Terminaison des systèmes de réécriture d'ordre supérieur basée sur la notion de clôture de calculabilité

Frédéric Blanqui

Habilitation à Diriger des Recherches

Université Paris Diderot (Paris 7)

Document 2 : monographie

Terminaison des systèmes de réécriture d'ordre supérieur basée sur la notion de clôture de calculabilité

Frédéric Blanqui (INRIA)

http://who.rocq.inria.fr/Frederic.Blanqui/

Rédaction : 28 mars 2012
Soutenance : 13 juillet 2012

Jury

Gilles Barthe (rapporteur)
Nachum Dershowitz
Gilles Dowek
Bernhard Gramlich (rapporteur)
Delia Kesner (rapporteur)
Claude Kirchner
Dale Miller (président)

Table des matières

1 Introduction **3**

2 Types, termes et réécriture **5**
2.1 Types et pré-termes 5
2.2 Termes et réécriture 6
2.3 Relations bien fondées 7

3 Calculabilité **9**

4 Clôture de calculabilité **12**
4.1 Variables liées 16
4.2 Sous-termes 17
4.3 Filtrage sur des symboles définis 22
4.4 Réécriture modulo une théorie équationnelle 24
4.5 Filtrage d'ordre supérieur 29
4.6 Ordre récursif sur les chemins 36
4.7 Paires de dépendance 38
4.8 Annotations de taille 40
4.9 Réécriture conditionnelle 44
4.10 Étiquetage sémantique 45

Références **47**

Cette monographie fournit une présentation synthétique d'une partie de mes travaux de recherche sur le thème de la terminaison des systèmes de réécriture d'ordre supérieur, en mettant en évidence la notion qui y est commune, celle de clôture de calculabilité. Pour une description synthétique de mes autres travaux, je renvoie le lecteur à [Bla10a].

1 Introduction

Le λ-calcul est un langage Turing-complet permettant d'exprimer intentionnellement, de manière constructive, la notion de fonction : formation d'une fonction (abstraction) et application d'une fonction (substitution des arguments formels par les arguments réels). La réécriture (du premier ordre) est un langage également Turing-complet permettant d'exprimer des transformations syntaxiques basées sur la notion de motif. Dans le λ-calcul, il n'y a qu'une seule transformation possible (l'application d'une fonction à un argument) et celle-ci est inconditionnelle : une fonction peut être appliquée à n'importe quel terme, quelque soit sa structure. Il n'en va pas de même en réécriture : une transformation basée sur un certain motif ne peut s'appliquer à une expression que si celle-ci a la même structure syntaxique que le motif (modulo certaines théories équationnelles éventuellement). Par contre, la réécriture (du premier ordre) ne permet pas d'exprimer simplement des fonctions de manière dynamique et des motifs faisant appel à des variables liées (filtrage d'ordre supérieur) [Hue76, Mil91, Sti09]. Voir par exemple les travaux sur la logique combinatoire [CF58], la définition de la substitution [ACCL91] sur les termes de de Bruijn [dB72] ou l'encodage au premier ordre de systèmes de réécriture d'ordre supérieur [BKR05].

La réécriture d'ordre supérieur a pour but d'unifier ces deux langages. Plusieurs approches existent : les systèmes de réduction combinatoires (CRS) de Klop [Klo80, KvOvR93], les systèmes de réduction d'expressions (ERS) de Khasidashvili [Kha90, GKK05], ou les systèmes de réécriture d'ordre supérieur (HRS) de Nipkow [Nip91, MN98]. Van Oostrom et van Raamsdonk ont étudié les relations entre CRS et HRS [vOvR93] et ont développé un cadre très général, les HORS, qui inclue la plupart des calculs précédents [vO94, vR96]. Par ailleurs, des chercheurs se sont intéressés à des calculs où les motifs sont des objets du premier ordre : le λ-calcul avec motifs de van Oostrom [vO90, KvOdV08], le calcul de réécriture ou ρ-calcul de Cirstea et Kirchner [CK01], le calcul de motifs de Jay and Kesner [Jay04, JK09], ou certaines extensions de ML ou Haskell [Erw96, Tul10]. Dans mon travail, je considère des CRS (typés) et un peu les HRS aussi.

Pour montrer la terminaison de systèmes de réécriture d'ordre supérieur, on peut chercher à adapter à la réécriture des techniques développées pour le λ-calcul et, vice versa, étendre au λ-calcul des techniques développées pour la réécriture du premier ordre. Dans mes travaux, j'utilise l'une et l'autre approche.

Le λ-calcul pur ne terminant pas, c'est généralement un sous-ensemble de celui-ci que l'on considère : les λ-termes typables dans un système de types donné. Les types ont également d'autres utilités : garantir l'exécution normale

d'un programme ou, via l'isomorphisme de Curry-de Bruijn-Howard, représenter des propositions logiques.

Pour montrer la terminaison d'un λ-calcul typé, il n'y a à ma connaissance que trois techniques essentiellement :

1. Dans le cas du λ-calcul simplement typé, une preuve par induction sur le type de la variable substituée peut être donnée [San67, vD80]. Mais cette technique ne s'étend ni au polymorphisme, ni à la réécriture d'ordre supérieur, où les types des variables d'un motif peuvent être plus grands que le type du motif.
2. Dans le cas du λ-calcul simplement typé encore, les λI-termes où toute abstraction λxt vérifie $x \in \mathrm{FV}(t)$, peuvent être interprétés par des fonctions héréditairement monotones sur $\mathbb{N}$ [Gan80]. Or, il existe une transformation des λ-termes en λI-termes pour laquelle une β-réduction sur les λ-termes se traduit par une décroissance stricte dans $\mathbb{N}$. Cette technique a été étendue à la réécriture d'ordre supérieure simplement typée par van de Pol [vdP96]. Par ailleurs, Hamana a développé une sémantique catégorique des termes avec lieurs [Ham06] basée sur les travaux de Fiore, Plotkin et Turi [FPT99] qui est complète pour la terminaison, contrairement à celle de Van de Pol, et étendu ainsi à l'ordre supérieur la technique de l'étiquetage sémantique introduite au premier ordre par Zantema [Zan95, Ham07]. Cependant, Roux a montré dans sa thèse que son application à la β-réduction elle-même n'est pas immédiate car la version étiquetée de la β-réduction n'est pas la β-réduction [Rou11].
3. Une autre technique fait appel à la notion de "calculabilité"[1] [2] introduite par Tait [Tai67] pour le λ-calcul simplement typé, et étendue au polymorphisme par Girard [Gir71]. Van de Pol a étudié dans sa thèse les liens précis qu'il y a entre (2) et (3) : (2) peut être vu comme une réalisation/extraction d'une généralisation de (3) avec des informations sur la longueur des réductions.

Cette notion de calculabilité est au coeur de mes travaux. Je vais montrer dans cette monographie qu'elle peut être aisément adaptée à différentes formes de réécriture, et comment elle peut être mise en relation avec d'autres techniques : la notion d'ordre récursif sur les chemins [Der82, JR99] ou la notion de paire de dépendance [AG00]. Je montrerais également comment la terminaison basée sur les annotations de taille, qui est une extension naturelle de la calculabilité en présence de types inductifs [Gim96, Abe04], peut être étendue à la réécriture et vue comme un étiquetage sémantique [Zan95, Ham07].

1. À distinguer de la notion de calculabilité de Church et Turing. Ceci dit, étant donné un λ-terme $t : U \Rightarrow V$ calculable au sens de Tait-Girard, la fonction qui à un λ-terme calculable $u : U$ associe la forme normale de tu est bien une fonction calculable au sens de Church-Turing.
2. À vrai dire, Tait parle de "convertibilité" et Girard de "réductibilité". Le terme de "calculabilité" est semble-t-il dû à Troelstra [Tro73] bien que Troelstra lui-même invoque Tait.

2 Types, termes et réécriture

Par simplicité, nous nous limiterons dans cette monographie aux termes simplement typés. Les notions développées ici s'étendent sans trop de difficultés aux types polymorphes et dépendants, et à la réécriture au niveau des types qui généralise l'élimination forte et les types inductifs-récursifs [Dyb00]. Cependant, des restrictions sur les variables de type dans les règles de réécriture sont alors nécessaires [Ste98, WC03]. Nous renvoyons le lecteur vers [Bla05b, Bla05c] pour plus de détails.

Commençons maintenant par rappeler quelques définitions et notations sur les termes et la réécriture [Bar84, DJ90, Bar92, TeR03].

2.1 Types et pré-termes

Nous considérons un ensemble $\mathcal{B}$ non vide de *constantes de type*. L'ensemble $\mathcal{T}$ des *types* est alors défini récursivement ainsi :

– une constante de type $B \in \mathcal{B}$ est un type ;
– si T et U sont des types, alors $T \Rightarrow U$ est un type.

Soit $\mathcal{X}$ un ensemble de *variables* et $\mathcal{F}$ un ensemble de *symboles de fonction* disjoint de $\mathcal{X}$. L'ensemble $\mathcal{P}$ des *pré-termes* est défini ainsi :

– une variable ou un symbole de fonction est un pré-terme ;
– si x est une variable et t est un pré-terme, alors λxt est un pré-terme ;
– si t et u sont des pré-termes, alors tu est un pré-terme.

Étant donné un ensemble X, soit X^* le monoïde libre engendré par X, *i.e.* l'ensemble des séquences finies d'éléments de X ou *mots* sur X. Nous noterons le mot vide par ε, la concaténation par juxtaposition, et la longueur d'un mot w par $|w|$. Nous notons aussi parfois un mot $x_1, \ldots, x_n$ par $\vec{x}$. La relation préfixe $x \leq y$ s'il existe z tel que $y = xz$ est un ordre. Nous notons par $x\#y$ si x et y ne sont pas comparables (l'ordre préfixe n'est pas total).

L'ensemble des *positions* d'un pré-terme t, $\mathrm{Pos}(t)$, est le sous-ensemble de $\{0,1\}^*$ tel que :

– $\mathrm{Pos}(x) = \mathrm{Pos}(\mathsf{f}) = \{\varepsilon\}$ si $x \in \mathcal{X}$ et $\mathsf{f} \in \mathcal{F}$
– $\mathrm{Pos}(tu) = \{\varepsilon\} \cup \{0w \mid w \in \mathrm{Pos}(t)\} \cup \{1w \mid w \in \mathrm{Pos}(u)\}$
– $\mathrm{Pos}(\lambda xt) = \{\varepsilon\} \cup \{0w \mid w \in \mathrm{Pos}(t)\}$

Étant donné un pré-terme t, nous notons par $t|_p$ son sous-pré-terme à la position $p \in \mathrm{Pos}(t)$, et par $t[u]_p$ son remplacement par un pré-terme u.

Soit $\mathrm{FV}(t)$ l'ensemble des variables ayant une occurrence libre dans t (*i.e.* non liée par un λ). Un pré-terme t est *linéaire* si aucune variable n'a plus d'une occurrence libre dans t.

Un *contexte* c est un pré-terme avec une unique occurrence libre de $[]$. Étant donné un contexte c et un pré-terme t, $c[t]$ est le pré-terme obtenu en remplaçant $[]$ par t dans c. L'ordre *sous-pré-terme*, $\trianglelefteq$, est défini ainsi : $t \trianglelefteq u$ s'il existe un contexte c tel que $u = c[t]$. Nous noterons par $\triangleleft$ la partie stricte de $\trianglelefteq$. Un pré-terme est *algébrique* s'il ne contient aucun sous-pré-terme de la forme λxt ou xt.

2.2 Termes et réécriture

Nous supposons que chaque $a \in \mathcal{X} \cup \mathcal{F}$ est équipé d'un type τ_a. L'ensemble $\mathcal{L}$ des *termes* est le quotient de $\mathcal{P}$ par la relation d'*α-équivalence* identifiant deux termes modulo renommage de leurs variables liées de même type (pour chaque type T, nous supposons l'ensemble des variables de type T infini). La notion de position s'étend aux termes sans difficulté. Il n'en va pas de même pour la notion de contexte, sauf dans le cas où aucun λ ne se trouve au dessus de $[]$.

Le sous-ensemble des *termes* de *type* $T \in \mathcal{T}$, $\mathcal{L}(T)$, est défini ainsi :

- $a \in \mathcal{X} \cup \mathcal{F}$ est un terme de type τ_a ;
- si $x \in \mathcal{X}$ et t est un terme de type T, alors $\lambda x t$ est un terme de type $\tau_x \Rightarrow T$;
- si t est un terme de type $U \Rightarrow V$ et u un terme de type U, alors tu est un terme de type V.

Nous écrivons $t : T$ si $t \in \mathcal{L}(T)$.

Une *substitution* θ est une fonction de $\mathcal{X}$ dans $\mathcal{L}$ respectant les types (pour tout $x \in \mathcal{X}$, $\theta(x)$ est un terme de type τ_x) et dont le support ou *domaine* $\mathrm{dom}(\theta) = \{x \in \mathcal{X} \mid \theta(x) \neq x\}$ est fini. Soit alors $\mathrm{FV}(\theta) = \bigcup\{\mathrm{FV}(\theta(x)) \mid x \in \mathrm{dom}(\theta)\}$. Étant donné un terme t et une substitution θ, nous notons par $t\theta$ le terme obtenu en remplaçant dans t chaque occurrence libre d'une variable x par $\theta(x)$ en renommant, si nécessaire, les variables liées dans t de façon à ne pas lier les variables libres de $\theta(x)$. En particulier, nous notons par $\binom{u}{x}$ la substitution θ telle que $x\theta = u$ et $y\theta = y$ si $y \neq x$.

Une relation $\to$ sur les termes est *monotone* si, pour tous termes t, t' tels que $t \to t'$, nous avons $\lambda x t \to \lambda x t'$ et, pour tout terme u, nous avons $tu \to t'u$ et $ut \to ut'$. Une relation $\to$ sur $\mathcal{L}$ est *stable par substitution* si, pour tous termes t, t' tels que $t \to t'$, et toute substitution θ, nous avons $t\theta \to t'\theta$.

L'ordre sous-pré-terme ne se généralise pas bien aux termes (quels sont les *termes* apparaissant dans le terme $\lambda x x =_\alpha \lambda y y$?). À la place, nous considérons l'ordre *sous-terme stable* défini de la manière suivante :

- $t \trianglelefteq_s u$ si $t \lhd_s u$ ou $t = u$
- $v \lhd_s tu$ si $v \trianglelefteq_s t$ ou $v \trianglelefteq_s u$
- $v \lhd_s \lambda x t$ si $v \trianglelefteq_s t$ et $x \notin \mathrm{FV}(v)$

L'ordre sous-terme stable est stable par substitution et préserve les variables libres : si $t \trianglelefteq_s u$ alors $\mathrm{FV}(t) \subseteq \mathrm{FV}(u)$.

La *β-réduction* $\to_\beta$ est la clôture monotone de $\{(\lambda x t)u, t^u_x) \mid t, u \in \mathcal{T}, x \in \mathcal{X}\}$. La *$\eta$-réduction* $\to_\eta$ est la clôture monotone de $\{\lambda x(tx), t) \mid t \in \mathcal{T}, x \in \mathcal{X}, x \notin \mathrm{FV}(t)\}$.

Un terme $t : T$ est en *forme β-normale η-longue* si :

- $T \in \mathcal{B}$, $t = h\vec{t}$, $h \in \mathcal{F} \cup \mathcal{X}$ et chaque t_i est en forme β-normale η-longue ;
- ou $T = U \Rightarrow V$, $t = \lambda x v$ et v est en forme β-normale η-longue.

Une *règle de réécriture* (resp. *équation*) est une paire (l, r) de termes de même type notée $l \to r$ (resp. $l = r$). La *relation de réécriture* engendrée par un ensemble de règles $\mathcal{R}$, $\to_\mathcal{R}$, est la clôture monotone et par substitution de $\mathcal{R}$. Étant donné un ensemble de règles $\mathcal{R}$, soit $\mathcal{D}(\mathcal{R}) = \{\mathsf{f} \in \mathcal{F} \mid \exists \vec{l}, \exists r, \mathsf{f}\vec{l} \to r \in \mathcal{R}\}$ le sous-ensemble des symboles *définis* par $\mathcal{R}$.

2.3 Relations bien fondées

Un élément t est *fortement normalisant* pour une relation R s'il n'existe aucune chaîne infinie $t = t_0 R t_1 R \ldots$ Soit $\mathcal{SN}(R)$ l'ensemble des termes fortement normalisants pour R. La relation R *termine*, est *noethérienne* ou *bien fondée*[3] si tout terme est fortement normalisant.

Dans notre travail, nous nous intéressons à la terminaison de la relation $\to_\beta \cup \to_\mathcal{R}$ ou des variantes de celle-ci (on pourrait également considérer $\to_\beta \cup \to_\eta \cup \to_\mathcal{R}$). À noter que $\to_\beta$ et $\to_{\beta\eta} = \to_\beta \cup \to_\eta$ terminent. Cependant, la terminaison n'étant pas une propriété modulaire (déjà au premier ordre) [Toy87], la terminaison de $\to_\mathcal{R}$ ne suffit généralement pas à garantir la terminaison de $\to_\beta \cup \to_\mathcal{R}$. De plus, tel que $\to_\mathcal{R}$ est défini ici, considérer $\to_\mathcal{R}$ seul ne fait pas de sens quand, dans un membre droit de règle, une variable est appliquée à un terme. Ce n'est pas le cas avec les CRS et les HRS qui, dans la définition de $\to_\mathcal{R}$, incluent un certain nombre de β-réductions après application d'une règle.

Étant donnée que nous faisons une utilisation importante de l'induction bien fondée, nous rappelons ici des définitions, notations et résultats de base concernant les relations et pré-ordres bien fondées.

Étant donnée une relation R sur un ensemble X, nous notons par $R(t)$ l'ensemble $\{u \in X \mid tRu\}$ des *successeurs* de t, $R^=$ sa clôture réflexive, R^+ sa clôture transitive, R^* sa clôture réflexive et transitive, et R^{-1} son inverse. Cependant, nous noterons par $\leftarrow_\beta$, $\leftarrow_\eta$ et $\leftarrow_\mathcal{R}$ les inverses de $\to_\beta$, $\to_\eta$ et $\to_\mathcal{R}$ respectivement. Un élément t tel que $R(t) = \emptyset$ est dit en *forme normale* ou irréductible. Étant données deux relations R et S, nous notons leur composition par juxtaposition. Une relation R est *confluente* si $(R^{-1})^* R^* \subseteq R^* (R^{-1})^*$. C'est le cas de $\to_\beta$, $\to_\eta$ et $\to_{\beta\eta}$. Si R termine (resp. conflue) alors tout élément admet au plus (resp. au moins) une forme normale. Ainsi, tout terme t admet une unique forme normale pour $\to_\beta$, $\to_\eta$ ou $\to_{\beta\eta}$, notée $t\downarrow_\beta$, $t\downarrow_\eta$ et $t\downarrow_{\beta\eta}$ respectivement.

Étant donnée une relation d'équivalence R sur un ensemble X, nous notons par $[t]_R$ la classe d'équivalence d'un élément t, et par X/R l'ensemble des classes d'équivalence modulo R.

Étant donné un pré-ordre $\geq$ sur un ensemble X (relation réflexive et transitive), soit $\simeq \; = \; \geq \cap \leq$ son *équivalence associée* et $> \; = \; \geq - \leq$ sa *partie stricte* (relation irréflexive et transitive). Par abus de langage, nous disons qu'un pré-ordre $\geq$ est bien fondé si sa partie stricte est bien fondée.

Étant données des relations $R_1, \ldots, R_n$ sur des ensembles $X_1, \ldots, X_n$, la *relation produit* sur $X_1 \times \ldots \times X_n$ est $\vec{x}(R_1, \ldots, R_n)_{\text{prod}}\vec{x'}$ si, pour tout $i \in [1, n]$, $x_i R_i x'_i$. Si les R_i sont des pré-ordres, alors $(R_1, \ldots, R_n)_{\text{prod}}$ est un pré-ordre. Si, de plus, les parties strictes de $R_1, \ldots, R_n$ sont bien fondées, alors la partie stricte de $(R_1, \ldots, R_n)_{\text{prod}}$ est également bien fondée.

Étant données des pré-ordres $\geq_1, \ldots, \geq_n$ sur des ensembles $X_1, \ldots, X_n$, le *pré-ordre lexicographique* sur $X_1 \times \ldots \times X_n$ est l'union des relations suivantes :

3. Contrairement à la tradition mathématique où une relation R est dite *bien fondée* s'il n'existe pas de chaîne infinie descendante $t_0 R^{-1} t_1 R^{-1} \ldots$

– $\simeq_{\text{lex}} = (\simeq_1, \ldots, \simeq_n)_{\text{prod}}$
– $\vec{x} >_{\text{lex}} \vec{x'}$ s'il existe $i \in [1, n]$ tel que $x_i >_i x'_i$ et, pour tout $j < i$, $x_i \simeq_i x'_i$.

Si $>_1, \ldots, >_n$ sont bien fondées, alors $>_{\text{lex}}$ est également bien fondée.

Étant donné un pré-ordre $\geq$ sur un ensemble X, nous noterons également par $\geq_{\text{lex}}$ (resp. $\geq_{\text{prod}}$) le pré-ordre lexicographique (resp. produit) sur X^n avec chaque composante ordonnée par $\geq$.

Étant donnés deux ensembles X et Y et, pour chaque $x \in X$, un ensemble $Y_x \subseteq Y$, le *produit dépendant* de X et $(Y_x)_{x \in X}$ est l'ensemble $\Sigma_{x \in X} Y_x$ des paires $(x, y) \in X \times Y$ telles que $y \in Y_x$. Dans la suite, nous allons utiliser une généralisation du pré-ordre lexicographique au produit dépendant :

Definition 1 (Pré-ordre lexicographique dépendant) Soient $\Sigma_{x \in X} Y_x$ un produit dépendant, $\geq$ un pré-ordre sur X et, pour chaque classe d'équivalence E modulo $\simeq$, un ensemble Z_E muni d'un pré-ordre $\geq_E$. Soit enfin, pour chaque $x \in X$, une fonction $\psi_x : Y_x \to Z_{[x]}$. Le pré-ordre lexicographique dépendant sur $\Sigma_{x \in X} Y_x$ associé à $\geq$, $(\psi_x)_{x \in X}$ et $(\geq_E)_{E \in X/\simeq}$ est l'union des relations suivantes :
– $(x, y) \simeq_{\text{lex}} (x', y')$ si $x \simeq x' \wedge \psi_x(y) \simeq_{[x]} \psi_{x'}(y')$,
– $(x, y) >_{\text{lex}} (x', y')$ si $x > x' \vee (x \simeq x' \wedge \psi_x(y) >_{[x]} \psi_{x'}(y'))$.

Si $>$ et chaque $>_E$ sont bien fondées, alors $>_{\text{lex}}$ est bien fondée.

Étant donné un ensemble X, soit $\mathcal{M} = \mathbb{M}(X)$ l'ensemble des *multi-ensembles finis* sur X, *i.e.* des fonctions de X dans $\mathbb{N}$ à support fini. Étant donné un pré-ordre $\geq_X$ sur X, l'*extension* de $\geq_X$ aux multi-ensembles est le plus petit pré-ordre $\geq_{\mathcal{M}}$ contenant $>^1_{\mathcal{M}} \cup \simeq_{\mathcal{M}}$ où $>^1_{\mathcal{M}}$ et $\simeq_{\mathcal{M}}$ sont définies ainsi [CJ03] :
– $M + \{x\} >^1_{\mathcal{M}} M + \{y_1, \ldots, y_n\}$ $(n \geq 0)$ si, pour tout $i \in [0, n]$, $x >_X y_i$
– $\emptyset \simeq_{\mathcal{M}} \emptyset$
– $M + \{x\} \simeq_{\mathcal{M}} N + \{y\}$ si $M \simeq_{\mathcal{M}} N$ et $x \simeq_X y$

Son équivalence associée est $\simeq_{\mathcal{M}}$. Sa partie stricte $>_{\mathcal{M}} = (>^1_{\mathcal{M}})^+ \simeq_{\mathcal{M}}$ est bien fondée si $>_X$ est bien fondée.

Enfin, nous noterons par $\geq_{\text{mul}}$ le pré-ordre sur X^n tel que $\vec{x} \geq_{\text{mul}} \vec{y}$ si $\{\vec{x}\} \geq_{\mathcal{M}} \{\vec{y}\}$.

3 Calculabilité

La méthode de Tait-Girard [Tai75, GLT88] consiste à :

1. définir un sous-ensemble **Cand** de $\mathcal{P}(\mathcal{SN})$ dans lequel interpréter les types (un terme appartenant à un candidat $P \in \mathbf{Cand}$ est dit *calculable*) ;
2. interpréter chaque type T par un candidat $[\![T]\!] \in \mathbf{Cand}$;
3. montrer ensuite que tout terme de type T appartient à $[\![T]\!]$, d'où l'on peut déduire que tout terme typé est fortement normalisable.

Il va de soit que, pour pouvoir montrer (3), **Cand** ne peut pas être quelconque. Ainsi, il est toujours requis que :

- les variables sont calculables : pour tout $P \in \mathbf{Cand}$, $\mathcal{X} \subseteq P$;
- **Cand** est stable par la fonction $a : \mathcal{P}(\mathcal{L}) \times \mathcal{P}(\mathcal{L}) \to \mathcal{P}(\mathcal{L})$ définie par :

$$a(P, Q) = \{v \in \mathcal{L} \mid \forall t \in P, vt \in Q\}$$

  i.e. si $P, Q \in \mathbf{Cand}$, alors $a(P, Q) \in \mathbf{Cand}$;
- **Cand** est stable par intersection arbitraire [4] non vide :
  si $(A_i)_{i \in I}$ est une famille non vide de candidats, alors $\bigcap_{i \in I} A_i \in \mathbf{Cand}$;
- **Cand** contient $\mathcal{SN}$.

Les deux dernières conditions font que **Cand** doit être un treillis complet pour l'inclusion, la borne inférieure d'un sous-ensemble de **Cand** étant donnée par l'intersection (en prenant $\mathcal{SN}$ dans le cas de l'ensemble vide). Par contre, la borne supérieure d'un sous-ensemble n'est pas nécessairement donnée par l'union.

L'intersection sert à interpréter la quantification sur les types tandis que l'opération a sert à interpréter $\Rightarrow$ et ainsi avoir, par définition, vt calculable si v et t sont calculables, ce qui est le noeud du problème dans la preuve de terminaison de la β-réduction.

Plusieurs ensembles de candidats ont été proposé dans la littérature :

- **Red** : l'ensemble des **candidats de réductibilité** de Girard [Gir72, GLT88]. Un ensemble $P \subseteq \mathcal{SN}$ appartient à **Red** si les conditions suivantes sont satisfaites :

(R1) $\mathcal{X} \subseteq P$;

(R2) P est stable par réduction : si $t \in P$ et $t \to u$, alors $u \in P$;

(R3) si t est un terme *neutre* [5] et $\to(t) \subseteq P$, alors $t \in P$.

Dans le λ-calcul pur, un terme est neutre s'il n'est pas une abstraction. Ainsi, les termes neutres vérifient la propriété fondamentale suivante : si t est neutre alors, pour tout terme u, tu est neutre et non réductible en tête.

4. Finie ou infinie.
5. Appelé "simple" dans [Gir72].

– **Sat** : l'ensemble des **ensembles saturés**[6] de Tait [Tai75]. Un ensemble $P \subseteq \mathcal{SN}$ appartient à **Sat** si les conditions suivantes sont satisfaites :

(S1) P contient tous les termes fortement normalisables de la forme $x\vec{t}$;

(S2) si $t^u_x\vec{v} \in P$ et $u \in \mathcal{SN}$, alors $(\lambda xt)u\vec{v} \in P$.

– **SatInd** : le plus petit sous-ensemble de **Sat** contenant $\mathcal{SN}$ et stable par a et $\bigcap$ [Par97]. Comme l'a justement remarqué Parigot, il n'est pas nécessaire de considérer tous les ensembles saturés mais seulement ceux engendrés à partir de $\mathcal{SN}$.

– **Bi** : l'ensemble des **bi-orthogonaux** de Parigot [Par97] est l'ensemble $\{a^*(E, \mathcal{SN}) \mid \emptyset \neq E \subseteq \mathcal{SN}^*\}$ où $\mathcal{SN}^*$ est l'ensemble des séquences finies d'éléments de $\mathcal{SN}$ et $a^* : \mathcal{P}(\mathcal{L}^*) \times \mathcal{P}(\mathcal{L}) \to \mathcal{P}(\mathcal{L})$ généralise a de la manière suivante :

$$a^*(E, Q) = \{v \in \mathcal{L} \mid \forall \vec{t} \in E, v\vec{t} \in Q\}$$

Une séquence $\vec{t} \in \mathcal{L}^*$ peut être vue comme un contexte $[]\vec{t}$. Ainsi,

$$a^*(E, Q) = \{v \in \mathcal{L} \mid \forall e \in E, e[v] \in Q\}$$

À vrai dire, Parigot n'utilise pas l'expression "bi-orthogonal" qui sous-entend une certaine relation d'orthogonalité entre $\mathcal{P}(\mathcal{SN})$ et $\mathcal{P}(\mathcal{SN}^*)$, à savoir $P \perp E$ ssi $\forall v \in P, \forall \vec{t} \in E, v\vec{t} \in \mathcal{SN}$. L'orthogonal (à droite) de $P \subseteq \mathcal{SN}$ est alors $P^\perp = \{\vec{t} \in \mathcal{SN}^* \mid \forall v \in P, v\vec{t} \in \mathcal{SN}\}$, tandis que l'orthogonal (à gauche) de $E \subseteq \mathcal{SN}^*$ est ${}^\perp E = a^*(E, \mathcal{SN})$. On voit alors que $\mathbf{Bi} = \{P \subseteq \mathcal{SN} \mid P \neq \emptyset \wedge {}^\perp(P^\perp) = P\}$.

Les candidats de réductibilité et les ensembles saturés sont étudiés dans [Gal90]. En particulier, tout candidat de réductibilité est un ensemble saturé : $\mathbf{Red} \subseteq \mathbf{Sat}$. L'inverse est faux car un ensemble saturé n'est pas nécessairement stable par réduction : par exemple, le plus petit ensemble saturé contenant $\lambda x(\lambda yy)x$ ne contient pas λxx. Cependant, Riba a montré dans sa thèse [Rib07] que tout ensemble saturé stable par réduction est un candidat de réductibilité. Ainsi, $\mathbf{Red} = \mathbf{Sat}_\to = \{P \in \mathbf{Sat} \mid \to(P) \subseteq P\}$. Par ailleurs, dans [Par97], Parigot montre que tout élément de **SatInd** est un bi-orthogonal : $\mathbf{SatInd} \subseteq \mathbf{Bi}$. Enfin, Riba a montré dans sa thèse que tout bi-orthogonal est un candidat : $\mathbf{Bi} \subseteq \mathbf{Red}$. Ainsi, les bi-orthogonaux sont stables par réduction. Par contre, je ne connais aucune preuve pour l'instant du fait que **SatInd**, **Bi** et **Red** soient distincts. En conclusion, nous avons les relations suivantes :

$$\mathbf{SatInd} \subseteq \mathbf{Bi} \subseteq \mathbf{Red} = \mathbf{Sat}_\to \subset \mathbf{Sat}$$

Une question naturelle est alors de savoir dans quelle mesure chacun de ses ensembles peut être étendu à la réécriture, et si un ensemble permet de montrer la terminaison de davantage de systèmes qu'un autre. Toutes les définitions s'appuient sur la forme des redex (expressions réductibles) : **Red** fait appel à la notion de terme neutre, un ensemble $P \in \mathbf{Sat}$ doit être stable par expansion

6. Cette expression semble être due à Gallier [Gal90].

de tête (relation inverse de la réduction de tête), et **Bi** est défini comme l'ensemble des bi-orthogonaux relativement à une relation entre termes et contextes permettant de construire un redex de tête.

- **Bi** étant exclusivement basé sur la notion de contexte, il n'est pas clair comment l'étendre à la réécriture.
- Les ensembles saturés pourraient quant à eux s'étendre en remplaçant (S2) par :

  (S2') si $l \to r \in \mathcal{R}$, $r\sigma\vec{t} \in P$ et $\sigma \in \mathcal{SN}_\mathcal{R}$, alors $l\sigma\vec{t} \in P$.

  Pour avoir $\mathcal{SN}_\mathcal{R} \in \mathbf{Sat}_\mathcal{R}$, il faut alors pouvoir montrer que $l\sigma\vec{t} \in \mathcal{SN}_\mathcal{R}$ si $r\sigma\vec{t} \in \mathcal{SN}_\mathcal{R}$ et $\sigma \in \mathcal{SN}_\mathcal{R}$, ce qui n'est généralement pas le cas si $\mathcal{R}$ n'est pas orthogonal, *i.e.* linéaire gauche et sans paire critique, une propriété garantissant également la confluence de $\mathcal{R}$. Ce problème pourrait peut-être être résolu en considérant *tous* les réduits de tête de $l\sigma$, mais on arrive alors à une condition semblable à (R3).
- Les candidats de réductibilité s'étendent quant à eux facilement à la réécriture en déclarant neutre tout terme de la forme $\mathsf{f}\vec{t}$ avec $\mathsf{f} \in \mathcal{D}(\mathcal{R})$ et $|\vec{t}| \geq \alpha_\mathsf{f} = sup\{|\vec{l}| \mid \mathsf{f}\vec{l} \to r \in \mathcal{R}\}$ supposé fini, ce qui est une hypothèse raisonnable toujours vérifiée si $\mathcal{R}$ est fini ou si les symboles f sont simplement typés. C'est cette notion de calculabilité que nous avons considérée dans nos travaux.

Il est également intéressant de chercher à étudier les propriétés de ces ensembles. Ainsi, dans sa thèse, Riba a étudié la stabilité par union des candidats. En observant que les candidats de réductibilité sont exactement les ensembles saturés stables par réduction, on déduit que **Red**, comme **Sat**, est stable par union arbitraire non vide, résultat obtenu indépendamment par Tatsuta [Tat07].

Alors que l'ensemble $\mathcal{N}$ des termes neutres n'est pas stable par réduction, l'ensemble $\mathcal{V}$ des termes non neutres ou *valeurs* est stable par réduction. Soit $\mathrm{Val}(t)$ l'ensemble des réduits non neutres ou valeurs de t : $\mathrm{Val}(t) = \{v \in \mathcal{V} \mid t \to^* v\}$, et $\leq$ l'ordre sur les termes obtenu en comparant leurs ensembles de valeurs : $t \leq u$ si $\mathrm{Val}(t) \subseteq \mathrm{Val}(u)$. Riba a montré que les candidats de réductibilité sont tous les sous-ensembles non vides de $\mathcal{SN}$ clos par le bas pour $\leq$: si $t \leq u \in P \in \mathbf{Red}$ alors $t \in P$. En effet, dans $\mathcal{SN}$, (R1)$\wedge$(R2)$\wedge$(R3) est équivalent à l'unique condition suivante :

$$\text{(R) } t \in P \text{ ssi } \mathrm{Val}(t) \subseteq P.$$

4 Clôture de calculabilité

[Bla05c] F. Blanqui. Definitions by rewriting in the calculus of constructions. *Mathematical Structures in Computer Science*, 15(1) :37–92, 2005.

Nous avons vu dans la section précédente qu'il existait différentes notions de calculabilité mais que celle due à Girard [Gir72, GLT88] semblait la mieux adaptée pour être étendue à la réécriture. C'est celle que nous allons considérer dans la suite.

Definition 2 (Calculabilité) Étant donné un ensemble $\mathcal{R}$ de règles de réécriture de la forme $\mathsf{f}\vec{l} \to r$, un terme est *neutre* s'il est de la forme $x\vec{v}$, $(\lambda xt)u\vec{v}$ ou $\mathsf{f}\vec{v}$ avec $\mathsf{f} \in \mathcal{D}(\mathcal{R})$ et $|\vec{v}| \geq \alpha_{\mathsf{f}} = sup\{|\vec{l}| \mid \mathsf{f}\vec{l} \to r \in \mathcal{R}\}$. Soit $\mathbf{Red}_{\mathcal{R}}$ l'ensemble des $P \subseteq \mathcal{L}$ tels que :

(R0) $P \subseteq \mathcal{SN}(\to)$ où $\to = \to_\beta \cup \to_{\mathcal{R}}$;
(R1) $\mathcal{X} \subseteq P$;
(R2) P est stable par réduction : si $t \in P$ et $t \to u$, alors $u \in P$;
(R3) si t est un terme neutre et $\to(t) \subseteq P$, alors $t \in P$.

Nous vérifions que $\mathbf{Red}_{\mathcal{R}}$ a les bonnes propriétés :

Lemma 3 $\mathbf{Red}_{\mathcal{R}}$ est stable par a et intersection non vide et admet $\mathcal{SN}(\to)$ comme plus grand élément.

Proof. La stabilité par intersection non vide et le fait que $\mathcal{SN}(\to) \in \mathbf{Red}_{\mathcal{R}}$ se montrent aisément. Nous détaillons la stabilité par a. Soient $P, Q \in \mathbf{Red}_{\mathcal{R}}$.

(R0) Soit $t \in a(P,Q)$. Par (R1), $X \subseteq P$. Soit donc $x \in X$. Par définition de a, $tx \in Q$. Par (R0), $tx \in \mathcal{SN}$. Donc, $t \in \mathcal{SN}$.
(R1) Soient $x \in X$ and $u \in P$. Nous montrons que $xu \in Q$ par induction bien fondée sur u avec $\to$ comme relation bien fondée ($u \in \mathcal{SN}$ par (R0)). Par hypothèse d'induction $\to(xu) \subseteq Q$. Comme xu est neutre, par (R3), $xu \in Q$.
(R2) Soient $t \in a(P,Q)$, t' tel que $t \to t'$, et $u \in P$. Par définition de a, $tu \in Q$. Par (R2), $t'u \in Q$.
(R3) Soient t neutre tel que $\to(t) \subseteq a(P,Q)$, et $u \in P$. Nous montrons que $tu \in Q$ par induction bien fondée sur u avec $\to$ comme relation bien fondée ($u \in \mathcal{SN}$ par (R0)). Comme tu est neutre, par (R3), il suffit de montrer que $\to(tu) \subseteq Q$. Soit $v \in \to(tu)$. Nous montrons maintenant (*) : soit $v = t'u$ avec $t \to t'$, soit $v = tu'$ avec $u \to u'$.
  – $tu \to_\beta v$. Comme t est neutre, t n'est pas une abstraction et (*) est vérifié.
  – $tu \to_{\mathcal{R}} v$. S'il existe $\mathsf{f}\vec{l}\vec{l} \to r \in \mathcal{R}$ et σ tels que $tu = \mathsf{f}\vec{l}\sigma l\sigma$ et $v = r\sigma$, alors $t = \mathsf{f}\vec{l}\sigma$ et $|\vec{l}| < |\vec{l}\vec{l}| \leq \alpha_{\mathsf{f}}$. Comme t est neutre, cela n'est pas possible. Donc (*) est vérifié.

Nous montrons maintenant que $v \in Q$.
– $v = t'u$ avec $t \to t'$. Par hypothèse, $t' \in a(P, Q)$. Comme $u \in P$, $v \in Q$.
– $v = tu'$ avec $u \to u'$. Par (R2), $u' \in P$. Donc, par induction, $v \in Q$. ■

Notons que seule (R3) dépend de la définition de la relation $\to$.

Étant donnée une *interprétation* $I : \mathcal{B} \to \mathbf{Red}_{\mathcal{R}}$ des constantes de type, l'interprétation d'un type T dans $\mathbf{Red}_{\mathcal{R}}$ est :
– $[\![\mathsf{B}]\!]^I = I(\mathsf{B})$ si $\mathsf{B} \in \mathcal{B}$,
– $[\![T \Rightarrow U]\!]^I = a([\![T]\!]^I, [\![U]\!]^I)$ où $a(P, Q) = \{v \in \mathcal{L} \mid \forall t \in P, vt \in Q\}$.
Un terme $t : T$ est *calculable* si $t \in [\![T]\!]^I$.

L'étape suivante consiste alors à définir une interprétation I pour les constantes de type et montrer que tous les symboles de fonction sont calculables avec I, c'est-à-dire, $\mathsf{f} \in [\![\tau_{\mathsf{f}}]\!]^I$ pour tout $\mathsf{f} \in \mathcal{F}$. Supposons que $\tau_{\mathsf{f}} = \vec{T} \Rightarrow U$ avec $|\vec{T}| = \alpha_{\mathsf{f}}$. Alors, $\mathsf{f} \in [\![\tau_{\mathsf{f}}]\!]^I$ ssi, pour tout $\vec{t} \in [\![\vec{T}]\!]$, $\mathsf{f}\vec{t}$ est calculable. Si de plus $\mathsf{f} \in \mathcal{D}(\mathcal{R})$, alors $\mathsf{f}\vec{t}$ est neutre et donc calculable si tous ses réduits le sont. La notion de clôture de calculabilité permet de garantir cette propriété.

Definition 4 (Clôture de calculabilité) Une *$\mathcal{F}$-famille* est une fonction qui, à chaque $\mathsf{f} \in \mathcal{F}$ et $\vec{t} \in \mathcal{L}^*$ tel que $\mathsf{f}\vec{t}$ soit bien typé, associe un ensemble $\mathrm{CC}_{\mathsf{f}}(\vec{t})$ de termes bien typés. Une $\mathcal{F}$-famille CC est *close par substitution* si, pour tout $u \in \mathrm{CC}_{\mathsf{f}}(\vec{t})$ et substitution σ, nous avons $u\sigma \in \mathrm{CC}_{\mathsf{f}}(\vec{t}\sigma)$. Une $\mathcal{F}$-famille CC est *close par calculabilité* si, pour tout $u \in \mathrm{CC}_{\mathsf{f}}(\vec{t})$, nous avons u calculable si $\vec{t}$ est calculable.

Theorem 5 La relation $\to_\beta \cup \to_{\mathcal{R}}$ termine s'il existe $I : \mathcal{B} \to \mathbf{Red}_{\mathcal{R}}$ et une $\mathcal{F}$-famille CC close par substitution et calculabilité telles que :
– tout symbole $\mathsf{f} \in \mathcal{F} - \mathcal{D}(\mathcal{R})$ est calculable ;
– pour toute règle $l \to r \in \mathcal{R}$, l est de la forme $\mathsf{f}\vec{l}$ et $r \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$.

Proof. Comme évoqué précédemment, il suffit de montrer que, pour tout $\mathsf{f} \in \mathcal{D}(\mathcal{R})$, $\mathsf{f} : \vec{T} \Rightarrow U$, $|\vec{T}| = \alpha_{\mathsf{f}}$ et $\vec{t} \in [\![\vec{T}]\!]^I$, tous les réduits t de $\mathsf{f}\vec{t}$ sont calculables. Pour cela, nous procédons par induction bien fondée sur $\vec{t}$ avec la partie stricte de $(\to^*)_{\mathrm{prod}}$ comme relation bien fondée ($\vec{t} \in \mathcal{SN}$), notée $\to_{\mathrm{prod}}$ dans la suite. Il y a alors deux cas possibles :
– Il existe $\vec{u}$ tel que $t = \mathsf{f}\vec{u}$ et $\vec{t} \to_{\mathrm{prod}} \vec{u}$. Par (R2), $\vec{u}$ est calculable. Donc, par hypothèse d'induction, $\mathsf{f}\vec{u}$ est calculable.
– Il existe $\vec{w}$, $\mathsf{f}\vec{l} \to r \in \mathcal{R}$ et σ tels que $\vec{t} = \vec{l}\sigma\vec{w}$ et $t = r\sigma\vec{w}$. Comme $r \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$ et CC est stable par substitution, nous avons $r\sigma \in \mathrm{CC}_{\mathsf{f}}(\vec{l}\sigma)$. Comme $\vec{l}\sigma$ est calculable et CC est clos par calculabilité, nous avons $r\sigma$ calculable. Enfin, comme $\vec{w}$ est calculable, nous avons t calculable. ■

Ainsi, la terminaison de $\to_\beta \cup \to_{\mathcal{R}}$ peut se ramener en grande partie à la recherche d'opérations préservant la calculabilité. Parmi celles-ci, on peut évidemment compter les opérations de la Figure 1.

FIGURE 1 – Opérations de clôture I

(arg)	$\{\vec{l}\} \subseteq \mathrm{CC}_{\mathsf{f}}(\vec{l})$
(app)	si $t \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$, $t : U \Rightarrow V$, $u \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$ et $u : U$, alors $tu \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$
(red)	si $t \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$ et $t \to u$, alors $u \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$

On pourrait ensuite autoriser des termes de la forme $\mathsf{g}\vec{m}$ si $(\mathsf{g}, \vec{m}) < (\mathsf{f}, \vec{l})$ avec $\geq$ un pré-ordre lexicographique dépendant bien fondé. Dans ce cas, que CC soit clos par calculabilité et que les symboles de fonction soient calculables (Théorème 5), doit se montrer en même temps, par induction sur $>$.

Considérer un pré-ordre plutôt qu'un ordre permet de traiter le cas de fonctions mutuellement définies. Par ailleurs, afin de pouvoir comparer deux paires $(\mathsf{f}, \vec{l})$ et $(\mathsf{g}, \vec{m})$ ayant un nombre distinct d'éléments (y compris quand $\mathsf{f} = \mathsf{g}$), nous utiliserons un *système de filtrage* des arguments [AG00] :

Definition 6 (Filtrage des arguments) Un *filtre* pour un symbole f est un mot $\varphi_{\mathsf{f}} = k_1 \dots k_n$ sur $[1, \alpha_{\mathsf{f}}]$. Étant donné un ensemble X, un mot $\vec{x}$ sur X est compatible avec φ_{f} si $|\vec{x}| \geq \max\{\varphi_{\mathsf{f}}\}$. Nous notons par φ_{f}^X la fonction qui à tout mot $\vec{x}$ compatible avec φ_{f} associe $x_{k_1} \dots x_{k_n} \in X^n$.

Definition 7 ($\mathcal{F}$-pré-ordre) Un *$\mathcal{F}$-pré-ordre* est un pré-ordre lexicographique dépendant sur $\Sigma = \Sigma_{\mathsf{f} \in \mathcal{F}} Y_{\mathsf{f}}$ où Y_{f} est l'ensemble des mots $\vec{t}$ compatibles avec φ_{f} tels que $\mathsf{f}\vec{t}$ est bien typé. Étant donné un pré-ordre $\geq_{\mathcal{F}}$ sur $\mathcal{F}$ et un système de filtrage φ compatible avec $\simeq_{\mathcal{F}}$, un *$(\mathcal{F}, \varphi)$-pré-ordre* est un pré-ordre lexicographique dépendant $\geq$ sur $\Sigma^\varphi = \Sigma_{\mathsf{f} \in \mathcal{F}} Y_{\mathsf{f}}^\varphi$ où, si $\mathsf{f} : \vec{T} \Rightarrow U$, $|\vec{T}| = \alpha_{\mathsf{f}}$ et $\varphi_{\mathsf{f}} = k_1 \dots k_p$, alors $Y_{\mathsf{f}}^\varphi = [\![T_{k_1}]\!]^I \times \dots \times [\![T_{k_p}]\!]^I$. Un $(\mathcal{F}, \varphi)$-pré-ordre $\geq$ est naturellement étendu en un $\mathcal{F}$-pré-ordre $\geq^\varphi$ de la façon suivante : $(\mathsf{f}, \vec{l}) \geq^\varphi (\mathsf{g}, \vec{m})$ si $(\mathsf{f}, \varphi_{\mathsf{f}}^{\mathcal{L}}(\vec{l})) \geq (\mathsf{g}, \varphi_{\mathsf{g}}^{\mathcal{L}}(\vec{m}))$.

Si $\geq$ est un $(\mathcal{F}, \varphi)$-pré-ordre bien fondé (resp. stable par substitution), alors $\geq^\varphi$ est un $\mathcal{F}$-pré-ordre bien fondé (resp. stable par substitution).

Lemma 8 La relation $\geq$ sur Σ telle que $(\mathsf{f}, \vec{t}) \geq (\mathsf{g}, \vec{u})$ si $\mathsf{f} = \mathsf{g}$ et $\vec{t} \to_{\mathrm{prod}} \vec{u}$ est un $\mathcal{F}$-pré-ordre bien fondé et stable par substitution que nous appellerons *$\mathcal{F}$-pré-ordre de réduction* et que nous noterons également par $\to_{\mathrm{prod}}$.

Avec un $(\mathcal{F}, \varphi)$-pré-ordre bien fondé, on peut ajouter à la clôture l'opération de la Figure 2.

Theorem 9 La relation $\to_\beta \cup \to_{\mathcal{R}}$ termine s'il existe $I : \mathcal{B} \to \mathbf{Red}_{\mathcal{R}}$ et un $(\mathcal{F}, \varphi)$-pré-ordre $\geq$ tels que $>$ est stable par substitution, $>^\varphi \cup \to_{\mathrm{prod}}$ est bien fondé et :

– tout symbole $\mathsf{f} \in \mathcal{F} - \mathcal{D}(\mathcal{R})$ est calculable ;

FIGURE 2 – Opérations de clôture II

(rec) si $(\mathsf{f}, \vec{l}) >^{\varphi} (\mathsf{g}, \vec{m})$ et $\vec{m} \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$, alors $\mathsf{g}\vec{m} \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$

– pour toute règle $l \to r \in \mathcal{R}$, l est de la forme $\mathsf{f}\vec{l}$ et $r \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$ où CC est la plus petite $\mathcal{F}$-famille close par les opérations I à II.

Proof. Comme dans le Théorème 5, il suffit de montrer que, pour tout $(\mathsf{f}, \vec{t}) \in \Sigma$ tel que $\mathsf{f} \in \mathcal{D}(\mathcal{R})$, nous avons $\mathsf{f}\vec{t}$ calculable. Pour cela, nous procédons par induction bien fondée sur $(\mathsf{f}, \vec{t})$ ordonné par $>^{\varphi} \cup \to_{\mathrm{prod}}$. Comme $\mathsf{f}\vec{t}$ est neutre, il suffit de montrer que tous ses réduits t sont calculables. Il y a deux cas possibles :

– Il existe $\vec{u}$ tel que $t = \mathsf{f}\vec{u}$ et $\vec{t} \to_{\mathrm{prod}} \vec{u}$. Par (R2), $\vec{u}$ est calculable. Donc, par hypothèse d'induction, $\mathsf{f}\vec{u}$ est calculable.
– Il existe $\vec{w}$, $\mathsf{f}\vec{l} \to r \in \mathcal{R}$ et σ tels que $\vec{t} = \vec{l}\sigma\vec{w}$ et $t = r\sigma\vec{w}$. Par hypothèse, $r \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$. Comme $>$ est stable par substitution, CC est stable par substitution et $r\sigma \in \mathrm{CC}_{\mathsf{f}}(\vec{l}\sigma)$. Il suffit alors de montrer que, pour tout $u \in \mathrm{CC}_{\mathsf{f}}(\vec{l}\sigma)$, u est calculable, par induction sur $\mathrm{CC}_{\mathsf{f}}(\vec{l}\sigma)$. Les cas (arg), (app) et (red) sont immédiats. Le cas (rec) est résolu par hypothèse d'induction. ■

Le $(\mathcal{F}, \varphi)$-pré-ordre le plus simple qu'on puisse imaginer est le produit lexicographique dépendant d'un pré-ordre $\geq_{\mathcal{F}}$ sur $\mathcal{F}$ et de l'extension lexicographique ou multi-ensemble de l'ordre sous-terme stable.

Lemma 10 Étant donné un pré-ordre bien fondé $\geq_{\mathcal{F}}$ sur $\mathcal{F}$, un système de filtrage φ et, pour chaque classe E, un *statut* $\mathrm{stat}_E \in \{\mathrm{lex}, \mathrm{mul}\}$ tels que :

– si $\mathrm{stat}_E = \mathrm{mul}$: il existe un type T_E tel que, pour tout $\mathsf{f} : \vec{T} \Rightarrow \mathsf{B}$ dans E de filtre $k_1 \dots k_n$, nous avons $T_{k_i} = T_E$ pour tout $i \in [1, n]$,
– si $\mathrm{stat}_E = \mathrm{lex}$: pour tous $\mathsf{f} : \vec{T} \Rightarrow \mathsf{B}$ et $\mathsf{g} : \vec{U} \Rightarrow \mathsf{C}$ dans E de filtre $k_1 \dots k_p$ et $l_1 \dots l_q$ respectivement, nous avons $p = q$ et $T_{k_i} = U_{l_i}$ pour tout $i \in [1, p]$,

alors le pré-ordre lexicographique dépendant $\geq$ défini par $\geq_{\mathcal{F}}$ et :

– si $\mathrm{stat}_E = \mathrm{mul}$: $Z_E = \mathbb{M}([\![T_E]\!]^I)$, $\psi_{\mathsf{f}}(\vec{t}) = \{\varphi^{\mathcal{L}}_{\mathsf{f}}(\vec{t})\}$ et $\geq_E = (\to^* \trianglerighteq_s)_{\mathrm{mul}}$,
– si $\mathrm{stat}_E = \mathrm{lex}$: $Z_E = Y_E^{\varphi}$, $\psi_{\mathsf{f}}(\vec{t}) = \varphi^{\mathcal{L}}_{\mathsf{f}}(\vec{t})$ et $\geq_E = (\to^* \trianglerighteq_s)_{\mathrm{lex}}$

est un $(\mathcal{F}, \varphi)$-pré-ordre tel que $>$ est stable par substitution et $>^{\varphi} \cup \to_{\mathrm{prod}}$ est bien fondée.

L'union avec $\to_{\mathrm{prod}}$ est bien fondée car $\rhd_s \to \subseteq \to \rhd_s$.

4.1 Variables liées

Dans les opérations préservant la calculabilité, on peut également autoriser l'abstraction et l'ensemble X des variables liées correspondantes en paramétrant CC par celles-ci (voir Figure 3).

FIGURE 3 – Opérations de clôture IV

(var)	si $x \in X - \mathrm{FV}(\vec{l})$, alors $x \in \mathrm{CC}_{\mathsf{f}}^{X}(\vec{l})$
(abs)	si $t \in \mathrm{CC}_{\mathsf{f}}^{X\cup\{x\}}(\vec{l})$ et $x \notin \mathrm{FV}(\vec{l})$, alors $\lambda x t \in \mathrm{CC}_{\mathsf{f}}^{X}(\vec{l})$

Mais il faut alors relativiser toutes les définitions par rapport à un tel ensemble de variables liées :

Definition 11 (Clôture avec variables liées) Une *$\mathcal{F}$-famille* est une fonction qui, à chaque $\mathsf{f} \in \mathcal{F}$, $\vec{t} \in \mathcal{L}^*$ tel que $\mathsf{f}\vec{t}$ soit bien typé, et ensemble fini $X \subseteq \mathcal{X}$ tel que $\mathrm{FV}(\vec{t}) \cap X = \emptyset$, associe un ensemble $\mathrm{CC}_{\mathsf{f}}^{X}(\vec{t}) \subseteq \mathcal{L}$. Une $\mathcal{F}$-famille CC est *close par substitution* si, pour tout $u \in \mathrm{CC}_{\mathsf{f}}^{X}(\vec{t})$ et σ telle que $\mathrm{dom}(\sigma) \subseteq \mathrm{FV}(\vec{t})$ et $\mathrm{FV}(\sigma) \cap X = \emptyset$, nous avons $u\sigma \in \mathrm{CC}_{\mathsf{f}}(\vec{t\sigma})$. Une $\mathcal{F}$-famille CC est *close par calculabilité* si, pour tout $u \in \mathrm{CC}_{\mathsf{f}}^{X}(\vec{t})$ et $\vec{x}$ tel que $X = \{\vec{x}\}$, nous avons $\lambda\vec{x}u$ calculable.

4.2 Sous-termes

[Bla05b] F. Blanqui. Inductive types in the calculus of algebraic constructions. *Fundamenta Informaticae*, 65(1-2) :61–86, 2005.

La clôture définie par les opérations I à II ne permet pas encore de montrer la terminaison de beaucoup de fonctions. Pour cela, il faudrait pouvoir prendre des sous-termes de $\vec{l}$. Malheureusement, $\rhd_s$ ne préserve pas nécessairement la calculabilité : un sous-terme stable d'un terme calculable n'est pas nécessairement calculable. Par contre, il préserve la normalisation forte. Ainsi, on peut rajouter à la clôture l'opération de la Figure 4.

FIGURE 4 – Opérations de clôture III

(subterm-SN) si $t \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$, $t \rhd_s u : \mathsf{B} \in \mathcal{B}$ et $I(\mathsf{B}) = \mathcal{SN}$, alors $u \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$

Dans le premier article utilisant implicitement une clôture de calculabilité pour la réécriture [JO91], Jouannaud et Okada prennent $I(\mathsf{B}) = \mathcal{SN}$ pour tout $\mathsf{B} \in \mathcal{B}$ et, pour CC, ils définissent un "schéma général" qui généralise au λ-calcul simplement typé le schéma de récursion primitive. Ce schéma est en fait inclut dans la clôture définie par les opérations I à III en prenant l'ordre du Lemme 10 comme $\mathcal{F}$-pré-ordre. La présente formulation, sous forme d'ensemble défini inductivement, n'apparaîtra que dans [BJO02].

La clôture permet de montrer la calculabilité des symboles définis mais la calculabilité des symboles non définis (constructeurs) dépend de la définition de l'interprétation I des types de base dans $\mathbf{Red}_{\mathcal{R}}$. On cherchera donc à définir I de façon à garantir la calculabilité des symboles non définis. Une solution simple consiste à prendre $I(\mathsf{B}) = \mathcal{SN}$ pour tout B. Mais, comme nous venons de le voir, l'ordre sous-terme stable ne préserve pas nécessairement la calculabilité. Prendre $I(\mathsf{B}) = \mathcal{SN}$ ne permet donc pas de traiter le cas des fonctions définies par récursion sur des types inductifs d'ordre supérieur, c'est-à-dire, ayant des constructeurs prenant des fonctions en argument, comme l'addition sur la notation ordinale définie par les symboles suivants [CPM88] :

$$\mathsf{o} : \mathsf{O} \quad \mathsf{s} : \mathsf{O} \Rightarrow \mathsf{O} \quad \mathsf{L} : (\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow \mathsf{O} \quad + : \mathsf{O} \Rightarrow \mathsf{O} \Rightarrow \mathsf{O}$$

$$\begin{array}{rcl} \mathsf{o} + y & \to & y \\ (\mathsf{s}x) + y & \to & \mathsf{s}(x + y) \\ (\mathsf{L}x) + y & \to & \mathsf{L}(\lambda n(xn) + y) \end{array}$$

les règles de simplification sur une algèbre de processus avec opérateur de choix sur un type de données D [GP91] :

$$\delta : \mathsf{P} \quad +, \cdot : \mathsf{P} \Rightarrow \mathsf{P} \Rightarrow \mathsf{P} \quad \Sigma : (\mathsf{D} \Rightarrow \mathsf{P}) \Rightarrow \mathsf{P} \quad \ldots$$

$$\begin{array}{rcl} \Sigma(\lambda d P) & \to & P \\ \Sigma(\lambda d (Pd) + (Qd)) & \to & (\Sigma P) + (\Sigma Q) \\ \Sigma(\lambda d (Pd) \cdot Q) & \to & (\Sigma P) \cdot Q \end{array}$$
$$\ldots$$

le calcul de la forme normale prenex dans le calcul des prédicats [MN98] :

$$\bot, \top : \mathsf{F} \quad \neg : \mathsf{F} \Rightarrow \mathsf{F} \quad \wedge, \vee : \mathsf{F} \Rightarrow \mathsf{F} \Rightarrow \mathsf{F} \quad \forall, \exists : (\mathsf{T} \Rightarrow \mathsf{F}) \Rightarrow \mathsf{F}$$

$$\begin{array}{rcl} (\forall P) \wedge Q & \to & \forall(\lambda x (Px) \wedge Q) \\ \neg(\forall P) & \to & \exists(\lambda x \neg(Px)) \end{array}$$
$$\ldots$$

ou le parcours en largeur d'abord des étiquettes d'un arbre en utilisant des continuations [Mat00] :

$$\mathrm{nil} : \mathsf{L} \quad \mathrm{cons} : \mathsf{N} \Rightarrow \mathsf{L} \to \mathsf{L} \quad \mathsf{d} : \mathsf{C} \quad \mathsf{c} : ((\mathsf{C} \Rightarrow \mathsf{L}) \Rightarrow \mathsf{L}) \to \mathsf{C} \quad \mathrm{ex} : \mathsf{C} \Rightarrow \mathsf{L}$$

$$\begin{array}{rcl} \mathrm{ex}\,\mathsf{d} & \to & \mathrm{nil} \\ \mathrm{ex}(\mathsf{c}x) & \to & x\,\mathrm{ex} \end{array}$$

En effet, dans ces cas-là, en prenant $I(\mathsf{B}) = \mathcal{SN}$, il n'est pas possible de déduire que les arguments d'un terme constructeur calculable sont calculables, à savoir : si $\mathsf{c} : \vec{T} \Rightarrow \mathsf{B}$, $\mathsf{c} \notin \mathcal{D}(\mathcal{R})$ et $\mathsf{c}\vec{t} \in I(\mathsf{B})$, alors $\vec{t} \in [\![\vec{T}]\!]^I$. Heureusement, il est possible de construire une interprétation I vérifiant cette propriété en utilisant le fait que $\mathbf{Red}_{\mathcal{R}}$ est un treillis complet sur lequel donc, toute fonction monotone admet un point fixe [Tar55]. Suivant [Mat98], plusieurs définitions sont possibles :

1. Une définition basée sur "l'élimination" utilise des symboles de fonction de récursion sur le type considéré. Par exemple, pour O, nous pouvons définir la famille indexée sur $T \in \mathcal{T}$ des récurseurs :

$$\mathrm{rec}_T : \mathsf{O} \Rightarrow T \Rightarrow (\mathsf{O} \Rightarrow T \Rightarrow T) \Rightarrow ((\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow (\mathsf{N} \Rightarrow T) \Rightarrow T) \Rightarrow T$$

$$\begin{array}{rcl} \mathrm{rec}_T \mathsf{o} uvw & \to & u \\ \mathrm{rec}_T (\mathsf{s}x) uvw & \to & vx(\mathrm{rec}_T x uvw) \\ \mathrm{rec}_T (\mathsf{L}x) uvw & \to & wx(\lambda n \mathrm{rec}_T (xn) uvw) \end{array}$$

   On peut alors définir $I(\mathsf{O})$ comme un point fixe de la fonction :

$$\begin{array}{rl} F_{\mathsf{O}}(X) = \{t \in \mathcal{L} \mid \forall T \in \mathcal{T}, \forall P \in \mathbf{Red}_{\mathcal{R}}, \forall u \in [\![\mathsf{A}]\!]^J, \forall v \in [\![\mathsf{O} \Rightarrow \mathsf{A} \Rightarrow \mathsf{A}]\!]^J, \\ \qquad \forall w \in [\![(\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow (\mathsf{N} \Rightarrow \mathsf{A}) \Rightarrow \mathsf{A}]\!]^J, \mathrm{rec}_T tuvw \in [\![\mathsf{A}]\!]^J\} \end{array}$$

   où A est une constante de type différente [7] de O et N, $J(\mathsf{A}) = P$, $J(\mathsf{O}) = X$ et $J(\mathsf{N}) = \mathcal{SN}$. Maintenant, pour montrer que x est calculable si $(\mathsf{L}x)$ est calculable, il suffit de prendre $T = \mathsf{O}$ et $w = \lambda x \lambda y x$ qui est clairement calculable. Alors, $\mathrm{rec}_{\mathsf{O}}(\mathsf{L}x)uvw \to wx(\lambda n \mathrm{rec}_{\mathsf{O}}(xn)uvw) \to x$. Cette propriété permet ensuite de montrer la calculabilité des constructeurs. Enfin, la calculabilité de rec_T est inscrite dans la définition de $I(\mathsf{O})$.

7. On suppose $\mathcal{B}$ infini. On pourrait aussi bien considérer des variables de type.

2. Une définition basée sur "l'introduction" n'utilise que les constructeurs. Dans cette approche, $I(\mathsf{O})$ est défini comme un point fixe de la fonction :

$$F_{\mathsf{O}}(X) = \{t \in \mathcal{SN} \mid \forall u, (t \to^* \mathsf{s}u \Rightarrow u \in X) \wedge (t \to^* \mathsf{L}u \Rightarrow u \in [\![\mathsf{N} \Rightarrow \mathsf{O}]\!]^J)\}$$

où $J(\mathsf{O}) = X$ et $J(\mathsf{N}) = \mathcal{SN}$. Cette fois-ci, c'est la calculabilité des arguments qui est inscrite dans la définition de $I(\mathsf{O})$.

Dans [Mat98], Matthes considère des ensembles saturés (non stables par réduction) et, dans ce cas, les deux définitions (en prenant le plus petit point fixe dans les deux cas) peuvent ne pas être équivalentes. Avec des candidats de réductibilité, ce n'est pas le cas : les deux définitions sont équivalentes.

Dans les deux cas, la monotonie de F_{O} est garantie par le fait que O n'apparaît que *positivement* dans le type des arguments des constructeurs de O, en sachant que A apparaît positivement dans $B \Rightarrow A$ et *négativement* dans $A \Rightarrow B$. Formellement :

Definition 12 (Positions positives et négatives) Étant donné un type T, les positions *positives* de T, $\mathrm{Pos}^+(T)$, et les positions *négatives* de T, $\mathrm{Pos}^-(T)$, sont les sous-ensembles de $\{0,1\}^*$ définis ainsi :

- $\mathrm{Pos}^+(\mathsf{B}) = \{\varepsilon\}$
- $\mathrm{Pos}^-(\mathsf{B}) = \emptyset$
- $\mathrm{Pos}^+(T \Rightarrow U) = \{0w \mid w \in \mathrm{Pos}^-(T)\} \cup \{1w \mid w \in \mathrm{Pos}^+(U)\}$
- $\mathrm{Pos}^-(T \Rightarrow U) = \{0w \mid w \in \mathrm{Pos}^+(T)\} \cup \{1w \mid w \in \mathrm{Pos}^-(U)\}$

Par ailleurs, les positions des occurrences d'une constante B dans T, $\mathrm{Pos}(\mathsf{B}, T)$, sont :

- $\mathrm{Pos}(\mathsf{B}, \mathsf{B}) = \{\varepsilon\}$
- $\mathrm{Pos}(\mathsf{B}, \mathsf{C}) = \emptyset$ si $\mathsf{B} \neq \mathsf{C}$
- $\mathrm{Pos}(\mathsf{B}, T \Rightarrow U) = \{0w \mid w \in \mathrm{Pos}(\mathsf{B}, T)\} \cup \{1w \mid w \in \mathrm{Pos}(\mathsf{B}, U)\}$

En fait, Mendler a montré que si un type B a un constructeur c dont le type d'un des arguments a une occurrence négative de B, alors $\to_\beta \cup \to_{\mathcal{R}}$ où $\mathcal{R} = \{\mathsf{p}_i(\mathsf{c}\vec{x}) \to x_i\}$ ne termine pas [Men87]. Par exemple, avec $\mathsf{c} : (\mathsf{B} \Rightarrow \mathsf{N}) \Rightarrow \mathsf{B}$, $\mathsf{p} : \mathsf{B} \Rightarrow (\mathsf{B} \Rightarrow \mathsf{N})$ et la règle $\mathsf{p}(\mathsf{c}x) \to x$ signifiant que c est injective, on peut reproduire le contre-exemple du λ-calcul non typé : en prenant $t = \lambda x \mathsf{p} x x$, nous avons $t(\mathsf{c}t) \to_\beta \mathsf{p}(\mathsf{c}t)(\mathsf{c}t) \to_{\mathcal{R}} t(\mathsf{c}t)$.

Cela conduit aux restrictions générales suivantes qu'on trouve par exemple dans le calcul des constructions inductives [CPM88, Wer94] :

Definition 13 (Système inductif standard) Étant donné un ensemble $\mathcal{R}$ de règles de réécriture, l'ensemble des constantes de type $\mathcal{B}$ et l'ensemble des symboles non définis $\mathcal{F} - \mathcal{D}(\mathcal{R})$ forment un *système inductif standard* s'il existe un pré-ordre $\geq_{\mathcal{B}}$ sur $\mathcal{B}$ dont la partie stricte est bien fondée et tel que, pour tout $\mathsf{B} \in \mathcal{B}$, $\mathsf{c} : \vec{T} \Rightarrow \mathsf{B}$, $\mathsf{c} \notin \mathcal{D}(\mathcal{R})$, $i \in [1, |\vec{T}|]$ et $\mathsf{C} \in \mathcal{B}$:

- si $\mathsf{C} >_{\mathcal{B}} \mathsf{B}$ alors $\mathrm{Pos}(\mathsf{C}, T_i) = \emptyset$;
- si $\mathsf{C} \simeq_{\mathcal{B}} \mathsf{B}$ alors $\mathrm{Pos}(\mathsf{C}, T_i) \subseteq \mathrm{Pos}^+(T_i)$.

Prendre un pré-ordre plutôt qu'un ordre permet de traiter le cas des types inductifs mutuellement définis.

Dans un tel système, on peut définir $I : \mathcal{B} \to \mathbf{Red}_{\mathcal{R}}$ par induction sur $>_{\mathcal{B}}$ et, pour chaque classe d'équivalence E modulo $\simeq_{\mathcal{B}}$, comme le point fixe d'une fonction monotone sur le treillis complet $E \to \mathbf{Red}_{\mathcal{R}}$ ordonné point à point par inclusion ($I \leq J$ si, pour tout $\mathsf{B} \in E$, $I(\mathsf{B}) \subseteq J(\mathsf{B})$), de façon à ce que I soit stable par $\simeq_{\mathcal{B}}$: $I(\mathsf{B}) = I(\mathsf{C})$ si $\mathsf{B} \simeq_{\mathcal{B}} \mathsf{C}$.

Avec cette interprétation, tous les symboles $\mathsf{f} \in \mathcal{F} - \mathcal{D}(\mathcal{R})$ sont calculables et on peut rajouter les opérations de clôture de la Figure 5.

FIGURE 5 – Opérations de clôture V^- pour les systèmes inductifs standards

(undef)	$\mathcal{F} - \mathcal{D}(\mathcal{R}) \subseteq \mathrm{CC}_{\mathsf{f}}(\vec{l})$
(subterm-undef)	si $\mathsf{c}\vec{t} \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$, $\mathsf{c}\vec{t} : \mathsf{B}$ et $\mathsf{c} \notin \mathcal{D}(\mathcal{R})$, alors $\{\vec{t}\} \subseteq \mathrm{CC}_{\mathsf{f}}(\vec{l})$

Par contre, l'ordre sous-terme stable n'est plus suffisant pour montrer la terminaison des systèmes du début de la section. Par exemple, pour l'addition sur la notation ordinale définie par O, partant d'un argument de la forme $(\mathsf{L}x)$, nous avons un appel récursif avec un argument de la forme (xn) où n est une variable liée. Bien que x soit un sous-terme de $(\mathsf{L}x)$, xn n'en est pas un. Pire, dans le cas des continuations, partant d'un argument de la forme $(\mathsf{c}x)$, la fonction ex n'est appliquée à aucun argument mais est elle-même argument de x...

Si, pour chaque classe d'équivalence E modulo $\simeq_{\mathcal{B}}$, nous prenons le *plus petit* point fixe (l'ensemble des points fixes forment un treillis [Tar55]), alors celui-ci peut être obtenu par itération transfinie (un résultat dû à Kleene) : il existe un ordinal $\mathfrak{a}$ tel que, pour tout $\mathsf{B} \in E$, $I(\mathsf{B}) = F^{\mathfrak{a}}_{\mathsf{B}}(\bot)$ où $\bot$ est le plus petit candidat et $F^{\mathfrak{a}}$ est défini par récursion transfinie :

- $F^0(X) = X$
- $F^{\mathfrak{a}+1}(X) = F^{\mathfrak{a}}(F(X))$
- $F^{\mathfrak{l}}(X) = \bigvee\{F^{\mathfrak{a}}(X) \mid \mathfrak{a} < \mathfrak{l}\}$ si $\mathfrak{l}$ est un ordinal limite

Cela nous fournit une notion de rang avec laquelle comparer les termes calculables entre eux :

Definition 14 (Rang d'un terme calculable) Le *rang* d'un terme $t \in I(\mathsf{B})$, $\mathrm{rk}_{\mathsf{B}}(t)$, est le plus petit ordinal $\mathfrak{a}$ tel que $t \in F^{\mathfrak{a}}_{\mathsf{B}}(\bot)$. Soit alors $\succeq_{\mathsf{B}}$ l'ordre sur $I(\mathsf{B})$ tel que $t \succeq_{\mathsf{B}} u$ si $\mathrm{rk}_{\mathsf{B}}(t) \geq \mathrm{rk}_{\mathsf{B}}(u)$.

La relation $\succeq_{\mathsf{B}}$ est un pré-ordre et $\succ_{\mathsf{B}} \cup \to$ est bien fondé car $\succ_{\mathsf{B}} \to \subseteq \succ_{\mathsf{B}}$. Malheureusement, $\succ_{\mathsf{B}}$ n'est pas stable par substitution. Par exemple, dans le type N permettant de noter les entiers naturels en base 1 (entiers de Peano), $\mathsf{s0} >_{\mathsf{N}} y$ mais $\mathsf{s0} <_{\mathsf{N}} \mathsf{s}(\mathsf{s0})$. Restreindre $t \succ_{\mathsf{B}} u$ au cas où $\mathrm{FV}(u) \subseteq \mathrm{FV}(t)$ n'est pas une solution car, dans l'exemple de l'addition ordinale, il nous faut pouvoir comparer $(\mathsf{L}x)$ avec (xn). À la place, nous allons donc considérer un sous-pré-ordre de $\succeq_{\mathsf{B}}$ ayant cette propriété pour les arguments *strictement positifs* [Coq92] :

Definition 15 (Ordre structurel) Le i-ième argument de $\mathsf{c} : \vec{T} \Rightarrow \mathsf{B}$ est *strictement positif* si $T_i = \vec{U} \Rightarrow \mathsf{C}$, $\mathsf{C} \simeq \mathsf{B}$ et, pour tout $\mathsf{D} \simeq_{\mathcal{B}} \mathsf{B}$, $\mathrm{Pos}(\mathsf{D}, \vec{U}) = \emptyset$. Soit $\rhd_c$ la plus petite relation transitive sur $\mathcal{L}$ telle que, pour tout $\mathsf{c} : \vec{T} \Rightarrow \mathsf{B}$, $\vec{t}$ et $i \in [1, |\vec{t}|]$, nous avons $\mathsf{c}\vec{t} \rhd_c t_i$ si $|\vec{t}| = |\vec{T}|$ et i est strictement positif. Étant données deux termes calculables $t : \mathsf{B}$ et $u : \mathsf{C}$ tels que $\mathsf{B} \simeq \mathsf{C}$, t est *structurellement plus grand* que u, $t >_s u$, s'il existe v et $\vec{u}$ calculables tels que $t \rhd_c v$ et $u = v\vec{u}$. Soit enfin $\geq_s$ la clôture réflexive de $>_s$.

La relation $\geq_s$ est un ordre stable par substitution inclus dans $\succeq_{\mathsf{B}}$.

Lemma 16 Étant donné un pré-ordre bien fondé $\geq_{\mathcal{F}}$ sur $\mathcal{F}$, un système de filtrage φ et, pour chaque classe E, un statut $\mathrm{stat}_E \in \{\mathrm{lex}, \mathrm{mul}\}$ et :

- si $\mathrm{stat}_E = \mathrm{mul}$: une constante de type B_E tel que, pour tout $\mathsf{f} : \vec{T} \Rightarrow \mathsf{B}$ dans E, chaque élément de $\varphi^{\mathcal{T}}_{\mathsf{f}}(\vec{T})$ est équivalent à B_E,
- si $\mathrm{stat}_E = \mathrm{lex}$: une séquence de types de base $\vec{\mathsf{B}}_E$ telle que, pour tout $\mathsf{f} : \vec{T} \Rightarrow \mathsf{B}$ dans E, nous avons $\varphi^{\mathcal{T}}_{\mathsf{f}}(\vec{T}) \simeq_{\mathcal{B}} \vec{\mathsf{B}}_E$,

alors le produit lexicographique dépendant $\geq$ défini par $\geq_{\mathcal{F}}$ et :

- si $\mathrm{stat}_E = \mathrm{mul}$: $Z_E = \mathbb{M}(I(\mathsf{B}_E))$, $\psi_{\mathsf{f}}(\vec{t}) = \{\varphi^{\mathcal{L}}_{\mathsf{f}}(\vec{t})\}$ et $\geq_E = (\rightarrow^* >_s)_{\mathrm{mul}}$,
- si $\mathrm{stat}_E = \mathrm{lex}$: $Z_E = [\![\vec{\mathsf{B}}_E]\!]$, $\psi_{\mathsf{f}}(\vec{t}) = \varphi^{\mathcal{L}}_{\mathsf{f}}(\vec{t})$ et $\geq_E = (\rightarrow^* >_s)_{\mathrm{lex}}$,

est un $(\mathcal{F}, \varphi)$-pré-ordre tel que $>$ est stable par substitution et $>^{\varphi} \cup \rightarrow_{\mathrm{prod}}$ est bien fondée.

L'union avec $\rightarrow_{\mathrm{prod}}$ est bien fondée car $>_s \rightarrow \subseteq \rightarrow >_s$.

Malheureusement, l'ordre structurel ne permet pas d'orienter l'exemple utilisant le type des continuations qui n'est pas strictement positif. Nous verrons en Section 4.8 comment résoudre ce problème.

4.3 Filtrage sur des symboles définis

En vérifiant les conditions de positivité pour chaque argument séparément, on voit que la définition basée sur les constructeurs peut être généralisée de façon à étendre, d'une part, les opérations de clôture de la Figure 5 à des systèmes qui ne sont pas des systèmes inductifs standards et, d'autre part, la propriété des sous-termes calculables ($\mathsf{f}\vec{t} \in I(\mathsf{B}) \Rightarrow t_i$ calculable) aux symboles définis $\mathsf{f} \in \mathcal{D}(\mathcal{R})$:

Definition 17 (Argument accessible) Étant donné un pré-ordre bien fondé $\geq_{\mathcal{B}}$ sur $\mathcal{B}$, l'ensemble des *positions accessibles* d'un symbole $\mathsf{f} : \vec{T} \Rightarrow \mathsf{B}$, $\mathrm{Acc}(\mathsf{f})$, est l'ensemble des entiers $i \in [1, |\vec{T}|]$ tels que :
- pour tout symbole $\mathsf{C} >_{\mathcal{B}} \mathsf{B}$, $\mathrm{Pos}(\mathsf{C}, T_i) = \emptyset$;
- pour tout symbole $\mathsf{C} \simeq_{\mathcal{B}} \mathsf{B}$, $\mathrm{Pos}(\mathsf{C}, T_i) \subseteq \mathrm{Pos}^{+}(T_i)$.

Pour O, nous pouvons alors prendre :

$$F_{\mathsf{O}}(X) = \{t \in \mathcal{SN} \mid \forall \mathsf{f} \in \mathcal{M}(\mathcal{R}), \forall \vec{T}, \forall \vec{u}, \tau_{\mathsf{f}} = \vec{T} \Rightarrow \mathsf{O} \wedge |\vec{T}| = |\vec{u}| \wedge t \to^{*} \mathsf{f}\vec{u} \Rightarrow \forall i \in \mathrm{Acc}(\mathsf{f}), u_i \in [\![T_i]\!]^{J}\}$$

où $\mathcal{M}(\mathcal{R})$ est l'ensemble des symboles f, dits *filtrés*, qui sont sous-terme *strict* d'un membre gauche de règle et pour lesquels $\mathrm{Acc}(\mathsf{f})$ est non vide, à condition que les termes de la forme $\mathsf{f}\vec{t}$ avec $\mathsf{f} \in \mathcal{M}(\mathcal{R})$ ne soient pas considérés comme neutre, afin que la propriété $(R3)$ soit bien vérifiée.

Definition 18 (Terme neutre - Nouvelle définition) Étant donné un ensemble $\mathcal{R}$ de règles de réécriture, un terme est *neutre* s'il est de la forme $x\vec{v}$, $(\lambda x t)u\vec{v}$ ou $\mathsf{f}\vec{v}$ avec $\mathsf{f} \in \mathcal{D}(\mathcal{R}) - \mathcal{M}(\mathcal{R})$ et $|\vec{v}| \geq \alpha_{\mathsf{f}} = sup\{|\vec{l}| \mid \mathsf{f}\vec{l} \to r \in \mathcal{R}\}$.

Cependant, même en changeant ainsi la définition des termes neutres, nous avons toujours la propriété suivante qui est en fait suffisante pour montrer les Théorèmes 5 et 9 :

Lemma 19 Pour tout symbole $\mathsf{f} \in \mathcal{F}$, un terme de type de base de la forme $\mathsf{f}\vec{t}$ est calculable si tous ses réduits et arguments accessibles sont calculables.

On peut alors généraliser les opérations de clôture de la Figure 5 qui ne sont valables que pour les systèmes inductifs standards par les opérations de clôture de la Figure 6.

FIGURE 6 – Opérations de clôture V

(undef)	$\mathcal{F} - \mathcal{D}(\mathcal{R}) \subseteq \mathrm{CC}_{\mathsf{f}}(\vec{l})$
(subterm-acc)	si $\mathsf{g}\vec{t} \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$, $\mathsf{g}\vec{t} : \mathsf{B}$ et $i \in \mathrm{Acc}(\mathsf{g})$, alors $t_i \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$

Ainsi, on peut par exemple étendre par une règle d'associativité l'addition sur la notation ordinale définie par O ou la composition alternative dans μCRL :

$$(x + y) + z \ \rightarrow \ x + (y + z)$$

Par ailleurs, on peut également donner un critère syntaxique pour savoir si $I(\mathsf{B}) = \mathcal{SN}$, condition utilisée dans (subterm-SN) :

Definition 20 (Type basique) Une constante de type est *basique* si sa classe d'équivalence modulo $\simeq_{\mathcal{B}}$ est basique. Une classe d'équivalence E est *basique* si pour tout $\mathsf{B} \in E$, $\mathsf{f} \in \mathcal{M}(\mathcal{R})$, $\mathsf{f} : \vec{T} \Rightarrow \mathsf{B}$, $i \in \mathrm{Acc}(\mathsf{f})$, T_i est une constante de type C telle que $\mathsf{C} \in E$ ou $\mathsf{C} <_{\mathcal{B}} \mathsf{B}$ et $[\mathsf{C}]$ est basique.

Lemma 21 Si B est basique, alors $I(\mathsf{B}) = \mathcal{SN}$.

En particulier, tous les types du premier ordre sont basiques.

4.4 Réécriture modulo une théorie équationnelle

[Bla03] F. Blanqui. Rewriting modulo in deduction modulo. In *Proceedings of the 14th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 2706, 2003.

La réécriture a été à l'origine introduite comme outil de décision pour les théories équationnelles [KB70]. En effet, une théorie équationnelle $=_{\mathcal{E}}$, *i.e.* la clôture réflexive, symétrique, transitive, monotone et par substitution d'un ensemble $\mathcal{E}$ d'équations, est décidable s'il existe un ensemble $\mathcal{R}$ de règles de réécriture telles que $\to_{\mathcal{R}}$ termine et conflue, $\mathcal{R} \subseteq =_{\mathcal{E}}$ (correction) et $\mathcal{E} \subseteq =_{\mathcal{R}}$ (complétude). Knuth et Bendix ont inventé une procédure dite de complétion qui permet, en cas de succès, d'obtenir un tel ensemble $\mathcal{R}$ à partir de $\mathcal{E}$. Cette procédure consiste notamment à orienter les équations de $\mathcal{E}$ de façon à les utiliser comme règles de réécriture. Or, certaines équations ou ensembles d'équations, comme la commutativité, ne sont pas orientables (aucune orientation ne permet d'obtenir une relation qui termine). Une solution consiste alors à raisonner modulo ces équations non orientables $\mathcal{E}$ et considérer de la réécriture modulo $\mathcal{E}$, *i.e.* la relation $t =_{\mathcal{E}} \to_{\mathcal{R}} u$ s'il existe t' tel que $t =_{\mathcal{E}} t'$ et $t' \to_{\mathcal{R}} u$ [LB77, Hue80]. Une autre solution, de complexité moins élevée, consiste à considérer de la réécriture avec *filtrage modulo* $\mathcal{E}$, *i.e.* la relation $t \to_{\mathcal{R},\mathcal{E}} u$ s'il existe une position $p \in \mathrm{Pos}(t)$, une règle $l \to r \in \mathcal{R}$ et une substitution σ tels que $t|_p =_{\mathcal{E}} l\sigma$ et $u = t[r\sigma]_p$ [PS81, JK86]. Des implémentations efficaces de ce type de réécriture ont été développé [Eke96, KM01] et permettent la simulation et le model-checking de systèmes utilisant des équations non orientables, en particulier, l'associativité et la commutativité, comme c'est par exemple le cas dans la simulation de réactions chimiques ou de protocoles cryptographiques (l'ordre des molécules dans une formule ou l'ordre dans lequel des messages envoyés sont reçus peut être changé).

Nous allons maintenant montrer [8] comment la clôture de calculabilité permet de prouver la terminaison de $\to = \to_\beta \cup =_{\mathcal{E}} \to_{\mathcal{R}}$ pour les ensembles d'équations $\mathcal{E}$ tels que $\to_\beta$ commute avec $=_{\mathcal{E}}$, *i.e.* $=_{\mathcal{E}} \to_\beta \subseteq \to_\beta =_{\mathcal{E}}$. C'est en particulier le cas pour les ensembles suivants :

Lemma 22 Étant donné un ensemble d'équations $\mathcal{E}$, $\to_\beta$ commute avec $=_{\mathcal{E}}$ si :
- $\mathcal{E}$ est linéaire : $\forall l = r \in \mathcal{E}$, l et r sont linéaires ;
- $\mathcal{E}$ est régulier : $\forall l = r \in \mathcal{E}$, $\mathrm{FV}(l) = \mathrm{FV}(r)$;
- $\mathcal{E}$ est algébrique.

Ces conditions excluent les équations non régulières comme $x \times 0 = 0$ ou non linéaires comme $x \times (y + z) = (x \times y) + (x \times z)$, qui sont souvent orientables de toute façon. De plus, s'il y a une équation non régulière, la relation $=_{\mathcal{E}} \to_{\mathcal{R}}$ ne

8. Cette section corrige certaines lacunes de [Bla03].

termine pas. En effet, s'il existe $g = d \in \mathcal{E}$, $x \in \mathrm{FV}(g) - \mathrm{FV}(d)$ et $l \to r \in \mathcal{R}$, alors $d = d^l_x =_\mathcal{E} g^l_x \to_\mathcal{R} g^r_x =_\mathcal{E} d^r_x = d$ [JK86].

La condition d'algébricité au sens strict peut être relâchée dans certains cas. Par exemple, la commutation des quantificateurs nécessaire à la confluence des règles de réécriture permettant de calculer la forme normale prenex d'une formule [MN98] commute avec β :

$$\forall(\lambda x \forall(\lambda y P x y)) \;=\; \forall(\lambda y \forall(\lambda x P x y))$$

Maintenant, nous généralisons la notion de calculabilité au cas équationnel :

Definition 23 (Calculabilité modulo des équations) Étant donnés un ensemble $\mathcal{R}$ de règles de réécriture de la forme $\mathsf{f}\vec{l} \to r$ et un ensemble $\mathcal{E}$ d'équations de la forme $\mathsf{f}\vec{l} = \mathsf{g}\vec{m}$, un terme est *neutre* s'il est de la forme $x\vec{v}$, $(\lambda x t)u\vec{v}$ ou $\mathsf{f}\vec{v}$ avec $\mathsf{f} \in \mathcal{D}(\mathcal{R} \cup \mathcal{E} \cup \mathcal{E}^{-1}) - \mathcal{M}(\mathcal{R} \cup \mathcal{E} \cup \mathcal{E}^{-1})$ et $|\vec{v}| \geq \alpha_\mathsf{f} = sup\{|\vec{l}| \mid \mathsf{f}\vec{l} \to r \in \mathcal{R} \cup \mathcal{E} \cup \mathcal{E}^{-1}\}$. Soit $\mathbf{Red}^\mathcal{E}_\mathcal{R}$ l'ensemble des $P \subseteq \mathcal{L}$ tels que :

(R0) $P \subseteq \mathcal{SN}(\to)$ où $\to = \to_\beta \cup =_\mathcal{E} \to_\mathcal{R}$;
(R1) $\mathcal{X} \subseteq P$;
(R2) P est stable par $\to \cup =_\mathcal{E}$;
(R3) si t est un terme neutre et $\to(t) \subseteq P$, alors $t \in P$.

$\mathbf{Red}^\mathcal{E}_\mathcal{R}$ est défini de telle sorte que $\mathbf{Red}^\emptyset_\mathcal{R} = \mathbf{Red}_\mathcal{R}$. On vérifie par ailleurs que $\mathbf{Red}^\mathcal{E}_\mathcal{R}$ a toujours les bonnes propriétés :

Lemma 24 $\mathbf{Red}^\mathcal{E}_\mathcal{R}$ est stable par a et intersection non vide et admet $\mathcal{SN}(\to)$ comme plus grand élément si $\to_\beta$ commute avec $=_\mathcal{E}$ et si les termes neutres sont stables par $=_\mathcal{E}$.

Proof. La preuve est similaire à celle du Lemme 3. La stabilité par intersection non vide se montre aisément. Le fait que $\mathcal{SN}(\to) \in \mathbf{Red}_\mathcal{R}$ découle du fait que $\to_\beta$ commute avec $=_\mathcal{E}$. Pour la stabilité par a, il n'y pas de changement pour (R0), (R1) et (R2). Nous détaillons maintenant (R3). Soient $P, Q \in \mathbf{Red}^\mathcal{E}_\mathcal{R}$ et t neutre tel que $\to(t) \subseteq a(P, Q)$, et $u \in P$. Nous montrons que $tu \in Q$ par induction bien fondée sur u avec $\to$ comme relation bien fondée ($u \in \mathcal{SN}$ par (R0)). Comme tu est neutre, par (R3), il suffit de montrer que $\to(tu) \subseteq Q$. Soit $v \in \to(tu)$. Si $tu \to_\beta \cup \to_\mathcal{R} v$, nous avons déjà vu que (*) : soit $v = t'u$ avec $t \to t'$, soit $v = tu'$ avec $u \to u'$. Pour $=_\mathcal{E} \to_\mathcal{R}$, nous montrons (**) : si t est neutre et $tu =_\mathcal{E} \to_\mathcal{R} v$, alors $v = t'u'$ avec, soit $t \to t'$ et $u =_\mathcal{E} u'$, soit $t =_\mathcal{E} t'$ neutre et $u \to u'$. Comme $=_\mathcal{E}$ est la clôture réflexive et transitive de $\leftrightarrow_\mathcal{E} = \to_\mathcal{E} \cup \leftarrow_\mathcal{E}$, nous montrons (**) par induction sur le nombre d'étapes équationnelles entre tu et v. S'il n'y en a aucune, cela découle de (*). Supposons maintenant que $tu \leftrightarrow_\mathcal{E} w =_\mathcal{E} \to_\mathcal{R} v$. Comme les équations vérifient les mêmes conditions que les règles de réécriture, et de manière symétrique, par (*), soit $w = t'u$ avec $t \leftrightarrow_\mathcal{E} t'$, soit $w = tu'$ avec $u \leftrightarrow_\mathcal{E} u'$. Comme les termes neutres sont stables par $=_\mathcal{E}$, t' est neutre et, par hypothèse d'induction, (**) est vérifié. Nous montrons maintenant que $v = t'u' \in Q$.

– $t \to t'$ et $u =_{\mathcal{E}} u'$. Par hypothèse, $t' \in a(P,Q)$. Par (R2), $u' \in P$. Donc, par définition de a, $v \in Q$.
– $t =_{\mathcal{E}} t'$ neutre et $u \to u'$. Comme $\to_\beta$ commute avec $=_{\mathcal{E}}$, $\to$ commute avec $=_{\mathcal{E}}$ et, par (R2), $\to(t') \subseteq a(P,Q)$. Comme t' est neutre, par hypothèse d'induction sur u, nous avons $v \in Q$. ■

Les conditions sur les équations excluent les équations dites *effondrantes* dont l'un des membres est une variable, comme par exemple $x + 0 = x$, mais ces équations sont souvent orientables. De plus, s'il y a une équation $t = x \in \mathcal{E}$ avec au moins deux occurrences de x dans t (*e.g.* $x + x = x$), alors $=_{\mathcal{E}}\to_{\mathcal{R}}$ ne termine pas. En effet, soit p une des occurrences de x dans t, et soit $t' = t[y]_p$ où $y \notin \mathrm{FV}(t)$. Si $l \to r \in \mathcal{R}$, alors $l =_{\mathcal{E}} t\binom{l}{x} = t'\binom{l}{x}\binom{l}{y} \to_{\mathcal{R}} t'\binom{l}{x}\binom{r}{y} \dots$ [JK86].

Par ailleurs, les termes neutres sont stables par $=_{\mathcal{E}}$ si, pour chaque équation $l = r \in \mathcal{E}$, l est de la forme $\mathsf{f}\vec{l}$, r est de la forme $\mathsf{g}\vec{m}$, et l est neutre ssi r est neutre.

Theorem 25 Pour tout ensemble de règles $\mathcal{R}$ et tout ensemble d'équations $\mathcal{E}$ tel que $\to_\beta$ commute avec $=_{\mathcal{E}}$, la relation $\to = \to_\beta \cup =_{\mathcal{E}}\to_{\mathcal{R}}$ termine s'il existe $I : \mathcal{B} \to \mathbf{Red}^{\mathcal{E}}_{\mathcal{R}}$ et un $(\mathcal{F},\varphi)$-pré-ordre $\geq$ tel que $\simeq$ et $>$ sont stables par substitution, $\simeq^\varphi$ contient $(=_{\mathcal{E}})_{\mathrm{prod}}$, $>^\varphi \cup \to_{\mathrm{prod}}$ est bien fondé, et :
– tout symbole $\mathsf{f} \in \mathcal{F} - \mathcal{D}(\mathcal{R})$ est calculable ;
– pour toute règle $l \to r \in \mathcal{R}$, l est de la forme $\mathsf{f}\vec{l}$ et $r \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$;
– pour toute équation $l = r \in \mathcal{E}$, l est de la forme $\mathsf{f}\vec{l}$, r est de la forme $\mathsf{g}\vec{m}$, $\vec{m} \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$, $\vec{l} \in \mathrm{CC}_{\mathsf{g}}(\vec{m})$ et $(\mathsf{f},\vec{l}) \simeq^\varphi (\mathsf{g},\vec{m})$;
où CC est la plus petite $\mathcal{F}$-famille close par les opérations I à V.

Proof. Nous procédons comme pour le Théorème 9. Nous montrons que, pour tout $(\mathsf{f},\vec{t}) \in \Sigma$, $\mathsf{f}\vec{t}$ est calculable, par induction bien fondée sur $>^\varphi \cup \to_{\mathrm{prod}}$. D'après le Lemme 19, il suffit de montrer que tous les réduits t de $\mathsf{f}\vec{t}$ sont calculables. Deux cas sont possibles :

1. $t = \mathsf{f}\vec{u}$ avec $\vec{t}(\to_\beta)_{\mathrm{prod}}\vec{u}$. Par (R2), $\vec{u}$ est calculable. Donc, par hypothèse d'induction, t est calculable.
2. Autrement, $\mathsf{f}\vec{t} =_{\mathcal{E}} u \to_{\mathcal{R}} t$. Nous commençons par montrer par induction sur le nombre d'étapes équationnelles entre $\mathsf{f}\vec{t}$ et u, que u est de la forme $\mathsf{g}\vec{u}$ avec $\vec{u}$ calculable et $(\mathsf{f},\vec{t}) \simeq^\varphi (\mathsf{g},\vec{u})$. S'il n'y a aucune étape équationnelle, c'est immédiat. Supposons maintenant que $\mathsf{f}\vec{t} =_{\mathcal{E}} u' \leftrightarrow_{\mathcal{E}} u$. Par hypothèse d'induction, u' est de la forme $\mathsf{g}\vec{u}$ avec $\vec{u}$ calculable et $(\mathsf{f},\vec{t}) \simeq^\varphi (\mathsf{g},\vec{u})$. Les conditions étant symétriques, le cas de $\leftarrow_{\mathcal{E}}$ est similaire à celui de $\to_{\mathcal{E}}$ pour lequel il y a deux possibilités :

(a) $u = \mathsf{g}\vec{v}$ avec $\vec{u}(\to_{\mathcal{E}})_{\mathrm{prod}}\vec{v}$. Par (R2), $\vec{v}$ est calculable et $(\mathsf{g},\vec{u}) \simeq^\varphi (\mathsf{g},\vec{v})$ car $\simeq^\varphi$ contient $(=_{\mathcal{E}})_{\mathrm{prod}}$. Donc, par transitivité, $(\mathsf{f},\vec{t}) \simeq^\varphi (\mathsf{g},\vec{v})$.

(b) Il existe $\mathsf{g}\vec{l} = \mathsf{h}\vec{m} \in \mathcal{E}$, σ et $\vec{w}$ tels que $\vec{u} = \vec{l}\sigma\vec{w}$ et $u = \mathsf{h}\vec{m}\sigma\vec{w}$. Par hypothèse, $\vec{m} \in \mathrm{CC}_{\mathsf{g}}(\vec{l})$ et $(\mathsf{g},\vec{l}) \simeq^\varphi (\mathsf{h},\vec{m})$. Comme $>$ est clos par substitution, CC est clos par substitution et $\vec{m}\sigma \in \mathrm{CC}_{\mathsf{g}}(\vec{l}\sigma)$. Comme $\simeq$ est clos par substitution,

$(\mathsf{g}, \vec{l}\sigma) \simeq^{\varphi} (\mathsf{h}, \vec{m}\sigma)$. Donc, $(\mathsf{g}, \vec{l}\sigma\vec{w}) \simeq^{\varphi} (\mathsf{h}, \vec{m}\sigma\vec{w})$ et, par transitivité, $(\mathsf{f}, \vec{t}) \simeq^{\varphi} (\mathsf{h}, \vec{m}\sigma\vec{w})$. On peut alors montrer par induction sur $v \in \mathrm{CC}_{\mathsf{g}}(\vec{l}\sigma)$ que v est calculable. Dans le cas de (rec), si nous avons $v = \mathsf{k}\vec{v}$ avec $(\mathsf{g}, \vec{l}\sigma) >^{\varphi} (\mathsf{k}, \vec{v})$, alors on peut conclure par hypothèse d'induction car $(\mathsf{f}, \vec{t}) \simeq^{\varphi} (\mathsf{g}, \vec{l}\sigma) >^{\varphi} (\mathsf{k}, \vec{v})$.

Maintenant, pour obtenir t, il y a aussi deux possibilités :

(a) $t = \mathsf{g}\vec{v}$ avec $\vec{u}(\to_{\mathcal{R}})_{\mathrm{prod}}\vec{v}$. Par (R2), $\vec{v}$ est calculable et, par hypothèse d'induction, t est calculable.

(b) Il existe $\mathsf{g}\vec{l} \to r \in \mathcal{R}$, σ et $\vec{w}$ tels que $\vec{u} = \vec{l}\sigma\vec{w}$ et $t = r\sigma\vec{w}$. Par hypothèse, $r \in \mathrm{CC}_{\mathsf{g}}(\vec{l})$. Comme CC est clos par substitution, nous avons $r\sigma \in \mathrm{CC}_{\mathsf{g}}(\vec{l}\sigma)$. On peut alors montrer par induction sur $v \in \mathrm{CC}_{\mathsf{g}}(\vec{l}\sigma)$ que v est calculable. Dans le cas de (rec), on peut conclure par hypothèse d'induction. ■

Nous allons maintenant définir un pré-ordre suffisant pour traiter les équations d'associativité et de commutativité (AC).

Definition 26 (Sous-termes étrangers) Étant donné un ensemble $E \subseteq \mathcal{F}$, le *E-chapeau algébrique* d'un terme t est, à renommage près des variables libres, le plus grand terme algébrique linéaire $\mathrm{cap}_E(t)$ ne contenant que des symboles de E et des variables tel qu'il existe une substitution σ vérifiant $t = \mathrm{cap}_E(t)\sigma$. Les termes $\sigma(x)$ pour $x \in \mathrm{FV}(\mathrm{cap}_E(t))$ sont appelés les sous-termes *E-étrangers* de t. Soit $\mathrm{aliens}_E(t)$ le multi-ensemble des sous-termes E-étrangers de t. Les *E-étrangers* d'une séquence de termes $\vec{t}$ est l'union multi-ensemble $\mathrm{Aliens}_E(t)$ des multi-ensembles $\mathrm{aliens}_E(t_i)$.

Lemma 27 Soit $\geq_{\mathcal{F}}$ un pré-ordre sur $\mathcal{F}$ et $\mathcal{E}$ un ensemble d'équations de la forme $\mathsf{f}\vec{l} = \mathsf{g}\vec{m}$ avec $\mathsf{f} \simeq_{\mathcal{F}} \mathsf{g}$ telle que $\to_\beta$ commute avec $=_{\mathcal{E}}$ et, dans chaque classe d'équivalence modulo $=_{\mathcal{E}}$, la taille des termes est bornée. Alors, étant donnée une classe d'équivalence E modulo $\simeq_{\mathcal{F}}$, la relation $\vec{t} \geq_E \vec{u}$ définie par $\mathrm{Aliens}_E(\vec{t}) \geq_{\mathcal{M}} \mathrm{Aliens}_E(\vec{u})$ où $\mathcal{M} = \mathbb{M}(\mathcal{SN})$ et $\geq$ est le plus petit pré-ordre contenant $=_{\mathcal{E}}$, $\to$ et $\rhd_s$, est un pré-ordre tel que $\simeq$ est stable par substitution et contient $(=_{\mathcal{E}})_{\mathrm{prod}}$, $>$ est stable par substitution et $> \cup \to_{\mathrm{prod}}$ est bien fondée.

Proof. La partie stricte de $\geq$, $>$, est la clôture transitive de $=_{\mathcal{E}}\to$ et $=_{\mathcal{E}} \rhd_s$. C'est une relation bien fondée sur $\mathcal{SN}$ car $=_{\mathcal{E}}\to$ commute avec $=_{\mathcal{E}} \rhd_s$ et chacune des relations est bien fondée sur $\mathcal{SN}$. La relation $=_{\mathcal{E}}\to$ est bien fondée sur $\mathcal{SN}$ car $\to_\beta$ commute avec $=_{\mathcal{E}}$ et $\to$ est bien fondée sur $\mathcal{SN}$. La relation $=_{\mathcal{E}} \rhd_s$ est bien fondée car $\rhd_s$ commute avec $=_{\mathcal{E}}$ et, dans chaque classe d'équivalence modulo $=_{\mathcal{E}}$, la taille des termes est bornée.

Nous ne détaillons ensuite que la stabilité par substitution, les autres propriétés étant faciles à vérifier.

– Stabilité par substitution de $\simeq_E$. Supposons $\mathrm{Aliens}_E(\vec{t}) \simeq_{\mathcal{M}} \mathrm{Aliens}_E(\vec{u})$. Alors, il existe une "bijection" π entre $\mathrm{Aliens}_E(\vec{t})$ et $\mathrm{Aliens}_E(\vec{u})$ telle que, pour tout $a \in \mathrm{Aliens}_E(\vec{t})$, $a =_{\mathcal{E}} \pi(a)$. Soit θ une substitution et $a \in$

$\text{Aliens}_E(\vec{t})$. Si a est une variable et $a \in \text{dom}(\theta)$ alors $\pi(a) = a$ (car toutes les équations sont de la forme $\mathsf{f}\vec{l} = \mathsf{g}\vec{m}$) et $\text{aliens}_E(a\theta) = \text{aliens}_E(\pi(a)\theta)$. Autrement, $\text{aliens}_E(a\theta) = \{a\theta\} =_{\mathcal{E}} \{\pi(a)\theta\} = \text{aliens}_E(\pi(a)\theta)$. Ainsi, dans tous les cas, $\text{Aliens}_E(\vec{t}\theta) \simeq_{\mathcal{M}} \text{Aliens}_E(\vec{u})$.

– Stabilité par substitution de $>_E$. Supposons $\text{Aliens}_E(\vec{t}) >_{\mathcal{M}} \text{Aliens}_E(\vec{u})$. Alors, il existe M, $X \neq \emptyset$ et Y tels que $\text{Aliens}_E(\vec{t}) = M + X$, $\text{Aliens}_E(\vec{u}) = M + Y$ et, pour tout $y \in Y$, il existe $x \in X$ tel que $x > y$. Soit maintenant θ une substitution et $y \in Y$. Il existe $x \in X$ tel que $x > y$. Par définition de $>$, x n'est pas une variable et $x\theta \in \text{Aliens}(\vec{t}\theta)$. Comme les aliens de $y\theta$ sont des sous-termes stables de $y\theta$, nous avons $\{x\theta\} >_{\mathcal{M}} \text{aliens}_E(y\theta)$. Donc $\text{Aliens}_E(\vec{t}\theta) >_{\mathcal{M}} \text{Aliens}_E(\vec{u}\theta)$. ■

Les termes d'une classe d'équivalence sont de taille bornée si les classes d'équivalence sont de cardinalité finie, comme c'est le cas pour AC.

Nous vérifions maintenant que les conditions du Théorème 25 sont bien vérifiées pour l'addition modulo associativité et commutativité en prenant $\varphi_+ = 12$:

$$\begin{array}{rcl} x + 0 & \to & x \\ x + (\mathsf{s}y) & \to & \mathsf{s}(x + y) \\ x + y & = & y + x \\ (x + y) + z & = & x + (y + z) \end{array}$$

Pour la première règle, nous avons $x \in \text{CC}_+(x0)$ par (arg).

Pour la seconde règle, nous avons $x + y \in \text{CC}_+(x(\mathsf{s}y))$ par (rec) puisque $x(\mathsf{s}y) > xy$ car $\text{Aliens}(x(\mathsf{s}y)) = \{x, \mathsf{s}y\} >_{\mathcal{M}} \{x, y\} = \text{Aliens}(xy)$, et donc $\mathsf{s}(x + y) \in \text{CC}_+(x(\mathsf{s}y))$ par (undef).

Pour la commutativité, nous avons $xy \simeq yx$ puisque $\text{Aliens}(xy) = \{x, y\} = \text{Aliens}(yx)$, et nous avons $\{y, x\} \subseteq \text{CC}_+(xy)$ et $\{x, y\} \subseteq \text{CC}_+(yx)$ par (arg).

Pour l'associativité, nous avons $(x+y)z \simeq x(y+z)$ puisque $\text{Aliens}((x+y)z) = \{x, y, z\} = \text{Aliens}(x(y + z))$, et nous avons $x \in \text{CC}_+((x + y)z)$ par (arg) et (subterm-acc), et $y + z \in \text{CC}_+((x + y)z)$ par (rec) puisque $(x + y)z > yz$ car $\text{Aliens}((x + y)z) = \{x, y, z\} >_{\mathcal{M}} \{y, z\} = \text{Aliens}(yz)$. De même, nous avons $x + y \in \text{CC}_+(x(y + z))$ et $z \in \text{CC}_+(x(y + z))$.

4.5 Filtrage d'ordre supérieur

[Bla00] F. Blanqui. Termination and confluence of higher-order rewrite systems. In *Proceedings of the 11th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 1833, 2000.

[Bla07] F. Blanqui. Computability closure : Ten years later. In *Rewriting, Computation and Proof – Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, volume 4600 of *Lecture Notes in Computer Science*, 2007.

Jusqu'à maintenant, nous n'avons considéré que du filtrage syntaxique (modulo α-équivalence) ou du filtrage modulo une théorie équationnelle à caractère algébrique (pour simuler efficacement la réécriture modulo des équations). Or, comme nous l'avons vu en introduction, plusieurs approches considèrent du filtrage d'ordre supérieur, c'est-à-dire, modulo $\beta\eta$-équivalence. Pour voir pourquoi cela est nécessaire, considérons la règle de dérivation formelle suivante :

$$D(\lambda x \sin(Fx)) \;\to\; \lambda x DFx \times \cos(Fx)$$

Modulo α-équivalence seulement, cette règle ne peut s'appliquer ni à $D(\sin)$ ni à $D(\lambda x \sin(x))$. Par contre, elle peut s'appliquer à la deuxième expression si l'on considère du filtrage modulo β-équivalence, car $x \leftarrow_\beta (\lambda xx)x$, et à la première expression si l'on considère du filtrage modulo $\beta\eta$-équivalence, car $\sin \leftarrow_\eta \lambda x \sin(x)$.

Le problème du filtrage modulo $\beta\eta$ a été montré décidable assez récemment seulement [Sti09], alors que le problème de l'unification modulo $\beta\eta$ est indécidable [Hue76], et que celui du filtrage modulo β seulement est également indécidable [Loa03]. Par contre, sa complexité est non élémentaire [Sta79]. Il existe cependant des fragments importants dont la complexité est linéaire, comme les termes en forme β-normal η-longue dont les variables libres sont appliquées à des termes η-équivalents à des variables liées distinctes, ou motifs de Miller [Mil91, Qia93].

Or, comme l'a remarqué Miller [Mil91], si l est un tel terme et $l\sigma =_{\beta\eta} t$ avec t et σ en forme β-normal η-longue, alors $l\sigma =_{\beta_0\eta} t$, où $\to_{\beta_0}$ est la restriction de $\to_\beta$ aux redex de la forme $(\lambda xt)x$. En effet, comme $\to_{\beta\eta}$ est confluent et $\to^*_{\beta\eta} \subseteq \to^*_\beta \to^*_\eta$ [Bar84, Tak95], on a $l\sigma \to^*_{\beta_0} =_\eta t$.

Ainsi, une étape de réécriture dans un CRS [KvOvR93] ou un HRS [MN98] est incluse dans la relation $\to_{\mathcal{R},\beta_0\eta} \to^*_\beta$ où $\to_{\mathcal{R},\beta_0\eta}$ est définie de la manière suivante :

Definition 28 (Réécriture β_0-normalisée avec filtrage modulo $\beta_0\eta$) Étant donné un ensemble $\mathcal{R}$ de règles de réécriture, $t \to_{\mathcal{R},\beta_0\eta} u$ s'il existe $p \in \mathrm{Pos}(t)$, $l \to r \in \mathcal{R}$ et σ tels que $t|_p$ est en forme β_0-normale, $t|_p =_{\beta_0\eta} l\sigma$ et $u = t[r\sigma]_p$.

Dans un CRS, un terme est soit une variable x, soit une abstraction λxt, soit enfin l'application d'un symbole de fonction f à des termes $\vec{t}$. Un terme est donc nécessairement en forme β-normal. Seules les règles de réécriture peuvent contenir des termes de la forme $x\vec{t}$, mais toute étape de réécriture est suivie d'une β-normalisation.

Dans un HRS, les termes sont des λ-termes en forme β-normale η-longue, et une étape de réécriture est suivie par une mise en forme β-normale η-longue. En toute rigueur, si la réécriture ne préserve pas les formes η-longues, il faudrait montrer la terminaison de $\to \cup \to_{\overline{\eta}}$ où $\to_{\overline{\eta}}$ est la relation d'η-*expansion*, c'est-à-dire, $\leftarrow_\eta$ restreinte aux termes qui ne sont pas de la forme λxt et aux contextes qui ne sont pas de la forme $C[[]u]$ [CK96].

Nous montrons maintenant [9] que la clôture de calculabilité peut être également étendue pour traiter ce type de réécriture.

Definition 29 (Calculabilité modulo $\beta_0\eta$) Étant donné un ensemble $\mathcal{R}$ de règles de réécriture de la forme $\mathsf{f}\vec{l} \to r$ avec $\mathsf{f}\vec{l}$ η-équivalent à un motif de Miller, un terme est *neutre* s'il est de la forme $x\vec{v}$, $(\lambda xt)u\vec{v}$ ou $\mathsf{f}\vec{v}$ avec $\mathsf{f} \in \mathcal{D}(\mathcal{R}) - \mathcal{M}(\mathcal{R})$ et $|\vec{v}| \geq \alpha_\mathsf{f} = sup\{|\vec{l}| \mid \mathsf{f}\vec{l} \to r \in \mathcal{R}\}$. Soit $\mathbf{Red}_\mathcal{R}^{\beta_0\eta}$ l'ensemble des $P \subseteq \mathcal{L}$ tels que :

(R0) $P \subseteq \mathcal{SN}(\to)$ où $\to = \to_\beta \cup \to_{\mathcal{R},\beta_0\eta}$;
(R1) $\mathcal{X} \subseteq P$;
(R2) P est stable par $\to$;
(R3) si t est un terme neutre et $\to(t) \subseteq P$, alors $t \in P$.

On vérifie que $\mathbf{Red}_\mathcal{R}^{\beta_0\eta}$ a les propriétés requises :

Lemma 30 $\mathbf{Red}_\mathcal{R}^{\beta_0\eta}$ est stable par a et intersection non vide et admet $\mathcal{SN}(\to)$ comme plus grand élément.

Proof. La preuve est similaire à celle du Lemme 3. La stabilité par intersection non vide et le fait que $\mathcal{SN}(\to) \in \mathbf{Red}_\mathcal{R}^{\beta_0\eta}$ se montrent aisément. Pour la stabilité par a, il n'y pas de changement pour (R0), (R1) et (R2). Nous détaillons maintenant (R3). Soient $P, Q \in \mathbf{Red}_\mathcal{R}^\mathcal{E}$ et t neutre tel que $\to(t) \subseteq a(P,Q)$, et $u \in P$. Nous montrons que $tu \in Q$ par induction bien fondée sur u avec $\to$ comme relation bien fondée ($u \in \mathcal{SN}$ par (R0)). Comme tu est neutre, par (R3), il suffit de montrer que $\to(tu) \subseteq Q$. Soit $v \in \to(tu)$. Si $tu \to_\beta v$, nous avons déjà vu que $v \in Q$. Supposons maintenant que $tu \to_{\mathcal{R},\beta_0\eta} v$. Si la réduction a lieu dans t ou dans u alors, comme pour $\to_\beta$, $v \in Q$. Sinon, il existe $\mathsf{f}\vec{l} \to r \in \mathcal{R}$ et σ tels que $tu =_{\beta_0\eta} \mathsf{f}\vec{l}\sigma$ et $v = r\sigma$. Comme t est neutre, soit :

– $t = x\vec{t}$. Par confluence de $\to_{\beta_0\eta}$, ce n'est pas possible.

9. Cette section corrige certaines lacunes de [Bla07].

– $t = \mathsf{g}\vec{m}$ avec $\alpha_{\mathsf{g}} \leq |\vec{m}|$. Par confluence de $\to_{\beta_0\eta}$, $\mathsf{f} = \mathsf{g}$ et $\alpha_{\mathsf{f}} \leq |\vec{m}| < |\vec{m}u| = |\vec{l}| \leq \alpha_{\mathsf{f}}$, ce qui n'est pas possible.
– $t = (\lambda x a)\vec{bt}$. Par confluence de $\to_{\beta_0\eta}$, il existe $t' \in {\to}(t)$ tel que $t'u \to_{\mathcal{R},\beta_0\eta} v$. Donc, $v \in Q$. ■

FIGURE 7 – Opérations de clôture VI

(subterm-abs)	si $\lambda x t \in \mathrm{CC}^X_{\mathsf{f}}(\vec{l})$ et $x \notin \mathrm{FV}(\vec{l})$, alors $t \in \mathrm{CC}^{X\cup\{x\}}_{\mathsf{f}}(\vec{l})$
(subterm-app)	si $tu \in \mathrm{CC}^X_{\mathsf{f}}(\vec{l})$ et $u{\downarrow_\eta} \in X$, alors $t \in \mathrm{CC}^X_{\mathsf{f}}(\vec{l})$

Maintenant, pour étendre le Théorème 9 à la réécriture β_0-normalisée par filtrage modulo $\beta_0\eta$, il suffit de remarquer deux propriétés importante. D'une part, les règles de déstructuration d'un motif de Miller (Figure 7) préservent la calculabilité (vérification laissée au lecteur). D'autre part, si t est calculable et $t =_{\beta_0\eta} l\sigma$ où l est η-équivalent à un motif de Miller, alors $l\sigma$ est calculable, à condition que $\mathcal{R}$ soit *compatible* avec $\to_{\beta_0\eta}$:

Definition 31 Un ensemble de règles de réécriture $\mathcal{R}$ est *compatible* avec $\to_{\beta_0\eta}$ si ${\leftarrow_{\beta_0\eta}}{\to_{\mathcal{R},\beta_0\eta}} \subseteq {\to^+}{=_{\beta_0\eta}}$.

Lemma 32 Nous avons ${\leftarrow_\eta}{\to_{\mathcal{R},\beta_0\eta}} \subseteq {\to_{\mathcal{R},\beta_0\eta}}{\leftarrow^*_{\beta_0\eta}}$ si :

- toute règle est de la forme $\mathsf{f}\vec{l} \to r$ avec $\mathsf{f}\vec{l}$ η-équivalent à un motif de Miller ;
- si $lx \to r \in \mathcal{R}$ et $x \notin \mathrm{FV}(l)$, alors :
  – si $r = sx$ avec $x \notin \mathrm{FV}(s)$, alors $l \to s \in \mathcal{R}$,
  – sinon $l \to \lambda x r \in \mathcal{R}$.

Proof. Supposons que $t \to_\eta u$ à la position p et $t \to_{\mathcal{R},\beta_0\eta} v$ à la position q. Si $p\#q$ alors $u \to_{\mathcal{R},\beta_0\eta}{\leftarrow_\eta} v$. Si $q \leq p$, alors $u \to_{\mathcal{R},\beta_0\eta} v$. Reste le cas où $p < q$. Par définition de $\to_\eta$, il existe s tel que $t|_p = \lambda x s x$ et $x \notin \mathrm{FV}(s)$. Comme les règles sont de la forme $\mathsf{f}\vec{l} \to r$, $p01 \not\leq q$. Si $p00 \leq q$, alors il existe s' tel que $s \to_{\mathcal{R},\beta_0\eta} s'$ et $v = t[\lambda x s' x]_p$. Dans ce cas, $u = t[s]_p \to_{\mathcal{R},\beta_0\eta} t[s']_p \leftarrow_\eta v$. Reste alors le cas où $q = p0$. Par définition de $\to_{\mathcal{R},\beta_0\eta}$, il existe $\mathsf{f}\vec{l} \to r \in \mathcal{R}$ et σ tels que sx est en forme β_0-normale, $sx =_{\beta_0\eta} \mathsf{f}\vec{l\sigma}$ et $v = t[\lambda x r\sigma]_p$. Comme $\to_{\beta_0\eta}$ termine et conflue, il existe $\vec{m}$ tel que $sx \to^*_{\beta_0\eta} \mathsf{f}\vec{m} \leftarrow^*_{\beta_0\eta} \mathsf{f}\vec{l\sigma}$. Comme ${\to^*_{\beta_0\eta}} \subseteq {\to^*_{\beta_0}}{\to^*_\eta}$, $sx \to^*_{\beta_0}{\to^*_\eta} \mathsf{f}\vec{m} \leftarrow^*_{\beta_0\eta} \mathsf{f}\vec{l\sigma}$. Comme sx est en forme β_0-normale, $sx \to^*_\eta \mathsf{f}\vec{m}$. Il existe donc $\vec{n}$ tel que $\vec{m} = \vec{n}x$ et $s \to^*_\eta \mathsf{f}\vec{n}$, et $\vec{k}$ et k tels que $\vec{l} = \vec{k}k$ et $k\sigma \to^*_{\beta_0\eta} x$. Comme k est η-équivalent à un motif de Miller, k est nécessairement η-équivalent à une variable y. Comme l'ensemble des variables libres d'un terme est invariant par $=_{\beta_0\eta}$, nous avons $y \notin \mathrm{FV}(\vec{k})$ car $x \notin \mathrm{FV}(s)$. Donc, $s =_{\beta_0\eta} \mathsf{f}\vec{k}\sigma'$ où σ' est la restriction de σ à $\mathrm{dom}(\sigma) - \{y\}$. Si $r = a$ avec $y \notin \mathrm{FV}(a)$, alors $s \to_{\mathcal{R},\beta_0\eta} a\sigma' \leftarrow_\eta \lambda x a\sigma' x \leftarrow^*_{\beta_0\eta} \lambda x r\sigma$. Sinon, $s \to_{\mathcal{R},\beta_0\eta} \lambda x r\sigma$. Ainsi, $u = t[s]_p \to_{\mathcal{R},\beta_0\eta}{\leftarrow^*_{\beta_0\eta}} v$. ■

Par exemple, avec $\mathcal{R} = \{\mathsf{f}x \to x\}$, la relation $\to_{\mathcal{R},\beta_0\eta}$ ne commute pas par $\to_\eta$. À cause de la paire critique $\lambda xx \leftarrow_{\mathcal{R}} \lambda x\mathsf{f}x \to_\eta \mathsf{f}$, il faut rajouter la règle $\mathsf{f} \to \lambda xx$ pour commuter. Nous avons également commutation en ne prenant que $\mathcal{R} = \{\mathsf{f} \to \lambda xx\}$.

En fait, la troisième condition (cas $r = sx$ avec $x \notin \mathrm{FV}(s)$) n'est pas nécessaire mais permet de ne pas introduire de règles de réécriture dont le membre droit n'est pas en forme η-normale. En éliminant cette condition, on obtient la propriété plus forte que $\to_{\mathcal{R},\beta_0\eta}$ commute avec $\to_\eta : \leftarrow_\eta \to_{\mathcal{R},\beta_0\eta} \subseteq \to_{\mathcal{R},\beta_0\eta} \leftarrow_\eta^=$ et donc que $\to_{\mathcal{R},\beta_0\eta}$ commute avec $\to_\eta^*$. On obtiendrait également cette propriété en imposant que σ soit en forme β_0-normale dans la définition de la réécriture, car alors $\leftarrow_\eta \to_{\mathcal{R},\beta_0\eta} \subseteq \to_{\mathcal{R},\beta_0\eta} \leftarrow_\eta^*$.

Lemma 33 Nous avons $\leftarrow_{\beta_0} \to_{\mathcal{R},\beta_0\eta} \subseteq \to_{\mathcal{R},\beta_0\eta} \leftarrow_{\beta_0}^=$ si :

- toute règle est de la forme $\mathsf{f}\vec{l} \to r$;
- si $l \to r \in \mathcal{R}$, $l : T \Rightarrow U$, r n'est pas une abstraction et $x \notin \mathrm{FV}(l)$, alors $lx \to rx \in \mathcal{R}$;
- si $l \to \lambda xr \in \mathcal{R}$ et $x \notin \mathrm{FV}(l)$, alors $lx \to r \in \mathcal{R}$.

Proof. Supposons que $t \to_{\beta_0} u$ à la position p et $t \to_{\mathcal{R},\beta_0\eta} v$ à la position q. Si $p\#q$ alors $u \to_{\mathcal{R},\beta_0\eta} \leftarrow_{\beta_0} v$. Si $q \leq p$, alors $u \to_{\mathcal{R},\beta_0\eta} v$. Par définition de $\to_{\beta_0}$, il existe s tel que $t|_p = (\lambda xs)x$ et $u = t[s]_p$. Si $p0 < q$ alors il existe s' tel que $s \to_{\mathcal{R},\beta_0\eta} s'$ et $u \to_{\mathcal{R},\beta_0\eta} t[s']_p \leftarrow_{\beta_0} t[(\lambda xs')x]_p = v$. Reste le cas où $q = p0$. Alors, il existe $\mathsf{f}\vec{l} \to r \in \mathcal{R}$ et σ tels que $\lambda xs =_{\beta_0\eta} \mathsf{f}\vec{l}\sigma$ et $v = t[r\sigma x]_p$. Par confluence de $\to_{\beta_0\eta}$, il existe a tel que $s =_{\beta_0\eta} ax$, $x \notin \mathrm{FV}(a)$ et $a =_{\beta_0\eta} \mathsf{f}\vec{l}\sigma$. Ainsi, $s =_{\beta_0\eta} \mathsf{f}\vec{l}\sigma x$. Comme $\lambda xs =_{\beta_0\eta} \mathsf{f}\vec{l}\sigma$, nous avons $\mathsf{f}\vec{l} : T \Rightarrow U$. Supposons que r ne soit pas une abstraction. Par ailleurs, sans perte de généralité, nous pouvons supposer que $x \notin \mathrm{FV}(\vec{l}) \cup \mathrm{dom}(\sigma)$. Alors, $\mathsf{f}\vec{l}x \to rx \in \mathcal{R}$, $s \to_{\mathcal{R},\beta_0\eta} r\sigma x$ et $u \to_{\mathcal{R},\beta_0\eta} v$. Si maintenant r est une abstraction λxw, alors $s \to_{\mathcal{R},\beta_0\eta} w\sigma \leftarrow_{\beta_0} r\sigma x$ et $u \to_{\mathcal{R},\beta_0\eta} \leftarrow_{\beta_0} v$. ■

Ainsi, le système $\mathcal{R} = \{\mathsf{f} \to \lambda xx\}$ commute avec $\to_\eta$ mais ne commute pas avec $\to_{\beta_0}$. Pour cela, il faut rajouter la règle $\mathsf{f}x \to x$. Par contre, le système $\mathcal{R} = \{\mathsf{f}x \to x, \mathsf{f} \to \lambda xx\}$ commute bien avec $\to_{\beta_0\eta}$.

La troisième condition n'est pas indispensable (si on accepte que r soit une abstraction dans la deuxième condition) mais évite de rajouter une règle dont le membre droit serait un β_0-redex.

Ainsi, un ensemble fini $\mathcal{S}$ de règles de la forme $\mathsf{f}\vec{l} \to r$ avec $\mathsf{f}\vec{l}$ η-équivalent à un motif de Miller peut toujours être complété en un nombre fini d'étape en un ensemble fini $\mathcal{R} \supseteq \mathcal{S}$ de règles de la forme $\mathsf{f}\vec{l} \to r$ avec $\mathsf{f}\vec{l}$ η-équivalent à un motif de Miller, de façon à ce que $\leftarrow_{\beta_0\eta} \to_{\mathcal{R},\beta_0\eta} \subseteq \to^+ =_{\beta_0\eta}$.

Nous pouvons maintenant montrer que, si t est calculable et $t =_{\beta_0\eta} l\sigma$ où l est η-équivalent à un motif de Miller, alors $l\sigma$ est calculable. Pour commencer, notons que :

Lemma 34 Un terme de type de base t est calculable si tous ses réduits et, lorsque $t = \mathsf{f}\vec{t}$ avec $\mathsf{f} \in \mathcal{M}(\mathcal{R})$, tous ses arguments accessibles, sont calculables.

Proof. Si t est neutre, alors cela est une conséquence de (R3). Sinon, comme t est de type de base, t est nécessairement de la forme $\mathsf{f}\vec{t}$ avec $\mathsf{f} \notin \mathcal{D}(\mathcal{R})$ ou $\mathsf{f} \in \mathcal{M}(\mathcal{R})$. Nous avons déjà vu que les symboles non définis sont calculables. Enfin, si $\mathsf{f} \in \mathcal{D}(\mathcal{R}) \cap \mathcal{M}(\mathcal{R})$ alors, d'après le Lemme 19, $\mathsf{f}\vec{t}$ est calculable puisque tous ses réduits et arguments accessibles sont calculables. ■

Lemma 35 Soit $\mathcal{R}$ un ensemble de règles de réécriture compatible avec $\to_{\beta_0\eta}$. Si t est calculable et $t =_{\beta_0\eta} l\sigma$ où l est η-équivalent à un motif de Miller, alors $l\sigma$ est calculable.

Proof. Soit $u = l\sigma$. Remarquons tout d'abord qu'il suffit de montrer le lemme dans le cas où t est de type de base. En effet, si $u : \vec{V} \Rightarrow \mathsf{B}$, alors u est calculable si, pour tout $\vec{v} : \vec{V}$ calculables, $u\vec{v} : \mathsf{B}$ est calculable. Or, si $t =_{\beta_0\eta} u$, alors $t\vec{v} =_{\beta_0\eta} u\vec{v}$.

Dans le cas où t est de type de base, nous procédons par induction sur le rang de t, t ordonné par $\to$, et le nombre d'étape de réécritures entre t et u. Il suffit alors de montrer que, si t est calculable et $t \leftrightarrow_{\beta_0\eta} u$, alors u est calculable. Par ailleurs, d'après le Lemme 34, u est calculable si tous ses réduits v et, lorsque $u = \mathsf{f}\vec{u}$ avec $\mathsf{f} \in \mathcal{M}(\mathcal{R})$, tous ses arguments accessibles, sont calculables.

Commençons par montrer que, si $u = \mathsf{f}\vec{u}$ avec $\mathsf{f} \in \mathcal{M}(\mathcal{R})$, alors tous les arguments u_i avec $i \in \mathrm{Acc}(\mathsf{f})$ sont calculables. Comme $u = \mathsf{f}\vec{u}$ est de type de base, soit $t \to_{\beta_0} u$, soit $t = \mathsf{f}\vec{t}$ et $\vec{t}(\leftrightarrow_{\beta_0\eta})_{\mathrm{prod}}\vec{u}$. Dans le premier cas, u_i est calculable car u est calculable. Dans le second cas, $u_i : \vec{V} \Rightarrow \mathsf{B}$ est calculable si, pour tout $\vec{v} : \vec{V}$ calculables, $u_i\vec{v}$ est calculable. Comme $t_i\vec{v} =_{\beta_0\eta} u_i\vec{v}$ et $t_i\vec{v}$ est de rang plus petit que $\mathsf{f}\vec{t}$, nous avons $u_i\vec{v}$ calculable par hypothèse d'induction.

Montrons maintenant que tous les réduits v de u à la position q sont calculables.

- $t \to_{\beta_0} u \to v$. Par (R2), v est calculable.
- $t \to_\eta u \to_\beta v$. Une analyse [10] des différents cas montre que $t \to_\beta^+ t' \to_\eta^* v$ et nous pouvons conclure par induction sur t'. Supposons que t se réduise en u à la position p. Si $p\#q$ alors $t \to_\beta t' \to_\eta v$. Si $p \leq q$ alors il existe a et a' tels que $t|_p = \lambda x a x$, $x \notin \mathrm{FV}(a)$, $u = t[a]_p$, $a \to_\beta a'$ et $v = t[a']_p$. Dans ce cas, $t \to_\beta t' = t[\lambda x a' x]_p \to_\eta v$. Autrement $q < p$ et il existe a et b tel que $t|_q = (\lambda x a)b$. Si $q1 \leq p$ alors il existe b' tel que $b \to_\eta b'$, $u = t[(\lambda x a)b']_q$ et $v = t[a_x^{b'}]_q$. Dans ce cas, $t \to_\beta t' = t[a_x^b] \to_\eta^* v$. Si $q0 = p$ alors il existe a' tel que $a = \lambda x a'$, $u = t[(\lambda x a')b]_q$ et $v = t[{a'}_x^b]_q$. Dans ce cas, $t \to_\beta t' = t[(\lambda x a')b]_q \to_\beta v$. Si enfin $q0 < p$ alors il existe a' tel que $a \to_\eta a'$, $u = (\lambda x a')b$ et $v = {a'}_x^b$. Dans ce cas, $t \to_\beta t' = t[a_x^b] \to_\eta v$.
- $t \to_\eta u \to_{\mathcal{R},\beta_0\eta} v$. Une analyse des différents cas montre que $t \to_{\mathcal{R},\beta_0\eta} t' \to_\eta^= v$ et nous pouvons conclure par induction sur t'. Supposons que t se réduise en u à la position p. Si $p\#q$ alors $t \to_{\mathcal{R},\beta_0\eta} t' \to_\eta v$. Si $q \leq p$ alors $t \to_{\mathcal{R},\beta_0\eta} v$. Si enfin $p < q$ alors il existe a et a' tels que

10. Ce cas pourrait être simplifié et traité par (R2) aussi en incluant $\to_\eta$ dans $\to$. Mais il faut alors vérifier le Lemme 30. La présente preuve montre cependant que cela n'est pas nécessaire.

$t|_p = \lambda xax$, $x \notin \mathrm{FV}(a)$, $u = t[a]_p$, $a \to_{\mathcal{R},\beta_0\eta} a'$ et $v = t[a']_p$. Dans ce cas, $t \to_{\mathcal{R},\beta_0\eta} t' = t[\lambda xa'x]_p \to_\eta v$.
- $t \leftarrow_{\beta_0\eta} u \to_{\mathcal{R},\beta_0\eta} v$. Par hypothèse, $t \to^+ t' =_{\beta_0\eta} v$ et nous pouvons conclure par induction sur t'.
- $t \leftarrow_\eta u \to_\beta v$. Une analyse des différents cas montre que, soit $v = t$, soit il existe t' tel que $t \to_\beta t' \leftarrow_\eta^* v$ et nous pouvons conclure par induction sur t'. Supposons que u se réduise en t à la position p. Nous avons nécessairement $p \neq q$. Si $p \# q$ alors $t \to_\beta t' \leftarrow_\eta v$. Si $p = q0$ alors il existe a et b tels que $u|_q = (\lambda xax)b$, $x \notin \mathrm{FV}(a)$, $t = ab$ et $v = ab = t$. Si $p > q0$ alors il existe a et a' tels que $u|_q = (\lambda xa)b$, $a \to_\eta a'$, $t = u[(\lambda xa')b]_q$ et $v = u[a^b_x]$. Dans ce cas, $t \to_\beta t' = u[{a'}^b_x]_q \leftarrow_\eta v$. Si $p \geq q1$ alors il existe a, b et b' tels que $u|_q = (\lambda xa)b$, $b \to_\eta b'$, $t = u[(\lambda xa)b']_q$ et $v = u[a^b_x]$. Dans ce cas, $t \to_\beta t' = u[a^{b'}_x]_q \leftarrow_\eta^* v$. Si $q = p0$ alors il existe a tel que $u|_p = \lambda x(\lambda xa)x$, $t = u[\lambda xa]_p$ et $v = u[\lambda xa] = t$. Si enfin $q > p0$ alors il existe a et a' tels que $u|_p = \lambda xax$, $x \notin \mathrm{FV}(a)$, $a \to_\beta a'$, $t = u[a]_p$ et $v = u[\lambda xa'x]_p$. Dans ce cas, $t \to_\beta t' = u[a']_p \leftarrow_\eta v$.
- $t \leftarrow_{\beta_0} u \to_\beta v$. Pour traiter ce cas, il convient de remarquer d'une part que si $t =_{\beta_0\eta} u$, alors $t \to^*_{\beta_0} =_\eta u' \leftarrow^*_{\beta_0} u$, et d'autre part que, comme l est η-équivalent à un motif de Miller, toute β-réduction d'un terme entre u et u' est soit une β_0-réduction d'un descendant d'une position p telles que l_p soit de la forme $x\vec{y}$ avec $p0^{|\vec{y}|}$ la position d'une variable libre de l et $\vec{y}$ des variables liées distinctes, soit une β-réduction [11] d'un descendant de $x\sigma$. Une analyse des différents cas montre que, soit $v = t$, soit il existe t' tel que $t \to_\beta t' \leftarrow^*_{\beta_0} v$ et nous pouvons conclure par induction sur t'. Supposons que t se réduise en u à la position p. Nous venons de voir que le cas $q < p$ n'est pas possible. Si $p \# q$ alors $t \to_\beta t' \leftarrow_{\beta_0} v$. Si $p = q$ alors $v = t$. Si enfin $p < q$ alors il existe a et a' tels que $u|_p = (\lambda xa)x$, $t = u[a]_p$, $a \to_\beta a'$ et $v = u[(\lambda xa')x]_p$. Dans ce cas, $t \to_\beta t' = u[a']_p \leftarrow_{\beta_0} v$. ■

On peut maintenant étendre le Théorème 9 à la réécriture β_0-normalisée avec filtrage modulo $\beta_0\eta$ (et donc aux CRS et HRS) :

Theorem 36 Soit $\mathcal{R}$ un ensemble de règles de réécriture compatible avec $\to_{\beta_0\eta}$. La relation $\to_\beta \cup \to_\mathcal{R}$ termine s'il existe $I : \mathcal{B} \to \mathbf{Red}_\mathcal{R}$ et un $(\mathcal{F}, \varphi)$-pré-ordre $\geq$ tels que $>$ est stable par substitution, $>^\varphi \cup \to_{\mathrm{prod}}$ est bien fondé et :
- tout symbole $\mathsf{f} \in \mathcal{F} - \mathcal{D}(\mathcal{R})$ est calculable ;
- toute règle est de la forme $\mathsf{f}\vec{l} \to r \in \mathcal{R}$ avec $\mathsf{f}\vec{l}$ η-équivalent à un motif de Miller et $r \in \mathrm{CC}_\mathsf{f}(\vec{l})$, où CC est la plus petite $\mathcal{F}$-famille close par les opérations I à VI.

Proof. Comme dans le Théorème 9, il suffit de montrer que, pour tout $(\mathsf{f}, \vec{t}) \in \Sigma$ tel que $\mathsf{f} \in \mathcal{D}(\mathcal{R})$, nous avons $\mathsf{f}\vec{t}$ calculable. Pour cela, nous procédons par induction bien fondée sur $(\mathsf{f}, \vec{t})$ ordonné par $>^\varphi \cup \to_{\mathrm{prod}}$. Comme $\mathsf{f}\vec{t}$ est

11. Ce cas pourrait être éliminé en imposant dans la définition de $\to_{\mathcal{R},\beta_0\eta}$ que σ soit en forme β-normale. La présente preuve montre cependant que cela n'est pas nécessaire.

neutre, il suffit de montrer que tous ses réduits t sont calculables. Il y a deux cas possibles :

- Il existe $\vec{u}$ tel que $t = \mathsf{f}\vec{u}$ et $\vec{t} \to_{\text{prod}} \vec{u}$. Par (R2), $\vec{u}$ est calculable. Donc, par hypothèse d'induction, $\mathsf{f}\vec{u}$ est calculable.
- Il existe $\vec{u}$, $\vec{w}$, $\mathsf{f}\vec{l} \to r \in \mathcal{R}$ et σ tels que $\vec{t} = \vec{u}\vec{w}$, $\mathsf{f}\vec{u} =_{\beta_0\eta} \mathsf{f}\vec{l}\sigma$ et $t = r\sigma\vec{w}$. Par hypothèse, $r \in \mathrm{CC}_{\mathsf{f}}(\vec{l})$. Comme $>$ est stable par substitution, CC est stable par substitution et $r\sigma \in \mathrm{CC}_{\mathsf{f}}(\vec{l}\sigma)$. Il suffit alors de montrer que, pour tout $u \in \mathrm{CC}_{\mathsf{f}}(\vec{l}\sigma)$, u est calculable, par induction sur $\mathrm{CC}_{\mathsf{f}}(\vec{l}\sigma)$. Puisque $\vec{u}$ est calculable et $\vec{u} =_{\beta_0\eta} \vec{l}\sigma$, le cas (arg) est une conséquence du Lemme 35. Les cas (app) et (red) sont immédiats. Le cas (rec) est résolu par hypothèse d'induction. ■

Nous venons de voir que la preuve de terminaison reposait en grande partie sur des propriétés de commutation locale entre la relation de réduction $\to = \to_\beta \cup \to_{\mathcal{R},\beta_0\eta}$ et la relation $\leftrightarrow_{\beta_0\eta}$. De telles conditions sont bien connues en réécriture du premier ordre : ce sont la notion de compatibilité de Peterson et Stickel [PS81], la notion de E-commutation locale de Jouannaud et Muñoz [JM84] et, plus généralement, la notion de cohérence locale modulo E de Jouannaud et Kirchner [JK86]. Nous montrons ci-après que $\to_{\mathcal{R},\beta_0\eta}$ $\beta_0\eta$-commute localement (et donc est localement cohérente modulo $\beta_0\eta$) :

Lemma 37 Si $\mathcal{R}$ est un ensemble de règles de réécriture de la forme $\mathsf{f}\vec{l} \to r$ avec $\mathsf{f}\vec{l}$ η-équivalent à un motif de Miller et vérifiant les conditions des Lemmes 32 et 33, alors $\leftrightarrow_{\beta_0\eta} \to_{\mathcal{R},\beta_0\eta} \subseteq \to^+_{\mathcal{R},\beta_0\eta} =_{\beta_0\eta}$.

Proof. Les conditions des Lemmes 32 et 33 assurent que $\mathcal{R}$ est compatible avec $\to_{\beta_0\eta}$, *i.e.* $\leftarrow_{\beta_0\eta} \to_{\mathcal{R},\beta_0\eta} \subseteq \to^+_{\mathcal{R},\beta_0\eta} =_{\beta_0\eta}$. Dans la preuve du Lemme 35, nous avons vu que $\to_\eta \to_{\mathcal{R},\beta_0\eta} \subseteq \to_{\mathcal{R},\beta_0\eta} \to^=_\eta$. Reste le cas où $t \to_{\beta_0} u \to_{\mathcal{R},\beta_0\eta} v$. La preuve est similaire. Supposons que t se réduise en u à la position u, et que u se réduise en v à la position q. Si $p \# q$ alors $t \to_{\mathcal{R},\beta_0\eta} \to_{\beta_0} v$. Si $p \leq q$ alors $t \to_{\mathcal{R},\beta_0\eta} v$. Si enfin $q < p$, alors il existe a et a' tels que $t|_p = (\lambda x a)x$, $u = t[a]_p$, $a \to_{\mathcal{R},\beta_0\eta} a'$ et $v = t[a']_p$. Dans ce cas, $t \to_{\mathcal{R},\beta_0\eta} t[(\lambda x a')x]_p \to_{\beta_0} v$. ■

Avec $\to_\beta$ par contre, cette propriété n'est pas tout à fait vérifiée (il se peut que $v = t$) et nécessite que u soit le β_0-réduit d'une instance d'un motif de Miller. Autrement, dans le cas $t \leftarrow_{\beta_0} u \to_\beta v$, la β-réduction de u à v pourrait transformer le β_0-redex de u en un β-redex (prendre $u = (\lambda x(\lambda x a)x)b$).

4.6 Ordre récursif sur les chemins

[Bla06b] F. Blanqui. (HO)RPO revisited. Technical Report 5972, INRIA, 2006.

[BJR07] F. Blanqui, J.-P. Jouannaud, and A. Rubio. HORPO with computability closure : A reconstruction. In *Proceedings of the 14th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Lecture Notes in Computer Science 4790, 2007.

L'ordre récursif sur les chemins (RPO) est un ordre de simplification sur les termes du premier ordre (*i.e.* contenant sous-terme) inventé par Dershowitz [Der79, Der82], simplifiant et généralisant un ordre introduit par Plaisted [Pla78] en étendant aux termes un ordre sur les symboles de fonction où chaque symbole f est équipé d'un statut $\text{stat}_\mathsf{f} \in \{\text{lex}, \text{mul}\}$ (Figure 8). Il a été étendu au cas lexicographique et à un ordre sur les termes plutôt que sur les symboles (ordre sémantique sur les chemins) par Kamin et Lévy [KL80]. Il a ensuite été étendu à l'ordre supérieur par Jouannaud et Rubio [JR99, JR07], puis au calcul des constructions par Walukiewicz [WC03]. Alors que ces deux derniers travaux utilisent la même technique de preuve que pour la clôture de calculabilité, aucune comparaison précise n'a été faite jusqu'en 2006. La clôture de calculabilité est même utilisée dans [JR99] pour obtenir un ordre plus puissant. Nous montrons maintenant quelle relation précise il existe entre l'ordre récursif sur les chemins et la notion de clôture de calculabilité.

FIGURE 8 – Ordre récursif sur les chemins (RPO)

(rpo1)	si $t_i \geq_{\text{rpo}} u$, alors $\mathsf{f}\vec{t} >_{\text{rpo}} u$
(rpo2)	si $\mathsf{f} >_\mathcal{F} \mathsf{g}$ et $\mathsf{f}\vec{t} >_{\text{rpo}} \vec{u}$, alors $\mathsf{f}\vec{t} >_{\text{rpo}} \mathsf{g}\vec{u}$
(rpo3)	si $\mathsf{f} \simeq_\mathcal{F} \mathsf{g}$, $\vec{t}(>_{\text{rpo}})_{\text{stat}_\mathsf{f}}\vec{u}$ et $\mathsf{f}\vec{t} >_{\text{rpo}} \vec{u}$, alors $\mathsf{f}\vec{t} >_{\text{rpo}} \mathsf{g}\vec{u}$

Pour commencer, il suffit de remarquer que la clôture de calculabilité définie une relation similaire à RPO. Afin de rendre cela évident, nous réécrivons dans la Figure 9 les règles définissant la clôture en utilisant une notation similaire à celle de RPO : $t \in \text{CC}^X_\mathsf{f}(\vec{l})$ est remplacé par $\mathsf{f}\vec{l} >^X_{\text{cc}} t$.

Rappelons que, dans cette définition, $>^\varphi$ est un paramètre. Mais $\to$ aussi peut être vu comme un paramètre. Si, pour simplifier, nous oublions les types et le système de filtrage des arguments, alors (rpo2) et (rpo3) correspondent à la règle (rec) avec, comme $\mathcal{F}$-pré-ordre l'extension lexicographique ou multi-ensemble de $>_{\text{cc}}$ lui-même. En ce sens, nous avons affaire à une définition récursive et celle-ci admet une solution si la fonction qui associe $>_{\text{cc}}$ à $>$ et $\to$ est bien monotone [Tar55], ce qui est immédiat à vérifier.

FIGURE 9 – Clôture de calculabilité

(arg)	$\mathsf{f}\vec{l} >_{\mathrm{cc}} \vec{l}$
(app)	si $\mathsf{f}\vec{l} >_{\mathrm{cc}} t : U \Rightarrow V$ et $\mathsf{f}\vec{l} >_{\mathrm{cc}} u : U$, alors $\mathsf{f}\vec{l} >_{\mathrm{cc}} tu$
(red)	si $\mathsf{f}\vec{l} >_{\mathrm{cc}} t$ et $t \to u$, alors $\mathsf{f}\vec{l} >_{\mathrm{cc}} u$
(rec)	si $(\mathsf{f}, \vec{l}) >^{\varphi} (\mathsf{g}, \vec{m})$ et $\mathsf{f}\vec{l} >_{\mathrm{cc}} \vec{m}$, alors $\mathsf{f}\vec{l} >_{\mathrm{cc}} \mathsf{g}\vec{m}$
(subterm-SN)	si $\mathsf{f}\vec{l} >_{\mathrm{cc}} t$, $t \rhd_s u : \mathsf{B} \in \mathcal{B}$ et $I(\mathsf{B}) = \mathcal{SN}$, alors $\mathsf{f}\vec{l} >_{\mathrm{cc}} u$
(var)	si $x \in X - \mathrm{FV}(\vec{l})$, alors $\mathsf{f}\vec{l} >^{X}_{\mathrm{cc}} x$
(abs)	si $\mathsf{f}\vec{l} >^{X\cup\{x\}}_{\mathrm{cc}} t$ et $x \notin \mathrm{FV}(\vec{l})$, alors $\mathsf{f}\vec{l} >^{X}_{\mathrm{cc}} \lambda xt$
(undef)	si $\mathsf{g} \in \mathcal{F} - \mathcal{D}(\mathcal{R})$, alors $\mathsf{f}\vec{l} >_{\mathrm{cc}} \mathsf{g}$
(subterm-acc)	si $\mathsf{f}\vec{l} >_{\mathrm{cc}} \mathsf{g}\vec{t} : \mathsf{B}$ et $i \in \mathrm{Acc}(\mathsf{g})$, alors $\mathsf{f}\vec{l} >_{\mathrm{cc}} t_i$
(subterm-abs)	si $\mathsf{f}\vec{l} >^{X}_{\mathrm{cc}} \lambda xt$ et $x \notin \mathrm{FV}(\vec{l})$, alors $\mathsf{f}\vec{l} >^{X\cup\{x\}}_{\mathrm{cc}} t$
(subterm-app)	si $\mathsf{f}\vec{l} >^{X}_{\mathrm{cc}} tu$ et $u{\downarrow_\eta} \in X$, alors $\mathsf{f}\vec{l} >^{X}_{\mathrm{cc}} t$

Ainsi, on peut montrer :

Theorem 38 Au premier ordre, RPO est le plus petit point fixe de la clôture de calculabilité restreinte aux règles (arg), (red) et (rec).

Theorem 39 La relation HORPO [12] définie dans [JR99] est incluse dans la clôture transitive et monotone du plus petit point fixe de la clôture de calculabilité.

Cela fournit une méthode simple d'étendre HORPO, y compris à des systèmes de types plus complexes.

Cette version de HORPO ne permet toutefois pas de montrer la terminaison de systèmes utilisant des types inductifs d'ordre supérieur. Comme nous l'avons vu dans la Section 4.2, pour cela, il est nécessaire de comparer les termes par leur rang dans une interprétation particulière des types (différente de $\mathcal{SN}$). Cela m'a conduit à développer avec Jouannaud et Rubio une nouvelle version de HORPO intégrant la notion d'accessibilité [BJR07, BJR08].

Dans [JR06], Jouannaud et Rubio propose une technique pour construire un ordre permettant de montrer la terminaison de HRSs (termes en forme β-normale η-longue avec filtrage modulo $\beta\eta$-équivalence) à partir de HORPO (termes quelconque avec filtrage modulo α-équivalence). Il conviendrait de comparer leur approche avec celle basée sur la clôture de calculabilité présentée dans la section précédente, qui ne nécessite aucune modification de la clôture ou du système considéré.

12. Contrairement à RPO et à ce que son nom laisse penser, HORPO n'est pas une relation transitive (un ordre).

4.7 Paires de dépendance

[Bla06a] F. Blanqui. Higher-order dependency pairs. In *Proceedings of the 8th International Workshop on Termination*, 2006.

[KISB09] K. Kusakari, Y. Isogai, M. Sakai, and F. Blanqui. Static dependency pair method based on strong computability for higher-order rewrite systems. *IEICE Transactions on Information and Systems*, E92-D(10) :2007–2015, 2009.

La notion de paire de dépendance est une notion introduite par Arts et Giesl pour la réécriture du premier ordre qui est maintenant à la base de tous les prouveurs automatiques de terminaison [AG97, AG00]. Nous allons voir dans cette section quelle relation on peut établir avec celle de clôture de calculabilité.

Definition 40 (Paire de dépendance) L'ensemble des *paires de dépendance* d'un ensemble $\mathcal{R}$ de règles de réécriture de la forme $\mathsf{f}\vec{l} \to r$, est l'ensemble $\mathrm{DP}(\mathcal{R})$ des paires $(\mathsf{f}\vec{l}, \mathsf{g}\vec{m})$ telle qu'il existe une règle $\mathsf{f}\vec{l} \to r \in \mathcal{R}$ telle que $\mathsf{g}\vec{m}$ est un sous-terme de r non inclus dans $\vec{l}$ [Der04] et g est un symbole défini par $\mathcal{R}$.

Le théorème fondamental des paires de dépendance est alors :

Theorem 41 ([AG97]) La relation $\to_{\mathcal{R}}$ termine ssi $\to_{\mathcal{R}i}^* \to_{\mathrm{DP}(\mathcal{R})h}$ termine, où $\to_{\mathcal{R}i}$ désigne une réécriture interne [13] et $\to_{\mathrm{DP}(\mathcal{R})h}$ une réécriture en tête.

Ainsi, pour montrer la terminaison de $\to_{\mathcal{R}}$, il suffit de montrer qu'il n'y a pas de chemins infinis dans le graphe, dit de dépendance, dont les noeuds sont les paires de dépendance et tel qu'il y a un arc entre deux paires $(\mathsf{f}\vec{l}, \mathsf{g}\vec{m})$ et $(\mathsf{h}\vec{n}, \mathsf{j}\vec{p})$ s'il existe une substitution σ telle $\mathsf{g}\vec{m}\sigma \to_{\mathcal{R}i}^* \mathsf{h}\vec{n}\sigma$.

La preuve usuellement donnée n'est pas intuitionniste. Elle procède de la manière suivante. Si $\to_{\mathcal{R}}$ ne termine pas, alors il existe un plus petit terme ne terminant pas mais dont les sous-termes terminent $t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \ldots$ Il existe alors i et $\mathsf{f}\vec{l} \to r \in \mathcal{R}$ tel que $t_0 \to_{\mathcal{R}i}^* t_i = \mathsf{f}\vec{l}\sigma \to_{\mathcal{R}h} r\sigma = t_{i+1}$. Comme t_{i+1} ne termine pas, il doit contenir un sous-terme u ne terminant pas. Comme les sous-termes de t_i terminent, u ne peut être que de la forme $\mathsf{g}\vec{m}\sigma$ avec $\mathsf{g} \in \mathcal{D}(\mathcal{R})$ et $\mathsf{g}\vec{m}$ non inclus dans $\vec{l}$.

La notion de clôture de calculabilité permet de donner une preuve intuitionniste (formalisée dans l'assistant à la preuve Coq [BK11]) et d'étendre ce théorème à l'ordre supérieur, sans changer la définition de paire de dépendance, sous certaines conditions :

13. Une présentation alternative consiste à étendre $\mathcal{F}$ par des symboles marqués $\{\mathsf{f}^\sharp \mid \mathsf{f} \in \mathcal{F}\}$ et à définir une paire de dépendance comme $(\mathsf{f}^\sharp\vec{l}, \mathsf{g}^\sharp\vec{m})$. Alors, $\to_{\mathcal{R}}$ termine ssi $\to_{\mathcal{R}}^* \to_{\mathrm{DP}(\mathcal{R})h}$ termine.

Theorem 42 Soit $\mathcal{R}$ un ensemble de règles de réécriture $l \to r$ telles que l est un motif de Miller de la forme $\mathsf{f}\vec{l}$ et $\to_{\mathrm{DP}(\mathcal{R})}$ préserve le typage et les variables libres $(\forall (t,u) \in \mathrm{DP}(\mathcal{R}), \mathrm{FV}(u) \subseteq \mathrm{FV}(t))$. La relation $\to_\beta \cup \to_{\mathcal{R}}$ termine ssi $\to_\beta \cup \to^*_{\mathcal{R}i} \to_{\mathrm{DP}(\mathcal{R})h}$ termine.

Étant donné qu'une paire de dépendance ne se réécrit qu'en tête, il n'est pas nécessaire de chercher un ordre monotone pour orienter celles-ci, contrairement aux règles. On peut donc se contenter de considérer la clôture de calculabilité plutôt que HORPO par exemple.

La condition sur le typage peut être limitante dans la mesure où les ordres actuellement connus à l'ordre supérieur (clôture de calculabilité ou HORPO) l'exige. Il est toutefois possible d'affaiblir cette condition en considérant une relation d'équivalence sur les types de base, voir un pré-ordre de simplification particulier sur les types [JR07].

La condition sur la préservation des variables libres est plus sérieuse et interdit de considérer des types inductifs d'ordre supérieur avec lesquels certains appels récursifs contiennent des variables liées. Ceci dit, une analyse plus fine permettrait peut-être d'éliminer cette condition en remplaçant les variables non libres par des constantes, comme cela est fait dans le cadre des HRS, donc sans β-réduction explicite, par Sakai *et al* [SWS01].

Dans [Kop11], Kop fait une analyse des causes possibles de non terminaison de $\to_\beta \cup \to_{\mathcal{R}}$ et propose une notion de paire de dépendance et de graphe de dépendance qui étend celle de Sakai *et al* à la β-réduction.

Dans ces travaux, sont également considérées comme paires de dépendance les paires $(\mathsf{f}\vec{l}, x\vec{m})$ telles qu'il existe une règle $\mathsf{f}\vec{l} \to r$ telle que $x\vec{m}$ est un sous-terme de r et x une variable libre de r, ce qui multiple le nombre d'arcs dans le graphe de dépendance de manière importante. Or, comme il est montré dans [KISB09, SKB11], il est possible d'éliminer ces paires quand x est accessible ou appartient à la clôture de calculabilité de $\mathsf{f}\vec{l}$.

4.8 Annotations de taille

[Bla04] F. Blanqui. A type-based termination criterion for dependently-typed higher-order rewrite systems. In *Proceedings of the 15th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 3091, 2004.

[Bla05a] F. Blanqui. Decidability of type-checking in the calculus of algebraic constructions with size annotations. In *Proceedings of the 19th International Conference on Computer Science Logic*, Lecture Notes in Computer Science 3634, 2005.

Dans la Section 4.2, nous avons vu que, contrairement à l'ordre sur les rangs, l'ordre structurel ne permet pas d'orienter des règles sur des types positifs non stricts. De façon à utiliser l'ordre sur les rangs, des chercheurs ont développé des systèmes de types où les types de base sont équipés d'une annotation fournissant une borne supérieure aux rangs des termes de ce type [Gim96, HPS96, Xi02, Abe04, BFG+04]. Ainsi, une constante de type B est remplacé par une famille de constantes de type B^a où a est un terme du premier ordre interprété dans les ordinaux. L'algèbre la plus simple est $a ::= \alpha \mid \mathsf{s}a \mid \infty$, où α est une variable, s est le successeur, et ∞ la borne supérieur de tous les rangs possibles. Étant donné une interprétation μ des variables dans les ordinaux, un type B^a peut alors être interprété comme l'ensemble des termes calculables de type B dont le rang est inférieur ou égal à l'interprétation de a (on parle alors de μ-calculabilité). On se retrouve ainsi avec un système de types avec sous-typage : B^a est un sous-type de B^b si, pour toute valuation μ, $a\mu$ est plus petit que $b\mu$.

J'ai montré que cette approche définie dans les travaux précédents pour le λ-calcul simplement typé ou polymorphe avec types inductifs standards et fonctions définies par point fixe et filtrage constructeur complet [HQM86] peut être étendue à la réécriture, y compris les systèmes de réécriture non orthogonaux et non confluents, et aux types dépendants (en utilisant notamment les travaux de Chen sur le sous-typage dans le calcul des constructions [Che98]).

Étant donné qu'une annotation de taille est interprétée comme une borne supérieure, pour prouver la terminaison, il ne suffit pas d'orienter les paires de dépendance $(\mathsf{f}\vec{l}, \mathsf{g}\vec{m})$ en comparant les annotations qui équipent les types de $\vec{l}$ et $\vec{m}$ ($\mathcal{F}$-pré-ordre sur les annotations de taille). Il convient également de s'assurer que les annotations de $\vec{l}$ sont bien minimales, ce qui n'est pas toujours possible. En effet, dans [BR09], nous montrons que, pour certains motifs non linéaires ou de profondeur supérieure ou égale à 3 sur des types inductifs non basiques (avec lesquels le rang d'un élément peut être supérieur au premier ordinal infini ω), il n'existe pas d'annotation minimale dans l'algèbre avec s et ∞.

Ceci dit, cela ne suffit toujours pas pour orienter la règle $[\mathrm{ex}(\mathsf{c}x) \to x\,\mathrm{ex}]$. Pour résoudre ce problème, il faut utiliser un système de types contraints [BR06],

où l'ensemble des types est définis de la manière suivante :

$$T ::= \mathsf{B} \mid T \Rightarrow T \mid \forall\vec{\alpha}(C)T \mid \exists\vec{\alpha}(C)T$$
$$C ::= \top \mid \bot \mid C \wedge C \mid C \vee C \mid a < b \mid a = b$$

et où l'ensemble des termes bien typés relativement à un environnement de typage Γ et une contrainte C (modulo équivalence logique) est définis de la manière suivante :

- si $C;\Gamma \vdash t : U \Rightarrow V$ et $C;\Gamma \vdash u : U$, alors $C;\Gamma \vdash tu : V$;
- si $C;\Gamma \vdash t : \forall\vec{\alpha}(P)T$ et $C \vdash P^{\vec{a}}_{\vec{\alpha}}$, alors $C;\Gamma \vdash t : T^{\vec{a}}_{\vec{\alpha}}$;
- si $C \wedge P;\Gamma \vdash t : T$, $\vec{\alpha} \notin \mathrm{FV}(C,\Gamma)$, $C \vdash \exists\vec{\alpha}P$, alors $C;\Gamma \vdash t : \forall\vec{\alpha}(P)T$;
- ...

où, étant donné une valuation μ, un type $\forall\vec{\alpha}(P)T$ est naturellement interprété comme l'intersection $\bigcap_{\mu^{\vec{\mathfrak{a}}}_{\vec{\alpha}} \models P} [\![I]\!]\mu^{\vec{\mathfrak{a}}}_{\vec{\alpha}}$. Dans un tel système, on peut montrer que, si $C;\Gamma \vdash t : T$, μ est une valuation satisfaisant C et σ une substitution μ-calculable, alors $t\sigma$ est μ-calculable. Ainsi, on peut définir la clôture de calculabilité d'une paire $(\mathsf{f}, \vec{l})$ avec $\mathsf{f} : \forall\vec{\alpha}(\top)\vec{\mathsf{B}}^{\vec{\alpha}} \Rightarrow T$ et $C;\Gamma \vdash \vec{l} : \vec{\mathsf{B}}^{\vec{a}}$ comme l'ensemble des termes typables en prenant $\mathsf{f} : \forall\vec{\alpha}(\vec{\alpha} < \vec{a})\vec{\mathsf{B}}^{\vec{\alpha}} \Rightarrow T$. Comme, par hypothèse d'induction sur le $\mathcal{F}$-pré-ordre sur les tailles, $\mathsf{f} \in [\![\forall\vec{\alpha}(\vec{\alpha} < \vec{a})\vec{\mathsf{B}}^{\vec{\alpha}} \Rightarrow T]\!]$, nous obtenons que tout terme de la clôture est calculable. C'est en particulier le cas de $(x\,\mathrm{ex})$ si $x : (\mathsf{C}^x \Rightarrow \mathsf{L}) \Rightarrow \mathsf{L}$ et $\mathrm{ex} : \forall\alpha(\alpha < \mathsf{s}x)\mathsf{C}^\alpha \Rightarrow \mathsf{L}$.

Dans [Bla05a], j'ai défini un algorithme général pour décider si un terme est d'un type donné, dans le cas où toute contrainte satisfaisable admet une solution minimale (modulo renommage des variables) et s'il existe une procédure pour décider si une contrainte est satisfaisable et calculer la solution minimale. En particulier, j'ai montré que, dans l'algèbre avec s et ∞, toute contrainte satisfaisable admet une solution minimale et j'ai fournit une procédure de décision et de calcul de cette solution minimale (utilisant de la programmation linéaire).

En étendant le système de types contraints précédent aux quantifications existentielles $\exists\vec{\alpha}(C)T$, il est possible, d'une part, d'interpréter B^a comme les termes calculables de rang exactement égal à a (comme dans [Xi02], les termes de rang inférieur à a s'exprimant alors comme $\exists\alpha(\alpha < a)\mathsf{B}^\alpha$), et d'autre part, de fournir davantage d'information utilisable pour montrer la terminaison. Mais cela soulève plusieurs problèmes :

- Comment interpréter un type existentiel $\exists\vec{\alpha}(C)T$? Pour être cohérent, il faudrait l'interpréter comme l'union $\bigcup_{\mu^{\vec{\mathfrak{a}}}_{\vec{\alpha}} \models P} [\![I]\!]\mu^{\vec{\mathfrak{a}}}_{\vec{\alpha}}$. Malheureusement, $\mathbf{Red}_{\mathcal{R}}$ n'est pas clos par union, contrairement à $\mathbf{Sat}_{\mathcal{R}}$. Heureusement, il existe un sous-ensemble intéressant de $\mathbf{Sat}_{\mathcal{R}}$ appartenant à $\mathbf{Red}_{\mathcal{R}}$ si $\rightarrow = \rightarrow_\beta \cup \rightarrow_{\mathcal{R}}$ est (localement) confluent : les unions des ensembles $[\![\mathsf{B}^a]\!]$. Pour **Cand**, on peut donc prendre $\mathbf{Sat}_{\mathcal{R}}$ plutôt que $\mathbf{Red}_{\mathcal{R}}$ à condition que $\rightarrow$ soit (localement) confluente et à condition de limiter l'usage des types existentiels aux types de base. C'est ce problème avec l'interprétation de la quantification existentielle qui a motivé ensuite une partie du travail de thèse de Colin Riba [Rib07].

– Comment exploiter l'information fournit par un type existentiel $\exists\vec{\alpha}(C)T$? Par exemple, dans l'algorithme de *quicksort*, une liste est séparée en, d'une part, la sous-liste des valeurs inférieures au pivot et, d'autre part, la sous-liste des valeurs supérieures ou égales au pivot, puis les deux sous-listes sont triées et concaténées. Les opérations de pivot et de concaténation peuvent être annotée de la manière suivante :

$$\text{pivot} : \mathsf{N}^\infty \Rightarrow \forall\alpha(\top)\mathsf{L}^\alpha \Rightarrow \exists\beta\gamma(\alpha = \beta + \gamma)\mathsf{L}^\beta \times \mathsf{L}^\gamma$$

$$\text{concat} : \forall\alpha(\top)\mathsf{L}^\alpha \Rightarrow \forall\beta(\top)\mathsf{L}^\beta \Rightarrow \mathsf{L}^{\alpha+\beta}$$

En éliminant l'existentielle de manière classique en introduisant des variables existentielles, il n'est pas possible de montrer que

$$\text{concat}(\text{sort}(\text{fst}(\text{pivot}\,\ell)))(\text{sort}(\text{snd}(\text{pivot}\,\ell)))$$

a la même longueur (type) que ℓ car chaque instance de $(\text{pivot}\,\ell)$ introduira une nouvelle variable. Pour éviter cela, il est nécessaire de partager $(\text{pivot}\,\ell)$. On considérera donc plutôt l'expression

$$\text{let } x = \text{pivot}\,\ell \text{ in } \text{concat}(\text{sort}(\text{fst}\,x)))(\text{sort}(\text{snd}\,x))$$

et la règle d'élimination de l'existentielle suivante :
– si $C;\Gamma \vdash t : \exists\vec{\alpha}(P)T,\ \ C \wedge P;\Gamma, x : T \vdash u : U,\ \ C \vdash \exists\vec{\alpha}P$ et $x, \vec{\alpha} \notin \text{FV}(C, \Gamma, U)$, alors $C;\Gamma \vdash \text{let } x = t \text{ in } u : U$.

– Comment décider qu'une contrainte est satisfaisable ? Dans le cas où certains ordinaux peuvent être supérieurs à ω, je ne connais pas de procédure de décision. Autrement, et c'est le cas si tous les types inductifs sont basiques (*i.e.* du premier ordre), nous pouvons utiliser une procédure de décision pour l'arithmétique de Presburger [Pre29, FR74].

Tous les travaux précédents considèrent la même annotation pour les constructeurs, à savoir $\vec{T} \Rightarrow \mathsf{B}^{\mathsf{s}\alpha}$ avec, dans $\vec{T}$, chaque occurrence de B annotée par α, si les annotations sont interprétées comme étant des bornes supérieures, ou $\vec{T} \Rightarrow \mathsf{B}^{1+\max(\vec{\alpha})}$ avec, dans T_i, chaque occurrence de B annotée par α_i (les α_i étant deux à deux distincts), si les annotations sont interprétées comme la taille exacte. La taille d'un terme correspond alors à son rang ce qui, au premier ordre, correspond à la hauteur de sa forme normale.

Dans [Bla10b, BR09], nous montrons de deux manières différentes (par une définition approprié de $[\![\mathsf{B}^a]\!]$ et par l'utilisation de l'étiquetage sémantique décrit en Section 4.10) qu'il est possible de considérer d'autres annotations pour les constructeurs et donc d'autres notions de taille, la seule contrainte étant que l'annotation de sortie d'un constructeur doit être une fonction monotone et strictement expansive des annotations d'entrée ($f(\vec{\alpha}) > \vec{\alpha}$).

Cela permet par exemple de montrer la terminaison du système de normalisation des expressions conditionnelles suivant [Pau86] :

$$\begin{array}{rcl} \mathrm{nm}\ \mathrm{at} & \to & \mathrm{at} \\ \mathrm{nm}\ (\mathrm{if}\ \mathrm{at}\ y\ z) & \to & \mathrm{if}\ \mathrm{at}\ (\mathrm{nm}\ y)\ (\mathrm{nm}\ z) \\ \mathrm{nm}\ (\mathrm{if}\ (\mathrm{if}\ u\ v\ w)\ y\ z) & \to & \mathrm{nm}\ (\mathrm{if}\ u\ (\mathrm{nm}\ (\mathrm{if}\ v\ y\ z))\ (\mathrm{nm}\ (\mathrm{if}\ w\ y\ z))) \end{array}$$

en prenant $\mathrm{if} : \mathsf{E}^{\alpha} \Rightarrow \mathsf{E}^{\beta} \Rightarrow \mathsf{E}^{\gamma} \Rightarrow \mathsf{E}^{(\alpha+1)(\beta+\gamma+3)}$ et $\mathrm{nm} : \mathsf{E}^{\alpha} \Rightarrow \mathsf{E}^{\alpha}$.

4.9 Réécriture conditionnelle

> [BR06] F. Blanqui and C. Riba. Combining typing and size constraints for checking the termination of higher-order conditional rewrite systems. In *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Lecture Notes in Computer Science 4246, 2006.

Que la clôture de calculabilité puisse également être étendues à la réécriture conditionnelle est une conséquence immédiate de l'utilisation de types contraints. En effet, considérons une règle de réécriture conditionnelle de la forme :

$$t_1 \to^* \mathsf{b}_1 \wedge \ldots \wedge t_n \to^* \mathsf{b}_n \quad \Rightarrow \quad \mathsf{f}\vec{l} \to r$$

où $\mathsf{b}_i \in \{\text{true}, \text{false}\}$ et $\text{FV}(r, \vec{t}) \subseteq \text{FV}(\vec{l})$. Et considérons également que l'algèbre des annotations de taille est multi-sortée avec, d'un côté les tailles ordinales avec s et ∞ et, de l'autre, les tailles booléennes true et false (sans ordre entre elles). Alors, pour typer r, il suffit de rajouter les contraintes engendrées par la satisfaction des conditions, à savoir, $a_1 = \mathsf{b}_1 \wedge \ldots a_n = \mathsf{b}_n$ où a_i est la taille (booléenne) de t_i obtenue par typage !

Cela permet par exemple de montrer la terminaison de la fameuse fonction "91" de Mc Carthy :

$$\begin{array}{lll} x < 100 \to^* \text{true} & \Rightarrow & \mathsf{f}\ x \ \to\ \mathsf{f}\ (\mathsf{f}\ (x + 11)) \\ x < 100 \to^* \text{false} & \Rightarrow & \mathsf{f}\ x \ \to\ x - 10 \end{array}$$

en prenant $\mathsf{f} : \mathsf{N}^\alpha \Rightarrow \mathsf{N}^{f(\alpha)}$ avec $f(\alpha) = 91$ si $x \leq 100$, et $f(\alpha) = x - 10$ sinon.

4.10 Étiquetage sémantique

> [BR09] F. Blanqui and C. Roux. On the relation between sized-types based termination and semantic labelling. In *Proceedings of the 23rd International Conference on Computer Science Logic*, Lecture Notes in Computer Science 5771, 2009.

Nous terminerons cette monographie en montrant que les annotations de taille sont un cas particulier de la technique dite d'étiquetage sémantique introduite au premier ordre par Zantema [Zan95] et étendu à l'ordre supérieur par Hamana [Ham07].

L'idée (au premier ordre) est la suivante. Soit $\mathcal{R}$ un ensemble de règles de réécriture, et soit $\mathcal{M}$ un modèle de $\mathcal{R}$ vu comme un ensemble d'équations, c'est-à-dire, un ensemble M et, pour chaque symbole $\mathsf{f} \in \mathcal{F}$ d'arité n, une fonction $\mathsf{f}_\mathcal{M} : M^n \to M$, tels que, pour toute règle $l \to r \in \mathcal{R}$ et valuation $\mu : \mathcal{X} \to M$, $[\![l]\!]\mu = [\![r]\!]\mu$, où l'interprétation d'un terme est défini par $[\![x]\!]\mu = \mu(x)$ et $[\![\mathsf{f}\vec{t}]\!]\mu = \mathsf{f}_\mathcal{M}([\![\vec{t}]\!]\mu)$. Étant donnés un terme t et une valuation μ, on peut alors étiqueter chaque symbole f de t par la sémantique des arguments auxquels f est appliqué. L'étiquetage transforme un terme t sur $\mathcal{F}$ en un terme $\mathrm{lab}_\mu(t)$ sur $\{\mathsf{f}_{\vec{a}} \mid \mathsf{f} \in \mathcal{F}, \vec{a} \in M\}$: $\mathrm{lab}_\mu(x) = x$ et $\mathrm{lab}_\mu(\mathsf{f}\vec{t}) = \mathsf{f}_{[\![\vec{t}]\!]\mu}\mathrm{lab}_\mu(\vec{t})$. Le premier théorème de l'étiquetage sémantique nous dit alors que la terminaison de $\mathcal{R}$ est équivalente à celle du système obtenu en considérant tous les étiquetages possibles des règles de $\mathcal{R}$:

Theorem 43 ([Zan95]) Étant donné un ensemble $\mathcal{R}$ de règles de réécriture, la relation $\to_\mathcal{R}$ termine si et seulement s'il existe un modèle $\mathcal{M}$ de $\mathcal{R}$ tel que $\to_{\mathrm{lab}(\mathcal{R})}$ termine, où $\mathrm{lab}(\mathcal{R}) = \{\mathrm{lab}_\mu(l) \to \mathrm{lab}_\mu(r) \mid l \to r \in \mathcal{R}, \mu : \mathcal{X} \to M\}$.

Bien que $\mathrm{lab}(\mathcal{R})$ puisse être beaucoup plus gros que $\mathcal{R}$, voir infini (si M est infini), sa terminaison est souvent plus facile à établir du fait que deux occurrences d'un même symbole est remplacée par deux symboles distincts, si la sémantique de leurs arguments est différente. Il existe d'ailleurs des systèmes $\mathcal{R}$ qui terminent mais ne sont pas inclus dans un ordre de simplification, tandis que $\mathrm{lab}(\mathcal{R})$ l'est. Il est possible de rendre cela plus précis en considérant non pas un modèle mais un "quasi-modèle" où une règle $l \to r$ est considérée non comme une équation $l = r$ mais comme une inégalité $l \geq r$. Dit autrement, $\mathcal{R}$ est interprété non dans la catégorie des ensembles mais dans la catégorie des ensembles partiellement ordonnés. C'est le second théorème de l'étiquetage sémantique :

Theorem 44 ([Zan95]) Étant donné un ensemble $\mathcal{R}$ de règles de réécriture, la relation $\to_\mathcal{R}$ termine si et seulement s'il existe un quasi-modèle $(\mathcal{M}, \geq_\mathcal{M})$ de $\mathcal{R}$ tel que chaque $\mathsf{f}_\mathcal{M}$ est monotone par rapport à $(>_\mathcal{M})_{\mathrm{prod}}$ et la relation $\to_{\mathrm{lab}(\mathcal{R})} \cup \to_{\mathrm{Decr}}$ termine, où $\mathrm{Decr} = \{\mathsf{f}_{\vec{a}}(\vec{x}) \to \mathsf{f}_{\vec{b}}(\vec{x}) \mid \vec{a}(>_\mathcal{M})_{\mathrm{prod}}\vec{b}\}$.

Le premier théorème est un cas particulier du second. La preuve de la partie "seulement si" ($\Rightarrow$) montre que, si $\mathcal{R}$ termine (sans être nécessairement inclus dans un ordre de simplification), alors il existe un quasi-modèle dans lequel $\mathrm{lab}(\mathcal{R})$ est inclus dans un ordre de simplification (il suffit de prendre $\mathcal{M} = \mathcal{L}$ et $\geq_{\mathcal{M}} = \to^{*}!$).

Dans [Ham07], Hamana a étendu ces résultats aux CRS simplement typés en interprétant les λ-termes dans la catégorie des $\mathcal{F}$-monoïdes [FPT99, Ham06].

La similarité entre l'étiquetage sémantique et les annotations de taille est frappante. L'étude de la relation précise entre ces deux approches a été le point de départ de la thèse de Cody Roux [Rou11]. Dans [BR09], nous montrons que les annotations de taille sont un cas particulier d'étiquetage sémantique. Des précautions doivent cependant être prises pour traiter les symboles annotés par ∞ (par une technique qui s'apparente à l'étiquetage prédictif [HM06]), interpréter les types fonctionnels (il faut se restreindre aux ordinaux réalisables), et la β-réduction elle-même (la version étiquetée de la β-réduction ne correspond pas à une étape de β-réduction mais à une étape de β-réduction modulo certaines règles structurelles).

Références

[Abe04] A. Abel. Termination checking with types. *Theoretical Informatics and Applications*, 38(4) :277–319, 2004.

[ACCL91] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4) :375–416, 1991.

[AG97] T. Arts and J. Giesl. Automatically proving termination where simplification orderings fail. In *Proceedings of the 7th International Joint Conference on Theory and Practice of Software Development*, Lecture Notes in Computer Science 1214, 1997.

[AG00] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236 :133–178, 2000.

[Bar84] H. Barendregt. *The Lambda Calculus : Its Syntax and Semantics*. North-Holland, 2nd edition, 1984.

[Bar92] H. Barendregt. Lambda calculi with types. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of logic in computer science*, volume 2. Oxford University Press, 1992.

[BFG+04] G. Barthe, M. J. Frade, E. Giménez, L. Pinto, and T. Uustalu. Type-based termination of recursive definitions. *Mathematical Structures in Computer Science*, 14(1) :97–141, 2004.

[BJO02] F. Blanqui, J.-P. Jouannaud, and M. Okada. Inductive-data-type systems. *Theoretical Computer Science*, 272 :41–68, 2002.

[BJR07] F. Blanqui, J.-P. Jouannaud, and A. Rubio. HORPO with computability closure : A reconstruction. In *Proceedings of the 14th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Lecture Notes in Computer Science 4790, 2007.

[BJR08] F. Blanqui, J.-P. Jouannaud, and A. Rubio. The computability path ordering : the end of a quest. In *Proceedings of the 22nd International Conference on Computer Science Logic*, Lecture Notes in Computer Science 5213, 2008. Invited paper.

[BK11] F. Blanqui and A. Koprowski. CoLoR : a Coq library on well-founded rewrite relations and its application to the automated verification of termination certificates. *Mathematical Structures in Computer Science*, 21(4) :827–859, 2011.

[BKR05] E. Bonelli, D. Kesner, and A. Ríos. Relating higher-order and first-order rewriting. *Journal of Logic and Computation*, 15 :901–947, 2005.

[Bla00] F. Blanqui. Termination and confluence of higher-order rewrite systems. In *Proceedings of the 11th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 1833, 2000.

[Bla03] F. Blanqui. Rewriting modulo in deduction modulo. In *Proceedings of the 14th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 2706, 2003.

[Bla04] F. Blanqui. A type-based termination criterion for dependently-typed higher-order rewrite systems. In *Proceedings of the 15th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 3091, 2004.

[Bla05a] F. Blanqui. Decidability of type-checking in the calculus of algebraic constructions with size annotations. In *Proceedings of the 19th International Conference on Computer Science Logic*, Lecture Notes in Computer Science 3634, 2005.

[Bla05b] F. Blanqui. Definitions by rewriting in the calculus of constructions. *Mathematical Structures in Computer Science*, 15(1) :37–92, 2005.

[Bla05c] F. Blanqui. Inductive types in the calculus of algebraic constructions. *Fundamenta Informaticae*, 65(1-2) :61–86, 2005.

[Bla06a] F. Blanqui. Higher-order dependency pairs. In *8th International Workshop on Termination*, 2006.

[Bla06b] F. Blanqui. (HO)RPO revisited. Technical Report 5972, INRIA, 2006.

[Bla07] F. Blanqui. Computability closure : Ten years later. In *Rewriting, Computation and Proof – Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, volume 4600 of *Lecture Notes in Computer Science*, 2007.

[Bla10a] F. Blanqui. Fonctions, réécriture et preuves : terminaison et certification. `http://www-rocq.inria.fr/~blanqui/divers/hdr-dossier.pdf`, 2010. Dossier d'habilitation à diriger des recherches.

[Bla10b] F. Blanqui. A size-based termination criterion for dependently-typed higher-order rule-based programs. `http://www-rocq.inria.fr/~blanqui/`, 2010. Draft for a journal. 62 pages.

[BR06] F. Blanqui and C. Riba. Combining typing and size constraints for checking the termination of higher-order conditional rewrite systems. In *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Lecture Notes in Computer Science 4246, 2006.

[BR09] F. Blanqui and C. Roux. On the relation between sized-types based termination and semantic labelling. In *Proceedings of the 23rd International Conference on Computer Science Logic*, Lecture Notes in Computer Science 5771, 2009.

[CF58] H. B. Curry and R. Feys. *Combinatory Logic*. North-Holland, 1958.

[Che98] G. Chen. *Subtyping, Type Conversion and Transitivity Elimination*. PhD thesis, Université Paris VII, France, 1998.

[CJ03] H. Comon and J.-P. Jouannaud. Les termes en logique et en programmation, 2003. Course notes, `www.lix.polytechnique.fr/~jouannaud/articles/cours-tlpo.pdf`.

[CK96] R. Di Cosmo and D. Kesner. Combining algebraic rewriting, extensional lambda calculi, and fixpoints. *Theoretical Computer Science*, 169(2) :201–220, 1996.

[CK01] H. Cirstea and C. Kirchner. The rewriting calculus. *Logic Journal of the Interest Group in Pure and applied Logic*, 9(3) :339–410, 2001.

[Coq92] T. Coquand. Pattern matching with dependent types. In *Proceedings of the International Workshop on Types for Proofs and Programs*, 1992.

[CPM88] T. Coquand and C. Paulin-Mohring. Inductively defined types. In *Proceedings of the International Conference on Computer Logic*, Lecture Notes in Computer Science 417, 1988.

[dB72] N. de Bruijn. Lambda-calculus notation with nameless dummies : a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indagationes Mathematicae*, 34(5) :381–392, 1972.

[Der79] N. Dershowitz. Orderings for term rewriting systems. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, 1979.

[Der82] N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17 :279–301, 1982.

[Der04] N. Dershowitz. Termination by abstraction. In *Proceedings of the 20th International Conference on Logic Programming*, Lecture Notes in Computer Science 3132, 2004.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 6. North-Holland, 1990.

[Dyb00] P. Dybjer. A general formulation of simultaneous inductive-recursive definitions in type theory. *Journal of Symbolic Logic*, 65(2) :525–549, 2000.

[Eke96] S. Eker. Fast matching in combinations of regular equational theories. In *Proceedings of the 1st International Workshop on Rewriting Logic and Applications*, Electronic Notes in Theoretical Computer Science 4, 1996.

[Erw96] M. Erwig. Active patterns. In *Proceedings of the 8th International Workshop on Implementation of Functional Languages*, Lecture Notes in Computer Science 1268, 1996.

[FPT99] M. Fiore, G. Plotkin, and D. Turi. Abstract syntax and variable binding. In *Proceedings of the 14th IEEE Symposium on Logic in Computer Science*, 1999.

[FR74] M. Fischer and M. Rabin. Super-exponential complexity of presburger arithmetic. In *Proceedings of the SIAM-AMS Symposium in Applied Mathematics*, 1974.

[Gal90] J. Gallier. On Girard's "candidats de réductibilité". In P.-G. Odifreddi, editor, *Logic and Computer Science*. North-Holland, 1990.

[Gan80] R. O. Gandy. Proofs of strong normalization. In J. R. Hindley and J. P. Seldin, editors, *To H. B. Curry : Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 457–477. Academic Press, 1980.

[Gim96] E. Giménez. *Un Calcul de Constructions infinies et son application à la vérification de systèmes communiquants*. PhD thesis, ENS Lyon, France, 1996.

[Gir71] J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse et son application à l'élimination des coupures dans l'analyse et la théorie des types. In J. Fenstad, editor, *Proc. of the 2nd Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1971.

[Gir72] J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures dans l'arithmetique d'ordre supérieur*. PhD thesis, Université Paris VII, France, 1972.

[GKK05] J. Glauert, D. Kesner, and Z. Khasidashvili. Expression reduction systems and extensions : An overview. In *Processes, Terms and Cycles : Steps to the Road of Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion os His 60th Birthday*, volume 3838 of *Lecture Notes in Computer Science*, 2005.

[GLT88] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, 1988.

[GP91] J. F. Groote and A. Ponse. Proof theory for μcrl. Technical Report CS-R9138, Centrum voor Wiskunde en Informatica, Netherlands, 1991.

[Ham06] M. Hamana. An initial algebra approach to term rewriting systems with variable binders. *Journal of Higher-Order and Symbolic Computation*, 19(2-3) :231–262, 2006.

[Ham07] M. Hamana. Higher-order semantic labelling for inductive datatype systems. In *Proceedings of the 9th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, 2007.

[HM06] N. Hirokawa and A. Middeldorp. Predictive labeling. In *Proceedings of the 17th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 4098, 2006.

[HPS96] J. Hughes, L. Pareto, and A. Sabry. Proving the correctness of reactive systems using sized types. In *Proceedings of the 23th ACM Symposium on Principles of Programming Languages*, 1996.

[HQM86] R. Harper, D. Mac Queen, and R. Milner. Standard ML. Technical Report ECS-LFCS-86-2, University of Edinburgh, UK, 1986.

[Hue76] G. Huet. Résolution d'équations dans les langages d'ordre 1, 2, ..., ω, 1976. Thèse d'État, Université Paris VII, France.

[Hue80] G. Huet. Confluent reductions : Abstract properties and applications to term-rewriting systems. *Journal of the ACM*, 27(4) :797–821, 1980.

[Jay04] C. B. Jay. The pattern calculus. *ACM Transactions on Programming Languages and Systems*, 26(6) :911–937, 2004.

[JK86] J.-P. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal on Computing*, 15(4) :1155–1194, 1986.

[JK09] C. B. Jay and Delia Kesner. First-class patterns. *Journal of Functional Programming*, 19(2) :191–225, 2009.

[JM84] J.-P. Jouannaud and M. Muñoz. Termination of a set of rules modulo a set of equations. In *Proceedings of the 7th International Conference on Automated Deduction*, Lecture Notes in Computer Science 170, 1984.

[JO91] J.-P. Jouannaud and M. Okada. A computation model for executable higher-order algebraic specification languages. In *Proceedings of the 6th IEEE Symposium on Logic in Computer Science*, 1991.

[JR99] J.-P. Jouannaud and A. Rubio. The higher-order recursive path ordering. In *Proceedings of the 14th IEEE Symposium on Logic in Computer Science*, 1999.

[JR06] J.-P. Jouannaud and A. Rubio. Higher-order orderings for normal rewriting. In *Proceedings of the 17th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 4098, 2006.

[JR07] J.-P. Jouannaud and A. Rubio. Polymorphic higher-order recursive path orderings. *Journal of the ACM*, 54(1) :1–48, 2007.

[KB70] D. Knuth and P. Bendix. Simple word problems in universal algebra. In *Computational problems in abstract algebra, Proceedings of a Conference held at Oxford in 1967*, pages 263–297. Pergamon Press, 1970.

[Kha90] Z. Khasidashvili. Expression reduction systems. Technical Report 36, I. Vekua Institute of Applied Mathematics of Tbilisi State University, 1990.

[KISB09] K. Kusakari, Y. Isogai, M. Sakai, and F. Blanqui. Static dependency pair method based on strong computability for higher-order rewrite systems. *IEICE Transactions on Information and Systems*, E92-D(10) :2007–2015, 2009.

[KL80] S. Kamin and J.-J. Lévy. Two generalizations of the recursive path ordering, 1980. Unpublished. Available on `http://perso.ens-lyon.fr/pierre.lescanne/not_accessible.html`.

[Klo80] J. W. Klop. *Combinatory Reduction Systems*. PhD thesis, Utrecht Universiteit, Netherlands, 1980. Published as Mathematical Center Tract 129.

[KM01] H. Kirchner and P.-E. Moreau. Promoting rewriting to a programming language : A compiler for non-deterministic rewrite programs in associative-commutative theories. *Journal of Functional Programming*, 11(2) :207–251, 2001.

[Kop11] C. Kop. Higher order dependency pairs for algebraic functional systems. In *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications*, Leibniz International Proceedings in Informatics 10, 2011.

[KvOdV08] J. W. Klop, V. van Oostrom, and R. de Vrijer. Lambda calculus with patterns. *Theoretical Computer Science*, 398(1-3) :16–31, 2008.

[KvOvR93] J. W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems : introduction and survey. *Theoretical Computer Science*, 121 :279–308, 1993.

[LB77] D. Lankford and A. Ballantyne. Decision procedures for simple equational theories with commutative-associative axioms : Complete sets of commutative-associative reductions. Technical Report ATP-39, Automatic Theorem Proving Project, University of Texas, Austin, USA, 1977.

[Loa03] R. Loader. Higher-order β-matching is undecidable. *Logic Journal of the Interest Group in Pure and applied Logic*, 11(1) :51–68, 2003.

[Mat98] R. Matthes. *Extensions of System F by Iteration and Primitive Recursion on Monotone Inductive Types*. PhD thesis, Ludwig Maximilians Universität, München, Germany, 1998.

[Mat00] R. Matthes. Lambda calculus : A case for inductive definitions, 2000.

[Men87] N. P. Mendler. *Inductive Definition in Type Theory*. PhD thesis, Cornell University, USA, 1987.

[Mil91] D. Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4) :497–536, 1991.

[MN98] R. Mayr and T. Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192(2) :3–29, 1998.

[Nip91] T. Nipkow. Higher-order critical pairs. In *Proceedings of the 6th IEEE Symposium on Logic in Computer Science*, 1991.

[Par97] M. Parigot. Proofs of strong normalization for second order classical natural deduction. *Journal of Symbolic Logic*, 62(4) :1461–1479, 1997.

[Pau86] L. Paulson. Proving termination of normalization functions for conditional expressions. *Journal of Automated Reasoning*, 2(1) :63–74, 1986.

[Pla78] D. A. Plaisted. A recursively defined ordering for proving termination of term rewriting systems. Technical report, University of Illinois, Urbana-Champaign, USA, 1978.

[Pre29] M. Presburger. über die vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchem die addition als einzige operation hervortritt. In *Sprawozdanie z I Kongresu Matematykow Krajow Slowcanskich, Warszawa, Poland*, 1929.

[PS81] G. Peterson and M. Stickel. Complete sets of reductions for some equational theories. *Journal of the ACM*, 28(2) :233–264, 1981.

[Qia93] Z. Qian. Linear unification of higher-order patterns. In *Proceedings of the 5th International Joint Conference on Theory and Practice of Software Development*, Lecture Notes in Computer Science 668, 1993.

[Rib07] C. Riba. *Définitions par réécriture dans le lambda-calcul : confluence, réductibilité et typage*. PhD thesis, Institut National Polytechnique de Lorraine, Nancy, France, 2007.

[Rou11] C. Roux. *Termination of higher-order rewrite systems*. PhD thesis, Université Henri Poincaré, Nancy, France, 2011.

[San67] L. E. Sanchis. Functionals defined by recursion. *Notre Dame Journal of Formal Logic*, 8 :161–174, 1967.

[SKB11] S. Suzuki, K. Kusakari, and F. Blanqui. Argument filterings and usable rules in higher-order rewrite systems. *IPSJ Transactions on Programming*, 4(2) :1–12, 2011.

[Sta79] R. Statman. The typed λ-calculus is not elementary recursive. *Theoretical Computer Science*, 9 :73–81, 1979.

[Ste98] M. Stefanova. *Properties of Typing Systems*. PhD thesis, Katholiecke Universiteit Nijmegen, Netherlands, 1998.

[Sti09] C. Stirling. Decidability of higher-order matching. *Logical Methods in Computer Science*, 5(3) :1–52, 2009.

[SWS01] M. Sakai, Y. Watanabe, and T. Sakabe. An extension of dependency pair method for proving termination of higher-order rewrite systems. *IEICE Transactions on Information and Systems*, E84-D(8) :1025–1032, 2001.

[Tai67] W. W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic*, 32(2) :198–212, 1967.

[Tai75] W. W. Tait. A realizability interpretation of the theory of species. In R. Parikh, editor, *Proceedings of the 1972 Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, 1975.

[Tak95] M. Takahashi. Parallel reduction in λ-calculus. *Information and Computation*, 118 :120–127, 1995.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its application. *Pacific Journal of Mathematics*, 5 :285–309, 1955.

[Tat07] M. Tatsuta. Simple saturated sets for disjunction and second-order existential quantification. In *Proceedings of the 8th International Conference on Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science 4583, 2007.

[TeR03] TeReSe. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[Toy87] Y. Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25(3) :141–143, 1987.

[Tro73] A. S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*, chapter Models and Computability, pages 97–174. Springer, 1973.

[Tul10] M. Tullsen. First class patterns. In *Proceedings of the 2nd International Symposium on Practical Aspects of Declarative Languages*, Lecture Notes in Computer Science 1753, 2010.

[vD80] D. van Daalen. *The language theory of Automath*. PhD thesis, Eindhoven University of Technology, Netherlands, 1980.

[vdP96] J. van de Pol. *Termination of higher-order rewrite systems*. PhD thesis, Utrecht Universiteit, Netherlands, 1996.

[vO90] V. van Oostrom. Lambda calculus with patterns. Technical Report IR 228, Vrije Universiteit, Amsterdam, Netherlands, 1990.

[vO94] V. van Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, Vrije Universiteit Amsterdam, Netherlands, 1994.

[vOvR93] V. van Oostrom and F. van Raamsdonk. Comparing combinatory reduction systems and higher-order rewrite systems. In *Proceedings of the 1st International Workshop on Higher-Order Algebra, Logic and Term Rewriting*, Lecture Notes in Computer Science 816, 1993.

[vR96] F. van Raamsdonk. *Confluence and Normalization for Higher-Order Rewriting*. PhD thesis, Vrije University Amsterdam, Netherlands, 1996.

[WC03] D. Walukiewicz-Chrząszcz. Termination of rewriting in the calculus of constructions. *Journal of Functional Programming*, 13(2) :339–414, 2003.

[Wer94] B. Werner. *Une Théorie des Constructions Inductives*. PhD thesis, Université Paris VII, France, 1994.

[Xi02] H. Xi. Dependent types for program termination verification. *Journal of Higher-Order and Symbolic Computation*, 15(1) :91–131, 2002.

[Zan95] H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24 :89–105, 1995.