star, x:\alpha.x/\mathsf{Ax}_1, \lambda\alpha:\star, x:\alpha.x/\mathsf{Ax}_2] \\
&\trianglerighteq \lambda H_x:\mathsf{nat}, H_y:\mathsf{nat}, X:\star, \\
&\qquad H_f : X \to X, H_z : X. \\
&\qquad H_y\, X \\
&\qquad\quad (\lambda H : X. H_f\, H) \\
&\qquad\quad (H_x\, X\, H_f\, H_z) \\
&= \lambda x\, y : \mathsf{nat}, \gamma:\star, f:\gamma\to\gamma, z:\alpha. \\
&\qquad y\,\gamma\,(\lambda z:\gamma. f\, z)\,(x\,\gamma\, f\, z)
\end{aligned}$$

La dernière étape est simplement une α-conversion qui nous permet de reconnaître un terme extensionnellement égal au terme plus vu page 23.

L'intérêt de cette méthode se situe dans le fait que l'étape d'extraction du programme et d'instanciation des axiomes par leur réalisateur canonique peut être entièrement automatisée. On voit ainsi que le programmeur dispose d'un langage expressif (les contextes auto-réalisés) pour spécifier ses programmes et que, une fois le programme f spécifié par un contexte Γ, il ne lui reste qu'à trouver une preuve que f est prouvablement inductif. Quelle que soit la preuve qu'il construit, il aura la garantie que le programme qui en est extrait est prouvablement égal –dans le contexte Γ– au programme f spécifié.

Lorsque cette méthodologie a été découverte, elle a suscité quelques espoirs de pouvoir découvrir de nouveaux algorithmes par des procédés automatiques. Hélas, la recherche de preuve en logique du second ordre est particulièrement difficile (en particulier l'instanciation des quantificateurs du second ordre est difficilement automatisable). Ainsi, il n'est pas aisé de prouver la formule requise sans avoir en tête une implémentation concrète de la fonction.

Un autre désavantage majeur dans l'utilisation de cette méthode est de devoir s'assurer que le contexte a un modèle non trivial (c'est-à-dire ne satisfiant pas $\bot_{\mathrm{w}}$) dans lequel on pourra interpréter raisonnablement la conformité aux spécifications. Or, montrer qu'il existe un tel modèle revient en général à montrer que les symboles de fonctions introduits dans le contexte sont implémentables. On obtient donc une implémentation correcte de la fonction spécifiée à condition que les spécifications soient implémentables : à l'impossible nul n'est tenu.

Une façon de pallier ce problème est d'imposer des restrictions sur les axiomes autorisés dans le contexte. On peut par exemple demander à ce que le contexte ne contienne que des formules dérivables dans **HA** ainsi que des symboles de fonctions frais et des formules égalitaires qui décrivent un système de réécriture implémentant des fonctions primitives récursives. Par exemple, les deux équations – lues de droite à gauche – du contexte Γ donné plus haut décrivent bien un tel système de réécriture. On a alors la garantie que les symboles de fonctions sont interprétables par ces fonctions primitives récursives et donc qu'il existe un modèle non trivial.

Extension de la méthodologie à $\mathcal{F}_\omega$ et à CC**.** Le théorème précédent est énoncé de façon suffisamment générale pour être appliqué aux systèmes engendrés par les systèmes du λ-cube qui contiennent

$\mathcal{F} : (\mathcal{F}_\omega)^2$, $(\lambda\Pi^2)^2$ (la notation n'est ici pas très heureuse) et CC^2. On peut également l'appliquer à la famille de système $(\mathcal{F}_n)^2$ (voir page 189).

On peut par exemple les comprendre comme des sous-systèmes du calcul des constructions avec `Set` et `Prop` imprédicatifs (CIC^- sans inductifs ni univers) à travers le morphisme suivant :

$$\begin{array}{rcl} \lceil\star\rceil & \mapsto & \texttt{Prop} \\ \star & \mapsto & \texttt{Set} \\ \lceil\Box\rceil & \mapsto & \texttt{Type} \\ \Box & \mapsto & \texttt{Type} \end{array}$$

Nous avons vu dans ce chapitre que l'on pouvait comprendre le système $\mathcal{F}^2$ comme une version de l'arithmétique du second-ordre avec des individus typés. Selon la même analogie le système $(\mathcal{F}_n)^2$ peut être compris comme l'arithmétique d'ordre $n+2$ avec des individus typés. Et le système $(\mathcal{F}_\omega)^2$ comme de l'arithmétique d'ordre supérieur.

En effet, le système $(\mathcal{F}_\omega)^2$ est un peu plus expressif que le système $\mathcal{F}^2$ on a :

$$\begin{array}{rcl} \mathcal{R}_{(\mathcal{F}_\omega)^2} & = & \mathcal{R}_{\mathcal{F}^2} \cup \mathcal{R}_{\mathcal{F}_\omega} \\ & \cup & \{\ (\lceil\Box\rceil, \lceil\Box\rceil, \lceil\Box\rceil), (\Box, \lceil\Box\rceil, \lceil\Box\rceil)\ \} \end{array}$$

On dispose donc de deux nouvelles règles pour fabriquer les formules :

– La règle $(\lceil\Box\rceil, \lceil\Box\rceil, \lceil\Box\rceil)$ permet de dériver par exemple :

$$\vdash_{(\mathcal{F}_\omega)^2} \lceil\star\rceil \to \lceil\star\rceil : \lceil\Box\rceil$$

Ce qui permet en outre de typer les "constructeurs de formules" comme :

$$\lambda X\, Y : \lceil\star\rceil . X \wedge Y : \lceil\star\rceil \to \lceil\star\rceil \to \lceil\star\rceil$$

De façon beaucoup plus intéressante pour l'expressivité du système, cette règle permet de former des prédicats sur les prédicats comme par exemple :

$$(\mathsf{nat} \to \lceil\star\rceil) \to \lceil\star\rceil : \lceil\Box\rceil$$

Qui peuvent s'interpréter comme des "ensembles d'ensembles d'entiers" ; on voit ici que l'on peut alors se servir de ce système pour faire de la logique d'ordre supérieur.

– La règle $(\Box, \lceil\Box\rceil, \lceil\Box\rceil)$ permet de dériver :

$$\vdash_{(\mathcal{F}_\omega)^2} \star \to \lceil\star\rceil : \lceil\Box\rceil$$

Cette règle permet de typer des prédicats sur les types. Ainsi dans $(\mathcal{F}_\omega)^2$ on peut définir "l'égalité de Leibniz" entre deux types τ et σ :

$$\forall X : \star \to \lceil\star\rceil . X\,\tau \to X\,\sigma$$

Le système $(\lambda\Pi^2)^2$ (resp. CC^2) rajoute à $\mathcal{F}$ (resp. $\mathcal{F}_\omega$) uniquement la règle $(\lceil\star\rceil, \lceil\Box\rceil, \lceil\Box\rceil)$ car la règle $(\star, \lceil\Box\rceil, \lceil\Box\rceil) \in \mathcal{R}_{\mathcal{F}^2}$. Cette règle permet de former des prédicats sur les preuves. On peut donc s'en servir pour encoder l'égalité de Leibniz entre deux preuves H_1 et H_2 d'une même formule P :

$$\forall X : P \to \lceil\star\rceil.X\ H_1 \to X\ H_2$$

On peut alors énoncer le principe d'équivalence des preuves qui affirme l'indistinguabilité des preuves d'un même énoncé :

$$\forall X : \lceil\star\rceil, h_1\ h_2 : X.\forall X : P \to \lceil\star\rceil.X\ h_1 \to X\ h_2$$

4.2.6 Fonctions prouvablement totales

Dans système $\mathcal{F}$ les habitants du type nat ne sont pas tous convertibles en un entier de Church. En effet, le terme $\lambda\alpha : \star, x : \alpha \to \alpha.x$ n'est pas convertible en $\overline{1}$, bien qu'il lui soit extensionnellement égal. Il y a deux moyens habituels de contourner ce problème :

- considérer les termes modulo "η-conversion" : on rajoute la règle de réduction $\lambda x : A.M\ x \twoheadrightarrow_\eta M$ qui est une règle de calcul compatible avec l'extensionalité des fonctions, au sens où où elle ne permet pas de distinguer deux fonctions extensionnellement égales. Grâce à cette règle, on a $\overline{1} \equiv_\eta \lambda\alpha : \star, x : \alpha \to \alpha.x$.
- utiliser le type $\forall\alpha : \star.\alpha \to (\alpha \to \alpha) \to \alpha$ pour représenter les entiers. En effet, on peut montrer que tous les habitants clos de ce type sont de la forme :

$$\forall\alpha : \star, x : \alpha, f : \alpha \to \alpha.f\ (f\ \cdots\ x)$$

Dans ce travail, nous allons utiliser une troisième solution, car la première requiert d'étendre tout le cadre des PTS pour prendre en compte l'η-conversion et la seconde nous paraît *ad-hoc* au type des entiers.

Nous allons introduire un *opérateur de relecture des données* qui nous assurera que l'objet inductif considéré est bâti avec ses constructeurs. Pour le type des entiers (qui est le seul que l'on prend en compte dans ce travail), cet opérateur est donné dans un système muni de notations pour l'arithmétique par le terme suivant :

$$\mathsf{read} = \lambda n : \mathsf{nat}.\mathsf{rec}[\mathsf{nat}]\ \mathsf{S}\ \overline{0}\ n$$

On montre alors facilement que cet opérateur ne change pas la valeur des entiers (du point de vue du système) :

4.2.38 Lemme
Si $\mathcal{P}^2 \models$ NOTATION,

$$\vdash_{\mathcal{P}^2} \forall n \in \mathbb{N}.\mathsf{read}\ n =_{\mathsf{nat}} n$$

Démonstration En utilisant le principe d'induction, on se ramène à prouver que :

- $(\mathsf{read}\ \overline{0}) =_{\mathsf{nat}} \overline{0}$: ce qui est une application directe de $\mathsf{rec\text{-}zero}[\mathsf{nat}]$.
- $(\mathsf{read}\ n) =_{\mathsf{nat}} n \to (\mathsf{read}\ (\mathsf{S}n)) =_{\mathsf{nat}} (\mathsf{S}\ n)$, or on a $\mathsf{read}\ (\mathsf{S}\ n) =_{\mathsf{nat}} \mathsf{S}\ (\mathsf{read}\ n)$ d'après $\mathsf{rec\text{-}succ}[\mathsf{nat}]$, on se ramène donc à démontrer $\mathsf{S}\ n =_{\mathsf{nat}} \mathsf{S}\ n$. ☺

Cet opérateur de lecture permet de décider si deux termes représentent le même entier.

4.2.39 Définition (Equi-représentabilité)
Soit $\mathcal{P}^2 \models$ NOTATION. On dit de deux termes M et N qu'ils sont équireprésentés, ce que l'on notera $M \cong N$, si on a :

$$(\mathsf{read}\, M) \equiv (\mathsf{read}\, N)$$

où $\mathsf{read} = \lambda n : \mathsf{nat}.\mathsf{rec}[\mathsf{nat}]\, \mathsf{S}\, \overline{0}\, n$.

Ainsi, dans système $\mathcal{F}$, on a bien $\overline{1} \cong \lambda\alpha : \star, x : \alpha \to \alpha.x$ car :

$$\mathsf{rec}[\mathsf{nat}]\, \mathsf{S}\, \overline{0}\, n\, (\lambda\alpha : \star, x : \alpha \to \alpha.x) \trianglerighteq (\lambda\alpha : \star, x : \alpha \to \alpha.x)\, \mathsf{nat}\, \mathsf{S}\, \overline{0} \trianglerighteq \mathsf{S}\, \overline{0}$$

Clairement,

4.2.40 Lemme
La relation $\cong$ est une relation d'équivalence contenant $\equiv$.

En pratique, les systèmes considérés vérifient les propriétés suivantes :

4.2.41 Définition (Système injectif)
Soit $\mathcal{P} \models$ NOTATION.
On dira de $\mathcal{P}$ qu'il est injectif si

$$\overline{n} \cong \overline{m} \text{ implique } n = m \text{ pour tous entiers } n \text{ et } m$$

On notera $\mathcal{P} \models$ INJECTIVE.

Réciproquement :

4.2.42 Définition (Système surjectif, système bijectif)
Soit $\mathcal{P} \models$ NOTATION.
On dira de $\mathcal{P}$ qu'il est surjectif si $\vdash t : \mathsf{nat}$ implique qu'il existe un entier k tel que $t \cong \overline{k}$.
On notera $\mathcal{P} \models$ SURJECTIVE.
De plus, on dira d'un système qu'il est bijectif si $\mathcal{P} \models$ INJECTIVE $\wedge$ SURJECTIVE ce que l'on notera BIJECTIVE.

On montre facilement que :

4.2.43 Lemme

$$\mathcal{F}^2 \models \text{BIJECTIVE}$$

On notera que c'est essentiellement une propriété des programmes, c'est-à-dire de $\mathcal{F}$ ici.
On dit d'un contexte qu'il est non-dégénéré : si deux termes clos de type nat sont prouvablement égaux dans ce contexte, alors ils représentent le même entier. Plus précisément :

4.2.44 Définition (Contexte non-dégénéré, Système non-dégénéré)
Supposons que $\mathcal{P} \models$ NOTATION.
On dira d'un contexte Γ qu'il est non-dégénéré, s'il vérifie pour tous termes clos t_1 et t_2 tels que $\vdash t_1 : \mathsf{nat}$ et $\vdash t_2 : \mathsf{nat}$ on a :

$$\text{Si } \Gamma \vdash_{\mathcal{P}} t_1 =_{\mathsf{nat}} t_2, \text{ alors } t_1 \cong t_2.$$

Enfin, on dira qu'un système muni de notations pour l'arithmétique est non-dégénéré si le contexte vide est non-dégénéré.
On notera $\mathcal{P} \models$ NONDEGENERATE.

Clairement, si Γ est un contexte non-dégénéré d'un système $\mathcal{P}$, alors $\mathcal{P} \models$ NONDEGENERATE, mais comme nous allons le voir, la réciproque est fausse.

Dans les systèmes injectifs, si $\Gamma \vdash \bot_{\mathrm{w}}$, alors Γ est dégénéré (puisque $\Gamma \vdash \overline{0} =_{\mathsf{nat}} \overline{1}$ et $0 \neq 1$).

En pratique, le meilleur moyen pour montrer qu'un contexte est non-dégénéré est de construire un modèle pour ce contexte qui interprète les termes de premier niveau par des λ-termes modulo conversion. C'est donc bien dans l'hypothèse de non-dégénérescence que devra se glisser la sémantique lorsque l'on appliquera les résultats de cette section à des systèmes particuliers.

4.2.45 Lemme
Dans un système non-dégénéré et bijectif, on a :

$$\mathsf{read}\,\overline{n} \equiv \overline{n}$$

On a donc $M \cong \overline{n}$ si et seulement si $\mathsf{read}\,M \equiv \overline{n}$.

Démonstration Par surjectivité, on sait qu'il existe un entier k tel que :

$$\mathsf{read}\,\overline{n} \equiv \overline{k}$$

Par réflexivité, on en déduit que

$$\vdash \mathsf{read}\,\overline{n} =_{\mathsf{nat}} \overline{k}$$

et d'après le lemme 4.2.38, on obtient :

$$\vdash \overline{n} =_{\mathsf{nat}} \overline{k}$$

la non-dégénérescence donne :

$$\overline{n} \cong \overline{k}$$

et l'injectivité nous permet de conclure que $n = k$. ☺

Si $\mathcal{P}^2 \models$ NOTATION alors le terme de $\mathcal{P}$ suivant implémente le test d'égalité des entiers :

$$\begin{aligned} \mathsf{comp} \quad = \quad & \mathsf{rec}[\mathsf{nat} \to \mathsf{nat}] \\ & (\lambda f : \ \mathsf{nat} \to \mathsf{nat}, \_ : \mathsf{nat.rec}[\mathsf{nat}]\,(\lambda \_ : \mathsf{nat}.f)\,\overline{0}) \\ & (\mathsf{rec}[\mathsf{nat}]\,(\lambda \_\_ : \mathsf{nat}.\overline{0})\,\overline{1}) \end{aligned}$$

où _ désigne un nom de variable quelconque qui n'apparaît pas dans le corps de l'abstraction.

On peut montrer :

4.2.46 Lemme
Si $\models$ NOTATION, *alors les formules suivantes sont prouvables dans un contexte vide :*

1. $\mathsf{comp}\,\overline{0}\,\overline{0} =_{\mathsf{nat}} \overline{1}$

2. $\forall n : \mathsf{nat.comp}\,\overline{0}\,(\mathsf{S}\,n) =_{\mathsf{nat}} \overline{0}$
3. $\forall n : \mathsf{nat.comp}\,(\mathsf{S}\,n)\,\overline{0} =_{\mathsf{nat}} \overline{0}$
4. $\forall n\,m : \mathsf{nat.comp}\,(\mathsf{S}\,n)\,(\mathsf{S}\,m) =_{\mathsf{nat}} \mathsf{comp}\,n\,m$
5. $\forall n \in \mathbb{N}.\forall m \in \mathbb{N}.(\mathsf{comp}\,n\,m) =_{\mathsf{nat}} \overline{0} \vee (\mathsf{comp}\,n\,m) =_{\mathsf{nat}} \overline{1}$
6. $\forall n \in \mathbb{N}.\forall m \in \mathbb{N}.(\mathsf{comp}\,n\,m) \in \mathbb{N}$
7. $\forall n \in \mathbb{N}.\mathsf{comp}\,n\,n =_{\mathsf{nat}} \overline{1}$
8. $\forall n \in \mathbb{N}.\forall m \in \mathbb{N}.\mathsf{comp}\,n\,m =_{\mathsf{nat}} \overline{1} \Rightarrow n =_{\mathsf{nat}} m$

Démonstration 1,2,3,4. Les quatre premières formules se démontrent à l'aide des règles rec-succ[_], rec-zero[_] et les règles de réécriture intro$_=$[_], elim$_=$[_].

5. Immédiat par induction sur n puis m et en utilisant les quatre cas précédents.
6. Direct d'après le cas précédent et le fait que $\overline{0} \in \mathbb{N}$ et $\overline{1} \in \mathbb{N}$.
7. La cinquième formule utilise le principe d'induction pour se ramener à démontrer la formule 1 pour le cas de base et 4 pour l'hérédité.
8. La sixième formule utilise les deux principes d'induction :
   - Le cas $n = m = 0$ est traité par 1.
   - Le cas $n = 0$ et $m = \mathsf{S}\,m'$ et le cas $n = \mathsf{S}\,n'$ et $m = 0$ utilisent 2. et 3. pour déduire $\overline{0} =_{\mathsf{nat}} \overline{1}$. Or on montre facilement que $\forall k \in \mathbb{N}.\overline{0} =_{\mathsf{nat}} \overline{1} \to k =_{\mathsf{nat}} \overline{0}$ (pour cela il faut utiliser le principe d'induction ainsi que les règles de réécriture).
   - Enfin, le cas $n = \mathsf{S}\,n'$ et $m = \mathsf{S}\,m'$ est traité grâce à l'hypothèse d'induction et aux règles de réécriture. ☺

4.2.47 Lemme

- *Dans un système non-dégénéré et injectif on a pour tous entiers n et m :*

$$\mathsf{comp}\,\overline{n}\,\overline{m} \cong \overline{1} \text{ si et seulement si } n = m$$

- *Dans un système non-dégénéré et bijectif on a pour tous entiers n et m :*

$$\mathsf{read}\,(\mathsf{comp}\,\overline{n}\,\overline{m}) \equiv \overline{1} \text{ si et seulement si } n = m$$

Démonstration Le deuxième point est une conséquence directe du premier et du lemme 4.2.45. Montrons donc le premier point. Soit n et m deux entiers.

- $\Leftarrow$: Supposons $n = m$. On peut utiliser la preuve de la formule 7 du lemme précédent pour démontrer que

$$\vdash \mathsf{comp}\,\overline{n}\,\overline{n} =_{\mathsf{nat}} \overline{1}$$

  on peut alors utiliser la non-dégénérescence du système pour en déduire que

$$\mathsf{comp}\,\overline{n}\,\overline{n} \cong \overline{1}$$

- $\Rightarrow$: Supposons que :

$$\mathsf{comp}\,\overline{n}\,\overline{m} \cong \overline{1}$$

Or, par conversion et par réflexivité ($\mathsf{intro}_=[\_]$), on a

$$\vdash \mathsf{read}\,(\mathsf{comp}\,\overline{n}\,\overline{m}) =_{\mathsf{nat}} (\mathsf{read}\,\overline{1})$$

Or, on a déjà vu que $\vdash \overline{n} \in \mathbb{N}$, $\vdash \overline{m} \in \mathbb{N}$, $\overline{1} \in \mathbb{N}$ et donc d'après le lemme précédent on en déduit que

$$\vdash \mathsf{read}\,(\mathsf{comp}\,\overline{n}\,\overline{m}) =_{\mathsf{nat}} \mathsf{comp}\,\overline{n}\,\overline{m}$$

et

$$\vdash \mathsf{read}\,\overline{1} =_{\mathsf{nat}} \overline{1}$$

et donc que :

$$\vdash \mathsf{comp}\,\overline{n}\,\overline{m} =_{\mathsf{nat}} \overline{1}$$

On peut donc utiliser la preuve de la formule 8 du lemme précédent pour en déduire que $\vdash \overline{n} =_{\mathsf{nat}} \overline{m}$ puis la non-dégénérescence nous donne $\overline{n} \cong \overline{m}$ et l'injectivité $n = m$. ☺

4.2.48 Définition (Fonction représentable)
Soit $\mathcal{P}$ tel que $\mathcal{P}^2 \models$ NOTATION. Une fonction $f : \mathbb{N}^k \to \mathbb{N}$ est représentable dans $\mathcal{P}$ s'il existe un terme clos t_f tel que

$$\vdash_{\mathcal{P}} t_f : \overbrace{\mathsf{nat} \to \cdots \to \mathsf{nat}}^{k \text{ fois}} \to \mathsf{nat}$$

et qui vérifie pour tout k-uplet d'entiers $(n_1, \ldots, n_k) \in \mathbb{N}^k$:

$$\mathsf{read}\,(t_f\,\overline{n_1} \cdots \overline{n_k}) \equiv \overline{f(n_1, \ldots, n_k)}$$

4.2.49 Lemme (Les fonctions primitives récursives sont représentables)
Soit un système $\mathcal{P}$ tel que $\mathcal{P}^2 \models$ NOTATION $\wedge$ NONDEGENERATE $\wedge$ BIJECTIVE.
Si f est une fonction primitive récursive, alors f est représentable dans $\mathcal{P}$.

Démonstration Par induction sur la "structure des fonctions primitives récursives" :

- constante 0 : On prend $t_0 = \overline{0}$. La vérification est une trivialité, il faut vérifier que $\mathsf{read}\,t_0 \equiv \overline{0}$ (pour tout 0-uplet). On utilise pour cela lemme 4.2.45.
- successeur : Soit $s(n) = n + 1$ la fonction successeur. On prend $t_s = \mathsf{S}$ et on a bien :

$$\mathsf{read}\,(t_s\,\overline{n}) = \mathsf{read}\,(\mathsf{S}\,\overline{n}) = \mathsf{read}\,\overline{n+1} \equiv \overline{n+1} = \overline{s(n)}$$

  d'après le lemme 4.2.45 et la définition de la notation $\overline{\cdot}$.
- projections : Soit $\pi_k^m(n_1, \ldots, n_m) = n_k$ $(1 \leq k \leq m)$ une fonction de projection. On prend $t_{\pi_k^m} = \lambda x_1 : \mathsf{nat}, \ldots, x_m : \mathsf{nat}.x_k$, on a bien :

$$\mathsf{read}\,(t_{\pi_k^m}\,\overline{n_1} \cdots \overline{n_m}) \rhd \mathsf{read}\,\overline{n_k} \equiv \overline{n_k} = \overline{\pi_k^m(n_1, \ldots, n_m)}$$

– composition : Soit $f : \mathbb{N}^k \to \mathbb{N}$, $g_1 : \mathbb{N}^m \to \mathbb{N}$, ..., $g_k : \mathbb{N}^m \to \mathbb{N}$. Alors par hypothèse d'induction il existe t_f, t_{g_1},...,t_{g_k} représentant ces fonctions. Soit $h(n_1, \ldots, n_m) = f(g_1(n_1, \ldots, n_m), \ldots, g_k(n_1, \ldots, n_m))$ la composée de ces fonctions. On peut alors prendre

$$t_h = \lambda x_1 : \mathsf{nat}, \ldots, x_m : \mathsf{nat}.t_f\,(\mathsf{read}\,(t_{g_1}\,x_1\,\cdots\,x_m))\,\cdots\,(\mathsf{read}\,(t_{g_k}\,x_1\,\cdots\,x_m))$$

On a alors :

$$\begin{array}{rcl}
\mathsf{read}\,(t_h\,\overline{n_1}\,\cdots\,\overline{n_m}) & \trianglerighteq & \mathsf{read}(t_f\,(\mathsf{read}\,(t_{g_1}\,\overline{n_1}\,\cdots\,\overline{n_m}))\,\cdots\,(\mathsf{read}\,(t_{g_k}\,\overline{n_1}\,\cdots\,\overline{n_m}))) \\
& \equiv & \mathsf{read}\,(t_f\,\overline{g_1(n_1, \ldots, n_m)}\,\cdots\,\overline{g_k(n_1, \ldots, n_m)}) \\
& \equiv & \overline{f(g_1(n_1, \ldots, n_m), \ldots, g_k(n_1, \ldots, n_m))} \\
& = & \overline{h(n_1, \ldots, n_m)}
\end{array}$$

– récursion primitive : Soit $f : \mathbb{N}^k \to \mathbb{N}$ et $g : \mathbb{N}^{k+2} \to \mathbb{N}$ et soit h une solution des équations :

$$\begin{array}{rcl}
h(n_1, \ldots, n_k, 0) & = & f(n_1, \ldots, n_k) \\
h(n_1, \ldots, n_k, m+1) & = & g(n_1, \ldots, n_k, h(n_1, \ldots, n_k, m), m)
\end{array}$$

Par hypothèse d'induction, on obtient l'existence des représentants t_f et t_g. On construit le représentant de h de la façon suivante :

$$t_h = \lambda x_1 : \mathsf{nat}, \ldots, x_k : \mathsf{nat}.\mathsf{rec}[\mathsf{nat}]\,(\lambda r : \mathsf{nat}.t_g\,x_1\,\cdots\,x_k\,(\mathsf{read}\,r))\,(t_f\,x_1\,\cdots\,x_k)$$

Soient n_1, ..., n_k k entiers et m un entier. On a :

$$t_h\,\overline{n_1}\,\cdots\,\overline{n_k}\,\overline{m} \trianglerighteq \mathsf{rec}[\mathsf{nat}]\,(\lambda r : \mathsf{nat}.t_g\,\overline{n_1}\,\cdots\,\overline{n_k}\,(\mathsf{read}\,r))\,(t_f\,\overline{n_1}\,\cdots\,\overline{n_k})\,\overline{m}$$

On montre par induction sur m que t_h représente bien une solution des équations définissant h :
– $m = 0$, On doit montrer :

$$\mathsf{rec}[\mathsf{nat}]\,(\lambda r : \mathsf{nat}.t_g\,\overline{n_1}\,\cdots\,\overline{n_k}\,(\mathsf{read}\,r))\,(t_f\,\overline{n_1}\,\cdots\,\overline{n_k})\,\overline{0} \equiv \overline{f(n_1, \ldots, n_k)}$$

Or, on peut utiliser $\mathsf{rec\text{-}zero}[\mathsf{nat}]$ pour montrer :

$$\vdash \mathsf{rec}[\mathsf{nat}]\,(\lambda r : \mathsf{nat}.t_g\,\overline{n_1}\,\cdots\,\overline{n_k}\,(\mathsf{read}\,r))\,(t_f\,\overline{n_1}\,\cdots\,\overline{n_k})\,\overline{0} =_{\mathsf{nat}} t_f\,\overline{n_1}\,\cdots\,\overline{n_k}$$

Et donc comme on est dans un système non-dégénéré, on en déduit :

$$\vdash \mathsf{read}\,\mathsf{rec}[\mathsf{nat}]\,(\lambda r : \mathsf{nat}.t_g\,\overline{n_1}\,\cdots\,\overline{n_k}\,(\mathsf{read}\,r))\,(t_f\,\overline{n_1}\,\cdots\,\overline{n_k})\,\overline{0} \equiv \mathsf{read}\,t_f\,\overline{n_1}\,\cdots\,\overline{n_k}$$

Puis, comme t_f représente f on montre bien :

$$\mathsf{read}\,t_f\,\overline{n_1}\,\cdots\,\overline{n_k} \equiv \overline{f(n_1, \ldots, n_k)}$$

On en conclut donc bien que :

$$\mathsf{read}\,t_h\,\overline{n_1}\,\cdots\,\overline{n_k}\,\overline{0} \equiv \overline{h(n_1, \ldots, n_k, 0)}$$

– $m = m' + 1$: Par hypothèse d'induction on a :

$$\mathsf{read}\,(\mathsf{rec}[\mathsf{nat}]\,(\lambda r : \mathsf{nat}.t_g\,\overline{n_1}\,\cdots\,\overline{n_k}\,(\mathsf{read}\,r))\,(t_f\,\overline{n_1}\,\cdots\,\overline{n_k})\,\overline{m'}) \equiv \overline{h(n_1,\ldots,n_k,m')}$$

et on doit montrer que

$$\mathsf{read}\,(\mathsf{rec}[\mathsf{nat}]\,(\lambda r : \mathsf{nat}.t_g\,\overline{n_1}\,\cdots\,\overline{n_k}\,(\mathsf{read}\,r))\,(t_f\,\overline{n_1}\,\cdots\,\overline{n_k})\,\overline{m'+1}) \equiv \overline{g(n_1,\ldots,n_k,h(n_1,\ldots,n_k,m'),m')}$$

Or, en utilisant $\mathsf{rec\text{-}succ}[\mathsf{nat}]$, on montre bien (on a $\overline{m'+1} = \mathsf{S}\,(\overline{m'})$) :

$$\begin{aligned}\vdash\ & \mathsf{rec}[\mathsf{nat}]\,(\lambda r : \mathsf{nat}.t_g\,\overline{n_1}\,\cdots\,\overline{n_k}\,(\mathsf{read}\,r))\,(t_f\,\overline{n_1}\,\cdots\,\overline{n_k})\,\overline{m'+1}\\ & =_{\mathsf{nat}}\ t_g\,\overline{n_1}\,\cdots\,\overline{n_k}\,(\mathsf{read}\,(\mathsf{rec}[\mathsf{nat}]\,(\lambda r : \mathsf{nat}.t_g\,\overline{n_1}\,\cdots\,\overline{n_k}\,(\mathsf{read}\,r))\,(t_f\,\overline{n_1}\,\cdots\,\overline{n_k})\,\overline{m}))\,\overline{m'})\end{aligned}$$

Or par conversion on en déduit :

$$\vdash \mathsf{rec}[\mathsf{nat}]\,(t_g\,\overline{n_1}\,\cdots\,\overline{n_k})\,(t_f\,\overline{n_1}\,\cdots\,\overline{n_k})\,\overline{m'+1} =_{\mathsf{nat}} t_g\,\overline{n_1}\,\cdots\,\overline{n_k}\,\overline{h(n_1,\ldots,n_k,m')}\,\overline{m'}$$

Puis en utilisant la non-dégénérescence du système on obtient :

$$\mathsf{rec}[\mathsf{nat}]\,(t_g\,\overline{n_1}\,\cdots\,\overline{n_k})\,(t_f\,\overline{n_1}\,\cdots\,\overline{n_k})\,\overline{m'+1} \cong t_g\,\overline{n_1}\,\cdots\,\overline{n_k}\,\overline{h(n_1,\ldots,n_k,m')}\,\overline{m'}$$

En utilisant le fait que t_g représente bien g et par conversion on en déduit :

$$\mathsf{read}\,(\mathsf{rec}[\mathsf{nat}]\,(t_g\,\overline{n_1}\,\cdots\,\overline{n_k})\,(t_f\,\overline{n_1}\,\cdots\,\overline{n_k})\,\overline{m'+1}) \equiv \overline{g(n_1,\ldots,n_k,h(n_1,\ldots,n_k,m'),m')}$$

C'est-à-dire :

$$\mathsf{read}\,(t_h\,\overline{n_1}\,\cdots\,\overline{n_k}\,\overline{m'+1}) \equiv \overline{h(n_1,\ldots,n_k,m'+1)}$$

☺

La définition suivante formalise la notion de récursivité comme une propriété de nos systèmes. Elle est donc paramétrée par un langage de programmation $\mathcal{P}$ qui joue un rôle analogue à celui des automates finis dans la formalisation de la calculabilité pour les machines de Turing. Ainsi, dès que $\mathcal{P}$ est un minimum expressif (c'est-à-dire dès qu'il est capable d'implémenter toutes les fonctions de transition des machines de Turing), alors la récursivité au sens de $\mathcal{P}$ sera équivalente à celle des machines de Turing.

On notera qu'on ne considère que le cas des fonctions unaires : le cas n-aire pourrait se développer de façon similaire (on ne perd pas en généralité car le codage des paires d'entiers dans les entiers permet de nous ramener au cas unaire).

4.2.50 Définition (Fonction récursive au sens de $\mathcal{P}$)
Soit un système $\mathcal{P}$ tel que $\mathcal{P}^2 \models$ NOTATION.
Une fonction numérique $f : \mathbb{N} \to \mathbb{N}$ est dite récursive au sens de $\mathcal{P}$ s'il existe un terme turing_f tel que $\vdash_{\mathcal{P}} \mathrm{turing}_f : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$ et vérifiant :

Pour tout couple d'entiers $n, m \in \mathbb{N}$, il existe un entier $k \in \mathbb{N}$ tel que $\mathsf{read}\,(\mathrm{turing}_f\,\overline{n}\,\overline{m}\,\overline{k}) \equiv \overline{1}$ si et seulement si $f(n) = m$.

Bien sûr avec une telle définition de la récursivité, on obtient "gratuitement" la récursivité au sens de $\mathcal{P}$ pour toutes les fonctions représentables dans $\mathcal{P}$:

4.2.51 Lemme (Représentable implique récursif)
Soit un système $\mathcal{P}$ tel que $\mathcal{P}^2 \models$ NOTATION $\wedge$ NONDEGENERATE $\wedge$ BIJECTIVE.
Les foncions représentables dans $\mathcal{P}$ sont récursives au sens de $\mathcal{P}$.

Démonstration On rappelle qu'une fonction $f : \mathbb{N} \to \mathbb{N}$ est représentable dans $\mathcal{P}$ s'il existe un terme $\vdash_{\mathcal{P}} t_f : \mathsf{nat} \to \mathsf{nat}$ tel que pour tout entier n, $t_f\,\overline{n}\overline{f(n)}$. On peut donc poser $\mathrm{turing}_f = \lambda n\,m\,k : \mathsf{nat.comp}\,(\mathsf{read}\,(t_f\,n))m$ (on notera que k n'est pas utilisé).

On a alors bien, d'après le lemme 4.2.47, pour tout n et tout m, l'existence d'un entier k (n'importe quel entier convient) tel que $\mathsf{read}\,(\mathrm{turing}_f\,\overline{n}\,\overline{m}\,\overline{k}) \cong \overline{1}$ si et seulement si $f(n) = m$ (car $\mathrm{turing}_f\,\overline{n}\,\overline{m}\,\overline{k} \equiv \mathsf{comp}\,(\mathsf{read}\,(t_f\,\overline{n}))\,\overline{m} \equiv \mathsf{comp}\,\overline{f(n)}\,\overline{m}$). ☺

4.2.52 Lemme
Soit un système $\mathcal{P}$ tel que $\mathcal{P}^2 \models$ NOTATION $\wedge$ NONDEGENERATE $\wedge$ BIJECTIVE et tel que $\mathcal{P} \models$ WEAKLY-NORMALISING, alors pour toute fonction $f : \mathbb{N} \to \mathbb{N}$, f est récursive au sens de $\mathcal{P}$ si et seulement si elle est récursive au sens de Turing (il existe une machine de Turing qui implémente f).

Démonstration
- $\Rightarrow$: Supposons f récursive au sens de $\mathcal{P}$. Alors il existe un terme turing_f typable dans $\mathcal{P}$. On peut implémenter f par une machine de Turing qui procède de la façon suivante :
  - Elle lit sur son ruban d'entrée un entier n,
  - Pour chaque couple d'entiers $(m,k) = (0,0),(0,1),(1,0),(0,2),(1,1),(0,2),\ldots$ énumérés selon une bijection $\mathbb{N} \to \mathbb{N}^2$:
    - Elle normalise[4] une représentation du terme $\mathsf{read}\,(\mathrm{turing}_f\,\overline{n}\,\overline{m}\,\overline{k})$.
    - Puis, elle teste si le résultat est convertible en $\overline{1}$:
      - Si c'est le cas, la machine écrit $\overline{m}$ sur son ruban de sortie et s'arrête.
      - Sinon elle continue avec le prochain couple.

  Alors le fait que $\mathcal{P} \models$ WEAKLY-NORMALISING nous montre que l'étape de normalisation termine toujours et on en déduit donc que la machine s'arrête sur l'entrée n si et seulement il existe k et m tels que $\mathsf{read}\,(\mathrm{turing}_f\,\overline{n}\,\overline{m}\,\overline{k}) \equiv \overline{1}$. C'est-à-dire si et seulement si $f(n) = m$.

- $\Leftarrow$: Soit $T_M(n,m,k)$ la fonction qui retourne 1 si et seulement si la machine de Turing M, lorsqu'elle démarre sur l'entrée n, s'arrête après k étapes en inscrivant m sur son ruban de sortie. On peut montrer (voir par exemple [Men09]) que T_M est une fonction primitive récursive.
  Soit M_f une machine de Turing qui implémente f, puisque $\mathcal{P}$ permet de représenter toutes les fonctions primitives récursives, on prend pour turing_f un terme représentant T_{M_f}. On a bien qu'il existe un entier $k \in \mathbb{N}$ tel que $\mathsf{read}\,(\mathrm{turing}_f\,\overline{n}\,\overline{m}\,\overline{k}) \equiv \overline{1}$ si et seulement si $f(n) = m$. ☺

4.2.53 Définition (Prouvablement totale)
Soit $\mathcal{P}$ un système tel que $\mathcal{P}^2 \models$ NOTATION. Une fonction $f : \mathbb{N} \to \mathbb{N}$ récursive au sens de $\mathcal{P}$ est dite prouvablement totale dans un contexte Γ si :

$$\Gamma \vdash_{\mathcal{P}^2} \forall x \in \mathbb{N}.\exists y \in \mathbb{N}.\exists k \in \mathbb{N}.\mathrm{turing}_f\,x\,y\,k =_{\mathsf{nat}} \overline{1}$$

4. En employant une stratégie de réduction en appel par nom au cas où la conjecture 1.1.2 est fausse.

4.2.54 Lemme (Prouvablement inductive implique prouvablement totale)
Soit un système $\mathcal{P}$ tel que $\mathcal{P}^2 \models$ NOTATION $\wedge$ NONDEGENERATE $\wedge$ BIJECTIVE,
et soit $f : \mathbb{N} \to \mathbb{N}$ une fonction représentée par un terme t_f.
Si $\Gamma \vdash_{\mathcal{P}^2} \forall x \in \mathbb{N}.(t_f\, x) \in \mathbb{N}$, alors la fonction f est prouvablement totale dans $\mathcal{P}^2$ (sous le contexte vide).

Démonstration On a vu au lemme 4.2.51 que le terme $\text{turing}_f = \lambda n\, m\, k : \mathsf{nat}.\mathsf{comp}\,(\mathsf{read}\,(t_f\, n))m$ est un témoin de la récursivité de f. Il faut alors montrer que le séquent suivant est prouvable :

$$\Gamma \vdash_{\mathcal{P}^2} \forall x \in \mathbb{N}.\exists y \in \mathbb{N}.\exists k \in \mathbb{N}.\text{turing}_f\, x\, y\, k =_{\mathsf{nat}} \overline{1}$$

C'est à dire par conversion, que :

$$\Gamma \vdash_{\mathcal{P}^2} \forall x \in \mathbb{N}.\exists y \in \mathbb{N}.\exists k \in \mathbb{N}.\mathsf{comp}\,(\mathsf{read}\,(t_f\, x))\, y =_{\mathsf{nat}} \overline{1}$$

Soit x de type nat vérifiant $x \in \mathbb{N}$. On prend comme témoin pour y le terme $(\mathsf{read}\,(t_f\, x))$, on montre alors que $(\mathsf{read}\,(t_f\, x)) \in \mathbb{N}$ grâce à l'hypothèse $\Gamma \vdash_{\mathcal{P}^2} \forall x \in \mathbb{N}.(t_f\, x) \in \mathbb{N}$ appliquée à x et H ; puis grâce au lemme 4.2.38. On peut prendre n'importe quel témoin pour k à condition qu'il vérifie le prédicat $\_ \in \mathbb{N}$ car k n'apparaît pas dans la conclusion (donc $k = \overline{0}$ convient). Reste donc à montrer que $\mathsf{comp}\,(\mathsf{read}\,(t_f\, x))\,(\mathsf{read}\,(t_f\, x)) =_{\mathsf{nat}} \overline{1}$. On utilise pour cela le lemme 4.2.46. ☺

Dans la section précédente nous avons vu que les fonctions représentables des systèmes qui contiennent $\mathcal{F}^2$ sont prouvablement inductives. On en déduit donc :

4.2.55 Lemme (Représentable implique prouvablement totale)
Supposons $\mathcal{F}^2 \subseteq \mathcal{P}^2$ tel que $\mathcal{P}^2 \models$ NONDEGENERATE $\wedge$ BIJECTIVE. Si $f : \mathbb{N} \to \mathbb{N}$ est représentée par un terme t_f de $\mathcal{P}$, alors f est prouvablement totale dans tout contexte Γ tel que $\Gamma \vdash \mathsf{ext\_nat}$.

Démonstration On a $\vdash_{\mathcal{P}} t_f : \mathsf{nat} \to \mathsf{nat}$. Donc d'après le lemme 4.2.36, on en déduit que $\Gamma \vdash_{\mathcal{P}^2} \forall x \in \mathbb{N}.(t_f\, x) \in \mathbb{N}$. Et donc d'après le lemme précédent, on en déduit que f est prouvablement totale dans Γ ☺

Le théorème suivant montre que les fonctions prouvablement totales sont représentables :

4.2.56 Théorème (Prouvablement totale implique représentable)
Soit $\mathcal{P}^2$ tel que $\mathcal{P}^2 \models$ NOTATION $\wedge$ NONDEGENERATE $\wedge$ BIJECTIVE. Soit Γ un contexte de $\mathcal{P}^2$ uniformément auto-réalisé, clos au second niveau et non-dégénéré tel que $\Gamma \vdash_{\mathcal{P}^2} \mathsf{datatype}$.
Si $f : \mathbb{N} \to \mathbb{N}$ est prouvablement totale dans Γ, alors f est représentable.

Démonstration Puisque f est prouvablement totale dans Γ, on a :

$$\Gamma \vdash_{\mathcal{P}^2} \forall x \in \mathbb{N}.\exists y \in \mathbb{N}.\exists k \in \mathbb{N}.\text{turing}_f\, x\, y\, k =_{\mathsf{nat}} \overline{1}$$

D'après le lemme 4.2.30, on en déduit que il existe deux termes t_1 et t_2 tels que $\vdash_{\mathcal{P}} t_1 : \mathsf{nat} \to \mathsf{nat}$, $\vdash_{\mathcal{P}} t_2 : \mathsf{nat} \to \mathsf{nat}$ et :

$$\Gamma \vdash_{\mathcal{P}^2} \forall x \in \mathbb{N}.(t_1\, x) \in \mathbb{N} \wedge (t_2\, x) \in \mathbb{N} \wedge \text{turing}_f\, x\, (t_1\, x)\, (t_2\, x) =_{\mathsf{nat}} \overline{1}$$

Montrons que t_1 représente f. Soit n un entier, il faut montrer que $\mathsf{read}\,(t_1\,\overline{n}) \equiv \overline{f(n)}$. On a $\vdash_{\mathcal{P}^2} \overline{n} \in \mathbb{N}$ (lemme 3.2.6), on en déduit donc que

$$\Gamma \vdash_{\mathcal{P}^2} \mathrm{turing}_f\,\overline{n}\,(t_1\,\overline{n})\,(t_2\,\overline{n}) =_{\mathsf{nat}} \overline{1}$$

De plus, on a, d'après le lemme 4.2.38 et le fait que $(t_1\,\overline{n}) \in \mathbb{N}$ et $(t_2\,\overline{n}) \in \mathbb{N}$:

$$\Gamma \vdash t_1\,\overline{n} =_{\mathsf{nat}} \mathsf{read}\,(t_1\,churchn) \text{ et } \Gamma \vdash t_2\,\overline{n} =_{\mathsf{nat}} \mathsf{read}\,(t_2\,churchn)$$

On en déduit :

$$\Gamma \vdash_{\mathcal{P}^2} \mathrm{turing}_f\,\overline{n}\,(\mathsf{read}\,(t_1\,\overline{n}))\,(\mathsf{read}\,(t_2\,\overline{n})) =_{\mathsf{nat}} \overline{1}$$

La non-dégénérescence du contexte montre alors que :

$$\mathsf{read}\,(\mathrm{turing}_f\,\overline{n}\,(\mathsf{read}\,(t_1\,\overline{n}))\,(\mathsf{read}\,(t_2\,\overline{n})) \equiv \overline{1}$$

Enfin, par surjectivité du système on en déduit qu'il existe k_1 et k_2 tels que $\mathsf{read}\,(t_1\,\overline{n}) \equiv \overline{k_1}$ et $\mathsf{read}\,(t_2\,\overline{n}) \equiv \overline{k_2}$ et donc :

$$\mathsf{read}\,(\mathrm{turing}_f\,\overline{n}\,\overline{k_1}\,\overline{k_2}) \equiv \overline{1}$$

Et par définition de turing_f on en déduit que $f(n) = k_1$ et donc que $\mathsf{read}\,(t_1\,\overline{n}) \equiv \overline{f(n)}$. ☺

Voici l'énoncé du théorème de représentation de Jean-Yves Girard tel qu'on le trouve dans *Proofs and Types* [GLT89] :

4.2.57 Théorème (Théorème de représentation de Jean-Yves Girard [Gir72])
Les fonctions représentables de $\mathcal{F}$ sont exactement les fonctions prouvablement totales dans l'arithmétique de Peano du second-ordre $\mathbf{PA}_2$.

Pour y parvenir, il commence par prouver le résultat pour l'arithmétique de Heyting $\mathbf{HA}_2$ du second-ordre qui est similaire à l'arithmétique de Peano, à l'exception de la possibilité de prouver le tiers-exclu. Une fois le résultat obtenu pour $\mathbf{HA}_2$, Jean-Yves Girard montre que l'ajout du tiers-exclu ne change pas les fonctions prouvablement totales. Il utilise pour cela les traductions standards de la logique classique dans la logique intuitionniste[5].

Les axiomes de l'arithmétique sont nécessaires afin de développer les codages des machines de Turing suffisant à énoncer la "prouvabilité totale". Comme nous avons pu le voir, dans notre cadre, toute la puissance de l'arithmétique n'est pas nécessaire puisque les individus sont suffisamment expressifs pour pouvoir exprimer cette propriété.

Nous généralisons donc le résultat de Girard selon deux axes :

- Notre théorème peut être appliqué à tous les systèmes qui contiennent $\mathcal{F}^2$ (c'est-à-dire qui représentent les entiers selon l'encodage imprédicatif),
- Il peut être utilisé avec n'importe quelle théorie à condition qu'elle vérifie les conditions du théorème : cela signifie que l'on peut ajouter à **HA** n'importe quel axiome uniformément autoréalisé (mais il faudra toutefois prouver que ces ajouts n'ont pas pour conséquence la perte de non-dégénérescence).

5. Nous n'aborderons pas ce sujet dans cette thèse, mais il pourrait être intéressant d'essayer de porter ces traductions dans notre contexte ; ce qui ne semble pas être une mince affaire puisqu'il faut montrer qu'elles se comportent bien vis-à-vis des produits dépendants.

Le théorème de représentation s'énonce donc de la façon suivante :

4.2.58 Théorème (Théorème de représentation)
Soit $\mathcal{P}$ tel que $\mathcal{F}^2 \subseteq \mathcal{P}^2$ et $\mathcal{P}^2 \models$ BIJECTIVE. Et soit Γ un contexte de $\mathcal{P}^2$ uniformément auto-réalisé, clos au second niveau et non-dégénéré tel que $\Gamma \vdash_{\mathcal{P}^2} \mathsf{ext\_nat}$.
Alors, $f : \mathbb{N} \to \mathbb{N}$ est prouvablement totale dans Γ si et seulement si f est représentable.

Démonstration Le sens direct est démontré par le lemme précédent et le lemme 4.2.55 prouve la réciproque. ☺

En particulier, on obtient :

4.2.59 Lemme
Pour $\mathcal{P} = \mathcal{F}, \mathcal{F}_n, \mathcal{F}_\omega$ ou CC*, si on suppose que le contexte $\Gamma = \mathsf{Ax} : \mathsf{ext\_nat}$ est non-dégénéré.*
Alors, $f : \mathbb{N} \to \mathbb{N}$ est prouvablement totale dans $\mathcal{P}^2$ sous le contexte Γ si et seulement si f est représentable.

Ce résultat n'est pas nouveau : Jean-Yves Girard avait déjà étendu ses résultats pour $\mathcal{F}$ à $\mathcal{F}_\omega$ (mentionné dans l'annexe de [Gir90]). De plus, nous avons constaté que les fonctions représentables de CC sont les mêmes que celles de $\mathcal{F}_\omega$ (lemme 1.1.22). Ce qui est original ici, c'est de fournir un cadre général dans lequel ces résultats sont des instances d'un même théorème.

4.3 Le cas des systèmes utilisant les inductifs

Dans cette section, nous allons chercher à appliquer les résultats de la section précédente. Dans un premier temps, nous allons voir sur des exemples comment les transformations de réalisabilité se comportent en présence d'inductifs ; nous admettrons donc dans un premier temps que le théorème 4.1.8 s'étend aux inductifs. Ce sera dans la section 4.3.3 que nous verrons formellement comment étendre la transformation $\cdot \mapsto \cdot^2$ aux inductifs, puis nous esquisserons la preuve du théorème 4.1.8.

4.3.1 Introduction par l'exemple : le cas de $\mathcal{T}$

Nous rappelons que le système $\mathcal{T}$ est le CTSI défini par les spécifications du λ-calcul simplement typé λ auquel on ajouta les trois inductifs présentés ci-dessous dans la syntaxe de Coq (voir page 127).

```
Inductive nat : Set :=
  | zero : nat
  | succ : nat→ nat.

Inductive bool : Set :=
  | true : bool
  | false : bool.

Inductive prod (α β : Set) : Set :=
  | pair : α→ β→ prod α β.
```

Afin que le lecteur puisse "tester" les exemples donnés avec la syntaxe de Coq, nous ne distinguerons pas les applications des paramètres des inductifs de l'application usuelle des fonctions. Ainsi, dans l'exemple ci-dessus le sous-terme `prod` α β représente le terme $\mathsf{prod}[\alpha, \beta]$ de $\mathcal{T}$. De même, la sorte `Set` des exemples en Coq désignera la sorte $\star$ de $\mathcal{T}$. Par ailleurs, on rappelle que l'on emploiera la notation $\sigma \times \tau$ pour désigner $\mathsf{prod}[\sigma, \tau]$.

Nous allons maintenant décrire la logique engendrée à partir de $\mathcal{T}$ de façon informelle ; elle sera notée $\mathcal{T}^2$. La définition générale sera donnée dans la section suivante : elle sera définie comme un CTSI dont les sortes, les axiomes et les règles sont les mêmes que le système λ^2 :

$$\begin{aligned}
\mathcal{S}_{\lambda^2} &= \{ \star, \Box, \lceil\star\rceil, \lceil\Box\rceil \} \\
\mathcal{A}_{\lambda^2} &= \{ (\star, \Box), (\lceil\star\rceil, \lceil\Box\rceil) \} \\
\mathcal{R}_{\lambda^2} &= \{(\star,\star,\star), (\lceil\star\rceil, \lceil\star\rceil, \lceil\star\rceil), (\star, \lceil\star\rceil, \lceil\star\rceil) \}
\end{aligned}$$

Dans les exemples illustrés avec la syntaxe de Coq, la sorte $\lceil\star\rceil$ sera représentée par la sorte `Prop`. En effet, le système λ^2 se plonge dans CIC via le morphisme suivant :

$$\star \mapsto \texttt{Set} \qquad \lceil\star\rceil \mapsto \texttt{Prop} \qquad \Box \mapsto \texttt{Type} \qquad \lceil\Box\rceil \mapsto \texttt{Type}$$

Comme on peut l'observer, les sortes $\Box$ et $\lceil\Box\rceil$ sont envoyées vers la même sorte de CIC. On notera toute fois que l'on ne perd pas beaucoup d'information puisque le seul habitant de $\Box$ (resp. $\lceil\Box\rceil$) est $\star$ (resp. $\lceil\star\rceil$) ; il est donc aisé, étant données l'image d'un terme par ce morphisme, de retrouver le terme de départ.

La logique sous-jacente au système λ^2 est une logique du premier-ordre avec :
- Une quantification sur les individus dont le type vit dans $\star$ (règle : $(\star, \lceil\star\rceil, \lceil\star\rceil)$),
- Ainsi que la possibilité de former l'implication entre deux formules (règle : $(\lceil\star\rceil, \lceil\star\rceil, \lceil\star\rceil)$).

Néanmoins, cette logique est aussi peu expressive que λ l'est en tant que langage de programmation. En effet, tout comme dans λ, les seuls habitants clos des sortes sont des sortes dans λ^2 : il n'y a pas d'atomes pour former les formules et les types. C'est pourquoi $\mathcal{T}$ est une extension intéressante de λ dans la mesure où elle fournit deux types de bases (`nat` et `bool`) qui permettent de construire des types clos et de représenter les données. Ainsi, dans $\mathcal{T}^2$, ce seront les inductifs qui fourniront les atomes logiques.

Inductifs autorisés dans $\mathcal{T}^2$. Les inductifs autorisés dans $\mathcal{T}^2$ seront définis comme les inductifs qui, lorsqu'on leur applique la fonction de projection $\lfloor\cdot\rfloor$, donnent la définition d'un des trois inductifs de $\mathcal{T}$ (modulo renommage du nom de l'inductif et des constructeurs).

Par exemple l'inductif suivant, qui nous servira à implémenter la notation pour l'arithmétique $x \in \mathbb{N}$, se projette bien vers l'inductif `nat`.

```
Inductive nat_ind : nat → Prop :=
  | zero_ind : nat_ind zero
  | succ_ind : forall y:nat, nat_ind y → nat_ind (succ y).
```

En effet, on a :
- $\lfloor \mathsf{nat} \to \mathsf{prop} \rfloor = \lfloor \mathsf{prop} \rfloor = \mathsf{set} : \mathsf{nat}$ est effacé car c'est un terme de premier niveau,
- $\lfloor \mathsf{nat\_ind}\,\mathsf{zero} \rfloor = \lfloor \mathsf{nat\_ind} \rfloor : \mathsf{zero}$ est effacé car il est également de premier niveau.

– $\lfloor \forall y : \mathsf{nat.nat\_ind}\, y \to \mathsf{nat\_ind}\,(\mathsf{S}\, y) \rfloor = \lfloor \mathsf{nat\_ind} \rfloor \to \lfloor \mathsf{nat\_ind} \rfloor$: car $\forall y : \mathsf{nat}$ est une quantification de premier niveau et $\mathsf{nat\_ind}\, y$ et $\mathsf{nat\_ind}\,(\mathsf{S}\, y)$ sont des sous-termes de premier niveau.

On étendra donc la fonction de projection aux noms des inductifs et des constructeurs en prenant l'inductif du système de départ vers lequel ils se projettent. Par exemple, on a aura ici :

$$\begin{aligned} \lfloor \mathsf{nat\_ind} \rfloor &= \mathsf{nat} \\ \lfloor \mathsf{zero\_ind} \rfloor &= \mathsf{zero} \\ \lfloor \mathsf{succ\_ind} \rfloor &= \mathsf{succ} \end{aligned}$$

De toute évidence, nat_ind n'est pas le seul inductif qui se projette sur nat. Comme par exemple le prédicat "être pair" suivant :

```
Inductive even : nat → Prop :=
  | zero_even : nat_ind zero
  | succ_even : forall y:nat, even y →
                              even (succ (succ y)).
```

On a $\lfloor \mathsf{even} \rfloor = \mathsf{nat}$. Nous verrons que tout comme dans le cas sans inductif, la fonction de projection permet d'obtenir le type des réalisateurs des formules. On voit ainsi que le prédicat "être pair" est réalisé par un entier k dont on devine qu'il témoignera de la parité de $2k$.

La transformation de réalisabilité peut être également appliquée aux inductifs avec paramètres, comme par exemple :

```
Inductive and (X Y : Prop) : Prop :=
  | conj : X→ Y→ and X Y.
```

Cet inductif permet d'encoder le connecteur logique de la conjonction. Dans notre syntaxe, il est donné par la déclaration suivante :

$$\mathtt{Ind}(X : \lceil\star\rceil, Y : \lceil\star\rceil, \mathsf{and} : \lceil\star\rceil, \mathsf{conj} : X \to Y \to \mathsf{and}[X, Y])$$

et on notera $P \wedge Q$ le terme $\mathsf{and}[P, Q]$. On constate alors $\lfloor \mathsf{and} \rfloor = \mathsf{prod}$, c'est-à-dire que les conjonctions sont réalisées par des produits cartésiens.

On peut également étendre la fonction de soulèvement aux inductifs du système de départ, en appliquant la fonction $\lceil\cdot\rceil$ à chacun des constructeurs de l'inductif, aux paramètres et à l'arité. Ainsi la projection demeure une rétraction du soulèvement ($\lfloor \lceil A \rceil \rfloor = A$).

Par exemple, on définit $\lceil \mathsf{nat} \rceil$ par l'inductif suivant :

```
Inductive ⌈nat⌉ : ⌈Set⌉ :=
  | ⌈zero⌉ : ⌈nat⌉
  | ⌈succ⌉ : ⌈nat⌉→ ⌈nat⌉.
```

où $\lceil \mathsf{set} \rceil = \mathsf{prop}$.

Élimination autorisée dans $\mathcal{T}^2$. Dans le paragraphe précédent, nous avons décrit les inductifs que l'on pouvait utiliser dans $\mathcal{T}^2$ et nous avons vu comment étendre la fonction de projection et la fonction de soulèvement aux définitions inductives. Il nous reste à voir comment se comporte l'élimination des inductifs pour avoir complètement illustré $\mathcal{T}^2$.

Le rôle de la fonction de projection est d'effacer les sous-termes de premier niveau pour pouvoir ensuite se propager au travers de la construction `case`. Il est donc primordial que l'on ne puisse pas détruire d'objet du premier niveau afin de construire un objet du second niveau. En effet, la fonction de projection serait alors dans l'impossibilité de "choisir une branche" de l'élimination.

Ainsi la fonction ci-dessous doit être rejetée par le système :

```
Fixpoint f (n : nat) : ⌈nat⌉ :=
  match n with
    | zero  ⇒  zero_ind
    | succ p  ⇒  succ_ind (f p)
  end.
```

Si cette définition était autorisée la fonction de projection effacerait la variable n et ne saurait pas quelle branche garder pour calculer $\lfloor f \rfloor$. C'est pourquoi l'ensemble $\mathcal{E}_{\mathcal{P}^2}$ est défini ainsi :

$$\mathcal{E}_{\mathcal{P}^2} = \mathcal{E}_{\mathcal{P}} \cup \{(I, \lceil s \rceil) | I \in \mathcal{I}_{\mathcal{P}^2}, \mathrm{lvl}(I) = 2, (\lfloor I \rfloor, s) \in \mathcal{E}_{\mathcal{P}} \}$$

où le niveau d'un inductif (noté $\mathrm{lvl}(I)$) est donné par le niveau de son arité. On peut paraphraser cette idée de la façon suivante :

> Il est autorisé de détruire un inductif I de second niveau pour fabriquer un objet dont le type est de sorte s si et seulement si on peut détruire l'inductif $\lfloor I \rfloor$ de premier niveau pour fabriquer un objet dont le type est de sorte $\lfloor s \rfloor$.

Le système $\mathcal{T}^{++}$. Le système $\mathcal{T}^2$ engendré par $\mathcal{T}$ ne contient pas suffisamment d'inductifs pour pouvoir engendrer tous les atomes nécessaires pour avoir un système de notations pour l'arithmétique. En effet, il manque :

- Un type vide pour typer les réalisateurs[6] de $\bot$,
- Un type à un élément pour typer les réalisateurs des égalités,
- Un type opérateur pour encapsuler les types afin de réaliser les connecteurs existentiels.

On rajoute donc à $\mathcal{T}^2$ les trois inductifs suivants :

```
Inductive empty : Set :=.
Inductive unit : Set :=
  | tt : unit.
Inductive box (α : Set) : Set :=
  | close : α→ box α.
```

Dans notre syntaxe il sont donnés par les déclarations suivantes :

- $\mathsf{Ind}(\mathsf{empty} : \star)$
- $\mathsf{Ind}(\mathsf{unit} : \star, \mathsf{tt} : \mathsf{unit})$
- $\mathsf{Ind}(\alpha : \star, \mathsf{box} : \star, \mathsf{close} : \alpha \to \mathsf{box}[\alpha])$

On appelle $\mathcal{T}^{++}$ le système ainsi obtenu.

On peut montrer que ces trois inductifs ne rajoutent pas de réel pouvoir calculatoire au système $\mathcal{T}$:

6. Il n'y en aucun, mais il leur faut tout de même un type.

4.3.1 Lemme
Pour toute fonction $f : \mathbb{N} \to \mathbb{N}$,
$f : \mathbb{N} \to \mathbb{N}$ *est représentable dans* $\mathcal{T}$ *si et seulement si elle est représentable dans* $\mathcal{T}^{++}$.

Démonstration (Esquisse) Le sens direct est trivial puisque $\mathcal{T} \subseteq \mathcal{T}^{++}$. Pour montrer la réciproque nous allons définir une transformation des termes de $\mathcal{T}^{++}$ dans ceux de $\mathcal{T}$ qui préserve la réduction.

On peut montrer facilement que les types de $\mathcal{T}^{++}$ sont générés par la grammaire suivante :

$$\begin{array}{rcl} \sigma, \tau & := & \mathsf{nat} \\ & | & \mathsf{bool} \\ & | & \mathsf{unit} \\ & | & \mathsf{empty} \\ & | & (\!|\tau|\!) \\ & | & \sigma \times \tau \\ & | & \sigma \to \tau \end{array}$$

Et on peut les envoyer vers les types clos de $\mathcal{T}$ par la transformation inductive suivante :

$$\begin{array}{rcl} |\mathsf{nat}| & = & \mathsf{nat} \\ |\mathsf{bool}| & = & \mathsf{bool} \\ |\mathsf{unit}| & = & \mathsf{nat} \\ |\mathsf{empty}| & = & \mathsf{nat} \\ |\ (\!|\tau|\!)| & = & \tau \\ |\sigma \times \tau| & = & |\sigma| \times |\tau| \\ |\sigma \to \tau| & = & |\sigma| \to |\tau| \end{array}$$

Pour la transformation des termes, le seul cas délicat est celui de l'élimination sur l'inductif vide. Elle nécessite en effet de trouver un habitant de n'importe quel type pour remplacer l'élimination ; heureusement les types de $\mathcal{T}$ ont la propriété appréciable d'être tous habités. Étant donné un type clos τ de $\mathcal{T}$, on définit un *habitant canonique* τ^* de τ de la façon suivante :

$$\begin{array}{rcl} \mathsf{nat}^* & = & \mathsf{zero} \\ \mathsf{bool}^* & = & \mathsf{true} \\ (\sigma \times \tau)^* & = & \mathsf{pair}[\sigma, \tau]\, \sigma^*\, \tau^* \\ (\sigma \to \tau)^* & = & \lambda x : \sigma.\tau^* \end{array}$$

On montre alors facilement que si $\vdash \tau : \star$ alors $\vdash \tau^* : \tau$.

On peut maintenant définir la transformation sur les termes (le seul cas non-trivial est le cas de la

traduction des éliminations sur empty) :

$$\begin{aligned}
|A| &= A \text{ pour } A = \mathsf{true}, \mathsf{false}, \mathsf{zero}, \mathsf{succ} \\
|(M\,N)| &= (|M|\,|N|) \\
|\lambda x : \tau.M| &= \lambda x : |\tau|.|M| \\
|\mathsf{tt}| &= \mathsf{zero} \\
|\mathsf{close}[\tau]| &= \lambda x : \tau.x \\
|\mathsf{pair}[\sigma,\tau]| &= \mathsf{pair}[|\sigma|,|\tau|] \\
|\,\mathsf{case}_{\mathsf{unit}}(M,\lambda_ : \mathsf{unit}.\tau,N)| &= |N| \\
|\,\mathsf{case}_{\mathsf{box}}(M,\tau,\lambda_ : (\!|\tau|\!).\sigma,\lambda x : \tau.N)| &= (\lambda x : \tau.|N|)\,|M| \\
|\,\mathsf{case}_{\mathsf{empty}}(M,\lambda_ : \mathsf{empty}.\tau)| &= |\tau|^* \\
|\,\mathsf{case}_{\mathsf{bool}}(M,\lambda_ : \mathsf{bool}.\tau,N_1,N_2)| &= \mathsf{case}_{\mathsf{bool}}(|M|,\lambda_ : \mathsf{bool}.|\tau|,|N_1|,|N_2|)| \\
|\,\mathsf{case}_{\mathsf{nat}}(M,\lambda_ : \mathsf{nat}.\tau,N_1,\lambda p : \mathsf{nat}.N_2)| &= \mathsf{case}_{\mathsf{nat}}(|M|,\lambda_ : \mathsf{nat}.|\tau|,|N_1|,\lambda p : \mathsf{nat}.|N_2|)|
\end{aligned}$$

$$\begin{aligned}
&|\,\mathsf{case}_{\mathsf{prod}}(M,\tau_1,\tau_2,\lambda_ : \mathsf{prod}[\tau_1,\tau_2].\sigma,\lambda x_1 : \tau_1,x_2 : \tau_2.N)| = \\
&\qquad \mathsf{case}_{\mathsf{prod}}(|M|,|\tau_1|,|\tau_2|,\lambda_ : \mathsf{prod}[|\tau_1|,|\tau_2|].|\sigma|,\lambda x_1 : |\tau_1|,x_2 : |\tau_2|.|N|)
\end{aligned}$$

On peut alors vérifier que si $\vdash_{\mathcal{T}^{++}} M : \tau : \star$, alors $\vdash_{\mathcal{T}} |M| : |\tau|$ et que $M \trianglerighteq M'$ implique $|M| \trianglerighteq |M'|$.

Soit un terme $\vdash_{\mathcal{T}^{++}} t_f : \mathsf{nat} \to \mathsf{nat}$ qui représente f dans $\mathcal{T}^{++}$ alors on en déduit bien que le terme $|t_f|$ représente également f dans $\mathcal{T}$ car $\overline{f(n)} \equiv (t_f\,\overline{n}) \equiv (|t_f|\,\overline{n})$. ☺

Ainsi dans le système $\mathcal{T}^{++^2}$, on peut utiliser les inductifs pour représenter l'absurdité logique, l'égalité de Leibniz et le connecteur existentiel. On utilise pour cela les inductifs suivants :

```
Inductive False : Prop := .
Inductive eq (α : Set) (x : α) : α→ Prop :=
  | refl : eq α x x.
Inductive exists (α : Set) (P : α→ Prop) : Prop :=
  | ex_intro : forall x:α, P x→ ex α P.
```

Dans notre syntaxe, ces inductifs sont représentés par les déclarations suivantes :

- $\mathtt{Ind}(\mathsf{False} : \lceil\star\rceil)$,
- $\mathtt{Ind}(\alpha : \star, x : \alpha, \mathsf{eq} : \alpha \to \lceil\star\rceil, \mathsf{refl} : \mathsf{eq}[\alpha, x]\,x)$,
- $\mathtt{Ind}(\alpha : \star, P : \alpha \to \lceil\star\rceil, \mathsf{exists} : \lceil\star\rceil, \mathsf{ex\_intro} : \forall x : \alpha, P\,x \to \mathsf{exists}[\alpha, P])$.

Ce sont bien des inductifs de $\mathcal{T}^{++^2}$ car :

$$\begin{aligned}
\lfloor\mathsf{False}\rfloor &= \mathsf{empty} \\
\lfloor\mathsf{eq}[\alpha,x]\rfloor &= \mathsf{unit} \\
\lfloor\mathsf{exists}[\alpha,P]\rfloor &= (\!|\lfloor P\rfloor|\!)
\end{aligned}$$

On notera $\bot$ le terme False, $M =_\tau N$ le terme $\mathsf{eq}[\tau, M]\,N$, et $\exists x : \tau.P$ le terme $\mathsf{exists}[\alpha, \lambda x : \tau.P]$.

On peut donc représenter dans $\mathcal{T}^{++^2}$ toutes les notations réquises pour le munir d'un système de notations pour l'arithmétique :

– nat est défini par l'inductif `nat` décrit page 246.
– S et $\overline{0}$ sont donnés par les constructeurs de l'inductif nat.
– Le produit cartésien $\sigma \times \tau$ est une notation pour l'inductif $\mathsf{prod}\,\sigma, \tau$ décrit page 246.
– $\mathsf{pair}[\sigma, \tau]$ est le constructeur $\mathsf{conj}[\sigma, \tau]$ de l'inductif précédent.
– Les projections $\mathsf{proj}^1[\sigma, \tau]$ et $\mathsf{proj}^2[\sigma, \tau]$ sont implémentées par les deux fonctions suivantes :

```
fun c:σ × τ  ⇒ match c return σ with pair x y  ⇒
   x end
fun c:σ × τ  ⇒ match c return τ with pair x y  ⇒
   y end
```

– $(\!|\tau|\!)$ est donné par l'inductif `box` décrit page 249.
– $\mathsf{close}[\tau]$ est donné par le constructeur $\mathsf{close}[\tau]$ de l'inductif précédent.
– $\mathsf{open}[\tau]$ est donné par la fonction suivante :

```
fun c:(|τ|)  ⇒ match c return τ with close x  ⇒ x
   end
```

– $\mathsf{rec}[\tau]$ est donné par la fonction suivante :

```
fun f : τ→ nat→ τ, x : τ  ⇒
  fix g (n : nat) : nat  ⇒
     match n with
      | zero  ⇒  x
      | succ p  ⇒  f (g p) p
     end
```

– $\bot$ est défini par $\lceil\mathsf{empty}\rceil$ décrit page 251.
– $P \wedge Q$ est défini par l'inductif $\mathsf{and}[P, Q]$ décrit page 248.
– $M =_\tau N$ est défini par l'inductif $\mathsf{eq}[\tau]\, M\, N$ décrit page 251.
– $M \in \mathbb{N}$ est défini par l'inductif $\mathsf{nat\_ind}\, M$ décrit page 247.
– $\exists x : \tau.P$ est défini par l'inductif $\mathsf{exists}[\tau, \lambda x : \tau.P]$ décrit page 251.

FIGURE 4.1 – Système de notations pour l'arithmétique dans $\mathcal{T}^{++^2}$ (première partie)

– $\mathsf{elim}_\bot[P, \pi]$ est donné par le terme suivant :

```
match π return P with end.
```

– $\mathsf{intro}_\wedge[P, Q, \pi_1, \pi_2]$ est donné par le constructeur de l'inductif $\mathsf{and}[P, Q]$.
– $\mathsf{elim\text{-}gauche}_\wedge[P, Q, \pi]$ et $\mathsf{elim\text{-}droite}_\wedge[P, Q, \pi]$ sont donnés par les deux termes :

```
match π return P with conj x y  ⇒ x
match π return Q with conj x y  ⇒ y
```

– $\mathsf{intro}_\exists[\tau, x, P, M, \pi]$ par le constructeur de l'inductif $\mathsf{exist}[\tau, \lambda x : \tau.P]$.
– $\mathsf{elim}_\exists[\tau, x, P, h, Q, \pi, \varrho]$ est donné par le terme suivant :

```
matchp π return Q return
  | ex_intro x h  ⇒  ϱ
end.
```

– $\mathsf{intro\text{-}zero}_\mathbb{N}$ et $\mathsf{intro\text{-}succ}_\mathbb{N}[M, \pi]$ sont donnés par les constructeurs de l'inductif *nat_ind*.
– $\mathsf{induction}_\mathbb{N}[x, P, \pi, N, \pi_1, \pi_2]$ est donné par le terme suivant :

```
(fix f (x : nat) (h : nat_ind x) : P :=
   match h in nat_ind x return P with
     | zero  ⇒  π2
     | succ x h  ⇒  π1
   end) N π
```

– $\mathsf{intro}_=[M, \pi]$ est donné par le constructeur de l'inductif eq.
– $\mathsf{elim}_=[\tau, x, P, \pi, \varrho]$ est donné par le terme suivant :

```
match π in eq _ _ x return P with
  | refl  ⇒  ϱ
end.
```

– Enfin, les termes $\mathsf{open\text{-}close}[\tau]$, $\mathsf{pair\text{-}proj}^1[\sigma, \tau]$, $\mathsf{pair\text{-}proj}^2[\sigma, \tau]$, $\mathsf{rec\text{-}succ}[\tau]$, et $\mathsf{rec\text{-}zero}[\tau]$ sont donnés par les "identités" respectives suivantes :

```
fun x:τ  ⇒ refl τ x
fun (x:σ) (y:τ)  ⇒ refl σ x
fun (x:σ) (y:τ)  ⇒ refl τ y
fun (f : τ→ nat→ τ) (x : τ) (n : nat) (h : nat_ind
   n)  ⇒ refl τ (f (rec[τ] f x n) n)
fun (f : τ→ nat→ τ) (x : τ)  ⇒ refl τ x
```

Ces termes de preuves sont tous des projections de la preuve de la réflexivité car dans ce système les programmes, dont on veut prouver l'égalité, sont convertibles. On notera en particulier que le terme pour $\mathsf{rec\text{-}succ}[\tau]$ n'a pas besoin d'utiliser le principe d'induction $\mathsf{nat\_ind}\, n$ (alors qu'il était nécessaire dans $\mathcal{F}^2$).

FIGURE 4.2 – Système de notations pour l'arithmétique dans $\mathcal{T}^{++^2}$ (seconde partie)

4.3.2 Lemme

$$\mathcal{T}^{++^2} \models \text{NOTATION} \wedge \text{NOTATION-REALIZABILITY}$$

Démonstration

- NOTATION : les différents connecteurs sont donnés par la figure 4.1 et les règles d'introduction et d'élimination par la figure 4.2.
- NOTATION-REALIZABILITY : on vérifie bien :
  - $\lfloor \mathsf{prop} \rfloor = \lfloor \lceil \star \rceil \rfloor = \star = \mathsf{set}$,
  - $\lfloor N \in \mathbb{N} \rfloor = \lfloor \mathsf{nat\_ind}\, M \rfloor = \mathsf{nat}$,
  - $\lfloor P \wedge Q \rfloor = \lfloor \mathsf{and}[P, Q] \rfloor = \mathsf{conj}[\lfloor P \rfloor, \lfloor Q \rfloor] = \lfloor P \rfloor \times \lfloor Q \rfloor$,
  - $\lfloor \exists x : \tau . P \rfloor = \lfloor \mathsf{exists}[\tau, \lambda x : \tau . P] \rfloor = \mathsf{box}[\lfloor P \rfloor] = (\!|\lfloor P \rfloor|\!)$. ☺

Réalisabilité en présence d'inductifs. La transformation de réalisabilité s'adapte aisément aux inductifs. Pour appliquer la tranformation de réalisabilité à un inductif, il suffit de l'appliquer à l'arité et à chacun des constructeurs. Ainsi si

$$\mathtt{Ind}(\overrightarrow{x : Q}^{p}, I : A, \overrightarrow{c : C}^{k})$$

est une définition inductive de second niveau (c-à-d. $\mathrm{lvl}(A) = 2$), alors l'inductif de nom I_R et de constructeurs $c_{1R}, \ldots, c_{kR}$ défini par la déclaration

$$\mathtt{Ind}(\|\overrightarrow{x : Q}^{p}\|, I_R : \lfloor I \rfloor [\overrightarrow{x}^{p}] \Vdash A, \overrightarrow{c_R : \lfloor c \rfloor [\overrightarrow{x}^{p}] \Vdash C}^{k})$$

est un inductif de second niveau bien formé qui a les mêmes réalisateurs que I :

$$\lfloor I \rfloor = \lfloor I_R \rfloor$$

Nous donnons quelques exemples de cette transformation :

L'inductif even, vu précédemment dans la syntaxe de Coq, se présente dans notre système de la façon suivante :

$$\begin{aligned} &\mathtt{Ind}(\mathsf{even} : \mathsf{nat} \to \mathtt{Prop},\\ &\quad \mathsf{zero\_even} : \mathsf{even}\,\mathsf{zero}\\ &\quad \mathsf{succ\_even} : \forall y : \mathsf{nat}.\mathsf{even}\, y \to \mathsf{even}\,(\mathsf{succ}\,(\mathsf{succ}\, y))) \end{aligned}$$

Nous avons déjà remarqué que les réalisateurs de ce prédicat sont de type $\lfloor \mathsf{even} \rfloor = \mathsf{nat}$. Voyons maintenant ce que signifie la formule $r \Vdash \mathsf{even}\, y$ en appliquant la transformation de réalisabilité à l'inductif even :

$$\begin{aligned} &\mathtt{Ind}(\mathsf{even}_R : \mathsf{even} \Vdash \ \mathsf{nat} \to \lceil \star \rceil,\\ &\quad \mathsf{zero\_even}_R : \mathsf{zero} \Vdash (\mathsf{even}\,\mathsf{zero}),\\ &\quad \mathsf{succ\_even}_R : \mathsf{succ} \Vdash (\forall y : \mathsf{nat}.\mathsf{even}\, y \to \mathsf{even}\,(\mathsf{succ}\,(\mathsf{succ}\, y)))) \end{aligned}$$

en dépliant la notation de réalisabilité dans les types des constructeurs et dans l'arité on obtient :

$$\begin{aligned}&\mathtt{Ind}(\mathsf{even}_R : \forall n : \mathsf{nat}.\mathsf{even}\, n \to \lceil\star\rceil\\ &\quad \mathsf{zero\_even}_R : \mathsf{even}_R\, \mathsf{zero}\, \mathsf{zero}\\ &\quad \mathsf{succ\_even}_R : \forall y\, r : \mathsf{nat}.\mathsf{even}_R\, r\, y \to \mathsf{even}_R\, (\mathsf{succ}\, r)\, (\mathsf{succ}\, (\mathsf{succ}\, y)))\end{aligned}$$

C'est-à-dire dans la syntaxe de Coq :

```
Inductive even_R : forall n, even n→ Prop :=
  | zero_even_R : even_R zero zero
  | succ_even_R : forall y:nat, r : nat, even_R r y→
                  even_R (succ r) (succ (succ y)).
```

On peut alors montrer que si $r \Vdash \mathsf{even}\, x$ alors r est égal à la moitié de x :

$$\vdash_{\mathcal{T}^{++^2}} \forall x\, r : \mathsf{nat}, \mathsf{even}_R\, r\, x \to 2 * r =_{\mathsf{nat}} x$$

On en déduit donc que le prédicat even est honnête :

$$\vdash_{\mathcal{T}^{++^2}} \forall x\, r : \mathsf{nat}, \mathsf{even}_R\, r\, x \to \mathsf{even}\, x$$

La liste des paramètre $\overrightarrow{x : Q}^p$ est traduite comme un contexte. Par exemple, en traduisant l'inductif and, on obtient :

$$\begin{aligned}\mathtt{Ind}\,(\quad & X : \star, X_R : X \to \lceil\star\rceil,\\ & Y : \star, X_R : Y \to \lceil\star\rceil,\\ & \mathsf{and}_R : X \times Y \to \lceil\star\rceil,\\ & \mathsf{conj}_R : \forall x : X, X_R\, x \to \forall y : Y, Y_R\, y \to \mathsf{and}_R\, (\mathsf{pair}[X, Y]\, x\, y))\end{aligned}$$

C'est-à-dire dans la syntaxe de Coq :

```
Inductive and_R (X : Set) (X_R : X→ Prop)
                (Y : Set) (Y_R : Y→ Prop) : pair X Y→ Prop :=
 | conj_R : forall x : X, X_R x→
            forall y : Y, Y_R y→
            and_R (pair X Y x y).
```

On voit alors qu'une paire (x, y) réalise une conjonction $P \wedge Q$ si et seulement si x réalise P et y réalise Q :

4.3.3 Lemme

Si $\Gamma \vdash P : \mathsf{prop}$ *et* $\Gamma \vdash Q : \mathsf{prop}$, *alors*

- $\Gamma \vdash_{\mathcal{T}^{++^2}} \forall c : \lfloor P\rfloor \times \lfloor Q\rfloor.c \Vdash P \wedge Q \to (\mathsf{proj}^1[\lfloor P\rfloor, \lfloor Q\rfloor]\, c) \Vdash P \wedge (\mathsf{proj}^2[\lfloor P\rfloor, \lfloor Q\rfloor]\, c) \Vdash Q$,
- $\Gamma \vdash_{\mathcal{T}^{++^2}} \forall x : \lfloor P\rfloor.x \Vdash P \to \forall y : \lfloor Q\rfloor.y \Vdash Q \to (\mathsf{pair}[\lfloor P\rfloor, \lfloor Q\rfloor]\, x\, y) \Vdash P \wedge Q$

Démonstration – La preuve s'effectue en détruisant l'inductif $c \Vdash P \wedge Q$ et montre que c est de la forme $\mathsf{pair}[\lfloor P\rfloor, \lfloor Q\rfloor]\, x\, y$ avec $x \Vdash P$ et $y \Vdash Q$. Puis par les réductions

$$\mathsf{proj}^1[\lfloor P\rfloor, \lfloor Q\rfloor]\, (\mathsf{pair}[\lfloor P\rfloor, \lfloor Q\rfloor]\, x\, y) \trianglerighteq x \quad \text{et} \quad \mathsf{proj}^2[\lfloor P\rfloor, \lfloor Q\rfloor]\, (\mathsf{pair}[\lfloor P\rfloor, \lfloor Q\rfloor]\, x\, y) \trianglerighteq y$$

on se ramène à montrer $x \Vdash P$ et $y \Vdash Q$; ce que l'on a par hypothèse.

– On utilise pour cela le constructeur conj_R pour conclure directement. ☺

De façon analogue et sans plus de difficulté, on peut démontrer l'équivalent du lemme précédent pour le connecteur existentiel.

4.3.4 Lemme
Si $\Gamma, x : \tau \vdash P : \mathsf{prop}$*, alors*
– $\Gamma \vdash_{\mathcal{T}^{++2}} \forall r : (\!|\lfloor P \rfloor|\!).r \Vdash (\exists x : \tau.P) \quad \rightarrow \quad \exists x : \tau.(\mathsf{open}[\lfloor P \rfloor]\, r) \Vdash P$
– $\Gamma \vdash_{\mathcal{T}^{++2}} \forall x : \tau, r : \lfloor P \rfloor.r \Vdash P \quad \rightarrow \quad (\mathsf{close}[\lfloor P \rfloor]\, r) \Vdash \exists x : \tau.P$

On peut également appliquer la transformation de réalisabilité aux termes de premier niveau, à condition de leur appliquer la fonction de soulèvement au préalable (la transformation de paramétricité dans notre prochain chapitre).

Par exemple, on peut appliquer la transformation de réalisabilité à l'inductif $\lceil \mathsf{nat} \rceil$ que l'on a précédemment vu avec la syntaxe de Coq. Cet inductif est défini par la déclaration suivante :

$$\mathtt{Ind}(\lceil \mathsf{nat} \rceil : \lceil \star \rceil, \lceil \mathsf{zero} \rceil : \lceil \mathsf{nat} \rceil, \lceil \mathsf{succ} \rceil : \lceil \mathsf{nat} \rceil \rightarrow \lceil \mathsf{nat} \rceil)$$

Si on lui applique la transformation de réalisabilité, on obtient l'inductif nat_ind :

$$\mathtt{Ind}(\lceil \mathsf{nat} \rceil_R : \mathsf{nat} \rightarrow \lceil \star \rceil, \lceil \mathsf{zero} \rceil_R : \lceil \mathsf{nat} \rceil_R\, \mathsf{zero}, \lceil \mathsf{succ} \rceil_R : \forall y : \mathsf{nat}.\lceil \mathsf{nat} \rceil_R\, y \rightarrow \lceil \mathsf{nat} \rceil_R\, (\mathsf{succ}\, y))$$

Ainsi on observe que la notation $x \in \mathbb{N}$ signifie que x appartient aux réalisateurs de $\lceil \mathsf{nat} \rceil$.

On applique une seconde fois la transformation de réalisabilité afin d'obtenir la relation $x \Vdash (y \Vdash \lceil \mathsf{nat} \rceil)$ que l'on a noté $x \sim_{\mathsf{nat}} y$ dans la section précédente. Ainsi, ces trois notations désignent trois formules syntaxiquement égales :

$$x \Vdash (y \Vdash \lceil \mathsf{nat} \rceil) \quad = \quad x \Vdash y \in \mathbb{N} \quad = \quad x \sim_{\mathsf{nat}} y$$

L'inductif généré est le suivant :

```
Inductive nat_param : nat→ nat→ Prop :=
  | zero_nat_param : nat_param zero zero
  | succ_nat_param : forall y y' : nat, nat_param y y'→
     nat_param (succ y) (succ y').
```

On peut montrer le lemme suivant :

4.3.5 Lemme
On a $\vdash_{\mathcal{T}^{++2}} \mathsf{datatype}$*, c'est-à-dire :*

$$\vdash_{\mathcal{T}^{++2}} \forall x\, y : \mathsf{nat}.x \sim_{\mathsf{nat}} y \leftrightarrow x =_{\mathsf{nat}} y \wedge x \in \mathbb{N}$$

Démonstration
– $\Rightarrow$: On procède en utilisant le principe d'induction associée à l'hypothèse $x \sim_{\mathsf{nat}} y$ pour se ramener à démontrer les trivialités suivantes :
– $\overline{0} =_{\mathsf{nat}} \overline{0} \wedge \overline{0} \in \mathbb{N}$,
– $\forall y\, y' : \mathsf{nat}, y =_{\mathsf{nat}} y' \wedge y \in \mathbb{N} \quad \rightarrow \quad \mathsf{S}\, y =_{\mathsf{nat}} \mathsf{S}\, y' \wedge (\mathsf{S}\, y) \in \mathbb{N}$.
– $\Leftarrow$: On utilise pour cela le principe d'induction associé à $x \in \mathbb{N}$ qui nous ramène à démontrer exactement le type des constructeurs de nat_param. ☺

On constate ici un avantage majeur de la représentation des entiers par les types inductifs : il n'est pas nécessaire d'admettre des axiomes pour déduire la propriété datatype qui est une des hypothèses du théorème de représentation 4.2.58.

Une des conséquences du lemme précédent est que le prédicat $x \in \mathbb{N}$ est honnête et auto-réalisé. Plus généralement, on démontre sans difficulté le lemme suivant :

4.3.6 Lemme
Le système $\mathcal{T}^{++^2}$ est adéquat pour la réalisabilité :

$$\mathcal{T}^{++^2} \models \text{REALIZABILITY}$$

Si on admet que le théorème d'adéquation de la réalisabilité 4.1.8 s'étend au CTSI (on rappelle qu'on verra une esquisse de sa démonstration dans la sous-section suivante), alors on peut démontrer le théorème 4.2.30 pour les CTSI (sans changer son énoncé) ainsi que les quatres lemmes et théorèmes ci-dessous. Les preuves sont similaires à ceci près qu'il n'est pas nécessaire de supposer datatype puisque nous avons vu qu'il est prouvable dans $\mathcal{T}^{++^2}$. On notera que le lemme 4.3.7 et le théorème 4.3.9 ne nécessitent pas de disposer de l'ensemble du système de notations ce qui implique qu'ils sont vrais pour tous les systèmes qui contiennent $\mathcal{T}$ (et non $\mathcal{T}^{++}$).

4.3.7 Lemme
Supposons $\mathcal{T}^2 \subseteq \mathcal{P}^2$. Si

$$\vdash_{\mathcal{P}^2} t : \overbrace{\mathsf{nat} \to \cdots \to \mathsf{nat}}^{n \text{ fois}} \to \mathsf{nat}$$

alors t est prouvablement inductif dans $\mathcal{P}^2$ (sous le contexte vide).

4.3.8 Lemme (Représentable implique prouvablement totale)
Supposons $\mathcal{T}^{++^2} \subseteq \mathcal{P}^2$ tel que $\mathcal{P}^2 \models$ NONDEGENERATE $\wedge$ BIJECTIVE. Si $f : \mathbb{N} \to \mathbb{N}$ est représentée par un terme t_f de $\mathcal{P}$, alors f est prouvablement totale dans $\mathcal{P}^2$ (sous le contexte vide).

4.3.9 Théorème (Programmation par preuve)
Soit $\mathcal{P}$ tel que $\mathcal{T}^2 \subseteq \mathcal{P}^2$.
Soit Γ un contexte uniformément auto-réalisé clos au second niveau.
On a $\Gamma \vdash_{\mathcal{P}^2} \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}.(t\, x_1 \ldots x_n) \in \mathbb{N}$, si et seulement il existe un terme p tel que

$$\vdash_{\mathcal{P}} p : \mathsf{nat} \to \ldots \to \mathsf{nat} \quad \text{et} \quad \Gamma \vdash_{\mathcal{P}^2} \forall x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N}.(t\, x_1 \ldots x_n) =_{\mathsf{nat}} (p\, x_1 \ldots x_n)$$

4.3.10 Théorème (Théorème de représentation)
Soit $\mathcal{P}$ tel que $\mathcal{T}^{++^2} \subseteq \mathcal{P}^2$ et $\mathcal{P}^2 \models$ BIJECTIVE. Et soit Γ un contexte de $\mathcal{P}^2$ uniformément auto-réalisé, clos au second niveau et non-dégénéré tel que $\Gamma \vdash_{\mathcal{P}^2}$ ext_nat.
Alors, $f : \mathbb{N} \to \mathbb{N}$ est prouvablement totale dans Γ si et seulement si f est représentable.

Nous discuterons d'une application du théorème 4.3.9 dans la sous-section suivante.

Le théorème de représentation semble particulièrement intéressant lorsqu'on l'applique aux systèmes qui sont entre $\mathcal{T}$ et $I\lambda$. À savoir les systèmes obtenus en ajoutant de nouveaux inductifs à $\mathcal{T}$. En effet, chaque définition inductive autorisée dans $\mathcal{P} \supseteq \mathcal{T}$ augmente (pas toujours strictement) le pouvoir expressif de $\mathcal{P}^2$.

Ainsi, on note $\mathcal{T}_\Omega$ le système obtenu en ajoutant à $\mathcal{T}^{++}$ l'inductif ci-dessous, déjà vu page 131, qui permet de représenter des ordinaux strictement plus grands que ε_0 :

$$\begin{array}{ll} \texttt{Ind}\,(\,\mathsf{ord\_brouwer} : \texttt{Set}, & \mathsf{zero\_brouwer} : \mathsf{ord\_brouwer}, \\ & \mathsf{succ\_brouwer} : \mathsf{ord\_brouwer} \to \mathsf{ord\_brouwer}, \\ & \mathsf{sup\_brouwer} : (\mathsf{nat} \to \mathsf{ord\_brouwer}) \to \mathsf{ord\_brouwer}\,) \end{array}$$

Alors le prédicat de $(\mathcal{T}_\Omega)^2$ suivant

$$\lambda x : \mathsf{ord\_brouwer}\,.x \Vdash \lceil \mathsf{ord\_brouwer} \rceil$$

nous donne accès à un principe d'induction transfini qui dépasse l'ordinal ε_0.

La terminaison des suites de Goodstein dans $\mathcal{T}_\Omega{}^2$. Le théorème d'incomplétude de Gödel affirme qu'une théorie qui contient l'arithmétique ne peut pas prouver sa propre cohérence. Ainsi, si on admet la cohérence de l'arithmétique, le théorème d'incomplétude prouve l'existence d'une formule "vraie" et non prouvable. Par la suite, d'autres énoncés indépendants de l'arithmétique ont été étudiés ; nous allons discuter dans ce paragraphe de l'un d'entre eux : la terminaison des suites de Goodstein. L'énoncé de la terminaison des suites de Goodstein est souvent informellement présenté comme étant de nature plus arithmétique que l'énoncé de la cohérence d'une théorie. Il a ainsi pu être utilisé pour illustrer le fait qu'il existait des énoncés "vrais" et non démontrables en dehors des énoncés "purement logiques" ; c'est-à-dire que l'incomplétude toucherait également les énoncés des "vraies mathématiques".

Les suites de Goodstein sont définies de la façon suivante : étant donné un entier $m \in \mathbb{N}$, on définit la suite $g_m(0),\ldots, g_m(n), \ldots$ par les équations suivantes :

$$\begin{aligned} g_m(0) &= m \\ g_m(n+1) &= \mathcal{G}_n(g_m(n)) \end{aligned}$$

où $\mathcal{G}_b : \mathbb{N} \to \mathbb{N}$ est une fonction primitive récursive obtenue de la façon suivante : soit $n \in \mathbb{N}$, on effectue la *décomposition héréditaire en base b de n* ; c'est-à-dire en décomposant l'entier n en base b, puis en décomposant en base b les exposants de la décomposition précédente, puis en décomposant en base b les exposants de la décomposition en base b des exposants des exposants, *etc...* Par exemple, l'entier 59780 se décompose héréditairement en base 3 de la façon suivante :

$$59780 = 3^{10} + 3^6 + 2 = 3^{3^2+1} + 3^{2\times 3} + 2$$

$\mathcal{G}_b(n)$ est alors obtenu en remplaçant tous les b par $b+1$ dans la décomposition héréditaire, puis en soustrayant 1. Ainsi :

$$\mathcal{G}_3(59780) = 4^{4^2+1} + 4^{2\times 4} + 2 - 1 = 17179934721$$

Le théorème de Goodstein [Goo44], affirme que pour tout entier m la suite g_m est ultimement constante égale à 0. L'auteur démontre que la suite atteint 0 grâce à une induction ordinale jusqu'à ε_0.

L'énnoncé de ce théorème peut s'exprimer dans tout système muni de notations pour l'arithmétique par la formule suivante :

$$\forall m \in \mathbb{N}.\exists n \in \mathbb{N}.\overline{g}\,m\,n =_{\mathsf{nat}} \overline{0}$$

où $\overline{g}$ désigne un terme qui représente la fonction $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

Plus tard, Laurie Kirby et Jeff Paris ont démontré [KP82] que le théorème de Goodstein était indépendant de **PA**. On peut en déduire qu'il n'existe pas de terme de $\mathcal{T}$ qui représente la fonction qui à m associe l'entier n tel que $g_m(n) = 0$. On en déduit donc que :

4.3.11 Lemme

$$\forall\!\!\!/_{\mathcal{T}^{++^2}}\, \forall m \in \mathbb{N}.\exists n \in \mathbb{N}.\overline{g}\, m\, n =_{\mathsf{nat}} \overline{0}$$

Un exercice intéressant pourrait être celui de formaliser dans ${\mathcal{T}_\Omega}^2$ une preuve de $\forall m \in \mathbb{N}.\exists n \in \mathbb{N}.\overline{g}\, m\, n =_{\mathsf{nat}} \overline{0}$. Cela permettrait d'obtenir par extraction un terme de $\mathcal{T}_\Omega$ qui représente une fonction qui n'est pas représentable dans $\mathcal{T}$. On pourrait alors en déduire que le système ${\mathcal{T}_\Omega}^2$ n'est pas une extension conservative de $\mathcal{T}^2$.

4.3.2 Programmation par preuve dans les systèmes qui contiennent $\mathcal{T}$

Le théorème 4.3.9 implique que la méthodologie de programmation expliquée dans la section précédente peut être étendue aux systèmes qui contiennent $\mathcal{T}$, c'est-à-dire aux systèmes qui utilisent les inductifs pour représenter leurs données.

Dans ce paragraphe, nous allons voir que l'on peut utiliser le système d'extraction de Coq pour mettre en œuvre ce paradigme. Pour cela, il faut aller à rebours de la philosophie de l'extraction de Coq. En effet, le principe de la "programmation par preuve" consiste à placer le contenu calculatoire non pas dans les programmes mais dans les preuves ; tandis que dans le système Coq, c'est l'inverse qui est encouragé : le rôle du mécanisme d'extraction est d'épurer les programmes de tout contenu logique.

Ainsi, l'astuce est d'inverser le rôle de `Prop` et `Set` : nous allons utiliser `Set` pour typer les formules et `Prop` pour typer les types des programmes. Nous observerons donc $\mathcal{T}^2$ au travers du morphisme suivant :

$$\begin{aligned} \star &\mapsto \texttt{Prop} \\ \lceil\star\rceil &\mapsto \texttt{Set} \\ \Box &\mapsto \texttt{Type} \\ \lceil\Box\rceil &\mapsto \texttt{Type} \end{aligned}$$

En guise d'illustration, nous allons maintenant décrire une session Coq dans laquelle nous utiliserons Coq pour programmer avec les preuves. On commence avec les définitions suivantes :

```
Inductive nat : Prop :=
  | O : nat
  | S : nat→ nat.

Inductive ind_nat : nat→ Set :=
  | ind_O : ind_nat O
  | ind_S : forall x, ind_nat x→ ind_nat (S x).
Notation "x∈'ℕ'" := (ind_nat x) (at level 40).
```

Elles définissent un nouveau type de donnée `nat` ainsi que le prédicat `ind_nat` qui sera noté $x \in \mathbb{N}$.

La commande suivante permet de renommer les constructeurs ainsi que l'identifiant type extrait de `nat_ind` (ce qui rendra les programmes extraits plus lisibles) :

```
Extract Inductive ind_nat  ⇒  "nat" [ "O" "S" ].
```

On définit ensuite l'égalité que nous allons utiliser. L'égalité étant une formule, son arité est dans `Set` :

```
Inductive eq_nat (x : nat) : nat→ Set :=
  | refl : eq_nat x x.

Notation "x==y" := (eq_nat x y) (at level 70).
```

Par la suite, on donne la liste des axiomes qui nous serviront à prouver les fonctions que l'on souhaite implémenter. Ces axiomes sont choisis de façon à spécifier complètement les fonctions l'on veut implémenter (ici la fonction prédecesseur, l'addition, la multiplication et la fonction de division par deux). On voit bien que ces axiomes décrivent un système de réécriture orthogonal ; l'auteur de ces énoncés avait probablement en tête une implémentation particulière de ces fonctions.

Il faut également prendre garde à ce que ces axiomes ne nuisent pas à la cohérence du système (car on aurait pu implémenter ces fonctions directement par des termes Coq).

```
Axiom pred : nat→ nat.
Axiom pred_O : pred O == O.
Axiom pred_S : forall x, pred (S x) == x.


Axiom plus : nat→ nat→ nat.
Axiom plus_O : forall x, plus O x == x.
Axiom plus_S : forall x y, plus (S x) y == S (plus x y).

Axiom mult : nat→ nat→ nat.
Axiom mult_O : forall x, mult O x == O.
Axiom mult_S : forall x y, mult (S x) y == plus (mult x y) y.

Axiom half : nat→ nat.
Axiom half_O : half O == O.
Axiom half_1 : half (S O) == O.
Axiom half_S : forall x, half (S (S x)) == S (half x).

Hint Rewrite pred_O pred_S plus_O plus_S mult_O mult_S half_O
   half_1 half_S : spec_db.
```

La dernière ligne permet d'enregistrer la liste de ces axiomes dans une base de donnée nommée `spec` qui nous permettra d'invoquer des tactiques automatiques pour réécrire dans les formules.

On renseigne également une seconde base de donnée nommée `ind_db` qui contiendra les preuves des fonctions dont on a déjà prouvé qu'elles étaient prouvablement inductives. La base `ind_db` est donc initialisée avec les constructeurs de l'inductif `ind_nat` qui prouvent que `O` et la fonction `S` sont prouvablement inductifs.

```
Hint Resolve ind_O ind_S : ind_db.
```

On définit ensuite une *tactique* (un petit programme écrit dans le langage spécialisé de Coq) qui nous permet d'implémenter une procédure de recherche de preuves automatique. Il n'est pas utile pour lire la suite de comprendre précisément son fonctionnement. Elle procède en inspectant le contexte à la recherche d'un principe d'induction (c'est-à-dire un élément du contexte de la forme "_ ∈ ℕ"), puis elle cherche à l'appliquer pour démontrer le but courant. Si cela est possible elle démontre le cas de base et l'hérédité en réécrivant le but grâce aux règles de la base de donnée `spec_db` et tente de conclure en utilisant les lemmes de la base de donné `ind_db`. Dans le cas, où elle ne parvient pas à résoudre les buts, la procédure se réexécute récursivement jusqu'à une profondeur de trois appels récursifs. Enfin, à chaque étape, si elle ne parvient pas à conclure en utilisant un principe d'induction, elle essaye le suivant. Cette procédure implémente donc une recherche de preuves rudimentaire "en profondeur d'abord" ; ce qui –comme nous allons le voir– n'est pas idéal puisque elle n'effectue pas sa recherche en énumérant les preuves par taille croissante (elle peut retourner une preuve obtenue à profondeur 3 même s'il en existe une à profondeur 1).

```
Ltac bruteforce :=
  let rec aux n :=
    match n with Datatypes.S ?n' ⇒
      match goal with
        [ H : _ ∈ ℕ ⊢ _] ⇒
         induction H; intros;
         autorewrite with spec_db in *;
         eauto with ind_db; aux n'
      end
    end
  in intuition; aux 3.
```

On peut maintenant utiliser cette tactique automatique afin de tenter de prouver que les fonctions à programmer sont prouvablement inductives. Nous commençons par prouver que la fonction prédécesseur est prouvablement inductive :

```
Lemma ind_pred :
  forall x, x ∈ ℕ → (pred x) ∈ ℕ.
Proof.
  bruteforce.
Defined.
Hint Resolve ind_pred : ind_db.
```

La tactique `bruteforce` parvient à démontrer automatiquement que la fonction prédécesseur est prouvablement inductive. On utilise alors la commande `Defined.` pour signifier au système que le terme de preuve est significatif. Enfin, on rajoute le lemme à la base de donnée de `ind_db`.

On peut maintenant demander au système d'extraire le contenu calculatoire grâce à la commande suivante :

```
Extraction ind_pred.
```

La mécanisme d'extraction génère alors le programme OCaml suivant :

```
(** val ind_pred : nat -> nat **)
```

```
let ind_pred = function
| O -> internal_eq_nat_rew_r O pred_O
| S i -> internal_eq_nat_rew_r i (pred_S __)
```

Ainsi que l'avertissement suivant :

```
Warning: The following axioms must be realized in the extracted code:
 pred_O pred_S.
```

Puisque les axiomes sont informatifs, Coq nous demande de fournir un réalisateur. Or, le terme[7] `internal_eq_nat_rew_r` est un terme généré par coq qui est intercalé à chaque fois que la tactique de réécriture `rewrite` est utilisée. On peut demander au système d'afficher le contenu calculatoire de ce terme avec la commande :

```
Extraction internal_eq_nat_rew_r.
```

Le système produit le terme suivant :

```
let internal_eq_nat_rew_r hC h = hC
```

Comme on peut le constater, ce terme est une identité qui "oublie" son argument `h`, correspondant à la preuve de l'égalité utilisée pour l'étape de réécriture.

Le mécanisme d'extraction de Coq offre la possibilité de déplier automatiquement ("d'inliner") certains programmes extraits. On peut utiliser ce dispositif pour faire disparaître ce terme qui pollue nos programmes. Ceci a pour conséquence d'éliminer également les utilisations des axiomes égalitaires (nous n'aurons donc pas besoin de leur trouver un réalisateur). On utilise pour ce faire la commande suivante :

```
Extraction Inline eq_nat_rew_r.
```

Le programme extrait optimisé est alors le suivant :

```
(** val ind_pred : nat -> nat **)
let ind_pred = function
| O -> O
| S i -> i
```

On a donc ici obtenu –par recherche de preuves– la fonction que l'on aurait pu programmer "naturellement". On constate bien qu'il implémente la fonction prédécesseur.

On peut faire de même avec la fonction `plus` et la fonction `mult` :

```
Lemma ind_plus :
  forall x y, x ∈ ℕ→ y ∈ ℕ→ (plus x y) ∈ ℕ.
Proof.
  bruteforce.
Defined.
Hint Resolve ind_plus : ind_db.
Lemma ind_mult :
```

7. On utilise ici la version 8.4 de Coq. Le lecteur qui voudrait tester ce script dans une version antérieure devra remplacer toutes les occurrences de `internal_eq_nat_rew_r` par `eq_nat_rew_r` ; en effet, l'identifiant généré par le système a changé entre la version 8.3 et la version 8.4.

```
let rec ind_plus h = function
| O ->
  let rec f = function
  | O -> O
  | S i0 -> S (f i0)
  in f h
| S i ->
  let rec f = function
  | O ->
    let rec f0 = function
    | O -> S i
    | S i2 -> S (f0 i2)
    in f0 h
  | S i1 -> f i1
  in f (ind_plus h i)
```

```
let rec ind_mult h = function
| O ->
  let rec f = function
  | O -> O
  | S i0 -> ind_plus (f i0) O
  in f h
| S i ->
  let rec f = function
  | O ->
    let rec f0 = function
    | O -> O
    | S i2 -> ind_plus (f0 i2) (S i)
    in f0 h
  | S i1 -> f i1
  in f (ind_mult h i)
```

FIGURE 4.3 – Extraction des fonctions `plus` et `mult`.

```
  forall x y, x ∈ ℕ→ y ∈ ℕ→ (mult x y) ∈ ℕ.
Proof.
  bruteforce.
Defined.
Hint Resolve ind_mult : ind_db.
```

La procédure de recherche de preuves parvient à prouver automatiquement ces deux lemmes. On peut extraire le résultat grâce aux commandes suivantes :

```
Extraction ind_plus.
Extraction ind_mult.
```

Le résultat de l'extraction est donné par la figure 4.3. On remarque alors que le programme produit est particulièrement compliqué : certaines inductions, inutiles dans la preuve, génèrent des sous-programmes qui implémentent des identités (par exemple la fonction locale `f` de la première branche de `ind_plus` implémente l'identité de façon coûteuse). Néanmoins, la théorie nous garantit sa correction.

Il nous reste à démontrer que `half` est prouvablement inductive. Comme on le constate dans le script ci-dessous, la tactique `bruteforce` ne permet pas de résoudre le but. En effet, cette procédure automatique n'est pas capable de généraliser les buts lorsqu'elle essaye d'appliquer un principe d'induction. C'est pourquoi nous sommes obligés de prouver manuellement le lemme.

L'idée de la preuve est d'introduire la coupure "$x \in \mathbb{N} \wedge (\mathsf{S}\, x) \in \mathbb{N}$" grâce à la tactique `cut`. Ainsi, l'hypothèse d'induction est suffisamment générale pour en déduire facilement le lemme. Mais avant d'effectuer la coupure, il est nécessaire de définir la conjonction.

```
Lemma ind_half :
  forall x, x ∈ ℕ→ (half x) ∈ ℕ.
try bruteforce. (* ne parvient pas à résoudre le but *)
```

```
Inductive and (X : Set) (Y : Set) :=
  | conj : X→ Y→ and X Y.
Notation "X∧Y" := (and X Y) (at level 60).
(* On donne les notations pour extraire les couples en caml: *)
Extract Inductive and ⇒ "(*)" [ "(,)" ].

intros.
(* On fournit la coupure suivante: *)
cut (half x ∈ ℕ ∧ half (S x) ∈ ℕ).
(* La coupure implique trivialement le résultat: *)
intuition.
(* Le motif [| ? ? [? ?]] permet de détruire la conjonction dans
   l'hypothèse d'induction: *)
induction H as [| ? ? [? ?]];
  autorewrite with spec_db; split; auto with ind_db.
Defined.
```

Le terme généré par la preuve de `ind_half` est le suivant :

```
(** val ind_half : nat -> nat **)
let ind_half h =
  let h0 =
    let rec f = function
    | O -> O,O
    | S i0 -> let h1,h2 = f i0 in h2,(S h1)
    in f h
  in
  let x,x0 = h0 in x
```

On voit que ce programme calcule par récursion sur `n` le couple $(\mathsf{half}\ n, \mathsf{half}\ (\mathsf{S}\ n))$ en suivant les équations de récurrence suivantes :

$$\begin{aligned} f(0) &= (0,0) \\ f(n+1) &= (y, Sx) \text{ où } (x,y) = f(n) \end{aligned}$$

Ce programme est satisfaisant du point de vue de l'efficacité, mais nous savons qu'il existe un programme plus simple qui implémente la division par deux. La question que l'on pourrait donc se poser est : existe-t-il un preuve `ind_half_opt` que `half` est prouvablement inductive dont on pourrait extraire le programme ci-dessous ?

```
(** val ind_half_opt : nat -> nat **)
let rec ind_half_opt = function
| O -> O
| S hp ->
  (match hp with
   | O -> O
```

```
    | S hq -> S (ind_half_opt hq))
```

La réponse est oui ; et pour cela nous allons "tricher" et utiliser la preuve de la réciproque du théorème 4.3.9. Nous allons voir que si le programme généré est certainement le plus "naturel" pour implémenter cette fonction, la preuve elle ne le sera pas.

Nous savons que le programme `half` est implémentable par le terme de $\mathcal{T}^2$ suivant :

```
Fixpoint cheat_half (n : nat) :=
  match n with
   | O  ⇒  O
   | S O  ⇒  O
   | S (S p)  ⇒  S (cheat_half p)
  end.
```

Ce programme est témoin que la fonction de division par deux est représentable dans $\mathcal{T}^{++^2}$. Nous allons donc maintenant utiliser la preuve du lemme 4.3.7 pour montrer que ce terme est prouvablement inductif. Nous rappelons que la preuve procède en appliquant la transformation de réalisabilité au soulèvement de cheat_half :

$$\|\lceil \mathsf{cheat\_half} \rceil\| : \forall n : \mathsf{nat}. n \in \mathbb{N} \to (\mathsf{cheat\_half}\, n) \in \mathbb{N}$$

Nous allons donc calculer (à la main !) le terme $\|\lceil \mathsf{cheat\_half} \rceil\|$. On obtient :

```
Fixpoint cheat_half_ind (n : nat) (h : n ∈ ℕ)
            : (cheat_half n) ∈ ℕ :=
  match h with
   | ind_O  ⇒  ind_O
   | ind_S p Hp  ⇒
      match Hp with
        | ind_O  ⇒  ind_O
        | ind_S q Hq  ⇒
           ind_S (cheat_half q) (cheat_half_ind q Hq)
      end
  end.
Extraction Inline cheat_half_ind.
```

On demande au système d'inliner cette fonction (on ne veut pas montrer que l'on a triché). On peut constater la similarité de la structure de `cheat_half_ind` avec celle de `cheat_half`. Cela tient au fait que :

$$\lfloor \|\lceil \mathsf{cheat\_half} \rceil\| \rfloor = \mathsf{cheat\_half}$$

Rester à prouver que `cheat_half` est égal à `half` :

```
Lemma half_eq_cheat_half :
  forall x, x ∈ ℕ→ half x == cheat_half x.
Proof.
  intros x H.
  cut ((half (S x) == cheat_half (S x)) ∧ (half x == cheat_half x)).
```

```
  intuition.
  induction H as [| ? ? [H1 H2]];
  split; autorewrite with spec_db;
    try rewrite H2; try rewrite H1; reflexivity.
Defined.
```

On constate que la preuve est similaire à celle du lemme `ind_half` à la différence prêt que la conclusion est une égalité (elle disparaîtra donc lors de "l'inlinage" des étapes de réécritures).

On peut maintenant conclure facilement en utilisant les deux lemmes précédents :

```
Lemma ind_half_opt :
  forall x, x ∈ ℕ→ (half x) ∈ ℕ.
Proof.
  intros.
  rewrite half_eq_cheat_half.
  apply cheat_half_ind; assumption.
  assumption.
Defined.
```

Et on peut alors vérifier que l'extraction appliquer à `ind_half_opt` fournit bien le programme OCaml que l'on attendait.

Dans cette section, nous avons constaté les limites de cette méthodologie de programmation :

- Les spécifications paraphrasent le programme que l'on souhaite implémenter.
- La recherche de preuves –lorsqu'elle fonctionne– peut fabriquer des programmes particulièrement inefficaces.
- Il semble compliqué de mécaniser la recherche de preuves lorsque l'on a à faire à des spécifications non-triviales : il faut des "idées" pour trouver les coupures nécessaires.
- Enfin, on prouve difficilement les lemmes demandés sans avoir en tête une idée de l'implémentation de la fonction à programmer.

4.3.3 Logique engendrée par un CTSI

Dans la section précédente, nous avons vu, par l'exemple, comment on pouvait étendre la construction du chapitre 3 aux types inductifs. Nous allons ici étendre de façon formelle la transformation de réalisabilité aux inductifs.

Le but de cette section sera de convaincre le lecteur que le théorème d'adéquation de la réalisabilité s'étend au cas des inductifs, nous donnerons une preuve complète de ce théorème mais nous admettrons néanmoins quelques résultats métathéoriques sur les CTSI (comme par exemple le fait que l'on peut annoter les variables avec leur niveau afin d'étendre la fonction lvl(·)).

4.3.12 Définition (CTSI pré-engendré)
Soit $\mathcal{P}$ un CTSI. On dit d'un CTSI bien-formé $\mathcal{P}'$ qu'il est un système pré-engendrée par $\mathcal{P}$ si

$$\begin{array}{rcl} \mathcal{S}_{\mathcal{P}'} & = & \mathcal{S}_{\mathcal{P}^2} \\ \mathcal{A}_{\mathcal{P}'} & = & \mathcal{A}_{\mathcal{P}^2} \\ \mathcal{R}_{\mathcal{P}'} & = & \mathcal{R}_{\mathcal{P}^2} \\ \mathcal{I}_{\mathcal{P}} & \subseteq & \mathcal{I}_{\mathcal{P}'} \\ \mathcal{E}_{\mathcal{P}} & \subseteq & \mathcal{E}_{\mathcal{P}'} \end{array}$$

et si $(I, s) \in \mathcal{E}_{\mathcal{P}'}$ implique $\mathrm{lvl}(I) = \mathrm{lvl}(s)$ où $\mathrm{lvl}(I)$ désigne le niveau d'un inductif I de sorte r (on rappelle que la sorte d'un inductif désigne la sorte qui apparaît à la conclusion de son arité).

CTSI annoté. Soit $\mathcal{P}'$ est un CTSI pré-engendré par $\mathcal{P}$.

On annote les variables des expressions de $\mathcal{P}'$ et on étend la fonction de niveau $\mathrm{lvl}(\cdot)$ de la façon suivante :

$$\begin{array}{rcl} \mathrm{lvl}(I[\overrightarrow{P}]) & = & \mathrm{lvl}(I) \\ \mathrm{lvl}(c[\overrightarrow{P}]) & = & \mathrm{lvl}(I) \text{ (où } c \text{ est un constructeur de } I) \\ \mathrm{lvl}(\mathtt{case}_I(M, \overrightarrow{P}, \lambda y^a : B, i^{a'} : C.T, \overrightarrow{\overrightarrow{\lambda z^{a''} : E.F}})) & = & \mathrm{lvl}(I) \\ \mathrm{lvl}(\mathtt{fix}\, f^a : A, \overrightarrow{x^{a'} : B}^{l+1}.M) & = & \mathrm{lvl}(M) \end{array}$$

De façon similaire à la section 3.3, on contraint les réductions pour qu'elles préservent les annotations de la façon suivante :

– Si $a''_{1,j} = \mathrm{lvl}(M_1), \ldots, a''_{m_j,j} = \mathrm{lvl}(M_j)$, alors :

$$\mathtt{case}_I(c_j[\overrightarrow{P}^p]\, \overrightarrow{M}^{m_j}, \overrightarrow{Q}^p, \lambda \overrightarrow{y^a : B}^n, i^{a'} : C.T, \overrightarrow{\overrightarrow{\lambda z^{a''} : E}^m.F}^k) \rhd^+ F_j[\overrightarrow{M}^{m_j}/\overrightarrow{z}^{m_j}]$$

– Si $\mathrm{lvl}(N_i) = \mathrm{lvl}(B_i)$ pour $1 \leq i \leq k$, alors

$$((\mathtt{fix}\, f : A, \overrightarrow{x : B}^{l+1}.M)\quad \overrightarrow{N}^{l+1}) \rhd^+ M[\mathtt{fix}\, f : A, \overrightarrow{x : B}^{l+1}.M/f, \overrightarrow{N}^{l+1}/\overrightarrow{x}^{l+1}]$$

On peut alors montrer de façon similaire au lemme 3.3.7 que :

4.3.13 Lemme
Si $M \preccurlyeq^+ N$, *alors* $\mathrm{lvl}(M) = \mathrm{lvl}(N)$

Le fait que les systèmes pré-engendrés ne contiennent pas d'élimination autorisée entre un inductif et une sorte de premier niveau est essentiel pour vérifier que :

$$\mathrm{lvl}(\mathtt{case}_I(c_j[\overrightarrow{P}^p]\, \overrightarrow{M}^{m_j}, \overrightarrow{Q}^p, \lambda \overrightarrow{y^a : B}^n, i^{a'} : C.T, \overrightarrow{\overrightarrow{\lambda z^{a''} : E}^m.F}^k)) = \mathrm{lvl}(F_j[\overrightarrow{M}^{m_j}/\overrightarrow{z}^{m_j}])$$

On admettra qu'il est possible d'introduire une relation de typage $\vdash^+_{\mathcal{P}'}$ de façon similaire au système de la sous-section 3.3 : la définition de cette relation de dérivabilité est obtenue en rajoutant les annotations sur les variables et un "+" sur le symbole de dérivation dans les règles Case, Point-Fixe, Inductif, Constructeur (seules les règles Variable et Affaiblissement sont renforcées par des conditions sur les niveaux).

Nous admettrons que les démonstrations de la section 3.3 s'adaptent aux inductifs et que l'on disposera, en particulier, des deux lemmes suivants :

4.3.14 Lemme
Si $\Gamma \vdash^+_{\mathcal{P}'} M : T$, *alors* $\mathrm{lvl}(M) = \mathrm{lvl}(T)$.

4.3.15 Lemme
On a $\Gamma \vdash_{\mathcal{P}'} M : T$ *si et seulement s'il existe* Γ^+, M^+ *et* T^+ *tels que* $\Gamma^+ \vdash^+_{\mathcal{P}'} M^+ : T^+$ *avec* $|\Gamma^+| = \Gamma$, $|M^+| = M$ *et* $|T^+| = T$.
Où $|\cdot|$ *est la fonction d'oubli des annotations.*

Tout comme dans le cas des CTS, on s'autorise à n'écrire les annotations que lorsqu'elles sont significatives et à écrire $\vdash$ à la place de $\vdash^+$.

Projection. La définition 2.1.3 induit un ordre[8] $I_1, I_2, \ldots$ sur les inductifs d'un CTSI bien-formé $\mathcal{P}$ tel que

$$\mathcal{P} = \mathrm{Cts}(\mathcal{P}) + \mathtt{Ind}(\overrightarrow{x : Q^1}^{p^1}, I_1 : A^1, \overrightarrow{c^1 : C^1}^{k^1}) + \mathtt{Ind}(\overrightarrow{x : Q^2}^{p^2}, I_2 : A^2, \overrightarrow{c^2 : C^2}^{k^2}) + \cdots$$

où $\mathrm{Cts}(\mathcal{P})$ est le CTS sous-jacent à $\mathcal{P}$, "+" désigne la notation de la définition 2.1.2, $\overrightarrow{x : Q^i}^{p^i}$, A^i, $\overrightarrow{c^i : C^i}^{k^i}$ désignent respectivement les paramètres, l'arité et les constructeurs de I^i. Cet ordre correspond à l'ordre dans lequel les définitions inductives sont introduites.

Ainsi, en toute rigueur, il faudrait définir la transformation de projection dans la définition ci-dessous par induction sur cet ordre (en plus de l'induction structurelle). En effet, un inductif I^i peut contenir dans sa déclaration des occurrences d'inductifs I^j pour $j < i$ dont la projection doit être "définie avant".

4.3.16 Définition (Projection, adéquat pour la réalisabilité)
Soit $\mathcal{P}'$ un CTSI pré-engendré de $\mathcal{P}$. On dit qu'il est adéquat pour la réalisabilité, si :

– pour tout inductif de second niveau I dont la déclaration est de la forme

$$\mathtt{Ind}_{\mathcal{P}'}(\overrightarrow{x : Q}^{p}, I : A, \overrightarrow{c : C}^{k})$$

il existe un inductif, noté $\lfloor I \rfloor$, de constructeurs notés $\lfloor c_1 \rfloor, \ldots, \lfloor c_k \rfloor$, tel que :

$$\mathtt{Ind}_{\mathcal{P}'}(\overrightarrow{\lfloor x : Q \rfloor}^{p}, \lfloor I \rfloor : \lfloor A \rfloor, \overrightarrow{\lfloor c \rfloor : \lfloor C \rfloor}^{k})$$

où la fonction $\lfloor \cdot \rfloor$ est étendue aux inductifs de la façon suivante :

$$\lfloor I[\overrightarrow{P}^{p}] \rfloor = \lfloor I \rfloor [\overrightarrow{\lfloor P \rfloor}^{p}]$$
$$\lfloor c[\overrightarrow{P}^{p}] \rfloor = \lfloor c \rfloor [\overrightarrow{\lfloor P \rfloor}^{p}]$$
$$\lfloor \mathtt{fix}\, f : A, \overrightarrow{x : B}^{l}.M \rfloor = \mathtt{fix}\, f : \lfloor A \rfloor, \overrightarrow{\lfloor x : B \rfloor}^{l}.\lfloor M \rfloor$$

8. Il faut admettre l'axiome du choix pour montrer l'existence de cet ordinal pour tout ensemble $\mathcal{I}$.

$$\lfloor \mathtt{case}_I(N, \overrightarrow{P}^p, \lambda \overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\ \overrightarrow{y}^n.T, \overrightarrow{\lambda z : E}^{m\ k}.F\ )\rfloor =$$
$$\mathtt{case}_{\lfloor I\rfloor}(\lfloor N\rfloor, \lfloor \overrightarrow{P}^p\rfloor, \lambda \lfloor \overrightarrow{y : B}^n\rfloor, i : \lfloor I[\overrightarrow{P}^p]\ \overrightarrow{y}^n\rfloor.\lfloor T\rfloor, \overrightarrow{\lambda \lfloor z : E}^{m\ k}\rfloor.\lfloor F\rfloor\ )$$

et où :

$$\begin{aligned}
\lfloor \overrightarrow{A}^{p+1}\rfloor &= \lfloor \overrightarrow{A}^p\rfloor && \text{si } \mathrm{lvl}(A) = 1\\
\lfloor \overrightarrow{A}^{p+1}\rfloor &= \lfloor \overrightarrow{A}^p\rfloor, \lfloor A_{p+1}\rfloor && \text{si } \mathrm{lvl}(A) = 2\\
\lfloor \overrightarrow{x : A}^{p+1}\rfloor &= \lfloor \overrightarrow{x : A}^p\rfloor && \text{si } \mathrm{lvl}(A) = 1\\
\lfloor \overrightarrow{x : A}^{p+1}\rfloor &= \lfloor \overrightarrow{x : A}^p\rfloor, x : \lfloor A_{p+1}\rfloor && \text{si } \mathrm{lvl}(A) = 2
\end{aligned}$$

– Les éliminations autorisées sur les inductifs de second niveau sont exactement celles qui se projettent sur les éliminations des inductifs de premier niveau :

$$\mathcal{E}_{\mathcal{P}'} = \mathcal{E}_{\mathcal{P}} \cup \{(I, \lceil s\rceil) | I \in \mathcal{I}_{\mathcal{P}'}, \mathrm{lvl}(I) = 2, (\lfloor I\rfloor, s) \in \mathcal{E}_{\mathcal{P}}\}$$

4.3.17 Exemple
Par exemple dans T^2, on dispose de l'inductif de premier niveau :

$$\mathtt{Ind}(\mathsf{nat} : \star, \mathsf{zero} : \mathsf{nat}, \mathsf{succ} : \mathsf{nat} \to \mathsf{nat})$$

et de l'inductif de second niveau :

$$\mathtt{Ind}(\mathsf{nat\_ind} : \mathsf{nat} \to \lceil\star\rceil, \mathsf{zero\_ind} : \mathsf{nat\_ind}\,\mathsf{zero}, \mathsf{succ\_ind} : \forall n : \mathsf{nat}.\mathsf{nat\_ind}\,n \to \mathsf{nat\_ind}\,(\mathsf{succ}\,n))$$

On peut alors calculer :

$$\mathtt{Ind}(\lfloor\mathsf{nat\_ind}\rfloor : \lfloor\mathsf{nat} \to \lceil\star\rceil\rfloor, \lfloor\mathsf{zero\_ind}\rfloor : \lfloor\mathsf{nat\_ind}\,\mathsf{zero}\rfloor, \lfloor\mathsf{succ\_ind}\rfloor : \lfloor\forall n : \mathsf{nat}.\mathsf{nat\_ind}\,n \to \mathsf{nat\_ind}\,(\mathsf{succ}\,n)\rfloor)$$

qui est égal à :

$$\mathtt{Ind}(\lfloor\mathsf{nat\_ind}\rfloor : \lfloor\star\rfloor, \lfloor\mathsf{zero\_ind}\rfloor : \lfloor\mathsf{nat\_ind}\rfloor, \lfloor\mathsf{succ\_ind}\rfloor : \lfloor\mathsf{nat\_ind}\rfloor \to \lfloor\mathsf{nat\_ind}\rfloor)$$

Ce type inductif est identique à nat *: ainsi, on peut supposer, sans perdre en généralité, que* $\lfloor\mathsf{nat\_ind}\rfloor = \mathsf{nat}$ *(*$\lfloor\mathsf{zero\_ind}\rfloor = \mathsf{zero}$*,* $\lfloor\mathsf{succ\_ind}\rfloor = \mathsf{succ}$*).*

On peut prouver le lemme :

4.3.18 Lemme
Si $\Gamma \vdash_{\mathcal{P}'} M : T$ *et* $\mathrm{lvl}(M) = 2$*, alors* $\lfloor\Gamma\rfloor \vdash_{\mathcal{P}} \lfloor M\rfloor : \lfloor T\rfloor$*.*

Démonstration On étend sans problème la preuve du lemme 3.4.8 aux cas des règles Point-Fixe, Case, Inductif et Constructeur (dans chacun des cas il suffit d'appliquer les hypothèses d'induction, puis de ré-appliquer la règle). ☺

On peut maintenant définir la notion de "CTSI engendré par $\mathcal{P}$" : il s'agit des systèmes adéquats pour la réalisabilité pour $\mathcal{P}$ dans lesquels une définition inductive est autorisée si et seulement si elle se projette sur un inductif de $\mathcal{P}$. Ce que l'on formalise par la définition suivante :

4.3.19 Définition
Soient $\mathcal{P}'$ un CTSI pré-engendré de $\mathcal{P}$. On dit de $\mathcal{P}'$ qu'il est un CTSI engendré par $\mathcal{P}$ si pour toute famille de termes Q_1, ..., Q_n, A, C'_1,...,C'_n qui vérifie :

– les conditions de la définition 2.1.3, c'est-à-dire :

1. $\mathrm{WF}_{\Gamma_P}(A)$ où $\Gamma_P = x_1 : Q_1, \ldots, x_p : Q_p$,
2. A est de la forme $\forall \overrightarrow{y : B}^n.s$
3. $\Gamma_P, \alpha : A \vdash C'_j : s$
4. pour tout j, C_j est de la la forme

$$\forall \overrightarrow{y : E_j}^{m_j}.\alpha\, \overrightarrow{D_j}^n$$

où α ne peut apparaître à l'intérieur de E_j que comme une conclusion.

– il existe un inductif de premier niveau I_1 bien formé :

$$\mathtt{Ind}(\lfloor \overrightarrow{x : Q}^p \rfloor, I_1 : \lfloor A \rfloor, \lfloor \overrightarrow{c : C}^k \rfloor)$$

où les $C_i = \lfloor C' \rfloor [I_1 [\lfloor \overrightarrow{x}^p \rfloor]/\alpha]$.
il existe un inductif de second niveau I_2 bien formé tel que :

$$\mathtt{Ind}(x_1 : Q_1, \ldots, x_p : Q_p, I_2 : A, c_1 : C_1, \ldots, c_k : C_k)$$

où les $C_i = C'[I[\overrightarrow{x}^p]/\alpha]$.

On admettra l'existence, pour tout CTSI $\mathcal{P}$, d'un CTSI engendré, que l'on notera $\mathcal{P}^2$ (il est construit par "saturation" de l'ensemble $\mathcal{I}_{\mathcal{P}^2}$). On adoptera donc la convention suivante :

4.3.20 Convention
On désignera par $\mathcal{P}^2$ n'importe quel CTSI engendré par $\mathcal{P}$.

Ainsi par "saturation" de $\mathcal{P}^2$ on a :

4.3.21 Lemme
On a :

$$\mathtt{Ind}_{\mathcal{P}^2}(\overrightarrow{x : Q}^p, I : A, \overrightarrow{c : C}^k)$$

si et seulement si

$$\mathrm{WF}_{\overrightarrow{x:Q}^p}(A) \text{ et } \mathrm{WF}_{\overrightarrow{x:Q}^p, \alpha:A}(C') \text{ et } \mathtt{Ind}_{\mathcal{P}}(\lfloor \overrightarrow{x : Q}^p \rfloor, \lfloor I \rfloor : \lfloor A \rfloor, \lfloor \overrightarrow{c : C}^k \rfloor)$$

où $C = C'[I[\overrightarrow{x}^p]/\alpha]$.

Morphismes de CTSI. On définit de façon analogue aux CTS la notion de morphisme de CTSI :

4.3.22 Définition (Morphisme de CTSI)
Soient $\mathcal{P}$ et $\mathcal{P}'$ deux CTSI et $\varphi : \mathcal{S}_\mathcal{P} \longrightarrow \mathcal{S}_{\mathcal{P}'}$.
On dit que la donnée φ est un morphisme de CTSI, si :

- Pour tout $I \in \mathcal{I}_\mathcal{P}$, tel que
$$\texttt{Ind}_\mathcal{P}(\overrightarrow{x : Q}^p, I : A, \overrightarrow{c : C}^k)$$
il existe un inductif de $\mathcal{P}'$, noté $\varphi(I)$, tel que :
$$\texttt{Ind}_{\mathcal{P}'}(\overrightarrow{x : \varphi(Q)}^p, \varphi(I) : \varphi(A), \overrightarrow{c : \varphi(C)}^k)$$
- Si $(I, s) \in \mathcal{E}_\mathcal{P}$, alors $(\varphi(I), \varphi(s)) \in \mathcal{E}_{\mathcal{P}'}$.

où φ est étendue aux termes de façon analogue au cas des morphismes de CTS : $\varphi(M)$ est obtenu à partir d'un terme M, en remplaçant chaque occurrence de chaque sorte s apparaissant dans M par $\varphi(s)$.
On notera $\mathcal{P} \hookrightarrow_\varphi \mathcal{P}'$ pour signifier que φ est un morphisme entre $\mathcal{P}$ et $\mathcal{P}'$.

Tout comme dans le cas des CTS, les morphismes ne changent pas la structure des termes ; ainsi on obtient :

4.3.23 Lemme
Si $\mathcal{P} \hookrightarrow_\varphi \mathcal{P}'$ *alors,* $A \rhd A' \Leftrightarrow \varphi(A) \rhd \varphi(A')$, $A \trianglerighteq A' \Leftrightarrow \varphi(A) \trianglerighteq \varphi(A')$ *et* $A \equiv A' \Leftrightarrow \varphi(A) \equiv \varphi(A')$.

On montre de façon identique au lemme 1.1.62 que les morphismes préservent la relation de cumulativité :

4.3.24 Lemme
Si $\mathcal{P} \hookrightarrow_\varphi \mathcal{P}'$ *alors,*
$$A \preccurlyeq_\mathcal{P} B \qquad \Rightarrow \qquad \varphi(A) \preccurlyeq_{\mathcal{P}'} \varphi(B)$$

La définition de morphisme contient donc exactement les hypothèses suffisantes pour montrer le lemme :

4.3.25 Lemme
Si $\mathcal{P} \hookrightarrow_\varphi \mathcal{P}'$ *alors,*
$$\Gamma \vdash_\mathcal{P} A : B \Rightarrow \varphi(\Gamma) \vdash_{\mathcal{P}'} \varphi(A) : \varphi(B)$$

Démonstration Par une induction immédiate sur la dérivation de $\Gamma \vdash_\mathcal{P} A : B$. ☺

Soulèvement. Tout comme dans le cas des CTS, on dispose de deux morphismes entre $\mathcal{P}$ et $\mathcal{P}^2$: l'identité (car $\mathcal{P} \subseteq \mathcal{P}^2$) et le soulèvement :

4.3.26 Définition
Soit $\mathcal{P}$ un CTSI et $\mathcal{P}^2$ un CTSI engendré par $\mathcal{P}$. La transformation de soulèvement $\lceil\cdot\rceil$ est définie par le morphisme de CTSI $s \mapsto \lceil s \rceil$ où à chaque inductif I de déclaration :
$$\texttt{Ind}(\overrightarrow{x : Q}^p, I : A, \overrightarrow{c : C}^k)$$

on associe un inductif $\lceil I \rceil$ dont la déclaration est

$$\texttt{Ind}(\lceil \overrightarrow{x : Q}^p \rceil, \lceil I \rceil : \lceil A \rceil, \overrightarrow{\lceil c \rceil : \lceil C \rceil}^k)$$

L'existence est assurée par le fait que $\lfloor \lceil A \rceil \rfloor = A$ et le lemme 4.3.21.

Ainsi comme $\lceil \cdot \rceil$ est un morphisme, on a :

4.3.27 Lemme
Si $\Gamma \vdash_{\mathcal{P}} M : T$, *alors* $\lceil \Gamma \rceil \vdash_{\mathcal{P}^2} \lceil M \rceil : \lceil T \rceil$.

Transformation de réalisabilité. Tout comme pour la projection (et le soulèvement), il faudrait en toute rigueur introduire la définition suivante par induction sur l'ordre dans lequel les inductifs sont introduits.

4.3.28 Définition (Réalisabilité pour les CTSI)
On étend $\| \cdot \|$ et $\cdot \Vdash \cdot$ aux inductifs de la façons suivante :

- Si I est un inductif de second niveau, on pose :
$$\|I[\overrightarrow{P}^p]\| = I_R[\|\overrightarrow{P}^p\|]$$
- Si c est le constructeur d'un inductif de second niveau, on pose :
$$\|c[\overrightarrow{P}^p]\| = c_R[\|\overrightarrow{P}^p\|]$$
- Si $\texttt{fix}\, f : A, \Delta.M$ est un point fixe de second niveau, on pose :
$$\|\texttt{fix}\, f : A, \Delta.M\| = (\texttt{fix}\, f_R : f \Vdash A, \|\Delta\|.\|M\|)\,[\lfloor \texttt{fix}\, f : A, \Delta.M \rfloor / f]$$
- Si $\texttt{case}_I(M, \overrightarrow{P}^p, \lambda \overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\ \overrightarrow{y}^n.T, \overrightarrow{\lambda \overrightarrow{z : E}^m.F}^k)$ est une élimination de second niveau, on pose :
$$\|\texttt{case}_I(M, \overrightarrow{P}^p, \lambda \overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\ \overrightarrow{y}^n.T, \overrightarrow{\lambda \overrightarrow{z : E}^m.F}^k)\| =$$
$$\texttt{case}_{I_R}(\|M\|, \|\overrightarrow{P}^p\|, \lambda \|\overrightarrow{y : B}^n\|, i : \lfloor I[\overrightarrow{P}^p]\ \overrightarrow{y}^n \rfloor, i_R : i \Vdash I[\overrightarrow{P}^p]\ \overrightarrow{y}^n.\Theta \Vdash \|T\|, \overrightarrow{\lambda \|\Xi\|.\|F\|}^k)$$

où :

- $\Theta = \lfloor \texttt{case}_I(i, \overrightarrow{P}^p, \lambda \overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\ \overrightarrow{y}^n.T, \overrightarrow{\lambda \overrightarrow{z : E}^m.F}^k) \rfloor$
- $\|\overrightarrow{P}^p\|$ est défini par les équations suivantes :
$$\begin{aligned} \|\overrightarrow{P}^p, Q\| &= \|\overrightarrow{P}^p\|, Q \qquad \text{si lvl}(Q) = 1 \\ \|\overrightarrow{P}^p, Q\| &= \|\overrightarrow{P}^p\|, \lfloor Q \rfloor, \|Q\| \qquad \text{si lvl}(Q) = 2 \end{aligned}$$
- Si T est de la forme $I[\overrightarrow{P}]$, $c[\overrightarrow{P}]$, $\texttt{fix}\, f : A, \overrightarrow{x : B}.N$, ou $\texttt{case}(M, \overrightarrow{P}, \lambda \overrightarrow{x : B}, i : C.T, \overrightarrow{\lambda \overrightarrow{y : Z}.F})$, alors on pose :
$$M \Vdash T \quad = \quad \|T\|\, M$$

– Si

$$\texttt{Ind}(\overrightarrow{x : Q}^{p}, I : A, \overrightarrow{c : C}^{k})$$

alors I_R est un inductif tel que :

$$\texttt{Ind}(\|\overrightarrow{x : Q}^{p}\|, I_R : \lfloor I \rfloor \Vdash A, \overrightarrow{c_R : \lfloor c \rfloor \Vdash C}^{k})$$

L'existence de I_R est assurée par le fait que la transformation de réalisabilité "conserve la structure du second niveau" et le lemme 4.3.21 :

4.3.29 Lemme
Soient t tel que $\mathrm{lvl}(t) = 1$ *et P tel que* $\mathrm{lvl}(P) = 2$, *on a* $\lfloor t \Vdash P \rfloor = \lfloor P_R \rfloor$ *et pour tout contexte* $\lfloor \|\Gamma\| \rfloor = \lfloor \Gamma_R \rfloor$.

Démonstration Par une induction immédiate sur les formes possibles de P. ☺

Exemple. Le cas le plus difficile à comprendre dans la définition précédente est celui de la transformation de Case ; en particulier l'introduction du terme Θ. Le plus simple pour le comprendre est probablement de le voir déplié sur un exemple. Nous prendrons la syntaxe de Coq pour la rendre la définition plus lisible ; on supposera donc ici que `Set` est une sorte de premier niveau et `Prop` une sorte de second niveau. Et nous utilisons pour cet exemple les inductifs définis à la figure 4.4.

Nous allons traduire une preuve de l'énoncé

$$\forall n : \mathsf{nat.even}\, n \to \mathsf{nat\_ind}\, n$$

définie par le terme Coq :

```
Fixpoint even_nat_ind (n : nat) (hn : even n) : nat_ind n :=
  match hn in even x return nat_ind x with
   | zero_even  ⇒  zero_ind
   | succ_even p hp ⇒  succ_ind (succ p)
                 (succ_ind p (even_nat_ind p hp))
  end.
```

Les prédicats inductifs even et nat_ind se projettent tous deux sur nat ; ainsi $\lfloor \mathsf{even} \rfloor = \lfloor \mathsf{nat\_ind} \rfloor = \mathsf{nat}$. Si on applique la fonction de projection à even_nat_ind, on obtient le terme suivant :

```
Fixpoint even_nat_ind_proj (hn : nat) : nat :=
  match hn in nat return nat with
   | zero  ⇒ zero
   | succ hp  ⇒ succ (succ (even_nat_ind_proj hp))
  end.
```

Nous pouvons maintenant calculer le terme $\|\mathsf{even\_nat\_ind}\|$. Les commentaires désignent le terme "replié" dans notre syntaxe.

```
(* détail: on demande à coq de ne pas générer les schémas
   d'induction des inductifs sinon il va réserver
   l'identifiant "nat_ind". *)
Unset Elimination Schemes.

Inductive nat : Set :=
| zero : nat
| succ : nat→nat.

Inductive even : nat→Prop :=
| zero_even : even zero
| succ_even : forall n, even n→even (succ (succ n)).

Inductive even_R : nat→nat→Prop :=
| zero_even_R : even_R zero zero
| succ_even_R : forall n r, even_R n r →
                    even_R (succ (succ n)) (succ r).

Inductive nat_ind : nat→Prop :=
| zero_ind : nat_ind zero
| succ_ind : forall n, nat_ind n→nat_ind (succ n).

Inductive nat_ind_R : nat→nat→Prop :=
| zero_ind_R : nat_ind_R zero zero
| succ_ind_R : forall n r, nat_ind_R n r →
                    nat_ind_R (succ n) (succ r).
```

FIGURE 4.4 – Exemple de transformation avec la syntaxe de COQ.

```
Fixpoint even_nat_ind_R
       (n : nat)
       (* hn : ⌊even n⌋ *)
       (hn : nat)
       (* hn_R : hn ⊩ even n *)
       (hn_R : even_R n hn)
    :  (* ⌊even_nat_ind n hn⌋ ⊩ nat_ind n *)
       nat_ind_R n (even_nat_ind_proj hn) :=
  match hn_R as i_R in even_R y i
        return
            (* ⌊even_nat_ind⌋ i ⊩ nat_ind y *)
           nat_ind_R y (even_nat_ind_proj i)
        with
    | zero_even_R  ⇒  zero_ind_R (* ‖zero_ind‖ *)
    | succ_even_R p hp hp_R  ⇒
     (* ‖succ_ind (succ p) (succ_ind p (even_nat_ind p hp))‖ *)
        succ_ind_R (succ p) (succ (even_nat_ind_proj hp))
            (succ_ind_R p (even_nat_ind_proj hp)
                          (even_nat_ind_R p hp hp_R))
end.
```

Ainsi, dans cet exemple $\Theta = \lfloor \mathsf{even\_nat\_ind} \rfloor\, i$, et on peut vérifier le typage des branches (ou demander au système Coq de le faire) :

$$\vdash \|\mathsf{zero\_ind}\| : \lfloor \mathsf{even\_nat\_ind} \rfloor\, \mathsf{zero} \Vdash \mathsf{nat\_ind}\, \mathsf{zero}$$

et

$$p : \mathsf{nat}, hp : \mathsf{nat}, hp_R : hp \Vdash \mathsf{even}\, n \vdash \|\mathsf{succ\_ind}\, (\mathsf{succ}\, p)\, (\mathsf{succ\_ind}\, p\, (\mathsf{even\_nat\_ind}\, p\, hp))\| : \lfloor \mathsf{even\_nat\_ind} \rfloor\, (\mathsf{succ}\, p) \Vdash \mathsf{nat\_ind}\, (\mathsf{succ}\, (\mathsf{succ}\, hp))$$

Théorème d'abstraction. On admet que la réalisabilité se comporte de façon similaire vis-à-vis de la substitution lorsqu'on l'étend aux inductifs :

4.3.30 Lemme
Soit A un terme de second niveau :

1. *Si $x^2 \notin \mathcal{FV}(A)$:*
   - $\|A\|[M/x] = \|A[M/x]\|$ *si* $\mathrm{lvl}(M) = 1$,
   - $(B \Vdash A)[M/x] = (B[M/x] \Vdash A[M/x])$ *si* $\mathrm{lvl}(M) = 1$,
2. *Si $x^1 \notin \mathcal{FV}(A)$:*
   - $\|A\|[\lfloor M \rfloor/x, \|M\|/x_R] \trianglerighteq \|A[M/x]\|$ *si* $\mathrm{lvl}(M) = 2$,
   - $(B \Vdash A)[\lfloor M \rfloor/x, \|M\|/x_R] \trianglerighteq (B[\lfloor M \rfloor/x] \Vdash A[M/x])$ *si* $\mathrm{lvl}(M) = 2$.

On en déduit le lemme suivant qui nous sera utile pour la preuve du théorème d'adéquation :

4.3.31 Lemme
1. $I[\lfloor \overrightarrow{P}^p \rfloor] \Vdash \mathtt{Arity}_I(\overrightarrow{P}^p) \trianglelefteq \mathtt{Arity}_{I_R}(\|\overrightarrow{P}^p\|)$
2. $\lfloor c \rfloor[\lfloor \overrightarrow{P}^p \rfloor] \Vdash \mathtt{Constr}^j_I(\overrightarrow{P}^p) \trianglelefteq \mathtt{Constr}^j_{I_R}(\|\overrightarrow{P}^p\|)$

Démonstration Soit $\mathtt{Ind}(\overrightarrow{x:Q}^p, I:A, \overrightarrow{c:C}^k)$ la déclaration de I. Par définition, on a :

$$\begin{aligned} \mathtt{Arity}_I(\overrightarrow{P}^p) &= A[\overrightarrow{P}^p/\overrightarrow{x}^p] \\ \mathtt{Constr}^j_I(\overrightarrow{P}^p) &= C_j[\overrightarrow{P}^p/\overrightarrow{x}^p] \end{aligned}$$

On a donc

$$\begin{aligned} \lfloor I \rfloor[\lfloor \overrightarrow{P}^p \rfloor] \Vdash \mathtt{Arity}_I(\overrightarrow{P}^p) &= I[\lfloor \overrightarrow{P}^p \rfloor] \Vdash (A[\overrightarrow{P}^p/\overrightarrow{x}^p]) \\ &= (\lfloor I \rfloor[\overrightarrow{x}^p][\lfloor \overrightarrow{P}^p \rfloor/\overrightarrow{x}^p]) \Vdash (A[\overrightarrow{P}^p/\overrightarrow{x}^p]) \\ &\quad \text{(lemme 4.3.30)} \\ &\trianglelefteq (\lfloor I \rfloor[\overrightarrow{x}^p] \Vdash A)\|[\overrightarrow{P}^p/\overrightarrow{x}^p]\|) \\ &= \mathtt{Arity}_{I_R}(\|\overrightarrow{P}^p\|) \end{aligned}$$

où $\|[\overrightarrow{P}^p/\overrightarrow{x}^p]\|$ désigne la substitution définie par les équations :

$$\begin{aligned} M\|[\overrightarrow{P}^p/\overrightarrow{x}^p, P_{p+1}/x_{p+1}]\| &= (M\|[\overrightarrow{P}^p/\overrightarrow{x}^p]\|)[P_{p+1}/x_{p+1}] \qquad \text{si } \mathrm{lvl}(P_{p+1}) = 1 \\ M\|[\overrightarrow{P}^p/\overrightarrow{x}^p, P_{p+1}/x_{p+1}]\| &= (M\|[\overrightarrow{P}^p/\overrightarrow{x}^p]\|)[\lfloor P_{p+1} \rfloor/x_{p+1}, \|P_{p+1}\|/x_{p+1_R}] \qquad \text{si } \mathrm{lvl}(P_{p+1}) = 2 \end{aligned}$$

La preuve de $\lfloor c \rfloor[\lfloor \overrightarrow{P}^p \rfloor] \Vdash \mathtt{Constr}^j_I(\overrightarrow{P}^p) \trianglelefteq \mathtt{Constr}^j_{I_R}(\|\overrightarrow{P}^p\|)$ est similaire. ☺

Nous pouvons à présent nous attaquer à la preuve du théorème d'adéquation. Tout comme la définition de $\|\cdot\|$, la preuve procède implicitement par induction sur l'ordre dans lequel les inductifs de $\mathcal{P}^2$ sont définis, ainsi, on pourra admettre la bonne formation des inductifs I_R qui peuvent apparaître dans les termes à transformer (on utilisera cette hypothèse dans le cas Case).

4.3.32 Théorème (Théorème d'adéquation de la réalisabilité)
Soit $\mathcal{P}$ un CTS tel que $\mathcal{P}^2 \models$ STRENGTHENING $\wedge$ WEAK-DOWNSTABLE.
Si $\Gamma \vdash M : T$, alors

$$\begin{cases} \mathrm{lvl}(M) = \mathrm{lvl}(T) = 1 \Rightarrow \|\Gamma\| \vdash M : T & (1) \\ \Gamma \vdash T : \lceil s \rceil \Rightarrow \|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash T & (2) \\ T \equiv \lceil s \rceil \Rightarrow \|\Gamma\|, x : \lfloor M \rfloor \vdash x \Vdash M : \lceil s \rceil & (3) \end{cases}$$

Démonstration On étend la preuve du théorème 4.1.8. On ne traitera que les cas (2) et (3) (le cas (1) se traite de façon similaire).

– Inductif : On a $M = I[\overrightarrow{P}^p]$ et $T = \mathtt{Arity}_I(\overrightarrow{P}^p)$ avec :

$$(\Gamma \vdash P_i : Q_i[P_1/x_1, \cdots, P_{i-1}/x_{i-1}]\ )_{i=1\ldots p}$$

D'après la bonne formation de l'inductif I on déduit que :

$$\mathrm{WF}\,(x_1 : Q_1, \ldots, x_p : Q_p)$$

on en déduit donc que pour $1 \leq i \leq p$ il existe une sorte s_i telle que :

$$x_1 : Q_1, \ldots, x_{i-1} : Q_{i-1} \vdash Q_i : s_i$$

ainsi par hypothèse d'induction (et par induction sur p) on en déduit que :

$$\|\Gamma\| \vdash P_i : Q_i[P_1/x_1, \cdots, P_{i-1}/x_{i-1}] \qquad \text{si } \mathrm{lvl}(Q_i) = 1$$

et

$$\|\Gamma\| \vdash \|P_i\| : \lfloor P_i \rfloor \Vdash (Q_i[P_1/x_1, \cdots, P_{i-1}/x_{i-1}]) \qquad \text{si } \mathrm{lvl}(Q_i) = 1$$

De plus, comme on a $\|\Gamma\| \subseteq \lfloor \Gamma \rfloor$, d'après le lemme 4.3.18, on obtient :

$$\|\Gamma\| \vdash \lfloor P_i \rfloor : \lfloor Q_i[P_1/x_1, \cdots, P_{i-1}/x_{i-1}] \rfloor \qquad \text{si } \mathrm{lvl}(Q_i) = 1$$

On peut donc utiliser la règle Inductif, pour en déduire :

$$\|\Gamma\| \vdash I_R[\|\overrightarrow{P}^p\|] : \mathtt{Arity}_{I_R}(\|\overrightarrow{P}^p\|)$$

Or, par réduction du type, et d'après le lemme précédent, on a :

$$\|\Gamma\| \vdash I_R[\|\overrightarrow{P}^p\|] : \lfloor I_R \rfloor[\lfloor \overrightarrow{P}^p \rfloor] \Vdash \mathtt{Arity}_I(\overrightarrow{P}^p)$$

– Implication (2) : par définition, $\|M\| = I_P[\|\overrightarrow{P}^p\|]$ et $M \Vdash T = \lfloor I_R \rfloor[\lfloor \overrightarrow{P}^p \rfloor] \Vdash \mathtt{Arity}_I(\overrightarrow{P}^p)$. On montre donc bien l'implication (2) (sans avoir besoin d'utiliser la prémisse $\Gamma \vdash T : \lceil s \rceil$).
– Implication (3) : Supposons $T = \mathtt{Arity}_I(\overrightarrow{P}^p) \equiv \lceil s \rceil$.

$$\lfloor I_R \rfloor[\lfloor \overrightarrow{P}^p \rfloor] \Vdash \mathtt{Arity}_I(\overrightarrow{P}^p) \trianglerighteq \lfloor I_R \rfloor[\lfloor \overrightarrow{P}^p \rfloor] \Vdash \lceil s \rceil = \lfloor I_R \rfloor[\lfloor \overrightarrow{P}^p \rfloor] \to \lceil s \rceil$$

On obtient par réduction du type :

$$\|\Gamma\| \vdash I_R[\|\overrightarrow{P}^p\|] : \lfloor I_R \rfloor[\lfloor \overrightarrow{P}^p \rfloor] \to \lceil s \rceil$$

et donc :

$$\|\Gamma\|, x : \lfloor I_R \rfloor[\lfloor \overrightarrow{P}^p \rfloor] \vdash I_R[\|\overrightarrow{P}^p\|]\, x : \lceil s \rceil$$

Or $\lfloor I_R \rfloor[\lfloor \overrightarrow{P}^p \rfloor] = \lfloor M \rfloor$ et $I_R[\|\overrightarrow{P}^p\|]\, x = x \Vdash M$. Nous avons ainsi bien montré :

$$\|\Gamma\|, x : \lfloor M \rfloor \vdash x \Vdash M : \lceil s \rceil$$

– Constructeur : On montre de façon similaire au cas précédent que :

$$\|\Gamma\| \vdash c_R[\|\overrightarrow{P}^p\|] : \lfloor c_R \rfloor[\lfloor \overrightarrow{P}^p \rfloor] \Vdash \mathtt{Constr}_I(\overrightarrow{P}^p)$$

– Implication (2) : par définition, $\|M\| = c_P[\|\overrightarrow{P}^p\|]$ et $M \Vdash T = \lfloor c_R \rfloor [\lfloor \overrightarrow{P}^p \rfloor] \Vdash \mathtt{Constr}_I(\overrightarrow{P}^p)$. On montre donc bien l'implication (2) (sans avoir besoin d'utiliser la prémisse $\Gamma \vdash T : \lceil s \rceil$).
– Implication (3) : Le cas est impossible car $\lfloor c_R \rfloor [\lfloor \overrightarrow{P}^p \rfloor] \Vdash \mathtt{Constr}_I(\overrightarrow{P}^p)$ n'est jamais convertible en une sorte (c'est une suite de produits dont la conclusion est un inductif).

– Case : on a

$$M = \mathtt{case}_I(N, \overrightarrow{P}^p, \lambda \overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\ \overrightarrow{y}^n.T, \overrightarrow{\lambda z : E}^{m\ k}.F\ )$$

et T de la forme $T = R[\overrightarrow{G}^n / \overrightarrow{y}^n, N/i]$ avec :

$$\Gamma \vdash N : I[\overrightarrow{P}^p]\ \overrightarrow{G}^n \tag{H1}$$

$$\Gamma, \overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\ \overrightarrow{y}^n \vdash R : \lceil r \rceil \tag{H2}$$

$$\Gamma, \overrightarrow{z : E_j}^{m_j} \vdash F_j : R[\overrightarrow{D_j}^n / \overrightarrow{y}^n, c_j[\overrightarrow{P}^p]\ \overrightarrow{z}^{m_j} / i] \text{ pour } 1 \leq j \leq k \tag{H3}$$

où :

$$\mathtt{Arity}_I(\overrightarrow{P}^p) = \forall \overrightarrow{y : B}^n.s \qquad \text{et} \qquad \mathtt{Constr}_I^j(\overrightarrow{P}^p) = \forall \overrightarrow{z : E_j}^{m_j}.I[\overrightarrow{P}^p]\ \overrightarrow{D_j}^{m_j}$$

On montre l'implication (2) sans utiliser la prémisse (nous n'en avons pas besoin car on peut la déduire des hypothèses).
On a :

$$\lfloor M \rfloor \Vdash T = \lfloor M \rfloor \Vdash (R[\overrightarrow{G}^n / \overrightarrow{y}^n, N/i])$$

et

$$\|M\| = \mathtt{case}_{I_R}(\|M\|, \|\overrightarrow{P}^p\|, \lambda \|\overrightarrow{y : B}^n\|, i : \lfloor I \rfloor [\lfloor \overrightarrow{P}^p \rfloor].\Theta \Vdash R.\overrightarrow{\|\lambda z : Z}^{m\ k}\|.\|F\|\ )$$

où :

$$\Theta = \lfloor \mathtt{case}_I(i, \overrightarrow{P}^p, \lambda \overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\ \overrightarrow{y}^n.T, \overrightarrow{\lambda z : E}^{m\ k}.F\ ) \rfloor$$

Or, d'après lemme 4.3.30 et le fait que $\overrightarrow{y} \notin \mathcal{FV}(\Theta)$ on a :

$$(\Theta \Vdash R)\|[\overrightarrow{G}^n / \overrightarrow{y}^n, N/i]\| \trianglerighteq \Theta[\lfloor N \rfloor / i] \Vdash (R[\overrightarrow{G}^n / \overrightarrow{y}^n, N/i]) = M \Vdash T$$

Ainsi, par réduction du type, pour montrer $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash T$, il suffit de démontrer

$$\|\Gamma\| \vdash \|M\| : (\Theta \Vdash R)\|[\overrightarrow{G}^n / \overrightarrow{y}^n, N/i]\| \tag{BUT}$$

Pour cela, nous pouvons appliquer la règle Case, et on se ramène à démontrer les séquents suivants :
– le typage de la transformation $\|N\|$ de l'objet détruit :

$$\|\Gamma\| \vdash \|N\| : \lfloor N \rfloor \Vdash I[\overrightarrow{P}^p]\ \overrightarrow{G}^n \tag{BUT1}$$

– le typage du type de retour $\Theta \Vdash R$:

$$\|\Gamma, \overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\ \overrightarrow{y}^n\| \vdash \Theta \Vdash R : \lceil r \rceil \tag{BUT2}$$

– le typage de la transformation de chacune des branches :

$$\|\Gamma, \overrightarrow{z : E_j}^{m_j}\| \vdash \|F_j\| : (\Theta \Vdash R)\|[\overrightarrow{D_j}^{n}/\overrightarrow{y}^{n}, c_j[\overrightarrow{P}^{p}]\ \overrightarrow{z}^{m_j}/i]\| \tag{BUT3}$$

pour $1 \leq j \leq k$.

Nous allons à présent montrer que l'on peut dériver ces séquents :

– (BUT1) : on déduit de (H1) que $\mathrm{WF}_\Gamma(I[\overrightarrow{P}^{p}]\ \overrightarrow{G}^{n})$; il existe donc une sorte s_1 telle que $\Gamma \vdash I[\overrightarrow{P}^{p}]\ \overrightarrow{G}^{n} : \lceil s_1 \rceil$. Ainsi par hypothèse d'induction appliquée à (H1), on obtient bien (BUT1).

– (BUT2) : le cas (3) de l'hypothèse d'induction associée à (H2), nous donne :

$$\|\Gamma, \overrightarrow{y : B}^{n}, i : I[\overrightarrow{P}^{p}]\ \overrightarrow{y}^{n}\|, x : \lfloor R \rfloor \vdash x \Vdash R : \lceil r \rceil \tag{1'}$$

Nous démontrerons (BUT2) en substituant dans cette dernière équation ; pour cela, nous commençons par montrer que Θ est de type $\lfloor R \rfloor$ dans le contexte de (BUT2). Ainsi nous appliquons le lemme 4.3.18 à (H1), (H2), (H3), puis nous utilisons la règle Case, pour démontrer :

$$\lfloor \Gamma, \overrightarrow{y : B}^{n}, i : I[\overrightarrow{P}^{p}]\ \overrightarrow{y}^{n} \rfloor \vdash \Theta : \lfloor R \rfloor$$

Or $\lfloor \Gamma, \overrightarrow{y : B}^{n}, i : I[\overrightarrow{P}^{p}]\ \overrightarrow{y}^{n} \rfloor \subseteq \|\Gamma, \overrightarrow{y : B}^{n}, i : I[\overrightarrow{P}^{p}]\ \overrightarrow{y}^{n}\|$, on en déduit :

$$\|\Gamma, \overrightarrow{y : B}^{n}, i : I[\overrightarrow{P}^{p}]\ \overrightarrow{y}^{n}\| \vdash \Theta : \lfloor R \rfloor$$

Ainsi, par substitution dans (1'), on montre bien (BUT2).

– (BUT3) : Soit $1 \leq j \leq k$. D'après la bonne formation de I, on a pour $1 \leq l \leq m_j$:

$$\overrightarrow{z : E_j}^{m_j} \vdash D_{j,l} : B_l[D_{j,1}/y_1, \ldots, D_{j,l-1}/y_{l-1}] \tag{2'}$$

et $\overrightarrow{z : E_j}^{m_j} \vdash c_j[\overrightarrow{P}^{p}]\ \overrightarrow{z}^{m_j} : I[\overrightarrow{P}^{p}]\ \overrightarrow{D}^{m_j}$ Ainsi, par substitution dans (H3), on obtient :

$$\Gamma, \overrightarrow{z : E_j}^{m_j} \vdash R[\overrightarrow{D_j}^{n}/\overrightarrow{y}^{n}, c_j[\overrightarrow{P}^{p}]\ \overrightarrow{z}^{m_j}/i] : \lceil r' \rceil \tag{3'}$$

pour une sorte r'. On peut donc appliquer le cas (2) de l'hypothèse d'induction à (H3) pour déduire :

$$\|\Gamma, \overrightarrow{z : E_j}^{m_j}\| \vdash \|F_j\| : \lfloor F_j \rfloor \Vdash \left(R[\overrightarrow{D_j}^{n}/\overrightarrow{y}^{n}, c_j[\overrightarrow{P}^{p}]\ \overrightarrow{z}^{m_j}/i]\right) \tag{4'}$$

On a

$$\Theta[\lfloor c_j[\overrightarrow{P}^{p}]\ \overrightarrow{z}^{m_j}/i \rfloor] \trianglerighteq \lfloor F_j \rfloor \tag{5'}$$

D'après la bonne formation de I_R on a, pour tout $1 \leq j \leq k$ et $1 \leq l \leq m_j$:

$$\left.\begin{aligned} \|\overrightarrow{z : E_j}^{m_j}\| \vdash D_{j,l} : B_l\|[D_{j,1}/y_1, \ldots, D_{j,l-1}/y_{l-1}]\| \text{ si } \mathrm{lvl}(D_{j,l}) = 1 \\ \|\overrightarrow{z : E_j}^{m_j}\| \vdash \lfloor D_{j,l} \rfloor : \lfloor B_l \rfloor\|[D_{j,1}/y_1, \ldots, D_{j,l-1}/y_{l-1}]\| \text{ si } \mathrm{lvl}(D_{j,l}) = 2 \\ \|\overrightarrow{z : E_j}^{m_j}\| \vdash \|D_{j,l}\| : (\lfloor D_{j,l} \rfloor \Vdash B_l)\|[D_{j,1}/y_1, \ldots, D_{j,l-1}/y_{l-1}]\| \text{ si } \mathrm{lvl}(D_{j,l}) = 2 \end{aligned}\right\} \tag{6'}$$

Ainsi, en substituant dans (1'), on obtient :

$$\|\Gamma, \overrightarrow{z : E_j}^{m_j}\| \vdash (\Theta \Vdash R)\|[\overrightarrow{D_j}^{n}/\overrightarrow{y}^{n}, c_j[\overrightarrow{P}^{p}]\ \overrightarrow{z}^{m_j}/i]\| : \lceil r \rceil \tag{7'}$$

D'après le lemme 4.3.30 et le fait que $\overrightarrow{y} \notin \Theta$, on a :

$$(\Theta \Vdash R)\|[\overrightarrow{D_j}^n/\overrightarrow{y}^n, c_j[\overrightarrow{P}^p]\ \overrightarrow{z}^{m_j}/i]\| \trianglerighteq \Theta[\lfloor c_j[\overrightarrow{P}^p]\ \overrightarrow{z}^{m_j}/i\rfloor] \Vdash \left(R[\overrightarrow{D_j}^n/\overrightarrow{y}^n, c_j[\overrightarrow{P}^p]\ \overrightarrow{z}^{m_j}/i]\right)$$
$$= \lfloor F_j \rfloor \Vdash \left(R[\overrightarrow{D_j}^n/\overrightarrow{y}^n, c_j[\overrightarrow{P}^p]\ \overrightarrow{z}^{m_j}/i]\right) \quad (8')$$

Ainsi, par conversion (règle Cumulativité), on déduit de (4'), (7') et (8'), la dérivation du séquent (BUT3).

Nous avons donc bien montré l'implication (2) sans utiliser la prémisse. Ainsi, on peut utiliser le fait que $\|\Gamma\| \vdash \|M\| : \lfloor M \rfloor \Vdash T$ pour démontrer l'implication (3) : supposons que $T \equiv \lceil s \rceil$. Puisque $\lfloor M \rfloor \Vdash \lceil s \rceil \trianglerighteq \lfloor M \rfloor \to \lceil s \rceil$, on en déduit bien que : $\|\Gamma\|, x : \lfloor M \rfloor \vdash \|M\|\, x : \lceil s \rceil$. Or $(\|M\|\, x) = x \Vdash M$ par définition.

– Point-Fixe : On a M de la forme $\texttt{fix}\, f : A, \overrightarrow{x : B}^{l+1}.N$ avec :

$$\Gamma, f : A \vdash \lambda \overrightarrow{x : B}^{l+1}.N : A$$

– Implication (2) : Supposons $\Gamma \vdash A : \lceil s \rceil$. Ainsi, par hypothèse d'induction, on en déduit :

$$\|\Gamma\|, f : \lfloor A \rfloor, f_R : f \Vdash A \vdash \|\lambda \overrightarrow{x : B}^{l+1}.N\| : \lfloor \lambda \overrightarrow{x : B}^{l+1}.N \rfloor \Vdash A$$

Or, d'après le lemme 4.3.18, on en déduit

$$\lfloor \Gamma \rfloor \vdash \lfloor M \rfloor : \lfloor A \rfloor$$

Et donc, puisque $\lfloor \Gamma \rfloor \subseteq \|\Gamma\|$, on obtient :

$$\|\Gamma\| \vdash \lfloor M \rfloor : \lfloor A \rfloor$$

Ainsi, par substitution, on a :

$$\|\Gamma\|, f_R : M \Vdash A \vdash \|\lambda \overrightarrow{x : B}^{l+1}.N\| : (\lfloor \lambda \overrightarrow{x : B}^{l+1}.N \rfloor \Vdash A)[\lfloor M \rfloor / f] \quad (1')$$

De plus, on en déduit que :

$$\|\Gamma\| \vdash M \Vdash A : \lceil t \rceil \quad (2')$$

pour une sorte t.

Puis, d'après le lemme 4.3.30, on a

$$\begin{aligned}
& (\lfloor \lambda \overrightarrow{x : B}^{l+1}.N \rfloor \Vdash A)[\lfloor M \rfloor / \lfloor f \rfloor] \\
& \quad (\text{car } f \notin \mathcal{FV}(A) \text{ implique } f_R \notin \mathcal{FV}(\lfloor \lambda \overrightarrow{x : B}^{l+1}.N \rfloor \Vdash A)) \\
= \ & (\lfloor \lambda \overrightarrow{x : B}^{l+1}.N \rfloor \Vdash A)[\lfloor M \rfloor / \lfloor f \rfloor, \|M\| / f_R] \\
\trianglerighteq \ & (\lfloor \lambda \overrightarrow{x : B}^{l+1}.N \rfloor [\lfloor M \rfloor / f] \Vdash (A[M/f]) \\
& \quad (\text{car } f \notin \mathcal{FV}(A)) \\
= \ & \lfloor (\lfloor \lambda \overrightarrow{x : B}^{l+1}.N \rfloor \rfloor [\lfloor M \rfloor / f] \Vdash A \\
& \quad (\text{car } \lfloor M \rfloor \text{ est un point fixe}) \\
\trianglelefteq \ & \lfloor M \rfloor \Vdash A
\end{aligned}$$

Ainsi, par conversion (règle CUMULATIVITÉ), appliquée à (1') et (2') on obtient que :

$$\|\Gamma\|, f_R : M \Vdash A \vdash \|\lambda \overrightarrow{x : B}^{l+1}.N\| : F \Vdash A$$

On peut donc appliquer la règle POINT-FIXE, pour en déduire que :

$$\|\Gamma\| \vdash \texttt{fix}\, f_R : M \Vdash A, \|\lambda \overrightarrow{x : B}^{l+1}.N\| : F \Vdash A$$

– Implication (3) : ce cas est impossible car T est nécessairement convertible en un produit. ☺

Chapitre 5

Paramétricité

La *paramétricité* est un concept introduit par John Charles Reynolds [Rey83] afin d'étudier l'abstraction de type polymorphe de système $\mathcal{F}$. Elle renvoie au fait que des programmes bien typés ne peuvent "inspecter le type de leurs arguments" : ils doivent se comporter uniformément au regard des types abstraits. Reynolds formalise cette notion en montrant que les programmes polymorphes de système $\mathcal{F}$ satisfont des *relations logiques* définies par induction sur la structure des types. Dans ce travail nous souhaitons montrer que la logique engendrée $\mathcal{P}^2$ est un bon cadre pour formaliser ces relations logiques.

Harry Mairson [Mai91] semble être le premier à formaliser la logique dans laquelle on peut définir ces relations, tout en reconnaissant que cette logique se plonge dans l'arithmétique du second-ordre de Leivant [Lei90]. La différence principale entre la logique de Mairson et celle de Leivant est que les individus sont des λ-termes purs alors que, dans l'arithmétique du second-ordre de Leivant, les individus sont représentés par des "expressions arithmétiques simples". Ainsi, la logique de Mairson se passe de codages pour exprimer des propriétés sur les programmes. Par la suite, d'autres auteurs ont poursuivi dans cette voie en considérant des individus typés (les termes du premier ordre dans ces logiques du second-ordre sont des λ-termes typés dans système $\mathcal{F}$). En exemple, on peut citer les travaux de Martin Abadi, Luca Cardelli & Pierre-Louis Curien [ACC93], ceux de Gordon Plotkin et Martin Abadi [PA93], ceux de Izumi Takeuti [Tak97], et enfin ceux de Philip Wadler [Wad07]. Les systèmes présentés dans [ACC93, Tak97, Wad07] se plongent dans $\mathcal{F}^2$. À la différence des autres, la logique pour la paramétricité de Abadi-Plotkin dispose de schémas d'axiomes pour la paramétricité : la logique prouve que tous les individus sont paramétriques. C'est pourquoi il est nécessaire d'ajouter des schémas d'axiomes à $\mathcal{F}^2$ pour y interpréter les preuves de [PA93].

Dans ce chapitre, nous verrons comment représenter les relations logiques comme des formules de $\mathcal{P}^2$ (section 5.1 et 5.2).

Le premier article présentant une façon de représenter les relations logiques dans le cadre des PTS est l'article de Jean-Philippe Bernardy *et. al* [BJP10]. Les auteurs y présentent une théorie de la paramétricité pour une classe de PTS. Nous verrons dans la section 5.3 comment cette présentation se compare avec la nôtre.

5.1 Présentation par l'exemple

Dans cette section, nous introduirons les définitions de notre théorie de la paramétricité sur des exemples, puis nous reviendrons dans la section suivante sur les définitions formelles.

Les relations logiques $M \sim_\tau N$ sont définies entre deux termes de premier niveau M et N et indicées par un type de premier niveau τ.

Dans $\mathcal{F}^2$, on peut définir la relation logique $M \sim_\tau N$ par les équations suivantes :

$$\begin{array}{rcll} M \sim_{\sigma\to\tau} N & = & \forall x:\sigma, x':\sigma'.x \sim_\sigma x' \to (M\,x) \sim_\tau (N\,x') & (*) \\ M \sim_{\forall\alpha:\star.\tau} N & = & \forall \alpha:\star, \alpha':\star, \alpha_P : \alpha\to\alpha'\to\lceil\star\rceil.(M\,\alpha) \sim_\tau (N\,\alpha') & (**) \\ M \sim_\alpha N & = & \alpha_P\,M\,N & \end{array}$$

où σ' désigne le type obtenu en remplaçant toutes les occurrences des variables libres α par α' (par exemple si $\sigma = \forall\alpha:\star.\alpha\to\beta\to\gamma$, alors $\sigma' = \forall\alpha:\star.\alpha\to\beta'\to\gamma'$). À titre d'exemple, on peut calculer la relation :

$$\begin{array}{rcl} f \sim_{\forall\alpha:\star.\alpha\to\alpha} f' & = & \forall\alpha\,\alpha':\star, \alpha_P:\alpha\to\alpha'\to\lceil\star\rceil.f\,\alpha \sim_{\alpha\to\alpha} f'\,\alpha' \\ & = & \forall\alpha\,\alpha':\star, \alpha_P:\alpha\to\alpha'\to\lceil\star\rceil, x:\alpha, x':\alpha'.x\sim_\alpha x' \to f\,\alpha\,x \sim_\alpha f'\,\alpha'\,x' \\ & = & \forall\alpha\,\alpha':\star, \alpha_P:\alpha\to\alpha'\to\lceil\star\rceil, x:\alpha, x':\alpha'.\alpha_P\,x\,x' \to \alpha_P\,(f\,\alpha\,x)\,(f'\,\alpha'\,x') \end{array}$$

On peut alors montrer dans $\mathcal{F}^2$ la formule suivante :

$$\forall f\,f' : \forall\alpha.\alpha\to\alpha.f \sim_{\forall\alpha:\star.\alpha\to\alpha} f' \to \forall\gamma:\star, y:\gamma.f\,\gamma\,y =_\gamma y$$

En effet, supposons $f \sim_{\forall\alpha:\star.\alpha\to\alpha} f'$ et soit γ un type et y de type γ. Si on prend $X = \lambda x:\gamma\_:\gamma.x =_\gamma y$ et $\alpha = \alpha' = \gamma$ dans $f \sim_{\forall\alpha:\star.\alpha\to\alpha} f'$, alors on obtient : $\forall x\,x':\gamma.x =_\gamma y \to f\,\gamma\,x =_{\mathsf{nat}} y$. Et donc que $f\,\gamma\,y =_\gamma y$.

Tout comme la réalisabilité, les relations logiques disposent d'un théorème "d'adéquation" parfois appelé *théorème d'abstraction*. Il peut s'exprimer dans $\mathcal{F}^2$ de la façon suivante :

Si $\vdash_\mathcal{F} t : \tau$, alors il existe une preuve, notée $[\![t]\!]$ tel que $\vdash_{\mathcal{F}^2} [\![t]\!] : t \sim_\tau t$.

Ainsi, en combinant les résultats précédents, on montre que tous les habitants clos du type de l'identité sont extensionnellement égaux à l'identité :

5.1.1 Lemme
Si $\vdash_\mathcal{F} t : \forall\alpha:\star.\alpha\to\alpha$, *alors*

$$\vdash_{\mathcal{F}^2} \forall\alpha:\star, x:\alpha.t\,\alpha\,x =_\alpha x$$

Ceci est probablement l'exemple le plus simple de "theorem for free" tels qu'ils sont présentés par Philip Wadler [Wad89]. L'idée étant que l'on peut obtenir "gratuitement" certaines propriétés des programmes en générant, par la transformation de paramétricité, des preuves à partir de leur dérivation de typage. Ainsi, on déduit de la dérivation de typage d'un terme de type $\forall\alpha:\star.\alpha\to\alpha$ qu'il est extensionnellement égal à l'identité. Bien sûr, seule la preuve générée par le théorème d'abstraction est calculée automatiquement. L'utilisateur de cette "méthodologie" doit trouver par lui-même une bonne façon d'instancier les relations, dans l'énoncé de paramétricité, afin d'en déduire des propriétés intéressantes.

La paramétricité définie en termes de réalisabilité. Comme nous avons pu le mentionner dans le chapitre précédent, la relation de paramétricité est ici définie en termes de réalisabilité et de soulèvement. On posera donc :

$$M \sim_T N \quad = \quad N \Vdash (M \Vdash \lceil T \rceil)$$

et la transformation sera donnée par :

$$[\![M]\!] \quad = \quad |||\lceil M \rceil|||$$

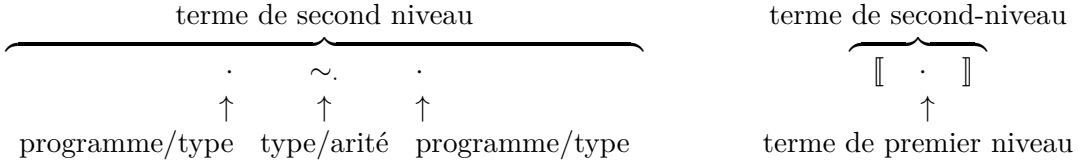
Nous ne sommes pas les premiers à constater ce lien syntaxique entre réalisabilité et paramétricité ; il est mentionné pour $\mathcal{F}^2$ dans l'article [Wad07]. Nous reviendrons sur les propriétés formelles de cette définition dans la section suivante.

On peut constater que le traitement des deux produits de système $\mathcal{F}$ est uniforme ; ainsi, les équations $(*)$ et $(**)$ sont deux instances de :

$$\begin{aligned} M \sim_{\forall x:A.B} N = N \Vdash (M \Vdash \lceil \forall x : A.B \rceil) &= \forall x : A, x' : A', x_P : x' \Vdash (x \Vdash \lceil A \rceil).(N\, x') \Vdash ((M\, x) \Vdash \lceil B \rceil) \\ &= \forall x : A, x' : A', x_P : x \sim_A x'.(M\, x) \sim_B (M'\, x') \end{aligned}$$

Comme on peut le voir, la formule $\alpha \sim_s \alpha' = \alpha' \Vdash (\alpha \Vdash \lceil s \rceil) = \alpha \to \alpha' \to \lceil s \rceil$ est une arité correspondant à la "signature" de la relation associée au type α.

Ainsi, les relations logiques $\cdot \sim_\cdot \cdot$ sont des termes de second niveau qui mettent en relation deux objets de premier niveau et qui sont indicés par un terme de premier niveau qui habite une sorte. De même, la transformation $[\![\cdot]\!]$ envoie les termes de premier niveau vers des termes de second niveau qui habitent (et donc prouvent) une relation logique. Ce que l'on peut résumer par la figure ci-dessous.



relation de paramétricité **transformation de paramétricité**

Paramétricité et inductifs. Tout comme la réalisabilité, les relations logiques s'étendent naturellement aux inductifs en appliquant la transformation de paramétricité à chacun des constructeurs. Ainsi, on associera à tout inductif de premier niveau :

$$\texttt{Ind}(\overrightarrow{x : Q}^p, I : A, \overrightarrow{c : C}^k)$$

l'inductif de second niveau suivant :

$$\texttt{Ind}([\![\overrightarrow{x : Q}^p]\!], I_P : I \sim_A I, \overrightarrow{c_P : c \sim_C c}^k)$$

À titre d'exemple, nous proposons d'étudier la traduction d'inductifs de $I\lambda$ dans la logique engendrée $I\lambda^2$. On rappelle que le système $I\lambda$ est l'extension du système $\mathcal{T}$ à tous les inductifs bien formés autorisés dans notre cadre. La logique engendré $I\lambda^2$ est l'extension aux inductifs du système λ^2 où les éliminations

autorisées $\mathcal{E}_{I\lambda^2}$ sont constituées de l'union de celles autorisés dans $I\lambda$ et de celles qui se projettent sur $I\lambda$:

$$\mathcal{E}_{I\lambda^2} = \mathcal{E}_{I\lambda} \cup \{(I,s) | \, \mathrm{lvl}(I) = \mathrm{lvl}(s) = 2 \wedge (\lfloor I \rfloor, \lfloor s \rfloor) \in \mathcal{E}_{I\lambda} \, \}$$

Dans ce système, il est donc impossible de construire des objets de second niveau par élimination d'objet de premier niveau.

Le système $I\lambda^2$ contient le système $\mathcal{T}^{++^2}$, c'est pourquoi on utilisera les notations établies pour $\mathcal{T}^{++^2}$ dans la suite de cette section.

Nous pouvons calculer la relation associée à l'inductif encodant les listes. Nous rappelons que, dans notre syntaxe, on définit cet inductif par la déclaration :

$$\mathtt{Ind}(\alpha : \star, \mathsf{list} : \star, \mathsf{nil} : \mathsf{list}[\alpha], \mathsf{cons} : \alpha \to \mathsf{list}[\alpha] \to \mathsf{list}[\alpha])$$

qui s'exprime en Coq par la définition :

```
Inductive list (A : Set) : Set :=
  | empty : list A
  | cons : A→ list A→ list A.
```

On peut alors calculer la définition de $I\lambda^2$ suivante :

$$\begin{aligned}\mathtt{Ind}\,(&\alpha : \star, \alpha' : \star, \alpha_P : \alpha \to \alpha' \to \lceil\star\rceil, \mathsf{list}_P : \mathsf{list} \sim_\star \mathsf{list},\\ &\mathsf{nil}_P : \mathsf{nil}[\alpha] \sim_{\mathsf{list}[\alpha]} \mathsf{nil}[\alpha'],\\ &\mathsf{cons}_P : \mathsf{cons}[\alpha] \sim_{\alpha \to \mathsf{list}[\alpha] \to \mathsf{list}[\alpha]} \mathsf{cons}[\alpha'])\end{aligned}$$

qui se déplie, dans la syntaxe Coq, de la façon suivante :

```
Inductive list_P (α α' : Set) (α_P : α→ α'→ Prop) :
                 list α→ list α'→ Prop :=
  | empty_P : list_P α α' α_P (empty α) (empty α')
  | cons_P : forall (x : α) (x':α'), α_P x x'→
                forall (tl : list α) (tl' :list α'),
                  list_P α α' α_P tl tl'→
                    list_P α α' α_P (cons α x tl) (cons α x' tl').
```

On voit bien que $\mathsf{list}_P[\alpha, \alpha', \alpha_P]$ définit un prédicat inductif vérifié exactement par les listes de même taille, dont les éléments sont points à points en relation pour α_P.

Ainsi, on montre qu'en instanciant le prédicat α_P par un prédicat toujours vrai, alors le prédicat $\mathsf{list}_P[\alpha, \alpha', \lambda x : \alpha, y : \alpha'.0 =_{\mathsf{nat}} 0]$, que l'on note $\mathsf{same\_size}[\alpha, \alpha']$, est vérifié uniquement par les listes de même taille.

Ce qui peut se traduire (dans $I\lambda^2$ par exemple) par :

$$\alpha : \star, \alpha' : \star \vdash \forall l : \mathsf{list}[\alpha], l' : \mathsf{list}[\alpha'].\, (\mathsf{same\_size}[\alpha, \alpha']\, l\, l') \to ((\mathsf{length}[\alpha]\, l) =_{\mathsf{nat}} \mathsf{length}[\alpha']\, l'))$$

où $\mathsf{length}[\alpha]$ désigne le terme qui calcule la longueur d'une liste :

$$\mathtt{fix}\ f : \mathsf{list}[\alpha] \to \mathsf{nat}, l : \mathsf{list}[\alpha].\, \mathsf{case}_{\mathsf{list}}(l, \alpha, \lambda\_ : \mathsf{list}[\alpha].\mathsf{nat}, \overline{0}, \lambda\_ : \alpha, tl : \mathsf{list}[\alpha].\mathsf{S}\,(f\, tl))$$

La réciproque le l'implication précédente n'est pas prouvable, on dispose pas d'un principe d'induction sur les listes. Nous détaillerons ce phénomène sur un autre exemple dans le paragraphe suivant.

Ainsi, on obtient (dans $I\lambda^2$ ici), un nouveau "theorem for free" : les termes de types $\mathsf{list}[\alpha] \to \mathsf{nat}$ sont des fonctions constantes sur les listes de même taille. Ce que l'on peut exprimer ainsi :

5.1.2 Lemme
Si $\alpha : \star \vdash_{I\lambda} t : \mathsf{list}[\alpha] \to \mathsf{nat}$*, alors*

$$\alpha : \star \vdash_{I\lambda^2} \forall l\, l' : \mathsf{list}[\alpha].\, (\mathsf{same\_size}[\alpha, \alpha']\, l\, l') \to (t\, l) =_{\mathsf{nat}} (t\, l')$$

Paramétricité et "naturalité". Plus généralement, les relations logiques permettent de prouver des propriétés de "naturalité". Soit f de type $\alpha \to \beta$, on note $\mathbf{Graph}_f$ le prédicat $\lambda x : \alpha, y : \beta. f\, x =_\beta y$, le prédicat $(\mathsf{list}_P\, \alpha\, \beta\ \mathbf{Graph}_f\ l_1\, l_2)$ s'interprète comme le fait que $l_2 = \mathsf{map}[\alpha, \beta]\, f\, l_1$ où $\mathsf{map}[\alpha, \beta]$ désigne le terme

$$\begin{aligned}\mathsf{map}[\alpha, \beta] = \mathtt{fix}\, m : \mathsf{list}[\alpha] \to \mathsf{list}[\beta], f : \alpha \to \beta, l : \mathsf{list}[\alpha].\\ \mathsf{case}_{\mathsf{list}}\, (l, \alpha, \lambda\_ : \mathsf{list}[\alpha].\mathsf{list}[\beta],\\ \mathsf{nil}[\beta],\\ \lambda hd : \alpha, tl : \mathsf{list}[\alpha].\mathsf{cons}[\beta]\, (f\, hd)\, (m\, f\, tl))\end{aligned}$$

vérifiant $\alpha : \star, \beta : \star \vdash \mathsf{map}[\alpha, \beta] : (\alpha \to \beta) \to \mathsf{list}[\alpha] \to \mathsf{list}[\beta]$.

On peut alors montrer que :

$$\begin{aligned}\alpha : \star, \beta : \star, f : \alpha \to \beta \quad \vdash_{I\lambda^2} \quad \forall l_1 : \mathsf{list}[\alpha], l_2 : \mathsf{list}[\beta].\\ (\mathsf{list}_P[\alpha, \beta, \mathbf{Graph}_f]\, l_1\, l_2) \to (\mathsf{map}[\alpha, \beta]\, f\, l_1 =_{\mathsf{list}[\beta]} l_2)\end{aligned}$$

Il est important de constater que la réciproque n'est pas prouvable. En effet, on peut montrer que si $r \Vdash \mathsf{list}_P[\alpha, \beta, \mathbf{Graph}_f]\, l_1\, l_2$, alors r est de type $\lfloor \mathsf{list}_P[\alpha, \beta, \mathbf{Graph}_f]\, l_1\, l_2 \rfloor = \mathsf{list}[\mathsf{unit}]$ qui correspond "moralement" à une liste $[r_1, \ldots, r_n]$ telle que $r_k \Vdash u_k =_\beta (f\, v_k)$ pour $1 \leq k \leq n$ où n est la longueur commune aux listes l_1 et l_2 d'éléments respectifs $[u_1, \ldots, u_n]$ et $[v_1, \ldots, v_n]$. En particulier, on en déduit que $r \Vdash \mathsf{list}_P[\alpha, \beta, \mathbf{Graph}_f]\, l_1\, l_2 \to \mathsf{length}\, r =_{\mathsf{nat}} \mathsf{length}\, l_1$.

Or le contexte

$$\Gamma = \alpha : \star, \beta : \star, f :\ \alpha \to \beta, l_1 : \mathsf{list}[\alpha], l_2 : \mathsf{list}[\beta], h : \mathsf{map}[\alpha, \beta]\, f\, l_1 =_{\mathsf{list}[\beta]} l_2$$

est un contexte uniformément auto réalisé. Donc si $\mathsf{list}_P\ \mathbf{Graph}_f\ l_1\, l_2$ est prouvable sous le contexte Γ, il existe un programme r tel que $\Gamma \vdash r \Vdash \mathsf{list}_P[\alpha, \beta, \mathbf{Graph}_f]\, l_1\, l_2$ et vérifiant $\vdash_{I\lambda} r : \mathsf{list}[\mathsf{unit}]$. On peut alors en déduire la dérivabilité du séquent :

$$\alpha : \star \vdash_{I\lambda^2} \forall l : \mathsf{list}[\alpha].\mathsf{length}\, r =_{\mathsf{nat}} \mathsf{length}\, l$$

Ce séquent signifie que toutes les listes ont la même longueur qu'une liste fixée. À partir de là, il est aisé de prouver que $\vdash_{I\lambda^2} \bot_{\mathrm{w}}$. On en déduit :

5.1.3 Lemme
Si on a :

$$\alpha : \star, \beta : \star, f : \alpha \to \beta \quad \vdash_{I\lambda^2} \quad \forall l_1 : \mathsf{list}[\alpha], l_2 : \mathsf{list}[\beta]. \\ (\mathsf{map}[\alpha, \beta]\, f\, l_1 =_{\mathsf{list}[\beta]} l_2) \to (\mathsf{list}_P[\alpha, \beta, \mathbf{Graph}_f]\, l_1\, l_2)$$

alors $\vdash_{I\lambda^2} \bot_{\mathrm{w}}$ *(c-à-d. $I\lambda^2$ n'admet que des modèles triviaux).*

La réalisabilité nous permet donc de prouver des résultats d'indépendance : c'est parce que la conclusion $\mathsf{list}_P\, \alpha\, \beta\, \mathbf{Graph}_f\, l_1\, l_2$ est "informative" qu'elle ne peut pas être déduite d'un contexte uniformément auto-réalisé.

Néanmoins, en renforçant le contexte avec une hypothèse "informative", on peut prouver une implication moins forte. Soit le prédicat :

$$\begin{array}{l} \mathtt{Ind}\,(\alpha : \star, \mathsf{list\_ind} : \mathsf{list}[\alpha] \to \lceil\star\rceil, \\ \quad \mathsf{nil\_ind} : \mathsf{list\_ind}\,\mathsf{nil}[\alpha], \\ \quad \mathsf{nil\_cons} : \forall x : \alpha, l : \mathsf{list}[\alpha].\mathsf{list\_ind}\, l \to \mathsf{list\_ind}\,(\mathsf{cons}[\alpha]\, x\, l)) \end{array}$$

Ce prédicat énonce un principe d'induction pour les listes qui peut être utilisé pour prouver l'implication :

$$\alpha : \star, \beta : \star, f : \alpha \to \beta \vdash_{I\lambda^2} \forall l : \mathsf{list}[\alpha].\mathsf{list\_ind}\, l \to (\mathsf{list}_P[\alpha, \beta, \mathbf{Graph}_f]\, l\, (\mathsf{map}[\alpha, \beta]\, f\, l))$$

Ce qui nous permet de déduire :

5.1.4 Lemme

$$\alpha : \star, \beta : \star, f : \alpha \to \beta \quad \vdash_{I\lambda^2} \quad \forall l_1 : \mathsf{list}[\alpha], l_2 : \mathsf{list}[\beta]. \\ (\mathsf{list}_P[\alpha, \beta, \mathbf{Graph}_f]\, l_1\, l_2) \leftrightarrow (\mathsf{map}[\alpha, \beta]\, f\, l_1 =_{\mathsf{list}[\beta]} l_2 \wedge \mathsf{list\_ind}\, l_1)$$

Considérons à présent un programme t tel que

$$\alpha : \star \vdash_{I\lambda} t : \mathsf{list}[\alpha] \to \mathsf{list}[\alpha]$$

et grâce au théorème d'abstraction, on en déduit que :

$$\alpha : \star, \alpha' : \star, \alpha_P : \alpha \to \alpha' \to \lceil\star\rceil \vdash_{I\lambda^2} \forall l : \mathsf{list}[\alpha], l' : \mathsf{list}[\alpha'].\mathsf{list}_P[\alpha, \alpha', \alpha_P]\, l\, l' \to \mathsf{list}_P[\alpha, \alpha', \alpha_P]\, (t\, l)\, (t\, l')$$

On en déduit alors que :

$$\alpha : \star, \beta : \star, f : \alpha \to \beta \vdash_{I\lambda^2} \forall l : \mathsf{list}[\alpha], l' : \mathsf{list}[\beta].\mathsf{list}_P[\alpha, \beta, \mathbf{Graph}_f]\, l\, l' \to \mathsf{list}_P[\alpha, \beta, \mathbf{Graph}_f]\, (t\, l)\, (t\, l')$$

On peut alors utiliser le lemme précédent pour montrer que tous les programmes qui envoient des listes d'éléments de type α vers des listes d'éléments de type α sont prouvablement "naturels" :

5.1.5 Lemme
Si $\alpha : \star \vdash_{I\lambda} t : \mathsf{list}[\alpha] \to \mathsf{list}[\alpha]$, *alors*

$$\alpha : \star \vdash_{I\lambda^2} \forall l : \mathsf{list}[\alpha].\mathsf{list\_ind}\, l \to t\, (\mathsf{map}[\alpha, \beta]\, f\, l) =_\beta \mathsf{map}[\alpha, \beta]\, f\, (t\, l)$$

La relation de paramétricité est définie pour les termes du premier niveau (les termes de $\mathcal{P}$ donc) et la logique engendrée $\mathcal{P}^2$ peut alors être comprise comme une logique minimaliste, dans laquelle on peut exprimer la paramétricité des termes de $\mathcal{P}$. Ainsi, comme la paramétricité ne concerne que les propriétés des programmes, il peut être utile de plonger la logique $\mathcal{P}^2$ dans des logiques plus expressives. En particulier, en présence d'inductifs, les éliminations interdites afin d'être compatible avec la réalisabilité (celles de la forme (I, s) lorsque $\mathrm{lvl}(I) = 1$ et $\mathrm{lvl}(s) = 2$) peuvent être autorisées une fois la traduction effectuée. Ainsi, on peut déduire du lemme précédent que :

5.1.6 Lemme
Soit $\mathcal{P}$ tel que $I\lambda^2 \subseteq \mathcal{P}$ et tel que $\alpha : \mathsf{Set} \vdash_{\mathcal{P}} \forall l : \mathsf{list}[\alpha].\mathsf{list\_ind}\, l$.
Si $\alpha : \star \vdash_{I\lambda} t : \mathsf{list}[\alpha] \to \mathsf{list}[\alpha]$, alors

$$\alpha : \star \vdash_{\mathcal{P}} \forall l : \mathsf{list}[\alpha].t\,(\mathsf{map}[\alpha, \beta]\, f\, l) =_\beta \mathsf{map}[\alpha, \beta]\, f\, (t\, l)$$

Nous verrons d'autres exemples de ce type d'interprétations dans des systèmes plus expressifs dans la section 5.3.

5.2 Définitions & théorème d'abstraction

On définit la relation et la transformation de paramétricité en termes de réalisabilité itérée. On ne traite ici que le cas des relations logiques binaires. Le cas des relations n-aires pourrait toutefois être développé de façon similaire en itérant n fois la réalisabilité (au lieu de deux fois).

5.2.1 Définition (Relation de paramétricité)
Soit $\mathcal{P}$ un CTSI.
On définit la relation de paramétricité $\cdot \sim_\cdot \cdot$ par l'équation :

$$M \sim_T N = N \Vdash (M \Vdash \lceil T \rceil)$$

pour trois termes M, N et T de $\mathcal{P}$. De même, on définit la transformation de paramétricité $[\![\cdot]\!]$ par

$$[\![M]\!] = |||\lceil M \rceil|||$$

pour un terme M de $\mathcal{P}$ et par

$$[\![\Gamma]\!] = |||\lceil \Gamma \rceil|||$$

pour un contexte Γ de $\mathcal{P}$. Ainsi, $[\![\Gamma]\!]$ est un contexte de $\mathcal{P}^2$ et les termes $[\![M]\!]$ et $M \sim_T N$ sont des termes de second niveau de $\mathcal{P}^2$.

On notera que dans la définition précédente le "générateur de noms frais" $x \mapsto x_R$ est donc itéré deux fois. Observons ce phénomène sur la transformation d'un produit :

$$\begin{aligned}
f \sim_{\forall x:A.B} f' &= f' \Vdash (f \Vdash \lceil \forall x : A.B \rceil) \\
&= f' \Vdash (\forall x : \lfloor \lceil A \rceil \rfloor, x_R : x \Vdash \lceil A \rceil.(f\,x) \Vdash B) \\
&= f' \Vdash (\forall x : A, x_R : x \Vdash \lceil A \rceil.(f\,x) \Vdash B) \\
&= \forall x : A.f' \Vdash (\forall x_R : x \Vdash \lceil A \rceil.(f\,x) \Vdash B) \\
&= \forall x : A, x_R : \lfloor x \Vdash \lceil A \rceil \rfloor, x_{RR} : x_R \Vdash (x \Vdash \lceil A \rceil).(f'\,x_R) \Vdash ((f\,x) \Vdash B) \\
&\quad \text{(lemme 4.1.3)} \\
&= \forall x : A, x_R : \lfloor \lceil A \rceil_R \rfloor, x_{RR} : x_R \Vdash (x \Vdash \lceil A \rceil).(f'\,x_R) \Vdash ((f\,x) \Vdash B) \\
&= \forall x : A, x_R : \lfloor \lceil A \rceil_R \rfloor, x_{RR} : x \sim_A x_R.f\,x \sim_B f'\,x_R
\end{aligned}$$

On voit donc que les deux premières prémisses A et $\lfloor \lceil A \rceil_R \rfloor$ ne sont pas égales : la seconde voit ses variables libres renommées par le générateur de noms frais. Ce que l'on peut observer par exemple lorsqu'on déplie (elles sont soulignées ici) :

$$f \sim_{\forall \alpha:\star.\alpha\to\alpha} f' = \forall \alpha : \star, \alpha_R : \star, \alpha_{RR} : \alpha \to \alpha_R \to \lceil \star \rceil, x : \underline{\boldsymbol{\alpha}}, x_R : \underline{\boldsymbol{\alpha_R}}.\alpha_{RR}\,x\,x_R \to \alpha_{RR}\,(f\,x)\,(f'\,x_R)$$

Puisque les doubles indices x_{RR} ne facilitent pas la lecture, nous renommons les variables liées dans les formules de paramétricité, selon la convention suivante :

5.2.2 Convention
Dans un contexte où l'on décrira une formule de paramétricité, c'est-à-dire une formule obtenue par itération de la construction de réalisabilité, nous adopterons les conventions suivantes :

- *On notera x' les variables de premier niveau de la forme x_R.*
- *Si A est un terme de premier niveau, on désignera par A' le renommage des variables x de A par x_R, c'est-à-dire :*
$$A' = \lfloor \lceil A \rceil_R \rfloor$$
- *On désignera par x_P les variables de la forme x_{RR}.*

En adoptant cette convention, on peut déplier l'exemple précédent de la façon suivante :

$$f \sim_{\forall \alpha:\star.\alpha\to\alpha} f' = \forall \alpha : \star, \alpha' : \star, \alpha_P : \alpha \to \alpha' \to \lceil \star \rceil, x : \alpha, x' : \alpha'.\alpha_P\,x\,x' \to \alpha_P\,(f\,x)\,(f'\,x')$$

De façon générale, on utilisera le lemme suivant pour déplier les relations de paramétricité (en adoptant la convention précédente) :

5.2.3 Lemme (Paramétricité dépliée)
On a :

$$\begin{aligned}
M \sim_s N &= M \to N \to \lceil s \rceil \\
M \sim_{\forall x:A.B} N &= \forall x : A, x' : A', x_P : x \sim_A x'. M\, x \sim_B N\, x' \\
M \sim_C N &= [\![C]\!]\, M\, N \text{ si } C \text{ est une application, une abstraction ou une variable}
\end{aligned}$$

$$\begin{aligned}
[\![x]\!] &= x_P \\
[\![\lambda x : A.M]\!] &= \lambda x : A, x' : A', x_P : x \sim_A x'. [\![M]\!] \\
[\![(M\, N)]\!] &= ([\![M]\!]\, N\, N'\, [\![N]\!]) \\
[\![M]\!] &= \lambda x : M, x' : M'. x \sim_M x' \quad \text{si } M \text{ est une sorte ou un produit}
\end{aligned}$$

$$\begin{aligned}
[\![\langle\rangle]\!] &= \langle\rangle \\
[\![\Gamma, x : A]\!] &= [\![\Gamma]\!], x : A, x' : A', x_P : x \sim_A x'
\end{aligned}$$

Démonstration Dans chacun des cas, il suffit de déplier les définitions de réalisabilité. ☺

Le théorème d'abstraction de la paramétricité est une conséquence du théorème d'adéquation 4.1.8 :

5.2.4 Théorème (Théorème d'abstraction)
Soit $\mathcal{P}$ un CTSI tel que $\mathcal{P}^2 \models$ STRENGTHENING $\wedge$ WEAK-DOWNSTABLE.
Si $\Gamma \vdash_{\mathcal{P}} M : T$, alors

1. $[\![\Gamma]\!] \vdash_{\mathcal{P}^2} M : T$ *et* $[\![\Gamma]\!] \vdash_{\mathcal{P}^2} M' : T'$,
2. *Si* $\Gamma \vdash T : s$, *alors* $[\![\Gamma]\!] \vdash_{\mathcal{P}^2} [\![M]\!] : M \sim_T M'$,
3. *Si* $T \equiv s$, *alors* $[\![\Gamma]\!], x : M, x' : M' \vdash_{\mathcal{P}^2} x \sim_M x' : \lceil s \rceil$.

Démonstration Supposons $\Gamma \vdash_{\mathcal{P}} M : T$, alors on a $\Gamma \vdash_{\mathcal{P}^2} M : T$ (1) avec $\mathrm{lvl}(M) = \mathrm{lvl}(T) = 1$ (car $\mathcal{P} \subseteq \mathcal{P}^2$). En appliquant le morphisme $\cdot \mapsto \lceil \cdot \rceil$, on obtient alors $\lceil \Gamma \rceil \vdash_{\mathcal{P}^2} \lceil M \rceil : \lceil T \rceil$ (2).

1. En utilisant le cas (1) du théorème 4.1.8, on montre que $\|\Gamma\| \vdash_{\mathcal{P}^2} M : T$, puis en l'appliquant une seconde fois, on en déduit $[\![\Gamma]\!] \vdash_{\mathcal{P}^2} M : T$ (3). En appliquant le lemme 4.1.1 à (2), on obtient $\lceil \Gamma \rceil_R \vdash_{\mathcal{P}^2} \lceil M \rceil_R : \lceil T \rceil_R$. Puis le lemme 3.4.8, nous permet d'obtenir : $\lfloor \lceil \Gamma \rceil_R \rfloor \vdash_{\mathcal{P}^2} \lfloor \lceil M \rceil_R \rfloor : \lfloor \lceil T \rceil_R \rfloor$. Or $\lfloor \lceil \Gamma \rceil_R \rfloor \subseteq [\![\Gamma]\!]$. On en déduit donc bien que $[\![\Gamma]\!] \vdash M' : T'$.
2. Supposons $\Gamma \vdash_{\mathcal{P}} T : s$. En appliquant le soulèvement on obtient $\lceil \Gamma \rceil \vdash_{\mathcal{P}^2} \lceil T \rceil : \lceil s \rceil$ (4). En appliquant le cas (2) du théorème 4.1.8 à (2) et (4), on en déduit que :

   $$\|\lceil \Gamma \rceil\| \vdash_{\mathcal{P}^2} \|\lceil M \rceil\| : M \Vdash \lceil T \rceil$$

   Puis, en appliquant le cas (3) du théorème 4.1.8 à (4) on obtient que

   $$\|\lceil \Gamma \rceil\|, x : T \vdash_{\mathcal{P}^2} x \Vdash \lceil T \rceil : \lceil s \rceil$$

Et donc d'après les lemmes de substitution 1.1.23 et 4.1.4 et (3), on en déduit que

$$\|\lceil\Gamma\rceil\| \vdash_{\mathcal{P}^2} M \Vdash \lceil T\rceil : \lceil s\rceil$$

On peut donc appliquer une seconde fois le cas (2) du théorème 4.1.8 :

$$\|\|\lceil\Gamma\rceil\|\| \vdash_{\mathcal{P}^2} \|\|\lceil M\rceil\|\| : \lfloor\|\lceil M\rceil\|\rfloor \Vdash (M \Vdash \lceil T\rceil)$$

On peut alors simplifier $\lfloor\|\lceil M\rceil\|\rfloor = M'$ (lemme 4.1.3) :

$$\|\|\lceil\Gamma\rceil\|\| \vdash_{\mathcal{P}^2} \|\|\lceil M\rceil\|\| : M' \Vdash (M \Vdash \lceil T\rceil)$$

Et en conclure que :

$$[\![\Gamma]\!] \vdash_{\mathcal{P}^2} [\![M]\!] : M \sim_T M'$$

3. Si $T \equiv s$, alors $\lceil T\rceil \equiv \lceil s\rceil$ et le cas (3) du théorème 4.1.8, montre que :

$$\|\lceil\Gamma\rceil\|, x : M \vdash_{\mathcal{P}^2} x \Vdash M : \lceil s\rceil$$

En appliquant une nouvelle fois le cas (3) du théorème 4.1.8, on obtient :

$$\|\|\lceil\Gamma\rceil\|, x : M\|, x' : \lfloor x \Vdash M\rfloor \vdash_{\mathcal{P}^2} x' \Vdash (x \Vdash M) : \lceil s\rceil$$

Or x est une variable de premier niveau, on en déduit donc que $\|\|\lceil\Gamma\rceil\|, x : M\| = \|\|\lceil\Gamma\rceil\|\|, x : M$. De plus, d'après le lemme 4.1.3, on a $\lfloor x \Vdash M\rfloor = M'$. On en déduit que :

$$\|\|\lceil\Gamma\rceil\|\|, x : M, x' : M' \vdash_{\mathcal{P}^2} x' \Vdash (x \Vdash M) : \lceil s\rceil$$

Ce qui nous permet de conclure :

$$[\![\Gamma]\!], x : M, x' : M' \vdash_{\mathcal{P}^2} x \sim_M x' : \lceil s\rceil$$

Paramétricité dépliée pour les inductifs. Comme nous avons pu le voir dans la section 5.1, on étend la paramétricité aux inductifs en associant à toute déclaration d'inductif :

$$\texttt{Ind}(\overrightarrow{x : Q}^p, I : A, \overrightarrow{c : C}^k)$$

l'inductif de second niveau suivant :

$$\texttt{Ind}([\![\overrightarrow{x : Q}^p]\!], I_P : I \sim_A I, \overrightarrow{c_P : c \sim_C c}^k)$$

Par exemple, dans système $\mathcal{T}^2$, on pourra associer à l'inductif nat, le prédicat nat_P de paramétricité suivant (ici dans la syntaxe de Coq pour faciliter la lecture) :

```
Inductive nat_P : nat→ nat→ Prop :=
  | O_P : nat_P O O
  | S_P : forall x y, nat_P x y→ nat_P (S x) (S y).
```

On étend ensuite la transformation $[\![\cdot]\!]$:

– aux identifiants d'inductifs et aux constructeurs :

$$[\![I[\overrightarrow{P}^p]]\!] = I_P[\overrightarrow{P, P', [\![P]\!]}^p] \qquad [\![c[\overrightarrow{P}^p]]\!] = c_P[\overrightarrow{P, P', [\![P]\!]}^p]$$

– aux point-fixes F de la forme $\texttt{fix}\, f : A, \overrightarrow{x : B}^l.M$:

$$[\![F]\!] = \left(\texttt{fix}(f_P : f \sim_A f'), [\![\overrightarrow{x : B}^l]\!].[\![M]\!]\right)[F/x][F'/x']$$

– aux analyses par cas D de la forme $\texttt{case}_I(M, \overrightarrow{P}^p, \lambda\overrightarrow{y : B}^n, i : C.T, \overrightarrow{\lambda\overrightarrow{z : E}^m.F}^k)$:

$$\begin{aligned}[\![D]\!] = \;& \texttt{case}_{I_P}([\![M]\!], \overrightarrow{P, P', [\![P]\!]}^p, && \text{(terme détruit et paramètres)}\\ & \overrightarrow{\lambda y : B, y' : B', y_P : y \sim_B y'}^n, i : C, i' : C', i_P : i \sim_C i'.\Theta \sim_T \Theta', && \text{(type de retour)}\\ & \overrightarrow{\overrightarrow{\lambda z : E, z' : E', z_P : z \sim_E z'}^m.[\![F]\!]}^k) && \text{(branches)}\end{aligned}$$

où $\Theta = \texttt{case}_I(i, \overrightarrow{P}^p, \lambda\overrightarrow{y : B}^n, i : C.T, \overrightarrow{\lambda\overrightarrow{z : E}^m.F}^k)$.

De plus, dans chacun de ces cas, la relation $M \sim_T N$ est définie par $[\![T]\!]\, M\, N$.

Il est possible de voir à l'œuvre ces transformations dans la traduction ci-dessous. On y définit par point-fixe une fonction identité `id` sur les entiers, puis on calcule sa transformation `id_P` :

```
Fixpoint id (n : nat) : nat :=
    match n with
      | O  ⇒  O
      | S k  ⇒  S (id k)
    end.

Fixpoint id_P (n n' : nat) (n_P : nat_P n n') :
              nat_P (id n) (id n') :=
  match n_P in nat_P n n' return nat_P (id n) (id n') with
    | O_P  ⇒  O_P
    | S_P k k' k_P  ⇒  S_P (id k) (id k') (id_P k k' k_P)
  end.
```

5.3 Plongement de la logique engendrée

Comme nous avons pu le voir à la fin de la section 5.1, il est souvent intéressant de plonger la logique $\mathcal{P}^2$ dans des logiques plus expressives pour étudier la paramétricité de $\mathcal{P}$.

Nous considérerons donc des extensions de $\mathcal{P}^2$ qui préservent les programmes mais peuvent étendre la logique engendrée :

5.3.1 Définition (Morphisme préservant le premier niveau)
Soient $\mathcal{P}$ et $\mathcal{L}$ deux CTSI et φ un morphisme tel que $\mathcal{P}^2 \hookrightarrow_\varphi \mathcal{L}$. On dit de φ qu'il préserve le premier niveau si $\mathcal{P} \subseteq \mathcal{L}$ et $\varphi(\mathcal{P}) = \mathcal{P}$.

On peut alors relativiser les notations de paramétricité en suivant ces plongements :

5.3.2 Définition (Paramétricité plongée)
Soient $\mathcal{P}$ et $\mathcal{L}$ deux CTSI et $\mathcal{P}^2 \hookrightarrow_\varphi \mathcal{L}$ un morphisme préservant le premier niveau. On notera $M \sim_T^\varphi N$ la formule $\varphi(M \sim_T N)$, $[\![M]\!]_\varphi$ la formule $\varphi([\![M]\!])$ et $[\![\Gamma]\!]_\varphi$ le contexte $\varphi([\![\Gamma]\!])$.

Le théorème d'abstraction devient alors :

5.3.3 Théorème (Théorème d'abstraction plongée)
Soient $\mathcal{P}$ et $\mathcal{L}$ deux CTSI et φ un morphisme préservant le premier niveau tel que $\mathcal{P}^2 \hookrightarrow_\varphi \mathcal{L}$.
Si $\Gamma \vdash_\mathcal{P} M : T$, alors

1. $[\![\Gamma]\!]_\varphi \vdash_\mathcal{L} M : T$ *et* $[\![\Gamma]\!]_\varphi \vdash_\mathcal{L} M' : T'$,
2. *Si* $\Gamma \vdash_\mathcal{P} T : s$, *alors* $[\![\Gamma]\!]_\varphi \vdash_\mathcal{L} [\![M]\!] : M \sim_T^\varphi M'$,
3. *Si* $T \equiv s$, *alors* $[\![\Gamma]\!]_\varphi, x : M, x' : M' \vdash_\mathcal{L} x \sim_M^\varphi x' : \lceil s \rceil$.

Un des avantages de la paramétricité plongée est qu'il n'est pas nécessaire de calculer la logique intermédiaire $\mathcal{P}^2$ pour l'utiliser. Ainsi, on peut étendre le lemme 5.2.3 de la façon suivante :

5.3.4 Lemme (Paramétricité plongée dépliée)

$$\begin{aligned}
M \sim_s^\varphi N &= M \to N \to \varphi(\lceil s \rceil) \\
M \sim_{\forall x:A.B}^\varphi N &= \forall x : A, x' : A', x_P : x \sim_A^\varphi x'. M\,x \sim_B^\varphi N\,x' \\
M \sim_C^\varphi N &= [\![C]\!]_\varphi\, M\, N \text{ si } C \text{ est une application ou une abstraction}
\end{aligned}$$

$$\begin{aligned}
[\![x]\!]_\varphi &= x_P \\
[\![\lambda x : A.M]\!]_\varphi &= \lambda x : A, x' : A', x_P : x \sim_A^\varphi x'. [\![M]\!]_\varphi \\
[\![(M\,N)]\!]_\varphi &= [\![M]\!]_\varphi\, N\, N'\, [\![N]\!]_\varphi \\
[\![M]\!]_\varphi &= \lambda x : M, x' : M'. x \sim_M^\varphi x' \quad \text{si } M \text{ est une sorte ou un produit}
\end{aligned}$$

$$\begin{aligned}
[\![\langle\rangle]\!]_\varphi &= \langle\rangle \\
[\![\Gamma, x : A]\!]_\varphi &= [\![\Gamma]\!]_\varphi, x : A, x' : A', x_P : x \sim_A^\varphi x'
\end{aligned}$$

La définition suivante présente un cas intéressant de tel plongement. En effet, dans les systèmes suffisamment complets, il est possible d'interpréter la paramétricité dans le système de départ.

5.3.5 Définition (Système réflexif)
On dira d'un CTSI $\mathcal{P}$ qu'il est un système réflexif s'il existe un morphisme φ préservant le premier niveau de $\mathcal{P}^2$ dans $\mathcal{P}$.

5.3.6 Lemme
Soit $\mathcal{P}$ un CTS vérifiant les deux conditions suivantes :
- *Pour tout $(s_1, s_2) \in \mathcal{A}_\mathcal{P}$, alors $(s_1, s_2, s_2) \in \mathcal{R}_\mathcal{P}$.*

– *Pour tout* $(s_1, s_2, s_3) \in \mathcal{R}_\mathcal{P}$*, alors* $(s_1, s_3, s_3) \in \mathcal{R}_\mathcal{P}$.

Alors $\mathcal{P}$ *est réflexif.*

Démonstration Soit $\varphi : \mathcal{S}_{\mathcal{P}^2} \longrightarrow \mathcal{S}_\mathcal{P}$ définie par :

$$\begin{aligned} \varphi(s) &= s \text{ si } s \in \mathcal{S}_\mathcal{P} \\ \varphi(\lceil s \rceil) &= s \text{ si } s \in \mathcal{S}_\mathcal{P} \end{aligned}$$

Montrons que φ est un morphisme de $\mathcal{P}^2$ dans $\mathcal{P}$.

- si $(s, r) \in \mathcal{A}_{\mathcal{P}^2}$ alors :
  - soit $\mathrm{lvl}(s) = \mathrm{lvl}(r) = 1$ et on a alors $(\varphi(s), \varphi(r)) = (s, r) \in \mathcal{A}_\mathcal{P}$.
  - soit $\mathrm{lvl}(s) = \mathrm{lvl}(r) = 2$ et alors il existe $(s', r') \in \mathcal{A}_\mathcal{P}$ tel que $\lceil s' \rceil = s$ et $\lceil r' \rceil = r$. Dans ce cas, on a $(\varphi(s), \varphi(r)) = (s', r') \in \mathcal{A}_\mathcal{P}$.
- si $(s_1, s_2, s_3) \in \mathcal{R}_{\mathcal{P}^2}$ alors on est dans l'un des trois cas suivants :
  - $(s_1, s_2, s_3) \in \mathcal{R}_\mathcal{P}$: on a bien $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) = (s_1, s_2, s_3) \in \mathcal{R}_\mathcal{P}$
  - $(s'_1, s'_2, s'_3) \in \mathcal{R}_\mathcal{P}$ avec $\lceil s'_i \rceil = s_i$ pour $1 \leq i \leq 3$: on a bien $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) = (s'_1, s'_2, s'_3) \in \mathcal{R}_\mathcal{P}$
  - $(s_1, s'_2, s'_3) \in \mathcal{R}_\mathcal{P}$ avec $\lceil s'_3 \rceil = s_2 = s_3$: par hypothèse, on en déduit que $(s_1, s'_3, s'_3) \in \mathcal{R}_\mathcal{P}$ et donc on a bien $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) = (s'_1, s'_3, s'_3) \in \mathcal{R}_\mathcal{P}$
  - $(s_1, s'_2) \in \mathcal{A}_\mathcal{P}$ et $s_2 = s_3 = \lceil s'_2 \rceil$: alors par hypothèse, $(s_1, s'_2, s'_2) \in \mathcal{R}_\mathcal{P}$ et donc $(\varphi(s_1), \varphi(s_2), \varphi(s_3)) = (s_1, s'_2, s'_2) \in \mathcal{R}_\mathcal{P}$.
- si $(s, r) \in \mathcal{C}_{\mathcal{P}^2}$, on montre de façon similaire au cas axiome que $(\varphi(s), \varphi(r)) \in \mathcal{C}_\mathcal{P}$ (tout comme la relation d'axiome, la relation de cumulativité de $\mathcal{P}^2$ ne compare que deux sortes de même niveau). ☺

5.3.7 Exemple

Les systèmes $\lambda\star$*,* $\lambda\Pi$*,* $\lambda\Pi_\omega$ $\lambda\Pi^2$*,* CC*,* CC_α*,* $\lambda\star_\alpha$ *sont réflexifs.*

Les systèmes réflexifs sont en quelque sorte capables d'interpréter leur propre théorie de la paramétricité. À titre d'exemple, on peut utiliser CC dans lequel on utilise la sorte $\star$ pour représenter à la fois les preuves et les programmes.

5.3.8 Exemple

On a :

$$\vdash_{\mathrm{CC}} \forall n : \mathsf{nat}. n \sim^\varphi_{\mathsf{nat}} n \to n \overset{ext^\varphi}{=}_{\mathsf{nat}} n\, \mathsf{nat}\, \mathsf{S}\, \overline{0}$$

et donc $\vdash_{\mathrm{CC}} t : \mathsf{nat}$ *implique* $\vdash_{\mathrm{CC}} t \overset{ext^\varphi}{=}_{\mathsf{nat}} t\, \mathsf{nat}\, \mathsf{S}\, \overline{0}$*. Où on a adopté les notations suivantes dans* CC *:*

$$\begin{aligned} \mathsf{nat} &= \forall \alpha : \star. (\alpha \to \alpha) \to \alpha \to \alpha \\ n \overset{ext^\varphi}{=}_{\mathsf{nat}} m &= \varphi(n \overset{ext}{=}_{\mathsf{nat}} m) = \forall \alpha : \star, f : \star \to \star, x : \star, X : \alpha \to \star, h : X\,(n\,\alpha\,f\,x) \to X\,(m\,\alpha\,f\,x) \\ \mathsf{S} &= \lambda n : \mathsf{nat}, \alpha : \star, f : \alpha \to \alpha, x : \alpha. f\,(n\,\alpha\,f\,x) \\ \overline{0} &= \lambda n : \mathsf{nat}, \alpha : \star, f : \alpha \to \alpha, x : \alpha. x \end{aligned}$$

Démonstration – On a $\mathcal{F}^2 \subseteq \mathrm{CC}^2$, on peut donc utiliser le lemme 4.2.33 pour prouver

$$\vdash_{\mathrm{CC}^2} \forall n : \mathsf{nat}.n \sim_{\mathsf{nat}} n \to n \overset{\mathrm{ext}}{=}_{\mathsf{nat}} n\,\mathsf{nat}\,\mathsf{S}\,\overline{0}$$

puis en appliquant le morphisme φ on en déduit :

$$\vdash_{\mathrm{CC}} \forall n : \mathsf{nat}.n \sim^{\varphi}_{\mathsf{nat}} n \to n \overset{\mathrm{ext}^{\varphi}}{=}_{\mathsf{nat}} n\,\mathsf{nat}\,\mathsf{S}\,\overline{0}$$

– D'après le théorème d'abstraction plongée, on a $\vdash_{\mathrm{CC}} t \sim^{\varphi}_{\mathsf{nat}} t$; on en déduit donc bien :

$$\vdash_{\mathrm{CC}} t \overset{\mathrm{ext}^{\varphi}}{=}_{\mathsf{nat}} t\,\mathsf{nat}\,\mathsf{S}\,\overline{0}$$

☺

Comparaison avec la relation de paramétricité de Bernardy *et. al.* Dans l'article [BJP10], la transformation de paramétricité présentée correspond à notre théorème d'abstraction plongée restreint aux PTS réflexifs vérifiant la propriété $(*)$: pour toute sorte $s \in \mathcal{S}_{\mathcal{P}}$, il existe une sorte r telle que $(s,r) \in \mathcal{A}_{\mathcal{P}}$. Ainsi, cette propriété n'est pas vérifiée par les systèmes du cube par exemple. Cette condition permet aux auteurs de [BJP10] de définir la transformation de paramétricité plus simple suivante :

$$\begin{array}{rcl}
[\![x]\!]_* & = & x_P \\
[\![s]\!]_* & = & \lambda x : s, x' : s.x \to x' \to \varphi(s) \\
[\![\lambda x : A.M]\!]_* & = & \lambda x : A, x' : A', x_P : [\![A]\!]_*\, x\, x'.[\![M]\!]_* \\
[\![(M\,N)]\!]_* & = & [\![M]\!]_*\, N\, N'\, [\![N]\!]_* \\
[\![\forall x : A.B]\!]_* & = & \lambda f : \forall x : A.B, f' : \forall x : A'.B'.\forall x : A, x' : A', x_P : [\![A]\!]_*\, x\, x'.[\![B]\!]_*\,(f\,x)\,(f\,x') \\
 & & \\
[\![\langle\rangle]\!]_* & = & \langle\rangle \\
[\![\Gamma, x : A]\!]_* & = & [\![\Gamma]\!]_*, x : A, x' : A', x_P : [\![A]\!]_*\, x\, x'
\end{array}$$

Ici, la relation $M \sim^*_T N$ est une notation pour le terme $[\![T]\!]\, M\, N$. La différence principale entre la définition de Bernardy et la paramétricité plongée, présentée plus haut, tient au fait qu'elle crée des rédexs superflus. Comme par exemple :

$$\begin{array}{rcl}
[\![\lambda\alpha : s, h : \alpha.h]\!]_* & = & \lambda\alpha : s, \alpha' : s.\alpha_P : [\![\boldsymbol{s}]\!]_*\, \boldsymbol{\alpha}\, \boldsymbol{\alpha'}.h_P \\
 & \trianglerighteq & \lambda\alpha : s, \alpha' : s.\alpha_P : \alpha \to \alpha' \to \lceil\star\rceil.h_P
\end{array}$$

Si $\mathcal{P}$ ne vérifie pas la condition $(*)$, alors ces abstractions superflues ne sont pas typables. C'est donc au prix d'une légère complication des définitions de paramétricité que nous pouvons "réduire à la volée" ces rédexs, et ainsi capturer les systèmes qui ne vérifient pas $(*)$.

5.4 Plongements pour CIC, CIC$^-$ et CIC$^+$

On peut vérifier que les systèmes CIC, CIC$^-$ et CIC$^+$ sont réflexifs pour le morphisme φ de CTSI, défini par :

$$\begin{aligned} \varphi(s) &= s \text{ si } s \in \mathcal{S}_\mathcal{P} \\ \varphi(\lceil s \rceil) &= s \text{ si } s \in \mathcal{S}_\mathcal{P} \end{aligned}$$

où $\mathcal{P} = \text{CIC}, \text{CIC}^-$ ou CIC^+.

Ce plongement de $\mathcal{P}^2$ dans $\mathcal{P}$ est correct, mais il peut désarçonner les utilisateurs de Coq. On effet, ces derniers sont habitués à représenter les relations par des prédicats à valeur dans la sorte `Prop` des propositions. En utilisant ce plongement, un type T de sorte s générera une relation d'arité $T \sim_s T' = T \to T' \to s$. Par exemple, dans CIC, les types informatifs sont encodés par des inductifs de sorte $\texttt{Type}_i$; ainsi, l'inductif `nat` est de type $\texttt{Type}_0$. L'inductif $\texttt{nat}_\mathcal{P}$ engendré par `nat` est alors un inductif d'arité $\texttt{nat} \to \texttt{nat} \to \texttt{Type}_0$. Cela pose plusieurs problèmes :

- Cette relation ne pourra pas être utilisée par les lemmes et définitions déjà prouvés et qui attendent comme hypothèse une relation à valeur dans `Prop`,
- Il est possible que ces définitions utilisent l'imprédicativité de `Prop` et qu'elles ne puissent pas être adaptées pour être utilisées avec des relations à valeur dans une sorte prédicative.
- Les preuves de paramétricité ne seront pas effacées par le mécanisme d'extraction.

C'est donc afin d'obtenir une relation de paramétricité plus fidèle à la philosophie de Coq, que nous allons utiliser le plongement suivant pour $\mathcal{P} = \text{CIC}, \text{CIC}^-, \text{CIC}^+$:

$$\begin{aligned} \chi_\mathcal{P}(s) &= s \text{ si } s \in \mathcal{S}_\mathcal{P} \\ \chi_\mathcal{P}(\lceil s \rceil) &= \texttt{Prop} \text{ si } s \in \mathcal{S}_\mathcal{P} \text{ et } s \text{ minimal pour } \mathcal{A}_\mathcal{P} \\ \chi_\mathcal{P}(\lceil s \rceil) &= s \text{ si } s \in \mathcal{S}_\mathcal{P} \text{ et } s \text{ n'est pas minimal pour } \mathcal{A}_\mathcal{P} \end{aligned}$$

Ce qui nous donne dans chacun de ces systèmes :

CIC	CIC$^-$	CIC$^+$
$\chi_{\text{CIC}}(\lceil \texttt{Prop} \rceil) = \texttt{Prop}$	$\chi_{\text{CIC}^-}(\lceil \texttt{Prop} \rceil) = \texttt{Prop}$	$\chi_{\text{CIC}^+}(\lceil \texttt{Prop} \rceil) = \texttt{Prop}$
$\chi_{\text{CIC}}(\lceil \texttt{Type}_0 \rceil) = \texttt{Prop}$	$\chi_{\text{CIC}^-}(\lceil \texttt{Set} \rceil) = \texttt{Prop}$	$\chi_{\text{CIC}^+}(\lceil \texttt{Set}_i \rceil) = \texttt{Prop}$
$\chi_{\text{CIC}}(\lceil \texttt{Type}_{i+1} \rceil) = \texttt{Type}_{i+1}$	$\chi_{\text{CIC}^-}(\lceil \texttt{Type}_i \rceil) = \texttt{Type}_i$	$\chi_{\text{CIC}^+}(\lceil \texttt{Type}_i \rceil) = \texttt{Type}_i$

Dans ces systèmes, les types de données informatifs habitent les sortes minimales pour la relation $\mathcal{A}$. Par exemple, le type $\mathsf{list}[\mathsf{nat}]$ des listes d'entiers habitent les sortes $\texttt{Type}_0$, `Set` et $\texttt{Set}_0$ selon que le système considéré est CIC, CIC$^-$ ou CIC$^+$. Ainsi dans ces systèmes on a :

$$\vdash_{\text{CIC}} \mathsf{list}[\mathsf{nat}] : \texttt{Type}_0 \qquad \vdash_{\text{CIC}^-} \mathsf{list}[\mathsf{nat}] : \texttt{Set} \qquad \vdash_{\text{CIC}^+} \mathsf{list}[\mathsf{nat}] : \texttt{Set}_0$$

et le théorème d'abstraction nous permettra de dériver pour $\mathcal{P} = \text{CIC}, \text{CIC}^-, \text{CIC}^+$:

$$\vdash_\mathcal{P} [\![\mathsf{list}[\mathsf{nat}]]\!]_\chi : \mathsf{list}[\mathsf{nat}] \to \mathsf{list}[\mathsf{nat}] \to \texttt{Prop}$$

Nous avons introduit, dans l'article [KL12], le système CIC^+ car le système CIC est défaillant du point de vue de la représentation des types informatifs. En effet, les types de données polymorphes, comme par exemple le type de l'identité $\forall\alpha : \texttt{Type}_0 .\alpha \to \alpha$, n'habitent pas la sorte $\texttt{Type}_0$ mais une sorte plus élevée (ici $\texttt{Type}_1$ pour le type de l'identité).

Puisque, dans CIC, les types de données polymorphes n'habitent pas une sorte minimale pour la relation $\mathcal{A}_{\text{CIC}}$, les relations induites par ces types ne sont pas à valeur dans la sorte des propositions. Comme par exemple :

$$\vdash_{\text{CIC}} [\![\forall\alpha : \texttt{Type}_0 .\alpha \to \alpha]\!]_\chi : (\forall\alpha : \texttt{Type}_0 .\alpha \to \alpha) \to (\forall\alpha : \texttt{Type}_0 .\alpha \to \alpha) \to \texttt{Type}_1$$

C'est pourquoi les systèmes CIC^+ et CIC^- sont plus satisfaisants du point de vue de la paramétricité puisqu'ils arrivent à contenir les types polymorphes dans des sortes minimales pour la relation $\mathcal{A}$:

$$\vdash_{\text{CIC}^-} \forall\alpha : \texttt{Set} .\alpha \to \alpha \quad : \quad \texttt{Set} \qquad \vdash_{\text{CIC}^+} \forall\alpha : \texttt{Set}_i .\alpha \to \alpha \quad : \quad \texttt{Set}_{i+1}$$

Et donc les relations induites sont bien à valeur dans $\texttt{Prop}$:

$$\vdash_{\text{CIC}^-} [\![\forall\alpha : \texttt{Set} .\alpha \to \alpha]\!]_\chi \quad : \quad (\forall\alpha : \texttt{Set} .\alpha \to \alpha) \to (\forall\alpha : \texttt{Set} .\alpha \to \alpha) \to \texttt{Prop}$$

et :

$$\vdash_{\text{CIC}^+} [\![\forall\alpha : \texttt{Set}_i .\alpha \to \alpha]\!]_\chi \quad : \quad (\forall\alpha : \texttt{Set}_i .\alpha \to \alpha) \to (\forall\alpha : \texttt{Set}_i .\alpha \to \alpha) \to \texttt{Prop}$$

5.4.1 Lemme
Soit $\mathcal{P} \in \{\text{CIC}, \text{CIC}^-, \text{CIC}^+\}$.
La fonction $\chi_\mathcal{P}$ *est un morphisme de CTS préservant le premier niveau de* $\mathcal{P}^2$ *dans* $\overline{\mathcal{P}}$.

Démonstration On montre le résultat pour $\mathcal{P} = \text{CIC}^+$, les autres vérifications sont similaires.

- Soit $(s, r) \in \mathcal{A}_{\mathcal{P}^2}$, montrons que $(\chi(s), \chi(r)) \in \mathcal{A}_{\overline{\mathcal{P}}}$. On distingue les deux cas possibles :
  - $(s, r) \in \mathcal{A}_\mathcal{P}$: dans ce cas, on a $(\chi(s), \chi(r)) = (s, r) \in \mathcal{A}_\mathcal{P} \subseteq \mathcal{A}_{\overline{\mathcal{P}}}$.
  - $(s', r') \in \mathcal{A}_\mathcal{P}$ et $\lceil s' \rceil = s$ et $\lceil r' \rceil = r$, on distingue selon les valeurs possibles de (s', r') :
    - $s' = \texttt{Prop}, r' = \texttt{Type}_1$: $(\chi(s), \chi(r)) = (\texttt{Prop}, \texttt{Type}_1) \in \mathcal{A}_\mathcal{P} \subseteq \mathcal{A}_{\overline{\mathcal{P}}}$,
    - $s' = \texttt{Set}_i, r' = \texttt{Type}_j$ avec $i < j$: $(\chi(s), \chi(r)) = (\texttt{Prop}, \texttt{Type}_j) \in \mathcal{A}_{\overline{\mathcal{P}}}$,
    - $s' = \texttt{Type}_i, r' = \texttt{Type}_j$ avec $i < j$: $(\chi(s), \chi(r)) = (\texttt{Prop}, \texttt{Type}_j) \in \mathcal{A}_{\overline{\mathcal{P}}}$.
- Soit $(s, r, r) \in \mathcal{R}_{\mathcal{P}^2}$ (on rappelle que $\mathcal{P}^2 \models$ HOMOGENEOUS), montrons que $(\chi(s), \chi(r), \chi(r)) \in \mathcal{R}_{\overline{\mathcal{P}}}$. on distingue tous les cas possibles de la construction $\mathcal{P}^2$:
  - $(s, r, r) \in \mathcal{R}_\mathcal{P}$: Dans ce cas $(\chi(s), \chi(r), \chi(r)) = (s, r, r) \in \mathcal{R}_\mathcal{P} \subseteq \mathcal{R}_{\mathcal{P}^2} \subseteq \mathcal{R}_{\overline{\mathcal{P}^2}}$.
  - $(s', r', r') \in \mathcal{R}_\mathcal{P}$ avec $\lceil s' \rceil = s$, $\lceil r' \rceil = r$: on distingue selon les valeurs possibles de s' et r' :
    - $r' = \texttt{Prop}$ ou $r' = \texttt{Set}_i$: $(\chi(s), \chi(r), \chi(r)) = (\chi(s), \texttt{Prop}, \texttt{Prop}) \in \mathcal{R}_\mathcal{P} \subseteq \mathcal{R}_{\overline{\mathcal{P}}}$,
    - $s' = \texttt{Set}_i$ ou $s' = \texttt{Prop}$ et $r' = \texttt{Type}_j$ avec $i \leq j$:

      $$(\chi(s), \chi(r), \chi(r)) = (\texttt{Prop}, \texttt{Type}_j, \texttt{Type}_j) \in \mathcal{R}_\mathcal{P} \subseteq \mathcal{R}_{\overline{\mathcal{P}}}$$

    - $s' = \texttt{Type}_i, r' = \texttt{Type}_j$ avec $i \leq j$: $(\chi(s), \chi(r), \chi(r)) = (s', r', r') \in \mathcal{R}_\mathcal{P} \subseteq \mathcal{R}_{\mathcal{P}^2}$.
  - $(s, r', r') \in \mathcal{R}_\mathcal{P}$ avec $\lceil r' \rceil = r$: On distingue selon les valeurs possibles de r' :
    - Si $r' = \texttt{Prop}$ ou $r' = \texttt{Set}_i$, alors $(\chi(s), \chi(r), \chi(r)) = (\chi(s), \texttt{Prop}, \texttt{Prop}) \in \mathcal{R}_\mathcal{P} \subseteq \mathcal{R}_{\overline{\mathcal{P}}}$.
    - Si $r' = \texttt{Type}_i$, on a $(\chi(s), \chi(r), \chi(r)) = (s, r', r') \in \mathcal{R}_\mathcal{P} \subseteq \mathcal{R}_{\overline{\mathcal{P}}}$.

- $(s, r') \in \mathcal{A}_\mathcal{P}$ avec $\lceil r' \rceil = r$: le système $\mathcal{P}$ vérifie la condition de réflexivité vue plus haut, à savoir : $(s, r') \in \mathcal{A}_\mathcal{P}$ implique $(s, r', r') \in \mathcal{R}_\mathcal{P}$. Or on a nécessairement $r' = \texttt{Type}_i$ et donc $(\chi(s), \chi(r), \chi(r)) = (s, r', r') \in \mathcal{R}_\mathcal{P} \subseteq \mathcal{R}_{\mathcal{P}^2}$.

– Soit $(s, r) \in \mathcal{C}_{\mathcal{P}^2}$, montrons que $\chi(s) \preccurlyeq_\mathcal{P} \chi(r)$. On distingue selon les cas :
  - Si $(s, r) \in \mathcal{C}_\mathcal{P}$, alors $(\chi(s), \chi(r)) = (s, r) \in \mathcal{C}_\mathcal{P}$ et donc $\chi(s) \preccurlyeq_\mathcal{P} \chi(r)$.
  - Si $(s, r) = (\lceil \texttt{Type}_i \rceil, \lceil \texttt{Type}_j \rceil)$, alors $(\chi(s), \chi(r)) = (\texttt{Type}_i, \texttt{Type}_j) \in \mathcal{C}_\mathcal{P}$ et donc $\chi(s) \preccurlyeq_\mathcal{P} \chi(r)$.
  - Si $(s, r) = (\lceil \texttt{Set}_i \rceil, \lceil \texttt{Set}_j \rceil)$, alors$\chi(s) = \chi(r) = \texttt{Prop}$ et donc $\chi(s) \preccurlyeq_\mathcal{P} \chi(r)$. ☺

Hélas, ces morphismes ne sont pas des morphismes de CTSI, le fait de transformer les arités de sorte `Set` en arités de sorte `Prop`, enverra certaines éliminations fortes vers des éliminations fortes non-autorisées.

On peut apercevoir ce phénomène en essayant de typer la traduction de la fonction `n_or_b` suivante :

```
Inductive nat_P : nat→ nat→ Prop :=
  | O_P : nat_P O O
  | S_P : forall x y, nat_P x y→ nat_P (S x) (S y).

Inductive bool_P : bool→ bool→ Prop :=
  | true_P : bool_P true true
  | false_P : bool_P false false.

Definition n_or_b (b : bool) : Set :=
  match b with
   | true  ⇒  nat
   | false  ⇒  bool
  end.

Definition n_or_b_P (b b' : bool) (b_P : bool_P b b') :
      n_or_b b→ n_or_b b'→ Prop  :=
  match b_P in bool_P b b' return n_or_b b→ n_or_b b'→ Prop  with
   | true_P  ⇒  nat_P
   | false_P  ⇒   bool_P
  end. (* ERREUR *)
```

Au moment de typer le terme `n_or_b_P` le système retourne l'erreur suivante :

```
Error: Incorrect elimination of "b_P" in the inductive type "bool_P":
       the return type has sort "Type" while it should be "Prop".
Elimination of an inductive object of sort Prop is not allowed
on a predicate in sort Type because proofs can be eliminated only
to build proofs.
```

En effet, le type de l'objet `b_P` est donné par l'inductif `bool_P` qui n'est pas un petit inductif logique (car il a deux constructeurs). C'est pourquoi l'élimination n'est pas autorisée. Seules les éliminations fortes posent ici problème. Soit $\mathcal{P} \in \{\text{CIC}, \text{CIC}^-, \text{CIC}^+\}$; on peut définir le système $\mathcal{P}_{\text{we}}$ comme ayant les mêmes paramètres que $\mathcal{P}$, à l'exception de $\mathcal{E}_\mathcal{P}$ qui est défini de la façon suivante :

$$\mathcal{E}_{\mathcal{P}_{\text{we}}} = \mathcal{E}_\mathcal{P} \setminus \{(I, \texttt{Type}_i) | I \text{ n'est pas un petit inductif logique et il est de sorte minimale pour } \mathcal{A}_\mathcal{P} \}$$

En d'autres termes, $\mathcal{P}_{\text{we}}$ est obtenu, à partir de $\mathcal{P}$, en interdisant les éliminations fortes des inductifs de sorte minimale qui ne sont pas des petits inductifs logiques.

On peut alors vérifier que :

5.4.2 Lemme
Soit $\mathcal{P} \in \{\text{CIC}, \text{CIC}^-, \text{CIC}^+\}$.
La fonction $\chi_{\mathcal{P}}$ *est un morphisme de CTSI préservant le premier niveau de* $(\mathcal{P}_{\text{we}})^2$ *dans* $\overline{\mathcal{P}_{\text{we}}}$.

On en déduit alors le théorème d'abstraction suivant :

5.4.3 Théorème (Abstraction dans `Prop` avec élimination forte interdite)
Soient $\mathcal{P} \in \{\text{CIC}, \text{CIC}^-, \text{CIC}^+\}$ *et soit le* $\chi = \chi_{\mathcal{P}}$ *le morphisme défini précédemment.*
Si $\Gamma \vdash_{\mathcal{P}_{\text{we}}} M : T$, *alors*

1. $[\![\Gamma]\!]_\chi \vdash_{\mathcal{P}_{\text{we}}} M : T$ *et* $[\![\Gamma]\!]_\chi \vdash_{\mathcal{P}_{\text{we}}} M' : T'$,
2. *Si* $\Gamma \vdash T : s$, *alors* $[\![\Gamma]\!]_\chi \vdash_{\mathcal{P}_{\text{we}}} [\![M]\!] : M \sim_T^\chi M'$,
3. *Si* $T \equiv s$, *alors* $[\![\Gamma]\!]_\chi, x : M, x' : M' \vdash_{\mathcal{P}_{\text{we}}} x \sim_M^\chi x' : \lceil s \rceil$.

L'élimination forte est un outil essentiel de Coq, puisqu'elle permet, comme nous avons pu le voir au chapitre 2, de dériver $0 \neq 1$ et qu'elle est souvent utilisée dans les programmes avec types dépendants. Il est donc primordial de pouvoir dépasser cette restriction, si on veut pouvoir appliquer la théorie de la paramétricité au système Coq.

Nous allons à présent voir que dans bien des cas on peut réussir à traduire les éliminations fortes par des éliminations autorisées. Regardons sur un exemple comment on peut contourner cette restriction.

```
Inductive NB : Set :=
  | N : nat→ NB
  | B : bool→ NB.

Inductive vector : nat→ Set :=
  | vector_nil : vector O
  | vector_succ : forall n, nat→ vector n→ vector (S n).

Definition f (x : NB) : Set :=
  match x with
  | N n  ⇒  vector n
  | B b  ⇒  nat
  end.
```

L'inductif `NB` implémente la somme disjointe du type des entiers et des booléens. Nous cherchons à traduire la fonction `f` qui associe à un élément x de type `NB` un type de sorte `Set`. Elle retourne le type des vecteurs de taille n si x est de la forme $N\,n$ et le type des entiers si x est de la forme $B\,b$ (on notera que b n'est pas utilisé dans cette branche). Dans cet exemple, nous avons choisi de retourner un type dépendant (ici `vector`) et un type non-dépendant (`nat`), afin d'illustrer la différence entre le cas dépendant et le cas non-dépendant. Mais nous aurions pu adapter cet exemple en remplaçant `vector` n et `nat` par n'importe quel autre habitant de `Set`, sans que cela ne pose problème.

Si l'on appliquait directement la traduction de paramétricité on obtiendrait le terme :

```
Definition f_P (x x' : NB) (x_P : NB_P x x') :=
  match x_P with
  | N_P n n' n_P  ⇒  vector_P n n' n_P
  | B_P b b' b_P  ⇒  nat_P
  end.
```

Or nous avons vu qu'une telle définition est refusée par le système (car `x_P` est une preuve d'un prédicat inductif de sorte `Prop` qui n'est pas un petit inductif logique). Nous allons donc proposer une traduction alternative pour f_P qui procède en échangeant la destruction de x_P par deux destructions imbriquées de x et x'. Ce qui a pour effet de produire k^2 branches (où k est le nombre de constructeurs de l'inductif, ici $k = 2$) ; mais seules k d'entre elles sont effectivement possibles :

```
Program Definition f_P (x x' : NB) (x_P : NB_P x x') :=
  match x with
  | N n  ⇒  match x' with
    | N n'  ⇒  let n_P := inv n n' x_P in vector_P n n' n_P
    | B b'  ⇒  absurd (vector n→ nat→ Prop) (abs12 n b' x_P)
    end
  | B b  ⇒  match x' with
    | N n'  ⇒  absurd (nat→ vector n'→ Prop) (abs21 b n' x_P)
    | B b  ⇒  nat_P
    end
  end.
```

où les termes suivants sont implémentés grâce à des éliminations autorisées :

$$\begin{aligned}
\texttt{inv} &: \forall (n\, n' : \mathsf{nat}).\mathsf{NB}_P\,(\mathrm{N}\,n)\,(\mathrm{N}\,n') \to \mathsf{nat}_P\ n\, n' \\
\texttt{abs}_{12} &: \forall (n : \mathsf{nat})(b' : \mathsf{bool}).\mathsf{NB}_P\,(\mathrm{N}\,n)\,(\mathrm{B}\,b') \to \mathsf{False} \\
\texttt{abs}_{21} &: \forall (b : \mathsf{bool})(n' : \mathsf{nat}).\mathsf{NB}_P\,(\mathrm{B}\,b)\,(\mathrm{N}\,n') \to \mathsf{False} \\
\texttt{absurd} &: \forall (\alpha : \mathsf{Type}_1).\,\mathsf{False} \to \alpha
\end{aligned}$$

Les termes $\texttt{abs}_{12}$ et $\texttt{abs}_{21}$ permettent de déduire de x_P que le cas où x et x' ne commencent pas par le même constructeurs est impossible. Et le terme `inv` permet de reconstruire à partir de x_P une preuve que n et n' sont en relation pour nat_P (on pourrait faire de même pour construire une preuve b_P que b et b' sont en relation pour bool_P mais ce n'est pas nécessaire car b_P n'est pas liée dans cette branche).

À titre indicatif, nous donnons leur implémentation sur la figure 5.1 : les termes `T1` et `T2` donnent les types de retour des éliminations autorisées. Ces types de retour permettent d'avoir à prouver des propositions triviales dans les cas impossibles des éliminations qui les suivent.

On remarque que cet exemple se déroule correctement car les arguments des constructeurs de NB sont de types `Prop` ou `Set` ; NB est un petit inductif. Il ne serait pas possible d'effectuer la même "astuce" pour traduire la destruction de l'inductif de CIC$^+$ suivant :

$$\texttt{Ind}(\mathsf{box\_set}_i : \mathsf{Set}_{i+1}, \mathsf{close}_i : \mathsf{Set}_i \to \mathsf{box\_set}_i)$$

```
Definition T1 (x x' : NB) : Prop :=
  match x, x' with
   | N n, N n' ⇒ nat_P n n'
   | _, _ ⇒ True
  end.

Definition inv n n' (n_P : NB_P (N n) (N n'))
               : nat_P n n' :=
  match n_P in NB_P x x' return T1 x x' with
   | N_P n n' n_P ⇒ n_P
   | B_P _ _ _ ⇒ I (* preuve de True *)
  end.

Definition T2 (x x' : NB) : Prop :=
  match x, x' with
   | N _, N _ ⇒ True
   | B _, B _ ⇒ True
   | _, _ ⇒ False
  end.

Definition abs12 (n : nat) (b' : bool)
                 (h : NB_P (N n) (B b')) : False :=
   match h in NB_P x x' return T2 x x' with
     | N_P _ _ _ ⇒ I (* preuve de True *)
     | B_P _ _ _ ⇒ I (* preuve de True *)
   end.

Definition abs21 (b : bool) (n' : nat)
                 (h : NB_P (B b) (N n')) : False :=
   match h in NB_P x x' return T2 x x' with
     | N_P _ _ _ ⇒ I (* preuve de True *)
     | B_P _ _ _ ⇒ I (* preuve de True *)
   end.

Definition absurd (X : Type) (b : False) : X :=
   match b return X with end.
```

FIGURE 5.1 – Implémentation des fonctions `inv`, $\texttt{abs}_{12}$, $\texttt{abs}_{21}$ et `absurd`.

En effet, cet inductif se traduit par :

$$\begin{array}{l}\texttt{Ind}\,([\![\mathsf{box\_set}_i]\!] : \mathsf{box\_set}_i \to \mathsf{box\_set}_i \to \texttt{Prop},\\ \qquad [\![\mathsf{close}_i]\!] : \forall(A\,A' : \texttt{Set}_i).(A \to A' \to \texttt{Prop}) \to [\![\mathsf{box\_set}_i]\!]\,(\mathsf{close}_i\,A)\,(\mathsf{close}_i\,A'))\end{array}$$

Ce n'est pas un petit inductif car le type de l'argument de close est de type $\texttt{Type}_{i+1}$ et n'est pas minimal pour la relation $\mathcal{A}$. Ainsi, il n'est pas possible de programmer la fonction suivante :

$$\texttt{inv} : \forall(\alpha\,\alpha' : \texttt{Set}_i).\mathsf{box\_set}_P\,(\mathsf{close}\,\alpha)\,(\mathsf{close}\,\alpha') \to \alpha \sim^{\chi}_{\texttt{Set}_i} \alpha'$$

Car $\mathsf{box\_set}_P\,(\mathsf{close}\,\alpha)\,(\mathsf{close}\,\alpha')$ est de type Prop et $\alpha \sim^{\chi}_{\texttt{Set}_i} \alpha'$ est de type $\texttt{Type}_{i+1}$. Et donc, l'élimination de l'argument de type $\mathsf{box\_set}_P\,(\mathsf{close}\,\alpha)\,(\mathsf{close}\,\alpha')$ n'est pas autorisée pour construire un objet de type $\alpha \sim^{\chi}_{\texttt{Set}_i} \alpha'$.

Ainsi, si on modifie la définition de $\mathcal{P}_{\text{small}}$ de façon analogue à $\mathcal{P}_{\text{we}}$ en posant :

$$\mathcal{E}_{\mathcal{P}_{\text{small}}} = \mathcal{E}_{\mathcal{P}} \setminus \{(I, \texttt{Type}_i) | I \text{ n'est pas un petit inductif et il est de sorte minimale pour } \mathcal{A}_{\mathcal{P}}\}$$

On remarque que :

- Cette restriction est exactement celle présente pour l'élimination des inductifs de Set dans CIC^-,
- De même, dans CIC, la sorte $\texttt{Type}_0$ ne contient uniquement que des petits inductifs. En effet, si un inductif de sorte $\texttt{Type}_i$ a un constructeur dont un des types est de sorte $\texttt{Type}_j$, alors on a $j \leq i$; et donc si $j \neq 0$ alors $i \neq 0$.

On en déduit donc :

5.4.4 Lemme

$$\text{CIC}^-_{\text{small}} = \text{CIC}^- \qquad \text{CIC}_{\text{small}} = \text{CIC}$$

Par contre, dans CIC^-, cette construction impose une nouvelle restriction. Néanmoins, en pratique, la plupart des termes utiles ne sont pas concernés par cette restriction puisque les petits inductifs contiennent la grande majorité des types de données utilisés (cette restriction était présente jusqu'à la version 8.0 de Coq).

Parmi les éliminations interdites dans CIC^+ par cette restriction, on trouve l'encodage de Peter Aczel qui permet de représenter les ensembles de la théorie des ensembles intuitionniste (voir [Wer97, Acz86]) :

$$\texttt{Ind}(\mathsf{Ens} : \texttt{Set}_{i+1}, \mathsf{sup} : \forall\alpha : \texttt{Set}_i\,.(\alpha \to \mathsf{Ens}) \to \mathsf{Ens})$$

Cet inductif est autorisé dans CIC^+ mais il ne peut pas être détruit pour fabriquer un objet dont le type est une sorte minimale dans $\text{CIC}^+_{\text{small}}$.

Bien que nous ne l'avons pas formalisée dans ses détails, nous sommes convaincu qu'il est possible de généraliser l'exemple de la figure 5.1 au cas général. Nous donnons une ébauche de cette modification de la transformation de paramétricité dans le cas case à la figure 5.2. Cette dernière ne s'adapte pas exactement dans notre cadre car cette modification à besoin du contexte dans lequel les termes sont définis pour pouvoir récupérer :

- la sorte du type de retour du case (la sorte s dans laquelle habite T dans les notations de la figure 5.2),
- et la valeur des arguments de l'inductif qui apparaît dans le type principal de l'objet détruit (ie. si M est de type principal $I[\overrightarrow{P}^p]\,\overrightarrow{G}^n$, les $\overrightarrow{G}^n$ est la valeur des arguments de l'inductif I).

Nous utilisons donc ici le fait que les systèmes CIC, CIC^- et CIC^+ satisfont la propriété d'existence d'un type principal.

On modifie la définition de $[\![\cdot]\!]_\chi$ pour détruire un petit inductif I avec p paramètres, n arguments et k constructeurs

$$[\![\mathtt{case}_I(M, \overrightarrow{P}^p, \lambda\overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\,\overrightarrow{y}^n.T, \overrightarrow{\lambda z : E^m.F}^k)]\!]_\chi = \zeta\,[\![M]\!]$$

avec :

$$\begin{aligned}
\zeta = \mathtt{case}_I \Big(& M, \overrightarrow{P}^p, \lambda\overrightarrow{y : B}^n, i : I[\overrightarrow{P}^p]\,\overrightarrow{y}^n. \quad \forall i_P : i \sim^\chi_{I[\overrightarrow{P}^p]\,\overrightarrow{y}^n} M'.(\Theta \sim^\chi_T \Theta')[M'/i', \overrightarrow{G'}^n/\overrightarrow{y'}^n], \\
& \lambda\overrightarrow{x : E_1}^{m_1}.\,\mathtt{case}\,\big(M', \lambda\overrightarrow{y' : B'}^{m_1}, i' : I[\overrightarrow{P'}^p]\,\overrightarrow{y'}^n. \quad \forall i_P : c_1[\overrightarrow{P}^p]\,\overrightarrow{x}^{m_1} \sim^\chi_{I[\overrightarrow{P}^p]\,\overrightarrow{y}^n} i'.\Theta \sim^\chi_T \Theta', \\
& \quad \lambda\overrightarrow{x' : E'_1}^{m_1}. \quad \lambda z_P : c_1[\overrightarrow{P}^p]\,\overrightarrow{x}^{m_1} \sim^\chi_{I[\overrightarrow{P}^p]\,\overrightarrow{y}^n} c_1[\overrightarrow{P'}^p]\,\overrightarrow{x'}^{m_1}. \quad [\![F_1]\!][\overrightarrow{\pi_1}^{m_1}/\overrightarrow{x_P}^{m_1}], \\
& \quad \lambda\overrightarrow{x' : E'_2}^{m_2}. \quad \lambda z_P : c_1[\overrightarrow{P}^p]\,\overrightarrow{x}^{m_1} \sim^\chi_{I[\overrightarrow{P}^p]\,\overrightarrow{y}^n} c_2[\overrightarrow{P'}^p]\,\overrightarrow{x'}^{m_2}. \quad \mathsf{absurd}_s\,\vartheta_{1,2}\,\delta_{1,2}, \\
& \quad \vdots \\
& \quad \lambda\overrightarrow{x' : E'_k}^{m_k}. \quad \lambda z_P : c_1[\overrightarrow{P}^p]\,\overrightarrow{x}^{m_1} \sim^\chi_{I[\overrightarrow{P}^p]\,\overrightarrow{y}^n} c_k[\overrightarrow{P'}^p]\,\overrightarrow{x'}^{m_k}. \quad \mathsf{absurd}_s\,\vartheta_{1,k}\,\delta_{1,k}\big) \\
& \vdots \\
& \lambda\overrightarrow{x : E_k}^{m_k}.\,\mathtt{case}\,\big(M', \lambda\overrightarrow{y' : B'}^{m_1}, i' : I[\overrightarrow{P'}^p]\,\overrightarrow{y'}^n. \quad \forall i_P : c_k[\overrightarrow{P}^p]\,\overrightarrow{x}^{m_k} \sim^\chi_{I[\overrightarrow{P}^p]\,\overrightarrow{y}^n} i.\Theta \sim^\chi_T \Theta'), \\
& \quad \lambda\overrightarrow{x' : E'_1}^{m_1}. \qquad \lambda z_P : c_k[\overrightarrow{P}^p]\,\overrightarrow{x}^{m_k} \sim^\chi_{I[\overrightarrow{P}^p]\,\overrightarrow{y}^n} c_1[\overrightarrow{P'}^p]\,\overrightarrow{x'}^{m_1}. \qquad \mathsf{absurd}_s\,\vartheta_{k,1}\,\delta_{k,1}, \\
& \quad \vdots \\
& \quad \lambda\overrightarrow{x' : E'_{k-1}}^{m_{k-1}}. \quad \lambda z_P : c_k[\overrightarrow{P}^p]\,\overrightarrow{x}^{m_k} \sim^\chi_{I[\overrightarrow{P}^p]\,\overrightarrow{y}^n} c_{k-1}[\overrightarrow{P'}^p]\,\overrightarrow{x'}^{m_{k-1}}. \quad \mathsf{absurd}_s\,\vartheta_{k,k-1}\,\delta_{k,k-1}, \\
& \quad \lambda\overrightarrow{x' : E'_k}^{m_k}. \qquad \lambda z_P : c_k[\overrightarrow{P}^p]\,\overrightarrow{x}^{m_k} \sim^\chi_{I[\overrightarrow{P}^p]\,\overrightarrow{y}^n} c_k[\overrightarrow{P'}^p]\,\overrightarrow{x'}^{m_k}. \qquad [\![F_k]\!][\overrightarrow{\pi_k}^{m_k}/\overrightarrow{x_P}^{m_k}]\big)\Big)
\end{aligned}$$

Lorsque :

- $\Theta = \mathtt{case}_I(i, \overrightarrow{P}^p, \lambda\overrightarrow{y : B}^n, i : C.T, \overrightarrow{\lambda z : E^m.F}^k)$.
- le type principal de M est de la forme $I[\overrightarrow{P}^p]\,\overrightarrow{G}^n$ et que le type principal de T est $\mathtt{Type}_i$ avec $i > 0$.
- les instanciations de l'arité et des constructeurs par les paramètres sont de la forme :

$$\mathtt{Arity}_I(\overrightarrow{P}^p) = \forall\overrightarrow{y : B}^n.s \qquad \mathtt{Constr}^j_I(\overrightarrow{P}^p) = \forall\overrightarrow{z : E_j}^{m_j}.I[\overrightarrow{P}^p]\,D_j$$

- pour $1 \leq j \leq k$, les termes $\pi_{j,1}$, …, π_{j,m_j} sont construits de façon similaire à la fonction `inv` de la figure 5.1.
- le terme absurd_s est l'élimination de $\bot$ de type : $\forall X : s.\bot \to X$ avec s la sorte de T,
- pour $1 \leq j, l \leq k$ avec $j \neq l$:
  - on pose $\vartheta_{j,l} = [\![T]\!][c_j[\overrightarrow{P}^p]\,\overrightarrow{x}^{m_j}/i, c_l[\overrightarrow{P'}^p]\,\overrightarrow{x'}^{m_l}/i']$,
  - et les termes $\delta_{j,l}$ sont construits de façon similaire aux fonctions `abs12` et `abs21` de la figure 5.1.

FIGURE 5.2 – Traduction des éliminations fortes des petits inductifs.

Ainsi, cette transformation de paramétricité "patchée" devrait être, en toute rigueur, indicée par le contexte dans lequel les termes sont définis. C'est donc avec un certain abus des notations que nous admettrons le théorème suivant. C'est ce théorème qui sera utilisé pour développer les exemples de la prochaine section.

5.4.5 Théorème (Abstraction dans `Prop` avec élimination forte des petits inductifs)
Soient $\mathcal{P} \in \{\mathrm{CIC}, \mathrm{CIC}^-, \mathrm{CIC}^+\}$ *et soit le* $\chi = \chi_{\mathcal{P}}$ *le morphisme défini précédemment.*
Si $\Gamma \vdash_{\mathcal{P}_{\mathrm{small}}} M : T$, *alors*

1. $[\![\Gamma]\!]_\chi \vdash_{\mathcal{P}_{\mathrm{small}}} M : T$ *et* $[\![\Gamma]\!]_\chi \vdash_{\mathcal{P}_{\mathrm{small}}} M' : T'$,
2. *Si* $\Gamma \vdash T : s$, *alors* $[\![\Gamma]\!]_\chi \vdash_{\mathcal{P}_{\mathrm{small}}} [\![M]\!] : M \sim^\chi_T M'$,
3. *Si* $T \equiv s$, *alors* $[\![\Gamma]\!]_\chi, x : M, x' : M' \vdash_{\mathcal{P}_{\mathrm{small}}} x \sim^\chi_M x' : \lceil s \rceil$.

Que ce soit sur papier ou sur machine, formaliser complètement la preuve d'un tel théorème serait une tâche de très longue haleine. Ce qui semble néanmoins beaucoup plus raisonnable serait d'implémenter une nouvelle tactique de Coq qui calcule les termes générés par la transformation $[\![\cdot]\!]_\chi$. Il ne serait alors pas nécessaire de certifier l'implémentation de cette transformation puisque la typabilité des termes générés serait de toute façon vérifiée en aval par le système. Avec Chantal Keller, nous avons commencé à ébaucher une telle tactique [1] (néanmoins, à l'heure de la rédaction, cette ébauche n'est pas capable de traiter les inductifs).

5.5 Applications dans l'assistant de preuve Coq

Dans cette section, nous donnons quelques exemples de "théorèmes gratuits" pour l'assistant de preuve Coq. Il ne serait pas difficile de formaliser dans ce cadre tous les exemples que Philip Wadler donne dans [Wad89]. Nous préférerons plutôt donner ici des exemples originaux d'applications.

Dans le premier de ces exemples (sous-section 5.5.1), nous montrerons que les habitants clos du type des entiers de Church sont extensionnellement égaux à un opérateur d'itération de fonctions. Le second (sous-section 5.5.2) montrera la naturalité de tout terme ayant le même type que l'opérateur μ associée à une monade d'arbre. Dans le troisième exemple (sous-section 5.5.3), nous verrons que l'on peut étendre la paramétricité en présence de l'axiome d'équivalence des preuves et nous verrons également comment utiliser le théorème d'abstraction pour déduire des propriétés d'indépendance ; nous montrerons pour cela que l'axiome du tiers-exclu n'est pas dérivable. Enfin, nous donnerons un exemple d'application de la théorie de la paramétricité utile pour la formalisation des mathématiques (sous-section 5.5.4).

Dans cette section, nous donnerons des exemples formalisés dans le système CIC^-. Néanmoins, en indiçant convenablement la sorte `Set`, on pourra facilement porter ces exemples dans CIC^+ car aucun d'entre-eux n'utilise l'imprédicativité de `Set`. De même, mis à part l'exemple de la sous-section 5.5.1, les autres exemples pourront être formalisés dans CIC en remplaçant `Set` par $\mathtt{Type}_0$. Ainsi, les séquents $\vdash$ (resp. $\vdash_{\mathrm{small}}$) désigneront $\vdash_{\mathcal{P}}$ (resp. $\vdash_{\mathcal{P}_{\mathrm{small}}}$) pour $\mathcal{P} = \mathrm{CIC}^-$, $\mathcal{P} = \mathrm{CIC}^+$ ou $\mathcal{P} = \mathrm{CIC}$ (sauf dans la sous-section 5.5.1).

1. http://www.lix.polytechnique.fr/~keller/Recherche/coqparam.html

5.5.1 Le type des entiers de Church

Dans cette sous-section, on désignera par church le type $\forall\alpha : \texttt{Set}.(\alpha \to \alpha) \to \alpha \to \alpha$ et par nat l'inductif usuel (dans Set) pour représenter les entiers. Soit alors la fonction iter qui permet de composer une fonction n fois avec elle-même :

```
Fixpoint iter (n : nat) : church :=
  fun (α : Set) (f : α→α) (z : α)  ⇒
    match n with
      | O  ⇒  z
      | S p  ⇒  f (iter p α f z)
    end.
```

On peut prouver que tout objet de type church en relation paramétrique avec lui-même est extensionnellement égal à $\mathsf{iter}\, k$ pour un certain k. Autrement dit les "habitants paramétriques" de church sont tous des opérateurs d'itération. Ce que l'on peut exprimer par la formule suivante (on utilise les encodages standards de Coq pour le connecteur existentiel et l'égalité) :

5.5.1 Lemme

$$\vdash \forall f : \mathsf{church}. f \sim^{\chi}_{\mathsf{church}} f \to \exists k : nat. \forall \alpha : \texttt{Set}, g : \alpha \to \alpha, z : \alpha. \mathsf{iter}\, k\, \alpha\, g\, z =_{\alpha} f\, \alpha\, g\, z$$

Démonstration On décrit la dérivation dans un style informel.
Soit f de type church vérifiant $f \sim^{\chi}_{\mathsf{church}} f$ (H).
On pose $k = (f\, \mathsf{nat}\, \mathsf{S}\, \mathsf{O})$ (où S et O sont les constructeurs de nat).
Soit α de type Set, g de type $\alpha \to \alpha$ et z de type α.
On doit montrer $\mathsf{iter}\, (f\, \mathsf{nat}\, \mathsf{S}\, \mathsf{O})\, \alpha\, g\, z =_{\alpha} f\, \alpha\, g\, z$.
Nous allons pour cela utiliser l'hypothèse (H). La proposition $f \sim^{\chi}_{\mathsf{church}} f$ se déplie en :

$$\begin{array}{l}\forall \alpha\, \alpha' : \texttt{Set}, \alpha_P : \alpha \to \alpha' \to \texttt{Prop}, \forall g : \alpha \to \alpha, g' : \alpha' \to \alpha'. \\ \quad (\forall x : \alpha, x' : \alpha'. \alpha_P\, x\, x' \to \alpha_P\, (g\, x)\, (g'\, x')) \to \\ \quad \forall z : \alpha, z' : \alpha'. \alpha_P\, z\, z' \to \alpha_P\, (f\, \alpha\, g\, z)\, (f\, \alpha'\, g'\, z')\end{array}$$

Ainsi en instanciant cette formule avec $\alpha = \alpha$, $\alpha' = \mathsf{nat}$, $\alpha_P = \lambda y : \alpha, n : \mathsf{nat}. \mathsf{iter}\, n\, \alpha\, g\, z = y$, $g = g$ et $g' = \mathsf{S}$, on ramène la preuve de $\mathsf{iter}\, (f\, \mathsf{nat}\, \mathsf{S}\, \mathsf{O})\, \alpha\, g\, z =_{\alpha} f\, \alpha\, g\, z$ aux preuves de :

- $\forall y : \alpha, n : \mathsf{nat}. \mathsf{iter}\, n \alpha g z =_{\alpha} y \to \mathsf{iter}\, (\mathsf{S}\, n)\, \alpha\, g\, z =_{\alpha} g\, y$: or, on a

$$\mathsf{iter}\, (\mathsf{S}\, n)\, \alpha\, g\, z \rhd g\, (\mathsf{iter}\, n\, \alpha\, g\, z)$$

  et donc on peut utiliser la prémisse puis la réflexivité de l'égalité pour conclure ce cas.
- $\mathsf{iter}\, \mathsf{O}\, \alpha\, g\, z =_{\alpha} z$: trivial par réflexivité car $\mathsf{iter}\, \mathsf{O}\, \alpha\, g\, z \rhd z$. ☺

Le théorème d'abstraction nous permet alors de conclure que les termes clos de type church sont prouvablement extensionnellement égaux à $\mathsf{iter}\, k$ pour un certain k.

5.5.2 Lemme
Si $\vdash_{\text{small}} t : \mathsf{church}$, *alors* $\vdash_{\mathcal{P}} \exists k : nat. \forall \alpha : \texttt{Set}, g : \alpha \to \alpha, z : \alpha. \mathsf{iter}\, k\, \alpha\, g\, z =_{\alpha} t\, \alpha\, g\, z$

5.5.2 La monade d'arbres

On peut représenter les arbres binaires portant une information de type α sur leurs feuilles par la définition inductive suivante :

```
Inductive tree (α : Set) : Set :=
  | leaf : α→ tree α
  | node : tree α→ tree α→ tree α.
```

Il est alors aisé d'implémenter la fonction map qui applique une fonction à chacune des feuilles de l'arbre :

```
Fixpoint map (α β : Set) (f : α→ β) (t : tree α) : tree β :=
  match t with
  |  leaf x  ⇒  leaf β (f x)
  |  node l r  ⇒  node β (map α β f l) (map α β f r)
  end.
```

La relation générée par la transformation appliquée à l'inductif tree est donnée par la définition suivante :

```
Inductive tree_P (α α' : Set) (αP : α→ α'→ Prop) : Set :=
  | leaf : forall x x', α_P x x'→ tree α α' αP (leaf x) (leaf x')
  | node : forall l l', tree α α' αP l l'→
           forall r r', tree α α' αP r r'→
               tree α α' αP (node l r) (node l' r').
```

Deux feuilles sont en relation pour tree_P si et seulement si leurs informations sont en relation pour le paramètre α_P et deux nœuds sont en relations si et seulement si les deux branches de gauche et les deux branches de droite sont en relation.

On montre alors facilement :

5.5.3 Lemme

$$\vdash \forall \alpha\, \beta : \mathsf{Set}, f : \alpha \to \beta, t_1 : \mathsf{tree}[\alpha], t_2 : \mathsf{tree}[\beta].$$
$$(\mathsf{tree}_P[\alpha, \beta, \lambda x : \alpha, y : \beta. f\, x =_\beta y]\, t_1\, t_2) \leftrightarrow (t_2 =_{\mathsf{tree}[\beta]} \mathsf{map}\, \alpha\, \beta\, f\, t_1)$$

Démonstration Le sens direct utilise une induction sur la relation tree_P et le sens réciproque une induction sur la structure de l'arbre t_1. Nous donnons à titre indicatif, figure 5.3, une formalisation de cette démonstration en Coq. ☺

On en déduit alors le lemme suivant :

5.5.4 Lemme
Si $\vdash_{\text{small}} M : \forall \alpha : \mathsf{Set}.\mathsf{tree}[\mathsf{tree}[\alpha]] \to \mathsf{tree}[\alpha]$, *alors*

$$\vdash \forall \alpha\, \beta : \mathsf{Set}, f : \alpha \to \beta, t : \mathsf{tree}[\mathsf{tree}[\alpha]].$$
$$M\, \beta\, (\mathsf{map}\, \mathsf{tree}[\alpha]\, \mathsf{tree}[\beta]\, (\mathsf{map}\, \alpha\, \beta\, f)\, t) =_{\mathsf{tree}[\beta]} (\mathsf{map}\, \alpha\, \beta\, f\, (M\, \alpha\, t))$$

```
(* Some lemmas *)
Lemma tree_P_map1 :
  forall (α α' : Set) (f : α→α'),
    forall (a : tree α) (a' : tree α'),
      forall R,
        tree_P α α' R a a'→
        R = (fun x x' ⇒ f x = x')→
        map α α' f a = a'.
Proof.
  do 7 intro.
  induction H; intro; subst.
  simpl; rewrite H; reflexivity.
  simpl; rewrite IHtree_P1;[rewrite IHtree_P2|];
     reflexivity.
Qed.

Lemma tree_P_map2 :
  forall (α : Set) (a : tree α) (α' : Set)
    (f : α→α') (a' : tree α'),
    map α α' f a = a'→
    tree_P α α' (fun x x' ⇒ f x = x') a a'.
Proof.
  induction a; intros;
  subst; simpl; constructor; eauto.
Qed.
```

FIGURE 5.3 – Formalisation en COQ du lemme 5.5.3.

Démonstration Pour cela il suffit de montrer $(*)$:

$$\vdash \forall \mu : \forall \alpha : \mathsf{Set}\,.\mathsf{tree}[\mathsf{tree}[\alpha]] \to \mathsf{tree}[\alpha].\mu \sim^{\chi}_{\forall \alpha:\mathsf{Set}\,.\mathsf{tree}[\mathsf{tree}[\alpha]]\to\mathsf{tree}[\alpha]} \mu \to$$
$$\forall \alpha\, \beta : \mathsf{Set}, f : \alpha \to \beta, t : \mathsf{tree}[\mathsf{tree}[\alpha]].$$
$$\mu\, \mathsf{tree}[\beta]\, (\mathsf{map}\, \mathsf{tree}[\alpha]\, \mathsf{tree}[\beta]\, (\mathsf{map}\, \alpha\, \beta\, f)\, t) =_{\mathsf{tree}[\beta]} (\mathsf{map}\, \alpha\, \beta\, f\, (\mu\, \alpha\, t))$$

Puis d'appliquer d'utiliser le théorème d'abstraction pour déduire de

$$\vdash M : \forall \alpha : \mathsf{Set}\,.\mathsf{tree}[\mathsf{tree}[\alpha]] \to \mathsf{tree}[\alpha]$$

que

$$\vdash M \sim^{\chi}_{\forall \alpha:\mathsf{Set}\,.\mathsf{tree}[\mathsf{tree}[\alpha]]\to\mathsf{tree}[\alpha]} M$$

La preuve de $(*)$ se formalise relativement facilement en Coq, voir figure 5.4. ☺

Ainsi, il peut être utilisée pour montrer que tous les termes de type $\forall \alpha : \mathsf{Set}\,.\mathsf{tree}[\mathsf{tree}[\alpha]] \to \mathsf{tree}[\alpha]$ sont vérifient la propriété de naturalité. On dispose donc d'un moyen automatique pour prouver la naturalité de l'opérateur flatten de multiplication de la monade des arbres.

```
Fixpoint flatten (α : Set) (t : tree (tree α)) : tree α :=
 match t with
   | leaf t  ⇒  t
   | node l r  ⇒  node α (flatten α l) (flatten α r)
 end.
```

mais également de toutes les autres fonctions qui ont ce type ; et elles sont nombreuses, comme par exemple :

```
Fixpoint leftmost (α : Set) (t : tree (tree α)) : tree α :=
 match t with
   | leaf t  ⇒  t
   | node l r  ⇒  (leftmost α l)
 end.
Fixpoint flatten_swap (α : Set) (t : tree (tree α)) : tree α :=
 match t with
   | leaf t  ⇒  t
   | node l r  ⇒  node α (flatten_swap α r) (flatten_swap α l)
 end.
```

5.5.3 Axiomes classiques

Une des caractéristiques intéressantes de Coq est la possibilité d'ajouter des axiomes au système. Mais lorsque la transformation de paramétricité rencontre un de ces axiomes, elle demande une preuve que l'axiome est en relation avec lui-même. Soit P un axiome tel que $\vdash P : s$ avec $s = \texttt{Prop}$ ou $\texttt{Set}$. Alors trois cas sont possibles :

```
(* The translation : *)
Definition mu_P (α α′:Set) (α_P:α→α′→Prop)
  (μ : tree (tree α)→tree α)
  (μ′ : tree (tree α′)→tree α′) :=
  forall (a:tree (tree α))(a':tree (tree α′)),
    tree_P (tree α) (tree α′)
    (tree_P α α′ α_P) a a'→
    tree_P α α′ α_P (μ a) (μ′ a').


(* The proof that, if it satisfies the relation induced by
   its type, the function commutes with map *)
Lemma mu_map :
  forall
    (μ : forall α:Set, tree (tree α)→tree α),
    (forall α α′ α_P,
      mu_P α α′ α_P (μ α) (μ α′))→
    forall  (α α′ : Set) (f : α→α′)
      (z : tree (tree α)),
      μ α′ (map (tree α) (tree α′) (map α α′ f) z)
      = map α α′ f (μ α z).
Proof.
  intros.
  symmetry.
  eapply tree_P_map1; [|reflexivity].
  apply H.
  induction z; simpl; constructor; auto.
  apply tree_P_map2.
  reflexivity.
Qed.
```

FIGURE 5.4 – Formalisation en COQ du lemme 5.5.4.

- Soit P est ce qu'on appelle *prouvablement paramétrique* : L'utilisateur peut fournir une preuve de $\forall h : P.[\![P]\!]\, h\, h$ et cette preuve peut alors être utilisée par le théorème d'abstraction pour traduire les termes utilisant cet axiome.
- Soit P est *prouvablement non-paramétrique* : il existe une preuve de $\forall(h\, h' : P).\neg([\![P]\!]\, h\, h')$. Ceci signifie qu'il n'est pas possible de prouver la paramétricité de l'axiome sans compromettre la cohérence du système. Dans ce cas, il n'y a aucun moyen d'invoquer le théorème d'abstraction pour des termes qui utilisent cet axiome.
- Soit P n'est ni prouvablement paramétrique, ni prouvablement non paramétrique (ou l'utilisateur ne le sait pas). Dans ce cas, la paramétricité de l'axiome peut être ajoutée par l'utilisateur comme un nouvel axiome (mais cet ajout risque de rendre le système incohérent).

On remarque que si $\neg P$ est prouvable alors P est à la fois prouvablement paramétrique et prouvablement non-paramétrique. De même si P est prouvable, alors, d'après le théorème d'abstraction, il est prouvablement paramétrique. Il est également facile de déduire du théorème d'abstraction que si $P \to Q$ est prouvable, alors :

- P prouvablement paramétrique implique Q prouvablement paramétrique,
- Q prouvablement non-paramétrique implique P prouvablement non-paramétrique.

Ainsi si P est équivalent à Q, alors P est prouvablement paramétrique (resp. non-paramétrique) si et seulement si Q est prouvablement paramétrique (resp. non-paramétrique).

L'axiome d'équivalence des preuves. L'axiome d'équivalence des preuves affirme que quelle que soit la proposition, toutes les preuves de cette proposition sont égales. On peut exprimer cet axiome, noté PI, par la formule suivante :

$$\mathsf{PI} = \forall(X : \mathtt{Prop})(p\, q : X), p =_X q$$

où $p =_X q$ est un notation pour $\mathsf{eq\_prop}[X, p]\, q$ où $\mathsf{eq\_prop}$ est l'inductif encodant l'égalité de Leibniz pour la sorte `Prop` :

```
Inductive eq_prop (X : Prop) (h : X) : X→ Prop :=
  | refl_prop : eq_prop X h h.
```

Et l'énonce PI s'écrit dans la syntaxe de Coq de la façon suivante :

```
Definition PI :=
  forall (X : Prop)(x y : X), eq_prop X x y.
```

On peut traduire l'inductif $\mathsf{eq\_prop}$ par :

```
Inductive eq_prop_P (X X': Prop)(X_P : X→ X'→ Prop)
                    (h : X) (h' : X') (h_P : X_P h h') :
    forall (k : X) (k' : X') (k_P : X_P k k'),
      eq_prop X h k→  eq_prop X' h' k'→ Prop :=
| refl_prop_P : eq_prop_P X X' X_P h h' h_P h h' h_P (refl_prop X
   h) (refl_prop X' h').
```

Et la formule $pi \sim^{\chi}_{\mathsf{PI}} pi$ se déplie de la façon suivante :

```
Definition PI_P (pi pi' : PI) :=
  forall (X X' : Prop) (X_P : X→ X'→ Prop)
         (h : X) (h' : X') (h_P : X_P h h')
         (k : X) (k' : X') (k_P : X_P k k'),
    eq_prop_P X X' X_P h h' h_P k k' k_P (pi X h k) (pi' X' h' k').
```

Or en utilisant pi on peut montrer que :

$$\begin{array}{l}\forall pi : \mathsf{PI}.pi \sim^{\chi}_{\mathsf{PI}} pi \leftrightarrow \\ \quad \forall X\, X' : \mathtt{Prop}, X_P : X \to X' \to \mathtt{Prop}, \\ \quad\quad h : X, h' : X', h_P : X_P\, h\, h'. \\ \quad\quad\quad \mathsf{eq\_prop\_p}\, X\, X'\, X_P\, h\, h'\, h_P\, h\, h'\, h_P\, (\mathsf{refl\_prop}\, X\, h)\, (\mathsf{refl\_prop}\, X'\, h')\end{array}$$

Pour cela on utilise pi cinq fois afin de prouver les égalités : $h = k$, $h' = k'$, $h_P = k_P$, $(\mathsf{refl\_prop}\, X\, h) = (pi\, X\, h\, k)$, $(\mathsf{refl\_prop}\, X'\, h') = (pi\, X'\, h'\, k')$. On en déduit que, dans un contexte pi : PI, le type du constructeur refl_prop_P implique $pi \sim^{\chi}_{\mathsf{PI}} pi$. On obtient donc :

5.5.5 Lemme
L'axiome PI *est prouvablement paramétrique :*

$$\vdash \forall pi : \mathsf{PI}.pi \sim^{\chi}_{\mathsf{PI}} pi$$

Indépendance du tiers-exclu. Du point de vue de l'utilisateur, apprendre qu'un axiome est prouvablement non-paramétrique est une mauvaise nouvelle ; mais cela offre aux méta-théoriciens un nouvel outil pour prouver des résultats d'indépendance. En effet, d'après la réciproque du théorème d'abstraction, on a le lemme suivant :

5.5.6 Lemme
Si P est prouvablement non-paramétrique, alors il n'existe pas de terme M tel que $\vdash_{\text{small}} M : P$. C'est-à-dire, P n'est pas prouvable par une preuve qui n'utilise pas d'élimination forte sur les petits inductifs.

Par exemple, la loi de Peirce

$$\mathsf{Peirce} = \forall (X\, Y : \mathtt{Prop}).((X \to Y) \to X) \to X$$

connue pour être équivalente à l'axiome du tiers-exclu est prouvablement non paramétrique. En effet, si H est a pour type $h \sim^{\chi}_{\mathsf{Peirce}} h$, alors le terme

$$\begin{array}{ll}(H & \top\, \top\, (\lambda\_ : \top.\top) \\ & \top\, \bot\, (\lambda\_ : \top, \_ : \bot.\bot))\end{array}$$

a pour type, après réduction, le terme suivant :

$$((\top \to \top) \to \top) \to ((\top \to \bot) \to \top) \to ((\top \to \top) \to (\top \to \bot) \to (\top \to \top \to \bot \to \top) \to \bot) \to \bot$$

Or ce terme est bien équivalent à $\bot$ (le trois prémisses sont prouvables). On en déduit que :

5.5.7 Lemme
Peirce *est prouvablement non-paramétrique :*

$$\forall h\, h' : \mathsf{Peirce}\, .h \sim^{\chi}_{\mathsf{Peirce}} h \to \bot$$

Et donc :

5.5.8 Lemme (Indépendance du tiers-exclu)
Il n'existe pas de terme M tel que :

$$\vdash_{\text{small}} M : \mathsf{Peirce}$$

5.5.4 Théorie des groupes finis

Dans cette sous-section, nous allons montrer qu'il est possible d'obtenir des "théorèmes gratuits" sur des types de données avec une structure mathématique, ce qui, à notre connaissance, n'avait jamais était entrepris. Nous prendrons l'exemple de groupes finis, mais il aurait bien sûr été possible d'adapter la méthode présentée ici à d'autres structures algébriques.

Dans le chapitre 3.4 de sa thèse [Gar11], François Garillot observe que les formalisations d'algèbre requièrent de nombreuses preuves similaires. Intuitivement, une définition polymorphe, concernant un groupe abstrait, ne peut procéder qu'en composant avec les éléments communs à tous les groupes.

Nous proposons d'étudier des "opérateurs de groupe finis" : étant donné un groupe arbitraire de référence $\mathcal{H}$, possiblement infini, on appellera *opérateur de groupe fini* de $\mathcal{H}$, toute fonction Z qui associe à un sous-groupe fini G de $\mathcal{H}$, un sous-groupe fini $Z(G)$ de $\mathcal{H}$. Les sous-groupes finis seront alors représentés comme des listes d'éléments de $\mathcal{H}$, stables par les opérations du groupe ; ainsi Z sera implémenté par une fonction informative (dont le type du type est `Set`) des listes dans les listes.

Nous verrons les propriétés que l'on peut déduire de Z en fonction des axiomes auxquels cette fonction a accès sur $\mathcal{H}$. Plus particulièrement, nous allons montrer que si Z est défini par un terme de CIC, dans un contexte où $\mathcal{H}$ est décrit par :

- un domaine α : `Set`,
- un élément neutre $e : \alpha$,
- une loi de composition interne $\mathsf{op} : \alpha \to \alpha \to \alpha$,
- une fonction d'inversion $\mathsf{inv} : \alpha \to \alpha$,
- les axiomes standards des groupes :
  - l'associativité de la loi op,
  - la neutralité de e pour op,
  - le fait que inv soit l'inverse de op

alors Z satisfait nécessairement les deux propriétés suivantes, pour tout sous-groupe fini G de $\mathcal{H}$:

(1) $Z(G)$ est inclus dans G (et donc $Z(G)$ est un sous-groupe de G),
(2) pour tout endomorphisme f de G dans G, $Z(f(G)) = f(Z(G))$ (où $f(H)$ désigne l'image d'un sous-groupe H par f).

Nous dirons d'un opérateur qu'il est un *opérateur pleinement paramétrique* lorsqu'il vérifie ces deux propriétés.

On peut donner des exemples d'opérateurs de groupe implémentables avec cette axiomatique :

- Le *sous-groupe trivial* :

$$T(G) = \{e\}$$

– L'identité est un opérateur de groupe.
– Le *sous-groupe des puissances* n-ièmes ($n \in \mathbb{N}$) généralise les deux cas précédents (pour respectivement $n = 0$ et $n = 1$) :

$$P_n(G) = \{g^n | g \in G\}$$

– Le *sous-groupe dérivé* :

$$D(G) = \{ghg^{-1}h^{-1} | g, h \in G\}$$

– Plus généralement, pour toute famille C finie de mots $w_1, \ldots, w_m$ générés par la grammaire :

$$u, v \quad = \quad x \quad | \quad e \quad | \quad uv \quad | \quad u^{-1}$$

où $x \in \{x_1, \cdots, x_n\}$, alors on définit le *sous-groupe verbal* [2] W_C par :

$$W_C(G) = \{w_1, \ldots, w_m | x_1, \ldots, x_n \in G\}$$

On peut également définir des opérateurs de groupe qui ne sont pas définis de manière unique. En effet, les sous-groupes finis sont représentés comme des listes d'éléments. Or, ces listes peuvent très bien contenir plusieurs fois le même élément. Ainsi, deux listes différentes peuvent très bien représenter le même sous-groupe, que ce soit car l'ordre des éléments n'est pas le même ou car les éléments sont répétés un nombre de fois différent.

Notons $G \sim G'$ la formule $G \subseteq G' \wedge G' \subseteq G$ (où $\subseteq$ désigne l'inclusion des listes). Tous les opérateurs de groupes que nous avons vus jusqu'à présent satisfont la propriété suivante :

$$\forall GG', G \sim G' \rightarrow Z(G) \sim Z(G')$$

On appellera *opérateur bien défini* tout opérateur de groupe qui vérifie cette propriété. De plus, on nommera *opérateur monotone* tout opérateur qui vérifie la propriété :

$$\forall GG', G \subseteq G' \rightarrow Z(G) \subseteq Z(G')$$

Il est clair que les opérateurs monotones sont bien définis.

Il est aisé de construire des opérateurs mal définis, comme par exemple l'opérateur $G \mapsto P_{|G|}(G)$ où $|G|$ désigne la longueur de la liste qui représente G (et où P_n est l'opérateur du sous-groupe des puissances n-ième vu précédemment).

Si Z est un opérateur pleinement paramétrique et monotone, on en déduit que $Z(G)$ est invariant par endomorphisme : on a $f(Z(G)) = Z(f(G))$ d'après (2). Or, on a $f(G) \subseteq G$ (f est un endomorphisme) et la monotonie permet de déduire que $Z(f(G)) \subseteq Z(G)$. Ainsi nous obtenons que $f(Z(G)) \subseteq Z(G)$. Cette propriété de $Z(G)$ est parfois appelée "être un *sous-groupe pleinement caractéristique*" ou "*sous-groupe pleinement invariant*" [3]. En particulier, l'invariance par endomorphisme implique l'invariance par automorphisme. Lorsqu'un sous-groupe $\mathcal{H}$ d'un groupe G est invariant par auto-morphisme, on dit alors qu'il est un *sous-groupe caractéristique* de G. Tous les sous-groupes caractéristiques sont des sous-groupes normaux. Un *sous-groupe normal* est un sous-groupe invariant par les automorphismes intérieurs (un *automorphisme intérieur* est un morphisme f_g tel que $f_g(x) = gxg^{-1}$). La propriété de

2. http://groupprops.subwiki.org/wiki/Verbal_subgroup
3. http://groupprops.subwiki.org/wiki/Fully_invariant_subgroup

normalité est l'une des propriétés des sous-goupes les plus utilisées par les théoriciens des groupes. Elle est essentielle car elle caractérise les noyaux des homomorphismes et elle permet de définir la notion de groupe quotient très utile pour construire de nouveaux groupes.

Ainsi, les exemples d'opérateurs donnés plus haut sont tous des opérateurs monotones, et ils peuvent être programmés avec l'axiomatique présentée plus haut. On en déduit "for free" que leurs images sont toujours des sous-groupes pleinement caractéristiques de leur entrée.

Si Z est un opérateur pleinement paramétrique et bien défini, alors $Z(G)$ est un sous-groupe caractéristique de G. En effet, $f(Z(G)) = Z(f(G))$ et comme f est surjectif, on a $f(G) \sim G$, puis comme Z est bien défini, on en déduit : $f(Z(G)) \sim Z(G)$. Nous n'avons pas réussi à construire, sous l'axiomatique présenté ci-dessus, d'opérateur bien défini qui ne soit pas monotone.

Néanmoins, si on ajoute l'axiome de la décidabilité de l'égalité sur les éléments de **H**, c'est-à-dire une fonction informative qui implémente le test d'égalité, alors on peut implémenter d'autres opérateurs de groupe :

– Le *centre d'un groupe* :
$$C(G) = \{x | \forall y \in G, xy = yx \ \}$$
– Le *sous-groupe de Frattini* :
$$F(G) = \bigcap_{\substack{H \subseteq G \\ H \text{ sous-groupe maximal}}} H$$

On peut montrer que ces deux opérateurs sont bien définis (aucun d'entre eux n'est monotone). Mais, nous verrons que l'ajout d'un tel axiome affaiblira ce que l'on déduira du théorème d'abstraction. Nous montrerons qu'un opérateur Z implémenté dans un contexte contenant l'axiomatique précédent auquel on aura ajouté la décidabilité de l'égalité, vérifiera les propriétés suivantes (obtenues en remplaçant "endomorphisme" par "automorphisme" dans la définition d'opérateur pleinement paramétrique) :

(1') : $Z(G)$ est inclus dans G (et donc $Z\,G$ est un sous-groupe de G),

(2') : Pour tout automorphisme f de G dans G, $Z(f(G)) = f(Z(G))$ (où $f(H)$ désigne l'image d'un sous-groupe H par f).

Nous dirons d'un opérateur qu'il est un *opérateur paramétrique* lorsqu'il vérifie ces deux propriétés.

Ainsi, on pourra conclure que l'image des opérateurs implémentés avec l'axiome de décidabilité de l'égalité est un sous-groupe caractéristique de son entrée.

Formalisation. Soit $\Gamma_{\mathcal{H}}$ le contexte suivant (décrit avec la syntaxe de Coq) :

```
Variable α : Set.
Variable e : α.
Variable op : α→α→α.
Variable inv : α→α.

Infix "·" := (op) (at lvel 35).

Variable assoc :
  forall x y z, x·(y·z)=(x·y)·z.
Variable neutral_left :
```

```
    forall x, e·x = x.
  Variable inverse :
    forall x, (inv x)·x = e.
```

La commande `Infix` permet de déclarer la notation $M \cdot N$ pour signifier $\mathsf{op}\, M\, N$.

Une fois cette structure fixée, on peut définir le type des sous-groupes finis sur cette structure.

```
Structure fingrp := Fingrp {
   elements :> list α;
   e_comp : e ∈ elements;
   op_comp : forall x y, x ∈ elements→
                         y ∈ elements→ x·y ∈ elements;
   inv_comp : forall x, x ∈ elements→ inv x ∈ elements
}.
```

Le mot clef `Structure` de Coq permet de déclarer un nouvel inductif avec un unique constructeur, et de nommer les projections qui lui sont associées (un inductif à un constructeur représente un n-uplet dépendant). La définition de cette inductif dépend de α, *op*, *inv*, *e* qui sont implicitement ajoutés par le système comme des arguments de l'inductif. De plus, la première projection est désignée par la syntaxe "`:>`" comme étant une "coercion". Cette projection sera implicitement intercalée par le système lorsque que l'on fournira un terme de type fingrp là où on l'on attendrait un terme de type `list` α. Ce qui nous permettra d'écrire `x ∈ G` à la place de `x ∈ (elements G)`.

Soit Z un terme qui encode un opérateur de groupe, qui associe à un groupe fini sur la structure $\mathcal{H}$ un groupe sur cette même structure ; soit :

$$\Gamma_{\mathcal{H}} \vdash_{\text{small}} Z : \mathsf{fingrp}[\alpha, e, op, inv] \to \mathsf{fingrp}[\alpha, e, op, inv]$$

À titre d'exemple, on peut implémenter par un tel terme Z, l'opérateur des sous-groupes verbaux. Ils s'implémentent en construisant la liste des éléments par itération, puis en prouvant que cette liste est stable par les opérations du groupe.

Le théorème d'abstraction, nous permet de déduire que :

$$[\![\Gamma_{\mathcal{H}}]\!] \vdash [\![Z]\!] : Z \sim^{\chi}_{\mathsf{fingrp}[\alpha,e,op,inv]\to\mathsf{fingrp}[\alpha,e,op,inv]} Z$$

Pour la suite, nous n'avons pas besoin d'utiliser cette preuve avec des relations hétérogènes, ainsi nous allons instancier α' par α, e' par e, etc ... De plus, en présence de l'équivalence des preuves, il nous sera possible d'éliminer la dépendance en $assoc_P$, $neutral\_left_P$ et $inverse_P$; et ceci grâce au lemme suivant :

5.5.9 Lemme
En présence de l'axiome d'équivalence des preuves, les égalités sont "prouvablement paramétriques pour les relations homogènes", c'est-à-dire :

$$pi : \mathsf{PI} \vdash \forall \alpha : \mathsf{Set}, R : \alpha \to \alpha \to Prop, x : \alpha, x_P : R\, x\, x, y : \alpha, y_P : R\, y\, y, h : x =_\alpha y.$$
$$\left(h \sim^{\chi}_{x=_\alpha y} h\right)[\alpha/\alpha', R/\alpha_P, x/x', y/y']$$

Démonstration La preuve est formalisée en Coq par le script figure 5.5. ☺

```
Inductive eq_P (α α':Set) (α_P : α→α'→Set)
                    (x : α) (x' :α') (x_P : α_P x x') :
          forall y y', α_P y y'→x = y→x' = y'→Prop :=
   | refl_P : eq_P α α' α_P x x' x_P x x' x_P (eq_refl x) (eq_refl
      x').

Axiom pi : forall (X : Prop) (p q : X), p = q.

Lemma eq_P_auto :
  forall (α : Set) (R: α→α→Prop) x x_P y y_P (h : x = y),
     eq_P α α R x x x_P y y y_P h h.
intros. subst.
rewrite (pi _ y_P x_P).
apply refl_P.
Qed.
```

FIGURE 5.5 – Formalisation en COQ du lemme 5.5.9.

Soit Θ la substitution définie par :

$$\begin{aligned}\Theta(M) = M[&\alpha/\alpha', R/\alpha_P, e/e', op/op', inv/inv',\\ &assoc/assoc', \pi_{assoc}/assoc_P,\\ &neutral\_left/neutral\_left', \pi_{neutral\_left}/neutral\_left_P,\\ &inverse/inverse', \pi_{inverse}/inverse_P]\end{aligned}$$

où π_h est construit facilement à l'aide du lemme précédent.

Alors, par substitution, on peut en déduire que :

$$pi : \mathsf{PI}, \Gamma_{\mathcal{H}}, R : \alpha \to \alpha \to \mathtt{Prop}, \Delta \vdash \Theta([\![Z]\!]) : \Theta(Z \sim^{\chi}_{\mathsf{fingrp}[\alpha,e,op,inv]\to\mathsf{fingrp}[\alpha,e,op,inv]} Z)$$

où Δ est le contexte :

$$\begin{aligned}\Delta = \quad & R_e : R\,e\,e,\\ & R_{op} : \forall x\,x' : \alpha.R\,x\,x' \to \forall y\,y' : \alpha.R\,y\,y' \to R\,(x \cdot y)\,(x' \cdot y'),\\ & R_{inv} : \forall x\,x' : \alpha.R\,x\,x' \to R\,(\mathsf{inv}\,x)\,(\mathsf{inv}\,x')\end{aligned}$$

Si on note $\mathsf{fingrp\_rel}$ le terme :

$$\begin{aligned}\mathsf{fingrp\_rel} \quad = \quad \lambda \quad & R : \alpha \to \alpha \to \mathtt{Prop},\\ & R_e : R\,e\,e,\\ & R_{op} : \forall x\,x' : \alpha.R\,x\,x' \to \forall y\,y' : \alpha.R\,y\,y' \to R\,(x \cdot y)\,(x' \cdot y'),\\ & R_{inv} : \forall x\,x' : \alpha.R\,x\,x' \to R\,(\mathsf{inv}\,x)\,(\mathsf{inv}\,x').\\ & \qquad \mathsf{fingrp}_P[\alpha, \alpha, R, e, e, R_e, op, op, R_{op}, inv, inv, R_{inv}]\end{aligned}$$

Alors on a :

$$\Theta(Z \sim^{\chi}_{\mathsf{fingrp}[\alpha,e,op,inv]\to\mathsf{fingrp}[\alpha,e,op,inv]} Z) \quad \equiv$$
$$(\forall G\, G' : \mathsf{fingrp}[\alpha, op, inv, e].\mathsf{fingrp\_rel}\, R\, R_e\, R_{op}\, R_{inv}\, G\, G' \to \mathsf{fingrp\_rel}\, R\, R_e\, R_{op}\, R_{inv}\, (Z\, G)\, (Z\, G'))$$

Ainsi on prouve :

5.5.10 Lemme
Si $\Gamma_{\mathcal{H}} \vdash_{\text{small}} Z : \mathsf{fingrp}[\alpha, e, op, inv] \to \mathsf{fingrp}[\alpha, e, op, inv]$, *alors*

$$\begin{array}{rlll} pi : \mathsf{PI}, \Gamma_{\mathcal{H}} \vdash & \forall R & : \alpha \to \alpha \to \mathtt{Prop}, \\ & R_e & : R\, e\, e, \\ & R_{op} & : \forall x\, x' : \alpha. R\, x\, x' \to \forall y\, y' : \alpha. R\, y\, y' \to R\, (x \cdot y)\, (x' \cdot y'), \\ & R_{inv} & : \forall x\, x' : \alpha. R\, x\, x' \to R\, (\mathsf{inv}\, x)\, (\mathsf{inv}\, x') \\ & G\, G' & : \mathsf{fingrp}[\alpha, op, inv, e]. \\ & & \quad \mathsf{fingrp\_rel}\, R R_e R_{op} R_{inv}\, G\, G' \to \mathsf{fingrp\_rel}\, R\, R_e\, R_{op}\, R_{inv}\, (Z\, G)\, (Z\, G') \end{array}$$

On peut alors utiliser ce résultat en instanciant R par le graphe d'un automorphisme :

$$R = \lambda x : \alpha, y : \alpha. f\, x =_{\alpha} y$$

Puis, en dépliant correctement les définitions, on peut prouver la propriété (2) de la définition d'opérateur pleinement paramétrique. Nous avons formalisé en Coq le fait que les termes qui satisfont la conclusion du lemme précédent, satisfont également la condition (2) :

```
Variable Z : fingrp→ fingrp.
Hypothesis Z_param : forall R R_e R_op R_inv G G',
  fingrp_rel R R_e R_op R_inv G G'→
      fingrp_rel R R_e R_op R_inv (Z G) (Z G').

Variable G : fingrp.

Variable f : α→ α.
(* f is morphism : *)
Variable f_op : forall x y, x ∈ G→ y ∈ G→ (f x)·(f y) = f (x · y).
Variable f_endo : forall x, x ∈ G→ f x ∈ G.

(* G' is the image of G by f *)
Definition G' := image G f f_op.

Theorem fully_parametric :
  elements (Z G') = map f (elements (Z G)).
 (* ... *)
Qed.
```

Soit f un automorphisme de G, ce théorème, formalisé en Coq, affirme que l'image G' du groupe G par f est représentée par la même liste d'éléments que la liste des éléments du groupe $Z(G)$ auquel on a appliqué la fonction f.

Pour prouver la condition (1) de la définition de pleinement paramétrique, nous avons utilisé la paramétricité unaire, puisque cette dernière est suffisante pour l'inclusion. Tout ce qui a été dit précédemment en utilisant la paramétricité binaire s'adapte sans aucun problème à la paramétricité unaire. Ainsi, nous avons formalisé en Coq, le théorème ci-dessous :

```
Hypothesis Z_param_UR : forall UR UR_e UR_op UR_inv G,
  fingrp_urel UR UR_e UR_op UR_inv G→
      fingrp_urel UR UR_e UR_op UR_inv (Z G).

Variable G : fingrp.

Theorem inclusion : forall x, x ∈ (Z G)→ x ∈ G.
  (* ... *)
Qed.
```

La démonstration de ce théorème procède en instanciant la relation unaire UR par $\lambda x : \alpha.x \in G$, la relation d'appartenance à G.

Nous avons donc montré le théorème suivant :

5.5.11 Théorème
Si $\Gamma_{\mathcal{H}} \vdash_{\text{small}} Z : \mathsf{fingrp}[\alpha, e, op, inv] \to \mathsf{fingrp}[\alpha, e, op, inv]$, *alors*

$$pi : \mathsf{PI}, \Gamma_{\mathcal{H}} \vdash \mathsf{fully\_parametric\_operator}\, Z$$

où $\mathsf{fully\_parametric\_operator}\, Z$ *est une formalisation de la notion d'opérateur pleinement paramétrique.*

Ajouter l'axiome de décidabilité de l'égalité dans $\Gamma_{\mathcal{H}}$. Dans Coq, il est possible de formaliser la décidabilité d'une égalité par l'inductif suivant :

```
Inductive SB (α : Set) (x y : α) : Set :=
    | left : x = y→ SB α x y
    | right : x <> y→ SB α x y.
```

C'est un inductif informatif (dans `Set`) qui a deux constructeurs, l'un demandant une preuve de l'égalité, et l'autre une preuve de la différence de ses paramètres. On énonce l'axiome de décidabilité de l'égalité de la façon suivante :

$$\mathsf{eq\_dec}[\alpha] = \forall x\, y : \alpha.\mathsf{SB}[\alpha]\, x\, y$$

Cet axiome peut être utilisé pour construire une implémentation du centre du groupe de G en utilisant une fonction de filtrage sur les éléments de G qui sélectionne les éléments $x \in G$ vérifiant le prédicat qui affirme que x commute avec tous les éléments de G. Le fait que l'égalité soit décidable et que les groupes soient finis permet d'implémenter une telle fonction. De même, on peut implémenter le sous-groupe de Frattini de G en énumérant tous les sous-groupes de G, puis en filtrant les sous-groupes maximaux et en calculant leur intersection (la décidabilité de l'égalité permet d'implémenter une fonction qui teste

si une liste est sous-groupe d'un groupe arbitraire ainsi qu'une fonction qui implémente l'intersection de deux sous-groupes).

Hélas, eq_dec n'est pas prouvablement paramétrique pour toutes les relations R. En, effet, on peut montrer que s'il existe trois éléments a,b et b' de type α tels que $R\,a\,b$, $R\,a\,b'$ et $b \neq b'$, alors on a $\neg\left(\left(\mathsf{eq\_dec}[\alpha]\,a\,a \sim_{\mathsf{SB}[\alpha,a,b]} \mathsf{eq\_dec}[\alpha]\,a\,b'\right)[\alpha'/\alpha, R/\alpha_P, a/a']\right)$. Ce que l'on peut formaliser par le script figure 5.6. Il est toutefois possible de prouver que si R est le graphe d'une fonction injective, alors eq_dec est prouvablement paramétrique pour R. Soit inj_fun le terme défini par le script suivant :

```
Definition inj_fun (α α' : Set) (R : α→ α'→ Prop) :=
  (forall x y y', R x y→R x y'→y = y') ∧
    (forall x x' y, R x y→R x' y→x = x').
```

Alors, $\mathsf{inj\_fun}\,\alpha\,\alpha'\,R$ signifie que R est une relation fonctionnelle et injective. On peut alors prouver que :

5.5.12 Lemme
En présence de l'équivalence des preuves, l'axiome eq_dec *est "prouvablement paramétrique" pour les relations homogènes, fonctionnelles et injectives :*

$$\begin{aligned} pi : \mathsf{PI} \vdash \forall \alpha : \mathtt{Set}, R : \alpha \to \alpha \to \mathtt{Prop}\,.\mathsf{inj\_fun}\,\alpha\,\alpha\,R \to \\ \forall ax : \mathsf{eq\_dec}[\alpha], x\,x' : \alpha, x_P : R\,x\,x', y\,y' : \alpha, y_P : R\,y\,y'. \\ \left(ax\,x\,y \sim^{\chi}_{\mathsf{SB}[\alpha,x,y]} ax\,x'\,y'\right)[\alpha/\alpha', R/\alpha_R] \end{aligned}$$

Démonstration La preuve est formalisée en Coq par le script figure 5.6. Elle procède en distinguant selon les valeurs possibles de $(ax\,x\,y)$ et $(ax\,x'\,y')$. Le cas $x = y \wedge x' = y'$ et le cas $x \neq y \wedge x' \neq y'$ sont triviaux en utilisant le constructeur de SB_P. Et les autres cas sont impossibles car on a $R\,x\,x'$ et $R\,y\,y'$ et cette relation est fonctionnelle et injective. ☺

En utilisant le lemme précédent, on peut alors montrer que :

5.5.13 Lemme
Si $\Gamma_{\mathcal{H}}, ax : \mathsf{eq\_dec}[\alpha] \vdash_{\text{small}} Z : \mathsf{fingrp}[\alpha, e, op, inv] \to \mathsf{fingrp}[\alpha, e, op, inv]$, *alors*

$$\begin{aligned} pi : \mathsf{PI}, \Gamma_{\mathcal{H}}, ax : \mathsf{eq\_dec}[\alpha] \vdash \quad & \forall R \quad : \alpha \to \alpha \to \mathtt{Prop}, \\ & R_e \quad : R\,e\,e, \\ & R_{op} \quad : \forall x\,x' : \alpha.R\,x\,x' \to \forall y\,y' : \alpha.R\,y\,y' \to R\,(x \cdot y)\,(x' \cdot y'), \\ & R_{inv} \quad : \forall x\,x' : \alpha.R\,x\,x' \to R\,(\mathsf{inv}\,x)\,(\mathsf{inv}\,x') \\ & G\,G' \quad : \mathsf{fingrp}[\alpha, op, inv, e]. \\ & \qquad \mathsf{fingrp\_rel}\,RR_eR_{op}R_{inv}\,G\,G' \to \mathsf{fingrp\_rel}\,R\,R_e\,R_{op}\,R_{inv}\,(Z\,G)\,(Z\,G') \end{aligned}$$

On peut donc formaliser le fait que les termes qui vérifient la conclusion du lemme précédent vérifient l'invariance par automorphisme (la condition (2') de la définition d'opérateur paramétrique). Nous n'axiomatisons que la condition d'injectivité de l'endomorphisme pour deux raisons : seule la condition d'injectivité est nécessaire pour la preuve ; et la surjectivité est une conséquence de l'injectivité pour les endomorphismes de groupes finis.

```
Inductive SB_P α α' α_P x x' x_P y y' y_P :
                SB α x y→ SB α' x' y'→ Prop :=
| left_P : forall h h',
   eq_P α α' α_P x x' x_P y y' y_P h h'→
   SB_P α α' α_P x x' x_P y y' y_P (left _ _ _ h) (left _ _ _ h')
| right_P : forall h h',
   (forall k k', eq_P α α' α_P x x' x_P y y' y_P k k'→ False)→
   SB_P α α' α_P x x' x_P y y' y_P (right _ _ _ h) (right _ _ _ h').

Axiom pi : forall (X : Prop) (p q : X), p = q.

Lemma SB_prov_not_param :
   forall (α : Set) (R : α→ α→ Prop)
      a b b' (x_P : R a b) (y_P : R a b'), b <> b'→
      forall (ax : forall (x y : α), SB _ x y),
         ∼(SB_P α α R a b x_P a b' y_P (ax a a) (ax b b')).
Proof.
  intros.
  destruct (eq_dec a a); destruct (eq_dec b b'); subst;
  intro Habs;inversion Habs; auto.
Qed.

Lemma SB_prov_inj_param :
   forall (α : Set) (R : α→ α→ Prop), inj_fun _ _ R→ forall
     (ax : forall (x y : α), SB _ x y),
      forall x x' x_P y y' y_P,
         SB_P α α R x x' x_P y y' y_P (ax x y) (ax x' y').
Proof.
  intros. destruct H.
  destruct (ax x y); destruct (ax x' y'); subst; try econstructor.
  erewrite (pi _ x_P); econstructor.
  absurd (x' = y'); eauto.
  absurd (x = y); eauto.
  auto.
Qed.
```

FIGURE 5.6 – Formalisation du lemme 5.5.12

```
Variable Z : fingrp→fingrp.
Variable Z_inj_param : forall R, inj_fun _ _ R→
   forall R_e R_op R_inv G G', fingrp_R R R_e R_op R_inv G G'→
   fingrp_R R R_e R_op R_inv (Z G) (Z G').

Variable G : fingrp.

Variable f : α→α.
Variable f_op : forall x y, x ∈ G→y ∈ G→(f x)·(f y) = f (x · y).
Variable f_endo : forall x, x ∈ G→f x ∈ G.
Variable f_inj : forall x y, x ∈ G→y ∈ G→f x = f y→x = y.

Definition G' := image G f f_op.

Theorem parametric : elements (Z (image G f f_op)) = map f (Z G).
(* ... *)
Qed.
```

Enfin, on prouve la condition (1') de façon similaire au cas sans l'axiome de décidabilité de l'égalité : il faut prouver que l'axiome de décidabilité de l'égalité est prouvablement paramétrique pour les relations unaires homogènes ; mais cela ne pose aucune difficulté.

Nous avons donc montré le théorème suivant :

5.5.14 Théorème

Si $\Gamma, ax : \mathsf{eq\_dec}[\alpha] \vdash_{\mathrm{small}} Z : \mathsf{fingrp}[\alpha, e, op, inv] \to \mathsf{fingrp}[\alpha, e, op, inv]$, *alors*

$$pi : \mathsf{PI}, \Gamma \vdash \mathsf{parametric\_operator}\ Z$$

où $\mathsf{parametric\_operator}\ Z$ *est une formalisation de la notion d'opérateur paramétrique.*

Conclusion de la sous-section. Il est possible de voir en germe dans la notion de sous-groupe verbal l'idée que les opérateurs de groupe définis sur des groupes abstraits sont invariants par auto-morphisme. Dans cette sous-section, nous avons montré qu'il est possible de donner une formalisation concrète de cette idée lorsque l'opérateur est défini dans le calcul des constructions inductives.

Ainsi, nous montrons que la théorie de la paramétricité, bien qu'elle soit originellement développée pour l'étude des langages de programmation, peut être appliquée à certaines définitions de nature mathématique.

En prouvant que tous les opérateurs, définis en utilisant uniquement les axiomes standards qui spécifient un groupe abstrait ainsi qu'un axiome de décidabilité de l'égalité, sont des opérateurs paramétriques, nous avons montré que la paramétricité pouvait être utilisée pour déduire des métapropriétés intéressantes. En plus d'offrir une meilleur compréhension de la nature de ces opérateurs, elle offre à l'utilisateur un moyen automatique pour déduire à partir de la définition des opérateurs une preuve qu'ils ne génèrent que des sous-groupes caractéristiques.

À titre d'exemple, François Garillot répertorie dans [Gar11] 16 opérateurs de groupe utilisés dans le développement de SSREFLECT[4], la bibliothèque utilisée pour la formalisation de théorème de Feit-

4. http://ssr2.msr-inria.inria.fr/doc/ssreflect-1.4

Thompson. Si la transformation de paramétricité était implémentée par une *tactique* Coq, alors elle pourrait fournir à l'utilisateur, pour chaque définition d'opérateur de groupe, une preuve générée à partir de la structure de la définition que l'opérateur ne produit que des sous-groupes caractéristiques.

Bien sûr, une telle tactique ne serait plus utile pour la formalisation du théorème de Feit-Thompson, puisque les preuves ont été écrites. Mais la théorie des groupes n'a été prise ici que comme exemple. Par ailleurs, il est raisonnable de penser que les résultats présentés ici peuvent s'adapter à toutes les structures algébriques finies axiomatisées par des égalités universellement quantifiées (monoïdes finis, anneaux finis, etc. . .).

Nous pensons donc qu'une tactique capable de calculer les preuves et les énoncés de paramétricité pourrait être utile aux utilisateurs de Coq.

Conclusion

La principale construction mise en place dans ce travail est un moyen systématique de spécifier une logique à partir d'un langage de programmation. Nous avons choisi le cadre des systèmes de types purs pour la réaliser. Les deux premiers chapitres introduisent de nombreux d'outils qui permettent de ramener l'étude des propriétés des systèmes à l'étude de propriétés décidables sur les paramètres qui les décrivent. Ils pourront de surcroît être réutilisés pour étudier d'autres systèmes que ceux que nous avons proposés ici.

La logique engendrée par les langages de programmation est construite de façon à contenir exactement "ce qu'il faut" pour exprimer les formules de réalisabilité. Ainsi, nous avons pu voir que l'expressivité du langage de programmation de départ conditionne l'expressivité de la logique engendrée. Dans ces conditions, on peut alors dire que la notion de calcul préexiste à celle de démonstration. La théorie de la réalisabilité, développée dans la logique engendrée, généralise la réalisabilité dite "modifiée" de Kreisel. Elle permet de considérer la réalisabilité pour système $\mathcal{F}$ et système $\mathcal{T}$ comme deux instances d'une même construction et est appropriée à l'étude de leurs extensions. Nous avons en particulier prouvé que les fonctions représentables dans les extensions de $\mathcal{F}$ et de $\mathcal{T}$ sont exactement les fonctions prouvablement totales dans la logique que ces extensions engendrent.

Nous nous sommes concentrés sur le lien syntaxique entre cette théorie de la réalisabilité et les relations logiques issues de la théorie de la paramétricité. Nous avons montré que les formules encodant les relations logiques pouvaient être définies en itérant la construction de réalisabilité. La logique engendrée fournit donc un moyen d'obtenir, pour tout langage de programmation décrit comme un système de types purs, un cadre formel pour exprimer et prouver les propriétés de paramétricité du système. Si notre méthode fournit de manière automatique une logique formelle, elle n'explique pas comment interpréter ces formules. Construire des modèles généraux des logiques engendrées pourrait faire l'objet d'un travail prochain, comme prolongement de la présente recherche. Néanmoins, nous avons pu voir que lorsque le langage de départ est suffisamment expressif, c'est le cas du calcul des constructions inductives, il est alors capable d'interpréter ses propres formules de paramétricité. Ainsi, nous avons montré dans quelle mesure on peut en déduire une théorie de la paramétricité pour le système d'aide à la preuve Coq. Elle peut être utilisée afin de formaliser les applications standards de la paramétricité et pour prouver des résultats d'indépendance ; en outre, nous avons constaté qu'elle pouvait aussi avoir des applications utiles à la formalisation des mathématiques.

Nous avons développé une théorie de la paramétricité pour trois variantes du calcul des constructions. Le plus souvent possible, nous avons représenté les relations de paramétricité en utilisant l'encodage standard des relations (à savoir comme des prédicats à valeur dans la sorte des propositions). La première variante correspond au système Coq actuel, la seconde, dite "avec `Set` imprédicatif" corres-

pond au système Coq dans ses versions antérieures à la version 8.0 et enfin la troisième a été introduite dans ce travail et constitue un compromis entre les deux premiers systèmes. L'avantage principal de ce nouveau système est de renouer avec la démarche originelle sous-jacente à la distinction `Prop`/`Set`, présente dans les versions antérieures de Coq, tout en préservant la compatibilité avec les axiomes classiques. De fait, dans ce système, le typage permet de distinguer les termes qui seront effacés de ceux qui seront conservés, par le mécanisme d'extraction. Ce système s'est avéré intéressant dans le cadre de notre recherche dans la mesure où il offre un système aussi souple pour la paramétricité que le système avec `Set` imprédicatif. En effet, les types qui induisent des relations à valeur dans la sorte des propositions sont restreints aux types monomorphes dans la première variante tandis qu'ils comprennent des types polymorphes dans les deux autres. En outre, dans la dernière version, l'élimination forte sur une classe d'inductifs doit être restreinte si l'on veut que les relations soient à valeur dans la sorte des propositions. Un prochain travail pourrait être celui de comprendre le pourquoi de cette restriction, qui est si difficile à contourner ; en effet, il semble que, en présence d'éliminations fortes sur tous les inductifs, les relations de paramétricité ne se modélisent pas par des relations ensemblistes usuelles.

Cette théorie de la paramétricité pour Coq est sans doute l'application la plus concrète de notre travail ; on l'a ici exprimée de façon théorique ; reste maintenant à implémenter la transformation de paramétricité au sein du système Coq. Cette perspective présente un double intérêt pour les utilisateurs : d'une part, bénéficier d'un outil pour calculer et vérifier les formules de paramétricité, d'autre part, développer leurs propres conséquences de la paramétricité, en s'appuyant sur les mécanismes automatiques de Coq.

Index des systèmes

Notation	Description	Page
λ	λ-calcul simplement typé	19
$\lambda\Pi$	λ-calcul simplement typé avec type dépendant	19
$\lambda\omega$	λ-calcul simplement typé avec ordre supérieur de type	19
$\lambda\Pi^2$	λ-calcul simplement typé avec type dépendant et imprédicativité	19
$\lambda\Pi_\omega$	λ-calcul simplement typé avec type dépendant et ordre supérieur de type	19
$\mathcal{F}$	Système $\mathcal{F}$ de Girard, λ-calcul polymorphe	19
$\mathcal{F}_n$	Système $\mathcal{F}_\omega$ de Girard obtenu en ajoutant l'ordre supérieur de type à $\mathcal{F}$	189
$\mathcal{F}_\omega$	Système $\mathcal{F}_\omega$ de Girard obtenu en ajoutant l'ordre supérieur de type à $\mathcal{F}$	19
$\lambda\star$	Théorie des types naïve	52
$\lambda\star_\alpha$	Théorie des types prédicative	93
$\mathcal{T}$	Système $\mathcal{T}$ de Gödel, λ-calcul simplement typé avec inductif pour les entiers, les booléens et le produit cartésien	127
$\mathcal{T}_\omega$	Système $\mathcal{T}$ de Gödel avec ordre supérieur de type	129
$\mathcal{T}^{++}$	Système $\mathcal{T}$ de Gödel auquel avec un type singleton, un type vide et un opérateur d'encapsulation	249
$\mathcal{T}_\Omega$	Système $\mathcal{T}$ avec un inductif pour représenter les ordinaux de Brouwer	257
ECC	Calcul des Constructions Étendu	95
CC	Calcul des Constructions avec univers	19
CIC	Calcul des Constructions Inductives (avec univers)	145
CIC^-	Calcul des Construcions Inductives avec `Set` imprédicatif	143
CIC^+	Calcul des Construcions Inductives avec `Set` prédicatif	147
$\lambda\star_{\texttt{Prop,Set}}$	Système à partir duquel CIC, CIC^- et CIC^+ sont engendrés	190

Index des propriétés

Index des notations

Index

Bibliographie

[ACC93] M. Abadi, L. Cardelli, and P.-L. Curien, "Formal parametric polymorphism," in *POPL*, 1993, pp. 157–170.

[Acz86] P. Aczel, "The type theoretic interpretation of constructive set theory : Inductive definitions," in *Logic, Methodology, and Philosophy of Science VII*, R. B. Marcus, G. J. Dorn, and G. J. W. Dorn, Eds. North–Holland, Amsterdam and New York, 1986, pp. 17–49.

[Bar84] H. P. Barendregt, *The Lambda Calculus – Its Syntax and Semantics*. North-Holland, 1984, vol. 103.

[Bar91] H. Barendregt, "Introduction to Generalized Type Systems," *J. Funct. Program.*, vol. 1, no. 2, pp. 125–154, 1991.

[Bar92] ——, "Lambda Calculi with Types," in *Handbook of Logic in Computer Science*. Oxford University Press, 1992, pp. 117–309.

[Bar04] B. Barras, "Self-validation of a proof assistant with inductives families," Ph.D. dissertation, Université Paris 7 - Denis Diderot, Nov. 2004.

[BB85] C. Böhm and A. Berarducci, "Automatic Synthesis of Typed Lambda-Programs on Term Algebras," *Theor. Comput. Sci.*, vol. 39, pp. 135–154, 1985.

[BDN09] A. Bove, P. Dybjer, and U. Norell, "A Brief Overview of Agda — A Functional Language with Dependent Types," in *Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics*, ser. TPHOLs '09. Berlin, Heidelberg : Springer-Verlag, 2009, pp. 73–78. [Online]. Available : http://dx.doi.org/10.1007/978-3-642-03359-9_6

[Ber89] S. Berardi, "Type Dependence and Constructive Mathematics," Ph.D. dissertation, University of Torino, 1989.

[BG05] B. Barras and B. Grégoire, "On the role of type decorations in the calculus of inductive constructions," in *CSL*, 2005, pp. 151–166.

[BJO99] F. Blanqui, J.-P. Jouannaud, and M. Okada, "The Calculus of algebraic Constructions," in *RTA*, 1999, pp. 301–316.

[BJO02] ——, "Inductive-data-type systems," *Theor. Comput. Sci.*, vol. 272, no. 1-2, pp. 41–68, 2002.

[BJP10] J.-P. Bernardy, P. Jansson, and R. Paterson, "Parametricity and dependent types," in *ICFP*, 2010, pp. 345–356.

[Bla04] F. Blanqui, "Inductive types in the Calculus of Algebraic Constructions," *Fundam. Inf.*, vol. 65, no. 1-2, pp. 61–86, Aug. 2004. [Online]. Available : http://dl.acm.org/citation.cfm?id=1227143.1227147

[BS00] G. Barthe and M. H. Sørensen, "Domain-free pure type systems," *J. Funct. Program.*, vol. 10, no. 5, pp. 417–452, 2000.

[CH88] T. Coquand and G. Huet, "The calculus of constructions," *Information and Computation*, vol. 76, no. 2–3, pp. 95 – 120, 1988. [Online]. Available : http://www.sciencedirect.com/science/article/pii/0890540188900053

[Chu36] A. Church, "An Unsolvable Problem of Elementary Number Theory," *American Journal of Mathematics*, vol. 58, no. 2, pp. 345–363, Apr. 1936. [Online]. Available : http://dx.doi.org/10.2307/2371045

[Chu40] ——, "A Formulation of the Simple Theory of Types," *The Journal of Symbolic Logic*, vol. 5, no. 2, pp. pp. 56–68, 1940. [Online]. Available : http://www.jstor.org/stable/2266170

[Coq86] T. Coquand, "An Analysis of Girard's Paradox," in *LICS*, 1986, pp. 227–236.

[Coq89] ——, "Metamathematical investigations of a calculus of constructions," INRIA, Tech. Rep. RR-1088, Sep. 1989. [Online]. Available : http://hal.inria.fr/inria-00075471

[Coq04] Coq development team, *The Coq proof assistant reference manual*, LogiCal Project, 2004, version 8.0. [Online]. Available : http://coq.inria.fr

[CP88] T. Coquand and C. Paulin, "Inductively defined types," in *Conference on Computer Logic*, 1988, pp. 50–66.

[Cur34] H. B. Curry, "Functionality in Combinatory Logic." in *Proceedings of the National Academy of Sciences of the United States of America*, vol. 20, no. 11, Nov. 1934, pp. 584–590. [Online]. Available : http://view.ncbi.nlm.nih.gov/pubmed/16577644

[FLO83] S. Fortune, D. Leivant, and M. O'Donnell, "The Expressiveness of Simple and Second-Order Type Structures," *J. ACM*, vol. 30, no. 1, pp. 151–185, 1983.

[FT63] W. Feit and J. G. Thompson, "Solvability of groups of odd order," *Pacific J. Math.*, vol. 13, pp. 775–1029, 1963.

[Gar11] F. Garillot, "Generic Proof Tools and Finite Group Theory," Ph.D. dissertation, École Polytechnique, 2011.

[Geu93] H. Geuvers, "Logics and Type Systems," Ph.D. dissertation, Nijmegen University, 1993.

[Geu01] ——, "Inconsistency of classical logic in type theory," 2001. [Online]. Available : www.cs.ru.nl/~herman/PUBS/newnote.ps.gz

[Gim94] E. Giménez, "Codifying Guarded Definitions with Recursive Schemes," in *TYPES*, 1994, pp. 39–59.

[Gir71] J.-Y. Girard, "Une extension de l'interpretation de Godel a l'analyse, et son application a l'elimination des coupures dans l'analyse et la theorie des types," in *Proceedings of the second Scandinavian logic symposium*, vol. 63. North-Holland, 1971, pp. 63–92.

[Gir72] ——, "Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur," Ph.D. dissertation, Université Paris 7, 1972.

[Gir90] ——, "The system F of variable types, 15 years later," in *Logical foundations of functional programming*, G. Huet, Ed. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1990, pp. 87–126. [Online]. Available : http://dl.acm.org/citation.cfm?id=106791.106797

[Glo09] S. Glondu, "Extraction certifiée dans coq-en-coq," in *JFLA*, 2009, pp. 383–410.

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor, *Proofs and Types (Cambridge Tracts in Theoretical Computer Science)*. Cambridge University Press, Apr. 1989. [Online]. Available : http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0521371813

[GN91] H. Geuvers and M.-J. Nederhof, "Modular proof of strong normalization for the calculus of constructions," *J. Funct. Program.*, vol. 1, no. 2, pp. 155–189, 1991.

[Goo44] R. L. Goodstein, "On the Restricted Ordinal Theorem," *The Journal of Symbolic Logic*, vol. 9, no. 2, pp. pp. 33–41, 1944. [Online]. Available : http://www.jstor.org/stable/2268019

[GV09] H. Geuvers and J. Verkoelen, "On Fixed point and Looping Combinators in Type Theory," 2009. [Online]. Available : http://www.cs.ru.nl/~herman/PUBS/TLCApaper.pdf

[Gö31] K. Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme," *Monatshefte für Mathematik und Physik*, vol. 38, no. 1, pp. 173–198, 1931.

[Har56] R. Harrop, "On disjunctions and existential statements in intuitionistic systems of logic," *Mathematische Annalen*, vol. 132, no. 4, pp. 347–361, 1956.

[How69] W. Howard, "To H.B. Curry : The formulae-as-types notion of construction," in *Essays on Combinatory Logic, Lambda Calculus, and Formalism*, J. Hindley and J. Seldin, Eds. Academic Press, 1969.

[HPJW+92] P. Hudak, S. Peyton Jones, P. Wadler, B. Boutel, J. Fairbairn, J. Fasel, M. M. Guzmán, K. Hammond, J. Hughes, T. Johnsson, D. Kieburtz, R. Nikhil, W. Partain, and J. Peterson, "Report on the programming language Haskell : a non-strict, purely functional language version 1.2," *SIGPLAN Not.*, vol. 27, no. 5, pp. 1–164, May 1992. [Online]. Available : http://doi.acm.org/10.1145/130697.130699

[Hur95] A. J. C. Hurkens, "A Simplification of Girard's Paradox," in *TLCA*, 1995, pp. 266–278.

[Jim99] B. Jiménez, "Condensing Lemmas for Pure Type Systems with Universes," in *Algebraic Methodology and Software Technology*, ser. Lecture Notes in Computer Science, A. Haeberer, Ed. Springer Berlin / Heidelberg, 1999, vol. 1548, pp. 422–437. [Online]. Available : http://dx.doi.org/10.1007/3-540-49253-4_30

[KL12] C. Keller and M. Lasson, "Parametricity in an Impredicative Sort," in *CSL*, 2012, pp. 381–395.

[Kle45] S. C. Kleene, "On the Interpretation of Intuitionistic Number Theory," *J. Symb. Log.*, vol. 10, no. 4, pp. 109–124, 1945.

[Kle71] ——, *Introduction to metamathematics*. Wolters-Noordhoff, 1971.

[KLN] F. Kamareddine, T. Laan, and R. Nederpelt, "De Bruijn's Automath and Pure Type Systems."

[KP82] L. Kirby and J. Paris, "Accessible Independence Results for Peano Arithmetic," *Bulletin of the London Mathematical Society*, vol. 14, no. 4, pp. 285–293, 1982. [Online]. Available : http://blms.oxfordjournals.org/content/14/4/285.short

[KP90] J.-L. Krivine and M. Parigot, "Programming with proofs," *J. Inf. Process. Cybern.*, vol. 26, no. 3, pp. 149–167, 1990.

[Kre59] G. Kreisel, "Interpretation of analysis by means of constructive functionals of finite types," in *Constructivity in mathematics*, A. Heyting, Ed., 1959, pp. 101–128.

[Kri93] J.-L. Krivine, *Lambda-calculus, types and models*, ser. Ellis Horwood series in computers and their applications. Masson, 1993.

[Kri09] ——, "Realizability in classical logic," *Panoramas et synthèses*, vol. 27, pp. 197–229, 2009. [Online]. Available : http://hal.archives-ouvertes.fr/hal-00154500

[Kri12] ——, "Realizability algebras II : new models of ZF + DC," *Logical Methods in Computer Science*, vol. 8, no. 1, 2012.

[LDF+] X. Leroy, D. Doligez, A. Frisch, J. Garrigue, D. Rémy, and J. Vouillon, *The OCaml system : Documentation and user's manual*, Institut National de Recherche en Informatique et en Automatique. [Online]. Available : http://caml.inria.fr

[Lei83] D. Leivant, "Reasoning about functional programs and complexity classes associated with type disciplines," in *FOCS*, 1983, pp. 460–469.

[Lei90] ——, "Contracting proofs to programs," in *Logic and Computer Science*, P. Odifreddi, Ed. Academic Press, London, 1990, pp. 279–327.

[Lei91] ——, "Finitely Stratified Polymorphism," *Inf. Comput.*, vol. 93, no. 1, pp. 93–113, 1991.

[Ler95] X. Leroy, "Applicative functors and fully transparent higher-order modules," in *Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, ser. POPL '95. New York, NY, USA : ACM, 1995, pp. 142–153. [Online]. Available : http://doi.acm.org/10.1145/199448.199476

[Ler06] ——, "Formal certification of a compiler back-end or : programming a compiler with a proof assistant," in *POPL*, 2006, pp. 42–54.

[Let02] P. Letouzey, "A New Extraction for Coq," in *TYPES*, 2002, pp. 200–219.

[Let04] ——, "Programmation fonctionnelle certifiée : L'extraction de programmes dans l'assistant Coq," Ph.D. dissertation, Université Paris-Sud, Jul. 2004.

[Let08] ——, "Extraction in Coq : An Overview," in *CiE*, 2008, pp. 359–369.

[Luo89] Z. Luo, "ECC, an Extended Calculus of Constructions," in *LICS*, 1989, pp. 386–395.

[Luo90] ——, "An Extended Calculus of Constructions," Ph.D. dissertation, University of Edinburgh, 1990.

[Lö71] P. M. Löf, "A theory of Type," 1971.

[Mai91] H. G. Mairson, "Outline of a proof theory of parametricity," in *FPCA*, 1991, pp. 313–327.

[Men09] E. Mendelson, *Introduction to Mathematical Logic*, 5th ed. Chapman & Hall/CRC, 2009.

[Mil78] R. Milner, "A Theory of Type Polymorphism in Programming," *J. Comput. Syst. Sci.*, vol. 17, no. 3, pp. 348–375, 1978.

[MLS84] P. Martin-Löf and G. Sambin, *Intuitionistic type theory*, ser. Studies in proof theory. Bibliopolis, 1984. [Online]. Available : http://books.google.fr/books?id=_D0ZAQAAIAAJ

[MW96] P.-A. Melliès and B. Werner, "A Generic Normalisation Proof for Pure Type Systems," in *TYPES*, 1996, pp. 254–276.

[MW02] A. Miquel and B. Werner, "The Not So Simple Proof-Irrelevant Model of CC," in *TYPES*, 2002, pp. 240–258.

[Nic70] Nicolaas Govert de Bruijn, "The mathematical language AUTOMATH, its usage, and some of its extensions," in *Symposium on Automatic Demonstration*, ser. Lecture Notes in Mathematics, M. Laudet, D. Lacombe, L. Nolin, and M. Schützenberger, Eds. Springer Berlin / Heidelberg, 1970, vol. 125, pp. 29–61, 10.1007/BFb0060623. [Online]. Available : http://dx.doi.org/10.1007/BFb0060623

[PA93] G. D. Plotkin and M. Abadi, "A Logic for Parametric Polymorphism," in *TLCA*, 1993, pp. 361–375.

[PE88] F. Pfenning and C. Elliott, "Higher-Order Abstract Syntax," in *PLDI*, 1988, pp. 199–208.

[Pie94] B. C. Pierce, "Bounded Quantification is Undecidable," *Information and Computation*, vol. 112, no. 1, pp. 131–165, Jul. 1994, also in C. A. Gunter and J. C. Mitchell, editors, *Theoretical Aspects of Object-Oriented Programming : Types, Semantics, and Language Design*, MIT Press, 1994. Summary in *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), Albuquerque, New Mexico.*

[PM89a] C. Paulin-Mohring, "Extracting F_ω's programs from proofs in the Calculus of Constructions," in *Sixteenth Annual ACM Symposium on Principles of Prog ramming Languages.* Austin : ACM, Jan. 1989.

[PM89b] ——, "Extraction de programmes dans le Calcul des Constructions," Thése d'université, Paris 7, Jan. 1989. [Online]. Available : http://www.lri.fr/~paulin/PUBLIS/these.ps.gz

[PM93] ——, "Inductive definitions in the system Coq rules and properties," in *Typed Lambda Calculi and Applications*, ser. Lecture Notes in Computer Science, M. Bezem and J. Groote, Eds. Springer Berlin / Heidelberg, 1993, vol. 664, pp. 328–345, 10.1007/BFb0037116. [Online]. Available : http://dx.doi.org/10.1007/BFb0037116

[PPM89] F. Pfenning and C. Paulin-Mohring, "Inductively Defined Types in the Calculus of Constructions," in *Mathematical Foundations of Programming Semantics*, 1989, pp. 209–228.

[PS99] F. Pfenning and C. Schurmann, "System Description : Twelf — A Meta-Logical Framework for Deductive Systems," in *Proceedings of the 16th International Conference on Automated Deduction (CADE-16).* Springer-Verlag LNAI, 1999, pp. 202–206.

[Rey74] J. C. Reynolds, "Towards a Theory of Type Structure," in *Symposium on Programming*, 1974, pp. 408–423.

[Rey83] ——, "Types, Abstraction and Parametric Polymorphism," in *IFIP Congress*, 1983, pp. 513–523.

[Ros] H. E. Rose, *Subrecursion : Functions and Hierarchies (Oxford Logic Guides)*. Clarendon Pr.

[Sch75] H. Schwichtenberg, "Definierbare Funktionen imλ-Kalkül mit Typen," *Archiv für mathematische Logik und Grundlagenforschung*, vol. 17, pp. 113–114, 1975. [Online]. Available : http://dx.doi.org/10.1007/BF02276799

[Sta73] J. Staples, "Combinator realizability of constructive finite type analysis," *Cambridge Summer School in Mathematical Logic*, pp. 253–273, 1973.

[Sta81] R. Statman, "Number theoretic functions computable by polymorphic programs," in *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*, ser. SFCS '81. Washington, DC, USA : IEEE Computer Society, 1981, pp. 279–282. [Online]. Available : http://dx.doi.org/10.1109/SFCS.1981.24

[Tak97] I. Takeuti, "An axiomatic system of parametricity," in *TLCA*, 1997, pp. 354–372.

[Tak98] ——, "An Axiomatic System of Parametricity," *Fundam. Inform.*, vol. 33, no. 4, pp. 397–432, 1998.

[Ter89] J. Terlouw, "Een nadere bewijstheoretische analyse van gstts," 1989.

[Tro98] A. Troelstra, *Handbook of proof theory*, S. Buss, Ed. Elsevier, 1998.

[Tur36] A. M. Turing, "On Computable Numbers, with an application to the Entscheidungsproblem," *Proc. London Math. Soc.*, vol. 2, no. 42, pp. 230–265, 1936.

[Van02] J. Van Oosten, "Realizability : a historical essay," *Mathematical Structures in Comp. Sci.*, vol. 12, no. 03, pp. 239–263, 2002.

[vBJ93] L. S. van Benthem Jutting, "Typing in Pure Type Systems," *Inf. Comput.*, vol. 105, no. 1, pp. 30–41, 1993.

[vD80] D. T. van Daalen, "The language Theory of AUTOMATH," Ph.D. dissertation, Technical University of Eindhoven, Eindhoven, Netherlands, 1980.

[vH02] J. van Heijenoort, *From Frege to Gödel : A Source Book in Mathematical Logic, 1879-1931 (Source Books in the History of the Sciences)*. Harvard University Press, January 2002. [Online]. Available : http://www.worldcat.org/isbn/0674324498

[Wad89] P. Wadler, "Theorems for free!" in *Proceedings of the fourth international conference on Functional programming languages and computer architecture*, ser. FPCA '89. New York, NY, USA : ACM, 1989, pp. 347–359. [Online]. Available : http://doi.acm.org/10.1145/99370.99404

[Wad07] ——, "The Girard-Reynolds isomorphism (second edition)," *Theor. Comput. Sci.*, vol. 375, no. 1-3, pp. 201–226, 2007.

[Wer94] B. Werner, "Une Théorie des Constructions Inductives," Ph.D. dissertation, Université Paris 7, 1994.

[Wer97] ——, "Sets in types, types in sets," in *Proceedings of TACS'97*. Springer-Verlag, 1997, pp. 530–546.

[Zwa99] J. Zwanenburg, "Pure Type Systems with Subtyping," in *Proceedings of the 4th International Conference on Typed Lambda Calculi and Applications*, ser. TLCA

'99. London, UK, UK : Springer-Verlag, 1999, pp. 381–396. [Online]. Available : http://dl.acm.org/citation.cfm?id=645894.671761