Realizability Games in Arithmetical Formulae.

Mauricio Guillermo,

REALIZABILITY GAMES IN ARITHMETICAL FORMULÆ

MAURICIO GUILLERMO

Jeux de réalisabilité en arithmétique classique

THÈSE

MAURICIO GUILLERMO
PHD STUDENT
UNIVERSITY PARIS VII
PPS TEAM
guillermo@pps.jussieu.fr

Contents

INTRODUCTION

0.1. **Contenu du travail.**

Section 1: On présente ici la majeure partie des définitions syntaxiques nécessaires à la compréhension de ce travail.

Dans la première sous-section sont présentés les langages de termes, formules et processus. La notion de formule est étendue pour s'adapter plus aisément à la sémantique. Ces formules étendues permettant d'exprimer des valeurs de vérité avec une grande flexibilité, sont indispensables pour définir les types de données dans la section 3.

Dans la sous-section 2 est présenté le système de typage. La présentation des règles de typage est la même que dans les références classiques sur la réalisabilité de Krivine ([6], [7]), mais nous ajoutons aussi des règles qui s'avérent bien pratiques pour pouvoir typer avec des formules étendues. Bien évidemment on arrive ainsi à typer avec des formules qui n'ont pas de signification logique. Cependant, dans la section 3 cette possibilité est particulièrement utile pour trouver des opérateurs de mise en mémoire.

Dans la sous-section 3, on définit les processus avec les règles de réduction. En d'autres termes, on définit une machine symbolique, qui est appelée la *Machine de Krivine*. Les processus sont des couples formés par un terme (nommé le *terme de tête* ou *terme actif*) et une liste finie de termes (nommée la *pile d'arguments* ou simplement la *pile*). La réduction est définie pour les processus (et non pas pour les termes). Cette réduction est déterministe puisque, à chaque étape, il y a au plus une seule règle de réduction applicable. En associant au couple (terme actif, pile) le terme consistant en l'application du terme actif à la pile d'arguments, on voit que la réduction de processus correspond à la réduction de tête faible.

La sous-section 4 présente une définition de substitution des constantes qui est fondamentale dans la suite. Cette substitution est appelée *statique* par opposition à la substitution *dynamique* qu'on définit dans la section 4.

Section 2: On définit ici différentes notions sémantiques autour des langages de formules.

Il y a une partie commune à toutes ces sémantiques qui est le fragment qu'interprétent les termes. Dans la première sous-section on commence en définissant la sémantique pour les termes, les modèles de Tarski pour le langage du deuxième ordre et on termine avec l'arithmétique de Peano du deuxième ordre. Dans cette sous-section, on insiste sur le fait que les modèles de Tarski qu'on utilise ne sont pas forcément pleins et on est donc en mesure d'utiliser le théorème de complétude.

Dans la deuxième sous-section on définit les modèles booléens, lesquels sont très proches des modèles de réalisabilité, nous définissons par la suite.

Dans la sous-section 3, on présente enfin les modèles de réalisabilité. Ces modèles sont définis à partir d'un modèle de Tarski (dit "modèle de départ") et un ensemble de processus clos par antiréduction qu'on dénote $\perp\!\!\!\perp$. On énonce et démontre aussi le lemme d'adéquation adapté au langage de formules étendues.

Dans la théorie des modèles, le “principe de correction” (“soundness principle”) établit que la déduction préserve la vérité. Autrement dit, c’est la correction qui détermine que démontrer sert à connaître la vérité (dans le sens de Tarski) dans les modèles. Le paradigme de la réalisabilité consiste à associer à chaque formule un ensemble de programmes comme valeur de vérité. En conséquence, l’adéquation en réalisabilité établit que démontrer sert à réaliser.

C’est dans la sous-section 4 qu’est énoncé et prouvé le lemme d’adéquation.

Puisque l’adéquation constitue le lien entre réalisabilité et déduction, l’introduction de nouvelles règles de typage demandera toujours de vérifier que la nouvelle règle est adéquate. Malgré cette contrainte, il est important de signaler que l’introduction de nouvelles règles ne peut jamais rendre inadéquates les règles préexistantes. Grâce à cette “modularité” de l’adéquation, le langage de termes peut évoluer, en ajoutant de nouvelles règles de typage tout en ayant un minimum de vérifications à faire.

Ainsi, le mécanisme pour passer de la logique intuitionniste à la logique classique utilisé en réalisabilité de Krivine consiste à ajouter une instruction cc réalisant la loi de Peirce (laquelle implique le tiers exclu). L’instruction introduite garde dans un pointeur – nommé “continuation” – la pile courante, pour la rétablir au cas où la continuation arriverait en position active. Le rapport entre cette instruction (très utile, surtout en programmation système) et la loi de Peirce, fut découvert par Griffin (c.f.: [3]). Le système de typage obtenu en ajoutant à la logique intuitionniste (qui correspond au lambda calcul pur), une règle qui déclare cc avec le type “loi de Peirce”, est-il adéquat? Pour répondre à cette question, il suffit de vérifier que la nouvelle règle est adéquate et cela correspond exactement à prouver que cc réalise son type. Voilà donc le paradigme pour ajouter des axiomes au système de types de la réalisabilité: il suffit de trouver une instruction et de prouver qu’elle réalise l’axiome.

La sous-section 5 explique dans quelles conditions il est possible de construire des modèles de Tarski à partir des modèles de réalisabilité. C’est ici que l’on peut voir la forte analogie entre le forcing et la réalisabilité. La technique du forcing, développée par Cohen, est utile pour prouver l’existence d’un modèle de Tarski d’une certaine théorie, pourvu qu’on admette l’existence d’un modèle d’une autre théorie “de départ”. Ce genre de preuves est utile pour répondre à la question de l’indépendance d’axiomes dans une certaine théorie (Cohen développa le forcing pour prouver l’indépendance de l’hypothèse du continu en théorie des ensembles. c.f.: [2]). L’idée est la suivante: soit $\mathscr{T}$ une théorie supposée consistante. On veut prouver qu’un certain axiome φ est indépendant de la théorie $\mathscr{T}$; autrement dit: qu’on ne peut pas déduire φ ni $\neg\varphi$ à partir de $\mathscr{T}$. On définit alors dans $\mathscr{T}$ un certain ensemble d’éléments – nommés “conditions” – et, à partir de chaque formule $\psi(\vec{y})$, on définit une nouvelle formule $\psi^*(x,\vec{y})$ (qui se lit “x force $\psi(\vec{v})$”). On peut prouver dans $\mathscr{T}$ que l’ensemble de formules qui sont forcées par une condition, est clos par déduction (adéquation du forcing) et cohérent (c'est à dire qu'on ne peut pas forcer $\perp$). Alors, l'existence d'un modèle de $\mathscr{T}$ implique aussi l'existence d'un modèle de $\mathscr{T}$ satisfaisant l'ensemble des formules forcées. En particulier, si on obtient une condition qui force φ, on obtient la consistance de $\mathscr{T}+\varphi$. La même méthode peut s'appliquer éventuellement à $\neg\varphi$ et montrer alors l'indépendance de φ.

En réalisabilité on a le même genre de construction: un modèle de départ, les termes qui jouent le rôle de l'ensemble de conditions ; une fois les termes codés dans le modèle de départ, la formule "le terme t réalise la formule φ" est une formule du langage de ce modèle. Il existe alors un modèle de l'arithmétique dans lequel tout ce qui est réalisé est vrai.

L'analogie entre le forcing et la réalisabilité va bien au delà d'une simple ressemblance: il existe une définition de "structure de réalisabilité" dont le forcing et la réalisabilité sont des cas particuliers.

La sixième sous-section contient de brefs commentaires au sujet de l'arithmétique dans les modèles de réalisabilité. En particulier, on explique que, pour pouvoir démontrer des théorèmes arithmétiques, on a besoin de restreindre les quantificateurs à la formule $\mathrm{int}(x)$ (c.f.:[6]). On trouvera un travail en profondeur sur les modèles de l'arithmétique en réalisabilité dans [10].

Section 3: Elle est consacrée à définir les types de données à partir des opérateurs de mise en mémoire. Selon la définition présentée, un type de données est une formule à une variable libre qui admet un opérateur de mise en mémoire. La formalisation de cette définition est une généralisation naturelle du cas des entiers, lesquels ont une représentation "canonique" par des entiers de la forme $(\mathrm{s})^n 0$. Intuitivement, l'opérateur de mise en mémoire des entiers permet d'étendre une fonction définie uniquement sur les formes canoniques, à tout le type de données. On généralise cette idée en mettant à la place de la formule des entiers, une formule quelconque à une variable libre. La définition est exprimée sémantiquement comme une condition de réalisabilité.

La première sous-section développe les définitions tandis que les sous-sections 2, 3, 4, 5 et 6 traitent respectivement les cas des entiers, des booléens, les produits de types de données, les listes chaînées et les arbres; le tout explicitant les opérateurs de mise en mémoire et les représentations canoniques. On a obtenu les opérateurs de mise en mémoire par preuve. La technique utilisée est une légère adaptation au présent contexte de celle présentée dans [4].

Finalement, dans la sous-section 7, on démontre qu'on ne peut pas utiliser des représentations canoniques normalisées pour les types de données récursifs. Ce théorème explique pourquoi on a choisi des représentations qui ne sont pas normalisées dans tous les exemples avec des types récursifs. L'origine de cette impossibilité est dans l'adoption de la réduction de tête faible, qui ne constitue pas une stratégie normalisante en $\lambda-$calcul.

Section 4: Cette section est consacrée à la méthode des fils. Une introduction s'impose à ce sujet: Étant donné un processus *P*, le fil de *P* est l'ensemble de tous les processus obtenus par réduction à partir de *P*. Dans [7], Krivine présente une définition de modèle de réalisabilité obtenu en prenant comme ensemble $\perp\!\!\!\perp$ le complément de la réunion de tous les fils débutant en un processus formé par un terme (plus précisément: une quasi-preuve, qui sont définies comme les termes sans continuations) et une constante de pile indexée par ce terme. Krivine appelle ce modèle le "générique standard" et la constante de pile associée à une quasi preuve est nommée "le boot", car elle garde l'information du début de l'exécution. D'autant plus que, si on introduit une instruction capable d'extraire l'indice de la constante de pile et le mettre en tant que terme dans la pile (ou en tête), on est capable de "rebooter" l'exécution à tout moment. Évidemment, pour pouvoir garder l'information du boot dans la constante de pile, il faut qu'aucune instruction ne soit susceptible de modifier cette constante. C'est le cas pour les processus qui ne contiennent pas de continuation ou même pour tous les processus qui apparaissent au cours du fil d'une quasi-preuve (puisque toutes les continuations introduites par cc ont la constante de pile du départ).

En regardant les démonstrations des propriétés de réalisabilité dans le modèle générique de Krivine, on peut remarquer un raisonnement simple et puissant qu'on peut faire dans ce modèle: Si un processus *P* n'appartient pas à $\perp\!\!\!\perp$, alors il est dans un fil, ce qui veut dire qu'il y a une tête de fil qui se réduit au processus *P*. Alors que dans le modèle générique de Krivine, on peut déterminer à quel fil appartient *P* en se basant sur la constante de pile, si on essaye de faire le même raisonnement dans un programme interactif, on trouve deux problèmes fondamentaux:

Dès qu'une instruction d'interaction arrive en tête, la réduction devient non déterministe et on perd la notion de fil ou bien on "ramifie" les fils qui deviennent des arbres. Dans ce cas, la linéarité du fil étant perdue, on a énormément affaibli l'information "*P* appartient à un fil", puisque on aurait à déterminer dans quelle branche du fil est *P*, ce qui revient à se demander dans quelle session d'exécution, parmi toutes celles qui sont possibles, apparaît *P*. On abandonne donc l'idée de ramifier les fils, et on préfère démarrer un nouveau fil une fois que l'interaction s'est produite, mais dans ce cas il faut signaler que les continuations dans le nouveau processus contiennent la constante de boot de l'ancien fil puisqu'elles furent calculées dans ce fil. Pour cette raison on abandonne aussi la tentative d'identifier le fil auquel appartient *P* par inspection de la constante de pile. On ne peut donc plus utiliser le modèle de réalisabilité défini par la réunion de tous les fils. Heureusement, rien n'empêche de prendre des modèles de fils définis comme le complément d'un seul fil, puis quand on redémarre l'exécution après une interaction, on peut prendre la réunion du premier et du deuxième fil et ainsi de suite. Ce choix donne une discussion finie sur *P*, qui implique uniquement les fils nécessaires à la compréhension du programme en question.

Le deuxième problème sera expliqué en détail dans la description de la section 5, mais on peut en donner une idée maintenant: pour avoir des programmes qui implémentent le "garbage collector", il est nécessaire d'utiliser la définition intuitionniste du quantificateur existentiel. Ce choix étant fait, on est amené à définir les jeux associés à certaines formules du *deuxième ordre* (nommées dans ce travail "$\mathcal{L}_G$-formules"). Comme on verra dans la section 5, ces jeux comportent le choix par l'opposant d'un paramètre du deuxième ordre déterminé par le choix d'une pile. Le marquage de la tête de fil par une constante de pile devient donc impossible puisque la constante est fournie par l'opposant. Or, cette liberté de réponse donnée à l'opposant est indispensable pour pouvoir prouver notre *Main Theorem* et donc il n'est pas question de s'en passer.

Finalement, suite à l'analyse de ces contraintes, on décide d'utiliser des modèles de fils dont l'ensemble $\perp\!\!\!\perp$ est le complément d'une réunion finie de fils dont l'origine est arbitraire.

Pour résumer cette discussion: Avec notre choix on perd "l'universalité" du modèle de fils "générique" (un seul modèle qui peut être utilisé en étudiant n'importe quel processus formé par une quasi-preuve). On perd aussi l'étiquette du boot dans la constante de pile ce qui conduit à ne pas savoir directement à quel fil appartient un processus qui n'est pas dans $\perp\!\!\!\perp$. En échange on gagne la possibilité d'étudier des fils de n'importe quel processus, le fait d'avoir des modèles de fils adaptés à l'interaction (en particulier aux jeux) et la possibilité de traiter des formules dont la spécification algorithmique nous assure d'avoir le "garbage collector".

Dans la première sous-section, on s'occupe de la définition des modèles de fils et d'expliquer comment en tirer profit sur des exemples simples et fort utiles par la suite.

La deuxième sous-section présente la substitution dite "dynamique". Cette substitution est un sous-produit de l'utilisation des modèles de fils pour décrire la réduction d'un processus. En effet, la substitution dynamique est un outil essentiel pour comprendre la *composition de stratégies* dans des jeux de réalisabilité. Cette composition pose des problèmes difficiles qui sont étudiés dans la section 6. Pour donner un "avant-goût" de cette définition, nous pouvons dire que l'idée est de substituer des termes à des instructions, tout en gardant la maîtrise des effets que cette substitution comporte. Dans cette section, on montre qu'en prenant des instructions assez restrictives (avec une règle de réduction qui n'accepte que des piles très particulières), on peut trouver un ensemble de termes qui se substituent à ces instructions de façon satisfaisante. On peut dire que la substitution dynamique est un raccourci pour dénoter brièvement le résultat d'un raisonnement compliqué qu'on utilise systématiquement.

La troisième sous-section montre comme application immédiate de la méthode de fils, quelle est la spécification de tous les termes réalisant la loi de Peirce, prouvant ainsi que cc est le terme le plus simple réalisant cette loi et que tous les autres réalisateurs sont en quelque sorte des cc affaiblis.

Section 5: Dans cette section sont présentés les jeux de réalisabilité. Un peu d'histoire: Lorsque j'ai commencé ma recherche en 2004, l'interprétation des formules en tant que jeux dans la réalisabilité de Krivine était cela exposée dans l'article [6]. Cette présentation associe des jeux aux formules arithmétiques en forme normale prénexe. Malgré la définition intuitionniste du quantificateur existentiel qui est donnée dans cet article ($\exists x\varphi := \forall X[\forall x(\varphi \to X) \to X]$), la variable X est vite remplacée par $\perp$ dans chaque preuve et même la règle de de réduction pour les constantes d'interaction n'impose à l'opposant aucune contrainte dans le choix sur la pile.

Ayant été confronté très vite au problème d'interpréter des preuves comme des combinateurs de stratégies, je me suis aperçu qu'il est nécessaire d'utiliser la forme intuitionniste de l'existentiel pour éviter que la pile cumule des valeurs auxiliaires qui ne sont plus nécessaires. C'est parce qu'on a besoin de faire des substitutions sûres pour étudier les combinateurs de stratégies, qu'il importe de ne pas avoir une pile inconnue gardant des arguments qui pourraient entrer dans le calcul à tout moment. Comme on l'a vu plus haut, si un terme réalise une formule comportant des quantificateurs existentiels intuitionnistes, il nettoie la pile avant chaque interaction avec l'opposant. Si, par contre, on utilise la forme classique de la formule, cette propriété n'est plus assurée.

Peu après est créée par Krivine la définition des jeux dites "$\mathscr{UVA}$"; lesquels sont associés aux formules Π^1_1 (donc, en particulier aux formules du premier ordre). Cette présentation est faite dans un langage – dit langage des formules "normales" – avec une définition classique du quantificateur existentiel, des conjonctions et des disjonctions. De plus, on ne peut pas utiliser l'égalité de Leibniz puisqu'il s'agit d'une formule du deuxième ordre. Par exemple:

$$\exists x\forall y\exists z\varphi(x,y,z) := \forall x\{\forall y[\forall z(\varphi(x,y,z) \to \perp) \to \perp] \to \perp\} \to \perp$$

a les quantificateurs existentiels classiques et peut être interprétée comme un jeu dans le sens de [7]. Sa présentation intuitionniste:

$$\forall X\{\forall x\{\forall y[\forall Z\forall z(\varphi(x,y,z) \to Z) \to Z] \to X\} \to X\}$$

ne peut pas être interprétée comme un jeu dans le sens de [7] puisque ce n'est pas une formule normale. Cependant, dans la définition des jeux donnée dans [6] il est possible de jouer avec cette formule, en remarquant qu'un terme qui réalise la formule en version intuitionniste, réalise aussi son homologue classique.

La définition des jeux associés aux formules générales du second ordre paraît actuellement hors de portée. Il fallait donc trouver une façon d'interpréter au moins les formules de deuxième ordre qui sont nécessaires pour exprimer les formes intuitionnistes des existentiels, conjonction, disjonction et l'égalité de Leibniz. Le langage $\mathcal{L}_G$ présenté dans la première sous-section de la présente section répond bien à ces requis.

Puisque les formules correspondant aux types de données les plus intéressants ne sont pas des formules de $\mathcal{L}_G$, on a prévu dans la définition de $\mathcal{L}_G$ une place spécifique servant à la relativisation des quantificateurs. Maintenant, on a un cadre général qui contient les formules du premier ordre relativisées totalement ou partiellement aux types de données, avec les connecteurs et les quantificateurs en forme intuitionniste.

Dans la deuxième sous-section on explique comment définir les jeux $\mathscr{UVA}$ dans le langage $\mathcal{L}_G$. C'est une généralisation de la définition donnée dans [7].

La troisième sous-section est dédiée à l'implémentation des jeux $\mathscr{UVA}$ dans la machine de Krivine. Cette définition est aussi une généralisation de celle qui est présentée dans [7], où la nouveauté consiste à traiter les quantificateurs du deuxième ordre qui sont dans $\mathcal{L}_G$.

Dans la quatrième et dernière sous-section, le théorème principal ("Main Theorem"), qui apparaît déjà dans [6], est la connexion entre les jeux et la réalisabilité. Plus précisément, un terme qui réalise une formule de $\mathcal{L}_G$, implémente le jeu $\mathscr{UVA}$ associé à cette formule et le joueur "qui défend" le théorème, a une stratégie gagnante pour ce jeu. De plus, si la formule est totalement relativisée à des types de données, alors le terme est "juge et partie" dans le jeu. En effet, il implémente le jeu et joue une stratégie gagnante à la place du défenseur de la formule. C'est ici, quand on prouve ce théorème, qu'on voit à quoi servent les restrictions imposées sur les variables du deuxième ordre des formules de $\mathcal{L}_G$. En effet, il y a une importante propriété de monotonie dans les valeurs de vérité qui découle de ces restrictions et qui est essentielle dans la preuve du Main Theorem.

Section 6: La section 6 contient plusieurs exemples et montre en action toute la théorie exposée dans les sections précédentes.

On démarre avec les exemples les plus simples de jeux avec back-tracking. Plus concrètement, les sous-sections 1 et 2 sont consacrées à étudier le cas des formules $\exists x \forall y (f(x,y) = 0)$ et $\exists x \forall y (f(x,y) \neq 0)$. On caractérise ici les termes qui réalisent ces formules comme ceux qui implémentent les jeux associés à ces formules (dans un sens, c'est le Main Theorem, mais il est remarquable que la réciproque soi aussi vraie pour ces formules). On montre ici aussi les "schémas de fils" qui seront utilisés par la suite et on explique que le back-tracking a une contrepartie sémantique dans le modèle $\perp\!\!\!\perp_G$ constitué par les processus qui ont une stratégie gagnante. Il y a une distinction importante entre l'utilisation de l'égalité (cas de $f(x,y) = 0$) et de l'inégalité (cas de $f(x,y) \neq 0$). Tandis que la première donne des exécutions qui s'arrêtent sur un pointeur indiquant la position finale, la deuxième s'arrête brutalement dès que l'opposant ne peut pas donner une réponse satisfaisante.

Dans la sous-section 3, on explique l'effet sur les jeux de la relativisation des quantificateurs. Le résultat déjà bien connu pour les entiers (voir [6] et [7]) est généralisé sans aucun effort particulier aux types de données: les réalisateurs des formules relativisées jouent eux mêmes une stratégie gagnante pour le défenseur de la formule – cela découle de la définition de l'interaction –. Cependant, dans le nouveau cadre il y a deux possibilités pour relativiser un quantificateur: Le relativiser à un type de données ou de façon plus restrictive, le relativiser aux représentants canoniques. Dans le premier cas on doit utiliser des opérateurs de mise en mémoire pour les constantes d'interaction.

Dans la sous-section 4, on donne, au niveau des réalisateurs, des mécanismes pour passer de la formule relativisée aux canoniques à la formule relativisée à tout le type de données et vice-versa. On obtient ainsi un terme qui implémente une stratégie gagnante jouant les mêmes individus que celle du départ.

La sous-section 5 montre une caractérisation qui généralise celle vue dans les sous-sections 1 et 2: Tout terme qui implémente le jeu d'une formule partiellement ou totalement relativisée à des canoniques est un réalisateur de cette formule dans tous les modèles de réalisabilité. Autrement dit, réaliser une de ces formules dans $\perp\!\!\!\perp_G$ équivaut à la réaliser dans tous les modèles.

Finalement on s'attaque au *leit motiv* de ce travail: l'analyse d'une preuve comme combinateur de stratégies. Considérez les deux formules suivantes: $\Psi := \exists\overset{\text{int}}{x}\forall\overset{\text{int}}{y}(f(x,y)=0)$ et $\Upsilon := \exists\overset{\text{int}}{x'}\forall\overset{\text{int}}{y'}(g(x',y')=0)$. On met la formule $\Psi \to \Upsilon$ sous forme prénexe :

$$\Phi' := \exists\overset{\text{int}}{x'}\forall\overset{\text{int}}{y'}x\exists\overset{\text{int}}{y}[(f(x,y){=}0) \to (g(x',y'){=}0)]$$

À l'origine, ce problème était formulé dans la définition des jeux de [6] ; il était donc important de travailler avec des formes prénexes et avec la partie propositionnelle en forme équationnelle. Pour obtenir cette forme, on considère, dans le modèle de départ, une fonction $h(x,y,x',y')$ telle que:

$$\forall xyx'y'[(h(x,y,x',y')=0) \leftrightarrow ((f(x,y){=}0) \to (g(x',y'){=}0))]$$

et on définit $\Phi := \exists\overset{\text{int}}{x'}\forall\overset{\text{int}}{y'}x\exists\overset{\text{int}}{y}(h(x,y,x',y'){=}0)$ Ainsi, il est immédiat que tout réalisateur de Φ est aussi un réalisateur de Φ', c'est-à-dire de la forme normale prénexe de l'implication $\Psi \to \Upsilon$. On sait alors que $\Phi,\Psi \vdash \Upsilon$ et on veut étudier comment une preuve de ce théorème combine des stratégies gagnantes pour Φ et Ψ pour obtenir une stratégie gagnante pour Υ.

Dans la sous-section 5 on considère la formule Φ et on étudie comment sont les fils dans une partie. Ce jeu est bien plus complexe que celui vu dans les cas précédent parce qu'il y a plus de possibilités. Pour mieux décrire ce jeu, on organise les positions jouées dans un arbre (dans la sémantique des jeux, cette pratique est usuelle et ces arbres sont appelés "arenas").

Finalement, dans la sous-section 6 on s'attaque à la description du combinateur en utilisant les schémas de fils définis dans les sous-sections 4 (pour Ψ) et 5 (pour Φ) pour décrire, à l'aide de la substitution dynamique définie dans la section 4, le comportement du combinateur obtenu par une preuve du théorème $\Phi,\Psi \vdash \Upsilon$.

0.2. **La réalisabilité de Krivine est un sujet multidisciplinaire.** À partir de ce que nous venons de commenter, nous pouvons énumérer les domaines des mathématiques et de l'informatique qui sont concernés par la réalisabilité de Krivine. Ainsi cette théorie repose sur quatre "pattes" dont deux relèvent de l'informatique et les deux autres de la logique.

(1) La réalisabilité est de la programmation: La machine de Krivine constitue un modèle de programmation impérative. Cette machine comporte un ensemble d'instructions, qui peut être enrichi à tout moment avec une remarquable simplicité et sans effets sur la partie du calcul déjà établie. Cet aspect de "modularité" qui est très cher aux programmeurs.

    De plus, les instructions qu'on a eu besoin d'introduire jusqu'ici pour réaliser les axiomes des mathématiques, ont toujours un sens clair et bien connu en programmation. Ainsi, le tiers exclu est le "call-with-current-continuation" et fait apparaître des pointeurs (les continuations); l'axiome de choix dépendant est l'horloge ou bien la signature de fichiers; l'axiome de choix général est obtenu à partir des instructions de lecture et d'écriture dans une mémoire globale ("tas"). D'autre part, une récente découverte de Krivine et Legrangérard montre que les formules valides sont des spécifications de protocoles réseau (voir [9] et [10]).

    La réalisabilité de Krivine est une théorie qui touche au cœur de la programmation, établissant des liens clairs entre les axiomes des mathématiques et la programmation impérative.

(2) La réalisabilité est de la théorie de la démonstration: Elle a un système de typage qui permet démontrer comme on le fait en mathématiques. En particulier le fait de réaliser les axiomes permet d'éviter de les charger comme des hypothèses tout au long de la preuve. La distinction entre ces deux modes de preuve paraît mineure. Elle n'est certainement pas très importante tant qu'on connaît des réalisateurs pour l'axiome. Prenons l'exemple de l'énoncé *L'axiome de choix implique le théorème de Tychonoff.* La distinction entre une preuve de ce théorème et une preuve du théorème de Tychonoff dans une théorie avec axiome du choix est capitale du point de vue de la programmation : ce n'est pas la même chose d'avoir un programme capable de trouver un réalisateur de Tychonoff à partir d'un réalisateur de l'axiome de choix que d'avoir explicitement un réalisateur de Tychonoff.

    On évite aussi d'introduire la logique classique par "non-non traduction" ; le style des preuves formelles dans cette théorie est ainsi beaucoup plus proche de celui des preuves informelles en mathématiques.

(3) La réalisabilité est une généralisation du forcing : Elle a des méthodes de travail similaires et permet de construire des modèles de l'analyse ou de la théorie des ensembles. On peut d'ailleurs espérer

de trouver des nouvelles preuves d'indépendance qu'on ne pourrait pas obtenir par forcing.

(4) La réalisabilité est de la théorie des jeux: Elle permet de spécifier des formules comme des jeux et d'établir des liens entre les stratégies gagnantes du jeu associé à une formule et les réalisateurs de cette formule. Les protocoles réseau dont on a parlé plus haut sont un cas particulier de jeux associés à des formules, plus précisément à des formules valides.

0.3. Contribution.

Je voudrais expliciter ici les idées et les techniques qui correspondent à mon travail personnel, par rapport à ce qui était connu auparavant. Ces idées ont été présentés dans l'introduction et seront développés dans les six sections qui suivent. J'en ferai donc ici une simple énumération :

Section 1:

- L'introduction dans le langage des formules étendues et des règles de typage pour la nouvelle flèche qu'elles comportent. Cependant, Krivine avait déjà manipulé des valeurs de vérité qui correspondent à des formules étendues avant que cette définition ne soit donnée.

Section 3:

- La définition de type de données. Cependant, il faut signaler que l'idée : *"avoir un opérateur de mise en mémoire c'est comme avoir un type de données"* me fut suggérée par Krivine lors d'un entretien de travail.
- La preuve que, pour aucun type de données, on ne peut réaliser, dans tous les modèles de réalisabilité, qu'il contient tous les individus ($\forall x\varphi(x)$ où φ est un type de données). C'est une généralisation du résultat déjà connu pour les entiers ; mais le fait de le prouver en utilisant les opérateurs de mise en mémoire permet de le généraliser à tous les types de données.
- La preuve qu'il n'est pas possible d'utiliser des représentations canoniques normalisées pour les types de données récursifs.

Section 4:

- La définition des modèles de fils.
- La propriété de réduction des opérateurs de mise en mémoire énoncée dans 4.5 et sa généralisation 4.9. La formulation actuelle, beaucoup plus compacte, me fut suggérée par Krivine lors d'un entretien de travail.
- La définition de la substitution dynamique et tout ce qui est présenté dans la sous-section 2 de la section 4.
- La présentation de l'exemple de la loi de Peirce. J'ignore si la réciproque était déjà connue.

Section 5:

- La définition du langage $\mathcal{L}_G$, lequel est une extension du langage des formules normales présenté par Krivine dans [7]. Toute l'adaptation des jeux et son implémentation au nouvel ensemble de formules; en particulier les paramètres de Δ et son usage pour instancier les variables du deuxième ordre dans les formules de $\mathcal{L}_G$.

Section 6: Dans cette section toute la présentation, les commentaires et les schémas choisis pour représenter l'algorithme du combinateur de stratégies correspondent à mon travail de recherche.

1. **Syntax**

1.1. **Languages.**

We shall define a λ-calculus with instructions denoted by Λ_c, together with a language of stacks denoted by Π. While terms represents programs, a stack represents a list of *arguments* or, in other words, an *environment* under which a program can be performed. In order to build these languages, we must begin choosing sets of constants and sets of variables:

Definition 1.1. *Let us consider the three countable sets*

- *A set of variables for Λ_c-terms that we will denote as* Λ_c`-var`
- *A set of constants for Λ_c-terms that we will denote as* Λ_c`-constant`
- *A set of constants for stacks that we will denote as* Π`-constant`

The Λ_c-constants will be denoted in general by capital letters H, U, sometimes by the lowercase letter k. For definitions involving Λ_c-constants, we will represent any constant by κ. These constant are also called *instructions*

The Λ_c-variables will be denoted by lowercase letters x, y, z, v, w or a, b, c.

The Π-constants will be denoted as π_i, where i is an integer.

For technical reasons, we define first the set of (possibly open) stacks $\overline{\Pi}$ but we are interested only on closed stacks, which set we will denote by Π. The definitions of Λ_c and $\overline{\Pi}$ are mutually recursive:

Definition 1.2.

$$\begin{aligned}
<\Lambda_c\texttt{-term}> ::= &\; k_{<\overline{\Pi}\texttt{-stack}>} \mid <\Lambda_c\texttt{-constant}> \mid <\Lambda_c\texttt{-var}> \mid \\
&\; (<\Lambda_c\texttt{-term}>)<\Lambda_c\texttt{-term}> \mid \lambda<\Lambda_c\texttt{-var}><\Lambda_c\texttt{-term}> \\
<\overline{\Pi}\texttt{-stack}> ::= &\; <\Pi\texttt{-constant}> \mid <\Lambda_c\texttt{-term}>.<\overline{\Pi}\texttt{-stack}>
\end{aligned}$$

Each stack π gives, in an injective way, a Λ_c-term denoted by k_π. Such a term is called *a continuation*.

Intuitively, a continuation k_π is an instruction saving an environment (a stack) π to restore it later.

As usually in Krivine's papers, we adopt for application of Λ_c-terms the notation $(t)u$ instead of tu.

We define the free variables of Λ_c-terms and stacks as follows:

Definition 1.3. $\mathrm{FV} : \Lambda_c \cup \overline{\Pi} \to \Lambda_c$`-var`

- $\mathrm{FV}(k_\pi) := \mathrm{FV}(\pi)$, *where π is a Π-stack.*
- $\mathrm{FV}(\kappa) := \emptyset$, *where κ is an instruction.*
- $\mathrm{FV}(x) := \{x\}$, *where x is a Λ_c-variable.*
- $\mathrm{FV}((t)u) := \mathrm{FV}(t) \cup \mathrm{FV}(u)$, *where t, u are Λ_c-terms.*
- $\mathrm{FV}(\lambda xt) := \mathrm{FV}(t) \setminus \{x\}$, *where x is a Λ_c-variable and t is a Λ_c-term.*
- $\mathrm{FV}(\pi_i) := \emptyset$, *where π_i is a Π-constant.*
- $\mathrm{FV}(t.\pi) := \mathrm{FV}(t) \cup \mathrm{FV}(\pi)$, *where t is a Λ_c-term and π is a Π-stack.*

A Λ_c-term t is closed *if and only if* $\mathrm{FV}(t) = \emptyset$. *A Π-stack π is closed if and only if* $\mathrm{FV}(\pi) = \emptyset$. *We denote the set of closed Λ_c-terms by Λ_c^0 and the set of closed stacks by* Π.

Thus, a closed stack is a finite sequence of closed Λ_c-terms ended by a Π-constant.

Definition 1.4. *We define the length of the stacks $\ell : \Pi \to \mathbb{N}$ by induction as follows:*

- $\ell(\pi_i) = 0$*, where* π_i *is a* Π*-constant.*
- $\ell(t.\pi) = \ell(\pi) + 1$*, where* t *is a closed* Λ_c*-term.*

Thereafter, we will define a language $\mathcal{P}$, which makes it possible to express mathematical properties and to type Λ_c-terms. Therefor, w define first the language of terms:

Definition 1.5. *Let be* $\mathcal{V}_1$ *a countable set of* first order variables *and* $\mathcal{F}$ *a countable set of* function symbols. *For each arity k, we denote as* $\mathcal{F}^k$ *the set of all k-ary function symbols. The* 0*-ary function symbols are called* constants.

$$<\mathcal{T}\texttt{-term}> ::= \ <\mathcal{V}_1\texttt{-var}> \ | \ <\mathcal{F}^k\texttt{-funct}>\underbrace{<\mathcal{T}\texttt{-term}>\ldots<\mathcal{T}\texttt{-term}>}_{k}$$

where $k \in \mathbb{N}$

The free variables *of terms are defined as usually. Given a term* τ*, we denote by* $\mathrm{FV}(\tau)$ *the set of the free variables of* τ[1].

A list of terms $\tau_1, \ldots, \tau_k$ will be sometimes abbreviated as $\vec{\tau}$.

We have at least the function symbols $\mathrm{s}, +, \times$ for the successor, the addition and the multiplication respectively and a constant symbol 0 for the zero.

In order to built extended formulæ we must add some sets of variables:

Definition 1.6. *Let us consider the following sets of variables:*

- *For each arity k, lets consider a countable set of k-ary second order variables which we will denote as* $\mathcal{V}_2^k$*.*
- *For each arity k, lets consider a countable set of k-ary variables that we will denote as* $\mathcal{E}^k$*.*

Notation 1.7. We denote as $\mathcal{V}_2$ the set $\bigcup_{k\in\mathbb{N}} \mathcal{V}_2^k$ containing all second order variables and as $\mathcal{E}$ the set $\bigcup_{k\in} \mathcal{E}^k$. The variables belonging to $\mathcal{E}$ will be called Λ_c*-set variables*. Usually we will denote second order variables by capital letters, specially X, Y, W and the Λ_c-set variables by lowercase greek letters, almost always ε_i where i is an integer. A k-ary Λ_c-set variable ε followed by k terms will be called a Λ_c*-set atom*.

The useful of Λ_c-set variables to describe semantics will be clear in the following sections.

We now define the language $\mathcal{P}$ of *extended formulæ* as a second order language with usual implication $\to$ and another implication $\rightsquigarrow$ of an extended formula by a Λ_c-set atom:

[1]since there is no ambiguity, we will use always the notation FV for free variables.

Definition 1.8.

$$\begin{aligned} <\mathcal{P}\texttt{-fla}> ::= \;& \top \mid <\mathcal{V}_2^k\texttt{-var}>\underbrace{<\mathcal{T}\texttt{-term}>\ldots<\mathcal{T}\texttt{-term}>}_{k} \mid \\ & <\mathcal{E}^h\texttt{-var}>\underbrace{<\mathcal{T}\texttt{-term}>\ldots<\mathcal{T}\texttt{-term}>}_{h} \rightsquigarrow <\mathcal{P}\texttt{-fla}> \mid \\ & <\mathcal{P}\texttt{-fla}> \to <\mathcal{P}\texttt{-fla}> \mid \forall<\mathcal{V}_1\texttt{-var}><\mathcal{P}\texttt{-fla}> \mid \\ & \forall<\mathcal{V}_2^k\texttt{-var}><\mathcal{P}\texttt{-fla}> \qquad \textit{where } h,k \in \mathbb{N} \end{aligned}$$

The free variables *of extended formulæ are defined in the usual way and we denote the set of the free variables of* φ *by* $\mathrm{FV}(\varphi)$. *We remark that, given an extended formula* $\psi := \varepsilon\tau_1\ldots\tau_k\rightsquigarrow\varphi$, $\mathrm{FV}(\psi) = \{\varepsilon\}\cup\bigcup_{i=1}^{i=k}\mathrm{FV}(\tau_i)\cup\mathrm{FV}(\varphi)$.

Remark 1.9. A Λ_c-set atom is not a $\mathcal{P}$-formula. However, the "double negation" of $\varepsilon\tau_1\ldots\tau_k$ can be written as $\forall X[(\varepsilon\tau_1\ldots\tau_k \rightsquigarrow X) \to X]$, which is a $\mathcal{P}$-formula.

Notation 1.10. As usually in Krivine's papers, we will denote $\varphi_1{\to}(\varphi_2{\to}\varphi_3)$ as $\varphi_1, \varphi_2 \to \varphi_3$. Moreover, we also define the analogous abbreviations for the connector $\rightsquigarrow$:

$$\varepsilon_1\vec{\tau};\varepsilon_2\vec{\sigma} \rightsquigarrow \varphi := \varepsilon_1\vec{\tau} \rightsquigarrow (\varepsilon_2\vec{\sigma} \rightsquigarrow \varphi)$$

$$\varepsilon\vec{\tau};\varphi_1{\to}\varphi_2 := \varepsilon\vec{\tau}{\rightsquigarrow}(\varphi_1{\to}\varphi_2)$$

The language of types $\mathcal{L}$ is the sub language of $\mathcal{P}$ defined without the constructor $\rightsquigarrow$, i.e. the language of the second order logic with the connector $\to$ and the quantifier $\forall$:

Definition 1.11.

$$\begin{aligned} <\mathcal{L}\texttt{-fla}> ::= \;& \top \mid <\mathcal{V}_2^k\texttt{-var}>\underbrace{<\mathcal{T}\texttt{-term}>\ldots<\mathcal{T}\texttt{-term}>}_{k} \mid \\ & <\mathcal{L}\texttt{-fla}> \to <\mathcal{L}\texttt{-fla}> \mid \forall<\mathcal{V}_1\texttt{-var}><\mathcal{L}\texttt{-fla}> \mid \\ & \forall<\mathcal{V}_2^k\texttt{-var}><\mathcal{L}\texttt{-fla}> \qquad \textit{where } k \in \mathbb{N} \end{aligned}$$

The language $\mathcal{L}$ of the second order logic will be used to type the Λ_c-terms and to write mathematical properties, while the language of extended formulæ will be useful from Section 2 to describe semantics.

We introduce the following abbreviations:

Definition 1.12.

$$\boxed{\begin{array}{c}\begin{array}{rcl}\bot & = & \forall X X\end{array}\\[1em] \textit{where } X\in\mathcal{V}_2^0\end{array}}$$

$$\boxed{\begin{array}{c}\begin{array}{rcl}\tau_1=\tau_2 & = & \forall W(W\tau_1\to W\tau_2)\end{array}\\[1em] \textit{where } W\in\mathcal{V}_2^1 \textit{ and } \tau_1,\tau_2\in\mathcal{T}\end{array}}$$

$$\boxed{\begin{array}{c}\begin{array}{rcl}\neg\varphi & = & \varphi\to\bot\\ \varphi_1\vee\cdots\vee\varphi_k & = & \forall X((\varphi_1\to X),\ldots,(\varphi_k\to X)\to X)\\ \varphi_1\wedge\cdots\wedge\varphi_k & = & \forall X((\varphi_1,\ldots,\varphi_k\to X)\to X)\\ \exists x(\varphi_1,\ldots,\varphi_k) & = & \forall X(\forall x(\varphi_1,\ldots,\varphi_k\to X)\to X)\\ \exists Y(\varphi_1,\ldots,\varphi_k) & = & \forall X(\forall Y(\varphi_1,\ldots,\varphi_k\to X)\to X)\end{array}\\[1em] \textit{where } X\in\mathcal{V}_2^0\ X\notin \mathrm{FV}(\varphi_1,\ldots,\varphi_k) \textit{ and } \varphi_1,\ldots,\varphi_k \textit{ are extended formulæ}\end{array}}$$

$$\boxed{\begin{array}{c}\begin{array}{rcl}\mathrm{int}(x) & = & \forall X(\forall y(Xy\to X(\mathrm{s}y)),X0\to Xx)\end{array}\\[1em] \textit{where } X\in\mathcal{V}_2^1 \textit{ and } \mathrm{s} \textit{ is the symbol of the successor function}\end{array}}$$

$$\boxed{\begin{array}{c}\begin{array}{rcl}\forall^{\psi_1}x_1\ldots\forall^{\psi_k}x_k\varphi & = & \forall x_1\ldots\forall x_k(\psi_1(x_1),\ldots,\psi_k(x_k)\to\varphi)\\ \exists^{\psi_1}x_1\ldots\exists^{\psi_k}x_k\varphi_1,\ldots\varphi_s & = & \exists x_1\ldots\exists x_k(\psi_1(x_1),\ldots,\psi_k(x_k),\varphi_1,\ldots,\varphi_s)\end{array}\\[1em] \textit{where } \varphi,\psi_1\ldots\ \psi_k,\ \varphi_1,\ldots,\varphi_s \textit{ are extended formulæ and each } \psi_i \textit{ depends on } x_i\end{array}}$$

Hence, existential quantifiers, disjunction and conjunction are defined as second order formulæ. The first order language $\mathcal{L}_1$ is defined by:

Definition 1.13.

$$\begin{array}{rl} <\mathcal{L}_1\texttt{-fla}> ::= & \top \mid \bot \mid <\mathcal{T}\texttt{-term}> = <\mathcal{T}\texttt{-term}> \mid \\ & <\mathcal{T}\texttt{-term}> \neq <\mathcal{T}\texttt{-term}> \mid \\ & <\mathcal{V}_2^k\texttt{-var}>\underbrace{<\mathcal{T}\texttt{-term}>\ldots<\mathcal{T}\texttt{-term}>}_{k} \mid \\ & <\mathcal{L}_1\texttt{-fla}> \to <\mathcal{L}_1\texttt{-fla}> \mid \forall<\mathcal{V}_1\texttt{-var}><\mathcal{L}_1\texttt{-fla}> \mid \\ & \exists<\mathcal{V}_1\texttt{-var}><\mathcal{L}_1\texttt{-fla}> \mid <\mathcal{L}_1\texttt{-fla}>\vee<\mathcal{L}_1\texttt{-fla}> \mid \\ & <\mathcal{L}_1\texttt{-fla}>\wedge<\mathcal{L}_1\texttt{-fla}> \qquad \textit{where } k\in\mathbb{N}\end{array}$$

Where the symbol $\neq$ is a new predicate symbol. The formula $\tau_1\neq\tau_2$ is atomic and hence different of $\tau_1=\tau_2\to\bot$.

1.2. **Typing rules.**

We must define the typing rules. In order to do that, we must define the sequents. The usual declarations in λ-calculus are of the form $x{:}\varphi$, where x is a variable and φ is a type (a formula). Here we will type with extended formulæ and hence we must also admit delcarations $x\in\varepsilon\vec{\tau}$, where x is a variable and $\varepsilon\vec{\tau}$ is a Λ_c-set atom.

Definition 1.14.

$$\begin{array}{rcl}
<\texttt{declaration}> & ::= & <\mathcal{V}_1> : <\mathcal{P}\texttt{-fla}> \mid \\
& & <\mathcal{V}_1\texttt{-var}> \in <\Lambda_c\texttt{-set atom}> \\
<\texttt{judgement}> & ::= & <\Lambda_c\texttt{-term}> : <\mathcal{P}\texttt{-fla}> \\
<\texttt{typing hypothesis}> & ::= & [] \mid <\texttt{typing hypothesis}>, <\texttt{declaration}> \\
<\texttt{sequent}> & ::= & <\texttt{typing hypothesis}> \vdash <\texttt{judgment}>
\end{array}$$

We will denote the judgements by capital greek letters, almost always Γ, Δ, Σ. For typing rules, the judgments will be considered equals at less of permutations, i.e. the order of declarations in a judgment is not relevant.

Definition 1.15. *The* free variables *of a judgment are defined by induction:*

- $\mathrm{FV}([]) = \emptyset$,
- $\mathrm{FV}(\Gamma, t{:}\varphi) = \mathrm{FV}(\Gamma) \bigcup \mathrm{FV}(\varphi)$,
- $\mathrm{FV}(\Gamma, u{\in}\varepsilon\tau_1 \ldots \tau_n) = \mathrm{FV}(\Gamma) \bigcup \{\varepsilon\} \bigcup_{i=1}^{i=n} \mathrm{FV}(\tau_i)$

The substitution in Λ_c-terms is defined in the usual way. The substitution of first order variables in formulæ is also defined in the usual way.

Given an extended formula, we define a substitution of a (free) second order variable by an extended formula :

Definition 1.16. *Let $x_1, \ldots, x_k$ be k first order variables, X a second order variable of arity k, φ and ψ arbitrary extended formulæ. We define the substitution $\varphi[^{\psi}/_{Xx_1\ldots x_k}]$ by induction on the length of φ:*

- *if $X \notin \mathrm{FV}(\varphi)$, then $\varphi[^{\psi}/_{Xx_1\ldots x_k}]$ is φ,*
- *if φ is $X\tau_1 \ldots \tau_k$, then $\varphi[^{\psi}/_{Xx_1\ldots x_k}]$ is $\psi[^{\tau_1\cdots\tau_k}/_{x_1\ldots x_k}]$,*
- *if φ is $\varphi_1 \to \varphi_2$, then $\varphi[^{\psi}/_{Xx_1\ldots x_k}]$ is $\varphi_1[^{\psi}/_{Xx_1\ldots x_k}] \to \varphi_2[^{\psi}/_{Xx_1\ldots x_k}]$,*
- *if φ is $\varepsilon\vec{\tau} \rightsquigarrow \phi_1$, then $\varphi[^{\psi}/_{Xx_1\ldots x_k}]$ is $\varepsilon\vec{\tau} \rightsquigarrow \varphi_1[^{\psi}/_{Xx_1\ldots x_k}]$*
- *if φ is $\forall y\chi$ and $y \notin \mathrm{FV}(\psi)$, then $\varphi[^{\psi}/_{Xx_1\ldots x_k}]$ is $\forall y(\chi[^{\psi}/_{Xx_1\ldots x_k}])$,*
- *if φ is $\forall Y\chi$ and $Y \notin \mathrm{FV}(\psi)$, then $\varphi[^{\psi}/_{Xx_1\ldots x_k}]$ is $\forall Y(\chi[^{\psi}/_{Xx_1\ldots x_k}])$.*

The typing rules are the following:

Definition 1.17.

$$(\mathrm{ax})\ \frac{}{\Gamma \vdash \alpha}\ \textit{where } \alpha \in \Gamma$$

$$(\mathrm{abs})\ \frac{\Gamma,\ x{:}\varphi \vdash t{:}\psi}{\Gamma \vdash \lambda xt{:}\varphi \to \psi}$$

$$(\rightsquigarrow_i)\ \frac{\Gamma, x{\in}\varepsilon\vec{\tau} \vdash t{:}\varphi}{\Gamma \vdash \lambda xt{:}\varepsilon\vec{\tau} \rightsquigarrow \varphi}$$

$$(\mathrm{app})\ \frac{\Gamma \vdash t{:}\varphi \to \psi \quad \Gamma \vdash u{:}\varphi}{\Gamma \vdash (t)u{:}\psi}$$

$$(\rightsquigarrow_e)\ \frac{\Gamma\vdash x{\in}\varepsilon\vec{\tau} \quad \Gamma\vdash t{:}\varepsilon\vec{\tau}\rightsquigarrow\varphi}{\Gamma\vdash (t)x{:}\varphi}$$

$$(\forall^1_i)\ \frac{\Gamma\vdash t{:}\chi}{\Gamma\vdash t{:}\forall x\chi}\ x\notin \mathrm{FV}(\Gamma)$$

$$(\forall^2_i)\ \frac{\Gamma\vdash t{:}\chi}{\Gamma\vdash t{:}\forall X\chi}\ X\notin \mathrm{FV}(\Gamma)$$

$$(\forall^1_e)\ \frac{\Gamma\vdash t{:}\forall x\chi}{\Gamma\vdash t{:}\chi[{}^{\tau}/_{x}]}\ \textit{with } \tau \textit{ a term}$$

$$(\forall^2_e)\ \frac{\Gamma\vdash t{:}\forall X\chi}{\Gamma\vdash t{:}\chi[{}^{\Psi}/_{Xx_1\ldots x_k}]}\ \textit{with } \psi \textit{ a } \mathcal{L}\textit{-formula}$$

The definition of equality (c.f.:1.12) is called the *Leibniz equality*. The classic formulation of this equality is $a=b:=\forall X(Xa\leftrightarrow Xb)$, which intuitively means that two individuals are equals if and only if they have the same properties according to our language. Our asymmetric presentation is due to the rule $(\forall^2_e)$, which allows us to prove $\vdash \lambda x(x)\,\mathrm{id}{:}\forall x\forall y(x{=}y\to y{=}x)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x:x=y\vdash\forall X(Xx\to Xy)}\,(\mathrm{ax})}{x:x=y\vdash x:(Xx\to Xx),Xy\to Xx}\,(\forall^2_e)\qquad \dfrac{\dfrac{}{z:Xx\vdash z:Xz}\,(\mathrm{ax})}{\vdash \mathrm{id}:Xx\to Xx}\,(\to)_i}{x:x=y\vdash (x)\,\mathrm{id}:Xy\to Xx}\,(\to)_e}{x:x=y\vdash (x)\,\mathrm{id}:y=x}\,(\forall^2_i)}{\vdash\lambda x(x)\,\mathrm{id}:x=y\to y=x}\,(\to)_e}{\vdash\lambda x(x)\,\mathrm{id}:\forall x\forall y(x=y\to y=x)}\,(\forall^1_i)$$

where in the second line we have replaced Xy by $\Psi:=Xy\to Xx$.

1.3. **Processes and reduction.**

Definition 1.18. *The processes are the pairs belonging to* $\Lambda^0_c\times\Pi$. *A process* (t,π) *is denoted by* $t\star\pi$. *We say that* t *is in* head position *for the process* $t\star\pi$ *and that* π *is its* argument *or* environment.

We will reduce processes, rather than Λ_c-terms. The reduction is defined by a schemata of reduction rules which we call *reduction system*. Our reduction system is open in the sense that it is possible –whenever necessary– to introduce new reduction rules. This posibility will be used to add a needed property to the system. The following rules will be in any reduction system and they are defined in order to compute the *weak head reduction*:

Definition 1.19.

- $(t)u \star \pi \succ_1 t \star u.\pi$ *(push)*
- $\lambda x t \star u.\pi \succ_1 t[^u/_x] \star \pi$ *(pop)*

Remark 1.20. As in [3], we can work in Classical Logic by adding a control instruction. Indeed, it suffices to add an instruction of Peirce's law type. However, this choice can not be arbitrary, because we need a kind of coherence between typing rules and semantics (soundness lemma). At this point we introduce the needed instruction leaving the check of the adequacy for later.

$$\frac{}{\mathrm{cc}: \forall X \forall Y[((X \to Y) \to X) \to X]}\,(\mathrm{cc})$$

We introduce also, for each stack π, a constant k_π with the reduction rule $k_\pi \star t.\pi' \succ_1 t \star \pi$ for all terms t and stacks π'.

Finally, we introduce for cc the reduction rule $\mathrm{cc} \star t.\pi \succ_1 t \star k_\pi.\pi$ for all terms t and stacks π.

Here we can argue that the continuation k_π is in fact an address to the stack π. Indeed, when we execute k_π, the execution takes the first argument in the stack and following executes this one with the stack π as argument.

This system will be sufficient in order to work in classical arithmetics. More details can be found in [6].

Definition 1.21. *We define* Pl *(the elements of which are called* proof-like terms*) as the set consisting of the Λ_c^0-terms without continuations.*

Definition 1.22. *We will denote by $\succ$ the transitive closure of the relation $\succ_1$. Hence, the notation $P_0 \succ P'$ signifies that there are n processes $P_1, \ldots, P_n$ such that $P_0 \succ_1 P_1 \succ_1 \cdots \succ_1 p_n \succ_1 p'$. We denote also by $\succcurlyeq$ the reflexive closure of $\succ$, i.e. $P \succcurlyeq Q$ if and only if $P \succ Q$ or $P = Q$. Given a process P, the thread of P is the* sequence *of all process obtained from P by reduction. It will be denoted by* th_P. *The* support *of* th_P *is the set*

$$\mathrm{th}_P := \{P' \in \Lambda_0 \times \Pi \mid P \succcurlyeq P'\}$$

For instance, th_P where $P = \lambda x(x)x \star \lambda x(x)x.\pi$ is an infinite sequence of constants and its support is the singleton $\{P\}$.

1.4. **(static) Substitution.**

Thereafter, we define the simultaneous substitution of a list of Λ_c-constants and Π-constants by a corresponding list of terms and stacks:

Definition 1.23. *Given k Λ_c-constants $H_1, \ldots, H_k$, k Λ_c^0-terms $t_1 \ldots t_k$, h Π-constants $\pi_1, \ldots, \pi_h$ and h stacks $\rho_1, \ldots, \rho_h$, consider the map ϕ from constants to Λ_c-terms and stacks defined by $(H_i \mapsto t_i)$ and $(\pi_i \mapsto \rho_i)$. The map ϕ defines a (static[2]) substitution function $\mathscr{S} : \Lambda_c \bigcup \Pi \to \Lambda_c \bigcup \Pi$ by induction.*

(1) *Substitution on terms:*

[2]This definition is independent of the reduction and defined for all terms. For this reason we say that it is a "static" substitution, by contrast with the "dynamic" substitution we will define later, which depends on reduction and is partially defined

- $\mathscr{S}(H_i) := t_i$
- $\mathscr{S}(H) := H$ *if* H *is a* Λ_c*-constant and* $H \neq H_i$ *for all* $i \in [1..k]$
- $\mathscr{S}((u_1)u_2) := (\mathscr{S}(u_1))\mathscr{S}(u_2)$
- $\mathscr{S}(\lambda x u) := \lambda x(\mathscr{S}(u))$
- $\mathscr{S}(\mathrm{cc}) := \mathrm{cc}$
- $\mathscr{S}(k_\pi) := k_{\mathscr{S}(\pi)}$

(2) *Substitution on stacks:*

- $\mathscr{S}(\pi_i) := \rho_i$
- $\mathscr{S}(\pi_0) := \pi_0$ *if* π_0 *is a stack constant and* $\pi_0 \neq \pi_i$ *for all* $i \in [1..h]$
- $\mathscr{S}(u.\pi) := \mathscr{S}(u).\mathscr{S}(\pi)$

Since this function is determined by ϕ*, this substitution can be denoted as* $[{}^{t_1 \cdots t_k, \rho_1 \cdots \rho_h}/_{H_1 \ldots H_k, \pi_1 \ldots \pi_h}]$. . *We say that* $\{H_1, \ldots, H_k, \pi_1, \ldots, \pi_k\}$ *is the* domain *of* $\mathscr{S}$ *and we denote it by* $\mathrm{dom}(\mathscr{S})$.

We now define a "join" (partial) operation on static substitutions and a partial order.

Definition 1.24. *Consider two static substitutions* $\mathscr{S}_1, \mathscr{S}_2$.

We say that $\mathscr{S}_1$ *and* $\mathscr{S}_2$ *are* compatible *if and only if they coincide over* $\mathrm{dom}(\mathscr{S}_1) \cap \mathrm{dom}(\mathscr{S}_2)$.

If $\mathscr{S}_1$ *and* $\mathscr{S}_2$ *are compatible, we denote by* $\mathscr{S}_1 + \mathscr{S}_2$ *the static substitution consisting in substituting simultaneously the constants of* $\mathrm{dom}(\mathscr{S}_1)$ *applying* $\mathscr{S}_1$ *and the constants of* $\mathrm{dom}(\mathscr{S}_2)$ *applying* $\mathscr{S}_2$. *If* $\mathscr{S}_1$ *and* $\mathscr{S}_2$ *are not compatible then* $\mathscr{S}_1 + \mathscr{S}_2$ *is not defined.*

We say that $\mathscr{S}_2$ extends $\mathscr{S}_1$ *if and only if* $\mathscr{S}_1$ *and* $\mathscr{S}_2$ *are compatible and* $\mathrm{dom}(\mathscr{S}_1) \subseteq \mathrm{dom}(\mathscr{S}_2)$. *We denote this relation as* $\mathscr{S}_2 \sqsupseteq \mathscr{S}_1$.

The (partial) operation $+$ is commutative. For this reason we adopted an additive notation. It is no difficult to prove that $\sqsupseteq$ is a transitive relation and thus it is a partial order.

Definition 1.25. *Given a reduction system, we say that it* respects substitution of constants *if and only if, for each substitution function* $\mathscr{S}$ *and for each instance* $t \star \pi \succ_1 u \star \rho$ *of a reduction rule, we have that* $\mathscr{S}(t) \star \mathscr{S}(\pi) \succ_1 \mathscr{S}(u) \star \mathscr{S}(\rho)$.

Typically, a reduction rule as the "signature" used in [6], does not respect the substitution. The reason is that this instruction uses the code of the stack to compute an integer. When we perform the substitution, the code of the stack will be changed and then the calculated integer also.

However, in the system necessary to work in classical arithmetics, the substitution respects reduction:

Lemma 1.26. *Given two processes* $\tau \star \pi$ *and* $\sigma \star \rho$ *satisfying* $\tau \star \pi \succ_1 \sigma \star \rho$ *and a substitution* $\mathscr{S}$ *defined as in 1.23, then* $\mathscr{S}(\tau) \star \mathscr{S}(\pi) \succ_1 \mathscr{S}(\sigma) \star \mathscr{S}(\rho)$. *Moreover, this reduction uses the same rules as the reduction* $\tau \star \pi \succ_1 \sigma \star \rho$.

Proof.

- For the (push) and (pop) rules, it is straightforward.
- For a given stack $t.\pi$, we have that $\mathrm{cc}\star t.\pi \succ_1 t \star k_\pi \pi$. We apply the substitution function $\mathscr{S}$ before the reduction and we obtain: $\mathscr{S}(\mathrm{cc}) \star \mathscr{S}(t.\pi) = \mathrm{cc}\star\mathscr{S}(t).\mathscr{S}(\pi)$. But this process reduces to $\mathscr{S}(t) \star k_{\mathscr{S}(\pi)}.\mathscr{S}(\pi) = \mathscr{S}(t) \star \mathscr{S}(k_\pi.\pi)$.
- For a given stack $t.\pi'$, we have that $k_\pi \star t.\pi' \succ_1 t \star \pi$. We apply the substitution function $\mathscr{S}$ before the reduction and we obtain: $\mathscr{S}(k_\pi) \star \mathscr{S}(t.\pi') = k_{\mathscr{S}(\pi)} \star \mathscr{S}(t).\mathscr{S}(\pi')$. But this process reduces to $\mathscr{S}(t) \star \mathscr{S}(\pi)$.

$\square$

2. **Semantics**

2.1. **Tarski models.**

We expose in this subsection the basis of Tarski models. Our presentation is adapted to the case of a second order language, but the basic ideas are the same the reader can found in classical books of model theory.

As it was defined, the language $\mathcal{L}$ has only function symbols as non logical symbols. A $\mathcal{L}$-structure is a triplet $\mathfrak{M} = (M, \mathcal{D}, \mathcal{Y})$ such that:

- M is a set.
- $\mathcal{D}$ is a set of predicates in M, i.e.: $\mathcal{D} \subseteq \bigcup_{k\in\mathbb{N}} M^k$. $\mathcal{D}$ must contain at least one predicate for each arity $k \in \mathbb{N}$.
- $\mathcal{Y}$ is a map $f \mapsto f^{\mathcal{Y}}$ from function symbols, such that for each k-ary symbol f, we have $f^{\mathcal{Y}} : M^k \to M$.

The set M is called *the ground set*, $\mathcal{D}$ *the second order domain* and $\mathcal{Y}$ *the interpretation* of the structure.

For instance, in arithmetics, we use a 1-ary function symbol s, interpreted as the successor function, two binary function symbols $+$, $\times$ interpreted as the addition and the multiplication respectively and a 0-ary function symbol 0, interpreted as the zero.

Definition 2.1. *Given a $\mathcal{L}$-structure $\mathfrak{M} = (M, \mathcal{D}, \mathcal{Y})$, an assignment $\mathscr{A}$ is a map $x \mapsto x^{\mathscr{A}}$ from first order variables to M, together with a map $X \mapsto X^{\mathscr{A}}$ from second order variables to $\mathcal{D}$ such that $X^{\mathscr{A}}$ is a k-ary predicate provided that X is a k-ary second order variable.*

A parametrical $\mathcal{L}$-term is a pair $(\tau, \mathscr{A})$, where τ is a $\mathcal{L}$-term and $\mathscr{A}$ is an assignment. In analogous way, a parametrical $\mathcal{L}$-formula is a pair $(\varphi, \mathscr{A})$, where φ is a $\mathcal{L}$-formula and $\mathscr{A}$ is an assignment.

Example 2.2. Let us consider a model of ZF and the ordinal ω with the usual definitions of zero, successor, addition and multiplication. We will call ω-structure of $\mathcal{L}$ each structure $(\omega, \mathcal{D}, \mathcal{Y})$ where $\mathcal{Y}$ interprets the function symbols $0, \mathrm{s}, +, \times$ as the zero, successor, addition and multiplication defined in ω. We will denote also ω as $\mathbb{N}$.

Notation 2.3. Consider an assignment $\mathscr{A}$ over a $\mathcal{L}$-structure $(M, \mathcal{D}, \mathcal{Y})$, h first order variables $x_1, \ldots, x_h$, k second order variables $X_1, \ldots, X_k$, h individuals $m_1, \ldots, m_h \in M$ and k predicates $D_1, \ldots, D_k \in \mathcal{D}$ such that D_i has the same arity that X_i for all $i \in [1..k]$. We denote as

$$\mathscr{A}\left[{}^{m_1, \ldots, m_h, D_1, \ldots, D_k}\!/_{x_1, \ldots, x_h, X_1, \ldots, X_k}\right]$$

the assignment which coincide with $\mathscr{A}$ on all variables except on the variables $x_1, \ldots, x_h, X_1, \ldots, X_k$; on which it takes respectively the values $m_1, \ldots, m_h, D_1, \ldots, D_k$.

Definition 2.4. *Given a $\mathcal{L}$-structure $\mathfrak{M} = (M, \mathcal{D}, \mathcal{Y})$ and an assignment $\mathscr{A}$, we define by induction the interpretation of the parametrical terms $(\tau, \mathscr{A})$. This one will be an element of M denoted as $\tau^{\mathfrak{M},\mathscr{A}}$:*

- $x^{\mathfrak{M},\mathscr{A}} := x^{\mathscr{A}}$.

- $(f\tau_1,\ldots,\tau_k)^{\mathfrak{M},\mathscr{A}} := f^{\mathscr{Y}}(\tau_1^{\mathfrak{M},\mathscr{A}},\ldots,\tau_k^{\mathfrak{M},\mathscr{A}})$.

Now, we can define satisfaction in $\mathcal{L}$-structures:

Definition 2.5. *Let us consider a $\mathcal{L}$-structure $\mathfrak{M} = (M,\mathcal{D},\mathscr{Y})$. For each parametrical formula $(\varphi,\mathscr{A})$, we define the satisfaction of $(\varphi,\mathscr{A})$ in $\mathfrak{M}$ by induction on φ. We denote the satisfaction of $(\varphi,\mathscr{A})$ in $\mathfrak{M}$ as $\mathfrak{M}\models(\varphi,\mathscr{A})$.*

- *$\mathfrak{M} \models (X\tau_1\ldots\tau_k,\mathscr{A})$ if and only if $(\tau_1^{\mathfrak{M},\mathscr{A}},\ldots,\tau_k^{\mathfrak{M},\mathscr{A}}) \in X^{\mathscr{A}}$.*
- *$\mathfrak{M} \models (\varphi\to\psi,\mathscr{A})$ if and only if $\mathfrak{M}\models(\psi,\mathscr{A})$ provided that $\mathfrak{M}\models(\psi,\mathscr{A})$.*
- *$\mathfrak{M} \models \forall x\varphi$ if and only if for each $m \in M$, $\mathfrak{M} \models (\varphi,\mathscr{A}[{}^{m}/_{x}])$.*
- *$\mathfrak{M} \models \forall X\varphi$ if and only if for all $D \in \mathcal{D}$, $\mathfrak{M} \models (\varphi,\mathscr{A}[{}^{D}/_{X}])$.*

Example 2.6. Informally, one could say that parametrical formulæ allows us to write properties about individuals and their relations. Consider an assignment $\mathscr{A}$ such that $x^{\mathscr{Y}}$ is the standard integer 4 and $y^{\mathscr{Y}}$ is the standard integer 7 and the formula $x=y$ as it is defined in 1.12. For instance, suppose that $\mathcal{D} = \bigcup_{k\in\mathbb{N}}\mathcal{P}ow(M^k)$. Then, $\mathfrak{M} \models (x=y,\mathscr{Y},\mathscr{A})$ if and only if $x^{\mathscr{Y}}$ and $y^{\mathscr{Y}}$ are the same element of M, which is false in our example.

Since $x=y$ contains only the free variables x and y, the interpretation and the assignment of all the other variables that are not x nor y is not relevant for the truth of $x=y$ in a given model. Our semantics definition reflects this property and hence we can write only $4=7$ instead of $(x=y,\mathscr{Y},\mathscr{A})$, thus indicating that the meaning of our formula depends only on the assignment of two different variables, one assigned as 4 and the other as 7.

More generally, the satisfaction of a formula in a $\mathcal{L}$-structure depends only on the assignment of their free variables. Using this fact, we can simplify notations, thus eliminating all references to the assignment $\mathscr{A}$. Given a formula, the values assigned to their free variables, are called the *parameters* of the formula.

Notation 2.7. In general, we will denote the parametrical term

$$(\tau,\mathscr{A}[{}^{m_1,\ldots,m_h}/_{x_1,\ldots,x_h}]) \text{ as } (\tau[{}^{m_1,\ldots,m_h}/_{x_1,\ldots,x_h}],\mathscr{A})$$

and the parametrical formula

$$(\varphi,\mathscr{A}[{}^{m_1,\ldots,m_h,D_1,\ldots,D_k}/_{x_1,\ldots,x_h,X_1,\ldots,X_k}])$$

as

$$(\varphi[{}^{m_1,\ldots,m_h,D_1,\ldots,D_k}/_{x_1,\ldots,x_h,X_1,\ldots,X_k}],\mathscr{A})$$

In particular, if $\mathscr{A}$ is well known, or not relevant for satisfaction, we can write simply:

$$\tau[{}^{m_1,\ldots,m_h}/_{x_1,\ldots,x_h}] \text{ and } \varphi[{}^{m_1,\ldots,m_h,D_1,\ldots,D_k}/_{x_1,\ldots,x_h,X_1,\ldots,X_k}]$$

respectively, thus omitting all references to $\mathscr{A}$.

Finally, if first we declare φ=$\varphi(x_1,\ldots,x_h,X_1,\ldots,X_k)$ (i.e. φ depends only upon $x_1,\ldots,x_h,X_1,\ldots,X_k$), we can denote the parametrical formula

$$\varphi[{}^{m_1,\ldots,m_h,D_1,\ldots,D_k}/_{x_1,\ldots,x_h,X_1,\ldots,X_k}]$$

as

$$\varphi(m_1,\dots,m_h,D_1,\dots,D_k)$$

In particular, if φ is a closed formula, φ represents itself all parametrical formulæ $(\varphi,\mathscr{A})$ and its satisfaction can be written as $\mathfrak{M}\models\varphi$. For parametrical terms we introduce the same notation, writing $\tau[m_1,\dots,m_h]$ instead of $\tau[{}^{m_1,\dots,m_h}/_{x_1,\dots,x_h}]$.

Definition 2.8. *The* comprehension schemata *is the axioms schemata*

$$CS_\psi \quad := \exists X\forall x_1,\dots,x_k(Xx_1\dots x_k \leftrightarrow \psi(x_1,\dots,x_k))$$

which gives one axiom for each formula $\psi(x_1,\dots,x_k)$

A $\mathcal{L}$-structure $\mathfrak{M}=(M,\mathcal{D},\mathscr{Y})$ is a model of second order logic with comprehension schemata (later called simply "a model") *if and only if $\mathfrak{M}\models CS_\psi$ for each $\mathcal{L}$-formula ψ.*

Comprehension schemata says that each formula $\psi(x_1,\dots,x_k)$ determines a second order predicate containing exactly all k-uplets $(m_1,\dots,m_k)$ of individuals satisfying the parametrical formula $\psi(m_1,\dots,m_k)$. Notice that in $\mathcal{L}$ we can not write the Russell's Paradox because we have no equivalent in $\mathcal{L}$ to the atomic formula $x\notin x$, which we can write in ZF language. In consequence there is no problem to assert that each formula defines a set.

Definition 2.9. *The axioms of Peano arithmetic can be given in the following way: take for $\mathcal{L}$ four function symbols,* s *for the successor function,* $+$ *for the addition,* $\times$ *for the product and* 0 *for the integer zero. The axioms written in $\mathcal{L}$ are the following:*

PA.1 $\forall x\neg(\mathrm{s}x=0)$
PA.2 $\forall x\forall y(sx=sy\rightarrow x=y)$
PA.3 $\forall x(x+0=0)$
PA.4 $\forall x\forall y(x+\mathrm{s}y=\mathrm{s}(x+y))$
PA.5 $\forall x(x\times 0=0)$
PA.6 $\forall x\forall y(x\times\mathrm{s}y=(x\times y)+x)$
PA.7 $\forall x\,\mathrm{int}(x)$

The axiom PA.1 *means that* 0 *is not a successor of any individual, the axiom* PA.2 *means that the successor is an injective function. The axiom* PA.7 *is the recursion principle. A $\mathcal{L}$-structure is a model of Peano arithmetic if and only if it satisfies the axioms* PA.1 *to* PA.7.

Example 2.10. A ω-structure satisfying PA.1 to PA.7 will be called a ω-model of Peano second order arithmetics. A particular case is the so called *standard model* $\mathfrak{N}:=(\mathbb{N},\bigcup_{k\in\mathbb{N}}\mathcal{P}ow(\mathbb{N}^k),\mathscr{Y})$

The reader should be noticed that they are two different ways to define Tarski models in second order logic: The first one is the so called *full* models, that forces the domain $\mathcal{D}$ to be equals to $\bigcup_{k\in\mathbb{N}}\mathcal{P}ow(M^k)$ where M is the ground set. The second one is the definition used in this work and it demands only that $\mathcal{D}$ contains the k-ary subsets of M which are defined by some formula (to ensure the comprehension schemata). While the full models have not a completeness theorem, the second definition admits completeness:

Theorem 2.11. *Given a consistent set S of $\mathcal{L}$-forumlæ, there is a Tarski model $\mathfrak{M}$ satisfying all formulæ belonging to S.*

Proof. (Sketch) The most simply procedure to prove this result is to code the second order language on a first order language with $\in$ and to apply the completeness of first order languages. Other possibility is to adapt the proof of completeness for first order languages that uses Henkin witness. $\square$

2.2. **Boolean-valued models.**

Once we define Tarski models, we assign to each k-ary second order variable a k-ary predicate of the ground set M. Let us consider an assignment $X^{\mathscr{A}}$ of a k-ary second order variable X. We can consider $X^{\mathscr{A}}$ as its characteristic function $X^{\mathscr{A}} : M^k \to \{0,1\}$ from M^k to the boolean algebra $\{0,1\}$, where the sup $\vee$, the inf $\wedge$ and the complement $\neg$ are defined as usual:

$\vee$	0	1
0	0	1
1	1	1

$\wedge$	0	1
0	0	0
1	0	1

	0	1
$\neg$	1	0

By means of the boolean algebra operations, we can define the *truth value* of $\mathcal{L}$-formulæ, which we defined in 2.1. Let us consider an assignment $\mathscr{A}$ and a $\mathcal{L}$-structure $\mathfrak{M} = (M, \mathcal{D}, \mathscr{Y})$. For simplicity, we denote as $||\varphi||$ the truth value of a parametrical formula $(\varphi, \mathscr{A})$:

- $||X\tau_1 \dots \tau_k|| := X^{\mathscr{A}}(\tau_1^{\mathfrak{M},\mathscr{A}}, \dots, \tau_k^{\mathfrak{M}\mathscr{A}})$
- $||\psi_1 \to \psi_2|| := \neg||\psi_1|| \vee ||\psi_2||$
- $||\forall x\psi|| := \wedge_{m\in M}||\psi(m)||$
- $||\forall X\psi|| := \wedge_{D\in\mathcal{D}}||\psi(D)||$

This definition gives the parametrical formulæ truth values according to the Tarski model $\mathfrak{M}$.

The reader should be noticed that universal quantification is treated as an infimum of a set of truth values. A boolean algebra $\mathcal{B}$ such that for each set $B \subseteq \mathcal{B}$, there exists $\inf(B)$ is called a *complete boolean algebra*. Naturally, a finite boolean algebra (as $\{0,1\}$ is), is a complete boolean algebra, but in general an infinite boolean algebra does not need to be complete.

We can generalise the semantics defined above taking any complete boolean algebra $\mathcal{B}$ instead of $\{0,1\}$. It suffices to assign each second order variable X as a function $X^{\mathscr{A}} : M^k \to \mathcal{B}$. The models thus defined are called *boolean-valued models*. In consequence, boolean-valued models are a generalisation of Tarski models. The reader can see [1] for more information about these models.

2.3. **Realizability models.**

In this subsection we define a semantics for the language $\mathcal{P}$ (i.e. the language $\mathcal{L}$ enriched with the implication $\rightsquigarrow$ and the Λ_c-set variables. c.f.:1.8).

The truth values of formulæ will be defined as sets of stacks. In order to define realizability, we start with a Tarski model $\mathfrak{M}=(M,\mathcal{D},\mathscr{Y})$ of PA. This model $\mathfrak{M}$ will be called the *start model* or *ground model* and in most cases will be chosen as the standard model of arithmetics, i.e. the language has only $\mathrm{s},0,+,\times$ as function symbols, $(M,\mathcal{D},\mathscr{Y})=(\mathbb{N},\bigcup_{k\in\mathbb{N}}\mathcal{P}ow(\mathbb{N}^k),\mathscr{Y})$, $\mathrm{s}^{\mathscr{Y}}$ is the successor function, $0^{\mathscr{Y}}$ is the natural 0 and $+^{\mathscr{Y}},\times^{\mathscr{Y}}$ are respectively the usual addition and multiplication.

The set of truth values $\mathscr{T}$ is the set $\mathcal{P}ow(\Pi)$, to which we will add a structure of boolean algebra[3]. The ground set is M and the interpretation is $\mathscr{Y}$, on the other hand, the assignment domain for second order variables is the set of all functions belonging to $\cup_{k\in\mathbb{N}}\mathscr{T}^{M^k}$. More precisely: an *assignment* is a map $x\mapsto x^{\mathscr{A}}$ from first order variables together with a map $X\mapsto X^{\mathscr{A}}$ from second order variables and a map $\varepsilon\mapsto\varepsilon^{\mathscr{Y}}$ from Λ_c-set variables; all three satisfying:

- For a first order variable x, its assignment $x^{\mathscr{A}}$ belongs to M.
- For a k-ary second order variable X, its assignment $X^{\mathscr{A}}$ is a function of domain M^k and codomain $\mathscr{T}$.
- For a k-ary Λ_c-set variable ε, its assignment $\varepsilon^{\mathscr{A}}$ is a function of domain M^k and codomain $\mathcal{P}ow(\Lambda_c)$.

Since the ground set M and the interpretation $\mathscr{Y}$ are the same that for the start model $\mathfrak{M}$, the terms of the language are interpreted as in $\mathfrak{M}$.

In order to define the *truth value* of a parametrical extended formula, it is required to explain how we interpret the arrow $\to$. To do this, we make the following definitions:

Definition 2.12.

(1) *A set of processes S is called* saturated *if and only if it is closed by antireduction. More explicitly, if Q is a process belonging to S and $P\succ Q$; then P belongs to S.*
(2) *Given a saturated set $\perp\!\!\!\perp$, the* orthogonal complement *map is defined by $()^{\perp\!\!\!\perp}:\mathcal{P}(\Pi)\to\mathcal{P}(\Lambda_c)$, $\mathbb{W}^{\perp\!\!\!\perp}:=\{t\in\Lambda_c\mid\forall\pi\in\mathbb{W}\ t\star\pi\in\perp\!\!\!\perp\}$. We denote by $t\Vdash_{\perp\!\!\!\perp}\mathbb{V}$ the statement $t\in\mathbb{V}^{\perp\!\!\!\perp}$.*

Remark 2.13. The orthogonal complement is a contravariant function (according to inclusion): Indeed, if we take $\mathbb{P}\subseteq\mathbb{Q}$, we have $\mathbb{P}^{\perp\!\!\!\perp}\supseteq\mathbb{Q}^{\perp\!\!\!\perp}$

We define some notations that will be in force thereafter.

Notation 2.14. Given two sets $\mathscr{V}\subseteq\Lambda_c$ and $\mathbb{W}\subseteq\Pi$, we denote by $\mathscr{V}\rightsquigarrow\mathbb{W}$ the set $\{t.\pi\mid t\in\mathscr{V}$ and $\pi\in\mathbb{W}\}$. Similarly, if $\mathbb{V},\mathbb{W}\subseteq\Pi$, we denote by $\mathbb{V}\to\mathbb{W}$ the set $\mathbb{V}^{\perp\!\!\!\perp}\rightsquigarrow\mathbb{W}$, i.e. $\{t.\pi\mid t\in\mathbb{V}^{\perp\!\!\!\perp}$ and $\pi\in\mathbb{W}\}$.

Remark 2.15. The operator $\rightsquigarrow$ is covariant on both arguments, but the operator $\to$ is covariant on the right argument and contravariant on the left argument:

[3] $\mathcal{P}ow(\Pi)$ has a boolean algebra structure with $\bigcup,\bigcap$ and complement, but this boolean algebra is not interesting for our purposes.

Consider $\mathbb{P}_1, \mathbb{P}_2, \mathbb{Q}_1, \mathbb{Q}_2 \subseteq \Pi$ and $\mathscr{V}_1, \mathscr{V}_2 \subseteq \Lambda_c$ such that $\mathbb{P}_1 \subseteq \mathbb{P}_2$, $\mathbb{Q}_1 \subseteq \mathbb{Q}_2$ and $\mathscr{V}_1 \subseteq \mathscr{V}_2$. By definition, $\mathscr{V}_1 \rightsquigarrow \mathbb{Q}_1 \subseteq \mathscr{V}_2 \rightsquigarrow \mathbb{Q}_2$.

Using 2.13 $\mathbb{P}_2^{\perp\!\perp} \subseteq \mathbb{P}_1^{\perp\!\perp}$ and hence $\mathbb{P}_2 \to \mathbb{Q}_1 = \mathbb{P}_2^{\perp\!\perp} \rightsquigarrow \mathbb{Q}_1 \subseteq \mathbb{P}_1^{\perp\!\perp} \rightsquigarrow \mathbb{Q}_2 = \mathbb{P}_1 \to \mathbb{Q}_2$, thus proving the remark.

Notation 2.16. In most cases, as for the Tarski models, the interpretation $\mathscr{Y}$ of function symbols is well known. Then, to simplify notation, we will omit all references to $\mathscr{Y}$. For the assignments on realizability models, we make the same denotational conventions we made in 2.3

Definition 2.17. *Let us consider a $\mathcal{L}$-structure $\mathfrak{M} = (M, \mathcal{D}, \mathscr{Y})$ and a saturated set of processes $\perp\!\!\perp$. We define for each parametrical extended formula $(\varphi, \mathscr{A})$ its* truth value relative to $\perp\!\!\perp$, *which we denote by $||(\varphi, \mathscr{A})||_{\perp\!\perp}$. The definition is made simultaneously for each assignment $\mathscr{A}$ and by induction on the formula φ:*

- $||(\top, \mathscr{A})||_{\perp\!\perp} := \emptyset$
- $||(X\tau_1 \ldots \tau_k, \mathscr{A})||_{\perp\!\perp} := X^{\mathscr{A}}(\tau_1^{\mathscr{Y}\mathscr{A}}, \ldots, \tau_k^{\mathscr{Y}\mathscr{A}})$
- $||(\varepsilon\tau_1 \ldots \tau_k \rightsquigarrow \varphi, \mathscr{A})||_{\perp\!\perp} := \varepsilon^{\mathscr{A}}(\tau_1^{\mathscr{Y}\mathscr{A}}, \ldots, \tau_k^{\mathscr{Y}\mathscr{A}}) \rightsquigarrow ||(\varphi, \mathscr{A})||_{\perp\!\perp}$
- $||(\varphi \to \psi, \mathscr{A})||_{\perp\!\perp} := ||(\varphi, \mathscr{A})||_{\perp\!\perp} \to ||(\psi, \mathscr{A})||_{\perp\!\perp}$
- $||(\forall x \varphi(x), \mathscr{A})||_{\perp\!\perp} := \bigcup_{n \in \mathbb{N}} ||(\varphi[^n/_x], \mathscr{A})||_{\perp\!\perp}$
- $||(\forall X \varphi(X), \mathscr{A})||_{\perp\!\perp} := \bigcup_{R \in \mathscr{T}^{\mathbb{N}^k}} ||(\varphi[^R/_X], \mathscr{A})||_{\perp\!\perp}$

For each saturated set $\perp\!\!\perp$, we have an equivalence relation $\sim_{\perp\!\perp}$ between parametrical extended formulæ such that $(\varphi, \mathscr{A}) \sim_{\perp\!\perp} (\psi, \mathscr{A})$ if and only if their truth values are equals. We say that two parametrical extended formulæ are *truth equivalents* if and only if they are equivalent for all $\perp\!\!\perp$.

Remark 2.18. Given a saturated set $\perp\!\!\perp$, the interpretation of a term depends only on the interpretation of their function symbols and the assignment of their variables. The truth value of an extended formula depends only on the interpretation of their function symbols and the assignment of their *free variables*.

Notation 2.19. We make similar denotational conventions about parameters that we made on 2.7.

Indeed, suppose that $\varphi = \varphi(x_1, \ldots, x_p, X_1, \ldots, X_q)$. Since the truth value of φ is determined by assignement of their free variables (c.f. 2.18), the assignement $\mathscr{A}$ on other variables is not relevant. In consequence, we can adopt the notation:

$$\varphi[^{a_1, \ldots, a_p, R_1, \ldots, R_q}/_{x_1, \ldots, x_p, X_1, \ldots, X_q}]$$

or

$$\varphi[a_1, \ldots, a_p, R_1, \ldots, R_q]$$

instead of

$$(\varphi, \mathscr{A}[^{a_1, \ldots, a_p, R_1, \ldots, R_q}/_{x_1, \ldots, x_p, X_1, \ldots, X_q}])$$

A similar notation will be used for parametrical terms. A few times, we do not want to explicite parameters, but we will also avoid the notation $(\varphi, \mathscr{A})$, preferring to write $\varphi^{\mathscr{A}}$.

In particular, if φ is a closed formula, φ represents itself all parametrical formulæ $\varphi^{\mathscr{A}}$ and its truth value can be written as $||\varphi||$.

The reader may not appreciate to introduce a lot of notations here. However, it allows us to choose the shortest possible notation without lost of precision, depending of the expression we want write.

Notation 2.20. We use indistinctly the notations $t \Vdash_{\perp\!\!\perp} \varphi[\vec{a},\vec{R}]$ (read t ***realizes*** $\varphi[\vec{a},\vec{R}]$ ***in*** $\perp\!\!\perp$) and $t \in |\varphi[\vec{a},\vec{R}]|_{\perp\!\!\perp}$ as synonyms of $t \in (||\varphi[\vec{a},\vec{R}]||_{\perp\!\!\perp})^{\perp\!\!\perp}$. Every time the set $\perp\!\!\perp$ is well known, we will omit the subscript reference to $\perp\!\!\perp$ in these notations.

The notation $t \Vvdash \varphi(\vec{a},\vec{R})$ means that for each saturated set $\perp\!\!\perp$, we have $t \Vdash_{\perp\!\!\perp} \varphi(\vec{a},\vec{R})$. A Λ_c-term t such that $t \Vvdash \varphi(\vec{a},\vec{R})$ is called a *universal realizer* of $\varphi(\vec{a},\vec{R})$.

As we will see in subsection 2.5, in most interesting cases we have Tarski models associated to the choice of a saturated set $\perp\!\!\perp$. By extension, we will say that saturated sets of processes are *models*.

Lemma 2.21. *Consider a Tarski model $\mathfrak{M} = (M,\mathcal{D},\mathscr{A})$, an extended formula φ and a a-ary second order variable X that appears only in positive (resp. negative) position on φ. Take $\mathbb{P},\mathbb{Q}$ two ($\mathscr{T}$-valued) a-ary predicates such that $\forall^M x_1 \dots x_a\ (\mathbb{P}(\vec{x}) \subseteq \mathbb{Q}(\vec{x}))$.*

Then, for each model $\perp\!\!\perp$, $||\varphi[{}^{\mathbb{P}}/_X]||$ is included (resp. includes) $||\varphi[{}^{\mathbb{Q}}/_X]||$.

Proof. If X in not a free variable of φ, then $||\varphi[{}^{\mathbb{P}}/_X]|| = ||\varphi[{}^{\mathbb{Q}}/_X]||$.

Suppose that X is free in φ. The proof is by induction on the formula φ.

- If φ is atomic, $\varphi = X\vec{\tau}$ for some terms $\vec{\tau}$. Then, $||\varphi[{}^{\mathbb{P}}/_X]|| = \mathbb{P}(\vec{\tau}) \subseteq \mathbb{Q}(\vec{\tau}) = ||\varphi[{}^{\mathbb{Q}}/_X]||$.
- If φ is $\varepsilon\vec{\tau} \rightsquigarrow \psi$ and X is in positive (resp. negative) position, then X is also in positive (resp. negative) position in ψ. By induction hypothesis $||\psi[{}^{\mathbb{P}}/_X]||$ is a subset (resp. includes) $||\psi[{}^{\mathbb{Q}}/_X]||$. By definition of truth values and 2.15, we have:

$$||\varphi[{}^{\mathbb{P}}/_X]|| = \varepsilon\vec{\tau} \rightsquigarrow ||\psi[{}^{\mathbb{P}}/_X]|| \subseteq \text{ (resp. } \supseteq) \ \varepsilon\vec{\tau} \rightsquigarrow ||\psi[{}^{\mathbb{Q}}/_X]|| = ||\varphi[{}^{\mathbb{Q}}/_X]||$$

- If φ is $\psi \to \chi$, then X must be in positive (resp. negative) position in χ and in negative (resp. positive) position in ψ. By induction hypothesis, we have that $||\chi[{}^{\mathbb{P}}/_X]||$ is a subset of (resp. includes) $||\chi[{}^{\mathbb{Q}}/_X]||$ and $||\psi[{}^{\mathbb{P}}/_X]||$ includes (resp. is a subset of) $||\chi[{}^{\mathbb{Q}}/_X]||$. By definition of truth values and 2.15, we have:

$$||\varphi[{}^{\mathbb{P}}/_X]|| = |\psi[{}^{\mathbb{P}}/_X]| \rightsquigarrow ||\chi[{}^{\mathbb{P}}/_X]|| \subseteq \text{ (resp. } \supseteq) \ |\psi[{}^{\mathbb{Q}}/_X]| \rightsquigarrow ||\chi[{}^{\mathbb{Q}}/_X]||$$

- For first and second order quantification is a direct verification.

□

The following lemma explains a relation between the second order substitution and second order parameters that justifies the similarity between the two notations we adopted:

Lemma 2.22. *Consider an extended formula χ, p first order variables $x_1,\dots,x_p$ and a p-ary second order variable X. Take a Tarski model $\mathfrak{M}=(M,\mathcal{D},\mathcal{Y})$ and a realizability model $\perp\!\!\!\perp$. Define a p-ary predicate $\mathbb{P}{:}M^p\to\mathcal{T}$ such that*

$$\mathbb{P}(a_1,\dots,a_p) := ||\chi[{}^{a_1,\cdots,a_p}/_{x_1,\dots,x_p}]||$$

Then, for each parametrical formula φ, we have $||\varphi[{}^{\chi}/_{Xx_1\dots x_p}]||{=}||\varphi[{}^{\mathbb{P}}/_{X}]||$.

Proof. If $X\notin \mathrm{FV}(\varphi)$, then $||\varphi[{}^{\chi}/_{Xx_1,\dots,x_p}]||=||\varphi||=||\varphi[{}^{\mathbb{P}}/_{X}]||$.

If $X\in \mathrm{FV}(\varphi)$, we prove the result by induction in φ, simultaneously for all assignment $\mathcal{A}$:

- Suppose that $\varphi = X\tau_1\dots\tau_p$. Then, $||\varphi[{}^{\chi}/_{Xx_1\dots x_p}]|| = ||\chi[{}^{\tau_1,\cdots,\tau_p}/_{x_1,\dots,x_p}]|| = \mathbb{P}(\tau_1,\dots,\tau_p) = ||X\tau_1\dots\tau_p[{}^{\mathbb{P}}/_{X}]|| = ||\varphi[{}^{\mathbb{P}}/_{X}]||$
- Suppose that $\varphi=\varepsilon\vec{\tau}\rightsquigarrow\psi$. Then, $||(\varepsilon\vec{\tau}\rightsquigarrow\psi)[{}^{\chi}/_{Xx_1\dots x_p}]|| = \varepsilon^{\mathcal{A}}(\vec{\tau}^{\mathcal{A}}){\rightsquigarrow}||\psi[{}^{\chi}/_{Xx_1\dots x_p}]|| =^{*} \varepsilon^{\mathcal{A}}(\vec{\tau}^{\mathcal{A}})\rightsquigarrow||\psi[{}^{\mathbb{P}}/_{X}]|| = ||(\varepsilon(\vec{\tau}^{\mathcal{A}})\rightsquigarrow\psi)[{}^{\mathbb{P}}/_{X}]||$.
- Suppose that $\varphi=\psi_1\to\psi_2$. Similar to the case above.
- Suppose that $\varphi=\forall x\psi$. Then, $||(\forall x\psi)[{}^{\chi}/_{Xx_1\dots x_p}]|| = \bigcup_{m\in M}||\psi[{}^{\chi}/_{Xx_1\dots x_p}][{}^{m}/_{x}]|| =^{*} \bigcup_{m\in M}||\psi[{}^{m,\mathbb{P}}/_{x,X}]|| = ||(\forall x\psi)[{}^{\mathbb{P}}/_{X}]||$.
- Suppose that $\varphi=\forall Y\psi$. Then, $||(\forall Y\psi)[{}^{\chi}/_{Xx_1\dots x_p}]|| = \bigcup_{\mathbb{Q}\in\mathcal{T}^{M^p}}||\psi[{}^{\chi}/_{Xx_1\dots x_p}][{}^{\mathbb{Q}}/_{Y}]|| =^{*} \bigcup_{\mathbb{Q}\in\mathcal{T}^{M^p}}||\psi[{}^{\mathbb{P},\mathbb{Q}}/_{X,Y}]||{=} ||(\forall Y\psi)[{}^{\mathbb{P}}/_{X}]||$.

The steps in the proof where we use the induction hypothesis are signalled with a star. In particular it might be worth noticing that in the last two places marked with a star, we use the fact that the induction hypothesis holds for *all* assignment $\mathcal{A}$. □

Definition 2.23. *Let us consider a Tarski model $\mathfrak{M}=(M,\mathcal{D},\mathcal{Y})$ and a realizability model $\perp\!\!\!\perp$. A sequent*

$$x_1:\varphi_1,\dots,x_p:\varphi_p,y_1\in\varepsilon_1\vec{\tau}_1,\dots,y_q\in\varepsilon_q\vec{\tau}_q\vdash t:\psi$$

is called a true *sequent in $\perp\!\!\!\perp$ if and only if, for each assignment $\mathcal{A}$ and for all terms $t_1,\dots,t_p,u_1,\dots,u_q$ such that $t_i\Vdash\varphi_i^{\mathcal{A}}$ and $u_j{\in}\mathcal{E}^{\mathcal{A}}(\vec{\tau}_j^{\mathcal{A}})$ for all i and j; we have that $t[{}^{t_1,\cdots,t_p,u_1,\cdots,u_q}/_{x_1,\dots,x_p,y_1,\dots,y_q}]\Vdash\psi^{\mathcal{A}}$. A sequent*

$$x_1:\varphi_1,\dots,x_p:\varphi_p,y_1\in\varepsilon_1\vec{\tau}_1,\dots,y_q\in\varepsilon_q\vec{\tau}_q\vdash y{\in}\varepsilon\vec{\tau}$$

is called a true *sequent in $\perp\!\!\!\perp$ if and only if, for each assignment $\mathcal{A}$ and for all terms $t_1,\dots,t_p,u_1,\dots,u_q$ such that $t_i\Vdash\varphi_i^{\mathcal{A}}$ and $u_j{\in}\mathcal{E}^{\mathcal{A}}(\vec{\tau}_j^{\mathcal{A}})$ for all i and j; then $y[{}^{t_1,\cdots,t_p,u_1,\cdots,u_q}/_{x_1,\dots,x_p,y_1,\dots,y_q}]{\in}\varepsilon^{\mathcal{A}}(\vec{\tau}^{\mathcal{A}})$.*

A deduction rule $\dfrac{\Gamma_1,\dots,\Gamma_k}{\Delta}$ (R) is correct *if and only if, for each model $\perp\!\!\!\perp$, if $\Gamma_1,\dots,\Gamma_k$ are true in $\perp\!\!\!\perp$; then Δ is also true in $\perp\!\!\!\perp$.*

2.4. **The soundness lemma.**

Traditionally, the *soundness lemma* guarantees that truth is preserved by deduction. In realizability models we have that the formulation below has the same meaning:

Lemma 2.24.

(1) *All proof-rules defined in 1.17 are correct.*
(2) *Consider a Tarski model $\mathfrak{M} = (M, \mathcal{D}, \mathcal{Y})$ and a realizability model $\perp\!\!\!\perp$. Consider a provable sequent*

$$x_1{:}\varphi_1,\dots,x_p{:}\varphi_p,y_1{\in}\mathcal{E}_1\vec{\tau}_1,\dots,y_q{\in}\mathcal{E}\vec{\tau}_q \vdash t{:}\psi$$

and an assignment $\mathscr{A}$. Choose $t_i \Vdash \varphi_i^{\mathscr{A}}$ and $u_j{\in}\mathcal{E}_j^{\mathscr{A}}\vec{\tau}_j^{\mathscr{A}}$ for all i and j. Then:

$$t\left[{}^{t_1,\dots,t_p,u_1,\dots,u_q}\big/{}_{x_1,\dots,x_p,y_1,\dots,y_q}\right] \Vdash \psi^{\mathscr{A}}$$

(3) *Consider a provable sequent $\vdash t{:}\psi$. Then, for each model $\perp\!\!\!\perp$ and for each assignment $\mathscr{A}$, we have that $t \Vdash \psi^{\mathscr{A}}$.*

Proof. We prove the statement (1). The second statement is a display of correctness definition while the third statement is a particularisation of the second one. For simplicity we denote as Γ the hypothesis

$$x_1 : \varphi_1,\dots,x_p : \varphi_p, y_1 \in \mathcal{E}_1\vec{\tau}_1,\dots,y_q \in \mathcal{E}_q\vec{\tau}_q$$

Consider a Tarski model $\mathfrak{M} = (M, \mathcal{D}, \mathcal{Y})$ and a realizability model $\perp\!\!\!\perp$. Take an assignment $\mathscr{A}$ and $t_1,\dots,t_p,u_1,\dots,u_q$ terms such that each t_i realizes $\varphi_i^{\mathscr{A}}$ and each u_j belongs to $\mathcal{E}^{\mathscr{A}}\vec{\tau}_j^{\mathscr{A}}$. For each term t, we adopt for brevity the notation $t' = t\left[{}^{x_1,\dots,x+p,y_1,\dots,y_q}\big/{}_{x_1,\dots,x_p,y_1,\dots,y_q}\right]$. We prove that each deduction rule is correct as follows:

- For the rule $\dfrac{}{\Gamma \vdash \alpha}$ (ax) where $\alpha{\in}\Gamma$, is immediate.
- For the rule $\dfrac{\Gamma, x : \varphi \vdash t : \psi}{\Gamma \vdash \lambda xt : \varphi \to \psi}$ (abs): We must prove that $\Gamma{\vdash}\lambda xt{:}\varphi{\to}\psi$ is true in $\perp\!\!\!\perp$ provided that $\Gamma, x : \varphi \vdash t : \psi$ is true in $\perp\!\!\!\perp$. Take a term $v \Vdash \varphi^{\mathscr{A}}$ and a stack $\pi \in ||\psi^{\mathscr{A}}||$. The process $\lambda xt' \star v.\pi$ reduces to $t'[{}^{v}/{}_{x}] \star \pi$. Since we have that $\Gamma, x : \varphi \vdash t : \psi$ is true in $\perp\!\!\!\perp$, $t'[{}^{v}/{}_{x}] \star \pi \in \perp\!\!\!\perp$. The reader should notice that, since substitution avoid capture of free variables -possibly by renaming variables- we can assert that x is not free in $t_1,\dots,t_p,u_1,\dots,u_q,v$. Hence, the sequential substitution $t'[{}^{v}/{}_{x}]$ gives the same result as the simultaneous substitution $t\left[{}^{t_1,\dots,t_p,v,u_1,\dots,u_q}\big/{}_{x_1,\dots,x_p,x,y_1,\dots,y_q}\right]$.
- For the rule $\dfrac{\Gamma \vdash t : \varphi \to \psi \quad \Gamma \vdash u : \varphi}{\Gamma \vdash (t)u : \psi}$ (app): Suppose that the sequents $\Gamma \vdash t : \varphi \to \psi$ and $\Gamma \vdash u : \varphi$ are true in $\perp\!\!\!\perp$. Take a stack $\pi \in ||\psi^{\mathscr{A}}||$. We must prove that $(t')u' \star \pi \in \perp\!\!\!\perp$. But $(t')u' \star \pi$ reduces to $t' \star u'.\pi$, that belongs to $\perp\!\!\!\perp$ by hypothesis.

- For the rule $\dfrac{\Gamma, x{\in}\varepsilon\vec{\tau} \vdash t : \psi}{\Gamma \vdash \lambda x t : \varepsilon\vec{\tau} \rightsquigarrow \psi}$ $(\rightsquigarrow_i)$: Similar to the case of the rule (abs)
- For the rule $\dfrac{\Gamma \vdash t{:}\varepsilon_j\vec{\tau}_j \rightsquigarrow \psi \quad \Gamma \vdash y{\in}\varepsilon\vec{\tau}}{\Gamma \vdash (t)y{:}\psi}$ $(\rightsquigarrow_e)$: Suppose that $\Gamma \vdash t : \varepsilon_j\vec{\tau}_j \rightsquigarrow \psi$ is true in $\perp\!\!\!\perp$. Then, $t' \Vdash \varepsilon_j\vec{\tau}_j \rightsquigarrow \psi$. By hypothesis $u_j \in \varepsilon_j^{\mathscr{A}}(\vec{\tau}_j^{\,\mathscr{A}})$ and hence $((t)y_j)' = (t')u_j \Vdash \psi^{\mathscr{A}}$.
- For the rule $\dfrac{\Gamma \vdash t : \psi}{\Gamma \vdash t : \forall x\psi}$ $(\forall_i)^1$ where $x \notin \mathrm{FV}(\Gamma)$: We must prove that $\Gamma \vdash t : \forall x\psi$ is true in $\perp\!\!\!\perp$ provided that $\Gamma \vdash t : \psi$ is true in $\perp\!\!\!\perp$. Take a stack $\pi \in ||\psi^{\mathscr{A}}[{}^{m}/_{x}]||$ for an individual m. Since the sequent $\Gamma \vdash t : \psi$ is true in $\perp\!\!\!\perp$; then $t' \Vdash \psi^{\mathscr{A}}[{}^{m}/_{x}]$ and hence $t' \star \pi \in \perp\!\!\!\perp$.
- For the rule $\dfrac{\Gamma \vdash t : \forall x\psi}{\Gamma \vdash t : \psi[{}^{\tau}/_{x}]}$ $(\forall_e)^1$: Suppose that $\Gamma \vdash t : \forall x\psi$ is true in $\perp\!\!\!\perp$ and take a stack $\pi \in ||\psi^{\mathscr{A}}[{}^{\tau}/_{x}]||$. We have that $||\psi^{\mathscr{A}}[{}^{\tau}/_{x}]|| = ||\psi^{\mathscr{A}}[{}^{m}/_{x}]|| \subseteq ||\forall x\psi||$ where $m = \tau^{\mathscr{A}}$. Since we suppose that $\Gamma \vdash t{:}\forall x\psi$ is true in $\perp\!\!\!\perp$, we have that $t' \star \pi \in \perp\!\!\!\perp$, thus proving the result.
- For the rule $\dfrac{\Gamma \vdash t : \psi}{\Gamma \vdash t : \forall X\psi}$ $(\forall_i)^2$ where $X \notin \mathrm{FV}(\Gamma)$: We must prove that $\Gamma \vdash t : \forall X\psi$ is true in $\perp\!\!\!\perp$ provided that $\Gamma \vdash t : \psi$ is true in $\perp\!\!\!\perp$. Take a stack $\pi \in ||\psi^{\mathscr{A}}[{}^{\mathbb{P}}/_{X}]||$ for a ($\mathscr{T}$-valued) second order parameter $\mathbb{P}$ with the arity of X. Since the sequent $\Gamma \vdash t : \psi$ is true in $\perp\!\!\!\perp$, we have that $t' \Vdash \psi^{\mathscr{A}}[{}^{\mathbb{P}}/_{X}]$ and hence $t' \star \pi \in \perp\!\!\!\perp$.
- For the rule $\dfrac{\Gamma \vdash t : \forall X\psi}{\Gamma \vdash t : \psi[{}^{\chi}/_{Xx_1,\dots,x_k}]}$ $(\forall_e)^2$: Suppose that $\Gamma \vdash \forall X\psi$ is true in $\perp\!\!\!\perp$. Hence, $t' \Vdash \forall X\psi$. On the other hand, by 2.22 we have that $||\psi[{}^{\chi}/_{Xx_1,\dots,x_k}]|| = ||\psi[{}^{\mathbb{P}}/_{X}]||$ for a suitable ($\mathscr{T}$-valued) predicate $\mathbb{P}$. In consequence $t' \Vdash \psi[{}^{\chi}/_{X}]$, as we want prove.
- For the rule $\dfrac{}{\Gamma \vdash \mathrm{cc} : \forall X\forall Y[((X \to Y) \to X) \to X]}$ (cc): Since cc is a closed Λ_c-term, we must prove that $\mathrm{cc} \Vdash \forall X\forall Y[((X{\to}Y){\to}X){\to}X]$ Consider two predicates $\mathbb{X}, \mathbb{Y}$, a Λ_c-term t such that $v \Vdash (\mathbb{X}{\to}\mathbb{Y}){\to}\mathbb{X}$ and a stack $\pi \in \mathbb{X}$. The process $\mathrm{cc}{\star}v.\pi \succ k_\pi.\pi$. It suffices to prove that $k_\pi \Vdash \mathbb{X} \to \mathbb{Y}$. Take a stack $\rho \in \mathbb{Y}$ and a Λ_c-term $u \Vdash \mathbb{X}$. Since the process $k_\pi \star u.\rho \succ u \star \pi$ that belongs to $\perp\!\!\!\perp$. The proof is finished.

□

A fundamental step in the proof above consists in proving that cc is a universal realizer for Peirce's Law. In general, to introduce a new typing rule $\dfrac{}{\Gamma \vdash t{:}\Phi}$, we need only to verify that

$$t[{}^{t_1,\dots,t_p,u_1,\dots,u_q}/_{x_1,\dots,x_p,y_1,\dots,y_q}] \Vdash\!\!\vdash \Phi$$

provided that $t_i \Vdash \varphi_i$ and $u_j \in \varepsilon^{\mathscr{A}}\vec{\tau}^{\mathscr{A}}$ for all i and j. This is necessary because the Soundness Lemma holds in the new type system if and only if each typing rule is correct. In consequence, the problem consisting in introducing an axiom in the typing system is in fact equivalent to the problem consisting in realizing the axiom. A simple but very strong property is that the correctness of a typing rule is independent of the other and hence, once the correctness of a rule is proven, we can add other typing rules without risk of injury to the already established rules. This kind of modularity makes safe and relatively simple to add instructions every time we want.

2.5. **Defining Tarski models from realizability models.**

We start putting, given a realizability model, a structure of boolean algebra on a quotient over $\mathscr{T}$:

Definition 2.25. *Given a Tarski model $\mathfrak{M} = (M, \mathcal{D}, \mathscr{Y})$ and a realizability model $\perp\!\!\!\perp$, we say that $\perp\!\!\!\perp$ is* consistent *if and only if there is no proof-like terms realizing $\perp$ on $\perp\!\!\!\perp$.*

Definition 2.26. *Consider a Tarski model $\mathfrak{M} = (M, \mathcal{D}, \mathscr{Y})$ and a consistent realizability model $\perp\!\!\!\perp$. We define in the set $\mathscr{T}$ the following relations:*

- $\mathbb{P} \leq \mathbb{Q}$ *if and only if there is a proof-like Λ_c-term t such that $t \Vdash \mathbb{P} \to \mathbb{Q}$*
- $\mathbb{P} \simeq \mathbb{Q}$ *if and only if $\mathbb{P} \leq \mathbb{Q}$ and $\mathbb{Q} \leq \mathbb{P}$*

The relation $\simeq$ is an equivalence relation in $\mathscr{T}$. We will denote as $\mathbb{P}$ both the truth value $\mathbb{P}$ and its equivalence class.

Definition 2.27. *In the hypothesis of the definition 2.26, we define on $\mathscr{T}/\simeq$ the following operations:*

- $\neg\mathbb{P} := ||\mathbb{P} \to \perp||$
- $\mathbb{P} \wedge \mathbb{Q} := ||\forall X((\mathbb{P}, \mathbb{Q} \to X) \to X)||$
- $\mathbb{P} \vee \mathbb{Q} := ||\forall X((\mathbb{P} \to X), (\mathbb{Q} \to X) \to X)||$

We define also the constants $\top := \emptyset$ and $\perp := \Pi$

Lemma 2.28. $(\mathscr{T}/\simeq, \perp, \top, \neg, \leq, \wedge, \vee)$ *constitutes a* Boolean algebra.

Proof. The proof is rather long but conceptually no difficult. We will only propose in most paradigmatic properties, the necessary terms to realize it and we left the verification for the reader.

First, we must verify that the operations are defined in the quotient. In other words, we must check that $\neg\mathbb{P} \simeq \neg\mathbb{P}'$, $\mathbb{P} \wedge \mathbb{Q} \simeq \mathbb{P}' \wedge \mathbb{Q}'$ and $\mathbb{P} \vee \mathbb{Q} \simeq \mathbb{P}' \vee \mathbb{Q}'$ provided that $\mathbb{P} \simeq \mathbb{P}'$ and $\mathbb{Q} \simeq \mathbb{Q}'$. To do this, it suffices to verify:

- $\lambda p \lambda f \lambda x (f)(p)x \Vdash \forall X \forall X'((X \to X'), \neg X' \to \neg X)$
- $\lambda p \lambda q \lambda f \lambda g (f) \lambda x \lambda y (g)(p)x(q)y \Vdash$
$$\forall X \forall X' \forall Y \forall Y'((X \to X'), (Y \to Y'), (X \wedge Y) \to X' \wedge Y')$$
- $\lambda p \lambda q \lambda f \lambda x \lambda y ((f)x \circ p)y \circ q \Vdash$
$$\forall X \forall X' \forall Y \forall Y'((X \to X'), (Y \to Y'), (X \vee Y) \to (X' \vee Y'))$$

After that, we must prove that $\leq$ is transitive, $\wedge$ and $\vee$ are respectively the inf and the sup. For $\leq$ is simple, for $\wedge$, we must check:

- $\lambda f(f)\lambda a\lambda ba \Vdash \forall A\forall B(A\wedge B\to A)$
- $\lambda f(f)\lambda a\lambda bb \Vdash \forall A\forall B(A\wedge B\to B)$
- $\lambda a\lambda b\lambda x\lambda f(f)(a)x(b)x \Vdash \forall A\forall B\forall X((X\to A),(X\to B),X\to(A\wedge B))$

and for $\vee$:

- $\lambda a\lambda f\lambda g(f)a \Vdash \forall A\forall B(A\to(A\vee B))$
- $\lambda b\lambda f\lambda g(g)b \Vdash \forall A\forall B(B\to(A\vee B))$
- $\lambda a\lambda b\lambda f(f)ab \Vdash \forall X\forall A\forall B((A\to X),(B\to X),(A\vee B)\to X)$

To prove the properties for the complement, we check:

- Since $\lambda a\lambda f(\mathrm{cc})\lambda k(f)a\circ k \Vdash \forall X(X\vee\neg X)$ we have that $\forall X(\top\to X\vee\neg X)$ is realized by a proof-like term.
- $\lambda f(f)\lambda a\lambda b(b)a \Vdash \forall X((X\wedge\neg X)\to\bot)$

One of the distributive laws can be proved using:

- $\lambda h\lambda v_1\lambda v_2(h)\lambda a(u)(\tau)v_1a(\tau)v_2a \Vdash$
$$\forall A\forall B\forall C[(A\wedge(B\vee C))\to((A\wedge B)\vee(A\wedge C))]$$
where $\tau=\lambda t\lambda x\lambda y(t)\lambda f(f)xy \Vdash \forall X\forall Y\forall Z(((X\wedge Y)\to Z),X,Y\to Z)$ and u is the term found to realize:
$$\forall X\forall A\forall B((A\to X),(B\to X),(A\vee B)\to X)$$

The other computations to prove the distributive laws are left as an exercise.

□

Thus, in the quotient $\mathscr{T}/\simeq$, the formulæ that are realizable by a proof-like term have truth value $\top$; while formulæ that are inconsistent with $\top$ have truth value $\bot$.

Let us consider an ultrafilter U in $\mathscr{T}/\simeq$. It represents a complete consistent set of formulæ: the set $\mathcal{L}_U$ of all $\mathcal{L}$-formulæ φ such that the equivalence class (module $\simeq$) of $||\varphi||$ belongs to U. By completeness, there is a Tarski model $\mathfrak{M}$ such that its $\mathcal{L}$-theory is exactly $\mathcal{L}_U$ (c.f.: 2.11). This remark proves that each consistent realizability model gives a class of Tarski models where all formulæ realized by a proof-like term are true. We will use the locution *realizability model* indistinctly for the saturated set of processes $\perp\!\!\!\perp$ and for any Tarski model given by an ultrafilter in the boolean algebra $\mathscr{T}/\simeq$.

Remark 2.29. *Consider a $\mathcal{L}$-structure $\mathfrak{M}=(M,\mathcal{D},\mathscr{Y})$ and a consistent realizability model $\perp\!\!\!\perp$. Then, for each parametrical first order formula $\varphi(m_1,\dots,m_h,R_1,\dots,R_k)$:*

$$\textit{if } \mathfrak{M}\models\varphi(m_1,\dots,m_h,R_1,\dots,R_k) \textit{ then } ||\varphi(m_1,\dots,m_h,R_1,\dots,R_k)||\simeq\top$$

Remark 2.30. Let us consider a $\mathcal{L}$-structure $\mathfrak{M}=(M,\mathcal{D},\mathscr{Y})$ and a consistent realizability model $\perp\!\!\!\perp$. According to 2.28, it determines a boolean algebra $\mathscr{T}/\simeq$ and, by composition with the canonical projection of $\simeq$, we have also a map $\mathcal{L}\to\mathscr{T}/\simeq$, which is a semantics over $\mathcal{L}$. Is this semantics a boolean-valued model over $\mathfrak{M}$? Strictly speaking not necessarily, because the universal quantifiers are not necessarily interpreted as infimums. More precisely, $||\forall x\varphi(x)||$ is not in general the infimum of the set $\{||\varphi(n)|| \mid n \in M\}$.

Certainly, $||\forall x\varphi(x)|| \leq ||\varphi(n)||$ for all $n{\in}M$. However, if we consider a truth value $\mathbb{P}$ such that for each individual $n{\in}M$ $\mathbb{P} \leq ||\varphi(n)||$, in general it is not true that $\mathbb{P} \leq ||\forall x\varphi(x)||$. We will see it in section 4, taking as counter-example $\varphi(x) = \mathrm{int}(x)$ and a suitable consistent realizability model. The deep reason for that is: While to realize each instance of $\varphi(n)$ in $\perp\!\!\!\perp$ means that for each individual n there is a term t_n that realizes $\varphi(n)$; to realize $\forall x\varphi(x)$ in $\perp\!\!\!\perp$ means that there is one term that realizes each instance of $\varphi(n)$.

Remark that, if we want to define a semantics such that the $\forall$-introduction and elimination rules are correct, we must define $||\forall x\varphi(x)||$ as $\bigcup_{n\in\mathbb{N}} ||\varphi(n)||$ (at least of $\simeq$). Indeed: the correctness of the rule $\dfrac{\vdash t : \forall x\varphi(x)}{\vdash t : \varphi[^{\tau}\!/_{x}]}$ implies $\mathrm{id} \Vdash ||\forall x\varphi(x)|| \to \bigcup_{m\in M} ||\varphi(m)||$ because for each $m{\in}M$, the term $\tau{=}\underbrace{\mathsf{s}\ldots\mathsf{s}}_{n}0$ is interpreted as n. Conversely, the correctness of the rule $\dfrac{\vdash t : \varphi(x)}{\vdash t : \forall x\varphi(x)}$ implies that $\mathrm{id} \Vdash \bigcup_{m\in M} ||\varphi(m)|| \to ||\forall x\varphi(x)||$ (c.f.: 2.23).

2.6. **Arithmetics in realizability models.**

On 2.9, we have defined the Peano axioms PA1–PA7. Consider a Tarski model $\mathfrak{M} = (M, \mathcal{D}, \mathcal{Y})$ such that $\mathfrak{M} \models$ PA1–PA6 (Peano arithmetics without recursion principle). We can realize (uniformly on $\perp\!\!\!\perp$) the Peano axioms PA1-PA6 (c.f. [6]). This fact can be seen as a direct consequence of 2.29, because all formulæ PA1–PA6 are *first order* formulæ with parameters on $\mathfrak{M}$ which are true in $\mathfrak{M}$.

Remark 2.31. On the other hand, as we will easily see in 3.9, there is no universal realizer for PA7. The reader should be aware that, if we choose as $\mathfrak{M}$ the *standard model* $(\mathbb{N}, \bigcup_{k\in\mathbb{N}} \mathcal{P}ow(\mathbb{N}^k), \mathcal{Y})$, then all parametrical formulæ $(\mathrm{int}(n))_{n\in\mathbb{N}}$ are realizable. The difference between to realize each instance of $\mathrm{int}(x)$ and to realize $\forall x\,\mathrm{int}(x)$ is that in the first case we can use a different term to realize each different instance of $\mathrm{int}(x)$, while in the second, a term realizing $\forall x\,\mathrm{int}(x)$ is a term which realizes *all* instances of $\mathrm{int}(x)$. This implies that we can not add PA7 to our typing system. In fact, there are interesting realizability models built from models of Peano arithmetics (PA7 included), which does not satisfies PA7 (c.f.: the *standard generic model* in [7]).

However, in order to transform arithmetical theorems into programs, in particular we must transform theorems which are proved using PA7. The solution we adopt to this problem is to relativize (the quantifiers of) $\mathcal{L}$-formulæ to the formula int.

Lemma 2.32. *Let Φ be a $\mathcal{L}$-formula provable in Peano arithmetics. Then, Φ^{int} is provable from PA1-PA6 and $\overset{\text{int}}{\forall} x_1 \dots \overset{\text{int}}{\forall} x_k \operatorname{int}(fx_1 \dots x_k)$ for all k-ary function symbol used in the proof of ϕ*

Then, in order to realize Φ^{int} we must realize the formulæ

$$\forall x_1 \dots \forall x_k \operatorname{int}(x_1), \dots, \operatorname{int}(x_k) \to \operatorname{int}(fx_1 \dots x_k)$$

for each k-ary function symbol f used in the proof of Φ. This is possible for function symbols which are interpreted as recursive functions.

Lemma 2.33. *Consider a start model $\mathfrak{M} = (M, \mathcal{D}, \mathscr{Y})$, a k-ary function symbol f and a realizability model $\perp\!\!\!\perp$. If $f^{\mathscr{Y}}$ is a recursive function, then the formula $\forall x_1 \dots \forall x_k(\operatorname{int}(x_1), \dots, \operatorname{int}(x_k) \to \operatorname{int}(fx_1 \dots x_k))$ is realized in $\perp\!\!\!\perp$ by a proof-like term.*

Proof. The proof can be founded in [6] □

A deep work about realizability models of arithmetics can be founded in [10].

3. **Data types**

In this section, we will consider realizability models where the start model is an ω-model of Peano arithmetics.

3.1. **Definitions and first results.**

Definition 3.1. *A* data set *is a function* $\mathcal{E}:\mathbb{N} \to \mathcal{P}\mathrm{ow}(\mathrm{Pl})$ *such that, for each integer* n*,* $\mathcal{E}(n)$ *has at most one term and whenever* $\mathcal{E}(n) \neq \emptyset$ *or* $\mathcal{E}(m) \neq \emptyset$*,* $\mathcal{E}(m) = \mathcal{E}(n)$ *implies* $m = n$*.*

In other words, a data set is a way to choose –in an injective way– at most one Pl-term for each integer n.

Definition 3.2. *The* data types *are of two possible shapes:*

(1) *A data set* $\mathcal{E}:\mathbb{N} \to \mathcal{P}\mathrm{ow}(\mathrm{Pl})$*.*
(2) *A* $\mathcal{P}$*-formula* $\varphi(x)$ *such that there is a data set* $\mathcal{E}_\varphi:\mathbb{N}\to\mathcal{P}\mathrm{ow}(\mathrm{Pl})$ *and a* Pl*-term* T_φ *verifying:*
   - *If* $\mathcal{E}_\varphi(n)$ *is not empty, then* $\mathcal{E}_\varphi(n) = \{e_n\}$ *for a suitable* Pl*-term* e_n *such that* $e_n \Vdash\!\vdash \varphi(n)$*.*
   - $\mathrm{T}_\varphi \Vdash\!\vdash \forall X \forall x[\mathcal{E}_\varphi(x) \rightsquigarrow X, \varphi(x) \to X]$*.*

Given a data type φ*, we call* T_φ *a* storage operator *associated with* φ *and* $\mathcal{E}_\varphi$ *a* canonical representation *for* φ*. Given a data set* $\mathcal{E}$*, we define the storage operator* $\mathrm{T}_\mathcal{E}$ *as the identity term* id *and we say that* $\mathcal{E}$ *is itself its canonical representation. A term* t *represents a data of type* φ *if and only if* $t \Vdash\!\vdash \varphi(n)$ *for a suitable integer* n *and represents a data of type* $\mathcal{E}$ *if and only if* $t \in \mathcal{E}(n)$ *for a suitable integer* n*. Given a data type* $\varphi(x)$*, the* scope *of* φ *is the set* $S_\varphi := \{n \in \mathbb{N} \mid \mathcal{E}_\varphi(n) \neq \emptyset\}$*.*

Intuitively, given a model, a data type $\varphi(x)$ is a set of integers: the set of all the integers n verifying that $\varphi(n)$ is realized in the model by a proof-like term. Each proof-like term realizing $\varphi(n)$ is a Λ_c-term representing n as a data of type φ in the given model. A canonical representation of $\varphi(n)$ represents n in all models and has a sort of universal property: in each model, a function defined in canonicals can be extended, by means of a storage operator, to the whole representation of the data type.

An important property of storage operators is that it represents the *call-by-value* of *imperative languages*. This is explained in the following lemma:

Lemma 3.3. *Consider a data type* φ *with storage operator* T_φ *and canonical representation* $\mathcal{E}_\varphi$*. Suppose that* ϕ *represents a data of type* φ *for all models (i.e.* $\phi \Vdash\!\vdash \varphi(n)$ *for an integer* n*). Take a* halt *instruction* H *-this is an instruction with no reduction rule- and a stack* π*. Then* $\mathrm{T}_\varphi H \star \phi.\pi \succ H \star e_n.\pi$*. In particular,* $\mathcal{E}_\varphi(n) \neq \emptyset$*, i.e.* $n \in S_\varphi$*.*

Proof. Consider $\perp\!\!\!\perp := \mathrm{th}^c_{\mathrm{T}_\varphi H \star \phi.\pi}$. Since $\mathrm{T}_\varphi \Vdash \mathcal{E}(n) \rightsquigarrow \{\pi\}, \varphi(n) \to \{\pi\}$ and $\phi \Vdash \varphi(n)$, we can assert that $H \not\Vdash \mathcal{E}(n) \rightsquigarrow \{\pi\}$. In consequence, $H \star e_n.\pi \notin$

$\perp\!\!\!\perp$ and then, it must appear in the thread of $\mathrm{T}_\varphi H \star \phi.\pi$, thus proving that $\mathrm{T}_\varphi H \star \phi.\pi$ reduces to $H \star e_n.\pi$. □

Corollary 3.4. *Given a data type $\varphi(x)$, the scope S_φ is the set*

$$\{n \in \mathbb{N} \mid \textit{there is a } \Lambda_c\textit{-term } \phi \textit{ such that } \phi \Vdash\!\!\Vdash \varphi(n)\}$$

Remark 3.5. *By substitution, for all terms θ and for all stacks π, $\mathrm{T}_\varphi \theta \star \phi.\pi \succ \theta \star e_n.\pi$.*

Corollary 3.6. *If $e_n \Vdash\!\!\Vdash \varphi(m)$, then $n = m$.*

Proof. By 3.3 the process $\mathrm{T}_\varphi H \star e_n.\pi$ reduces to $H \star e_n.\pi$ and $H \star e_m.\pi$, because $e_n \Vdash\!\!\Vdash \varphi(n)$ and $e_n \Vdash\!\!\Vdash \varphi(m)$. But, since H is a halt instruction, processes $H \star e_n.\pi$ and $H \star e_m.\pi$ are the same. Then $e_n = e_m$ and hence $n = m$. □

Corollary 3.7. *Given a data type $\varphi(x)$, there is no term realizing in all models simultaneously $\varphi(n)$ and $\varphi(m)$ with $n \neq m$. Consequently, given a data type $\varphi(x)$ containing more than one data, it is impossible to realize $\forall x \varphi(x)$ in all models.*

3.2. **Integers.**

Example 3.8. The formula $\mathrm{int}(x) := \forall X[\forall y(Xy \to X\mathrm{s}y), X0 \to Xx]$ is a data type. Indeed, define for all integers n the term $\bar{n}$ as $\underbrace{(\mathrm{s})(\mathrm{s})\dots(\mathrm{s})}_{n}0$, the Λ_c-term $\gamma := \lambda g\ g \circ \mathrm{s}$, and the function $\mathcal{E}_{\mathrm{int}}(n) := \{\underline{\bar{n}}\}$. Define as storage operator the term $\mathrm{T}_{\mathrm{int}} := \lambda f \lambda n(n)\gamma f \bar{0}$. We check that data type definition holds:

- We prove by induction that, for all integers n, $\bar{n} \Vdash\!\!\Vdash \mathrm{int}(n)$. For n=0 it is trivial. Consider a model $\perp\!\!\!\perp$ and suppose that $\bar{n} \Vdash \mathrm{int}(n)$. We must prove that $\overline{n{+}1} \Vdash \mathrm{int}(n{+}1)$. Take a stack $\rho \in ||\mathrm{int}(n{+}1)||$. The process $\overline{n{+}1} \star \rho$ reduces to $\mathrm{s} \star \bar{n}.\rho$. Since $\mathrm{s} \Vdash \mathrm{int}(n) \to \mathrm{int}(n{+}1)$, we have that $\mathrm{s} \star \bar{n}.\rho \in \perp\!\!\!\perp$ and hence $\overline{n{+}1} \star \rho \in \perp\!\!\!\perp$.
- Take a model $\perp\!\!\!\perp$, a parameter $\mathbb{X}$ and an integer n. Consider a Λ_c-term $\phi \Vdash \mathcal{E}_{\mathrm{int}}(n) \rightsquigarrow \mathbb{X}$, a Λ_c term $\nu \Vdash \mathrm{int}(n)$ and a stack $\pi \in \mathbb{X}$. The process $\mathrm{T}_{\mathrm{int}} \star \phi.\nu.\pi$ reduces to $\nu \star \gamma.\phi.\bar{0}.\pi$. Define a predicate $\mathbb{P}(i) := \{\overline{n{-}i}\} \rightsquigarrow \mathbb{X}$ if $i \le n$ and $\mathbb{P}(i) = \emptyset$ if $i > n$. It is no difficult to prove that $\gamma \Vdash \forall y(\mathbb{P}(y) \to \mathbb{P}(\mathrm{s}y))$. Moreover, $\phi \Vdash \{\bar{n}\} \rightsquigarrow \mathbb{X}$ and $\bar{0}.\pi \in \mathbb{P}(n)$. On the other hand, $\nu \Vdash \forall y(\mathbb{P}(y) \to \mathbb{P}(\mathrm{s}y)), \mathbb{P}(0) \to \mathbb{P}(n)$. Hence $\nu \star \gamma.\phi.\bar{0}.\pi \in \perp\!\!\!\perp$, thus proving the result.

Remark 3.9. Applying 3.7 to the particular case of integers, we can conclude the result stated in 2.31, more precisely that it is impossible to find a universal realizer of the formula $\forall x\, \mathrm{int}(x)$.

3.3. **Booleans.**

Example 3.10. The formula $\mathrm{Bool}(x) := \forall X[X1, X0 \to Xx]$ is a data type: Define $\mathcal{E}_{\mathrm{Bool}}(0) := \{\lambda x \lambda y y\}$, $\mathcal{E}_{\mathrm{Bool}}(1) := \{\lambda x \lambda y x\}$ and $\forall x > 1$, $\mathcal{E}_{\mathrm{Bool}}(x) := \emptyset$. Denote $\lambda x \lambda y y$ as $\mathbb{0}$ and $\lambda x \lambda y x$ as $\mathbb{1}$. It is immediate that $\mathbb{1} \Vdash \mathrm{Bool}(1)$ and $\mathbb{0} \Vdash \mathrm{Bool}(0)$. Consider the operator $\mathrm{T}_{\mathrm{Bool}} := \lambda f \lambda b(b)(f)\mathbb{1}(f)\mathbb{0}$. $\mathrm{T}_{\mathrm{Bool}}$ is a storage operator for Bool: Take a model $\perp\!\!\!\perp$ and a truth value $\mathbb{X}$. We must prove that $\mathrm{T}_{\mathrm{Bool}} \Vdash \forall x[\mathcal{E}_{\mathrm{Bool}}(x) \rightsquigarrow \mathbb{X}, \mathrm{Bool}(x) \to \mathbb{X}]$

- For $x = 0$: Take $f \Vdash \mathcal{E}_{\mathrm{Bool}}(0) \rightsquigarrow \mathbb{X}$, $b \Vdash \mathrm{Bool}(0)$ and $\pi \in \mathbb{X}$. The process $\mathrm{T}_{\mathrm{Bool}} \star f.b.\pi$ reduces to $b \star (f)\mathbb{1}.(f)\mathbb{0}.\pi$. On the other hand, $b \Vdash \top, \mathbb{X} \to \mathbb{X}$ and $(f)\mathbb{0} \Vdash \mathbb{X}$. Hence, $b \star (f)\mathbb{1}.(f)\mathbb{0}.\pi \in \perp\!\!\!\perp$ thus proving that $\mathrm{T}_{\mathrm{Bool}} \star f.b.\pi \in \perp\!\!\!\perp$.
- For $x = 1$: Is similar to the case $x = 0$.
- For $x > 1$: Take a term f, a term $b \Vdash \mathrm{Bool}(x)$ and a stack $\pi \in \mathbb{X}$. Since $x \neq 0$ and $x \neq 1$, $b \Vdash \top, \top \to \bot$. The process $\mathrm{T}_{\mathrm{Bool}} \star f.b.\pi$ reduces to $b \star (f)\mathbb{1}.(f)\mathbb{0}.\pi$ that belongs to $\perp\!\!\!\perp$. Hence $\mathrm{T}_{\mathrm{Bool}} \star f.b.\pi \in \perp\!\!\!\perp$, thus proving the result.

3.4. **Products.**

Theorem 3.11. *Consider* $\delta_1, \ldots, \delta_k$ *k data types with respective canonical representations* $\mathcal{E}_1, \ldots, \mathcal{E}_k$. *Then, there is an operator* $\mathrm{T}_{\delta_k, \ldots, \delta_1}$ *such that* $\mathrm{T}_{\delta_k, \ldots, \delta_1} \Vdash \forall X \forall \vec{x}[\mathcal{E}_k(x_k); \ldots; \mathcal{E}_1(x_1) \rightsquigarrow X, \delta_k(x_k), \ldots, \delta_1(x_1) \to X]$ *where* $\vec{x}$ *is the list of variables* $x_k, \ldots, x_1$.

Proof. We prove the result by induction on k. For $k = 1$ is a consequence of data type definition. Suppose the result holds for k. We prove it for $k+1$:

Define $\mathrm{T}_{\delta_{k+1}, \ldots, \delta_1} := \lambda f \lambda \phi_{k+1}(\mathrm{T}_{\delta_k, \ldots, \delta_1})(\mathrm{T}_{\delta_{k+1}}) f \phi_{k+1}$. Take a model $\perp\!\!\!\perp$, a truth value $\mathbb{X}$, $k+1$ integers $n_1, \ldots, n_{k+1}$, a term $f \Vdash \mathcal{E}_{k+1}(n_1); \ldots; \mathcal{E}_1(n_1) \rightsquigarrow \mathbb{X}$, a term $\phi_{k+1} \Vdash \delta_{k+1}(n_{k+1})$ and a stack $\rho \in ||\delta_k(n_k), \ldots, \delta_1(n_1) \to \mathbb{X}||$. We must prove that the process $\mathrm{T}_{\delta_{k+1}, \ldots, \delta_1} \star f.\phi_{k+1}.\rho$ belongs to $\perp\!\!\!\perp$. By reduction, it is sufficient to show that $(\mathrm{T}_{\delta_k, \ldots, \delta_1})(\mathrm{T}_{\delta_{k+1}}) f.\phi_{k+1} \star \rho \in \perp\!\!\!\perp$. By induction hypothesis:

$$\mathrm{T}_{\delta_k, \ldots, \delta_1} \Vdash \mathcal{E}_k(n_k); \ldots; \mathcal{E}_1(n_1) \rightsquigarrow \mathbb{X}, \delta_k(n_k), \ldots, \delta_1(1) \to \mathbb{X}$$

$\mathrm{T}_{\delta_{k+1}} \Vdash \mathcal{E}_{k+1}(n_{k+1}); \mathcal{E}_k(n_k); \ldots; \mathcal{E}_1(n_1) \rightsquigarrow \mathbb{X}, \delta_{k+1}, \mathcal{E}_k(n_k); \ldots; \mathcal{E}_1(n_1) \rightsquigarrow \mathbb{X}$
Then $(\mathrm{T}_{\delta_{k+1}}) f \phi_{k+1} \Vdash \mathcal{E}_k(n_{k+1}); \ldots; \mathcal{E}_1(n_1) \rightsquigarrow \mathbb{X}$ and $(\mathrm{T}_{\delta_k, \ldots, \delta_1})(\mathrm{T}_{k+1}) f \phi_{k+1} \Vdash \mathbb{X}$, thus proving the result. □

Intuitively, a such operator acts as an storage operator for a kind of product data type formed by the factors $\delta_1, \ldots, \delta_k$.

3.5. **Lists.**

We treat now the case of another recursive data type: the finite lists elements of wich are taken from a given data type δ (called *"lists of* δ*"* and denoted by L_δ). These lists are built from the empty list $\langle\rangle$, adding elements with a constructor $(:) : \delta, \mathrm{L}_\delta \to \mathrm{L}_\delta$. Thus, the list $\langle n_1, n_2, n_3 \rangle$ can be constructed in a model $\mathfrak{M}$ if and only if $\delta(n_1), \delta(n_2), \delta(n_3)$ are true in $\mathfrak{M}$ and this list is built by $n_1{:}n_2{:}n_3{:}\langle\rangle$ (using here infix notation for $(:)$).

Since our models are denumerable, we can define a recursive coding for the finite lists of individuals. We will represent in the model indistinctly the list of $n_1,\dots,n_k$ and its code as $\langle n_1,\dots,n_k\rangle$, or as $n_1{:}\langle n_2,\dots,n_k\rangle$, or as $n_1 : n_2 : \langle n_3,\dots,n_k\rangle$ and so one.

Definition 3.12. *Given a data type δ, we define the formula*

$$\mathrm{L}_\delta(x) := \forall X[\forall y\forall z(\delta(y), Xz \to Xy{:}z), X\langle\rangle \to Xx]$$

Remark 3.13. This formula means that an individual x in a model is the (code of a) list of δ if and only if it is in any set containing the (code of the) empty list and closed by the constructor $(:)$ (coded in the model), whenever the $(:)$ first argument is in the data type δ. Briefly: the set of lists of δ is the smallest set containing the empty list and closed by addition of δ data.

Definition 3.14. *Given k Λ_c-terms $a_1,\dots,a_k$, the normalized list formed by these terms is the Λ_c-term $\lambda f\lambda x(f)a_1\dots(f)a_kx$. We denote this term as $\underline{\langle} a_1,\dots,a_k\rangle$. In particular, the empty list $\langle\rangle$ is $\lambda f\lambda xx$; (already denoted as 0).*

Lemma 3.15. *Consider a data type δ, a model $\perp\!\!\!\perp$, k individuals $n_1,\dots,n_k$ and k Λ_c-terms $a_1,\dots,a_k$ such that $a_i \Vdash \delta(n_i)$ for all $i \in [1..k]$.*

Then, $\langle a_1,\dots,a_k\rangle \Vdash \mathrm{L}_\delta(\langle n_1,\dots,n_k\rangle)$

Proof. Choose a 1-ary predicate $\mathbb{X}$, two Λ_c-terms f,o and a stack π such that $f \Vdash \forall y\forall z(\delta(y), \mathbb{X}(z) \to \mathbb{X}(y{:}z))$, $o \Vdash \mathbb{X}\langle\rangle$ and $\pi \in \mathbb{X}(\langle n_1,\dots,n_k\rangle)$ in $\perp\!\!\!\perp$. We have that $\langle a_1,\dots,a_k\rangle \star f.o.\pi \succ (f)a_1\dots(f)a_ko \star \pi$.

Since $a_k \Vdash \delta(n_k)$, we have that $(f)a_ko \Vdash \mathbb{X}(\langle n_k\rangle)$. Then, we have that $(f)a_{k-1}(f)a_ko \Vdash \mathbb{X}(\langle a_{k-1},a_k\rangle)$ and so one. By induction we can conclude that $(f)a_1\dots(f)a_k \Vdash \mathbb{X}(\langle n_1,\dots,n_k\rangle)$. Such as $\mathbb{X}$ and π were chosen, π belongs to $\mathbb{X}(\langle n_1,\dots,n_k\rangle)$ and hence $(f)a_1\dots(f)a_k\star\pi \in \perp\!\!\!\perp$, thus proving the result. □

The operation of adding an element at the beginning of a list is given by:

Definition 3.16. cons $:= \lambda a\lambda l\lambda f\lambda x(f)a(l)fx$.

Lemma 3.17. cons $\Vdash \forall x\forall y(\delta(x), \mathrm{L}_\delta(y) \to \mathrm{L}_\delta(x{:}y))$

Proof. Take a model $\perp\!\!\!\perp$, two individuals n and l, two Λ_c-terms a,t and a stack π such that $a \Vdash \delta(n)$, $t \Vdash \mathrm{L}_\delta(l)$ and $\pi \in ||\mathrm{L}_\delta(n{:}l)||$. We have that $\mathrm{cons}\star a.t.\pi \succ \lambda f\lambda x(f)a(t)fx \star \pi$ and hence the result is proved if we can prove $\lambda f\lambda x(f)a(t)fx \Vdash \mathrm{L}_\delta(n{:}l)$. In order to do this, take a predicate $\mathbb{X}$, two Λ_c-terms u,v and a stack ρ such that $u \Vdash \forall y\forall z(\delta(y), \mathbb{X}(z) \to \mathbb{X}(y{:}z))$, $v \Vdash \mathbb{X}\langle\rangle$ and $\rho \in \mathbb{X}(n{:}l)$. The process $\lambda f\lambda x(f)a(t)fx \star u.v.\rho$ reduces to $(t)a(t)uv \star \rho$. Since $(t)uv \Vdash \mathbb{X}(l)$, we have that $(t)a(t)uv \Vdash \mathbb{X}(n{:}l)$. Furthermore, $\rho \in \mathbb{X}(n{:}l)$ and then $(t)a(t)uv \star \rho \in \perp\!\!\!\perp$, thus ending the proof. □

Corollary 3.18. *Consider a model $\perp\!\!\!\perp$, k individuals $n_1,\dots,n_k$ and k Λ_c-terms $a_1,\dots,a_k$ such that for $\underline{a}ll$ $i \in [1..k]$, we have that $a_i \Vdash \delta(n_i)$.*

Then, $(\mathrm{cons})a_1\dots(\mathrm{cons})a_k\underline{0} \Vdash \mathrm{L}_\delta(\langle n_1,\dots,n_k\rangle)$.

Proof. If $k=0$, we must verify that $\overline{0} \Vdash \mathrm{L}_\delta(\langle\rangle)$ which is true because $\overline{0} \star f.o.\pi \succ o \star \pi$ and hence, $\overline{0} \star f.o.\pi \in \perp\!\!\!\perp$ provided that $o \Vdash \mathbb{X}(0)$ and $\pi \in \mathbb{X}(0)$. If $k>0$ we can prove it by induction as a direct consequence of 3.17. This case is left as an exercise. □

Now, we are ready to prove that, given a data type δ, L_δ is also a data type. To do this, we must define canonical representations for lists:

Definition 3.19. *Consider a data type δ and k individuals $n_1,\ldots,n_k$ each of them belonging to the data type δ (i.e.: $\mathcal{E}_\delta(n_i) \neq \emptyset$ for each index $i \in [1..k]$). Denote as $e_\delta(m)$ the single element in $\mathcal{E}_\delta(m)$, whenever it exists.*

We define the canonical representation of the list $\langle n_1,\ldots,n_k\rangle$ *as the Λ_c-term* $(\mathrm{cons})e_\delta(n_1)\ldots(\mathrm{cons})e_\delta(n_k)\overline{0}$ *and we denote such a term as $\overline{\langle n_1,\ldots,n_k\rangle}$. Hence, we define $\mathcal{E}_{\mathrm{L}_\delta}(\langle n_1,\ldots,n_k\rangle) := \{\overline{\langle n_1,\ldots,n_k\rangle}\}$, whenever $n_1,\ldots,n_k$ are δ data. Otherwise $\mathcal{E}_{\mathrm{L}_\delta}(\langle n_1,\ldots,n_k\rangle) := \emptyset$*

After that, we must propose a storage operator for L_δ and thereafter we will prove it satisfies the definition of storage operator.

Definition 3.20. *Suppose δ is a data type, define $\zeta := \lambda a\lambda g\lambda x(g)(\mathrm{cons})ax$. Then, we define the storage operator for L_δ as:*

$$\mathrm{T}_{\mathrm{L}_\delta} := \lambda f \lambda l(((l)(\mathrm{T}_\delta)\zeta)\lambda y(y)\overline{0})f$$

Lemma 3.21. *If δ is a data type, then $\mathrm{T}_{\mathrm{L}_\delta} \Vvdash \forall X \forall x[\mathcal{E}_{\mathrm{L}_\delta}(x) \rightsquigarrow X, \mathrm{L}_\delta(x) \to X]$*

Proof. Consider a model $\perp\!\!\!\perp$, a predicate $\mathbb{X}$, an individual x, two Λ_c-terms f,l and a stack π such that $f \Vdash \mathcal{E}_\delta(x) \rightsquigarrow \mathbb{X}$, $l \Vdash \mathrm{L}_\delta(x)$ and $\pi \in \mathbb{X}$ in the model $\perp\!\!\!\perp$. We must prove that $\mathrm{T}_\delta \star f.l.\pi \in \perp\!\!\!\perp$. Since this process reduces to $l \star \mathrm{T}_\delta\,\zeta.\lambda y(y)\overline{0}.f.\pi$, it suffices to prove that $l \star \mathrm{T}_\delta\,\zeta.\lambda y(y)\overline{0}.f.\pi \in \perp\!\!\!\perp$. Let us define the predicate $\mathbb{Y}(z) := (\mathcal{E}_{\mathrm{L}_\delta}(z) \rightsquigarrow \mathbb{X}) \to \mathbb{X}$. Such as l was chosen, $l \Vdash \mathrm{L}_\delta(x)$ and in consequence:

$$l \Vdash \forall y \forall z(\delta(y), \mathbb{Y}(z) \to \mathbb{Y}(y{:}z)), \mathbb{Y}(0) \to \mathbb{Y}(x)$$

Then, it suffices to prove the following **claim**:

(1) $(\mathrm{T}_\delta)\zeta \Vdash \forall y \forall z(\delta(y), \mathbb{Y}(z) \to \mathbb{Y}(y{:}z))$
(2) $\lambda y(y)\overline{0} \Vdash \mathbb{Y}(0)$
(3) $f.\pi \in \mathbb{Y}(x)$

(1) Define the sequent

$$\Gamma := y \in \mathcal{E}(y),\ g{:}(\mathcal{E}'(z) \rightsquigarrow X) \to X,\ h{:}\mathcal{E}'(fyz) \rightsquigarrow X$$

where $\mathcal{E}$ and $\mathcal{E}'$ are Λ_c-set variables and f is a binary function symbol. Consider the following formal proof:

$$\dfrac{\dfrac{\Gamma, z \in \mathcal{E}'(z) \vdash (\mathrm{cons})yz \in \mathcal{E}'(fyz) \qquad \dfrac{}{\Gamma, z \in \mathcal{E}'(z) \vdash h{:}\mathcal{E}'(fyz) \rightsquigarrow X}}{\Gamma, z \in \mathcal{E}'(z) \vdash (h)(\mathrm{cons})yz{:}X}}{\Gamma \vdash \lambda z(h)(\mathrm{cons})yz : \mathcal{E}'(z) \rightsquigarrow X}$$

Since $\dfrac{}{\Gamma \vdash g{:}(\mathcal{E}'(z) \rightsquigarrow X) \to X}$ denoting for simplicity the formula $(\mathcal{E}'(z) \rightsquigarrow X) \to X$ as $\Upsilon(z)$, we have that:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash (g)\lambda z(h)(\text{cons})yz : X}}{y \in \mathcal{E}(y),\ g{:}\Upsilon(z) \vdash \lambda h(g)\lambda z(h)(\text{cons})yz : \Upsilon(fyz)}}{y \in \mathcal{E}(y) \vdash \lambda g\lambda h(g)\lambda z(h)(\text{cons})yz : \Upsilon(z) \to \Upsilon(fyz)}}{\vdash \lambda y\lambda g\lambda h(g)\lambda z(h)(\text{cons})yz : \mathcal{E}(y);\Upsilon(z) \to \Upsilon(fyz)}}{\vdash \lambda y\lambda g\lambda h(g)\lambda z(h)(\text{cons})yz : \forall y\forall z(\mathcal{E}(y),\Upsilon(z) \to \Upsilon(fyz))}$$

Now, if we assign the variable $\mathcal{E}$ as $\mathcal{E}_\delta$ (i.e. the set of canonicals for the type δ), $\mathcal{E}'$ as $\mathcal{E}_{L_\delta}$ (i.e. the set of canonicals for L_δ), X as $\mathbb{X}$ and we interpret the function symbol f as the constructor $(:)$ for lists, in the model $\perp\!\!\!\perp$ we have that:

- The truth value of $\Upsilon(z)$ is $\mathbb{Y}(z)$
- The sequent $\Gamma, z \in \mathcal{E}'(z) \vdash (\text{cons})yz \in \mathcal{E}'(fyz)$ is true in $\perp\!\!\!\perp$

Then, by correctness of formal deduction, we can conlcude that:

$$\lambda y\lambda g\lambda h(g)\lambda z(h)(\text{cons})yz \Vdash \forall y\forall z(\mathcal{E}_\delta(y),\mathbb{Y}(z) \to \mathbb{Y}(y{:}z))$$

on $\perp\!\!\!\perp$. Applying the storage operator for δ:

$$(\mathrm{T}_\delta)\lambda y\lambda g\lambda h(g)\lambda z(h)(\text{cons})yz \Vdash \forall y\forall z(\delta(y),\mathbb{Y}(z) \to \mathbb{Y}(y{:}z))$$

on $\perp\!\!\!\perp$. Since ζ is an abbreviation for $\lambda y\lambda g\lambda h(g)\lambda z(h)(\text{cons})yz$, we have proven the item (1) of the claim.

(2) Take a term $t \vdash \mathcal{E}_{L_\delta}(\underline{0}) \rightsquigarrow \mathbb{X}$ and a stack $\pi \in \mathbb{X}$. We must prove that $\lambda a(a)\underline{0} \star t.\pi \in \perp\!\!\!\perp$, but this process reduces to $t \star \underline{0}.\pi \in \perp\!\!\!\perp$, thus proving the item (2) of the claim.

(3) Since $f \Vdash \mathcal{E}_{L_\delta}(x) \rightsquigarrow \mathbb{X}$ and $\pi \in \mathbb{X}$, we have that $f.\pi \in (\mathcal{E}_{L_\delta}(x) \rightsquigarrow \mathbb{X}) \to \mathbb{X}$, truth value that we abbreviate as $\mathbb{Y}(x)$. We have proven the item (3) of the claim.

□

3.6. **Trees.**

Another recursive data type are the trees. Here we will introduce the binary trees built from a data type δ, which we will call *trees of* δ and denote as Tree_δ. These trees are built from data of type δ, adding left and right branches with a constructor fork : $\text{Tree}_\delta, \delta, \text{Tree}_\delta \to \text{Tree}_\delta$. Since we need to talk about trees of δ as well as about trees of Λ_c-terms, we introduce now a definition of trees built from an arbitrary set of constants and further we will specify the nature of elements interpreting the constants on a suitable structure. Take a set of constants C, the language of binary finite trees of C is defined as follows:

Definition 3.22.

$$t_1, t_2 ::= c \mid \text{fork}\, t_1 c t_2$$

where fork *is a* 3*-ary recursive constructor and c is a constant belonging to C. We call this language* the trees of *C and it will be denoted by* Tree_C

Furthermore, given a set S of elements taken from an structure, we will talk about *trees of S* by means of the following definition: we consider a set of constants C, one for each element of S. A tree of S is a pair $(t,\mathscr{Y})$ formed by a tree t of C and an injective interpretation $\mathscr{Y} : C \to S$ for the constants of C. Every time there is no confusion, we will omit the interpretation $\mathscr{Y}$, writing t instead of $(t,\mathscr{Y})$.

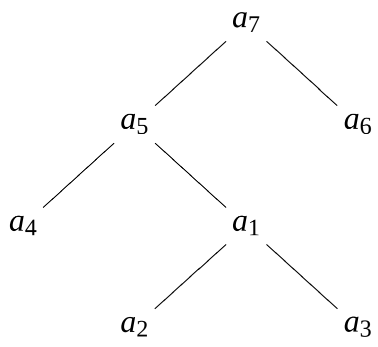
Definition 3.23. *Consider a set of constants C, two C-structures $\mathfrak{S}_1$ and $\mathfrak{S}_2$, a set A of individuals taken from $\mathfrak{S}_1$ and another set B of individuals taken from $\mathfrak{S}_2$. Take a function α from A to B. Each tree of A is a tree t together with an interpretation $\mathscr{Y}$ for the constants of C in the set A. Thus, $\alpha \circ \mathscr{Y}$ is an interpretation of C in B and hence $(t,\alpha\circ\mathscr{Y})$ is a tree of B. We denote this tree as $\alpha(t)$.*

Moreover, if the set A is finite, namely $A = \{a_1,\dots,a_k\}$, a tree of A can be denoted as $t[a_1,\dots,a_k]$ and the tree $\alpha(t)$ as $t[{}^{\alpha(a_1)\dots\alpha(a_k)}/_{a_1\dots a_k}]$ or simply as $t[\alpha(a_1),\dots,\alpha(a_k)]$.

As we did for lists, if we have a denumerable model $\mathfrak{M}$ of arithmetics, we can define a recursive coding for trees of individuals. In order to write simpler formulæ, given three individuals t_1, a and t_2, we adopt the parentheses notation $\{[t_1,a,t_2]\}$ for the code on $\mathfrak{M}$ of $\mathrm{fork}(t_1,a,t_2)$.

By instance, the tree

(3.23.1)



is coded in the model as $\{[\,\{[a_4,a_5,\{[a_2,a_1,a_3]\}\,]\}\,,a_7,a_6\,]\}$

Definition 3.24. *Given a data type δ, the formula* Tree_δ *is defined as follows:*

$$\mathrm{Tree}_\delta(x) := \forall X\,[\forall y\forall w\forall z(Xy,\delta(w),Xz \to X\{[y,w,z]\}\,),\forall y(\delta(y)\to Xy)\to Xx]$$

Remark 3.25. The formula $\mathrm{Tree}_\delta(x)$ means that the set of trees of δ is the smallest set containing δ and closed by the constructor fork, provided that the node added by fork satisfies δ.

We have chosen here a presentation without empty tree and hence trees have only filled leafs. This choice is as arbitrary as the one in which each branch is ended with the empty tree. However, this choice gives “smaller” trees, thus giving a presentation slightly different from the one given for lists.

In order to define a normalized representation for trees, as in the case of lists (c.f.: 3.14), we must define trees formed by terms and represent such a tree by a term. Given a tree of terms t, we will denote the associated normalized term as $\underline{t}$ and define it by induction on trees.

Definition 3.26. *Consider two Λ_c-variables f, x and $t \in \mathrm{Tree}_{\Lambda_c}$. we define by induction a term denoted by $\mathrm{st}_t^{f,x}$ as follows:*

- *If $t = a$ where a is a Λ_c-term, then $\mathrm{st}_t^{f,x} := (x)a$.*
- *If $t = \mathrm{fork}(t_1, a, t_2)$, where $t_1, t_2 \in \mathrm{Tree}_{\Lambda_c}$ and a is a Λ_c-term, then $\mathrm{st}_t^{f,x} := (f)\,\mathrm{st}_{t_1}^{f,x}\, a\, \mathrm{st}_{t_2}^{f,x}$.*

Given $t \in \mathrm{Tree}_{\Lambda_c}$, its normalized representation $\underline{t}$ is defined as $\lambda f \lambda x\, \mathrm{st}_t^{f,x}$

By instance, lets consider $t_1, \dots, t_7$ 7 Λ_c-terms and u the tree of the example 3.23.1. Hence, the normalized representation of $u[t_1, \dots, t_7]$ is:

$$\lambda f \lambda x((f)((f)(x)t_4)t_5((f)(x)t_2)t_1(x)t_3)t_7(x)t_6$$

Counterpart for trees of lemma 3.15 (for lists):

Lemma 3.27. *Consider a data type δ, a model $\perp\!\!\!\perp$, k individuals $a_1, \dots, a_k$, $u \in \mathrm{Tree}_{\{a_1,\dots,a_k\}}$ and k Λ_c-terms $t_1, \dots, t_k$ such that $t_i \Vdash \delta(a_i)$ for all $i \in [1..k]$. Then, $\underline{u[t_1, \dots, t_k]} \Vdash \mathrm{Tree}_\delta(u)$.*

Proof. Exercise. It can be proven by induction on trees, similarly to the one of 3.15. □

The Λ_c-term that computes the fork constructor on Λ_c-terms can be defined as:

Definition 3.28. $\mathrm{fork} := \lambda t \lambda a \lambda u \lambda f \lambda x((f)(t)fx)a(u)fx.$

Lemma 3.29. $\mathrm{fork} \Vdash \forall x \forall y \forall z(\mathrm{Tree}_\delta(x), \delta(y), \mathrm{Tree}_\delta(z) \to \mathrm{Tree}_\delta(\{[x,y,z\}))$

Proof. Is similar to the proof of 3.17. □

Definition 3.30. *Take $n_1, \dots, n_k$, data of type δ (i.e.: $\forall i \in [1..k]\mathcal{E}_\delta(n_i) \neq \emptyset$). Consider a tree $u \in \mathrm{Tree}_{\{n_1,\dots,n_k\}}$. Denote as e_{n_i} the canonical representation of n_i in δ. We define the canonical representation $\bar{u}$ of u by induction on trees:*

- *If $u = n_i$, then $\bar{u} := \lambda f \lambda x(x)e_{n_i}$*
- *If $u = \mathrm{fork}(u_1, n_j, u_2)$ for two trees u_1, u_2; then $\bar{u} := (\mathrm{fork})\bar{u}_1 e_{n_j} \bar{u}_2$.*

For all trees $v \in \mathrm{Tree}_{\{n_1,\dots,n_k\}}$, we define $\mathcal{E}_{\mathrm{Tree}_\delta}(v)$ as $\{\bar{v}\}$.

By instance, if u is the tree represented in 3.23.1, then $\bar{u}$ is the term:

$$((\mathrm{fork})((\mathrm{fork})\bar{a}_4 e_{a_5})(\mathrm{fork})\bar{a}_2 e_{a_1} \bar{a}_3)e_{a_7}\bar{a}_6$$

where $\bar{a}_j$ is (by definition) the term $\lambda f \lambda x(x)e_{a_j}$.

Lemma 3.31. *Take $n_1,\dots,n_k$ data of type δ (i.e.: $\forall i \in [1..k]\mathcal{E}_\delta(n_i) \neq \emptyset$). Consider a tree $u \in \mathrm{Tree}_{\{n_1,\dots,n_k\}}$.*
Then $\bar{u} \Vdash \mathrm{Tree}_\delta(u)$.

Proof. Similarly as in 3.18, the result is a direct consequence of 3.29 □

We prove now that Tree_δ is a data type, provided that δ is also a data type. In order to do this, we must define a storage operator for the trees of δ and after that, we must prove that it is a storage operator.

Definition 3.32. *Given a data type δ and a storage operator T_δ for δ, lets denote as ς the Λ_c-term $\lambda f(\mathrm{T}_\delta)\lambda z\lambda g\lambda h(f)\lambda y(g)\lambda w(h)(\mathrm{fork})yzw$ and as ρ the Λ_c-term $\lambda a\lambda b(b)\lambda f\lambda x(x)a$. We define a storage operator for Tree_δ by*

$$\mathrm{T}_{\mathrm{Tree}_\delta} := \lambda f\lambda t((t)\varsigma(\mathrm{T}_\delta)\rho)f$$

Lemma 3.33. $\mathrm{T}_{\mathrm{Tree}_\delta} \Vdash \forall X\forall x[\mathcal{E}_{\mathrm{Tree}_\delta}(x) \rightsquigarrow X, \mathrm{Tree}_\delta(x) \to X]$

Proof. The proof is similar to the one of 3.21. Consider a model $\perp\!\!\!\perp$, a 1-ary second order parameter $\mathbb{X}$ and an individual x. Take two Λ_c-terms f,t and a stack π such that $f \Vdash \mathcal{E}_{\mathrm{Tree}_\delta}(x) \rightsquigarrow \mathbb{X}$, $t \Vdash \mathrm{Tree}_\delta(x)$ on $\perp\!\!\!\perp$ and $\pi \in \mathbb{X}$. We must prove that $\mathrm{T}_{\mathrm{Tree}_\delta} \star f.t.\pi \in \perp\!\!\!\perp$. To prove this, it suffices to prove that $t \star \varsigma.(\mathrm{T}_\delta)\rho.f.\pi \in \perp\!\!\!\perp$.

Let us define the predicate $\mathbb{Y}(z) := (\mathcal{E}_{\mathrm{Tree}_\delta}(z) \rightsquigarrow \mathbb{X}) \to \mathbb{X}$. Hence:

$$t \Vdash \forall y\forall z\forall w(\mathbb{Y}(y), \delta(z), \mathbb{Y}(\{[y,z,w]\})), \forall(\delta(y) \to \mathbb{Y}(y)) \to \mathbb{Y}(x)$$

Then, we must prove the following **claim**:

(1) $\varsigma \Vdash \forall y\forall z\forall w(\mathbb{Y}(y), \delta(z), \mathbb{Y}(\{[y,z,w]\}))$
(2) $(\mathrm{T}_\delta)\rho \Vdash \forall y(\delta(y) \to \mathbb{Y}(y))$
(3) $f.\pi \in \mathbb{Y}(x)$

(1) Define the sequents

$$\Gamma := f : \Upsilon(y), z \in \mathcal{E}(z), g : \Upsilon(w), h : \mathcal{E}'(fyzw) \rightsquigarrow X$$

$$\Delta := \Gamma, y \in \mathcal{E}'(y), w \in \mathcal{E}'(w)$$

where $\mathcal{E}, \mathcal{E}'$ are two 1-ary Λ_c-set variables, f is a 3-ary function symbol, $\Upsilon(z) := (\mathcal{E}'(z) \rightsquigarrow X) \to X$ and X is a 0-ary second order variable. Let us consider the following formal proof:

$$\dfrac{\dfrac{\Delta \vdash (\mathrm{fork})yzw : \mathcal{E}'(fyzw) \quad \dfrac{}{\Delta \vdash h : \mathcal{E}'(fyzw) \rightsquigarrow X}}{\Delta \vdash (h)(\mathrm{fork})yzw : X}}{\Gamma, y \in \mathcal{E}'(y) \vdash \lambda w(h)(\mathrm{fork})yzw : \mathcal{E}'(w) \rightsquigarrow X}$$

Since $\dfrac{}{\Delta \vdash g : (\mathcal{E}'(w) \rightsquigarrow X) \to X}$ we have:

$$\dfrac{\dfrac{}{\Gamma, y \in \mathcal{E}'(y) \vdash (g)\lambda w(h)(\mathrm{fork})yzw : X}}{\Gamma \vdash \lambda y(g)\lambda w(h)(\mathrm{fork})yzw : \mathcal{E}'(y) \rightsquigarrow X}$$

Because $\dfrac{}{\Gamma \vdash f : (\mathcal{E}'(y) \rightsquigarrow X) \to X}$ we can prove:

$$\dfrac{\dfrac{}{\Gamma \vdash (f)\lambda y(g)\lambda w(h)(\text{fork})yzw : X}}{f : \Upsilon(y) \vdash \lambda z\lambda g\lambda h(f)\lambda y(g)\lambda w(h)(\text{fork})yzw : \mathcal{E}(z); \Upsilon(w) \to \Upsilon(fyzw)}$$

Let us assign the variables $\mathcal{E}$ as $\mathcal{E}_\delta$, $\mathcal{E}'$ as $\mathcal{E}_{\text{Tree}_\delta}$ and X as $\mathbb{X}$. Interpret the function symbol f as the $'$fork$'$ constructor coded in the model (i.e.: $\{[\_,\_,\_\}$). Then, in $\perp\!\!\!\perp$ we have:

- The truth value of $\Upsilon(z)$ is $\mathbb{Y}(z)$
- The sequent $\Delta \vdash (\text{fork})yzw : \mathcal{E}'(fyzw)$ is true in $\perp\!\!\!\perp$.

Then, by correctness, we have that

$$f : \Upsilon(y) \vdash \lambda z\lambda g\lambda h(f)\lambda y(g)\lambda w(h)(\text{fork})yzw : \mathcal{E}(z); \Upsilon(w) \to \Upsilon(fyzw)$$

is true in $\perp\!\!\!\perp$. By storage operators definition, we have that

$$f : \Upsilon(y) \vdash (\mathrm{T}_\delta)\lambda z\lambda g\lambda h(f)\lambda y(g)\lambda w(h)(\text{fork})yzw : \delta(z); \Upsilon(w) \to \Upsilon(fyzw)$$

is true in $\perp\!\!\!\perp$ and hence

$$\lambda f(\mathrm{T}_\delta)\lambda z\lambda g\lambda h(f)\lambda y(g)\lambda w(h)(\text{fork})yzw \Vdash \forall y \forall z \forall w(\mathbb{Y}(y), \delta(z), \mathbb{Y}(w) \to \mathbb{Y}(\{[y,z,w\}))$$

thus proving the claim item (1).

(2) Let us consider e_y as the canonical representation of y as a data of type δ and $u.\pi \in \mathbb{Y}(y)$, i.e.: $u \Vdash \mathcal{E}_{\text{Tree}_\delta}(y) \rightsquigarrow \mathbb{X}$ and $\pi \in \mathbb{X}$. Let us consider the reduction $\rho \star e_y.u.\pi \succ u \star \lambda f\lambda x(x)e_y.\pi$. By definition of $\mathcal{E}_{\text{Tree}_\delta}(y)$, we have that $\lambda f\lambda x(x)e_y.\pi \in \mathcal{E}_{\text{Tree}_\delta}(y) \rightsquigarrow \mathbb{X}$. Hence $u \star \lambda f\lambda x(x)e_y.\pi \in \perp\!\!\!\perp$, thus concluding that $\rho \star e_y.u.\pi \in \perp\!\!\!\perp$. By storage operators definition, the claim item (2) is proven.

(3) Such as the term f was chosen, $f \Vdash \mathcal{E}_{\text{Tree}_\delta}(x) \rightsquigarrow \mathbb{X}$ and in consequence $f.\pi \in (\mathcal{E}_{\text{Tree}_\delta}(x) \rightsquigarrow \mathbb{X}) \to \mathbb{X}$, thus proving the claim item (3).

Since the claim is proven, the result is too. □

3.7. **Canonical representations for recursive data types.**

As the reader has certainly overviewed, the canonical representations we chosen for positive integers, non empty lists and trees are not normalized proof-like terms. On the other hand, canonical representations for booleans are chosen –as usually– as normalized $\lambda-$terms. The following theorem proves that *it is impossible to choose normalized canonical representations for lists*. It is easily generalized to *"recursive"* data types. Indeed, the reader can check that all technical tools used in this proof suffices to prove the theorem for any recursive data type.

Theorem 3.34. *Let us consider a data type $\delta(x)$ with canonical. Then, the canonical representations of* $\mathrm{L}\,\delta$ *are not all normalized proof-like terms.*

Proof. By 3.3, for all $\phi \Vdash \mathrm{L}_\varphi(n)$, we have that $(\mathrm{T}_{\mathrm{L}_\delta})\,\mathrm{id}\,\phi \star \pi \succ e_n \star \pi$, where e_n is the canonical representation of n as a data of type L_δ. Define the set of k-lists $\mathrm{L}_k := \{\lambda f \lambda x(f)t_1,\dots,(f)t_k x \mid t_1,\dots,t_k \in \Lambda_c\}$. Pick a data n of type δ and consider d_n the normalized form of the canonical representation of n. The term $l_k = \underbrace{(\mathrm{cons})d_n(\mathrm{cons})d_n\dots(\mathrm{cons})d_n}_{k}\overline{0}$ is a universal representation of $\langle\underbrace{n,\dots,n}_{k}\rangle$ as a list of δ and its normalized form is the term $g_k = \lambda f\lambda x\underbrace{(f)d_n(f)d_n\dots(f)d_n}_{k}x$. Let us suppose we have a normalized canonical representation for L_δ. Then we can assert that

$$(\mathrm{T}_{\mathrm{L}_\delta})\,\mathrm{id}\,l_k \succ g_k \text{ for all } k \tag{3.34.2}$$

We will prove that there is a k from rich it is impossible.

Lemma 3.35. *For each integer k and for all Λ_c−terms t,u such that u is closed and for all variable y, if $t[^u/_y]$ contains a k-list as a subterm, then t or u contains a k-list as a subterm.*

Proof. We prove it by induction on t. If $y \notin \mathrm{FV}(t)$, we have nothing to do. If not, we have three cases:

- Suppose t is the variable y. Then $t[^u/_y] = u$ and we have the result.
- Suppose t is an application $(t_1)t_2$. Then $t[^u/_y] = (t_1[^u/_y])t_2[^u/_y]$. Since this term is an application, $l \neq t[^u/_y]$ and hence l is a subterm of $t_1[^u/_y]$ or $t_2[^u/_y]$. Applying the induction hypothesis, we obtain the result.
- Suppose t is an abstraction λzv. Then $t[^u/_y]$ is the term $\lambda zv[^u/_y]$. There are two possibilities:
  1. l is a subterm of $v[^u/_y]$. By induction hypothesis, we obtain the result.
  2. $l = t[^u/_y] = \lambda zv[^u/_y]$. We can rename the variable z as f. The term $v[^u/_y]$ is in fact $\lambda x(f)t_1\dots(f)t_k x$, where $t_1,\dots,t_k$ are Λ_c-terms. Since u is a closed term and the substitution $[^u/_y]$ is effective over v (because $y \in \mathrm{FV}(v)$), then u must be a closed subterm of $v[^u/_y]$. The only possible closed subterms of $v[^u/_y]$ are $t_1,\dots,t_k$. Although it is possible that there are several t_i which are equals to u, we can suppose that $t_1 = u$ without loss of generality. In this case, $t = \lambda f\lambda x(f)y(f)t_2,\dots,(f)t_k x$, which is itself a k-list.

□

Now, we proceed as follows. Consider a closed Λ_c-term p that reduces by weak head reduction to a k-list. Each step of reduction is of the form

$$(\lambda yt)u_1\dots u_h \succ t[^{u_1}/_y]u_2\dots u_h$$

for some terms $t,u_1,\dots,u_h$. If we know that $t[^{u_1}/_y]u_2\dots u_h$ contains a k-list as a subterm, then applying 3.35, one of the terms $t,u_1,\dots,u_h$ contains a

k-list as a subterm and in consequence $(\lambda yt)u_1 \ldots u_h$ also contains a k-list as a subterm. Now, we apply this property to our equation 3.34.2. For each integer k, the term $(\mathrm{T}_\delta)\,\mathrm{id}\,l_k$ must contain a k-list. Consider k such that $2k+5$ is greater than the maximum of the lengths of d_n, id and $\mathrm{T}_{\mathrm{L}_\delta}$. For these k, a k-list must be a subterm of $\mathrm{T}_{\mathrm{L}_\delta}$, d_k or id. But k-lists are too wide for it. □

The reason why we cannot get normalized canonical representations is because of the use of weak head reduction. It is in fact the deep reason for defining storage operators. If every β-reduction would be allowed, the identity would always be a storage operator and canonical representations would be normalized. But the reduction would not be sequential; so it would be impossible to use interactive instructions which are incompatible with β-reduction. Such instructions are essential for specification of valid formulæ.

4. **The threads method**

4.1. **Definitions and first applications.**

Take a process P and consider its thread th_P (c.f.1.22). By definition, it is a set of processes closed by reduction and hence its complement th_P^c is a saturated set. More generally, given a set of processes $S \subseteq \Lambda_0 \times \Pi$, the set $\bigcup_{P \in S} \mathrm{th}_P$ is also closed by reduction and its complement $\bigcap_{P \in S} \mathrm{th}_P^c$ is a saturated set.

Definition 4.1. *We say that a model* $\perp\!\!\!\perp$ *is a* threads model *if and only if there is a finite set of processes S such that* $\perp\!\!\!\perp = \bigcap_{P \in S} \mathrm{th}_P^c$.

Remark 4.2. If we allow an infinite set of threads in 4.1 definition, each saturated set $\perp\!\!\!\perp$ is a threads model because we have $\perp\!\!\!\perp = \bigcap_{P \in \perp\!\!\!\perp^c} \mathrm{th}_P$.

Threads models will be used in this work to study reduction properties. The main ideas from which this definition will be exploitable are the following: Suppose we can prove that a process Q does not belong to a threads model $\perp\!\!\!\perp = \bigcap_{P \in S} \mathrm{th}_P^c$. Then, we know that Q belongs to at least one thread th_P (where $P \in S$). Usually the list $(\mathrm{th}_P)_{P \in S}$ will be inductively defined. At each step, we will collect some information about the list of threads, thus determining which threads of $(\mathrm{th}_P)_{P \in S}$ can contain Q. This model is used as a tool in order to study the execution of a given process; the fact that $Q \notin \perp\!\!\!\perp$ means that, during the execution, some term Q' obtained from Q by substitution.

The following lemma shows that no thread model is coherent:

Lemma 4.3.

(1) *Consider a thread* th_P*, the support of which is infinite. Then, for all* $n \in \mathbb{N}$*, the term* $(\delta)\delta\overline{n}$ *–where* $\delta = \lambda x(x)x$*– does not appear in head position in* th_P.

(2) *For all thread models* $\perp\!\!\!\perp$ *there is an integer n such that* $(\delta)\delta\overline{n} \Vdash_{\perp\!\!\!\perp} \perp$

Proof.

(1) If $P \succ (\delta)\delta\overline{n} \star \rho$, then th_P ends up with the process $\delta \star \delta.\overline{n}.\rho$ and hence its support is finite.

(2) Let us consider a threads model $\perp\!\!\!\perp = \bigcap_{P \in S} \mathrm{th}_P^c$. We can split $\perp\!\!\!\perp^c$ into $(\bigcup_{P \in S_1} \mathrm{th}_P) \cup (\bigcup_{P \in S_2} \mathrm{th}_P)$, where

$$S_1 = \{P \in S \mid \mathrm{th}_P \text{ has a finite support}\}$$

and $S_2 = S \backslash S_1$. Since S is finite, S_1 is also finite and then there is an integer n such that $(\delta)\delta\overline{n}$ does not appears in head position in $\bigcup_{P \in S_1} \mathrm{th}_P$. But, applying (1) we can conclude that this term does not appears in head position in $\bigcup_{P \in S_2} \mathrm{th}_P$. In consequence, for all stacks ρ we have $(\delta)\delta\overline{n} \star \rho \in \perp\!\!\!\perp$, thus proving the result.

□

Suppose we have a ground model $\mathfrak{M} = (M, \mathcal{D}, \mathscr{Y})$, and a Λ_0-term t realizing in all models a given formula $\overset{\delta}{\exists} x \varphi(x)$, where δ is a data type. Suppose that $\mathcal{E}$ is the canonical representation of δ. Take a Λ_c-term h, a stack π and define the realizability model

$$\perp\!\!\!\perp := \{P \in \Lambda_0 \times \Pi \mid P \succcurlyeq h \star e_n.\xi.\pi, \text{ where } n{\in}M,\ e_n{\in}\mathcal{E}(n) \text{ and } \xi{\in}\Lambda_0\}$$

In this model, $h \Vdash \forall x[\mathcal{E}(x), \top \to \{\pi\}]$ and then, by definition of storage operators, $\mathrm{T}_\delta h \Vdash \forall x[\delta(x), \top \to \{\pi\}]$. In particular, $\mathrm{T}_\delta h \Vdash \overset{\delta}{\forall} x(\varphi(x) \to \{\pi\})$, thus concluding that $\mathrm{T}_\delta h.\pi \in ||\overset{\delta}{\exists} x \varphi(x)||$. Then $t \star \mathrm{T}_\delta h.\pi \in \perp\!\!\!\perp$ and hence there is a data n of type δ –which canonical representation is $e_n \in \mathcal{E}(n)$– and a Λ_0-term ξ such that $t \star \mathrm{T}_\delta h.\pi \succ h \star e_n.\xi.\pi$.

It is interesting information but in order to compose strategies, we need more: specifically, answers to the following questions:

(1) Will $\mathrm{T}_\delta h$ arrive many times in head position before the execution reaches the process $h \star e_n.\xi.\pi$? If it happens, can we find some information about the environment (stack) each time $\mathrm{T}_\delta h$ arrives in head position?
(2) Does e_n depend on h or π? Notice that the model $\perp\!\!\!\perp$ does depend on h and π.
(3) If $\mathrm{T}_\delta h$ arrives many times in head position, can we replace $\mathrm{T}_\delta h$ with some term in such a way that we can foresee the result? What are the conditions for such a term?

For simplicity, in the following, we will answer these questions first for the case of the data type $\mathrm{int}(x)$; further we will answer the same questions for an arbitrary product of data types.

Lemma 4.4. *Take an extended formula $\overset{\mathrm{int}}{\exists} x \varphi(x)$. Suppose that u is a Λ_c-term such that, for all terms ν and ξ and for all stacks π, there is a Λ_c-term $h_{\nu.\xi.\pi}$ and a stack $\rho_{\nu.\xi.\pi}$ verifying $u \star \nu.\xi.\pi \succ \mathrm{T}_{\mathrm{int}} \star h_{\nu.\xi.\pi}.\nu.\rho_{\nu.\xi.\pi}$. Consider a Λ_c-term t, a stack π and a model $\perp\!\!\!\perp_0$ satisfying $\perp\!\!\!\perp_0 \subseteq (\mathrm{th}_{t\star u.\pi})^c$ and $t \Vdash_0 \overset{\mathrm{int}}{\exists} x \varphi(x)$.*

Then, there is an integer n and two Λ_c-terms ν and ξ such that:

(1) $\nu \Vdash_0 \mathrm{int}(n)$
(2) $\xi \Vdash_0 \varphi(n)$
(3) $u \star \nu.\xi.\pi \notin \perp\!\!\!\perp_0$ *and* $h_{\nu.\xi.\pi} \star \bar{n}.\rho_{\nu.\xi.\pi} \notin \perp\!\!\!\perp_0$.

Proof. We have $u \not\Vdash_0 \overset{\mathrm{int}}{\forall} x(\varphi(x) \to \{\pi\})$ because $t \Vdash_0 \overset{\mathrm{int}}{\forall} x(\varphi(x) \to \{\pi\}) \to \{\pi\}$ and $t \star u.\pi \notin \perp\!\!\!\perp_0$. Then, by definition of realizability, there is an integer n and two Λ_c-terms ν and ξ satisfying $\nu \Vdash_0 \mathrm{int}(n)$, $\xi \Vdash_0 \varphi(x)$ and $u \star \nu.\xi.\pi \notin \perp\!\!\!\perp_0$.

Since $u \star \nu.\xi.\pi \succ \mathrm{T}_{\mathrm{int}} \star h_{\nu.\xi.\pi}.\nu.\rho_{\nu.\xi.\pi}$ and the storage operator realizes $\forall x[(\{\bar{x}\} \rightsquigarrow \{\rho_{\nu.\xi.\pi}\}), \mathrm{int}(x) \to \{\rho_{\nu.\xi.\pi}\}]$, the Λ_c-term $h_{\nu.\xi.\pi}$ does not realizes in $\perp\!\!\!\perp_0$ the extended formula $\{\bar{n}\} \rightsquigarrow \{\rho_{\nu.\xi.\pi}\}$. Hence, $h_{\nu.\xi.\pi} \star \bar{n}.\rho_{\nu.\xi.\pi} \notin \perp\!\!\!\perp_0$. □

The above result is valid in general for all reduction systems. However, if we know that the reduction system preserves substitution of constants, we can argue by substitution to give a more precise description of the $t\star u.\pi$ thread. First, we substitute the terms u and $h_{\nu.\xi.\pi}$ by instructions and the stacks $\rho_{\nu.\xi.\pi}$ by Π-constants, thus obtaining the following result.

Lemma 4.5. *Take a Π-constant π_0, a storage operator for integers $\mathrm{T_{int}}$ and a Λ_c-term t. Then, for all instructions U verifying:*

i. *t does not contain U.*
ii. *for all Λ_c-terms ν and ξ, there is a halt instruction $H_{\nu.\xi}$ and a Π-constant $\pi'_{\nu.\xi}$ verifying $U\star\nu.\xi.\pi_0 \succ \mathrm{T_{int}}\star H_{\nu.\xi}.\nu.\pi'_{\nu.\xi}$.*
iii. *A process containing U in head position reduces only if its stack is of the form $\nu.\xi.\pi_0$ for two Λ_c-terms ν and ξ.*
iv. *$t \Vdash_0 \overset{\mathrm{int}}{\exists}_x \top$ where $\perp\!\!\!\perp_0 := \mathrm{th}^c_{t\star U.\pi_0}$*

there are three integers n,k and $i\le k$ such that:

(1) *k is the number of times that U arrives in head position in $\mathrm{th}_{t\star U.\pi_0}$.*
(2) *There are $2k$ Λ_c-terms $\nu_1,\dots,\nu_k,\xi_1,\dots,\xi_k$ depending on U and π_0 such that the reduction of $t\star U.\pi_0$ is as follows:*

$$\begin{array}{rll} t\star U.\pi_0 & \succ U\star\nu_1.\xi_1.\pi_0 & \succ \dots \\ & \vdots & \vdots \\ & \succ U\star\nu_i.\xi_i.\pi_0 & \succ\dots \\ & \vdots & \vdots \\ & \succ U\star\nu_k.\xi_k.\pi_0 & \succ H_{\nu_i.\xi_i}\star\bar{n}.\pi'_{\nu_i.\xi_i} \end{array} \tag{4.5.3}$$

Proof. Take an instruction U satisfying i., ii., iii. and iv. By 4.4 there is an integer n and two Λ_c-terms ν and ξ such that $\nu \Vdash_0 \mathrm{int}(n)$, $\xi \Vdash_0 \top$[4] and the processes $U\star\nu.\xi.\pi_0$ and $H_{\nu.\xi}\star\bar{n}.\pi'_{\nu.\xi}$ belongs to $\mathrm{th}_{t\star U.\pi_0}$. Since $H_{\nu.\xi}$ is a halt instruction, the process $H_{\nu.\xi}\star\bar{n}.\pi'_{\nu.\xi}$ is at the end of the thread $\mathrm{th}_{t\star U.\pi_0}$ and hence $\mathrm{th}_{t\star U.\pi_0}$ is a finite thread. Define k as the number of times that U arrives in head position. By iii., a process containing U in head position reduces only if its stack is of the form $\nu.\xi.\pi_0$ for two terms ν and ξ. Then, there is a finite sequence $\nu_1,\dots,\nu_k,\xi_1,\dots,\xi_k$ of Λ_c-terms such that the j-th time that U arrives in head position , U has the stack $\nu_j.\xi_j.\pi_0$ as argument. Since $U\star\nu.\xi.\pi_0$ is in the thread of $t\star U.\pi_0$, there is an integer i between 1 and k such that $\nu=\nu_i$ and $\xi=\xi_i$. Finally, this implies that $H_{\nu_i.\xi_i}\star\bar{n}.\pi'_{\nu_i.\xi_i}$ is at the end of $\mathrm{th}_{t\star U.\pi_0}$. □

Remark 4.6. Since U does not use its second argument, in order to compute the integer n, the process $t\star U.\pi$ uses only the first argument of U. In fact each Λ_c-term ν_j contains the information required to compute n.

A scheme as 4.5.3 containing threads will be called a *threads scheme*. If we substitute the instruction U by a term u satisfying the property ii. of 4.5,

[4]it is true for all terms ξ

we obtain the same integer n. Moreover, the term u arrives in head position the same k times that U in the threads scheme 4.5.3.

Remark 4.7. We will use the result 4.5 as follows:
If a term t realizes an extended formula $\overset{\text{int}}{\exists} x \varphi(x)$ in all model $\perp\!\!\!\perp$, then t realizes the formula $\overset{\text{int}}{\exists} x \top$. Hence, 4.5 is applicable for such a term.

We can generalize these properties for a product of arbitrary data types:

Lemma 4.8. *Consider $\delta_1,\dots,\delta_r$ r-data types and $\mathcal{E}_1,\dots,\mathcal{E}_r$ their respective canonical representations. Take an extended formula $\overset{\delta_1}{\exists} x_1 \dots \overset{\delta_r}{\exists} x_r (\varphi_1,\dots,\varphi_s)$. Suppose that u is a Λ_c-term such that, for all terms $\nu_1,\dots,\nu_r,\xi_1,\dots,\xi_s$ and for all stacks π, there is a Λ_c-term $h_{\vec{\nu}\vec{\xi}.\pi}$ and a stack $\rho_{\vec{\nu}\vec{\xi}.\pi}$ satisfying:*

$$u \star \nu_1 \dots \nu_r.\xi_1 \dots \xi_s.\pi \succ \mathrm{T}_{\delta_1,\dots,\delta_r} \star h_{\vec{\nu}\vec{\xi}.\pi}.\nu_1 \dots \nu_r.\rho_{\vec{\nu}\vec{\xi}.\pi}$$

for a storage operator $\mathrm{T}_{\delta_1\dots\delta_k}$ defined as in 3.11. Consider a Λ_c-term t and a model $\perp\!\!\!\perp_0$ such that $t \Vdash_0 \overset{\delta_1}{\exists} x_1 \dots \overset{\delta_r}{\exists} x_r (\varphi_1,\dots,\varphi_s)$ and $\perp\!\!\!\perp_0 \subseteq \mathrm{th}^c_{t\star u.\pi}$.
Then, there are r integers $n_1,\dots,n_r$, r Λ_c-terms $\nu_1,\dots,\nu_r$ and s Λ_c-terms $\xi_1,\dots,\xi_s$ such that:

(1) *for all $i\in[1..r]$ $\nu_i \Vdash_0 \delta(n_i)$*
(2) *for all $j\in[1..s]$ $\xi_j \Vdash_0 \varphi_j(\vec{n})$*
(3) *$u\star\nu_1\dots\nu_r.\xi_1\dots\xi_s.\pi \notin \perp\!\!\!\perp_0$ and $h_{\vec{\nu}\vec{\xi}.\pi} \star e_1^{(n_1)} \dots e_r^{(n_r)}.\rho_{\vec{\nu}\vec{\xi}.\pi} \notin \perp\!\!\!\perp_0$*

where we denote by $e_i^{(m)}$ the only one element belonging to $\mathcal{E}_i(m)$, whenever it exists.

Proof. The proof is almost similar to the one of 4.8. We repeat the argument: Since $t \Vdash_0 \forall\vec{x}[\delta_1(x_1)\dots\delta_r(x_r),\varphi_1(\vec{x}),\dots,\varphi_s(\vec{x}) \to \{\pi\}] \to \{\pi\}$ and $t\star u.\pi \notin \perp\!\!\!\perp_0$, then $u \not\Vdash_0 \forall\vec{x}[\delta_1(x_1),\dots,\delta_r(x_r),\varphi_1(\vec{x}),\dots,\varphi_s(\vec{x})\to\{\pi\}]$. Hence, there are r integers $n_1,\dots,n_r$, r terms $\nu_1,\dots,\nu_r$ and s terms $\xi_1,\dots,\xi_s$ such that for all $i \in [1..r]$ $\nu_i \Vdash_0 \delta(n_i)$, for all $j \in [1..s]$ $\xi_j \Vdash_0 \varphi_j(\vec{n})$ and $u\star \nu_1\dots\nu_r.\xi_1\dots\xi_s.\pi \notin \perp\!\!\!\perp_0$. Since by hypothesis $u\star\nu_1\dots\nu_r.\xi_1\dots\xi_s\pi \succ \mathrm{T}_{\delta_1\dots\delta_r}\star h_{\vec{\nu}\vec{\xi}.\pi}.\nu_1\dots\nu_r.\rho_{\vec{\nu}\vec{\xi}.\pi}$, then $\mathrm{T}_{\delta_1\dots\delta_r}\star h_{\vec{\nu}\vec{\xi}.\pi}.\nu_1\dots\nu_r.\rho_{\vec{\nu}\vec{\xi}.\pi} \notin \perp\!\!\!\perp_0$. By definition, $\mathrm{T}_{\delta_1\dots\delta_r} \Vdash_0 \mathcal{E}(n_1);\dots;\mathcal{E}(n_r) \rightsquigarrow \{\rho_{\vec{\nu}\vec{\xi}.\pi}\},\delta_1(n_1),\dots,\delta_r(n_r) \to \{\rho_{\vec{\nu}\vec{\xi}.\pi}\}$; because of that $h_{\vec{\nu}\vec{\xi}.\pi} \not\Vdash_0 \mathcal{E}(n_1);\dots;\mathcal{E}(n_r) \to \{\rho_{\vec{\nu}\vec{\xi}.\pi}\}$ and in consequence $h_{\vec{\nu}.\vec{\xi}.\pi}\star e_1^{(n_1)}\dots e_r^{(n_r)}.\rho_{\vec{\nu}\vec{\xi}.\pi} \notin \perp\!\!\!\perp_0$, thus proving the result. □

Lemma 4.9. *Consider a Π-constant π_0, r data types $\delta_1,\dots,\delta_r$, a storage operator (corresponding to $\delta_1\dots\delta_r$) $\mathrm{T}_{\delta_1\dots\delta_r}$ and a Λ_c-term t.*
Then, for all instructions U verifying:

i. *t does not contain U*
ii. *for all terms $\nu_1,\dots,\nu_r,\xi_1,\dots,\xi_s$, there is a halt instruction $H_{\vec{nu}\vec{\xi}}$ and a constant stack $\pi'_{\vec{\nu}\vec{\xi}}$ such that:*

$$U\star\nu_1\dots\nu_r.\xi_1\dots\xi_s.\pi_0 \succ \mathrm{T}_{\delta_1\dots\delta_r}\star H_{\vec{\nu}\vec{\xi}}.\nu_1\dots\nu_r.\pi'_{\vec{\nu}\vec{\xi}}$$

iii. *A process containing U in head position reduces only if its stack is $\nu_1 \dots \nu_r.\xi_1 \dots \xi_s.\pi_0$ for $r+s$ suitable terms $\nu_1,\dots,\nu_r,\xi_1,\dots,\xi_s$.*

iv. $t \Vdash_0 \overset{\delta_1}{\exists} x_1 \dots \overset{\delta_r}{\exists} x_r (\underbrace{\top,\dots,\top}_{s})$ *where* $\perp\!\!\!\perp_0 := \mathrm{th}^c_{t\star U.\pi_0}$

there are $r+2$ integers $k, i \le k, n_1, \dots n_r$ such that:

(1) *k is the number of times that U arrives in head position in* $\mathrm{th}_{t\star U.\pi_0}$
(2) *There are $k(r+s)$ Λ_c-terms $(\nu_{i1},\dots,\nu_{ir},\xi_{i1},\dots,\xi_{is})_{i\in[1..k]}$ such that the reduction of $t \star U.\pi_0$ is as follows:*

(4.9.4)
$$\begin{array}{lll}
t\star U.\pi_0 & \succ U\star \nu_{11}\dots\nu_{1r}.\xi_{11}\dots\xi_{1s}.\pi_0 \succ & \dots \\
 & \vdots & \vdots \\
 & \succ U\star \nu_{i1}\dots\nu_{ir}.\xi_{i1}\dots\xi_{is}.\pi_0 \succ & \dots \\
 & \vdots & \vdots \\
 & \succ U\star \nu_{k1}\dots\nu_{kr}.\xi_{k1}\dots\xi_{ks}.\pi_0 \succ & H_{\vec{\nu}_i.\vec{\xi}_i}\star e_1^{(n_1)}\dots e_r^{(n_r)}.\pi'_{\vec{\nu}_i\vec{\xi}_i}
\end{array}$$

Proof. By 4.8, there are r integers $n_1,\dots,n_r$ and $r+s$ Λ_c-terms $\nu_1,\dots,\nu_r$, $\xi_1,\dots,\xi_s$ such that $H_{\vec{\nu}\vec{\xi}}\star e_1^{(n_1)}\dots e_r^{(n_r)}.\pi'_{\vec{\nu}\vec{\xi}} \notin \perp\!\!\!\perp_0$. Since $H_{\vec{\nu}\vec{\xi}}$ is a halt instruction, the process $H_{\vec{\nu}\vec{\xi}}\star e_1^{(n_1)}\dots e_r^{(n_r)}.\pi'_{\vec{\nu}\vec{\xi}}$ is at the end of $\mathrm{th}_{t\star U.\pi}$ and then $\mathrm{th}_{t\star U.\pi_0}$ is a finite thread. Define k as the number of times that U arrives in head position in this thread. Since U reduces only with a stack of length $\ell(\pi_0)+k+s$ ended by π_0, there are kr terms $(\nu_{ij})_{i\in[1..k]j\in[1..r]}$ and sr terms $(\xi_{ij})_{i\in[1..k],j\in[1..r]}$ verifying 4.9.4 □

4.2. **The dynamic substitution.**

Let us consider a reduction system that respects substitution of constants. The main idea around dynamic substitution is to put in a given process P, instructions having a specific behaviour which is common for a suitable family of Λ_c-terms. This allows to describe the behaviour of P when we replace these instructions by terms in its associated family. We will distinguish the instructions put to realize axioms –like cc and '`quote`'– of the instructions put to be substituted –instructions like U in 4.5 and 4.9 and halt instructions generally denoted as H–; which will be informally called "substitutable". The main example of dynamic substitution that we will use further is the following:

Example 4.10. Let us consider an instruction U and a stack π such that:

(1) Given two Λ_c-terms ν,ξ there is a halt instruction $H_{\nu,\xi}$ and a Π-constant $\pi_{\nu,\xi}$ such that $U\star\nu.\xi.\pi \succ \mathrm{T}_{\mathrm{int}}\star H_{\nu,\xi}.\nu.\pi_{\nu,\xi}$.
(2) $U\star\rho$ reduces only if the stack ρ has the form $\nu.\xi.\pi$ for some terms ν,ξ.

Intuitively the instruction U replaces any term putting $\mathrm{T}_{\mathrm{int}}$ in head position and its first argument in second place in the stack. Since $U\star\nu.\xi.\pi \succ H_{\nu\xi}.\nu.\pi_{\nu\xi}$ where $H_{\nu\xi}$ and $\pi_{\nu\xi}$ are constants; if we replace the constant U by a term θ such that $\theta \star \nu.\xi.\pi \succ h_{\nu\xi}.\nu.\rho_{\nu\xi}$ where $h_{\nu\xi}$ is a term and $\rho_{\nu\xi}$ is a stack, the execution will never put $h_{\nu\xi}$ in head position and will never pop arguments from $\rho_{\nu\xi}$.

For technical reasons it is desirable to define dynamic substitutions having three parts:

- A *static part*, which acts as a cumulative parameter
- The *boot*, which is the origin of the thread over which we will apply the substitution
- A *dynamic part*, which contains the correspondence between the instructions we dynamically replace and the terms we put instead these instructions.

More formally, the dynamic and static parts are determined by functions from substitutable instructions to Λ_c-terms. The domain of a static (resp. dynamic) part is the set of substitutable instructions which are assigned by the part. For instance, we will denote as:

$$[{}^{t,\rho}/_{H_1,\pi_1}]^P\langle\theta/U\rangle$$

the dynamic substitution with static part $[{}^{t,\rho}/_{H_1,\pi_1}]$, boot on the process P and which replaces dynamically (in the sense we explained above) U by θ. The static part domain is $\{H_1,\pi_1\}$ and the dynamic part domain is $\{U\}$. The formal definition is given by induction on the number of times the instructions belonging to the dynamic part domain arrives in head position along th_P:

Definition 4.11. *Consider a stack* π. *The set of* $\mathrm{T_{int}}K-$like terms according to π *is defined by:*
(4.11.5)
$\mathcal{F}_{\mathrm{T_{int}}K}(\pi) := \{\theta{\in}\Lambda_c \mid \forall\nu,\xi\ \exists h_{\nu\xi},\rho_{\nu\xi}$ *such that* $\theta\star\nu.\xi.\pi \succ \mathrm{T_{int}}{\star}h_{\nu\xi}.\nu.\rho_{\nu\xi}\}$

A substitutable instruction U *is called* $\mathrm{Tint}K$*-like if and only if:*

(1) *For all* Λ_c*-terms* ν,ξ *there is a halt instruction* $H_{\nu,\xi}$ *and a* Π*-constant* $\pi_{\nu,\xi}$ *such that* $U\star\nu.\xi.\pi \succ \mathrm{T_{int}}{\star}H_{\nu,\xi}.\nu.\pi_{\nu,\xi}$.
(2) $U{\star}\rho$ *reduces only if the stack* ρ *has the form* $\nu.\xi.\pi$ *for some terms* ν,ξ.

The set $\overline{\mathcal{F}}_{\mathrm{T_{int}}K}(\pi)$ *of the* $\mathrm{T_{int}}K-$like instructions according to π, *is defined as the one which elements are the substitutable instructions U satisfying:*

We denote by $\overline{\mathcal{F}}_{\mathrm{T_{int}}K}(\pi) \subset \mathcal{F}_{\mathrm{T_{int}}}(\pi)$ the set of $\mathrm{Tint}K$-like instructions. Obviously $\overline{\mathcal{F}}_{\mathrm{T_{int}}K}(\pi) \subset \mathcal{F}_{\mathrm{T_{int}}}(\pi)$.

Definition 4.12. *Consider a process P, a static substitution* $\mathscr{S}_0$*, a stack sequence* $(\pi_i)_{i\in\mathbb{N}}$*, an instruction sequence* $(U^i)_{i\in\mathbb{N}}$*,* $U^i{\in}\overline{\mathcal{F}}_{\mathrm{T_{int}}K}(\pi_i)$ *and a* Λ_c*-term sequence* $(\theta^i)_{i\in\mathbb{N}}$*,* $\theta^i \in \mathcal{F}_{\mathrm{T_{int}}K}(\pi_i)$*. Thus we have:*

$$U^i\star\nu.\xi.\pi_i \succ \mathrm{T_{int}}{\star}H^i_{\nu\xi}.\nu.\pi^i_{\nu\xi} \ \ \textit{and} \ \ \theta^j\star\nu.\xi.\pi_j \succ \mathrm{T_{int}}{\star}h^j_{\nu\xi}.\nu.\rho^j_{\nu\xi}$$

We define now the dynamic substitution $\mathscr{S}_0\ {}^P\langle\theta^i/U^i\rangle_{i\in\mathbb{N}}$, *which is a function which transforms the thread* th_P *into a sequence of processes. This definition is made by induction on the number of times some instruction* U^i *arrives in head position in* th_P*:*

- *If no instruction* U^i *comes in head position in the thread* th_P*, then* $\mathscr{S}_0\ {}^P\langle\theta^i/U^i\rangle_{i\in\mathbb{N}}$ *is the static substitution* $\mathscr{S}_0 + [{}^{\theta^i}/_{U^i}]_{i\in\mathbb{N}}$ *on the thread* th_P*.*
- *Suppose that* $P \succ Q$ *where* $Q = U^j \star \nu_1.\xi_1.\pi_j$ *is the first occurrence of an instruction* U^i *in head position in* th_P*. By hypothesis on* U^j*, we have* $Q \succ Q' = \mathrm{T}_{\mathrm{int}} \star H^j_{\nu_1\xi_1}.\nu_1.\pi^j_{\nu_1\xi_1}$*. We split the thread* th_P *into the "segment" from* P *to* Q *and the thread* $\mathrm{th}_{Q'}$*. In the first one, we perform the substitution* $\mathscr{S}_0 + [{}^{\theta^i}/_{U^i}]_{i\in\mathbb{N}}$ *and in* $\mathrm{th}_{Q'}$*, we perform the substitution*
$$(\mathscr{S}_0 + [{}^{h_{\nu_1'\xi_1'},\rho_{\nu_1'\xi_1'}}/_{H_{\nu_1\xi_1},\pi_{\nu_1\xi_1}}])\ {}^{Q'}\langle\theta^i/U^i\rangle_{i\in\mathbb{N}}$$
where $\begin{array}{rl}\nu_1' := & (\mathscr{S}_0 + [{}^{\theta^i}/_{U^i}]_{i\in\mathbb{N}})(\nu_1)\\ \xi_1' := & (\mathscr{S}_0 + [{}^{\theta^i}/_{U^i}]_{i\in\mathbb{N}})(\xi_1)\end{array}$

Remark 4.13. this definition is formally correct only in the case when the U^j's come in head position only a finite number of times. In the general case, we simply observe that we can define the desired function on any finite initial segment of th_P and take the common extension.

The reader should be aware that, while substitution defined in 1.23 is a syntactic transformation defined over all processes and independent of reduction; dynamic substitution is defined only on a thread th_P and hence it depends upon the definition of reduction and on the "boot" process P.

Remark 4.14. This definition of dynamic substitution is clearly a particular case (suitable for our purposes in this work) of a much more general method. It is a powerful method in order to analyse the properties of the reduction of processes. Its drawback is that the set of available instructions must be restricted.

According to the situation, we will use different notations for these substitutions: In addition to the previously defined notation, we will allow us to denote a dynamic substitution by $\mathscr{S}^P\mathscr{D}$, where $\mathscr{S}$ denotes the static part, P the boot and $\mathscr{D}$ the dynamic part. If we are not interested to be explicit, we can use also a letter $\mathscr{D}$ to represent a dynamic substitution.

The *domain* of a dynamic part $\mathscr{D}$, which is denoted by $\mathrm{dom}\mathscr{D}$, is the set of substitutable instructions which are assigned by $\mathscr{D}$. For instance, $\mathrm{dom}(\langle\theta^i/U^i\rangle_{i\in\mathbb{N}}) = \{U^i \mid i\in\mathbb{N}\}$. Given two dynamic parts $\mathscr{D}_1$ and $\mathscr{D}_2$, we say that $\mathscr{D}_1$ extends $\mathscr{D}_2$ (we denote $\mathscr{D}_1 \sqsupseteq \mathscr{D}_2$) if and only if $\mathrm{dom}(\mathscr{D}_2) \subseteq \mathrm{dom}(\mathscr{D}_1)$ and $\mathscr{D}_1, \mathscr{D}_2$ coincide over $\mathrm{dom}(\mathscr{D}_2)$. We say that $\mathscr{D}_1$ and $\mathscr{D}_2$ are *coherent* if and only if they coincide on $\mathrm{dom}(\mathscr{D}_1) \cap \mathrm{dom}(\mathscr{D}_2)$ and in this case, we denote by $\mathscr{D}_1\mathscr{D}_2$ the "union" of these two dynamic parts.

Suppose we have a dynamic substitution $\mathscr{D}$ with boot P and that th_P is a finite thread. Then, the dynamic substitution $\mathscr{D}$ defines a finite increasing sequence of static substitutions which is ended by a static substitution, which we will denote as $\mathscr{D}^*$. This allows us to define a notion of (partial) "composition" in dynamic substitutions:

Definition 4.15. *Let us consider two dynamic substitutions* $\mathscr{S}_1^{P_1}\mathscr{D}_1$, $\mathscr{S}_2^{P_2}\mathscr{D}_2$ *and suppose that* th_P *is finite. We define*

$$\mathscr{S}_1^{P_1}\mathscr{D}_1 \rhd \mathscr{S}_2^{P_2}\mathscr{D}_2 := ((\mathscr{S}_1^{P_1}\mathscr{D}_1)^* + \mathscr{S}_2)^{P_2}\mathscr{D}_1\mathscr{D}_2$$

whenever $(\mathscr{S}_1^{P_1}\mathscr{D}_1)^* + \mathscr{S}_2$ *and* $\mathscr{D}_1\mathscr{D}_2$ *are defined.*

This composition first performs completely the left argument $\mathscr{S}_1^{P_1}\mathscr{D}_1$ in th_{P_1} to obtain $(\mathscr{S}_1^{P_1}\mathscr{D}_1)^*$ and after that it adds to the right argument $(\mathscr{S}_1^{P_1}\mathscr{D}_1)^*$ to the static part and $\mathscr{D}_1$ to the dynamic part. It will be useful to concatenate substitutions, built from thread schemes.

4.3. **Specification of Peirce's law.**

Another example of threads method application is the one which gives a characterization for all Λ_c-terms realizing the Peirce's law. Here we will use another variant of dynamic substitution.

Lemma 4.16. *Suppose that C is a universal realizer for Peirce's Law, i.e.* $C \Vdash \forall X\forall Y[((X \to Y) \to X) \to X]$. *Take a stack π and suppose v is a Λ_c-term such that for each Λ_c-term ξ there is a halt instruction H_ξ and a stack π'_ξ verifying* $v \star \xi.\pi \succ \xi \star H_\xi.\pi'_\xi$. *Then, there is a Λ_c-term ξ such that* $C \star v.\pi \succ H_\xi \star \pi$. *In particular, the thread of* $C \star v.\pi$ *is finite.*

Proof. Take $\perp\!\!\!\perp_0 = (\text{th}_{C\star v.\pi})^c$. By hypothesis $C \Vdash_0 ((\{\pi\} \to \bot) \to \{\pi\}) \to \{\pi\}$ and hence $v \nVdash_0 (\{\pi\} \to \bot) \to \{\pi\}$. Then, there is a Λ_c-term ξ such that $\xi \Vdash_0 \{\pi\} \to \bot$ and $C \star v.\pi \succ v \star \xi.\pi$. By reduction hypothesis in v, $v \star \xi.\pi \succ \xi \star H_\xi.\pi'_\xi$ and then $\xi \star H_\xi.\pi'_\xi \notin \perp\!\!\!\perp_0$. Since $\xi \Vdash_0 \{\pi\} \to \bot$, we have that $H_\xi \nVdash_0 \{\pi\}$, i.e. $H_\xi \star \pi \in \text{th}_{C\star v.\pi}$. Because H_ξ is a halt instruction, the process $H_\xi \star \pi$ must be at the end of the thread $\text{th}_{C\star v.\pi}$. □

Theorem 4.17. *Consider a reduction system that respects substitutions and a Λ_c-term C. Suppose that $(H_i)_{i\in\mathbb{N}}$ is a sequence of halt instructions and $(\pi_i)_{i\in\mathbb{N}}$ a sequence of Π-constants such that the H_i's and the π_i's does not appear in C. Then, the following statements are equivalent:*

(1) $C \Vdash \forall X\forall Y[((X \to Y) \to X) \to X]$
(2) *There is an integer n, a sequence of Λ_c-terms $k_1, \dots, k_n$ and an integer $i \le n$ such that:*

$$\begin{aligned} C \star H_0.\pi_0 &\succ H_0 \star k_1.\pi_0 \\ k_1 \star H_1.\pi_1 &\succ H_0 \star k_2.\pi_0 \\ &\vdots \\ k_n \star H_n.\pi_n &\succ H_i \star \pi_0 \end{aligned} \tag{4.17.6}$$

Proof. First we prove that (1) implies (2). We construct by induction a (finite) sequence of models.

Step 0 Take $\perp\!\!\!\perp_0 := \mathrm{th}^c_{C\star H_0.\pi_0}$. Since $C \Vdash_0 ((\{\pi_0\} \to \perp) \to \{\pi_0\}) \to \{\pi_0\}$ and $C\star H_0.\pi_0 \notin \perp\!\!\!\perp_0$, we can conclude that $H_0 \nVdash_0 (\{\pi_0\}\to\perp)\to\{\pi_0\}$. In consequence, there is a Λ_c-term k_1 such that $k_1 \Vdash_0 \{\pi_0\} \to \perp$ and $C\star H_0.\pi_0 \succ H_0\star k_1.\pi_0$.

Step j As induction hypothesis, we suppose we have the following threads scheme:

$$\begin{aligned} C\star H_0.\pi &\succ H_0\star k_1.\pi_0\\ k_1\star H_1.\rho_1 &\succ H_0\star k_2.\pi_0\\ &\vdots\\ k_{j-1}\star H_{j-1}\rho_{j-1} &\succ H_0\star k_j.\pi_0 \end{aligned} \tag{4.17.7}$$

Take the model $\perp\!\!\!\perp_j := \perp\!\!\!\perp_{j-1} \cap \mathrm{th}^c_{k_j\star H_j.\rho_j}$. Here there are two posibilities:

– There is an integer i from 1 to j such that $k_i \Vdash_j \{\pi_0\} \to \perp$. In this case $H_i \nVdash_j \{\pi_0\}$ because $k_i\star H_i.\rho_i \notin \perp\!\!\!\perp_j$. Moreover, H_i is a halt instruction different of H_0 and then $H_i\star\pi_0 \in \mathrm{th}_{k_j\star H_j.\rho_j}$. The construction stops and the result is proven for $n = j$.

– For all integers i from 1 to j, we have that $k_i \nVdash_j \{\pi_0\} \to \perp$. Because $C\star H_0.\pi_0 \notin \perp\!\!\!\perp_j$ and $C \Vdash_j ((\{\pi_0\} \to \perp) \to \{\pi_0\}) \to \{\pi_0\}$, we have that $H_0 \nVdash_j (\{\pi_0\} \to \perp) \to \{\pi_0\}$. Then there is a Λ_c-term k_{j+1} satisfying $k_{j+1} \Vdash_j \{\pi_0\} \to \perp$ and $H_0\star k_{j+1}.\pi_0 \notin \perp\!\!\!\perp_j$. Since we suppose that the terms k_i does not realize $\{\pi_0\} \to \perp$ in $\perp\!\!\!\perp_j$, the process $H_0\star k_{j+1}.\pi$ can only belong to $\mathrm{th}_{k_j\star H_j.\rho_j}$. Hence, we have that $k_j\star H_j.\rho_j \succ H_0\star k_{j+1}.\pi_0$. We add this thread to 4.17.7 and the construction continues.

claim: This construction process must stop.

Indeed, lets suppose that this construction process is infinite and consider a term ν satisfying $\nu\star\xi.\pi_0 \succ \xi\star H_\xi.\pi'_\xi$ (c.f. 4.16. For instance, consider $v{:=}\lambda x(x)H$, where H is a halt instruction). Applying ${}^{C\star H_0.\pi_0}\langle\nu/H_0\rangle$, we have that $\mathrm{th}_{C\star H_0.\pi_0}$ is an infinite thread. This is a contradiction with 4.16.

We prove now that (2) implies (1). Suppose (2), consider a model $\perp\!\!\!\perp$ and two truth values $\mathbb{X}$ and $\mathbb{Y}$. Take a Λ_c-term v and a stack ρ_0 such that $v \Vdash (\mathbb{X} \to \mathbb{Y}) \to \mathbb{X}$ in $\perp\!\!\!\perp$ and $\rho_0 \in \mathbb{X}$. We must prove that $C\star v.\rho_0 \in \perp\!\!\!\perp$.

We proceed applying $[{}^{v,\rho_0}/_{H_o,\pi_0}]$ in the 4.17.6 first thread. Hence, there is a Λ_c-term $\tilde{k}_1$ such that $C\star v.\pi \succ v\star\tilde{k}_1.\pi$. In order to prove the result, it suffices to prove that $\tilde{k}_1 \Vdash \mathbb{X} \to \mathbb{Y}$. Consider a Λ_c-term $h_1 \Vdash \mathbb{X}$ and a stack $\rho_1{\in}\mathbb{Y}$. Once again by substitution, applying $[{}^{h_0,h_1,\rho_0,\rho_1}/_{H_0,H_1,\pi_0,\pi_1}]$ on the second thread, we obtain $\tilde{k}_1\star h_1.\rho_1 \succ v\star\tilde{k}_2.\pi$ for a suitable Λ_c-term $\tilde{k}_2$. If we repeat this reasoning n times, we obtain a finite sequence $\tilde{k}_1,\ldots,\tilde{k}_n$ of Λ_c-terms such that, in order to prove the result, it suffices to prove that $\tilde{k}_n \Vdash \mathbb{X} \to \mathbb{Y}$. Consider a Λ_c-term h_n and a stack ρ_n such that

$h_n \Vdash \mathbb{X}$ and $\rho_n \in \mathbb{Y}$. By substitution $\left[{}^{(h_i)_{i\in[1..n]}(\rho_i)_{i\in[1..n]}}\big/_{(H_i)_{i\in[1..n]}(\pi_i)_{i\in[1..n]}}\right]$ applied in the 4.17.6 last thread, we have $\tilde{k}_n \star h_n.\rho_n \succ h_i \star \rho_0$. By construction, the term h_i realizes the parametric extended formula $\mathbb{X}$ and $\rho_0 \in \mathbb{X}$ and hence $\tilde{k}_i \star \rho_0 \in \perp\!\!\!\perp$. In consequence $\tilde{k}_n \star h_n.\rho_n \in \perp\!\!\!\perp$ and then, for all j from 1 to n we have that $\tilde{k}_j \Vdash \mathbb{X} \to \mathbb{Y}$. In particular $v \star \tilde{k}_1.\rho_0 \in \perp\!\!\!\perp$. Since our reasoning is valid for all v realizing $(\mathbb{X} \to \mathbb{Y}) \to \mathbb{X}$ and for all ρ_0 belonging to $\mathbb{X}$, the result is proven. □

In particular, since cc verifies 4.17.6 for $n = 1$ and $k_1 = k_\pi$, we can conclude again that the term cc realizes the Peirce's law. We prove now that a term realizing the Peirce's law in all saturated models is a sort of weak cc. We will say that a Λ_c-term k has the *k_π-reduction property* if and only if, for all Λ_c-terms t and for all stacks ρ, the process $k \star t.\rho$ reduces to $t \star \pi$.

Corollary 4.18. *Consider a reduction system that preserves the substitution of constants. Take a Λ_c-term C realizing the Peirce's law and consider a stack π. Then, there is a Λ_c-term k depending on H_0 such that for all Λ_c-term v that puts his first argument in head position with a no-empty stack as argument, we have the following reduction properties:*

(1) *$C \star v.\pi \succ v \star k[{}^{v}/_{H_0}].\pi$.*
(2) *The term $k[{}^{v}/_{H_0}]$ has the k_π-reduction property.*

Proof. Applying 4.16 we have for C a threads scheme as 4.17.6. Hypothesis about v means that for all Λ_c-terms t and for all stacks ρ, there is a Λ_c-term h and a stack π' -both depending on t and ρ- such that $v \star t.\rho \succ t \star h.\pi'$.

(1) If we take the first i-lines of the scheme 4.17.6 and we apply the substitution $\mathscr{D} := [{}^{\pi}/_{\pi_0}]^{C\star H_0.\pi}\langle v/H_0\rangle$, we obtain:

$$C \star v.\pi \succ v \star \mathscr{D}(k_i).\pi \tag{4.18.8}$$

By 4.11, there is a term $k[H_0]$ such that $k[{}^{v}/_{H_0}] := \mathscr{D}(k_i)$ and we have the result.

(2) Consider a term t and a stack ρ. We proceed by substitution in the last $(n-i)+1$ lines of the scheme 4.17.6. Let us consider $\mathscr{S}_0$ the static substitution defined by $\mathscr{D}$ in the i-th 4.17.6 thread. We define $\mathscr{S}' := (\mathscr{S}_0 + [{}^{t,\rho}/_{H_i,\pi_i}])^{k_i\star H_i.\pi_i}\langle v/H_0\rangle$ Applying $\mathscr{S}'$ to the last $(n-i)+1$ threads, we obtain the result. Notice that, if we substitute the constants $H_1,\ldots H_i$ by arbitrary Λ_c-terms in $k_i[{}^{v}/_{H_0}]$, we obtain also a term with the k_π-reduction property.

□

5. **Realizability Games**

Once defined a game class named *Realisability Games*, we will implement these games by means of interaction instructions. The main result about realizability games is that a universal realizer for a formula implements the associated game of the formula. Moreover, a universally realized formula has an associated game having a winning strategy and, in some particular cases, such a universal realizer plays also a winning strategy.

We will proceed to generalise the definition of interaction instructions given by *Krivine* in [8] and [6]. While games definition introduced in [8] defines a game associated with each Π^1_1-formula (in particular, to first order formulæ), the existential quantifiers, conjunction and disjunction of these formulæ are classically defined. The principal consequence adopting classical existential quantifiers is that programs obtained by proofs have not *garbage collector*. Moreover, the game formulædefined in [8] are Π^1_1 and therefore cannot use Leibniz equality. As we will see in this section, this particularity gives strategies the execution of which is abruptly broken when player has win. On the other hand, games defined in [6] uses Leibniz equality. However, these games are defined only for prenex normal form formulæ. In order to use intuitionistic existential quantifiers, connectors and Leibniz equality, the main problem is to define the game associated with some second order formulæ. Our approach consist in accepting some second order quantification, which suffices to express our intuitionistic formulæ and Leibniz equality. Since informally we can say that playing games over formulæ consists in instantiating universal quantifiers, the problem which appears when playing with second order formulæ is how to define a denumerable choice set for instantiating second order quantifiers. Here we introduce such a choice set and, because of our particular restrictions on second order quantification, the main theorem in realizability games is still provable.

Remark 5.1. Consider a term τ such that $\tau \Vdash \overset{\text{int}}{\exists} x \varphi(x)$ and an instruction $U \in \overline{\mathcal{F}}_{\mathrm{T_{int}}K}(\pi_0)$, $U \star \nu.\xi.\pi_0 \succ \mathrm{T_{int}} \star H_{\nu\xi}.\nu.\pi_{\nu\xi}$, for some constant stack π_0. We know that there is an integer n such that $\tau \star U.\pi_0 \succ H_{\nu\xi} \star \overline{n}.\pi_{\nu\xi}$ for some terms ν, ξ (c.f.4.5). More generally, consider some $\theta \in \mathcal{F}_{\mathrm{T_{int}}K}(\pi)$, $\theta \star \nu.\xi.\pi \succ \mathrm{T_{int}} \star h_{\nu\xi}.\nu.\pi$. Applying $[{}^{\pi}/_{\pi_0}]^{\tau\star U.\pi_0}\langle\theta/U\rangle$, we obtain:

$$\tau \star \theta.\pi \succ h_{\nu'\xi'} \star \overline{n}.\rho_{\nu'\xi'}$$

where $\nu'\xi'$ are obtained from ν, ξ by substitution. In other words, the initial stack π is restored.

In our machine the only place to store intermediate results is the stack. For this reason, we say that a program like τ collects garbage, because it cleans its stack before giving the result.

If we take instead of $\overset{\text{int}}{\exists} x \varphi(x)$ the weaker classical version of this formula: $\forall x(\mathrm{int}(x), \varphi(x) \to \bot) \to \bot$, a term realizing this formula computes also an integer but in general does not collect garbage.

We define now the language we adopt for games.

5.1. **The language for games.**

Definition 5.2. *The $\mathcal{L}_G$- formulæ are defined by induction as follows:*

- *Consider $\vec{x} = x_1, \dots, x_p$ first order variables, $\delta_1, \dots, \delta_h$ data types (defined by a formula or by a Λ_c-set), $\varphi_1, \dots, \varphi_r$ $\mathcal{L}_G$-formulæ and A an atomic formula. Then the formula*
$$\Phi = \forall \vec{x}[\delta_1, \dots, \delta_h, \varphi_1, \dots, \varphi_r \to A]$$
is a $\mathcal{L}_G$-formula. We say that such a formula is of first kind, *$\delta_1, \dots, \delta_h$ is its* data types block *and $\varphi_1, \dots, \varphi_r$ is its* hypothesis block.
- *Consider $\vec{x} = x_1, \dots, x_p$ first order variables, W an a-ary second order variable, $\vec{\tau} = \tau_1, \dots, \tau_a$ $\mathcal{L}$-terms, $\delta_1, \dots, \delta_h$ data types (defined by a formula or by a Λ_c-set) and $\varphi_1, \dots, \varphi_r$ $\mathcal{L}_G$-formulæ of two possible form:*
  1. *$\mathcal{L}_G$-formula that does not contain the variable W.*
  2. *$\mathcal{L}_G$-first kind formula of the form:*
  $$\forall \vec{y}[\gamma_1, \dots, \gamma_k, \psi_1, \dots, \psi_s \to W\vec{\sigma}]$$
  where $\psi_, \dots, \psi_s$ does not contain W and $\vec{\sigma} = \sigma_1, \dots, \sigma_a$ where each σ_j is a $\mathcal{L}$-term.

  Then, the formula
  $$\Phi = \forall W \forall \vec{x}[\delta_1, \dots, \delta_h, \varphi_1, \dots, \varphi_r \to W\vec{\tau}]$$
  is a $\mathcal{L}_G$-formula. We say that such a formula is of second kind, *$\delta_1, \dots, \delta_h$ is its* data types block *and that $\varphi_1, \dots, \varphi_r$ is its* hypothesis block.

This recursive definition allows to built first and second kind formulæ having no data types block or hypothesis block. As a particular case of extended formulæ, provided a $\mathcal{L}$-structure $\mathfrak{M} = (M, \mathcal{D}, \mathscr{Y})$ and an assignment $\mathscr{A}$, each $\mathcal{L}_G$-formula φ together with $\mathscr{A}$ constitutes a parametrical formula.

Remark 5.3. An atomic formula A is a first kind $\mathcal{L}_G$-formula without data types block nor hypothesis block.

The formula $\text{int}(x) = \forall X[\forall y(Xy \to X\,\text{s}y), X0 \to Xx]$ is not a $\mathcal{L}_G$-formula because its hypothesis $\forall y(Xy \to X\,\text{s}y)$ has X in the left side of the arrow. More generally, a recursive data type is not a $\mathcal{L}_G$-formula. Indeed, the recursive definition of a data type δ must built from a data of type δ and other arguments a new data of type δ. Then, a recursive data type is a second order formula of the form $\forall X(\dots)$ that contains an hypothesis with the variable X both on the left and right side of the arrow (as the formula $\forall y(Xy \to X\,\text{s}y)$ in the type int). On the other hand, $\text{Bool}(x)$ is a $\mathcal{L}_G$-formula.

We prove now that we can embed the first order language $\mathcal{L}_1$ into the language of games $\mathcal{L}_G$ in such a way that:

(1) The embedding preserves truth values.

(2) The embedding applied to each $\mathcal{L}_1$-formula Φ gives a $\mathcal{L}_G$-formula logically equivalent to Φ.

Lemma 5.4.

(1) *$\mathcal{L}_G$ contains the $\mathcal{L}_1$ atomic formulæ.*
(2) *$\mathcal{L}_G$ is closed under disjunction, conjunction and first order existential quantification, the last one relativized to data types or not.*
(3) *Given a $\mathcal{L}_G$-formula Φ, a first order variable y and a data type δ, each of the formulæ $\forall y\Phi$ and $\forall \overset{\delta}{y}\Phi$ is truth equivalent to a $\mathcal{L}_G$-formula.*

Proof.

(1) The *false* formula $\perp$ belongs to $\mathcal{L}_G$ because $\forall XX$ is a second kind $\mathcal{L}_G$-formula without data types block nor hypothesis block. The *equality* formula $\tau_1 = \tau_2$ is $\forall X(X\tau_1 \to X\tau_2)$ (c.f. 1.12) which is a second kind $\mathcal{L}_G$-formula without data types block (the hypothesis formula $X\tau_1$ is a first kind formula containing X only in the atom).
(2) Let φ_1, φ_2 be two $\mathcal{L}_G$-formulæ. The formula $\varphi_1 \wedge \varphi_2$ is defined as $\forall X((\varphi_1, \varphi_2 \to X) \to X)$, which is a second kind $\mathcal{L}_G$-formula.

    The formula $\varphi_1 \vee \varphi_2$ is defined as $\forall X((\varphi_1 \to X), (\varphi_2 \to X) \to X)$, which is a second kind $\mathcal{L}_G$-formula.

    Consider Φ a $\mathcal{L}_G$-formula, y a first order variable and δ a data type. Then $\exists y\Phi$ is the formula $\forall Y[\forall y(\Phi \to Y) \to Y]$ (c.f.1.12), which is a second kind $\mathcal{L}_G$-formula.

    Similarly, $\exists \overset{\delta}{y}\Phi$ is $\forall Y[\forall y(\delta(y), \Phi \to Y) \to Y]$; which is also a second kind $\mathcal{L}_G$-formula.
(3) Consider Φ a $\mathcal{L}_G$-formula, γ a data type and y a first order variable. Suppose that Φ is a first kind formula $\forall \vec{x}[\delta_1, \dots, \delta_h, \varphi_1, \dots, \varphi_r \to A]$. Then, $\forall \overset{\gamma}{y}\Phi = \forall y[\gamma(y) \to \forall \vec{x}[\delta_1, \dots, \delta_h, \varphi_1, \dots, \varphi_r \to A]]$ is truth equivalent to the $\mathcal{L}_G$-formula $\forall y \forall \vec{x}[\gamma(y), \delta_1, \dots, \delta_k, \varphi_1, \dots, \varphi_r \to A]$.

    Similarly, suppose that $\Phi = \forall W \forall \vec{x}[\delta_1, \dots, \delta_h, \varphi_1, \dots, \varphi_r \to \vec{\tau}]$. Then $\forall \overset{\gamma}{y}\Phi = \forall y[\gamma(y) \to \forall W \forall \vec{x}[\delta_1, \dots, \delta_h, \varphi_1, \dots, \varphi_r \to W\vec{\tau}]]$ is truth equivalent to $\forall W \forall y \forall \vec{x}[\gamma(y), \delta_1, \dots, \delta_h, \varphi_1, \dots, \varphi_r \to W\vec{\tau}]$, which is also a $\mathcal{L}_G$-formula.

□

Corollary 5.5. *There is a map $\mathcal{G} : \mathcal{L}_1 \to \mathcal{L}_G$ such that for each realizability model $(\mathfrak{M}, \perp\!\!\!\perp)$ and for each $\Phi \in \mathcal{L}_1$, we have that:*

- $\vdash \Phi \leftrightarrow \mathcal{G}(\Phi)$
- $||\Phi|| = ||\mathcal{G}(\Phi)||$.

Proof. We define $\mathcal{G}$ by induction in $\mathcal{L}_1$.

- For atomics $\mathcal{L}_1$-formulæ A, we define $\mathcal{G}(A) := A$.
- Given three $\mathcal{L}_1$-formulæ $\varphi, \varphi_1, \varphi_2$, we define $\mathcal{G}(\forall x\varphi) := \forall x\mathcal{G}(\varphi)$, $\mathcal{G}(\exists x\varphi) := \exists x\mathcal{G}(\varphi)$, $\mathcal{G}(\varphi_1 \wedge \varphi_2) := \mathcal{G}(\varphi_1) \wedge \mathcal{G}(\varphi_2)$ and $\mathcal{G}(\varphi_1 \vee \varphi_2) := \mathcal{G}(\varphi_1) \vee \mathcal{G}(\varphi_2)$.

- Given φ_1, φ_2 two $\mathcal{L}_1$-formulæ. We define the embedding of $\varphi_1 \to \varphi_2$ by induction on the length of φ_2:
  - For A atomic, we define $\mathscr{G}(\varphi_1 \to A) := \mathscr{G}(\varphi_1) \to A$.
  - $\mathscr{G}(\varphi_1 \to \forall x\psi) := \forall x \mathscr{G}(\varphi_1 \to \psi)$.
  - $\mathscr{G}(\varphi_1 \to \forall X\psi) := \forall X \mathscr{G}(\varphi_1 \to \psi)$ (for instance, this is the case of $\varphi_1 \to \perp$ and $\varphi_1 \to \exists x\psi$).

First we check by induction that for each $\mathcal{L}_1$-formula Φ, we have that $||\Phi|| = ||\mathscr{G}(\Phi)||$. For atomics and the recursive cases of $\forall$, $\exists$, $\wedge$ and $\vee$ is a direct application of 5.4. Taking a formula $\varphi_1 \to A$ where A is atomic we have $||\mathscr{G}(\varphi_1 \to A)|| = ||\mathscr{G}(\varphi_1) \to A|| = |\mathscr{G}(\varphi_1)| \rightsquigarrow ||A|| = |\varphi_1| \rightsquigarrow ||A|| = ||\varphi_1 \to A||$. For a formula $\varphi_1 \to \forall x\psi$, we have $||\mathscr{G}(\varphi_1 \to \forall x\psi)|| = \bigcup_{n\in\mathbb{N}} ||\varphi_1 \to \psi(n)|| = \bigcup_{n\in\mathbb{N}} |\varphi_1| \rightsquigarrow ||\psi(n)|| = |\varphi_1| \rightsquigarrow \bigcup_{n\in\mathbb{N}} ||\psi(n)|||| = ||\varphi_1 \to \forall x\psi||$. For a formula $\varphi_1 \to \forall X\psi$ is similar. The logical equivalence of Φ and $\mathscr{G}(\Phi)$ is straightforward. □

Notation 5.6. For a $\mathcal{L}_G$-formula Φ, we denote as T^Φ the storage operator corresponding to the data types block of Φ.

5.2. **Playing with formulæ.**

We define now a game in $\mathcal{L}_G$-formulæ.

Definition 5.7. *Take a ω-model $\mathfrak{M} = (\mathbb{N}, \mathcal{D}, \mathscr{Y})$ of the second order arithmetics. Given a vector $\vec{n} \in \mathbb{N}^a$, the set $\{\vec{x} \in \mathbb{N}^a \mid \vec{x} \neq \vec{n}\}$ will be denoted as $D_{\vec{n}}$ and the set $\{D_{\vec{n}} \mid \vec{n} \in \mathbb{N}^a\}$ as $\mathcal{D}_a$.*

Remark 5.8. By comprehension schemata, $\forall a \in \mathbb{N}\ \mathcal{D}_a \subseteq \mathcal{D}$. Indeed, the formula $\varphi_n(x) := x = \underbrace{(s)\ldots(s)}_{n} 0 \to \perp$ is a $\mathcal{L}$-formula (without parameters) and in consequence $\varphi_{n_1}(x_1) \vee \cdots \vee \varphi_{n_a}(x_a)$ is also. Hence, for each vector $\vec{n}$, $D_{\vec{n}}$ is a second order parameter.

Definition 5.9. *The game is played between two players named ⓥ and ∃⃝, which play taking parametrical $\mathcal{L}_G$-formulæ belonging to three sets named $\mathscr{U}$, $\mathscr{V}$ and $\mathscr{A}$. These sets are redefined at each step during the play and we will denote them with a subscript which indicates the current step. We define first a rule for each $\mathcal{L}_G$-formula Φ and each player ∃⃝ and ∀⃝:*

(1) *Rules for Φ and ∀⃝:*

First kind *Suppose that Φ is $\forall\vec{x}[\delta_1(x_1), \ldots, \delta_r(x_r), \varphi_1, \ldots, \varphi_h \to A]$ where $\vec{x}$ is $x_1, \ldots, x_p$.*
∀⃝ must choose p individuals $\vec{n} = n_1, \ldots, n_p$ in such a way that $\forall i \in [1..r]\ \mathfrak{M} \models \delta_i(n_i)$. After that, $\mathscr{U}_{\mathbf{s+1}}$ is defined as $\mathscr{U}_{\mathbf{s}} \cup \{\varphi_1(\vec{n}), \ldots, \varphi_h(\vec{n})\}$ and $\mathscr{A}_{\mathbf{s+1}}$ as $\mathscr{A}_{\mathbf{s}} \cup \{A(\vec{n})\}$.

Second kind *Suppose that Φ is $\forall W \forall\vec{x}[\delta_1(x_1), \ldots, \delta_r(x_r), \varphi_1, \ldots, \varphi_h \to W\vec{\tau}]$ where $\vec{x}$ is $x_1, \ldots, x_p$ and W has arity a.*
∀⃝ must choose p individuals $\vec{n} = n_1, \ldots, n_p$ in such a way that $\forall i \in [1..r]\ \mathfrak{M} \models \delta_i(n_i)$. This choice determines the interpretation $\vec{t}$ of $\vec{\tau}$ in the model $\mathfrak{M}$ and a set $D_{\vec{t}} \in \mathcal{D}_a$ which is the only one

that does not contain $\vec{t}$. After that, $\mathscr{U}_{\mathbf{s+1}}$ is defined as $\mathscr{U}_{\mathbf{s}} \cup \{\varphi_1(D_{\vec{t}}, \vec{n}), \ldots, \varphi_h(D_{\vec{t}}, \vec{n})\}$ and $\mathscr{A}_{\mathbf{s+1}}$ as $\mathscr{A}_{\mathbf{s}} \cup \{D_{\vec{t}}(\vec{t})\}$.

(2) *Rules for Φ and $\textcircled{\exists}$:*

First kind *Suppose that Φ is $\forall\vec{x}[\delta_1(x_1), \ldots, \delta_r(x_r), \varphi_1, \ldots, \varphi_h \to A]$ where $\vec{x}$ is $x_1, \ldots, x_p$.*
$\textcircled{\exists}$ must choose p individuals $\vec{n} = n_1, \ldots, x_p$ in such a way that $\forall i{\in}[1..r]\ \mathfrak{M} \models \delta_i(n_i)$ and $A(\vec{n}){\in}\mathscr{A}_{\mathbf{s+1}}$. After that, $\mathscr{V}_{\mathbf{s+1}}$ is defined as $\{\varphi_1(\vec{n}), \ldots, \varphi_h(\vec{n})\}$.

Second kind *Suppose that Φ is $\forall W \forall\vec{x}[\delta_1(x_1), \ldots, \delta_r(x_r), \varphi_1, \ldots, \varphi_h \to W\vec{\tau}]$ where $\vec{x}$ is $x_1, \ldots, x_p$ and W has arity a.*
$\textcircled{\exists}$ must choose p individuals $\vec{n} = n_1, \ldots, n_p$. This choice determines the interpretation $\vec{t}$ of $\vec{\tau}$ in the model $\mathfrak{M}$ and a set $D_{\vec{t}} \in \mathcal{D}_a$ which is the only one that does not contain $\vec{t}$. After that, $\mathscr{V}_{\mathbf{s+1}}$ is defined as $\{\varphi_1(D_{\vec{t}}, \vec{n}), \ldots, \varphi_h(D_{\vec{t}}, \vec{n})\}$.

The game is defined by induction on the steps:

Step 0 *$\mathscr{U}_{\mathbf{0}} := \mathscr{A}_{\mathbf{0}} := \emptyset$ and $\mathscr{V}_{\mathbf{0}}$ is a finite set of parametrical $\mathcal{L}_G$-formulæ.*

Step $s{+}1$ *$\textcircled{\forall}$ plays first, choosing a formula Φ belonging to $\mathscr{V}_{\mathbf{s}}$ and applying the $\textcircled{\forall}$-rule that corresponds to Φ. After that, $\textcircled{\exists}$ plays choosing a formula Ψ belonging to $\mathscr{U}_{\mathbf{s+1}}$ and applying the $\textcircled{\exists}$-rule that corresponds to Ψ.*

The game stops once a player cannot play.

5.3. **Allowing interaction to implement games.**
We implement now these games using Λ_c-terms and processes.

In order to implement these games, we are going to associate new truth values to $\mathcal{L}_G$-formulæ. We will use these truth values as choice sets for $\textcircled{\forall}$, hence it must be recursively enumerable. However, these new truth values will be not compatible with soundness lemma (hence not definable by a realizability model). The comprehension schemata is not true with this semantics, because of the restriction on the second order parameters.

Definition 5.10. *For each integer k, $\vec{n} \in \mathbb{N}^k$ and $\pi \in \Pi$, we define the truth values:*

$$\Delta_k^{\pi,\vec{n}} : \mathbb{N}^k \to \mathscr{T} \text{ such that } \Delta_k^{\pi,\vec{n}}(\vec{m}) := \begin{cases} \{\pi\} & \text{if } \vec{n} = \vec{m} \\ \emptyset & \text{if } \vec{n} \neq \vec{m} \end{cases} \tag{5.10.9}$$

and $\phi_k : \mathbb{N}^k \to \mathscr{T}$, such that $\phi_k(\vec{m}) := \emptyset$. We define the parameter sets $\Delta_k{:=}\{\Delta_k^{\pi,\vec{n}} \mid \pi{\in}\Pi$ and $\vec{n}{\in}\mathbb{N}^k\} \cup$ $\{\phi_k\}$ and $\Delta = \bigcup_{k\in\mathbb{N}} \Delta_k$.

Remark 5.11. Suppose that $\mathbb{P}$ is a a-ary predicate, $\pi \in \Pi$ and $\vec{m} \in \mathbb{N}^a$. If $\pi \in \mathbb{P}(\vec{m})$, then $\forall^{\mathbb{N}^a}\vec{x}\Delta_a^{\pi,\vec{m}}(\vec{x}) \subseteq \mathbb{P}(\vec{x})$ (because on a-uplets other than $\vec{m}$, $\Delta_a^{\pi,\vec{m}}$ takes the empty value). In fact, Δ_a is a maximal set Δ satisfying this property:

$$\forall \mathbb{P} a\text{-ary}\ \forall \mathbb{D}{\in}\Delta\ \forall\pi{\in}\Pi\ \forall\vec{m}{\in}\mathbb{N}^a(\pi \in \mathbb{P}(\vec{m}) \cap \mathbb{D}(\vec{m}) \to \forall^{\mathbb{N}^a}\vec{x}\mathbb{D}(\vec{x}) \subseteq \mathbb{P}(\vec{x}))$$

Definition 5.12. *For each $\mathcal{L}_G$ parametrical formula Φ the parameters of which are taken from Δ, we add to Λ_c a new constant $\varkappa_\Phi$.*

Intuitively, when a constant $\varkappa_\Phi$ arrives in head position, it corresponds to the fact that $\textcircled{\exists}$ has chosen in $\mathscr{U}_s$ the formula Φ.

Corollary 5.13. *Let $\chi_1,\dots,\chi_k$ be parametrical formulæ, $\mathbb{P}$ a a-ary predicate and $\pi \in \mathbb{P}(\vec{\tau})$ for some and π. Then*

$$||\chi_1,\dots,\chi_k, \to \mathbb{P}(\vec{\tau})|| \subseteq ||\chi_1,\dots,\chi_k \to \Delta_a^{\pi,\vec{n}}(\vec{\tau})||$$

Proof. By 5.11 we have that $\Delta_a^{\pi,\vec{n}}(\tau^{\mathscr{Y}}) \subseteq \mathbb{P}(\tau^{\mathscr{Y}})$ and hence we have the result because the formulæ χ_i does not contain X as free variable. □

The reader should be noticed that we can apply 5.13 to the hypothesis on second kind formulæ.

Intuitively, the predicate $\Delta_k^{\pi,\vec{n}}$ plays the role of the $D_{\vec{n}}$ we defined in 5.7. Indeed, this predicate is equivalent to $\top$ in any vector $\vec{m} \neq \vec{n}$. , we must explain why we choose a singleton and not the whole truth value $||\bot|| = \Pi$ when $\vec{m} = \vec{n}$. We make this choice in order to get the properties 5.11 and 5.13, which are necessary in order to prove the *Main Theorem* 5.23.

We proceed to define, by induction, a truth value for $\mathcal{L}_G$ Δ-parametrical formulæ.

Definition 5.14. *Given a $\mathcal{L}_G$-formula Φ, we denote this new truth value as $[\Phi]$ and we call it the* G-value *of Φ.*

- *Consider a first kind $\mathcal{L}_G$ Δ-parametrical formula*

$$\Phi = \forall\vec{x}[\delta_1(x_1),\dots,\delta_r(x_r),\varphi_1,\dots,\varphi_h \to A]$$

  where $\vec{x} = x_1,\dots,x_p$. We define:

$$[\Phi] := \{d_1\dots d_r.\mathrm{T}^{\varphi_1}\varkappa_{\varphi_1(\vec{m})}\dots\mathrm{T}^{\varphi_h}\varkappa_{\varphi_h(\vec{m})}.\pi \mid \vec{m}\in\mathbb{N}^p, d_i\in\mathcal{E}_{\delta_i}(m_i), \pi\in||A(\vec{m})||\}$$

- *Consider a second kind $\mathcal{L}_G$ Δ-parametrical formula*

$$\Phi = \forall W\forall\vec{x}[\delta_1(x_1),\dots,\delta_r(x_r),\varphi_1,\dots,\varphi_h \to W\vec{\tau}]$$

  where $\vec{x} = x_1,\dots,x_p$ and W has arity a. We define:

$$[\Phi] := \{d_1\dots d_r.\mathrm{T}^{\varphi_1}\varkappa_{\varphi_1(\mathbb{P},\vec{m})}\dots\mathrm{T}^{\varphi_h}\varkappa_{\varphi_h(\mathbb{P},\vec{m})}.\pi \mid \vec{m}\in\mathbb{N}^p, \mathbb{P}\in\Delta_a, d_i\in\mathcal{E}_{\delta_i}(m_i), \pi\in\mathbb{P}(\vec{\tau})\}$$

In the following paragraph, we describe reduction of $\varkappa_\Phi$ constants where $\Phi \in \mathcal{L}_G$ is a Δ-parametrical formula. The reduction of these constants will be not deterministic, i.e. a process with a constant $\varkappa_\Phi$ in head position has many possible successors. We will use the symbol $\succ\!\succ$ to denote this reduction.

Definition 5.15.

- ***First kind formulæ:***
  Let be $\Phi = \forall\vec{x}(\delta_1,\dots,\delta_r,\varphi_1,\dots,\varphi_h \to A)$ where $\vec{x} = x_1,\dots,x_p$. We define the reduction of all the processes of the form
  $$\varkappa_\Phi \star d_1 \dots d_r.\xi_1 \dots \xi_h.\pi$$
  – *$\textcircled{\exists}$ chooses $\vec{m} \in \mathbb{N}^p$. This move is a* legal move *if and only if $\forall i{\in}[1..r](d_i \in \mathcal{E}_i(m_i))$ and $\pi \in ||A(\vec{m})||$. If the move $\vec{m}$ is not legal, $\textcircled{\exists}$ has lost.*
  – *$\textcircled{\forall}$ chooses an integer $j \in [1..h]$ and a stack ρ. This move is a* legal answer *if and only if $\rho \in [\![\varphi_j(\vec{m})]\!]$. If the answer (j,ρ) is not legal, $\textcircled{\forall}$ has lost.*
  If $\textcircled{\exists}$ plays a legal move $\vec{m}$ and $\textcircled{\forall}$ plays a legal answer (j,ρ), we say that $\varkappa_\Phi \star d_1 \dots d_h.\xi_1 \dots \xi_h.\pi$ reduces to $\xi_j \star \rho$. We denote this reduction by
  $$\varkappa_\Phi \star d_1 \dots d_h.\xi_1 \dots \xi_h.\pi \succ\!\succ \xi_j \star \rho$$
- ***Second kind formulæ:***
  Let be $\Phi = \forall W \forall\vec{x}(\delta_1(x_1),\dots,\delta_r(x_r),\varphi_1,\dots,\varphi_k \to W\vec{\tau})$, where $\vec{x} = x_1,\dots x_p$ and W has arity a. We define the reduction of all the processes of the form
  $$\varkappa_\Phi \star d_1 \dots d_r.\xi_1 \dots \xi_h.\pi$$
  – *$\textcircled{\exists}$ chooses $\vec{m} \in \mathbb{N}^p$ and a predicate $\mathbb{D}$ belonging to the domain Δ_a defined in 5.10. This move is a* legal move *if and only if $\forall i{\in}[1..r](d_i{\in}\mathcal{E}_{\delta_i}(m_i))$ and $\pi \in \mathbb{D}(\vec{\tau})$. If the move $(\vec{m},\mathbb{D})$ is not legal, $\textcircled{\exists}$ has lost.*
  – *$\textcircled{\forall}$ chooses an integer $j{\in}[1..h]$ and a stack ρ. This move is a* legal answer *if and only if $\rho \in [\![\varphi_i(\mathbb{D},\vec{m})]\!]$. If the answer (j,ρ) is not legal, $\textcircled{\forall}$ has lost.*
  If $\textcircled{\exists}$ plays a legal move $(\vec{m},\mathbb{D})$ and $\textcircled{\forall}$ plays a legal answer (j,ρ), we say that $\varkappa_\Phi \star d_1 \dots d_r.\xi_1 \dots \xi_h.\pi$ reduces to $\xi_j \star \rho$. We denote this reduction by
  $$\varkappa_\Phi \star d_1 \dots d_r.\xi_1 \dots \xi_k.\pi \succ\!\succ \xi_j \star \rho$$

Remark 5.16. Let be Φ a first kind formula $\forall\vec{x}(\delta_1,\dots,\delta_r \to A)$. The constant $\varkappa_\Phi$ does not reduce because there is no hypothesis to choose for $\textcircled{\forall}$.

Remark 5.17. Consider a first kind $\mathcal{L}_G$ parametrical formula Φ:
$$\Phi = \forall\vec{x}(\delta_1(x_1),\dots,\delta_r(x_r),\varphi_1,\dots,\varphi_h \to A)$$
where $\vec{x} = x_1,\dots,x_p$. Suppose that we are reducing the process
$$\varkappa_\Phi \star d_1 \dots d_r.\xi_1 \dots \xi_h.\pi$$
where each d_i belongs to $\mathcal{E}_{\delta_i}(m_i)$ for a suitable integer m_i. Then, a legal move for $\textcircled{\exists}$ must be a p-uplet $(n_1,\dots,n_p)$ such that for each $i \le r$, $n_i = m_i$. This one is a legal move if and only if $\pi \in [\![A(\vec{n})]\!]$.

In particular, if all the first order variables of Φ are relativized to data types (i.e. $p = r$), there is at most one possible move for $\textcircled{\exists}$: the p-uplet

$(m_1,\dots,m_p)$. Moreover, these m_i are determined by their canonical representations $d_1,\dots,d_p$.

Consider a second kind $\mathcal{L}_G$ parametrical formula

$$\Phi = \forall W \forall \vec{x}(\delta_1(x_1),\dots,\delta_r(x_r),\varphi_1,\dots,\varphi_h \to W\vec{\tau})$$

where $\vec{x} = x_1,\dots,x_p$ and W has arity a. Suppose that we are reducing the process $\varkappa_\Phi \star d_1 \dots d_r.\xi_1 \dots \xi_h.\pi$. Then, for all p-uplet of integers $\vec{n}$ such that for each $i \le r$ $n_i = m_i$, there is exactly one possible choice of $\mathbb{D} \in \Delta_a$ such that $(\vec{m},\mathbb{D})$ is a legal move: $\mathbb{D} := \Delta_a^{\pi,\vec{\tau}}$. Hence, $\exists$ has always a legal move and the predicate $\mathbb{D}$ is determined by the choice of $\vec{n}$.

Moreover, if all the first order variables of Φ are relativized to data types (i.e. $r = p$) and we are performing $\varkappa_\Phi \star d_1 \dots d_p.\xi_1 \dots \xi_k.\pi$, where each d_i belongs to $\mathcal{E}_{\delta_i}(m_i)$; then there is exactly one possible legal move for $\exists$: the p-uplet $(m_1,\dots,m_p)$ together with the second order parameter $\Delta_a^{\pi,\vec{\tau}}$.

In consequence: *for the formulæ that are completely relativized to data types, when a constant $\varkappa_\Phi$ arrives in head position, the move of $\exists$ is determined by the stack.* In other words, this fact implies that the machine replaces $\exists$ whenever the formula Φ has all their quantifiers relativized to data types.

Since $[\![\Phi]\!]$ sets are used as choice sets for player $\forall$, it is important to analyse in which conditions $[\![\Phi]\!]$ is empty:

Remark 5.18. Let us consider a first kind $\mathcal{L}_G$ Δ-parametrical formula:

$$\Phi = \forall \vec{x}[\delta_1(x_1),\dots,\delta_r(x_r),\varphi_1(\vec{x}),\dots,\varphi_s(\vec{x}) \to A(\vec{x})]$$

where $\vec{x} = x_1,\dots,x_p$. By definition, a stack belonging to $[\![\Phi]\!]$ is:

$$e_1 \dots e_r.\mathrm{T}^{\varphi_1(\vec{m})}\varkappa_{\varphi_1(\vec{x})} \dots \mathrm{T}^{\varphi_s(\vec{m})}\varkappa_{\varphi_s(\vec{m})}.\rho$$

where $\forall i \in [1..r]$ $e_i \in \mathcal{E}_{\delta_i}(m_i)$ and $\rho \in ||A(\vec{m})||$ for a suitable vector $\vec{m} \in \mathbb{N}^p$. The (parametric) atomic formula $A(\vec{x})$ is of two possible shapes:

- $A(\vec{x}) = \top$. In this case $||A(\vec{m})|| = \emptyset$ for each instantiation $\vec{m}$ and hence $[\![\Phi]\!] = \emptyset$.
- $A(\vec{x}) = \mathbb{W}\vec{\tau}$ where $\mathbb{W} : \mathbb{N}^p \to \mathcal{P}ow(\Pi)$. We can define $\mathrm{dom}(A(\vec{x})) := \{\vec{m} \in \mathbb{N}^p \mid \mathbb{W}(\vec{\tau}(\vec{m})) \neq \emptyset\}$. In particular, since $\mathbb{W}$ must be a Δ parameter, $\mathbb{W} = \Delta_a^{\pi,\vec{n}}$ for some $a \in \mathbb{N}, \vec{n} \in \mathbb{N}^a$ and $\pi \in \Pi$ (cf: 5.10) and then $\mathrm{dom}(A(\vec{x})) = \{\vec{m} \in \mathbb{N}^p \mid \vec{n} = \vec{\tau}(\vec{m})\}$. There is a stack belonging to $[\![\Phi]\!]$ if and only if there is an instantiation $\vec{m}$ belonging to $S_{\delta_1} \times \cdots \times S_{\delta_r} \times \mathbb{N}^{p-r} \cap \mathrm{dom}(A(\vec{x}))$ (S_{δ_1} is the *scope* of δ_i –c.f.:3.2).

Let us consider a second kind parametrical $\mathcal{L}_G$-formula:

$$\Phi = \forall W \forall \vec{x}[\delta_1(x_1),\dots,\delta_r(x_r),\varphi_1(W,\vec{x}),\dots,\varphi_s(W,\vec{x}) \to W\vec{\tau}]$$

Consider also $\vec{m} \in S_{\delta_1} \times \cdots \times S_{\delta_r} \times \mathbb{N}^{p-r}$ (the scopes never are empty) and a stack ρ. Instantiating $\vec{x}$ as $\vec{m}$ and W as $\mathbb{W} := \Delta_a^{\rho,\vec{\tau}}$ where a is the W-arity and taking $e_1,\dots,e_r$ such that $e_i \in \mathcal{E}_i(m_i)$ and $\rho \in \mathbb{W}$ we can built the stack $e_1 \dots e_r . \mathrm{T}^{\varphi_1(\mathbb{W},\vec{m})} \varkappa_{\varphi_1(\mathbb{W},\vec{m})} \dots \mathrm{T}^{\varphi_s(\mathbb{W},\vec{m})} \varkappa_{\varphi_s(\mathbb{W},\vec{m})} . \rho$ which belongs to $[\Phi]$. Hence, $[\Phi] \neq \emptyset$ for each second kind Δ-parametrical formula Φ.

5.4. **Winning strategies and the Main Theorem.**

Definition 5.19. *A* winning final position *is a process $P = \varkappa_\Phi \star \pi$ such that* ∃ *can move in such a way that* ∀ *can not answer (in chess terminology "*∃ moves and wins*"). We will denote by G the set of all* final winning positions.

More precisely, given a final winning position $\varkappa_\Phi \star \pi$ we have the following cases:

Let us suppose Φ is a first kind formula:

$$\Phi = \forall \vec{x} [\delta_1(x_1), \dots, \delta_r(x_r), \varphi_1(\vec{x}), \dots, \varphi_s(\vec{x}) \to A(\vec{x})]$$

where $\vec{x} = x_1, \dots, x_p$. If $\varkappa_\Phi \star \pi$ is a final winning position, ∃ can play and this implies that:

(1) $\pi = e_1 \dots e_r . \xi_1 \dots \xi_s . \rho$ for suitable terms $e_1, \dots, e_r, \xi_1, \dots, \xi_s$ and a suitable stack ρ (in other words, $\ell(\pi) \geq r + s$).
(2) $\{\vec{m} \in \mathbb{N}^p \mid \forall i \in [1..r]\ e_i \in \mathcal{E}_{\delta_i}(m_i) \text{ and } \rho \in ||A(\vec{m})||\ \} \neq \emptyset$

Let be $\vec{m}$ a ∃ (legal) move such that ∀ cannot answer. Then, there are two possibilities:

(1) There is not hypothesis block in Φ ($s = 0$). In this case, $\pi \in [\Phi]$.
(2) There is hypothesis block in Φ but the sets $[\varphi_i(\vec{m})]$ are each empty (c.f.:5.18)

Definition 5.20. *Given a set of processes S and a process P, we say that P has a* winning strategy according to *S if and only if the player* ∃ *can play in such a way that P reduces to a process of S independently of the answers of* ∀. *We denote this relation by $P \succ\!\succ S$. The set $\{P \mid P \succ\!\succ S\}$ is denoted by $\perp\!\!\perp_S$.*

The sets $\perp\!\!\perp_S$ are saturated because they are closed by *deterministic* anti-reduction. In other words, if P and P' are processes such that $P \succ P'$ and P' has a winning strategy for ∃, then P has also a winning strategy for ∃.

We denote by $\Vdash_S$ the realisability relation relative to $\perp\!\!\perp_S$ and by $||\ ||_S$ (or simply by $||\ ||$ when it is not ambiguous) the truth value relative to $\perp\!\!\perp_S$.

On this work, the set S that we use is always G, the set of all winning final positions. However, we can work with the general notion of winning strategy as follows:

- Instead of our defined reduction rules, we can take other rules, particularly we can define other interaction instructions.
- Instead of our set G of winning final positions, we can take an arbitrary set of processes S.

Lemma 5.21. *For all formula Φ in $\mathcal{L}_G$, $[\Phi] \subseteq ||\Phi||_G$ and $\mathrm{T}^\Phi \varkappa_\Phi \Vdash_G \Phi$.*

Proof. We prove the result by induction on Φ.

Suppose that Φ is a first kind $\mathcal{L}_G$-formula:

$$\Phi := \forall\vec{x}(\delta_1(x_1),\dots,\delta_r(x_r),\varphi_1(\vec{x}),\dots,\varphi_h(\vec{x}) \to A(\vec{x}))$$

where $\vec{x} = x_1,\dots,x_p$.

By definition

$$[\Phi] = \{d_1\dots d_r.\mathrm{T}^{\varphi_1}\varkappa_{\varphi_1(\vec{m})}\dots\mathrm{T}^{\varphi_h}\varkappa_{\varphi_h(\vec{m})}.\pi \mid \vec{m}\in\mathbb{N}^p, d_i\in\mathcal{E}_{\delta_i}(m_i), \pi\in||A(\vec{m})||\}$$

By induction hypothesis, each $\mathrm{T}^{\varphi_i}\varkappa_{\varphi_i(\vec{m})}$ realizes $\varphi_i(\vec{m})$ in $\perp\!\!\!\perp_G$ and every canonical representation of a data type realizes the data type: if $d_i \in \mathcal{E}_{\delta_i}(m_i)$, then $d_i \Vdash \delta_i(m_i)$. Then, we have that $[\Phi] \subseteq ||\Phi||_G$

By storage operators definition, in order to prove $\mathrm{T}^\Phi\varkappa_\Phi \Vdash_G \Phi$, it suffices to prove that $\varkappa_\Phi \Vdash_G \widetilde{\Phi}$, where

$$\widetilde{\Phi} := \forall\vec{x}(\mathcal{E}_{\delta_1}(m_1);\dots;\mathcal{E}_{\delta_r}(m_r);\varphi_1(\vec{x}),\dots,\varphi_h(\vec{x}) \to A(\vec{x}))$$

Take a stack $\pi\in||\widetilde{\Phi}||_G$. If the formula Φ has not hypothesis block (i.e. h=0), then $||\widetilde{\Phi}||_G = [\Phi]$ and then $\varkappa_\Phi\star\pi\in G\subseteq\perp\!\!\!\perp_G$. If the formula Φ has hypothesis block, then there are p integers $\vec{m} = m_1,\dots,m_p$, r canonical representations $d_i,\dots,d_r$, h terms $\xi_1,\dots,\xi_h$ and a stack π' such that:

(1) For all $i\in[1..r]$, $d_i\in\mathcal{E}_{\delta_i}(m_i)$
(2) For all $j\in[1..h]$, $\xi_j\Vdash_G\varphi_j(\vec{m})$
(3) $\pi'\in||A(\vec{m})||$
(4) $\pi = d_1\dots d_r.\xi_1\dots\xi_h.\pi'$

We assert that $\varkappa_\Phi\star\pi$ has a winning strategy: It suffices that ⓔ chooses $\vec{m}$. This move satisfies the condition $\pi'\in||A(\vec{m})||$ by (3), then $\vec{m}$ is a legal move for $\textcircled{\exists}$. After that, $\textcircled{\forall}$ must choose an index $j\in[1..h]$ and a stack $\rho\in[\varphi_j(\vec{m})]$. If it is impossible (i.e. $[\varphi_j(\vec{m})]$ is empty), $\varkappa_\Phi\star\pi\in G\subseteq\perp\!\!\!\perp_G$, thus proving the result. If it is possible, then $\varkappa_\Phi\star\pi\succ\!\succ\xi_j\star\rho$ and this process belongs to $\perp\!\!\!\perp_G$ by induction hypothesis.

Now, suppose that Φ is a second kind $\mathcal{L}_G$-formula:

$$\Phi := \forall W\forall\vec{x}(\delta_1(x_1),\dots,\delta_r(x_r),\varphi_1(W,\vec{x}),\dots,\varphi_h(W,\vec{x}) \to W\vec{\tau})$$

where $\vec{x} = x_1,\dots,x_p$ and W has arity a.

By definition

$$[\Phi] = \{d_1\dots d_r.\mathrm{T}^{\varphi_1}\varkappa_{\varphi_1(\mathbb{P},\vec{m})}\dots\mathrm{T}^{\varphi_h}\varkappa_{\varphi_h(\mathbb{P},\vec{m})}.\pi \mid$$
$$\vec{m}\in\mathbb{N}^p, \mathbb{P}\in\Delta_a, d_i\in\mathcal{E}_{\delta_i}(m_i), \pi\in\mathbb{P}(\vec{\tau})\}$$

By induction hypothesis, for each $i\in[1..h]$, $\mathrm{T}^{\varphi_i}\varkappa_{\varphi_i(\mathbb{P},\vec{m})}\Vdash_G\varphi_i(\mathbb{P},\vec{m})$. Furthermore, every canonical representation $d_i\in\mathcal{E}_{\delta_i}(m_i)$ realizes $\delta_i(m_i)$ in $\perp\!\!\!\perp_G$. In consequence, $[\Phi]\subseteq||\Phi||_G$.

Again using the storage operators definition, we can prove that $\varkappa_\Phi\Vdash_G\widetilde{\Phi}$ where

$$\widetilde{\Phi} = \forall W\forall\vec{x}(\mathcal{E}_{\delta_1}(x_1);\dots;\mathcal{E}_{\delta_r};\varphi_1(W,\vec{x}),\dots,\varphi_h(W,\vec{x}) \to W\vec{\tau})$$

thus concluding that $\mathrm{T}^\Phi\varkappa_\Phi\Vdash_G\Phi$

Take a stack $\pi \in ||\widetilde{\Phi}||_G$. If Φ has not hypothesis block, then π is $d_1 \dots d_r.\pi'$ where each d_i belongs to $\mathcal{E}_{\delta_i}(m_i)$ and π' belongs to $\mathbb{W}(\vec{\tau})$ for a suitable predicate $\mathbb{W}$ and a suitable p-uplet of integers $\vec{m}$. Since π' belongs to $\Delta_a^{\pi',\vec{\tau}}(\vec{\tau})$, we have that π belongs to $[\Phi]$ and hence $\mathrm{T}^{\Phi}\varkappa_{\Phi} \star \pi \in G \subseteq \perp\!\!\!\perp_G$. If Φ has hypothesis block, then there are an a-ary predicate $\mathbb{W}$, a p-uplet $\vec{m}$, r canonical representations $d_1, \dots, d_r$, h terms $\xi_1, \dots, \xi_h$ and a stack π' such that:

(a) For all $i \in [1..r]$, $d_i \in \mathcal{E}_{\delta_i}(m_i)$
(b) For all $j \in [1..h]$, $\xi_j \Vdash_G \varphi_j(\mathbb{W}, \vec{m})$
(c) $\pi' \in \mathbb{W}(\vec{\tau})$
(d) $\pi = d_1 \dots d_r.\xi_1 \dots \xi_h.\pi'$

We assert that $\varkappa_{\Phi} \star d_1 \dots d_r.\xi_1 \dots \xi_h.\pi'$ has a winning strategy: Indeed, to win $\exists$ can choose the predicate $\mathbb{P} := \Delta_a^{\pi',\vec{\tau}}$ and the p-uplet $\vec{m}$. This move satisfies that $\pi' \in \mathbb{P}(\vec{\tau})$ by definition and hence $(\vec{m}, \mathbb{P})$ is a legal move. After that, $\forall$ must choose an index $j \in [1..h]$ and a stack $\rho \in [\varphi_j(\mathbb{P}, \vec{m})]$. If it is impossible, $\varkappa_{\Phi} \star \pi \in G \subseteq \perp\!\!\!\perp_G$, thus proving the result. If it is possible, then $\varkappa_{\Phi} \star \pi \succ\!\succ \xi_j \star \rho$. By induction hypothesis, we have that $[\varphi_j(\mathbb{P}, \vec{m})] \subseteq ||\varphi_j(\mathbb{P}, \vec{m})||_G$. Since $\pi' \in \mathbb{W}(\vec{\tau})$, we have $||\varphi_j(\mathbb{P}, \vec{m})||_G \subseteq ||\varphi_j(\mathbb{W}, \vec{m})||_G$ (c.f. 5.13). In consequence, $\rho \in ||\varphi_j(\mathbb{W}, \vec{m})||_G$ and then $\xi_j \star \rho \in \perp\!\!\!\perp_G$. □

Definition 5.22. *Let us consider a Λ_c-term τ and a $\mathcal{L}_G$ parametrical formula parameters of which are taken from Δ. We say that τ has a winning strategy for Φ if and only if for each stack $\pi \in [\Phi]$ we have that $\tau \star \pi \in \perp\!\!\!\perp_G$.*

Now, we can deduce the *Main Theorem* about realizability games:

Theorem 5.23. *If Φ is a $\mathcal{L}_G$ formula the parameters of which are taken from Δ, and a Λ_c-term τ such that $\tau \Vdash \Phi$. Then τ has a winning strategy for Φ. If the formula Φ is completely relativized to data types, τ implements a winning strategy for the game associated with Φ.*

Proof. This is a direct consequence of the lemma above, because $\pi \in ||\Phi||_G$ and then $\tau \star \pi$ belongs to $\perp\!\!\!\perp_G$. Then, there is a winning strategy for the process $\tau \star \pi$. In particular, as we viewed in 5.17, for a formula completely relativized to data types, this implies that τ implements a winning strategy for the game associated with Φ. □

As we have seen in 5.5, we can embed the first order language $\mathcal{L}_1$ into the language of games $\mathcal{L}_G$ in such a way that the truth values in realizability models are preserved. In addition, the statement of an arithmetical theorem is an $\mathcal{L}_1$-formula. This formula relativized to int is provable in our system of types (c.f. [6]). The term associated with this proof realizes the corresponding $\mathcal{L}_G$-formula and therefore it implements the game associated with the formula and plays a winning strategy for $\exists$ in such a game.

6. **Examples and applications**

Notation 6.1. Let Φ be a parametrical $\mathcal{L}_G$-formula. We will denote by Φ_i the i-th hypothesis of Φ; thus, we have $\Phi = \forall\vec{x}(\delta_1,\dots,\delta_h,\Phi_1,\dots,\Phi_r \to A)$ if Φ is a first kind formula (resp. $\forall W\forall\vec{x}(\delta_1,\dots,\delta_h,\Phi_1,\dots,\Phi_r \to A)$ if Φ is a second kind formula). Hence, we will denote by Φ_{ij} the j-th hypothesis of Φ_i and so on.

6.1. **The formula** $\exists x\forall y(f(x,y)=0)$.

We start with an analogous result to the one of 4.4:

Lemma 6.2. *Consider* $(\mathfrak{M},\perp\!\!\!\perp_0)$ *a realizability model, a parametrical formula* $\Psi := \exists x\varphi(x)$, $\mathbb{P} := \{\pi\}$ *–i.e.* $\mathbb{P}$ *is a* 0*-ary predicate–, and* τ *a* Λ_c*-term such that* $\tau \Vdash_0 \Psi$ *and* $\perp\!\!\!\perp_0 \subseteq (\mathrm{th}_{\tau\star\varkappa_{\Psi_1(\mathbb{P})}.\pi})^c$. *Then, there is an individual* $m \in M$ *and a* Λ_c*-term* ξ *such that* $\xi \Vdash_0 \varphi(m)$ *and* $\varkappa_{\Psi_l(\mathbb{P})}\star\xi.\pi \notin \perp\!\!\!\perp_0$.

Proof. The proof is similar to the proof of 4.4 and is left as an exercise. □

Example 6.3. A first example is the game for $\Psi = \exists x\forall y(f(x,y)=0)$ where f is a function symbol. Let us consider a ω-model $\mathfrak{M} = (\mathbb{N},\mathcal{D},\mathscr{Y})$, a Λ_c-term τ such that $\tau \Vvdash \Psi$ and a stack π.

Define the 0-ary predicate $\mathbb{P} := \{\pi\}$. Then we have that $\tau \Vdash \Psi_1(\mathbb{P}) \to \mathbb{P}$, where $\Psi_1(\mathbb{P}) = \forall x(\forall y(f(x,y)=0) \to \mathbb{P})$. Since $[\![\Psi_1(\mathbb{P}) \to \mathbb{P}]\!] = \{\varkappa_{\Psi_1(\mathbb{P})}.\pi\}$, by 5.23 the process $\tau\star\varkappa_{\Psi_1(\mathbb{P})}.\pi$ has a winning strategy for $\textcircled{$\exists$}$. We will explain this strategy and its intermediate reduction steps.

For brevity, denote k_0 instead of $\varkappa_{\Psi_1(\mathbb{P})}$. We will construct by induction (as in 4.17) a finite sequence of threads models.

Let us define $\perp\!\!\!\perp_0 := (\mathrm{th}_{\tau\star k_0.\pi})^c$. By 6.2, there is an individual n_1 and a Λ_c-term ξ_1 such that $k_0\star\xi_1.\pi \notin \perp\!\!\!\perp_0$ and $\xi_1 \Vdash_0 \forall y(f(n_0,y)=0)$. By 5.15, $\textcircled{$\exists$}$ must choose an individual m_1 (since $\pi \in \mathbb{P}$ each move is a legal move) and $\textcircled{$\forall$}$ must answer a stack belonging to the truth value $[\![\forall y(f(m_1,y){=}0)]\!] = [\![\forall W\forall y(W f(m_1,y){\to}W0)]\!] = \{\varkappa_{\mathbb{W}f(m_1,p)}.\pi' | p{\in}\mathbb{N}, \mathbb{W}{\in}\Delta_1 \text{ and } \pi'{\in}\mathbb{W}(0)\}$. Consider that $\varkappa_{\mathbb{W}_1 f(m_1,p_1)}.\pi_1$ is the answer of $\textcircled{$\forall$}$ and for brevity write it as $k_1.\pi_1$.

Now, suppose we have:

$$\begin{aligned} \tau\star k_0.\pi &\succ k_0\star\xi_1.\pi \\ \xi_1\star k_1.\pi_1 &\succ k_0\star\xi_2.\pi \\ &\vdots \\ \xi_{j-1}\star k_{j-1}.\pi_{j-1} &\succ k_0\star\xi_j.\pi \end{aligned} \tag{6.3.10}$$

and define the realizability model $\perp\!\!\!\perp_{j-1} = (\mathrm{th}_{\tau\star k_0.\pi})^c \cap \bigcap_{i=1}^{j-1}(\mathrm{th}_{\xi_i\star k_i.\pi_i})^c$. Denote as m_j the j-th move of $\textcircled{$\exists$}$ and as $k_j.\pi_j$ the j-th answer of $\textcircled{$\forall$}$.

Define the realizability model $\perp\!\!\!\perp_j$ as $\perp\!\!\!\perp_{j-1} \cap (\mathrm{th}_{\xi_j\star k_j.\pi_j})^c$. Now, there are two cases:

(1) Suppose there are two integers i and n such that $\xi_i {\Vdash_j} \forall y(f(n,y){=}0)$. In particular, for each $p \in \mathbb{N}$ and for each predicate $\mathbb{X} \in \mathcal{D}$, the term ξ_i verifies $\xi_i \Vdash_j \mathbb{X}(f(n,p)) \to \mathbb{X}(0)$. Let us define $\mathbb{X} := \Delta_1^{\pi_i,0}$.

Since by construction $\xi_i \star k_i.\pi_i \notin \perp\!\!\!\perp_j$ and $\pi_i \in \Delta_1^{\pi_i,0}(0)$, we have that for all $p \in \mathbb{N}$, $k_i \nVdash_j \Delta_1^{\pi_i,0}(f(n,p))$. This implies that $\mathfrak{M} \models \forall p(f(n,p)=0)$. Then $k_i \star \pi_i \notin \perp\!\!\!\perp_j$ and this is possible only if $k_i \star \pi_i \in \mathrm{th}_{\xi_j \star k_j.\pi_j}$, because each other thread ends with k_0 in head position. Since k_i is a halt instruction, the execution stops. If $\mathfrak{M} \models f(n_i,p_i) = 0$, then $\textcircled{$\exists$}$ wins. In particular, if $\textcircled{$\exists$}$ has played in the i-th thread, the move $m_i = n$, $\textcircled{$\exists$}$ has win. The scheme 6.2 is completed by addition of the (last) line $\xi_j \star k_j.\pi_j \succ k_i \star \pi_i$.

(2) Suppose that for each integer i from 1 to j and for each n, the term ξ_i does not realize $\forall y(f(n,y)=0)$ in $\perp\!\!\!\perp_j$. Applying 6.2 for $\perp\!\!\!\perp_j$, we obtain an individual n and a Λ_c-term ξ_{j+1} such that $\xi_{j+1} \Vdash_j \forall y(f(n,y) = 0)$ and $k_0 \star \xi_{j+1}.\pi \notin \perp\!\!\!\perp_j$. Hence, for each $i \leq j$ the term ξ_j is different from ξ_i and then $k_0 \star \xi_j.\pi \in \mathrm{th}_{\xi_j \star k_j.\pi_j}$. Here player $\textcircled{$\exists$}$ must play an individual m_{j+1} (all move is legal for $\textcircled{$\exists$}$). $\textcircled{$\forall$}$ must play a stack $k_{j+1}.\pi_{j+1}$ (all answer is legal for $\textcircled{$\forall$}$). The line $\xi_j. \star k_j.\pi_j \succ k_0 \star \xi_{j+1}.\pi$ is added to the scheme 6.3.10 and the construction continues.

Since $\tau \star k_0.\pi$ has a winning strategy for $\textcircled{$\exists$}$ and player $\textcircled{$\forall$}$ has always a legal answer in the case (2), the inductive construction above must stop and in consequence $\mathfrak{M} \models \exists x \forall y(f(x,y) = 0)$. On the other hand, the reduction does not consider the moves of $\textcircled{$\exists$}$ nor the answers of $\textcircled{$\forall$}$. In consequence, there is an integer i such that, if both players plays legal moves, the execution stops always in a process $k_i \star \pi_i$ that "points" to the i-th played move and the i-th played answer. Player $\textcircled{$\exists$}$ wins if and only if $\mathfrak{M} \models f(n_i,p_i) = 0$. In particular, a winning strategy for $\textcircled{$\exists$}$ consists in playing in the i-th move an integer n such that $\mathfrak{M} \models \forall y(f(n,y) = 0)$. Notice that there is a semantic statement equivalent to the stop of execution: *The process stops in the j-th step if and only if there are two integers i and n such that* $1 \leq i < j$ *and* $\xi_i \Vdash_j \forall y(f(n,y)=0)$. Thus, we have described the threads when we execute a winning strategy given by a term τ such that $\tau \Vvdash \exists x \forall y(f(x,y) = 0)$. On the other hand, the converse of 5.23 is also true for the formula Ψ –and many other, we will treat this problem later–. Indeed: given a term that has a winning strategy for the formula $\exists x \forall y(f(x,y) = 0)$, we will see that it realizes the formula in all realizability models.

Lemma 6.4. *Suppose that* $\mathfrak{M} = (\mathbb{N}, \mathcal{D}, \mathscr{Y})$ *is an* ω*-model and* τ *is a* Λ_c*-term. Then, the following statements are equivalent:*

(1) $\tau \Vvdash \Psi$
(2) τ *implements a play having a winning strategy for* $\textcircled{$\exists$}$ *in the game associated with* $\Psi := \exists x \forall y(f(x,y) = 0)$.

Proof.

(1)→(2) By 5.23

(2)$\to$(1) By hypothesis τ implements a winning strategy for Ψ and this implies there is an integer q a sequence of constants $k_0, \dots k_q$, a sequence of stacks $\pi_0, \dots, \pi_q$ and a sequence of Λ_c-terms $\xi_1, \dots, \xi_q$ such that

$$\tag{6.4.11} \begin{aligned} \tau \star k_0.\pi_0 &\succ k_0 \star \xi_1.\pi_0 \\ \xi_1 \star k_1.\pi_1 &\succ k_0 \star \xi_2.\pi_0 \\ &\vdots \\ \xi_q \star k_q.\pi_q &\succ k_i \star \pi_i \end{aligned}$$

for an integer $i \leq q$. Moreover, the winning strategy for $\exists$ must play in the i-th line an individual m_i such that $\mathfrak{N} \models \forall y(f(m_i,y) = 0)$ because otherwise, $\forall$ wins playing p_i such that $\mathfrak{N} \models f(n_i, p_i) \neq 0$.

Consider a realizability model $\perp\!\!\!\perp$ and a 0-ary predicate $\mathbb{P} \subseteq \Pi$. Take a Λ_c-term t_0 realizing $\Psi_1(\mathbb{P})$ and a stack ρ_0 belonging to $\mathbb{P}$. We must prove that $\tau{\star}t_0.\rho_0{\in}\perp\!\!\!\perp$.

By substitution in 6.4.11, there is a Λ_c-term $\widetilde{\xi}_1$ such that $\tau \star t_0.\rho_0$ reduces to $t_0 \star \widetilde{\xi}_1.\rho_0$. Here $\widetilde{\xi}_1$ is in fact $\xi_1[{}^{t_0,\rho_0}/_{k_0,\pi_0}]$.

Let us consider a winning strategy which wins on the play 6.4.11 implemented by τ and consider n_1, the first move proposed by $\exists$ according with this strategy. Since $\Psi_1(\mathbb{P})$ is $\forall x(\forall y(f(x,y) = 0) \to \mathbb{P})$, it suffices to prove the following **claim:**

$$\widetilde{\xi}_1 \Vdash \forall y(f(n_1,y) = 0)$$

Let us consider a 1-ary predicate $\mathbb{W}_1 \in \mathcal{D}$, an individual p_1, a Λ_c-term $t_1 \Vdash \mathbb{W}(f(n_1, p_1))$ and a stack $\pi \in \mathbb{W}(0)$ and try to prove that $\widetilde{\xi}_1 \star t_1.\pi_1 \in \perp\!\!\!\perp$. Again by substitution on 6.4.11 we have that $\widetilde{\xi}_1 \star t_1.\pi_1 \succ t_0 \star \widetilde{\xi}_2.\rho_0$ where $\widetilde{\xi}_2{=}\xi_2[{}^{t_0,t_1,\rho_0,\rho_1}/_{k_0,k_1,\pi_0,\pi_1}]$. If we prove $\widetilde{\xi}_2 \Vdash (\forall y(f(n_2,y) = 0))$ for an integer n_2, the claim is proven. Consider n_2 the second move of $\exists$.

Repeating q times this reasoning, we obtain by substitution in 6.4.11:

$$\tag{6.4.12} \begin{aligned} \tau \star t_0.\rho_0 &\succ t_0 \star \widetilde{\xi}_1.\rho_0 \\ \widetilde{\xi}_1 \star t_1.\rho_1 &\succ t_0 \star \widetilde{\xi}_2.\rho_0 \\ &\vdots \\ \widetilde{\xi}_q \star t_q.\rho_q &\succ t_i \star \rho_i \end{aligned}$$

Since $\mathfrak{N} \models f(n_i, p_i) = 0, t_i \Vdash \mathbb{W}_i(f(n_i, p_i))$ and $\rho_i \in \mathbb{W}_i(0) = \mathbb{W}_i(f(n_i, p_i))$; the process $t_i \star \rho_i$ belongs to $\perp\!\!\!\perp$. In consequence, for all j such that $1 \leq j \leq q$, the process $\widetilde{\xi}_j \star t_j.\rho_j$ belongs to $\perp\!\!\!\perp$ and hence $\widetilde{\xi}_j \Vdash \forall y(f(n_j,y) = 0)$ in $\perp\!\!\!\perp$. Taking $j = 1$ the claim is proven, thus proving the result.

□

6.2. **The formula** $\exists x \forall y(f(x,y) \neq 0)$**.**

Example 6.5. As another example, we study the case seen in 6.3 but with $\neq$ instead of $=$:

Let us consider the formula $\Psi := \exists x \forall y(f(x,y) \neq 0)$ and an ω-model $\mathfrak{M} = (M, \mathcal{D}, \mathscr{Y})$. By 1.12

$$\Psi = \forall X(\forall x(\forall y f(x,y) \neq 0 \to X) \to X)$$

hence by definition $\Psi_1(X) = \forall x(\forall y f(x,y) \neq 0 \to X)$. As in 6.3, we will write k_0 instead of the constant $\varkappa_{\Psi_1(\{\pi\})}$. Consider τ a Λ_c-term realizing the formula Ψ. Each time k_0 arrives in head position, it is with a stack $\xi_i.\pi$ as argument. At this moment, Ⓔ must play an individual m_i and Ⓐ must answer a stack $\rho_i \in [\forall y(f(m_i,y) \neq 0)] = \bigcup_{p \in \mathbb{N}} [f(m_i,p) \neq 0]$. Here there are two possibilities:

(1) $\mathfrak{M} \models \forall y(f(m_i,y) \neq 0)$. In this case, the set $[f(m_i,p) \neq 0]$ is empty for each integer p and then Ⓐ has no legal answer. By definition 5.19, the process $k_0 \star \xi_i.\pi$ is a final winning position. The game stops and Ⓔ wins.
(2) $\mathfrak{M} \not\models \forall y(f(m_i,y) = 0)$. In this case the set $[f(m_i,p) \neq 0]$ is Π for a suitable integer p. Then, Ⓐ can play any stack ρ_i and the game continues.

By 5.23 the process $\tau \star k_0.\pi$ has a winning strategy for Ⓔ. In consequence $\mathfrak{M} \models \exists x \forall y(f(x,y) \neq 0)$ and Ⓔ can play in such a way that the execution stops. The i-threads other than the first are of the shape $\xi_i \star \rho_i \succ k_0 \star \xi_{i+1}.\pi_0$, where ξ_i, ξ_{i+1} are respectively the i-th and i+1-th exception handler. If Ⓔ wins, the thread corresponding to the winning move is the last thread.

Converse, suppose that $\mathfrak{M} \models \Psi$. We can compute the truth value of Ψ as follows:

Given an integer n such that $\mathfrak{M} \models \forall y \ (f(n,y) \neq 0)$, the truth value of $\forall y(f(n,y) \neq 0)$ is the empty set $\emptyset$, i.e. we can replace $\forall y(f(n,y) \neq 0)$ by $\top$. Likewise, for an integer n such that $\mathfrak{M} \not\models \forall y(f(n,y) \neq 0)$, we can replace $\forall y(f(n,y) \neq 0)$ by $\perp$.

Since $\mathfrak{M} \models \Psi$, the truth value of $\forall x[\forall y(f(n,y) \neq 0) \to X]$ is the set $\bigcup_{n \in \mathbb{N}} ||\forall y(f(n,y) \neq 0) \to X|| = ||\top \to X||$, because $||\top \to X|| \supseteq ||\perp \to X||$. In consequence, the truth value of Ψ is the set $||\forall X[(\top \to X) \to X]||$. Moreover, we can characterize using the threads method, the terms that realizes $\forall X[(\top \to X) \to X]$:

Remark 6.6. The following statements are equivalents :

(1) $\tau \Vdash \forall X[(\top \to X) \to X]$.
(2) For all Λ_c-terms u and for all stacks π, there is a Λ_c-term ξ such that $\tau \star u.\pi \succ u \star \xi.\pi$.

Proof.

(1)$\to$(2) Let us consider $\perp\!\!\!\perp_0 := \text{th}_{\tau\star u.\pi}$ and define the predicate $\mathbb{P} := \{\pi\}$. Since $\tau\star u.\pi \notin \perp\!\!\!\perp_0$, then $u \not\Vdash_0 \top \to \mathbb{P}$, there is a Λ_c-term ξ such that $u\star\xi.\pi \notin \perp\!\!\!\perp$ and hence $\tau\star u.\pi \succ u\star\xi.\pi$.

(2)$\to$(1) Is left as an exercise.

□

Corollary 6.7. *A Λ_c-term τ satisfies $\tau \Vdash \exists x\forall y(f(x,y) \neq 0)$, where $\mathfrak{M} \models \Psi$, if and only if, for all Λ_c-terms u and for all stacks π, there is a Λ_c-term ξ satisfying $\tau\star u.\pi \succ u\star\xi.\pi$*

By instance, if $\mathfrak{M} \models \Psi$, then

- $\tau_1 = \lambda x(x)H$
- $\tau_2 = \lambda x(x)(k_0)(k_1)\ldots(k_r)H$
- $\tau_3 = \lambda x(x)\,\text{id}$

(where H is a halt instruction) realizes Ψ. The first term gives a thread scheme with only one thread. Hence, the only one winning strategy for $\textcircled{\exists}$ is to play a good move the first time it plays. τ_2 gives $\textcircled{\exists}$ at most r possibilities to play correctly. Finally, τ_3 is more complicated: The only winning strategy is to play correctly in the first move, because if n_1 is the first move of $\textcircled{\exists}$ and $\mathfrak{M} \not\models \forall y(f(n_1,y) = 0)$, then $\textcircled{\forall}$ can answer a Π-constant π_i. Since $\text{id}\star\pi_i$ does not reduce, in this case $\textcircled{\exists}$ has no more opportunities. But, maybe $\textcircled{\forall}$ is gentle and it proposes $(k_0)\,\text{id}$, thus giving a new opportunity to $\textcircled{\exists}$. Playing the game implemented by τ_1 or τ_2, $\textcircled{\forall}$ cannot modify the maximum length of the play.

6.3. **Relativizing formulæ to data types.**

Until now, in this section we described games where its implementation acts as a referee. Indeed, if we analyze the execution behaviour on 6.3, we can conclude that:

- The play length is determined by the term (the program).
- The program never plays a relevant value, but exception handlers to continue the execution.
- The only trace of the moves played are the constants introduced each time $\textcircled{\forall}$ plays. Although this information is implicit in these constants, if we have no instructions able to “read” it, it is not available for calculus.
- Once the play is ended, the players can verify which is the winner. Nothing along the execution acts as a test to determine the winner.

Intuitively, we can imagine a play pacted in q moves between two players. The program will choose a move (the i-th in 6.4.12), independently of the moves played and players do not know which one. Once the play is finished, the referee show its chosen move and players must verify which was the winner.

Consider also the second one treated case (6.5): when we substitute $=$ by $\neq$.

The reader should remark that, while the formulæ $\exists x \forall y (f(x,y) = 0)$ and $\exists x \forall y (f(x,y) \neq 0)$ have the same expressiveness (all formula of first shape is equivalent to a formula of the second one and conversely), its specifications are very different. The choice of $\neq$ instead of $=$ gives sessions the length of which depends upon the moves and these sessions will be broken whenever $\textcircled{\exists}$ wins.

Reasonning by analogy with the case of $\exists x \forall y (f(x,y) = 0)$, we can describe the game of $\exists x \forall y (f(x,y) \neq 0)$ as follows:

The play between $\textcircled{\exists}$ and $\textcircled{\forall}$ is performed until $\textcircled{\exists}$ wins, in which case the execution is broken. In consequence, here the referee acts as a test, because it stops the execution if and only if $\textcircled{\exists}$ has win.

We will examine cases where the term acts as a referee and as a player. More precisely, it will decide when the play is finished and it will play instead of $\textcircled{\exists}$. The relativization to a data type gives this kind of behaviour. We will consider relativizations to naturals. But, we can do at least two possible relativizations: Relativization to the formula $\mathrm{int}(x)$ and relativization to the canonical representations $\{\overline{x}\}$. These two cases are essentially the same, but the case of a relativization to the formula $\mathrm{int}(x)$ requires we use storage operators for interaction constants.

In order to analyze simultaneously these two cases, we will describe the threads scheme of a play implemented by a realizer of

$$\widehat{\Psi} = \exists^{\{\overline{x}\}} x \forall^{\{\overline{y}\}} y (f(x,y) = 0)$$

and after that, we will explain how is the thread scheme for a realizator of

$$\Psi = \exists^{\mathrm{int}} x \forall^{\mathrm{int}} y (f(x,y) = 0)$$

More generally, we will study the transformation $\Psi \mapsto \widehat{\Psi}$ consisting to replace $\delta(x)$ by $\mathcal{E}_\delta(x)$ –where $\delta(x)$ is a data type– and how we can obtain a realizator of $\widehat{\Psi}$ from a realizator of Ψ and conversely.

In order to study the case of $\widehat{\Psi}$, we must use the analogous of 4.4 for an extended formula $\exists^{\{\overline{x}\}} x \varphi(x)$.

Lemma 6.8. *Let us consider an extended formula $\varphi(x)$, a realizability model $(\mathfrak{M}, \perp\!\!\!\perp_0)$, a stack π and two Λ_c-terms u and h such that $\perp\!\!\!\perp_0 \subseteq (\mathrm{th}_{u\star h.\pi})^c$ and $u \Vdash_0 \exists^{\{\overline{x}\}} x \varphi(x)$. Then, there is an integer n and a Λ_c-term ξ such that $\xi \Vdash_0 \varphi(n)$ and $h \star \overline{n}.\xi.\pi \notin \perp\!\!\!\perp_0$.*

Proof. Analogous to the one of 4.4 $\square$

We have seen that a term realizing $\widehat{\Psi}$ implements a winning strategy for the game associated with $\widehat{\Psi}$ (c.f. 5.23). These strategies involves backtracking and we will describe this one, giving also a semantic condition equivalent to the execution stop.

Example 6.9. Let us consider an ω-model $\mathfrak{M} := (\mathbb{N}, \mathcal{D}, \mathscr{Y})$ and a Λ_c-term $u \Vvdash \exists^{\{\overline{x}\}} x \forall^{\{\overline{y}\}} y (f(x,y) = 0)$. We will build recursively –as in 6.3– a finite sequence of thread models.

Consider a halt instruction sequence $(k_i)_{i\in\mathbb{N}}$ and a Π-constant sequence $(\pi_i)_{i\in\mathbb{N}}$.

Denote as th_0 the thread $\mathrm{th}_{u\star k_0.\pi}$ and define $\perp\!\!\!\perp_0 := (\mathrm{th}_0)^c$. We can apply 6.8 to obtain an individual n_1 and a Λ_c-term ξ_1 such that $\xi_1 \Vdash_0 \forall y(f(n_1,y)=0)$ and $k_0\star\overline{n}_1.\xi_1.\pi_0 \in \mathrm{th}_0$. Hence $u\star k_0.\pi_0 \succ k_0\star\overline{n}_1.\xi_1.\pi_0$.

Now we proceed by induction. Suppose that $n_1,\ldots,n_j$ are integers and $\xi_1,\ldots,\xi_j$ are Λ_c-terms such that

$$\begin{aligned} u\star k_0.\pi_0 &\succ k_0\star\overline{n}_1.\xi_1.\pi_0\\ \xi_1\star\overline{p}_1.k_1.\pi_1 &\succ k_0\star\overline{n}_2.\xi_2.\pi_0\\ &\vdots\\ \xi_{j-1}\star\overline{p}_{j-1}.k_{j-1}.\pi_{j-1} &\succ k_0\star\overline{n}_j.\xi_j.\pi_0 \end{aligned} \tag{6.9.13}$$

For each $l>0$, denote as th_l the thread $\mathrm{th}_{\xi_l\star\overline{p}_l.k_l.\pi_l}$ and as $\perp\!\!\!\perp_j$ the threads model $\bigcap_{l=0}^{j}(\mathrm{th}_l)^c$. Applying 6.8 with $\perp\!\!\!\perp_j$ we obtain an integer n_{j+1} and a Λ_c-term ξ_{j+1} such that $\xi_{j+1}\Vdash_j \forall^{\{\overline{y}\}}y(f(n_{j+1},y)=0)$ and $k_0\star\overline{n}_{j+1}.\xi_{j+1}.\pi_0$ does not belong to $\perp\!\!\!\perp_j$.

Here there are two possibilities:

(1) There is an integer $i\in[1..j]$ such that $k_0\star\overline{n}_{j+1}.\xi_{j+1}.\pi_0$ belongs to th_{i-1}. In this case $n_i=n_{j+1}$ and $\xi_i=\xi_{j+1}$ because k_0 can only be in head position at the end of a thread. In particular, the last implies that $\xi_i\Vdash_j\forall^{\{\overline{y}\}}y(f(n_i,y)=0)$ –i.e. the formula $\forall X\forall^{\{\overline{y}\}}y(X\,f(n_i,y)\to X\,0)$–. Take $X:=\Delta^{\pi_i,0}$ and $y:=p_i\in\mathbb{N}$. Since $\xi_i\star\overline{p}_i.k_i.\pi_i\notin\perp\!\!\!\perp_j$, then k_i does not realize $\Delta^{\pi_i,0}(f(n_i,p_i))$. In consequence $\mathfrak{M}\models f(n_i,p_i)=0$ and $k_i\star\pi_i\notin\perp\!\!\!\perp_j$. The only thread belonging to $(\perp\!\!\!\perp_j)^c$ such that k_i can be in head position is th_j, because it must be at the end of its thread and the other threads finishes with k_0 in head position. We add to 6.9.13 the line $\xi_j\star\overline{p}_j.k_j.\pi_j\succ k_i\star\pi_i$ and, taking $j=q$, we obtain:

$$\begin{aligned} u\star k_0.\pi_0 &\succ k_0\star\overline{n}_1.\xi_1.\pi_0\\ \xi_1\star\overline{p}_1.k_1.\pi_1 &\succ k_0\star\overline{n}_2.\xi_2.\pi_0\\ &\vdots\\ \xi_q\star\overline{p}_q.k_q.\pi_q &\succ k_i\star\pi_i \end{aligned} \tag{6.9.14}$$

And the execution stops.

(2) $k_0\star\overline{n}_{j+1}k_{j+1}.\pi_0$ belongs to th_j. In this case we define th_{j+1} as the thread of $\xi_{j+1}\star\overline{p}_{j+1}.k_{j+1}.\pi_{j+1}$, we add to 6.9.13 the line

$$\xi_j\star\overline{p}_j.\pi_j\succ\xi_{j+1}\star\overline{p}_{j+1}.k_{j+1}.\pi_{j+1}$$

and the execution continues.

By hypothesis u implements a winning strategy for the game associated with the extended formula $\widehat{\Psi}=\exists^{\{\overline{x}\}}x\forall^{\{\overline{y}\}}y(f(x,y)=0)$ and then, if we take a stack π and we denote as $\mathbb{P}$ the predicate $\{\pi\}$, the process $u\star\varkappa_{\widehat{\Psi}_1(\mathbb{P})}.\pi$ has a winning strategy for player $\textcircled{\exists}$. In consequence, if we replace k_0 by $\varkappa_{\widehat{\Psi}_1(\mathbb{P})}$ and π_o by π in 6.9.13, we have that the recursive process described in (1) and (2) must stop.

Lemma 6.10. *Consider an* ω*-model* $\mathfrak{M}$, *a* Λ_c*-term* u *and an extended formula* $\widehat{\Psi} = \exists^{\{\overline{x}\}}x\forall^{\{\overline{y}\}}y(f(x,y)=0)$. *Then the following statements are equivalents:*

(1) $u \Vdash \widehat{\Psi}$.
(2) *u implements a winning strategy for* $\textcircled{\exists}$ *in the game associated with the extended formula* $\widehat{\Psi}$.

Proof.

(1)$\to$(2) By 5.23.

(2)$\to$(1) The proof is almost the same the one we viewed in 6.4. However here the length of a play depends upon the answers of $\textcircled{\forall}$. For this reason, we can not consider the thread scheme as given. In consequence, we give a proof slightly more complicated.

Take a realizability model $\perp\!\!\!\perp$, a predicate $\mathbb{P} \subseteq \Pi$, a Λ_c-term t_0 realizing $\widehat{\Psi}_1(\mathbb{P})$ in $\perp\!\!\!\perp$ and a stack ρ_0 belonging to $\mathbb{P}$. We must prove that $u \star t_0.\rho_0 \in \perp\!\!\!\perp$. Trying to prove it, we will consider a play $\mathcal{M}$ between $\textcircled{\exists}$ and $\textcircled{\forall}$ according to the strategy given by u. By hypothesis this play must finish with $\textcircled{\exists}$ winning:

Start the play $\mathcal{M}$ implemented by u. By substitution in the first thread, we have that $u \star t_0.\rho_0 \succ t_0 \star \overline{n}_1.\widetilde{\xi}_1.\rho_0$ where n_1 is the first move played by $\textcircled{\exists}$ and $\widetilde{\xi}_1 = \xi_1[{t_0,\rho_0}/{k_0,\pi_0}]$. In order to prove the result it suffices to prove the following **claim:**

$$\widetilde{\xi}_1 \Vdash \forall^{\{\overline{y}\}}y(f(n_1,y)=0)$$

Take a stack $\overline{p}_1.t_1.\rho_1 \in ||\forall^{\{\overline{y}\}}y(f(n_1,y)=0)||$ and add n_2 to $\mathcal{M}$ as its $\textcircled{\exists}$ second move. Then (by substitution in the second thread of this play), the process $\widetilde{\xi}_1 \star \overline{p}_1.t_1.\rho_1$ reduces to $t_0 \star \overline{n}_2.\widetilde{\xi}_2.\rho_0$, where $\widetilde{\xi}_2 = \xi_2[{t_0,t_1,\rho_0,\rho_1}/{k_0,k_1,\pi_0,\pi_1}]$. In order to prove the claim, it suffices to prove that $\widetilde{\xi}_2 \Vdash \forall^{\{\overline{y}\}}y(f(n_2,y)=0)$.

We proceed to do this construction until the play is ended and $\textcircled{\exists}$ wins. At this moment the thread scheme we build is:

$$\begin{gathered} u\star t_0.\rho_0 \succ t_0\star\overline{n}_1.\widetilde{\xi}_1.\rho_0 \\ \widetilde{\xi}_1\star\overline{p}_1.t_1.\rho_1 \succ t_0\star\overline{n}_2.\widetilde{\xi}_2.\rho_0 \\ \vdots \\ \widetilde{\xi}_q\star\overline{p}_q.t_q.\rho_q \succ t_i\star\rho_i \end{gathered} \tag{6.10.15}$$

Furthermore, we know that $\mathfrak{M} \models f(n_i,p_i)=0$ because $\textcircled{\exists}$ has win. By construction, for each step j, we took $t_j \Vdash \mathbb{W}_j(f(n_j,p_j))$ and $\rho_j \in \mathbb{W}_j(0)$ for 1-ary predicates $\mathbb{W}_j \in \mathcal{D}$. Then $t_i \star \rho_i \in \perp\!\!\!\perp$ and hence each $\widetilde{\xi}_j \star \overline{p}_j.t_j.\rho_j \in \perp\!\!\!\perp$; in particular for $j = 1$. The claim is proved thus proving the result.

□

6.4. **Relativization to data types vs. relativization to canonical representations.**
We generalize now the correspondence that we used to transform Ψ into $\widehat{\Psi}$. Given a $\mathcal{L}_G$-formula Φ, we define the formula $\widehat{\Phi}$ as the one we obtain replacing each instance of a data type by the corresponding Λ_c-set of canonical representations. More precisely:

Definition 6.11. *We define by induction a function $\Phi \mapsto \widehat{\Phi}$ where Φ is a $\mathcal{L}_G$-formula:*

- *If $\Phi = \forall\vec{x}[\delta_1(x_1),\ldots,\delta_r(x_r),\Phi_1,\ldots,\Phi_s \to A]$, then:*
$$\widehat{\Phi} = \forall\vec{x}[\varepsilon_{\delta_1}(x_1);\ldots;\varepsilon_{\delta_r}(x_r),\widehat{\Phi}_1,\ldots,\widehat{\Phi}_s \to A]$$
- *If $\Phi = \forall W\forall\vec{x}[\delta_1,\ldots,\delta_r,\Phi_1,\ldots,\Phi_s \to W\vec{\tau}]$, then:*
$$\widehat{\Phi} = \forall W\forall\vec{x}[\varepsilon_{\delta_1}(x_1),\ldots,\varepsilon_{\delta_r}(x_r),\widehat{\Phi}_1,\ldots,\widehat{\Phi}_s \to W\vec{\tau}]$$

Definition 6.12. *Given a $\mathcal{L}_G$-formula Φ, we define two transformations $(t,\Phi) \mapsto t^*_\Phi$ and $(t,\Phi) \mapsto \widehat{t}_\Phi$ over $\Lambda_c \times \mathcal{L}_G$. These definitions are mutually recursive and defined by induction on Φ. Given a first (resp. second) kind $\mathcal{L}_G$-formula $\Phi = \forall\vec{x}[\delta_1(x_1),\ldots,\delta_r(x_r),\Phi_1,\ldots,\Phi_s \to A]$ (resp. $\Phi := \forall W\forall\vec{x}[\delta_1(x_1)\ldots\delta_r(x_r),\Phi_1\ldots\Phi_s \to W\vec{\tau}]$), then:*
$$\begin{array}{rcl} t^*_\Phi & := & (\mathrm{T}^\Phi)\lambda x_1\ldots\lambda x_r\lambda\xi_1\ldots\lambda\xi_s(t)x_1\ldots x_r\widehat{\xi}_{1\Phi_1}\ldots\widehat{\xi}_{s\Phi_s} \\ \widehat{t}_\Phi & := & \lambda x_1\ldots\lambda x_r\lambda\xi_1\ldots\lambda\xi_s(t)x_1\ldots x_r\xi^*_{1\Phi_1}\ldots\xi^*_{s\Phi_s} \end{array}$$

Lemma 6.13. *Let us consider a $\mathcal{L}_G$-formula Φ and a realizability model $(\mathfrak{M},\perp\!\!\!\perp)$, where $\mathfrak{M}$ is an ω-model. Then:*

(1) *Given a Λ_c-term τ satisfying $\tau \Vdash \Phi$, we have that $\widehat{\tau}_\Phi \Vdash \widehat{\Phi}$*
(2) *Given a Λ_c-term σ satisfying $\sigma \Vdash \widehat{\Phi}$, we have that $\sigma^*_\Phi \Vdash \Phi$.*

Proof. We prove (1) and (2) by induction on Φ. Suppose that Φ is a second kind formula.

To prove (1), lets consider a stack $\pi = e_{1m_1}\ldots e_{rm_r}.\xi_1\ldots\xi_s.\rho \in ||\widehat{\Phi}||$, where e_{im_i} denotes the unique term belonging to $\varepsilon_{\delta_i}(m_i)$, whenever it exist. We must prove that $\widehat{\tau}_\Phi \star \pi \in \perp\!\!\!\perp$. Reducing this process, we obtain
$$\tau \star e_{1m_1}\ldots e_{rm_r}.\xi^*_{1\Phi_1}\ldots\xi^*_{s\Phi_s}.\rho$$
which belongs to $\perp\!\!\!\perp$ applying the induction hypothesis over $\Phi_1\ldots\Phi_s$ and that canonicals realize their data type.

To prove (2), lets consider a stack
$$\pi' = e_{1m_1}\ldots e_{rm_r}.\xi_1\ldots\xi_s.\rho' \in ||\varepsilon_{\delta_1}(m_1);\ldots;\varepsilon_{\delta_r}(m_r);\Phi_1,\ldots,\Phi_s \to \mathbb{W}\vec{\tau}||$$
for a individuals vector $\vec{m}$ and a 1-ary predicate $\mathbb{W}$. By reduction we obtain:

$\lambda x_1\ldots\lambda x_r\lambda\xi_1\ldots\lambda\xi_s(\sigma)x_1\ldots x_r.\widehat{\xi}_{1\Phi_1}\ldots\widehat{\xi}_{s\Phi_s} \star \pi' \succ$

$$\sigma \star e_{1m_1}\dots e_{rm_r}.\widehat{\xi}_{1\Phi_1}\dots\widehat{\xi}_{s\Phi_s}.\rho'$$

By induction hypothesis $\widehat{\xi}_{i\Phi_i} \Vdash \widehat{\Phi}_i$. Then $e_{1m_1}\dots e_{rm_r}.\widehat{\xi}_{1\Phi_1}\dots\widehat{\xi}_{s\Phi_s}.\rho' \in ||\widehat{\Phi}||$. Thus, we have proved that $\lambda x_1\dots\lambda x_r\lambda\xi_1\dots\lambda\xi_s(\sigma)x_1\dots x_r.\widehat{\xi}_{1\Phi_1}\dots\widehat{\xi}_{s\Phi_s}$ realizes $\varepsilon_{\delta_1}(m_1);\dots;\varepsilon_{\delta_r}(m_r);\Phi_1,\dots,\Phi_s \to \mathbb{W}\vec{\tau}$ and hence –using that T^{Φ} si a storage operator– $\sigma^*_{\Phi} \Vdash \delta_1(m_1),\dots,\delta_r(m_r),\Phi_1,\dots,\Phi_s \to \mathbb{W}\vec{\tau}$. Since this reasoning is valid for all $\vec{m}$ and for all $\mathbb{W} \in \mathcal{D}$, we have proved that $\sigma^*_{\Phi} \Vdash \Phi$.
□

Remark 6.14. Given a term $\tau \Vdash \Phi$, it has a winning strategy for the game associated with Φ. Using 6.13, we prove that $\widehat{\tau}_{\Phi} \Vdash \widehat{\Phi}$ and then, this term has a winning strategy for the game associated with $\widehat{\Phi}$. Moreover, if Φ is a totally relativized formula, the strategies implemented by σ and $\widehat{\sigma}_{\Phi}$ are the same (both play the same individuals).

Example 6.15. Let us consider the formula $\Psi = \exists \overset{\mathrm{int}}{x} \forall \overset{\mathrm{int}}{y} f(x,y) = 0$. The sole Ψ hypothesis is the formula $\Psi_1(X) = \forall \overset{\mathrm{int}}{x}(\forall W \forall \overset{\mathrm{int}}{y}(W f(x,y) \to W0) \to X)$. Suppose that u is a Λ_c-term such that $u \Vdash \Psi$ and take a stack π. Denote by k the interaction instruction $\varkappa_{\Psi_l(\{\pi\})}$.

By 5.6 the operator T^{Ψ_1} is the 1-ary storage operator for integers, $\mathrm{T}_{\mathrm{int}}$. By 5.23 the process $u \star \mathrm{T}_{\mathrm{int}}\varkappa_{\Psi_1(\{\pi\})}.\pi$ has a winner strategy for $\textcircled{\exists}$. For brevity, we will denote $\mathrm{T}_{\mathrm{int}}$ as T.

Applying 6.13 and 6.14, the term $\widehat{u}_{\Psi}$ implements the same winning strategy that u, but playing the game of the formula $\widehat{\Psi}$. In consequence, the threads of a play implemented by u are (almost[5]) the same that the threads of a play implemented by $\widehat{u}$:

$$\begin{aligned} &u \star \mathrm{T}_{\mathrm{int}} k_0.\rho_0 \succ k_0 \star \overline{n}_1.\xi_1.\rho_0 \\ &\xi_1 \star \overline{p}_1.k_1.\rho_1 \succ k_0 \star \overline{n}_2.\xi_2.\rho_0 \\ &\qquad\vdots \\ &\xi_q \star \overline{p}_q.k_q.\rho_q \succ k_i \star \rho_i \end{aligned} \tag{6.15.16}$$

However, since our purpose is to compose winning strategies, we will use other threads schemes, more adequate to compose these strategies. We want proceed to substitute the Λ_c-terms $\mathrm{T}_{\mathrm{int}}k$ by Λ_c-terms. In order to formalise this idea, we will replace $\mathrm{T}_{\mathrm{int}}k_0$ terms by *instructions* with a suitable and more restrictive reduction property. After that, using dynamic substitutions, we can replace these instructions by Λ_c-terms provided that these terms have a behaviour analogous to the behaviour of the instructions it replaces (c.f.: section 4). For the moment, we will present adequately threads schemes for this formula:

[5]in fact, the transformations $\widehat{\ }$ and $*$ adds many η-expansions which add irrelevant reductions.

Example 6.16. Consider a Λ_c-term u realizing in all models the formula $\Psi = \exists \overset{\text{int}}{x} \forall \overset{\text{int}}{y}(f(x,y){=}0)$ and a play $\mathcal{N}$ implemented by u in the game associated with Ψ. Take a $\mathrm{T}_{\text{int}}K-$like instruction (c.f. 4.11) U satisfying:

i. u does not contain U.
ii. The function $(\nu,\xi) \mapsto \pi'_{\nu\xi}$ is injective on ξ.

i.e. the hypothesis about U assumed in 4.5 and the injective condition ii. over the map $(\nu,\xi) \mapsto \pi'_{\nu\xi}$. We built by induction a threads scheme associated with the match:

Step 0 Define $\mathrm{th}_0 := \mathrm{th}_{u\star U.\pi_0}$ and $\perp\!\!\!\perp_0 := \mathrm{th}_0^c$. Applying 4.5, we obtain two terms ν_1, ξ_1 and an integer n_1 such that

$$u \star U.\pi_0 \succ H_{\nu_1\xi_1} \star \overline{n}_1.\pi'_{\nu_1\xi_1}$$

and $\xi_1 \Vdash_0 \forall \overset{\text{int}}{y}(f(n_1,y) = 0)$. Start a match between $\textcircled{\exists}$ and $\textcircled{\forall}$ where $\textcircled{\exists}$ proposes n_1 as its first movement. Suppose that $\textcircled{\forall}$ answers p_1.

Step $k{+}1$ Let us consider we have the threads scheme:

$$\begin{aligned} u \star U.\pi_0 &\succ H_{\nu_1\xi_1} \star \overline{n}_1.\pi'_{\nu_1\xi_1} \\ (\xi_1)\overline{p}_1 \star k_1.\pi_1 &\succ H_{\nu_2\xi_2} \star \overline{n}_2.\pi'_{\nu_2\xi_2} \\ &\vdots \\ (\xi_k)\overline{p}_k \star k_k.\pi_k &\succ H_{\nu_{k+1}\xi_{k+1}} \star \overline{n}_{k+1}.\pi'_{\nu_{k+1}\xi_{k+1}} \end{aligned} \tag{6.16.17}$$

where $\textcircled{\exists}$ has played $n_1,\ldots,n_{k+1}$ and $\textcircled{\forall}$ has answered $p_1,\ldots,p_{k+1}$ respectively.

Define $\mathrm{th}_{k+1} := \mathrm{th}_{(\xi_{k+1})p_{k+1}\star k_{k+1}.\pi_{k+1}}$ and $\perp\!\!\!\perp_{k+1} := \perp\!\!\!\perp_k \cap \mathrm{th}^c_{k+1}$. Applying again 4.5, we obtain two terms ν_{k+2}, ξ_{k+2} and an integer n_{k+2} such that:

$$H_{\nu_{k+2}\xi_{k+2}} \star \overline{n}_{k+2}.\pi'_{\nu_{k+2}\xi_{k+2}} \notin \perp\!\!\!\perp_{k+1} \tag{6.16.18}$$

and $\xi_{k+2} \Vdash_{k+1} \forall \overset{\text{int}}{y}(f(n_{k+2},y) = 0)$. Now, there are two possibilities:

(1) Suppose that for each $i \in [1..k{+}1]$ $\xi_i \neq \xi_{k+2}$. Since $(\nu,\xi) \mapsto \pi'_{\nu\xi}$ is injective on ξ, we have that that $\pi'_{\nu_i\xi_i} \neq \pi'_{\nu_{k+2}\xi_{k+2}}$ and in consequence $H_{\nu_{k+2}\xi_{k+2}} \star \overline{n}_{k+2}.\pi'_{k+2} \in \mathrm{th}_{k+2}$. We add to 6.16.17 the line:

$$(\xi_{k+1})\overline{p}_{k+1} \star k_{k+1}.\pi_{k+1} \succ H_{\nu_{k+2}\xi_{k+2}} \star \overline{n}_{k+2}.\pi'_{\nu_{k+2}\xi_{k+2}}$$

and the construction continues.

(2) Suppose that there is an $i \in [1..k{+}1]$ such that $\xi_{k+2} = \xi_i$. Then we have that $\xi_i \Vdash_{k+1} \forall \overset{\text{int}}{y}(f(n_{k+2},y) = 0)$, from which it follows that $(\xi_i)p_i \Vdash_{k+1} f(n_{k+2},p_i) = 0$. Defining $\mathbb{W} := \Delta_1^{\pi_i 0}$, we have that $(\xi_i)p_i \Vdash_{k+1} \mathbb{W}(f(n_{k+2},p_i)) \to \mathbb{W}(0)$ from which we can deduce that $k_i \nVdash_{k+1} \mathbb{W}(0)$ and $\mathfrak{M} \models f(n_{k+2},p_i) = 0$. In consequence $k_i \star \pi_i \notin \perp\!\!\!\perp_{k+1}$ and hence $k_i \star \pi_i \in \mathrm{th}_{k+1}$ –because the other threads finish with $H_{\nu_i\xi_i}$ in head position–. Now, from 6.16.18 $H_{\nu_{k+2}\xi_{k+2}} \star \overline{n}_{k+2}.\pi'_{\nu_{k+2}\xi_{k+2}}$ belongs to a thread th_j, $j \leq k{+}1$. Moreover, the only thread that

may contain $H_{\nu_{k+2}\xi_{k+2}} \star \overline{n}_{k+2}.\pi'_{\nu_{k+2}\xi_{k+2}}$ is the i-th thread and hence $n_{k+1} = n_i$. We conclude that $\mathfrak{M} \models f(n_i, p_i) = 0$ and player ⓐ wins the match.

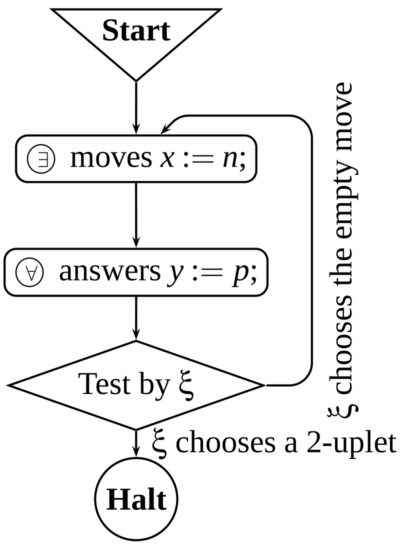
If we make a substitution putting $\mathrm{T}_{\mathrm{int}}\, k_0$ instead of U, we obtain the threads scheme of the match, we have seen in 6.15.16:

$$\begin{aligned} u \star \mathrm{T}_{\mathrm{int}}\, k_0.\pi_0 &\succ k_0 \star \overline{n}_1.\xi_1[{}^{\mathrm{T}_{\mathrm{int}}\, k_0}/_{U}].\pi_0 \\ (\xi_1[{}^{\mathrm{T}_{\mathrm{int}}\, k_0}/_{U}])\overline{p}_1 \star k_1.\pi_1 &\succ k_0 \star \overline{n}_2.\xi_2[{}^{\mathrm{T}_{\mathrm{int}}\, k_0}/_{U}].\pi_0 \\ &\vdots \\ (\xi_k[{}^{\mathrm{T}_{\mathrm{int}}\, k_0}/_{U}])\overline{p}_k \star k_k.\pi_k &\succ k_0 \star \overline{n}_{k+1}.\xi_{k+1}[{}^{\mathrm{T}_{\mathrm{int}}\, k_0}/_{U}].\pi_0 \\ &\vdots \end{aligned} \tag{6.16.19}$$

This implies that the construction given above must stop, because if not, the sequence of $(p_i)_{i\in\mathbb{N}}$ constitutes a winning strategy for ⓥ playing against the strategy implemented by u. This is impossible since $u \Vdash \exists^{\mathrm{int}} x \forall^{\mathrm{int}} y (f(x,y)=0)$.

We can represent this behaviour by a flux diagram:

(6.16.20)



6.5. **Strategies as realizators.**

We can generalize the result 6.10. First, we must give some definitions and after that, we can enunciate the generalisation and prove it.

Definition 6.17. *A $\mathcal{L}_G$-parametrical formula is said* set-relativized *if and only if all their relativized quantifiers (if there exists) are relativized to data sets. We can inductively define these formulæ:*

A first kind $\mathcal{L}_G$ formula

$$\varphi = \forall\vec{x}[\mathcal{E}_1(x_1);\ldots;\mathcal{E}_r(x_r);\varphi_1,\ldots,\varphi_s \to A]$$

is said set-relativized *if and only if $\mathcal{E}_1,\ldots,\mathcal{E}_r$ are data sets and the hypothesis $\varphi_1,\ldots,\varphi_s$ are set-relativized $\mathcal{L}_G$-formulæ.*

A second kind $\mathcal{L}_G$-formula

$$\varphi = \forall W\forall\vec{x}[\mathcal{E}_1(x_1);\ldots;\mathcal{E}_r(x_r);\varphi_1,\ldots,\varphi_s \to W\vec{\tau}]$$

is said set -relativized *if and only if $\mathcal{E}_1,\ldots,\mathcal{E}_r$ are data sets and the hypothesis $\varphi_1,\ldots,\varphi_s$ are set-relativized $\mathcal{L}_G$-formulæ.*

Remark 6.18. *For each parametrical set-relativized $\mathcal{L}_G$-formula Φ, if $[\Phi] = \emptyset$, then for each realizability model $\perp\!\!\!\perp$ we have that $||\Phi||_{\perp\!\!\!\perp} = \emptyset$.*

Definition 6.19. *Let us consider a static substitution $\mathscr{S}$. Given an a-ary second order parameter $\mathbb{A}$, we denote by $\mathscr{S}(\mathbb{A})$ the a-ary second order parameter defined as $(\mathscr{S}(\mathbb{A}))(\vec{n}) := \{\mathscr{S}(\pi) \mid \pi \in \mathbb{A}(\vec{n})\}$. Given a parametrical formula $\Phi = \Phi(\vec{R},\vec{m})$, we denote as $\mathscr{S}(\Phi)$ the parametrical formula $\Phi(\overrightarrow{\mathscr{S}(R)},\vec{m})$. In other words, we apply $\mathscr{S}$ to all the Φ parameters.*

In particular, if $\mathbb{A}$ is a Δ parameter, $\mathbb{A} = \Delta_a^{\pi,\vec{\tau}}$, then $\mathscr{S}(\mathbb{A}) = \Delta_a^{\pi,\vec{\tau}}$ and hence $\mathscr{S}(\mathbb{A})$ is also a Δ parameter.

We prove now a technical lemma:

Lemma 6.20. *Let us consider a first (resp. second) kind Δ-parametrical set-relativized $\mathcal{L}_G$-formula Φ, more precisely:*

$$\Phi := \forall\vec{x}[\mathcal{E}_1(x_1);\ldots;\mathcal{E}_r(x_r);\Phi_1(\vec{x}),\ldots,\Phi_s(\vec{x}) \to A(\vec{x})]$$

(resp.: $\Phi := \forall W\forall\vec{x}[\mathcal{E}_1(x_1);\ldots;\mathcal{E}_r(x_r);\Phi_1(W,\vec{x}),\ldots,\Phi_s(W,\vec{x}) \to W\vec{\tau}]$) and $(\Upsilon_i)_{i\in I}$ a set of $\mathcal{L}_G$ Δ-parametrical formulæ. Consider $\vec{x} := \vec{m}$ (resp. $\vec{x} := \vec{m}$, $W := \mathbb{W} \in \Delta$) an instantiation of Φ. Suppose there is no parameter of Φ nor $(\Upsilon_i)_{i\in I}$ containning $\varkappa_{\Phi_1(\vec{m})},\ldots,\varkappa_{\Phi_s(\vec{m})}$. Suppose that $\tau = \tau[(\varkappa_{\Upsilon_i})_{i\in I}]$ is a Λ_c-term such that τ has a winning strategy for Φ (c.f.: 5.22).

Let be $(\xi_i)_{i=1}^{i=s}$ and $(\iota_i)_{i\in I}$ Λ_c-terms. Consider the (static) substitution $\mathscr{S}_0 := [(\iota_i/\varkappa_{\Upsilon_i})_{i\in I}]$ and a realizability model $\perp\!\!\!\perp$ such that $\forall i\in[1..s]$ $\xi_i \Vdash \mathscr{S}_0(\Phi_i(\vec{m}))$ (resp.: $\xi_i \Vdash \mathscr{S}_0(\Phi_i(\mathbb{W},\vec{m}))$) and $\forall i\in I$ $\iota_i \Vdash \mathscr{S}_0(\Upsilon_i)$. Consider canonical representations $e_i\in\mathcal{E}_i(m_i)$ and a stack $\rho_0\in||A(\vec{m})||$ (resp.:$\rho_0\in\mathbb{W}(\vec{\tau})$).

Then: $\mathscr{S}_0(\tau)\star e_1\ldots e_r.\xi_1\ldots\xi_s.\mathscr{S}_0(\rho_0) \in \perp\!\!\!\perp$.

Proof. By hypothesis, we know that $\tau\star e_1\ldots e_r.\varkappa_{\Phi_1(\vec{m})}\ldots\varkappa_{\Phi_s(\vec{m})}.\rho_0 \in \perp\!\!\!\perp_G$. Then this process reduces to $\varkappa_\Sigma\star\rho_1$ for a suitable Δ parametrical $\mathcal{L}_G$-formula Σ. Let us define $\mathscr{S}_1 := \mathscr{S}_0 + [(\xi_i/\varkappa_{\Phi_i(\vec{m})})_{i=1}^{i=s}]$. Applying $\mathscr{S}_1$ we obtain:

$$\mathscr{S}_0(\tau)\star e_1\ldots e_r.\xi_1\ldots\xi_s\mathscr{S}_0(\rho_0) \succ \mathscr{S}_1(\varkappa_\Sigma)\star\mathscr{S}_1(\rho_1) \tag{6.20.21}$$

We remark that, since all Δ-parameters in Φ does not contain the constants $(\varkappa_{\Phi_i(\vec{m})})_{i=1}^{i=s}$, then $\mathscr{S}_1(\rho_0)=\mathscr{S}_0(\rho_0)$.

An interaction constant belonging to $\tau\star e_1\ldots e_r.\varkappa_{\Phi_1(\vec{m})}\ldots\varkappa_{\Phi_s(\vec{m})}.\rho_0$ must be $\varkappa_{\Upsilon_i}$, where $i\in I$ or $\varkappa_{\Phi_i(\vec{m})}$, where $i\in[1..s]$. In consequence, $\Sigma = \Upsilon_i$

or $\Sigma{=}\Phi_i(\vec{m})$ for a suitable i. By hypothesis $\mathscr{S}_1(\varkappa_\Sigma) \Vdash \mathscr{S}_0(\Sigma)$ and hence, it suffices to prove that $\mathscr{S}_1(\rho_1) \in ||\mathscr{S}_0(\Sigma)||$. We proceed by cases:

(1) $\varkappa_\Sigma \star \rho_1 \in G$. In other words, $\textcircled{$\exists$}$ can play but $\textcircled{$\forall$}$ can not answer. Suppose Σ is a first (resp. second) kind formula:

$$\Sigma = \forall\vec{y}[\tilde{\mathcal{E}}_1(y_1);\ldots;\tilde{\mathcal{E}}_{r'}(y_{r'});\Sigma_1(\vec{m},\vec{y}),\ldots,\Sigma_{s'}(\vec{m},\vec{y}) \to B(\vec{m},\vec{y})]$$

(or $\Sigma{=}\forall U\forall\vec{y}[\tilde{\mathcal{E}}_1(y_1);\ldots;\tilde{\mathcal{E}}_{r'}(y_{r'});\Sigma_1(U,\vec{m},\vec{y}),\ldots,\Sigma_{s'}(U,\vec{m},\vec{y}){\to}U\vec{\sigma}]$ if Σ is a second kind formula). Since $\textcircled{$\exists$}$ can play, there is an instantiation $\vec{y} := \vec{p}$ (resp.: $\vec{y} := \vec{p}$ and $U := \mathbb{U} \in \Delta$) such that $\rho_1 = e_1\ldots e_{r'}.\chi_1\ldots\chi_{s'}.\pi_1$, $\forall i{\in}[1..r']$ $d_i \in \tilde{\mathcal{E}}_i(p_i)$ and $\pi_1 \in ||B(\vec{m},\vec{p})||$ (resp.: $\pi_1 \in \mathbb{U}(\vec{\sigma})$). There are two possibilities:

(a) Σ has no hypothesis and hence $s' = 0$.
If Σ is of first kind, then $B(\vec{m},\vec{p}) = \mathbb{V}(\vec{t})$ for a Δ-parameter $\mathbb{V}$ that –by hypothesis– contain no constants $(\varkappa_{\Phi_i(\vec{m})})_{i=1}^{i=s}$. Then ρ_1 also contain no these constants and hence $\mathscr{S}_1(\rho_1) = \mathscr{S}_0(\rho_1) \in ||\mathscr{S}_0(\Sigma)||$.
Suppose Σ is of second kind. Since $\mathbb{U}$ is a Δ-parameter and hence $\mathscr{S}_0(\mathbb{U})$ is a Δ-parameter and $\mathscr{S}_0(\pi_1) \in (\mathscr{S}_0(\mathbb{U}))(\vec{\sigma})$, thus concluding that $\mathscr{S}_1(\rho_1) = \mathscr{S}_0(\rho_1) \in ||\mathscr{S}_0(\Sigma)||$.

(b) $s' \neq 0$ but $\forall j{\in}[1..s']$ $\llbracket\Sigma_j(\vec{m},\vec{p})\rrbracket = \emptyset$ (resp.: $\llbracket\Sigma_j(\mathbb{U},\vec{m},\vec{p})\rrbracket$).
We can apply 6.18, thus obtaining that $\forall j{\in}[1..s']$ $||\Sigma_j(\vec{m},\vec{p})||{=}\emptyset$, then $||\mathscr{S}_0(\Sigma_j(\vec{m},\vec{p}))||{=}\emptyset$ and hence $\chi_j \Vdash \Sigma_j(\vec{m},\vec{p})$
(resp.: $||\Sigma_j(\mathbb{U},\vec{m},\vec{p})|| = \emptyset$, then $||\mathscr{S}_0(\Sigma_j(\vec{m},\vec{p}))|| = \emptyset$ and hence $\chi_j \Vdash \Sigma_j(\mathbb{U},\vec{m},\vec{p})$).
If Σ is of first kind formula, –as have done in (a)– we can prove that $\mathscr{S}_1(\pi_1) = \mathscr{S}_0(\pi_1) \in ||\mathscr{S}_0(B(\vec{m},\vec{p}))||$, thus concluding that $\mathscr{S}_1(\rho_1) \in ||\mathscr{S}_0(\Sigma)||$.
If Σ is a second kind formula, we reason in an entirely similar way than we have done in (a).

(2) $\varkappa_\Sigma \star \rho_1 \in \perp\!\!\!\perp_G \backslash G$. In other words, $\textcircled{$\exists$}$ can play but whatever $\textcircled{$\exists$}$ moves,(resp. second) kind formula. $\textcircled{$\forall$}$ has always an answer. Suppose that Σ is a first (resp. second) kind formula:

$$\Sigma = \forall\vec{y}[\mathcal{E}_1(y_1);\ldots;\mathcal{E}_{r'}(y_{r'});\Sigma_1(\vec{m},\vec{y}),\ldots,\Sigma_{s'}(\vec{m},\vec{y}) \to B(\vec{m},\vec{y})]$$

(resp. $\Sigma = \forall U\forall\vec{y}[\mathcal{E}_1(y_1);\ldots;\mathcal{E}_{r'}(y_{r'});\Sigma_1(\vec{m},\vec{y}),\ldots,\Sigma_{s'}(\vec{m},\vec{y}){\to}U\vec{\sigma}]$)
Since $\textcircled{$\exists$}$ can move, there is an instantiation $\vec{y} := \vec{p}$ (resp. $\vec{y} := \vec{p}$ and $U := \mathbb{U}{\in}\Delta$) such that $\rho_1 = d_1\ldots d_{r'}.\chi_1\ldots\chi_{s'}.\pi_1$, $e_i \in \mathcal{E}_i(p_i)$, $\pi_1 \in ||B(\vec{m},\vec{p})||$ (resp. $\pi_1 \in \mathbb{U}(\vec{\sigma})$).

If Σ is a first kind formula, as we have seen in (a), we have that $\mathscr{S}_1(\pi_1) \in ||\mathscr{S}_0(B(\vec{m},\vec{p}))||$

If Σ is a second kind formula, then $\mathscr{S}_1(\pi_1) \in (\mathscr{S}_1(\mathbb{U}))(\vec{\sigma})$.

In this case, 6.20.21 becomes:

(6.20.22)
$$\mathscr{S}_0(\tau){\star}e_1\ldots e_r.\xi_1\ldots\xi_s\mathscr{S}_0(\rho_0) \succ \mathscr{S}_0(\varkappa_\Sigma){\star}d_1\ldots d_{r'}.\mathscr{S}_1(\chi_1)\ldots\mathscr{S}_1(\chi_{s'}).\mathscr{S}_1(\pi_1)$$

We must prove that $d_1 \dots d_{r'}.\mathscr{S}_1(\chi_1)\dots\mathscr{S}_1(\chi_{s'}).\mathscr{S}_1(\pi_1) \in ||\mathscr{S}_0(\Sigma)||$, it suffices to prove that $\forall j \in [1..s']\ \mathscr{S}_1(\chi_j) \Vdash \mathscr{S}_0(\Sigma_j(\vec{m},\vec{p}))$ (resp.: $\mathscr{S}_1(\chi_j) \Vdash \mathscr{S}_0(\Sigma_j(\mathbb{U},\vec{m},\vec{p}))$).

Each exception handler χ_j has a winning strategy for the hypothesis Σ_j because $\textcircled{\exists}$ wins against all $\textcircled{\forall}$ answer. Suppose

$$\Sigma_j = \forall\vec{z}[\mathcal{E}_1(z_1);\dots;\mathcal{E}_{r''}(z_{r''});\Sigma_{j1},\dots,\Sigma_{js''} \to C(\vec{m},\vec{p},\vec{z})]$$

(resp.:$\Sigma_j = \forall V \forall\vec{z}[\overline{\mathcal{E}}_1(z_1);\dots;\overline{\mathcal{E}}_{r''}(z_{r''});\Sigma_{j1},\dots,\Sigma_{js''} \to V\vec{\upsilon}]$). We must take a stack ρ_2 belonging to $||\Sigma_j||$ and prove that $\chi_j \star \rho_2 \in \perp\!\!\!\perp$. Consider $\vec{z} := \vec{q}$ (resp.: $\vec{z} := \vec{q}$ and $V := \mathbb{V}$) an instantiation of Σ_j. Consider canonical representations $f_1,\dots,f_{r''}$ such that $\forall i \in [1..r'']\ f_i \in \overline{\mathcal{E}}_i(q_i)$; $\gamma_1,\dots,\gamma_{s''}$ Λ_c-terms such that $\forall i \in [1..s'']\ \gamma_i \Vdash \Sigma_{ji}$ and π_2 a stack belonging to $||C(\vec{m},\vec{p},\vec{q})||$ (resp.: $\pi_2 \in \mathbb{V}(\upsilon)$). We must prove that

$$\mathscr{S}_1(\chi_j) \star f_1 \dots f_{r''}.\gamma_1 \dots \gamma_{s''}.\mathscr{S}_1(\pi_2) \in \perp\!\!\!\perp \tag{6.20.23}$$

Consider the hypothesis of this lemma. Adding the parametrical formulæ $\Phi_1,\dots,\Phi_r$ to the initial set of $(\Upsilon_i)_{i\in I}$, taking new "fresh" constants $(\varkappa_{\Sigma_{ji}})_{i=1}^{i=s''}$ instead of $(\varkappa_{\Phi_i})_{i=1}^{i=s}$, using the terms $(\gamma_i)_{i=1}^{i=s''}$ instead of the terms $(\xi_i)_{i=1}^{i=s}$ and $\mathscr{S}_1$ instead of $\mathscr{S}_0$; the proof of 6.20.23 is reduced to the proof of the lemma these particular objects.

Since this process builts a play for the game associated with Φ and by hypothesis τ has a winning strategy for Φ; the proof must stop in a "base case" once the play is ended. In this case the lemma is immediately proven.

The reader should be noticed that this proof is no other than a generalisation of the proof seen in 6.10. □

Corollary 6.21. *Let us consider a closed (without parameters) set-relativized $\mathcal{L}_G$-formula and a Λ_c-term τ. Then τ has a winning strategy for Φ if and only if $\tau \Vvdash \Phi$.*

6.6. **A more complicated back-tracking.**

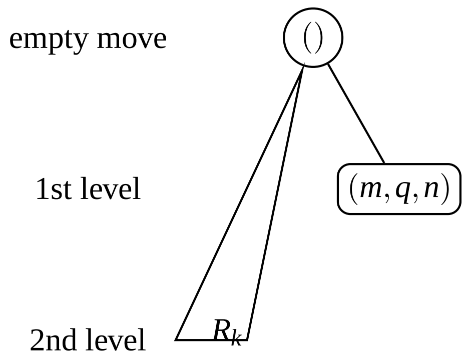
At the end of this work, we will analyze an example of strategies composition. More precisely, we will obtain a winning strategy from a proof of an implication. One of these formulæ, which will be used in the last example, is the following:

Example 6.22. Let us consider the following formula:

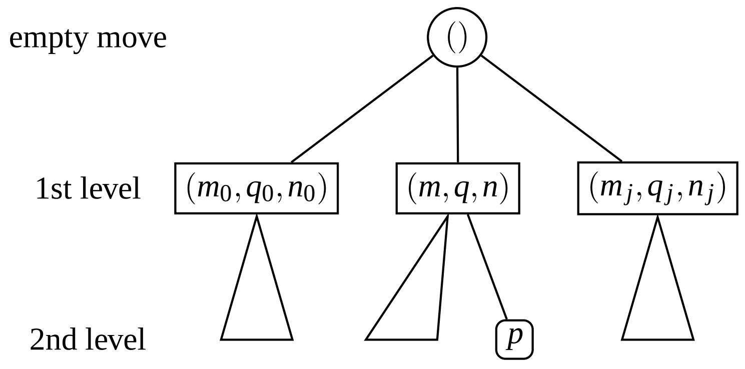
$$\Phi = \overset{\text{int}}{\exists} x' \overset{\text{int}}{\forall} y' x \overset{\text{int}}{\exists} y (h(x,y,x',y') = 0)$$

where h is a function symbol. There is a fundamental difference between this case and the one of 6.15: The *back-tracking* is more complicated. We can describe recursively the rules of the game associated with Φ (which we defined on 5.9), defining at each step k the tree of *reached positions* R_k. This one contains as root the 0-uplet $()$ –called the *empty position*– and it may contain 3-uplets in the first level and 1-uplets in the second level. In the beginning R_0 is defined as the tree containning only the node $()$. In the k-th step, first $\textcircled{\exists}$ chooses a reached position (a node) $P \in R_k$. This implies that, in the beginning, only the empty position $()$ is available. After that, $\textcircled{\exists}$ and $\textcircled{\forall}$ proceeds according to the following discussion:

- Suppose that $P = ()$. In this case $\textcircled{\exists}$ plays an integer m and player $\textcircled{\forall}$ answers two integers q and n. The tree R_{k+1} is defined by adding the node (m,q,n) to the first level of R_k:



- Suppose that $P = (m,q,n)$. In this case player $\textcircled{\exists}$ plays an integer p and we define R_{k+1} by adding the node p to the subtree having root (m,q,n):



- Suppose that $P = p$ –i.e. a node in the 2nd level– which belongs to a subtree with root (m,q,n). In this case the execution stops and $\textcircled{\exists}$ wins if and only if $\mathfrak{M} \models h(m,q,n,p) = 0$.

In other words, $\textcircled{\exists}$ chooses a reached position P and interacts with $\textcircled{\forall}$ in order to propose a new completion for P.

We describe now the thread scheme associated with a specific play between $\textcircled{\exists}$ and $\textcircled{\forall}$, where $\textcircled{\exists}$ plays according to the strategy implemented by a term t realizing the formula Φ. We will also find a semantic condition determining what is the reached position chosen by $\textcircled{\exists}$ at each step. Before that, we remember some notations about Φ subformulæ which we will use later:

- $\Phi_1(X') := \overset{\text{int}}{\forall} x' [\overset{\text{int}}{\forall} y' x \overset{\text{int}}{\exists} y (h(x',y',x,y)=0) \to X']$
- $\Phi_{11}(x') := \overset{\text{int}}{\forall} y' x \overset{\text{int}}{\exists} y (h(x',y',x,y)=0)$
- $\Phi_{111}(x',Y) := \overset{\text{int}}{\forall} y' x [\overset{\text{int}}{\forall} y (h(x',y',x,y)=0) \to Y]$
- $\Phi_{1111}(x',y',x) := \overset{\text{int}}{\forall} y (h(x',y',x,y)=0)$

At each step k, we will define a set of indexes I_k and for each index $i \in I_k$, another set of indexes J_{ik}. The set I_k indexes the reached positions of R_k that are 3-uplets (m_i,q_i,n_i) and the sets J_{ik} indexes the reached positions of R_k that are 4-uplets (m_i,q_i,n_i,p_{ij}) for a suitable integer p_{ij}.

Let us consider a sequence of Π-constants $(\pi_i)_{i\in\mathbb{N}}$ pairwise different and a sequence of instructions $(U^i)_{i\in\mathbb{N}}$ pairwise different such that:

(1) τ does not contain U^i for all i
(2) For all i, $U^i \in \overline{\mathcal{F}}_{\mathrm{T_{int}}K}(\pi_i)$ (c.f. 4.11).
(3) Each function $(\rho,\iota) \mapsto \pi^i_{\rho\iota}$ is injective on ι.

We proceed by induction on steps:

Step 0: Define $\mathrm{th}_0 := \mathrm{th}^*_0 := \mathrm{th}_{\tau\star U^0.\pi_0}$, the set of reached positions $R_0 := \{()\}$, the realizability model $\perp\!\!\!\perp_0 := \mathrm{th}^c_0$ and I_0 as the empty set. We have no J_{i0} sets. By 4.4 there is an integer m_1 and two Λ_c-terms μ_1,χ_1 such that $H^0_{\mu_1\chi_1} \star \overline{m}_1.\pi^0_{\mu_1\chi_1} \notin \perp\!\!\!\perp_0$ and $\chi_1 \Vdash_0 \Phi_{11}(m_1)$. This implies that the 0-th thread is of the form:

$$\tau \star U^0.\pi_0 \succ H^0_{\mu_1\chi_1} \star \overline{m}_1.\pi^0_{\mu_1\chi_1}$$

Let us start a play between ⓔ and ⓐ where ⓔ proposes m_1 as its first move. Player ⓐ must answer two integers q_1,n_1[6]. Define $I_1 := \{1\}$, $J_{11} := \emptyset$, the current thread $\mathrm{th}^*_1 := \mathrm{th}_{(\chi_1)\overline{q}_1\overline{n}_1\star U_1.\pi_1}$ and built according to 6.22 the reached position tree R_1 by addition of the node (m_1,q_1,n_1) to the R_0 first level.

Notice that, if $H^0_{\mu\chi}\star\overline{m}.\pi^0_{\mu\chi} \notin \perp\!\!\!\perp_1 := (\mathrm{th}_{\tau\star U_0.\pi_0})^c \cap (\mathrm{th}^*_1)^c$ and $\chi\neq\chi_1$, then $H^0_{\mu\chi}\star\overline{m}.\pi^0_{\mu\chi} \in \mathrm{th}^*_1$ because from injective condition (3), we know $\pi^0_{\mu\chi}\neq\pi^0_{\mu_1\chi_1}$.

Step $k+1$: Take $\perp\!\!\!\perp_{k+1} = (\mathrm{th}_{\tau\star U_{k+1}.\pi_0})^c \cap \bigcap_{i\in I_{k+1}} [(\mathrm{th}_{(\chi_i)\overline{q}_i.\overline{n}_i\star U_i.\pi_i})^c \cap \bigcap_{j\in J_{i(k+1)}} (\mathrm{th}_{\iota_{ij}\star k_{ij}.\pi_{ij}})^c]$.

Consider we have defined a current thread th^*_{k+1}, which is the thread of some process P_{k+1}. We define some clauses:

$(\mathcal{S}_{k+1})$: If $H^0_{\mu\chi}\star\overline{m}.\pi^0_{\mu\chi} \notin \perp\!\!\!\perp_{k+1}$ and $\forall i\in I_{k+1}\ \ \chi\neq\chi_i$, then $H^0_{\mu\chi}\star\overline{m}.\pi^0_{\mu\chi}\in\mathrm{th}^*_{k+1}$.

$(\mathcal{S}'_{i(k+1)})$: If $H^i_{\rho\iota}\star\overline{p}.\pi^i_{\rho\iota} \notin \perp\!\!\!\perp_{k+1}$ and $\forall j\in J_{i(k+1)}\ \ \iota\neq\iota_{ij}$, then $H^i_{\rho\iota}\star\overline{p}.\pi^i_{\rho\iota}\in\mathrm{th}^*_{k+1}$.

Suppose that $(\mathcal{S}_{k+1})$ is true and that for all $i\in I_{k+1}$ $(\mathcal{S}'_{i(k+1)})$ also is true. By 4.4, there is an integer m and two Λ_c-terms μ,χ such

[6]In the implemented game, ⓐ must choose a stack belonging to $[\![\Phi_{11}(m_1)]\!]$, i.e. a stack of the form $\overline{q}_1.\overline{n}_1.\mathrm{T_{int}}\,k_1.\pi_1$ where q_1,n_1 are integers, $\pi_1 \in \mathbb{Y}_1 \in \Delta_1$ and k_1 abbreviates the instruction $\varkappa_{\Phi_{111}(m_1,\mathbb{Y})}$. We define here, as in 6.16, a thread scheme using substitutable instructions in order to use dynamic substitution.

that $H^0_{\mu\chi}\star\overline{m}.\pi^0_{\mu\chi} \notin \perp\!\!\!\perp_{k+1}$ and $\chi \Vdash_{k+1} \Phi_{11}(m)$. Here there are two possibilities:

First case: $\forall i{\in}I_{k+1}\ \ \chi \neq \chi_i$. By $(\mathcal{S}_{k+1})$ $H^0_{\mu\chi}\star\overline{m}.\pi^0_{\mu\chi}$ belongs to the current thread th^*_{k+1}. Define $i'{:=}\max I_{k+1}+1$ and $(\mu_{i'},m_{i'},\chi_{i'}){:=}(\mu,m,\chi)$. Since $H^0_{\mu_{i'}\chi_{i'}}$ arrives in head position in the current thread th^*_{k+1}, it corresponds to the choice by $\textcircled{\exists}$ of the empty position $()\in R_{k+1}$. We add the line $P_{k+1} \succ H^0_{\mu_{i'}\chi_{i'}}\star\overline{m}_{i'}.\pi^0_{\mu_{i'}\chi_{i'}}$ as the $k{+}1$-th thread scheme line.
After $\textcircled{\exists}$ plays $m_{i'}$ as its $k{+}1$-th move, player $\textcircled{\forall}$ must answer two integers $q_{i'},n_{i'}$[7]. Define $I_{k+2} := I_{k+1}\cup\{i'\}$, $J_{i'(k+1)} := \emptyset$ and $\forall i{\in}I_{k+1}\ J_{i(k+2)} := J_{i(k+1)}$. Define the current thread th^*_{k+2} as $\mathrm{th}_{(\chi_{i'})\overline{q}_{i'}.\overline{n}_{i'}\star U^{i'}.\pi_{i'}}$ and the tree R_{k+2} of reached positions by addition of the node $(m_{i'},q_{i'},n_{i'})$ to the R_{k+1} first level . Notice that $(\mathcal{S}_{k+2})$ is true and, for each integer $i{\in}I_{k+2}$, $(\mathcal{S}'_{i(k+2)})$ is true.

Second case: $\exists i{\in}I_{k+1}\ \ \chi=\chi_i$. Since $\chi\Vdash_{k+1}\Phi_{11}(m)$, the term $(\chi_i)\overline{q}_i.\overline{n}_i$ realizes $\overset{\mathrm{int}}{\exists} y(h(m_i,q_i,n_i,y)=0)$ in $\perp\!\!\!\perp_k$. By definition, $(\chi_i)\overline{q}_i.\overline{n}_i\star U^i.\pi_i$ does not belong to $\perp\!\!\!\perp_{k+1}$. By 4.4, there is an integer p and two Λ_c-terms ρ,ι such that $H^i_{\rho\iota}\star\overline{p}.\pi^i_{\rho\iota}$ does not belong to $\perp\!\!\!\perp_{k+1}$ and $\iota\Vdash_{k+1} h(m_i,q_i,n_i,p)=0$. Define $I_{k+2}:=I_{k+1}$. Still, it remains two possibilities:

(a) $\forall j{\in}J_{i(k+1)}\ \ \iota\neq\iota_{ij}$. By $(\mathcal{S}'_{i(k+1)})$, $H^i_{\rho\iota}\star\overline{p}_{ij}.\pi^i_{\rho\iota}$ belongs to th^*_{k+1}. Define $j'{:=}\max J_{ik}+1$ and $(\rho_{ij'},p_{ij'}\iota_{ij'}){:=}(\rho,p,\iota)$. We add the line $P_{k+1}\succ H^i_{\rho_{ij'}\iota_{ij'}}\star\overline{p}_{ij'}.\pi^i_{ij'}$ as the $k{+}1$-th thread scheme line. Since $H^i_{\rho_{ij'}\iota_{ij'}}$ arrives in head position in the current thread, it corresponds to the choice by $\textcircled{\exists}$ of the reached position (m_i,q_i,n_i). After that $\textcircled{\exists}$ plays $p_{ij'}$ as its $k{+}1$-th move[8]. Define $J_{i(k+2)}:=J_{i(k+1)}\cup\{j'\}$, the current thread th^*_{k+2} as $\mathrm{th}_{\iota_{ij}\star k_{ij}.\pi_{ij}}$ and the tree R_{k+2} from reached positions by addition of the node p_{ij} to the R_{k+1} second level. The reader must check that the statement $(\mathcal{S}_{k+2})$ is true and that for all $i{\in}I_{k+2}$, $(\mathcal{S}'_{ik+2})$ is also true.

(b) $\exists j{\in}J_{i(k+1)}\ \ \iota=\iota_{ij}$. Since $\iota_{ij}\star k_{ij}.\pi_{ij}$ does not belong to $\perp\!\!\!\perp_{k+1}$ and $\iota_{ij}\Vdash_{k+1} h(m_i,q_i,n_i,p_{ij})=0$, we can conclude that $k_{ij}\nVdash_{k+1}\Delta^{\pi_{ij},0}(h(m_i,q_i,n_i,p_{ij}))$. In consequence $\mathfrak{M}\models h(m_i,q_i,n_i,p_{ij})=0$ and $k_{ij}\star\pi_{ij}\notin\perp\!\!\!\perp_k$. Since k_{ij} stops the execution, $k_{ij}\star\pi_{ij}$ must belong to the current thread th^*_{k+1}. The execution stops and $\textcircled{\exists}$ wins. We add

[7]In the implemented game, $\textcircled{\forall}$ plays a stack $\overline{q}_{i'}.\overline{n}_{i'}.\mathrm{T}_{\mathrm{int}}k_{i'}.\pi_{i'}$ belonging to $[\Phi_{11}(m_{i'})]$. We remember that we built thread schemes allowing to use dynamic substitutions.

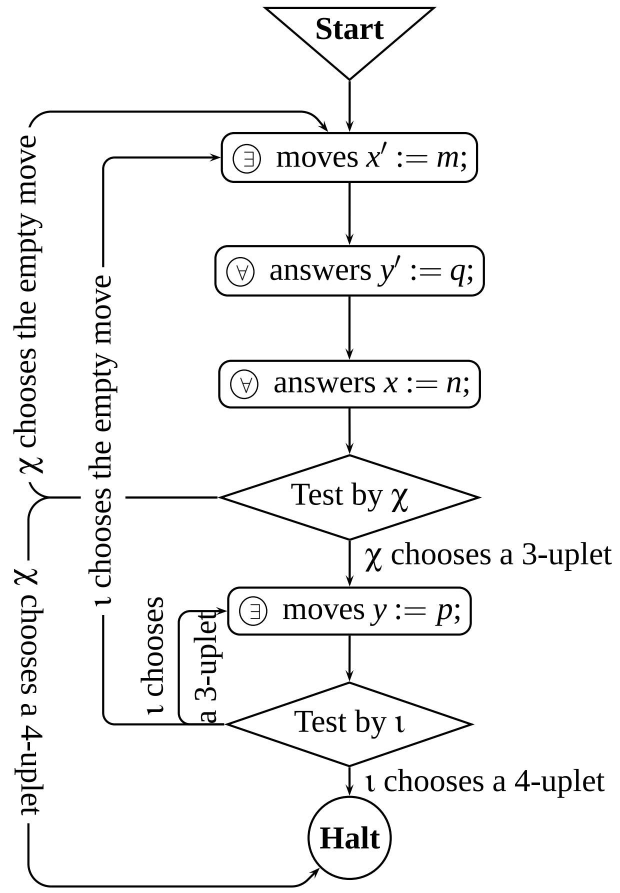
[8]In the implemented game, $\textcircled{\forall}$ must choose a stack $k_{ij}.\pi_{ij}\in[h(m_i,q_i,n_i,p_{ij}){=}0]$.

the line $P_{k+1} \succ k_{ij} \star \pi_{ij}$ as the k+1-th (and last) thread scheme line.

For each $i \in I_k$, we say that $\Phi_{11}(m_i)$ is *the associated formula* to the exception handler χ_i. For each $j \in J_{ik}$, we say that $h(m_i, q_i, n_i, p_{ij}) = 0$ is *the associated formula* of the exception handler ι_{ij}.

A flow diagram describes the behaviour of a winning strategy:

(6.22.24)



Remark 6.23. At each step k, if there is an $i \in I_k$ such that $\chi_i \Vdash_k \Phi_{11}(m_i)$, then i is unique. Furthermore, if there is also an index $j \in J_{ik}$ such that $\iota_{ij} \Vdash_k h(m_i, q_i, n_i, p_{ij}) = 0$, then j is unique.

Proof. Suppose that there are two integers $i, i' \in I_k$ an integer $j \in J_{ik}$ and another $j' \in J_{i'k}$ such that $\iota_{ij} \Vdash_k h(m_i, q_i, n_i, p_{ij}) = 0$ and $\iota_{i'j'} \Vdash_k h(m_{i'}, q_{i'}, n_{i'}, p_{i'j'}) = 0$. Since $\iota_{ij} \star k_{ij}.\pi_{ij}$ and $\iota_{i'j'} \star k_{i'j'}.\pi_{i'j'}$ does not belong to $\perp\!\!\!\perp_k$, we have that $k_{ij} \star \pi_{ij}$ and $k_{i'j'} \star \pi_{i'j'}$ neither belongs to $\perp\!\!\!\perp_k$ (c.f. the proof of 6.22). Hence $k_{ij} \star \pi_{ij}$ and $k_{i'j'} \star \pi_{i'j'}$ belongs to th_k^* and then $i = i'$ and $j = j'$.

Let us suppose that $\chi_i \Vdash_k \Phi_{11}(m_i)$ and $\chi_{i'} \Vdash_k \Phi_{11}(m_{i'})$ where $i, i' \in I_k$. Since $(\chi_i)\overline{q}_i.\overline{n}_i \star \mathrm{T}_{\mathrm{int}}\, k_i.\pi_i$ and $(\chi_{i'})\overline{q}_{i'}.\overline{n}_{i'} \star \mathrm{T}_{\mathrm{int}}\, k_{i'}.\pi_{i'}$ neither belongs to $\perp\!\!\!\perp_k$, applying 4.4, we obtain that there are two integers p, p' and two Λ_c-terms ι, ι' such that $k_i \star \overline{p}.\iota.\pi_i$ and $k_{i'} \star \overline{p}'.\iota'.\pi_{i'}$ does not belong to $\perp\!\!\!\perp_k$. Since k_i and $k_{i'}$ are differents, at least one of them does not belong to the current thread th^*_k. Suppose that $k_i \star \overline{p}.\iota.\pi \notin \mathrm{th}^*_k$. Then, there is a $j \in J_{ik}$ such that $p = p_{ij}$, $\iota = \iota_{ij}$ and $\iota_{ij} \star k_{ij}.\pi_{ij} \notin \perp\!\!\!\perp_k$. By 4.4 $\iota_{ij} \Vdash_k h(m_i, n_i, q_i, p_{ij}) = 0$ and by the proof of 6.22, $k_{ij} \star \pi_{ij} \in \mathrm{th}^*_k$. In consequence $k_{i'} \star \overline{p}'.\iota'.\pi_{i'} \notin \mathrm{th}^*_k$, from which we can conclude that $(\iota, p) = (\iota_{i'j'}, p_{i'j'})$ for a suitable $j' \in J_{i'k}$. By 4.4, $\iota_{i'j'} \Vdash h(m_{i'}, q_{i'}, n_{i'}, p_{i'j'}) = 0$ and, still by the proof of 6.22, we can conclude that $k_{i'j'} \star \pi_{i'j'} \in \mathrm{th}^*_k$. Hence $i = i'$ and $j = j'$. □

Corollary 6.24. *At each step k we have the following equivalences:*

(1) $\textcircled{\exists}$ *chooses the empty position if and only if there is no exception handler that realizes its associated formula in $\perp\!\!\!\perp_k$.*
(2) $\textcircled{\exists}$ *chooses the position (m_i, q_i, n_i) if and only if in $\perp\!\!\!\perp_k$ the only one exception handler realizing its associated formula is χ_i.*
(3) $\textcircled{\exists}$ *chooses the position (m_i, q_i, n_i, p_{ij}) if and only if χ_i and ι_{ij} both are the sole exception handlers realizing their respective associated formulæ in $\perp\!\!\!\perp_k$.*

6.7. **A proof as a strategy combinator.**

We will analyse a proof as a strategy combinator. Lets consider two functions $f, g : \mathbb{N}^2 \to \mathbb{N}$ and two formulæ:

$$\Psi := \overset{\mathrm{int}}{\exists} x \overset{\mathrm{int}}{\forall} y (f(x,y)=0) \text{ and } \Upsilon := \overset{\mathrm{int}}{\exists} x' \overset{\mathrm{int}}{\forall} y' (g(x',y')=0)$$

The implication $\Psi \to \Upsilon$ is logically equivalent to its prenex normal form:

$$\Phi' := \overset{\mathrm{int}}{\exists} x' \overset{\mathrm{int}}{\forall} y' x \overset{\mathrm{int}}{\exists} y [(f(x,y) = 0) \to (g(x',y') = 0)]$$

In order to use our precedent results to describe strategies, we prefer to write this formula in the following form:

$$\Phi := \overset{\mathrm{int}}{\exists} x' \overset{\mathrm{int}}{\forall} y' x \overset{\mathrm{int}}{\exists} y (h(x,y,x',y') = 0)$$

where h is a 4-ary function satisfying the property:

$$\forall xyx'y' [h(x,y,x',y') = 0 \leftrightarrow (f(x,y) = 0 \to g(x',y') = 0)]$$

A term σ realizing the implication $\Phi, \Psi \to \Upsilon$, is a combinator for winning strategies: Indeed, if we consider two terms $t \Vvdash \Phi$ and $u \Vvdash \Psi$. Then $(\sigma)tu$ realizes Υ and hence, it implements a winning strategy for the game associated with Υ. We wont to describe how σ combines these terms t, u so as to obtain a winning strategy for the game associated with Υ.

Lemma 6.25. *We consider the following proof-rule:*

$$(\overset{\mathrm{int}}{\forall} r)\ \frac{\Gamma, \overline{x} : \mathrm{int}(x) \vdash t : \varphi(x)}{\Gamma \vdash \lambda \overline{x} t : \overset{\mathrm{int}}{\forall} x \varphi(x)}\ x \notin \mathrm{FV}(\Gamma)$$

and the following transformation rules:

$$(\overset{\text{int}}{\forall} l)\quad \frac{\Pi}{\Gamma, a : \varphi(x) \vdash t : \psi} \Rightarrow \frac{\Pi'}{\Gamma, a : \overset{\text{int}}{\forall} x\varphi(x), \bar{x} : \text{int}(x) \vdash t[{}^{(a)\bar{x}}/_{a}] : \psi}$$

$$(\overset{\text{int}}{\exists} r)\quad \frac{\Pi}{\Gamma \vdash t : \varphi(x)} \Rightarrow \frac{\Pi'}{\Gamma, \bar{x} : \text{int}(x) \vdash \lambda y(y)\bar{x}t : \overset{\text{int}}{\exists} x\varphi(x)}$$

$$(\overset{\text{int}}{\exists} l)\quad \frac{\Pi}{\Gamma, a : \varphi(x), \bar{x} : \text{int}(x) \vdash t : \psi}\ x \notin \text{FV}(\Gamma, \psi) \Rightarrow \frac{\Pi'}{\Gamma, c : \overset{\text{int}}{\exists} x\varphi \vdash (c)\lambda\bar{x}\lambda at : \psi}$$

Where the implication symbol $\Rightarrow$ *means that each proof* Π *of the left sequent can be transformed into a proof* Π' *of the right sequent.*

Proof.

- $(\overset{\text{int}}{\forall} r)$ By application of $(\to\text{i})$ and $(\forall)$
- $(\overset{\text{int}}{\forall} l)$ By substitution in Π of $a : \varphi(x)$ for $a : \overset{\text{int}}{\forall} x\varphi(x), \bar{x} : \text{int}(x)$ in the left side and the substitution of a by $(a)x$ in the right side.
- $(\overset{\text{int}}{\exists} r)$ By addition of hypothesis to Π, it is possible to construct a proof Π'' such that:

$$\begin{array}{c}\Pi'' \\ \hline \Gamma, y : \forall x(\text{int}(x), \varphi(x) \to X), \bar{x} : \text{int}(x) \vdash t : \varphi(x) \\ \hline \Gamma, \bar{x} : \text{int}(x) \vdash \lambda y(y)\bar{x}t : \forall x(\text{int}(x), \varphi(x) \to X) \to X \\ \hline \Gamma, \bar{x} : \text{int}(x) \vdash \lambda y(y)\bar{x}t : \overset{\text{int}}{\exists} x\varphi(x)\end{array} \begin{array}{l} \\ (\to\text{e})\text{and}(\to\text{i}) \\ (\forall \overset{2}{i})\end{array}$$

- $(\overset{\text{int}}{\exists} l)$ By addition of hypothesis to Π, it is possible to construct a proof Π'' such that:

$$\begin{array}{c}\Pi'' \\ \hline \Gamma, c : \overset{\text{int}}{\exists} x\varphi, a : \varphi(x), \bar{x} : \text{int}(x) \vdash t : \psi \\ \hline \Gamma, c : \overset{\text{int}}{\exists} x\varphi(x), \bar{x} : \text{int} \vdash \lambda at : \varphi(x) \to \psi \\ \hline \Gamma, c : \overset{\text{int}}{\exists} x\varphi(x) \vdash \lambda\bar{x}\lambda a\,\text{int}(x), \varphi(x) \to \psi \\ \hline \Gamma \vdash (c)\lambda\bar{x}\lambda at : \psi\end{array} \begin{array}{l} \\ (\to\text{i}) \\ (\to\text{i}) \\ (\forall \overset{2}{e})\text{and}(\to\text{e})\end{array}$$

□

Lemma 6.26. *Given three functions* f, g, h *satisfying*

$$\mathfrak{N} \models \forall xyx'y'[h(x,y,x',y') = 0 \leftrightarrow (f(x,y) = 0 \to g(x',y') = 0)]$$

then for each realizability model $⊥\!\!⊥$ *and for each* Λ_c*-terms* ν, ϕ *and for all individuals* n, p, m, q*:*

$$\nu \Vdash (h(n,p,m,q) = 0), \phi \Vdash (f(n,p) = 0) \Rightarrow (\nu)\phi \Vdash g(m,q) = 0$$

Proof. Let us consider $n, p, m, q \in \mathfrak{N}$, a term $\nu \Vdash h(x,y,x',y') = 0$ and a term $\phi \Vdash f(n,p) = 0$. We have three cases:

(1) If $\mathfrak{N} \models f(n,p) \neq 0$, then:

$$|h(x,y,x',y') = 0| = |\forall X(X \to X)| \text{ and } |f(n,p) = 0| = |\top \to \bot|$$

Moreover, $||g(m,q) = 0|| \subseteq ||\top \to \bot||$ and hence $(\nu)\phi \Vdash g(m,q) = 0$.

(2) If $\mathfrak{M} \models f(n,p)=0$ and $\mathfrak{M} \models g(m,q)=0$, then:
$$|h(n,p,m,q)=0| = |\forall X(X\to X)|$$
and hence we have $(\nu)\phi \Vdash g(m,q)=0$.

(3) If $\mathfrak{M} \models f(n,p)=0$ and $\mathfrak{M} \models g(m,q)\neq 0$, then:
$$|h(n,p,m,q)=0| = |\top \to \bot|$$
and hence we have $(\nu)\phi \Vdash g(m,q)=0$.

□

Corollary 6.27.
$\lambda x\lambda y(x)y \Vvdash \forall xyx'y'[h(x,y,x',y')=0 \to (f(x,y)=0 \to g(x',y')=0)]$

We explain now the formal proof of the *modus ponens* in prenex normal form.

Lemma 6.28. *Consider the formulæ* $F(x,y) = (f(x,y){=}0)$ *and* $G(x',y') = (g(x',y'){=}0)$, *where* f *and* g *are* 2*-ary symbols of functions.*

Suppose that h *is a* 4*-ary function symbol such that:*
$$\mathfrak{M} \models \forall xyx'y'[h(x,y,x',y')=0 \leftrightarrow (f(x,y)=0 \to g(x',y')=0)]$$
Denote also as $H(x,y,x',y')$ *the formula* $h(x,y,x',y'){=}0$. *Then, we can derive a proof for the sequent:*
$$\exists^{\mathrm{int}} x'\forall^{\mathrm{int}} y'x\exists^{\mathrm{int}} yH(x,y,x',y'), \exists^{\mathrm{int}} x\forall^{\mathrm{int}} yF(x,y) \vdash \exists^{\mathrm{int}} x'\forall^{\mathrm{int}} y'G(x',y')$$

Proof.

$$\frac{}{x:H, y:F \vdash (x)y : G}\ (6.26)$$

$$\frac{x:H, y:F \vdash (x)y : G}{\bar{y}:\mathrm{int}(y), x:H, y:\forall^{\mathrm{int}} yF \vdash (x)(y)\bar{y} : G}\ (\forall^{\mathrm{int}}_{l})$$

$$\frac{\bar{y}:\mathrm{int}(y), x:H, y:\forall^{\mathrm{int}} yF \vdash (x)(y)\bar{y} : G}{z:\exists^{\mathrm{int}} yH, y:\forall^{\mathrm{int}} yF \vdash (z)\lambda\bar{y}\lambda x(x)(y)\bar{y} : G}\ (\exists^{\mathrm{int}}_{l})$$

$$\frac{z:\exists^{\mathrm{int}} yH, y:\forall^{\mathrm{int}} yF \vdash (z)\lambda\bar{y}\lambda x(x)(y)\bar{y} : G}{\bar{x}:\mathrm{int}(x), z:\forall^{\mathrm{int}} x\exists^{\mathrm{int}} yH, y:\forall^{\mathrm{int}} yF \vdash (z)\bar{x}\lambda\bar{y}\lambda x(x)(y)\bar{y} : G}\ (\forall^{\mathrm{int}}_{l})$$

$$\frac{\bar{x}:\mathrm{int}(x), z:\forall^{\mathrm{int}} x\exists^{\mathrm{int}} yH, y:\forall^{\mathrm{int}} yF \vdash (z)\bar{x}\lambda\bar{y}\lambda x(x)(y)\bar{y} : G}{z:\forall^{\mathrm{int}} x\exists^{\mathrm{int}} yH, u:\exists^{\mathrm{int}} x\forall^{\mathrm{int}} yF \vdash (u)\lambda\bar{x}\lambda y(z)\bar{x}\lambda\bar{y}\lambda x(x)(y)\bar{y} : G}\ (\exists^{\mathrm{int}}_{l})$$

$$\frac{z:\forall^{\mathrm{int}} x\exists^{\mathrm{int}} yH, u:\exists^{\mathrm{int}} x\forall^{\mathrm{int}} yF \vdash (u)\lambda\bar{x}\lambda y(z)\bar{x}\lambda\bar{y}\lambda x(x)(y)\bar{y} : G}{\bar{y}':\mathrm{int}(y'), z:\forall^{\mathrm{int}} y'x\exists^{\mathrm{int}} yH, u:\exists^{\mathrm{int}} x\forall^{\mathrm{int}} yF \vdash (u)\lambda\bar{x}\lambda y(z)\bar{y}'\bar{x}\lambda\bar{y}\lambda x(x)(y)\bar{y} : G}\ (\forall^{\mathrm{int}}_{l})$$

$$\frac{\bar{y}':\mathrm{int}(y'), z:\forall^{\mathrm{int}} y'x\exists^{\mathrm{int}} yH, u:\exists^{\mathrm{int}} x\forall^{\mathrm{int}} yF \vdash (u)\lambda\bar{x}\lambda y(z)\bar{y}'\bar{x}\lambda\bar{y}\lambda x(x)(y)\bar{y} : G}{z:\forall^{\mathrm{int}} y'x\exists^{\mathrm{int}} yH, u:\exists^{\mathrm{int}} x\forall^{\mathrm{int}} yF \vdash \lambda\bar{y}'(u)\lambda\bar{x}\lambda y(z)\bar{y}'\bar{x}\lambda\bar{y}\lambda x(x)(y)\bar{y} : \forall^{\mathrm{int}} y'G}\ (\forall^{\mathrm{int}}_{r})$$

$$\frac{z:\forall^{\mathrm{int}} y'x\exists^{\mathrm{int}} yH, u:\exists^{\mathrm{int}} x\forall^{\mathrm{int}} yF \vdash \lambda\bar{y}'(u)\lambda\bar{x}\lambda y(z)\bar{y}'\bar{x}\lambda\bar{y}\lambda x(x)(y)\bar{y} : \forall^{\mathrm{int}} y'G}{\bar{x}':\mathrm{int}(x'), z:\forall^{\mathrm{int}} y'x\exists^{\mathrm{int}} yH, u:\exists^{\mathrm{int}} x\forall^{\mathrm{int}} yF \vdash \lambda v(v)\bar{x}'\lambda\bar{y}'(u)\lambda\bar{x}\lambda y(z)\bar{y}'\bar{x}\lambda\bar{y}\lambda x(x)(y)\bar{y} : \exists^{\mathrm{int}} x'\forall^{\mathrm{int}} y'G}\ (\exists^{\mathrm{int}}_{r})$$

$$\frac{\bar{x}':\mathrm{int}(x'), z:\forall^{\mathrm{int}} y'x\exists^{\mathrm{int}} yH, u:\exists^{\mathrm{int}} x\forall^{\mathrm{int}} yF \vdash \lambda v(v)\bar{x}'\lambda\bar{y}'(u)\lambda\bar{x}\lambda y(z)\bar{y}'\bar{x}\lambda\bar{y}\lambda x(x)(y)\bar{y} : \exists^{\mathrm{int}} x'\forall^{\mathrm{int}} y'G}{t:\exists^{\mathrm{int}} x'\forall^{\mathrm{int}} y'x\exists^{\mathrm{int}} yH, u:\exists^{\mathrm{int}} x\forall^{\mathrm{int}} yF \vdash (t)\lambda\bar{x}'\lambda z\lambda v(v)\bar{x}'\lambda\bar{y}'(u)\lambda\bar{x}\lambda y(z)\bar{y}'\bar{x}\lambda\bar{y}\lambda x(x)(y)\bar{y} : \exists^{\mathrm{int}} x'\forall^{\mathrm{int}} y'G}\ (\exists^{\mathrm{int}}_{l})$$

□

Example 6.29. Given $t \Vdash \exists^{\mathrm{int}} x'\forall^{\mathrm{int}} y'x\exists^{\mathrm{int}} yH$ and $u \Vdash \exists^{\mathrm{int}} x\forall^{\mathrm{int}} yF$, by the soundness lemma: $(t)\lambda\bar{x}'\lambda z\lambda v(v)\bar{x}'\lambda\bar{y}'(u)\lambda\bar{x}\lambda y(z)\bar{y}'\bar{x}\lambda\bar{y}\lambda x(x)(y)\bar{y} \Vdash \exists^{\mathrm{int}} x'\forall^{\mathrm{int}} y'G$

In order to describe the strategy implemented by the above term, we will modify it slightly. Concretely, we add storage operators to be able to use 4.4 and their consequences.

Let us define:

$\theta := \lambda\bar{x}'\lambda z\lambda v(v)\bar{x}'\eta[\mathrm{z}]$;

$\eta[z] := \lambda\overline{y}'(u)\,\mathrm{T_{int}}\,\varphi[z,\overline{y}'];$
$\varphi[z,\overline{y}'] := \lambda\overline{x}\lambda y(z)\overline{y}'\overline{x}\,\mathrm{T_{int}}\,\psi[y];$
$\psi[y] := \lambda\overline{y}\lambda x(x)(y)\overline{y}.$

If a term $v \Vdash \overset{\text{int}}{\forall} x\varphi(x)$, then $\mathrm{T_{int}}\,v$ also realizes it. Applying this fact and the soundness lemma, we can assert that $\sigma := (t)\theta \Vdash \overset{\text{int}}{\exists} x' \overset{\text{int}}{\forall} y' G$ and hence σ is a program that computes a winning strategy for the game associated to the formula $\overset{\text{int}}{\exists} x' \overset{\text{int}}{\forall} y' G$.

As we have seen in 6.16, using 6.16.19 the thread scheme of a play implemented by σ is as follows:

(6.29.25)
$$\begin{gathered}
\sigma \star \mathrm{T_{int}}\,k_0.\pi_0 \succ k_0 \star \overline{m}_1.\zeta_1.\pi_0 \\
(\zeta_1)\overline{q}_1 \star k_1.\pi_1 \succ k_0 \star \overline{m}_2.\zeta_2.\pi_0 \\
\vdots \\
(\zeta_r)\overline{q}_r \star k_r.\pi_r \succ k_i \star \pi_i
\end{gathered}$$

where, we denote as k_0 the interaction instruction $\varkappa_{\Upsilon_1}$, ζ_i is the i-th exception handler and k_i denotes $\varkappa_{\mathbb{W}_i(m_i,q_i)}$ for a predicate $\mathbb{W}_i \in \Delta_1$ chosen by $\textcircled{\forall}$.

We start the algorithmic description of σ determining some notations and objects that will be in force thereafter. The description is about a play we will denote as $\mathcal{G}$, which is played between $\textcircled{\exists}$ and $\textcircled{\forall}$ in the game associated with the formula Υ. As the game evolves, we will play $\mathcal{M}$ according to the game associated with Φ. $\mathcal{M}$ will be played by (the strategy implemented by) t against $\textcircled{\forall}$. Each time the empty move is chosen by t, we will start a new thread on $\mathcal{G}$ and conversely, each new thread in $\mathcal{G}$ corresponds to the choice of the empty move in $\mathcal{M}$. Moreover, associated with the i-th thread of $\mathcal{G}$, we will have a play $\mathcal{N}^i$, which will be played by (the strategy implemented by) u against $\textcircled{\forall}$. Each $\mathcal{N}^i$ is created when the i-th thread of $\mathcal{G}$ is. Each $\mathcal{N}^i$ evolves independently from the other $(\mathcal{N}^{i'})_{i'\neq i}$.

The description will be performed by induction on the number of $\mathcal{G}$ threads, building at each step a family of terms, integers, threads, substitutions and notations. At each step, we must verify that the updated construction satisfies the properties (A) and (B) defined below. The reader should keep in mind the notations we adopted in 6.16 and 6.22 to describe thread schemes in the games associated with Φ and Ψ. In particular, the exception handler of Φ associated with a first level move will be denoted as χ and the exception handler associated with second level moves will be denoted as ι. For the exception handler of Ψ, we adopt the notation ξ. The terms which appears in thread scheme of $\mathcal{N}^i$ will be indexed with an upper index i, thus indicating they are took from $\mathcal{N}^i$.

Inductive construction hypothesis for the step i:

(6.29.26)

(A) $\mathcal{M}$ is a not ended play between t and $\textcircled{\forall}$ such that:

(A.1) There are $i-1$ already played "first level" moves $(m_{i'},q_{i'},n^{i'})_{1\leq i'\leq i}$, the exception handler of which is denoted as $\chi_{i'}$.
The process starting the thread corresponding to the move $(m_{i'},q_{i'},n^{i'})$, which is $(\chi_{i'})\overline{q}_{i'}\overline{n}^{i'}\star U^{i'}.\pi_{i'}$, will be abbreviated as $P_{i'}$. Since all threads of $\mathcal{M}$ are finite, we have also a process $R_{i'}$, which ends $\mathrm{th}_{P_{i'}}$.

(A.2) The "second level moves" are 4-uplets $(m_{i'},q_{i'},n^{i'},p^{i'}_{j'})$ indexed by pairs (i',j') such that $j'\in J_{i'}$ (c.f.: 6.22). The exception handler associated with $(m_{i'},q_{i'},n^{i'},p^{i'}_{j'})$ is denoted as $\iota^{i'}_{j'}$. The process starting the thread corresponding to the move $(m_{i'},q_{i'},n^{i'},p^{i'}_{j'})$, will be denoted as $P^{i'}_{j'}$ and the corresponding process ending up $\mathrm{th}_{P^{i'}_{j'}}$ in $R^{i'}_{j'}$.

(A.3) For each $i'<i$ there is a dynamic substitution $\mathscr{E}_{i'}$ such that $(\mathscr{E}_{i'})^*$ replace all the substitutable instructions appeared in all threads arising before the move $(m_{i'+1},q_{i'+1},n^{i'+1})$ was played in $\mathcal{M}$. Moreover, for all $i''<i'<i$ we have that $\mathscr{E}_{i'}\sqsupseteq\mathscr{E}_{i''}$.

(B) There are $i-1$ not ended plays $(\mathcal{N}^{i'})_{1\leq i'<i}$, which are played by u against $\textcircled{\forall}$.

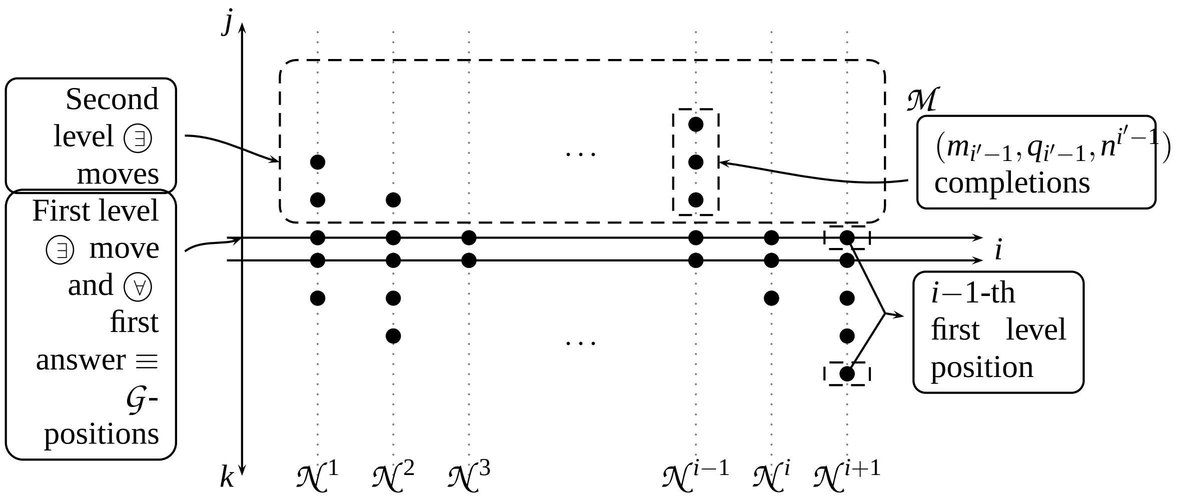
(B.1) There are $k^{i'}$ played moves on $\mathcal{N}^{i'}$, which are 2-uplets $(n^{i'},p^{i'}_{k'})$ where $1\leq k'\leq k^{i'}$. The exception handler associated with $(n^{i'},p^{i'}_{k'})$ is denoted as $\xi^{i'}_{k'}$. In particular, the last played exception handler $\xi^{i'}_{k^{i'}}$ is denoted as $\xi^{i'}$.

(B.2) For each $i'<i$ there is a dynamic substitution $\mathscr{F}^{i'}$ such that $(\mathscr{F})^*$ replace all the substitutable instructions appearing in the (current) threads scheme of $\mathcal{N}^{i'}$. The processes starting the thread corresponding to the move $(n^{i'},p^{i'}_{k'})$ (which is $\xi^{i'}_{k'}\star k^{i'}_{j'}.\overline{\pi}^{i'}_{j'}$), will be abbreviated as $Q^{i'}_{j'}$.

(B.3) Given a played position $(n^{i'}_k,p^{i'}_k)$ taken from the (current) play $\mathcal{N}^{i'}$ ($i'<i$ and $k\leq k^{i'}$), there are two integers (m,q) such that $h(m,q,n^{i'}_k,p^{i'}_k)=0$.

The reader should be aware that there are some objects which will be redefined during the play $\mathcal{G}$. More precisely, the integers $n^{i'}$, $k^{i'}$, the exception handler $\xi^{i'}$ and the dynamic substitutions are "current objects", which are determined only for each construction step.

In order to explain the interaction between these two strategies, we adopt a graphical simultaneous representation of $\mathcal{M}$ and the plays $\mathcal{N}^i$:



j
k
i
Second level ∃ moves
First level ∃ move and ∀ first answer ≡ $\mathcal{G}$-positions
$\mathcal{M}$
$(m_{i'-1}, q_{i'-1}, n^{i'-1})$ completions
$i-1$-th first level position
$\mathcal{N}^1$ $\mathcal{N}^2$ $\mathcal{N}^3$ $\mathcal{N}^{i-1}$ $\mathcal{N}^i$ $\mathcal{N}^{i+1}$

We take two parallel axes i, one containing dots that represents the pairs (m_i, q_i) and the other containing dots representing the first move n_1^i played by u in $\mathcal{N}^i$. All moves played by u are represented by dots at the down side of the scheme, more precisely, each new move played by u in $\mathcal{N}^i$ is represented by a new dot in the i-th vertical line. The last played move of each $\mathcal{N}^i$ –i.e.: the current value n^i– plays an important role and then it will be represented by a white-filled dot. Finally, the second level positions p_j^i played by t in $\mathcal{M}$ are represented in the upper side of the scheme. More precisely, a new move p_j^i that completes the first level position (m_i, q_i, n^i) will be represented as a new dot in the i-th vertical line.

In consequence, all current first level positions in $\mathcal{M}$ are represented by two dots in a vertical line: the black dot over the upper i-axis and the white-filled dot in the down side. All played completions of the i-th first level position are in their upper side of the scheme, over the i-th vertical line.

Start the play $\mathcal{G}$ implemented by σ against ∀ on the game associated with Υ.

Step 0: The process $\sigma \star \mathrm{T_{int}} \varkappa_{\Upsilon_1}.\pi_0$ reduces to $t \star \theta.\mathrm{T_{int}} \varkappa_{\Upsilon_1}.\pi_0$. Define the processes $P_0 := t \star U^0.\pi_0$. Start a play $\mathcal{M}$ between t and ∀, the thread scheme of which has

$$t \star U^0.\pi_0 \succ H^0_{\mu_1\chi_1} \star \overline{m}_1.\pi^0_{\mu_1\chi_1} \tag{6.29.27}$$

as its first thread, where μ_1, χ_1 are two suitable Λ_c-terms. For brevity, lets denote H_1^0 instead of $H^0_{\mu_1\chi_1}$ and π_1^0 instead of $\pi^0_{\mu_1\chi_1}$. As we have seen in 6.22, the integer m_1 and the term χ_1 are respectively the first ∃ move and its corresponding first exception handler –at least of substitution– played in $\mathcal{M}$.

Given any two Λ_c-terms μ, χ we have that

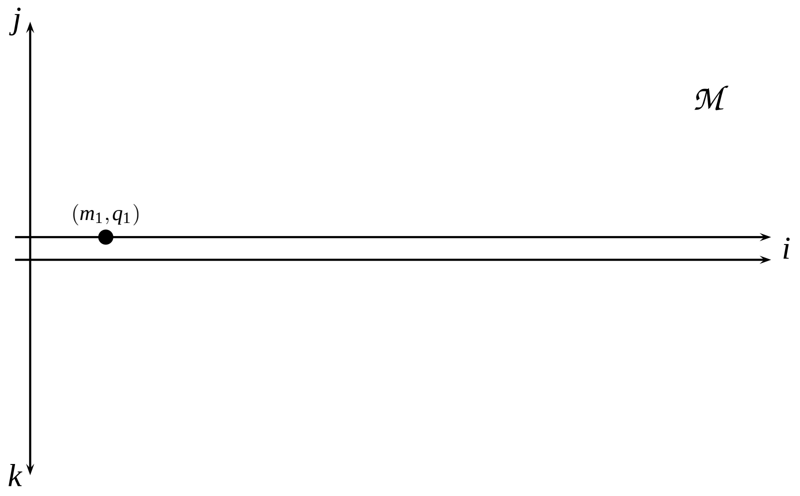
$$\theta \star \mu.\chi.\mathrm{T_{int}} \varkappa_{\Upsilon_1}.\pi_0 \succ \mathrm{T_{int}} \star \varkappa_{\Upsilon_1}.\mu.\eta[\chi].\pi_0$$

and hence $\theta \in \mathcal{F}_{\mathrm{T_{int}}K}(\mathrm{T_{int}}\varkappa_{\Upsilon_1}.\pi_0)$, i.e. we can dynamically replace U^0 by θ. Applying to 6.29.27 $\mathscr{E}_0 := [^{\mathrm{T_{int}}\varkappa_{\Upsilon_1}.\pi_0}/_{\pi_0}]\ ^{P_0}\langle\theta/U^0\rangle$ (c.f.: 4.11), we obtain:

$$t \star \theta.\,\mathrm{T_{int}}\varkappa_{\Upsilon_1}.\pi_0 \succ \varkappa_{\Upsilon_1} \star \overline{m}_1.\eta[\mathscr{E}_0(\chi_1)].\pi_0$$

In consequence, the first move proposed in $\mathcal{G}$ by the strategy implemented by σ is no other than the first move proposed by t in $\mathcal{M}$. The first exception handler ζ_1 proposed by σ is $\eta[\mathscr{E}_0(\chi_1)]$, term which contains the first exception handler proposed by t, but modified by substitution.

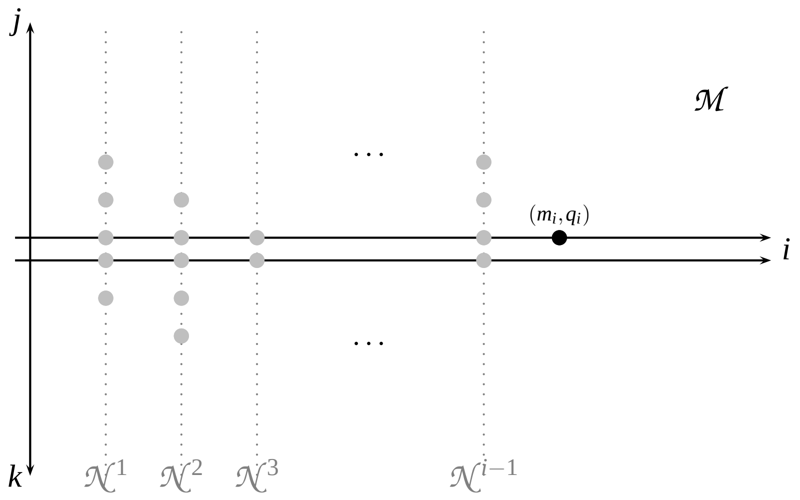
Since the interaction instruction $\varkappa_{\Upsilon_1}$ has arrived in head position, $\textcircled{\forall}$ must answer an integer q_1 and a predicate $\mathbb{W}_1 \in \Delta_1$. The execution continues reducing the process $\eta[\mathscr{E}_0(\chi_1)] \star \overline{q}_1.\varkappa_1.\pi_1$, where $\varkappa_1$ abbreviates $\varkappa_{\mathbb{W}_1(g(m_1,q_1))}$ and $\pi_1 \in \mathbb{W}_1(0)$. Our scheme takes its first dot:



Step i: Now we will describe the $i-$th thread of the play $\mathcal{G}$. We suppose we have all objects that were defined in 6.29.26 and that they satisfies properties A.1, A.2, A.3, B.1, B.2 and B.3. The strategy t has just proposed its i-th first level move m_i and $\textcircled{\forall}$ – playing against the composed strategy in $\mathcal{G}$– has just answered q_i. In order to do a simplest description, as we said, we will use for each index $i' \leq i$ three "current values" $k^{i'}, \xi^{i'}, n^{i'}$ defined respectively as the number of moves played in $\mathcal{N}^{i'}$, the last exception handler played in $\mathcal{N}^{i'}$ and its corresponding move. Hence, $\xi^{i'} = \xi^{i'}_{k^{i'}}$ and $n^{i'} = n^{i'}_{k^{i'}}$. These values will change along the play, but never we will use an "old" value of $\xi^{i'}$ or $n^{i'}$, except when the play $\mathcal{G}$ is stopped by the strategy u[9].

We are examining the thread of $(\eta[\mathscr{E}_{i-1}(\chi_i)])\overline{q}_i \star \varkappa_i.\pi_i$, where $\varkappa_i = \varkappa_{\mathbb{W}_i(g(m_i,q_i))}$, $\pi_i \in \mathbb{W}_i(0)$ and $\mathbb{W}_i \in \Delta_1$. The current played moves of $\mathcal{M}$ and $(\mathcal{N}^{i'})_{i'<i}$ are represented in the following scheme:

[9]This assertion will be established once this discussion will ended.



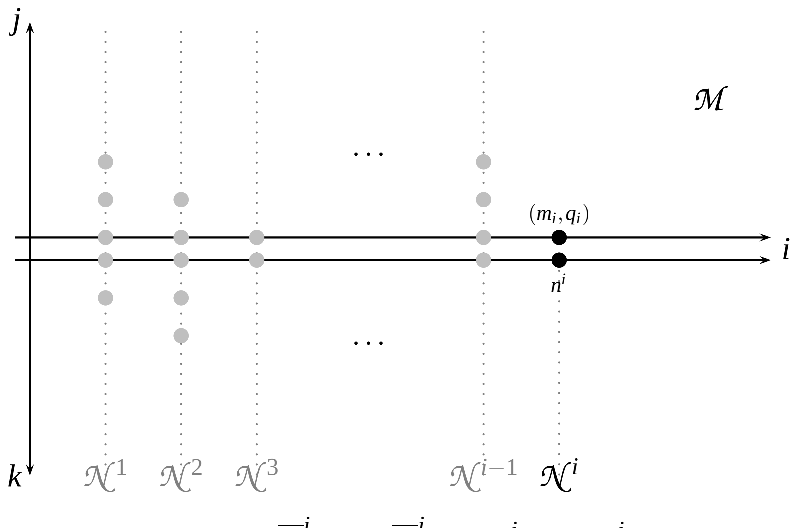
Reducing the first process of the i-th thread, we obtain:

$$u \star \mathrm{T_{int}}\,\varphi[\mathscr{E}_{i-1}(\chi_i),\overline{q}_i].\varkappa_i.\pi_i \tag{6.29.28}$$

In consequence, u is arrived in head position. Start a play $\mathcal{N}^i$ implemented by u against $\textcircled{\forall}$. The first thread of this play is

$$u \star V^i.\overline{\pi}_i \succ \overline{H}^i_{\nu^i_1\xi^i_1} \star \overline{n}^i_1.\overline{\pi}^i_{\nu^i_1\xi^i_1} \tag{6.29.29}$$

where ν^i_1,ξ^i_1 are two Λ_c-terms, ξ^i_1 is –at least of substitution– the first exception handler of the $i-$th play $\mathcal{N}^i$. Moreover, the Krivine's integer $\overline{n}^i_1$ is the first move played by u in $\mathcal{N}^i$. In particular, for all $i'<i$ $\overline{n}^i_1 = \overline{n}^{i'}_1$ but it is only true for the first moves played by u. Then, as we defined above, $k^i := 1$, $n^i := n^i_1$ and $\xi^i := \xi^i_1$. We add to our scheme the integer n^i just created by u:



For brevity, denote $\overline{H}^i_{\nu^i_1\xi^i_1}$ as $\overline{H}^i_1$ and $\overline{\pi}^i_{\nu^i_1\xi^i_1}$ as $\overline{\pi}^i_1$.

Since $\mathrm{T_{int}}\,\varphi \in \mathcal{F}_{\mathrm{T_{int}}K}(\varkappa_i.\pi_i)$, we can apply to 6.29.29 the substitution

$$\mathscr{F}^i := [{}^{\varkappa_i.\pi_i}/_{\overline{\pi}_i}]^{Q_0^i}\langle \mathrm{T_{int}}\,\varphi[\mathscr{E}_{i-1}(\chi_i),\overline{q}_i]/V^i\rangle \tag{6.29.30}$$

to obtain:

(6.29.31)

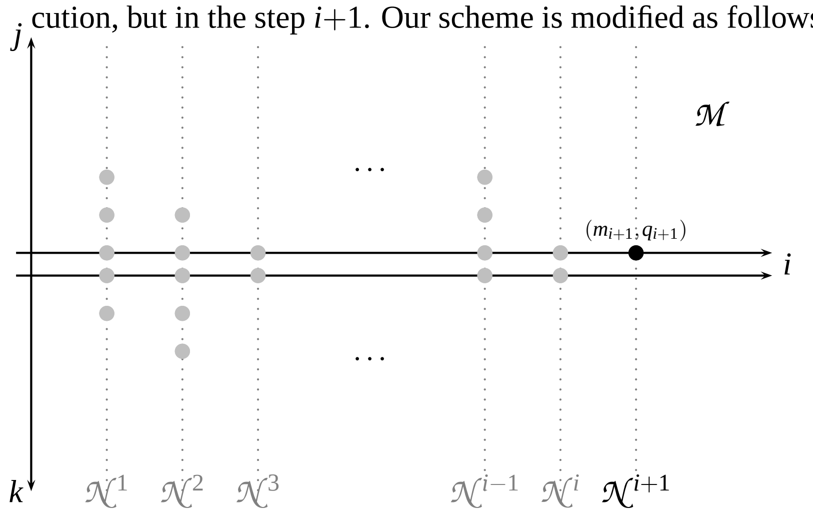
$$u \star \mathrm{T_{int}}\,\varphi[\mathscr{E}_{i-1}(\chi_i),\overline{q}_i].\varkappa_i.\pi_i \succ \varphi[\mathscr{E}_{i-1}(\chi_i),\overline{q}_i] \star \overline{n}^i.\mathscr{F}^i(\xi^i).\varkappa_i.\pi_i$$
$$\text{which reduces to} \quad \mathscr{E}_{i-1}(\chi_i) \star \overline{q}_i.\overline{n}^i.\,\mathrm{T_{int}}\,\psi[\mathscr{F}^i(\xi^i)].\varkappa_i.\pi_i$$

Remark that as it is defined, $\mathscr{F}^i$ satisfies (B.2) in 6.29.26. Define

$$\mathscr{E}_i := \mathscr{E}_{i-1} \rhd [{}^{\varkappa_i.\pi_i}/_{\pi_i}]^{P_i}\langle \mathrm{T_{int}}\,\psi[\mathscr{F}^i(\xi^i)]/U^i\rangle \tag{6.29.32}$$

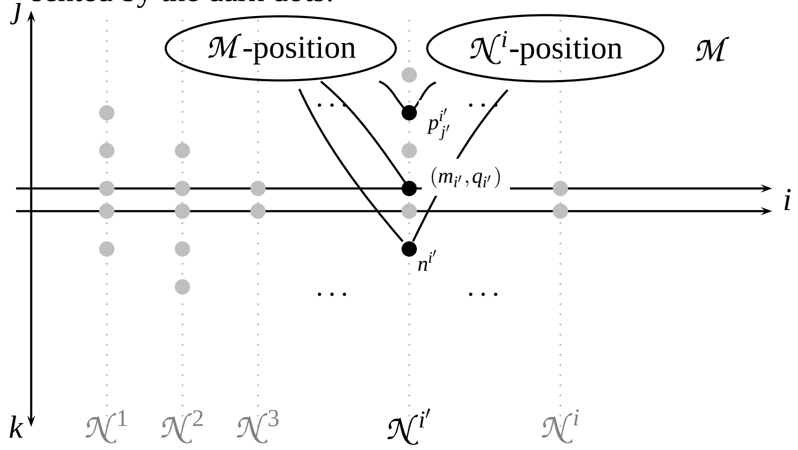
Now, we add the thread $\mathrm{th}_{(\chi_i)\overline{q}_i\overline{n}^i\star U^i.\pi_i}$ to the thread scheme of $\mathcal{M}$; which means that we add the position (m_i, q_i, n^i) to the play $\mathcal{M}$. This new thread ends in a process R_i that has a constant $H^{i'}$ in head position. We discuss according to R_i (c.f. 6.22):

(a) $R_i = H^0_{\mu_{i+1}\chi_{i+1}} \star \overline{m}_{i+1}.\pi^0_{\mu_{i+1}\chi_{i+1}}$. It corresponds to the fact that t, playing instead of ⓔ, chooses in $\mathcal{M}$ the empty position and proposes a new exception handler χ_{i+1} and a new integer $\overline{m}_{i+1}$. Applying $\mathscr{E}_i$ to the i-th thread of $\mathcal{M}$, we obtain the process $\varkappa_{\Upsilon_1} \star \overline{m}_{i+1}.\eta[\mathscr{E}_i(\chi_{i+1})].\pi_0$ and hence the i-th $\mathcal{G}$-thread is finished. The strategy σ has just proposed by means of t a new move m_{i+1} and it stands for a new ⓐ answer. Once the $i+1$-th answer is given by ⓐ the game continues according to our discution, but in the step $i+1$. Our scheme is modified as follows:



The conditions (A.1), (A.2), (A.3), (B.1) and (B.2) are satisfied. In particular, the new substitution $\mathscr{F}^i$ verifies (B.2) and the new substitution $\mathscr{E}_i$ verifies (A.3) because we added to the threads two new constants π_i and U^i, which are just the constants that $\mathscr{E}_i$ "adds" to $\mathscr{E}_{i-1}$.

(b) $R_i = K_{i'j'} \star \pi_{i'j'}$. It corresponds to the fact that t playing instead of $\textcircled{\exists}$ chooses in $\mathcal{M}$ the (complete) position $(m_{i'}, q_{i'}, n^{i'}, p^{i'}_{j'})$. Since the just played position (m_i, q_i, n^i) was not yet completed, we know that $i' < i$. In our scheme, $(m_{i'}, q_{i'}, n^{i'}, p^{i'}_{j'})$ is represented by the dark dots:



Furthermore, t implements a winning strategy for the game of Φ and hence the 4-uplet chosen by t satisfies the equation $h(m_{i'}, q_{i'}, n^{i'}, p^{i'}_{j'}){=}0$. Such as $\mathcal{E}_i$ was defined, we have:

$$\mathcal{E}_i \sqsupseteq \mathcal{E}_{i'+1} \sqsupseteq {}^{10}[\varkappa_{i'}.\pi_{i'} / \pi_{i'j'}]^{P^{i'}_{j'}} \langle (\mathscr{F}^{i'}(\xi^{i'}))\overline{p}^{i'}_{j'} / K_{i'j'} \rangle$$

where $K_{i'j'}$ abbreviates $K_{\mathbb{W} f(m_{i'}, q_{i'}, n^{i'}, p^{i'}_{j'})}$ and $\mathbb{W}$ is a Δ_1-parameter chosen by $\textcircled{\forall}$ after $\textcircled{\exists}$ has played $p^{i'}_{j'}$ in $\mathcal{M}$ (c.f.: 6.22). Applying $\mathcal{E}_i$ to the $i{+}1$-th thread of $\mathcal{M}$, we obtain:

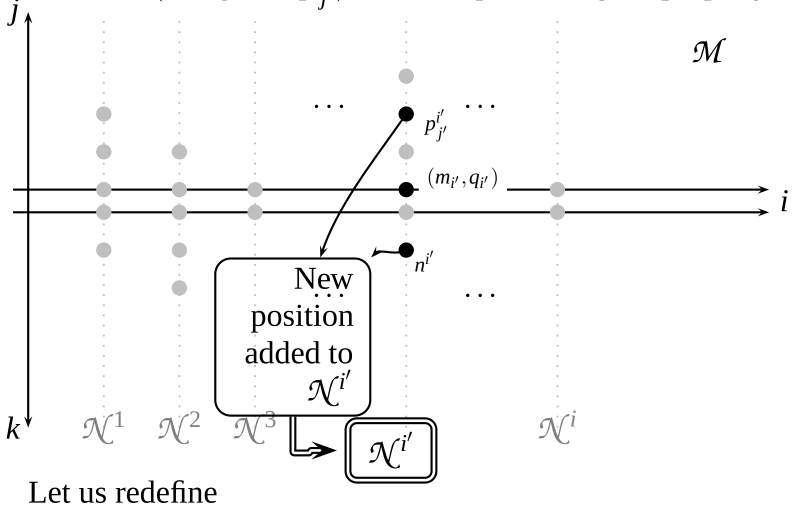
$$(\mathcal{E}_{i-1}(\chi_i))\overline{q}_i\overline{n}^i \star \mathrm{T}_{\mathrm{int}}\, \psi[\mathscr{F}^i(\xi^i)].\varkappa_i.\pi_i \succ (\mathscr{F}^{i'}(\xi^{i'}))\overline{p}^{i'}_{j'} \star \varkappa_{i'}.\pi_{i'}$$

Let us add to the play $\mathcal{N}^{i'}$ the position $(n^{i'}, p^{i'}_{j'})$. In other words, we are taking $p^{i'}_{j'}$ as the $\textcircled{\forall}$-answer to the $\textcircled{\exists}$-move $n^{i'}$. We must add to the thread scheme of $\mathcal{N}^{i'}$, the new thread $\mathrm{th}_{(\xi^{i'})\overline{p}^{i'}_{j'} \star K^{i'}_{k^{i'}}.\pi^{i'}_{k^{i'}}}$ –where $K^{i'}_{k^{i'}}$ abbreviates $K_{\mathbb{V} g(n^{i'}_{k^{i'}}, p^{i'}_{j'})}$ and $\mathbb{V}$ is a Δ_1-parameter. The reader should remark that it is from now that the position $(n^{i'}, p^{i'}_{j'})$ is effectively played in $\mathcal{N}^{i'}$[11] and that this new

[10] c.f. item (c) in this discussion and 6.29.32

[11] Usually in thread schemes, indexes in moves and answers corresponding to the same position are equals. Here, it seems to be a mistake because a $\mathcal{N}^{i'}$-move is indexed as $(n^{i'}, p^{i'}_{j'})$, which is in fact $(n^{i'}_{k^{i'}}, p^{i'}_{j'})$. Nothing is wrong about this "discoordination" because the index j' indicates that the move $p^{i'}_{j'}$ is the j'-th completion of the i'-th move of $\mathcal{M}$, while the index $k^{i'}$ in $n^{i'}_{k^{i'}}$ is related to the moves in $\mathcal{N}^{i'}$.

$\mathcal{N}^{i'}$-position is associated to the 4-uplet $(m_{i'}, q_{i'}, n^{i'}, p^{i'}_{j'})$ which satisfies $h(m_{i'}, q_{i'}, n^{i'}, p^{i'}_{j'}) = 0$ (thus preserving the property B.3):



Let us redefine

$$\mathscr{F}^{i'} := \mathscr{F}^{i'} \triangleright [{}^{\varkappa_{i'}, \pi_{i'}} / {}_{K^{i'}_{k^{i'}}, \pi^{i'}_{k^{i'}}}]^{Q^{i'}_{k^{i'}}} \langle\rangle \tag{6.29.33}$$

where $Q^{i'}_{k^{i'}} = (\xi^{i'})\overline{p}^{i'}_{j'} \star K^{i'}_{k^{i'}}.\pi^{i'}_{k^{i'}}$ Here there are two possibilities according to the choice given by the strategy implemented by u:

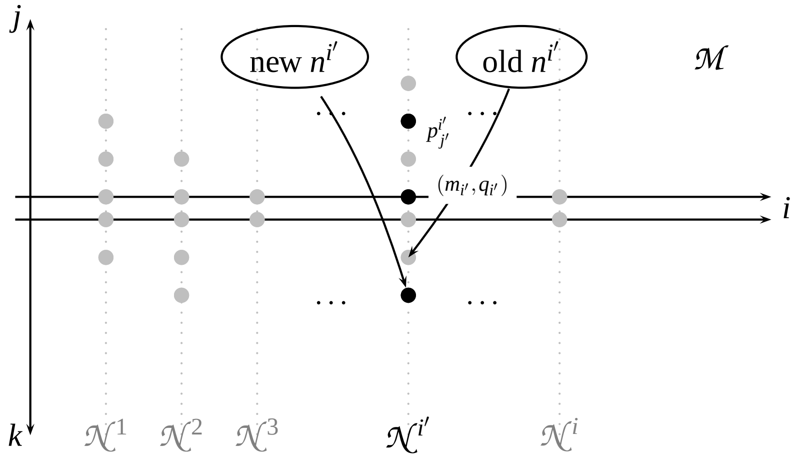
(i) u chooses a position $(n^{i'}_{k''}, p^{i'}_{j''})$ played before ($\xi^{i'}$ only knows about the $\mathcal{N}^{i'}$-positions played before $\xi^{i'}$ was created in $\mathcal{N}^{i'}$). Since u implements a winning strategy for the game of Ψ, we have that $f(n^{i'}_{k''}, p^{i'}_{j''}) = 0$. Moreover, by induction hypothesis, $(n^{i'}_{k''}, p^{i'}_{j''})$ has associated a 4-uplet $(m_{i'}, q_{i'}, n^{i'}_{k''}, p^{i'}_{j''})$, which was chosen before by the strategy implemented by t (c.f. the item (b) of this discussion). In consequence $h(m_{i'}, q_{i'}, n^{i'}_{k''}, p^{i'}_{j''}) = 0$. How is this choice reflected on the thread we are describing? The thread $\mathrm{th}_{(\xi^{i'})\overline{p}^{i'}_{j'} \star k_{i'}.\pi_{i'}}$ we added in (b) to the thread scheme of $\mathcal{N}^{i'}$ must be ended by $K^{i'}_{k''} \star \overline{\pi}^{i'}_{k''}$. Such as $\mathscr{F}^{i'}$ is defined (c.f.: 6.29.33), $\mathscr{F}^{i'} \sqsupseteq [{}^{\varkappa_{i'}, \pi_{i'}} / {}_{K^{i'}_{k''}, \overline{\pi}^{i'}_{k''}}]$. Hence applying $\mathscr{F}^{i'}$ to the last thread of $\mathcal{N}^{i'}$, we obtain:

$$(\mathscr{F}^{i'}(\xi^{i'}))\overline{p}^{i'}_{j'} \star \varkappa_{i'}.\pi_{i'} \succ \varkappa_{i'} \star \pi_{i'}$$

and then the session corresponding to $\mathcal{G}$ is ended in the process $\varkappa_{i'} \star \pi_{i'}$.
Since $h(m_{i'}, q_{i'}, n^{i'}_{k''}, p^{i'}_{j''}) = 0$ and $f(n^{i'}_{k''}, p^{i'}_{j''}) = 0$, then $g(m_{i'}, q_{i'}) = 0$ and hence $\exists$ has win[12].

[12]Of course, we know it because $\sigma \Vdash \Upsilon$. However, it is interesting to remark that, since σ combines any two strategies for Φ and Ψ, σ can choose a move (m_i, q_i) only if it knows

(ii) u chooses the empty position. Let us increment $k^{i'}$: $k^{i'}{:=}k^{i'}+1$. The strategy played by u must complete the empty move proposing a new integer $n^{i'}_{k^{i'}}$ and a new exception handler $\xi^{i'}_{k^{i'}}$. Our scheme will be modified in the line representing the play $\mathcal{N}^{i'}$, thus adding a new dot representing the new $n^{i'}$:



By definition, the current values $n^{i'}$ and $\xi^{i'}$ must be redefined by $n^{i'}{:=}n^{i'}_{k^{i'}}$ and $\xi^{i'}{:=}\xi^{i'}_{k^{i'}}$. The last (current) thread of $\mathcal{N}^{i'}$ must end with $\overline{H}^{i'}_{\nu^{i'}\xi^{i'}}\star\overline{n}^{i'}.\overline{\pi}^{i'}_{\nu^{i'}\xi^{i'}}$ for a suitable term $\nu^{i'}$. Such it was defined in the i'-th step (c.f.: 6.29.30), we have that $\mathscr{F}^{i'}\sqsupseteq[{}^{\varkappa_{i'}.\pi_{i'}}/_{\overline{\pi}_{i'}}]^{Q^{i'}_0}\langle \mathrm{T}_{\mathrm{int}}\,\varphi[\mathscr{E}_{i'-1}(\chi_{i'}),\overline{q}_{i'}]/V^{i'}\rangle$ Applying $\mathscr{F}^{i'}$ to this thread, we obtain that[13]:

$$(\mathscr{F}^{i'}(\xi^{i'}_{k^{i'}-1}))\overline{p}^{i'}_{j'}\star\varkappa_{i'}.\pi_{i'}\succ\varphi[\mathscr{E}_{i'-1}(\chi_{i'}),\overline{q}_{i'}]\star\overline{n}^{i'}.\xi^{i'}.\varkappa_{i'}.\pi_{i'} \tag{6.29.34}$$

which reduces to $\mathscr{E}_{i'-1}(\chi_{i'})\star\overline{q}_{i'}.\overline{n}^{i'}.\mathrm{T}_{\mathrm{int}}\,\psi[\mathscr{F}^{i'}(\xi^{i'})].\varkappa_{i'}.\pi_{i'}$

The term $\mathscr{E}_{i'-1}(\chi_{i'})$ was in head position at least in the $i'-$th thread (c.f.: 6.29.29). What is the meaning in $\mathcal{M}$ of the fact that in $\mathcal{G}$ an exception handler built by t arrives many times in head position? The term $\mathscr{E}_{i'-1}(\chi_{i'})$ is the $i'-$th $\mathcal{M}$-exception handler, modified by the dynamic substitution $\mathscr{E}_{i'-1}$. This exception handler only knows the positions of $\mathcal{M}$ played before it was created. It is raisonable to think that, at this moment, all positions created

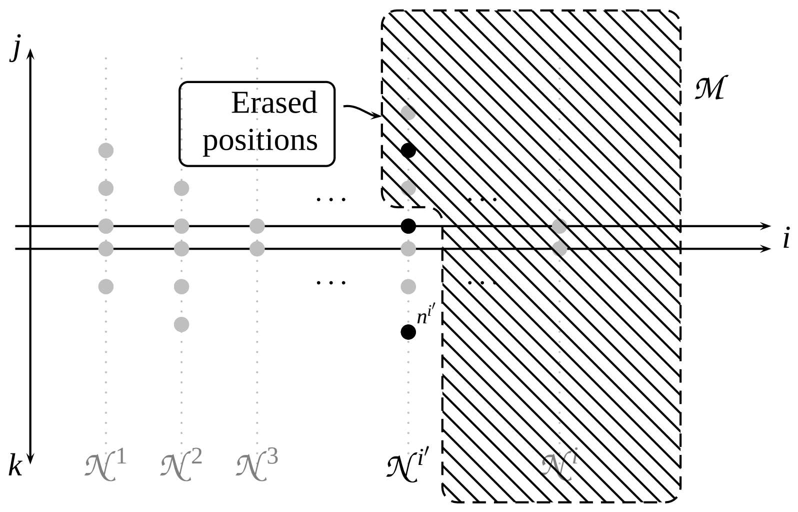
that there is a 4-uplet satisfying $h=0$, such that the first two coordinates are m_i,q_i and the last two satisfies the condition $f=0$. The program σ is built to find this 4-uplet using t and u.

[13]Remember that $k^{i'}$ is just updated and hence the old $k^{i'}$ becomes $k^{i'}-1$.

after the $i'-$th thread of $\mathcal{M}$ was; will be forgotten by the strategy t. Using dynamic substitutions, we can precise this intuition:
Let us erase all dynamic substitutions $\mathscr{E}_{i''}$ such that $i'' > i'$ and redefine $\mathscr{E}_{i'}$ by:

$$\mathscr{E}_{i'} := \mathscr{E}_{i'-1} \triangleright [{}^{\varkappa_{i'}.\pi_{i'}}/_{\pi_{i'}}]^{P_{i'}} \langle \mathrm{T}_{\mathrm{int}}\, \psi[\mathscr{F}^{i'}(\xi^{i'})/U^{i'}]\rangle \tag{6.29.35}$$

This is exactly the declaration used to define $\mathscr{E}_{i'}$ the first time we defined it, but now the current value $\xi^{i'}$ and the dynamic substitution $\mathscr{F}^{i'}$ are updated. Let us erase also all the positions and the corresponding threads played after $m_{i'}$ was proposed by t in $\mathcal{M}$. The situation in the scheme is modified as follows:



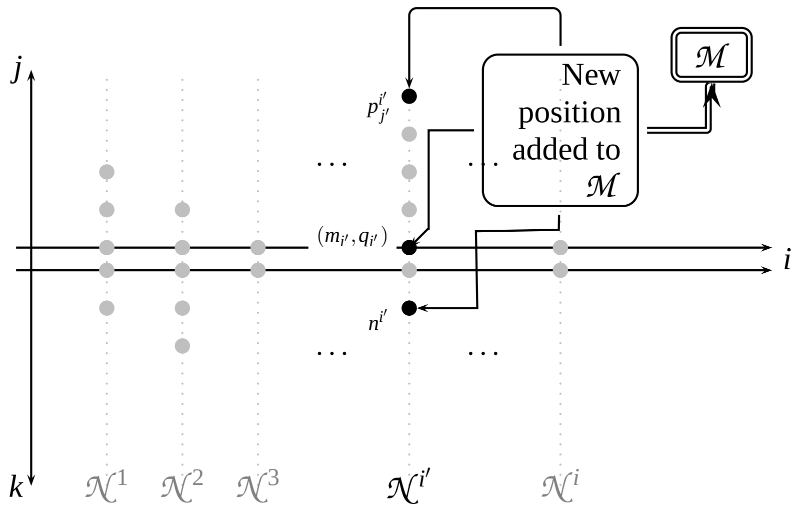
However, not all information we erased about the positions of $\mathcal{M}$ are forgotten by σ (the main program we are performing). Indeed, the new exception handler $\xi^{i'}$ knows all the positions played in $\mathcal{N}^{i'}$ and by construction each position played in $\mathcal{N}^{i'}$ is associated with a 4-uplet which was "approved" by the winning strategy implemented by t (i.e. chosen by t as a (winning) complete position for $\mathcal{M}$). In consequence, the move (m_i, q_i, n^i, p^i_j), which was forgotten by t, is still remembered by u. More precisely, u remember their last two coordinates, that it can choose at any moment to close the play $\mathcal{N}^{i'}$ (and hence the play $\mathcal{G}$). The reader should remark that, while normally only $\exists$ can back-track the execution, if another winning strategy (like u here) plays instead of $\forall$, it can back-track the execution too. By instance, here it was the strategy implemented by u that decides erase a

part of the play $\mathcal{M}$ and modify the integer $n^{i'}$ in the position $(m_{i'}, q_{i'}, n^{i'})$.
Let us add the thread $\mathrm{th}_{(\chi_{i'})\overline{q}_{i'}\overline{n}^{i'}\star U^{i'}.\pi_{i'}}$ to the thread scheme of $\mathcal{M}$. From now on, the discussion of this case continues according to the cases (a) and (b) treated before and the case (c) we treat inmediately, but now taking i' instead of i.

(c) $R_i = H^{i'}_{\rho^{i'}_{j'}\iota^{i'}_{j'}} \star \overline{p}^{i'}_{j'}.\pi^{i'}_{\rho^{i'}_{j'}\iota^{i'}_{j'}}$ for two suitable terms $\rho^{i'}_{j'}$, $\iota^{i'}_{j'}$. It corresponds to the fact that t, playing instead of $\exists\!\!\!\bigcirc$, chooses the (incomplete) position $(m_{i'}, q_{i'}, n^{i'})$, completes it by the move $p^{i'}_{j'}$ and proposes a new exception handler $\iota^{i'}_{j'}$. For brevity, lets denote $H^{i'}_{\rho^{i'}_{j'}\iota^{i'}_{j'}}$ as $H^{i'}_{j'}$ and $\pi^{i'}_{\rho^{i'}_{j'}\iota^{i'}_{j'}}$ as $\pi^{i'}_{j'}$.
By definition, $\mathscr{E}_i \sqsupseteq \mathscr{E}_{i'} \sqsupseteq [{}^{\varkappa_{i'}.\pi_{i'}}/_{\pi_{i'}}]^{P_{i'}} \langle \mathrm{T_{int}}\,\psi[\mathscr{F}^{i'}(\xi^{i'})]/U^{i'}\rangle$ (c.f.:6.29.32).
On the other hand, we have that $\mathrm{T_{int}}\,\psi[\mathscr{F}^{i'}(\xi^{i'})] \in \mathcal{F}_{\mathrm{T_{int}}K}(\varkappa_{i'}.\pi_{i'})$.
Then, we can apply $\mathscr{E}_i$ to the thread $\mathrm{th}_{(\chi_i)\overline{q}_i\overline{n}^i\star U^i.\pi_i}$, thus obtaining:

(6.29.36)
$$(\mathscr{E}_i(\chi_i))\overline{q}^i\overline{n}^i \star \mathrm{T_{int}}\,\psi[\mathscr{F}^{i'}(\xi^i)].\varkappa_i.\pi_i \succ \psi[\mathscr{F}^{i'}(\xi^{i'})] \star \overline{p}^{i'}_{j'}.\mathscr{E}_i(\iota^{i'}_{j'}).\varkappa_{i'}.\pi_{i'}$$
$$\text{which reduces to } \mathscr{E}_{i'}(\iota^{i'}_{j'}) \star (\mathscr{F}^{i'}(\xi^{i'}))\overline{p}^{i'}_{j'}.\varkappa_{i'}.\pi_{i'}$$[14]

The situation in the scheme is as follows:



We add $\mathrm{th}_{\iota^{i'}_{j'}\star K_{i'j'}.\pi_{i'j'}}$ to the threads scheme of $\mathcal{M}$, which corresponds to the fact that in $\mathcal{M}$ the position $(m_{i'}, q_{i'}, n^{i'}, p^{i'}_{j'})$ was just played. Let us redefine $\mathscr{E}_i$ by means of the declaration

$$\mathscr{E}_i := \mathscr{E}_i \rhd [{}^{(\mathscr{F}^{i'}(\xi^{i'}))\overline{p}^{i'}_{j'},\varkappa_{i'}.\pi_{i'}}/_{K_{i'j'},\pi_{i'j'}}]^{P_{i'j'}} \langle\rangle$$

Here, it is t that chooses a reached position and proposes a possible completion for it. This will be reflected in the thread we have just added to the thread scheme of $\mathcal{M}$. Indeed, this thread ends in a process, namely $R^{i'}_{j'}$, which determines the following discussion:

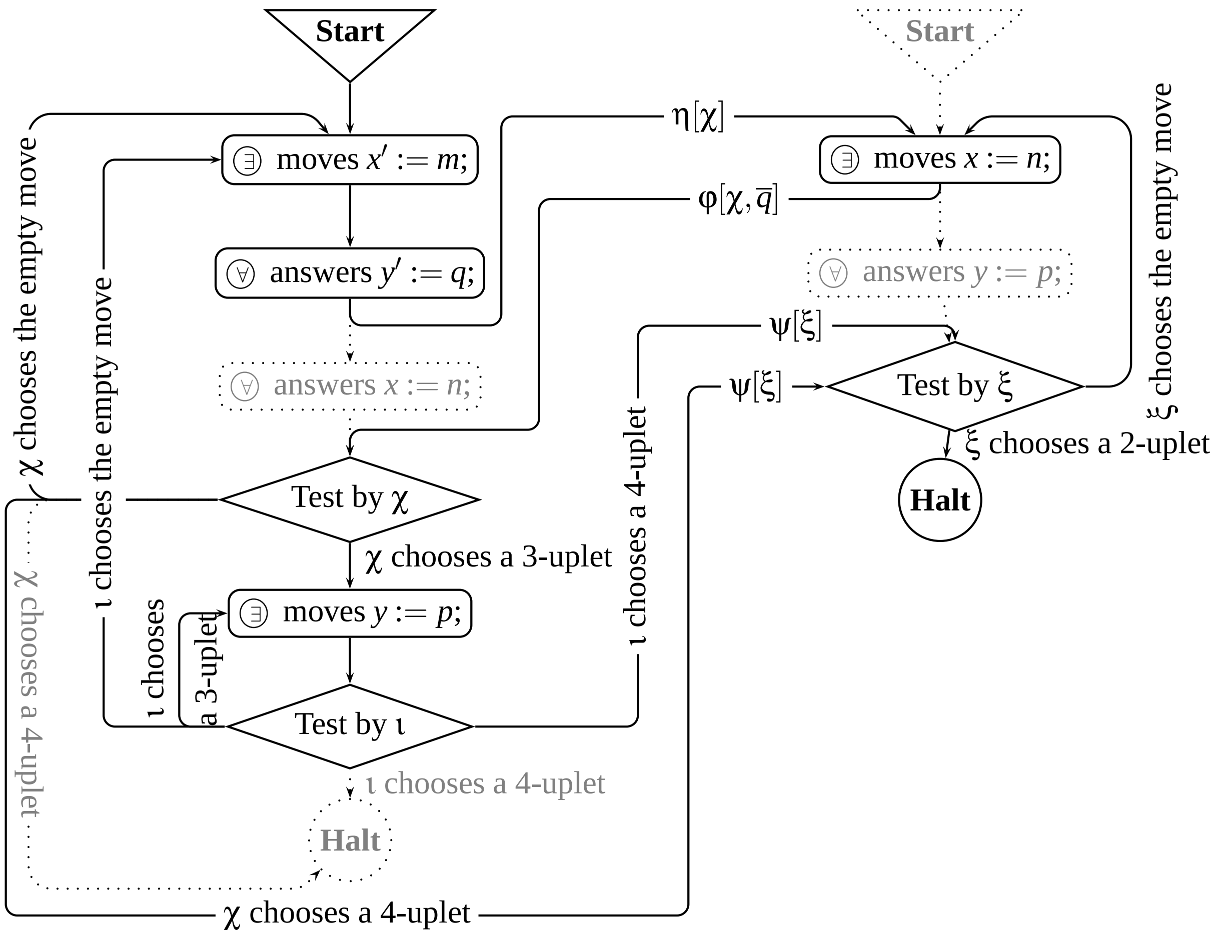
(i) $R^{i'}_{j'} = H^0_{\mu_{i+1}\xi_{i+1}} \star \overline{m}_{i+1}.\pi^0_{\mu_{i+1}\xi_{i+1}}$. The discussion continues exactly as in (a). The thread is finished and we start a new thread $i+1$ and a new play $\mathcal{N}^{i+1}$.

(ii) $R^{i'}_{j'} = K_{i''j''} \star \pi_{i''j''}$. the discussion continues as in (b). More precisely, this is the case when t chooses a complete position played before and gives the execution to u. If the last two coordinates of this complete position are “approved” by the exception handler of its corresponding play $\mathcal{N}^{i''}$, the game stops. If not, the game $\mathcal{M}$ is partially erased. It is the only one case in our discussion where the exception handlers built from u are used in $\mathcal{N}^i$.

(iii) $R^{i'}_{j'} = H^{i''}_{j''} \star \overline{p}^{i''}_{j''}.\pi^{i''}_{j''}$. This is the case where t takes a (first level) position $(m_{i''}, q_{i''}, n^{i''})$ played before and complete it proposing a new integer $p^{i''}_{j''}$. We have an analogous to the one of (c) we are analysing here.

We combine the flux diagram describing the winning strategies of Ψ (c.f.: 6.16.20) and Φ (c.f.: 6.22.24) to obtain the diagram 6.29.37, which describes “communication” between strategies u and t communicates to give the strategy σ. Subterms η, φ and ψ, which are the ”bricks” constituting σ, can be seen as the terms doing this communication.

(6.29.37)



Start
Start
η[χ]
∃ moves x′ := m;
∃ moves x := n;
φ[χ, q̄]
∀ answers y′ := q;
∀ answers y := p;
ψ[ξ]
∀ answers x := n;
ψ[ξ]
Test by ξ
ξ chooses the empty move
ξ chooses a 2-uplet
Halt
χ chooses the empty move
ι chooses the empty move
Test by χ
χ chooses a 3-uplet
ι chooses a 4-uplet
∃ moves y := p;
ι chooses a 3-uplet
Test by ι
ι chooses a 4-uplet
χ chooses a 4-uplet
Halt
χ chooses a 4-uplet

References

[1] Jhon L. Bell. Boolean-Valued Models and Independence Proofs. Oxford University Press.

[2] Paul J. Cohen. The Independence of the Continuum Hypothesis. Proceedings of the National Academy of Sciences of the United States of America, Vol. 50, No. 6. (Dec. 15, 1963), pp. 1143-1148.

[3] T. Griffin. A formulæ-as-type notion of control. In conference Record of the 17th A.C.M. Symposium on Principles of Programming Languages(1990).

[4] J.L.Krivine. A general storage theorem for integers in call-by-name lambda-calculus. Th. Comp. Sc., 129, p. 79-94 (1994).

[5] J.L.Krivine. Typed Lambda-Calculus in classical Zermelo-Frenkel set theory. Arch. Math. Log.40, 3, p.189-205 (2001).

[6] J.L.Krivine. Dependent choice 'quote' and the clock. Th. Comp. Sc., 308, p.259-276 (2003).

[7] J.L.Krivine. Realisability in classical logic. Lecture notes of the "École doctorale" of the Marseille Lumny University. May '04.

[8] J.L.Krivine. Realisability: a machine for Analysis and set theory. Lecture notes of the Realisability course. Winter school in Geometry of Computation. February '06. (Marseille-France).

[9] J.L.Krivine & Yves Legrandgérard. Valid formulas, games and network protocols. Published in HAL: http://hal.archives-ouvertes.fr/hal-00166880/fr/ (11 pages).

[10] Philippe Hesse. Réalisabilité Classique et protocoles réseaux. Université Denis Diderot - Paris VII. Thèse doctorale. (in french).