Divergence des systèmes de réécriture et schématisation des ensembles infinis de termes

Miki HERMANN

Remerciements

Je tiens à remercier ici :

Laurent Kott qui a accepté les deux taches de président du jury et de rapporteur.

Pierre Lescanne de m'avoir acueilli au sein de son équipe. Ses remarques et critiques m'ont été très utiles pendant la rédaction de cette habilitation.

Adam Cichon et Jean-Pierre Jouannaud d'avoir accepté d'être rapporteurs de cette habilitation, et d'avoir assuré ce rôle avec autant d'efficacité que de diligence.

Harald Ganzinger, Jieh Hsiang et Hélène Kirchner pour m'avoir fait l'honneur de siéger dans mon jury.

Hélène Kirchner et Jieh Hsiang pour les nombreuses discussions sur les différentes sujets de mes travaux.

Isabelle Gnaedig avec qui je partage un bureau, ce qui l'a souvent obligé à subir mes élucubrations.

Enfin je remercie tous ceux qui m'ont aidé dans la mise au point du manuscrit, notamment dans l'amélioration de l'orthographe, je suis ainsi reconaissant à Jean-Michel Hufflen, Hélène Kirchner, Jean-Luc Rémy et Paul Zimmermann pour leurs efforts. Si des fautes restent malgré tout, elles ne peuvent qu'être attribuées à ma déplorable propension à tordre la syntaxe française.

Table des matières

Table des figures

x

Introduction

La logique équationnelle est depuis longtemps l'un des axes de la logique et devient aujourd'hui un des fondements de l'informatique théorique. Suscité par les travaux de Tarski et Herbrand, l'intérêt de la logique équationnelle provient de sa clarté, de sa commodité, de sa rigueur et surtout de son aisance naturelle à formaliser beaucoup de problèmes algébriques et logiques. En effet, presque tous les concepts mathématiques (groupes, monoïdes, anneaux, etc.) peuvent être définis équationnellement. Cette observation très importante permet d'effectuer des preuves sur les structures algébriques grâce au raisonnement équationnel. Redécouverte au milieu des années 70 lors des travaux sur les spécifications algébriques, la logique équationnelle a connu une deuxième vie grâce à l'informatique, remettant au goût du jour des concepts forgés par des logiciens comme Gentzen, Tarski et Robinson, dans les années 30 à 50. Depuis, de nombreux travaux ont considérablement enrichi ce domaine.

Les bases du raisonnement équationnelle sont les identités universellement quantifiées, appelées aussi *axiomes*. Les axiomes forment, avec la réflexivité, la symmètrie, la transitivité, la congruence et l'instantiation, les *théories équationnelles*. Chaque théorie équationnelle caractérise une relation d'équivalence sur les objets représentés par les termes. Le premier *problème* qu'on se pose est celui *du mot*: "étant donné deux objets, sont-ils égaux dans la théorie équationnelle engendrée?". Une autre question intéressante est celle de l'*unification*: "étant donné deux objets représentés par les termes, existe-t-il des instances qui sont égales dans la théorie équationnelle donnée?". La théorie équationnelle, fondée sur le seul remplacement des égaux par des égaux, n'est pas la seule théorie à considérer. La *théorie inductive* ou la *théorie de la récurrence* contient toutes les égalités dont toutes les instances closes appartiennent à la théorie équationnelle engendrée par un même ensemble d'axiomes. Le troisième concept intéressant est celui de *théorème inductif* ou *théorème par récurrence*: "une égalité d'objets est-elle valide pour toutes leurs instances?".

Le principe du raisonnement appliqué pour résoudre les trois questions principales est celui du remplacement des égaux par des égaux. Cette notion a été formalisée d'abord en logique comme *paramodulation*. La paramodulation possède un inconvénient majeur, celui de l'explosion combinatoire pendant la recherche de la preuve d'égalité entre deux objets. Le remède consiste à orienter les équations (axiomes) en règles. Ainsi, l'application des équations est restreinte à une seule direction. Ainsi sont nés, les *systèmes de réécriture*. Mais, la production d'un système de réécriture n'est pas la panacée.

La première propriété d'un système de réécriture est sa *terminaison*: chaque calcul à partir d'un terme doit impérativement se terminer. Cette propriété est indécidable en général, car les systèmes de réécriture peuvent simuler les machines de Turing. Heureusement, il existe de nombreuses conditions suffisantes, les *ordres de terminaison*, qui guarantissent qu'un système

de réécriture est nœthérien. Ils font l'objet d'une vaste littérature.

Les systèmes de réécriture posent un autre problème: l'orientation des équations en règles empêche d'effectuer certains remplacements. Pour remédier à cet inconvénient, il faut *compléter* le système de réécriture, c'est-à-dire ajouter de nouvelles règles. Ce processus se déroule de la façon suivante: d'abord deux règles sont *superposées*, en produisant une nouvelle équation, qui est ensuite orientée en règle. Ce processus est répété jusqu'à ce qu'aucune nouvelle règle ne puisse être produite. Ceci constitue, grosso modo, la *procédure de complétion* de Knuth et Bendix [KB70, Hue81, BDH86, Bac91]. Si cette procédure s'arrête, elle produit un système de réécriture confluent. Les systèmes de réécriture confluents et nœthériens résolvent le problème du mot dans les théories équationnelles. Ils jouent aussi un rôle important dans l'unification en permettant de trouver un ensemble complet de solutions par le processus de *surréduction*. La procédure de complétion est aussi à la base de quelques méthodes de mécanisation de preuves par récurrence.

L'intérêt pratique de la logique équationnelle et de la procédure de complétion est limité par les résultats négatifs suivants. Il existe des théories équationnelles qui ne possèdent aucune axiomatisation finie — elle ne peuvent être engendrées qu'à partir d'un ensemble infini d'axiomes. L'exemple d'une telle théorie a été donné pour la première fois par Tarski. Par conséquent, il n'existe aucun système de réécriture fini pour ces théories. Il existe des théories équationnelles, avec une axiomatisation finie, qui sont indécidables — le problème du mot dans ces théories est en général indécidable. Ce résultat a été prouvé par Kleene, Post et Turing. Par conséquent, il n'existe aucun système de réécriture fini, nœthérien et confluent pour ces théories. Cependant, chaque théorie équationnelle avec une axiomatisation finie est semi-décidable. Après les deux résultats précédents, on aurait espéré que chaque théorie équationnelle décidable possède un système de réécriture fini, nœthérien et confluent qui décide le problème du mot. Pourtant, il existe des résultats négatifs même pour les théories décidables. Le théorie équationnelle décidable, engendrée par l'équation

$$abax = babx$$

présentée par Kapur et Narendran [KN85], n'a pas de système de réécriture fini et confluent. Néanmois, en élargissant la signature, nous pouvons produire un système de réécriture fini, nœthérien et confluent:

$$\begin{aligned} abx &\to cx \\ bcx &\to cax \\ acax &\to ccx \\ accx &\to ccbx \end{aligned}$$

Malgré sa puissance, la possibilité d'élargir la signature ne représente pas un moyen universel pour produire un système de réécriture fini et nœthérien. Siekmann et Szabó [SS82] ont prouvé que la théorie équationnelle engendrée par les axiomes d'associativité et d'idempotence ne peut pas avoir un système de réécriture (non-conditionnel) fini et confluent. Autrement, si l'on entre ce système d'axiomes dans une procédure de complétion en supposant que l'on sait orienter les équations produites pour maintenir la terminaison, la procédure va fatalement diverger.

La divergence est donc un phénomène auquel nous sommes confrontés très souvent en déduction automatique équationnelle. Sa détection constitue par conséquent un élément important des laboratoires de réécriture ou des autres outils de preuves équationnelles.

Dans le domaine de la divergence des systèmes de réécriture, les efforts se sont cristalisés d'abord sur la découverte de conditions suffisantes de divergence généralisant la notion de *règles croisées* introduite par Igor Prívara et moi-même en 1986 [HP85, HP86]. Un travail préliminaire sur les notions sous-jacentes, présenté aussi dans [Her89, Her90a], constitue les Sections 2.1 et 2.2 de cette monographie. Une théorie générale des systèmes de réécriture divergents, qui a été bâtie dans [Her90b], est présentée dans le Chapitre 2. En premier lieu, l'indécidabilité de la divergence est prouvée dans la Section 1.2. Puis des conditions suffisantes pour détecter la divergence sont introduites, plus précisement l'existence de *systèmes croisés en avant* et *en arrière*, dans la Section 2.3. On en déduit la forme générale des règles appartenant à un ensemble infini engendré par complétion. La dernière partie de ce chapitre considère l'impact de l'interréduction des règles en cours de complétion sur le phénomène de divergence.

Un large éventail d'exemples de divergence et leur relation avec des systèmes croisés ont été présentés dans [Her90c], ainsi que des méthodes empiriques permettant d'éviter la divergence et applicables au cours d'une session de complétion dans un laboratoire de réécriture. Ceux-ci sont présentés dans le Chapitre 3.

Lorsqu'une procédure de déduction automatique boucle en produisant une infinité d'objets, il est dans certains cas possible de schématiser cet ensemble infini engendré. Ces objets peuvent être des équations, des règles ou des substitutions, mais ils sont toujours représentables comme termes dans une signature appropriée. La *schématisation* présente un formalisme acceptable pour manipuler directement, par des moyens finis, ces ensembles infinis de termes. Ainsi, elle représente un moyen parfait pour manipuler les systèmes de réécriture infinis, engendrés par une complétion divergente. Outre ce lien étroit, avec la divergence, la schématisation est une problématique par elle-même et représente aujourd'hui un domaine autonome en déduction automatique, qui a gagné beaucoup d'intérêt ces derniers temps, comme en témoignent différents travaux sur ce sujet. Le chapitre 4 présente cinq schématisations de termes. Ce sont les *méta-règles* [Kir89], les *schémas de termes* [Gra88], les *domaines de récurrence* [CHK90, CH91a, CH91b], les systèmes de réécriture avec *contraintes d'appartenance* [Com92] et les *grammaires primales* [Her92].

Notre intérêt s'est porté surtout sur les méta-régles, dont la méthode de constructions systématique à partir des systèmes croisés a été présentée dans [KH90] et constitue la Section 4.1. Nous avons aussi mis au point le concept de grammaire primale, qui est largement abordé dans les Sections 5.1 et 5.2. Les grammaires primales, définies tout d'abord dans [Her92] et présentées ici dans la Section 5.1, sont une nouvelle schématisation, plus puissante que les schématisations récurrentes précédentes. Comme ce concept est l'une des principales contributions de ce document, nous allons tenter d'en donner une approche intuitive. Tout d'abord, l'un des défis posés par la schématisation est le suivant: si la formalisme est trop faible on ne peut rien modéliser d'intéressant, si la formalisation est trop forte les principaux problèmes, l'unification notamment, deviennent indécidables. Il faut donc trouver un juste milieu. En arithmétique, le problème est bien connu. L'arithmétique rudimentaire avec zéro et successeur est trop faible, celle de Péano avec $+$ et $\times$ est trop forte dans le sens oú

Matijasevič par sa preuve de l'indécidabilité du 10^e problème de Hilbert a montré que l'unification y est indécidable. L'arithmétique de Presburger avec essentiellement 0, sucesseur et $+$, est décidable. Elle nous servira de modèle. Il nous faut créer une schématisation à la Presburger. Pour cela notre idée est d'engendrer les ensembles infinis par des principes récursifs assez contraints les *systèmes de réécriture primitives récursifs* (*systèmes de Presburger*) qui jouent le rôle de reproducteurs du principe de base avec les variations que la réécriture permet. Les points de départ de cette reproduction sont les *générateurs*. Le couple système de réécriture Presburger – générateur forme ce que nous appelons une *grammaire primale*. Malgré une puissance de schématisation très grande, l'unification des grammaires primales reste décidable. J'ai pu prouver que la classe des grammaires primales correspond exactement à la classe des systèmes croisés. L'algorithme d'unification a été présenté dans [GH92, HG93] et constitue, après une révision profonde, la Section 5.2.

Le document s'achève par une conclusion qui donne un état de l'art des autres travaux présentant un lien avec la divergence.

0.1 Notions et définitions de base

Comme tout repose sur la réécriture, il nous faut maintenant rappeler les concepts de base de cette approche et parler de termes, de substitutions, d'équations, de règles et d'ordre de réduction. C'est ce que nous allons faire dans les paragraphes qui suivent.

Ce paragraphe contient les notions et définitions de base utilisées dans ce mémoire et conformes aux notations introduites par Dershowitz et Jouannaud [DJ91]. Le lecteur intéressé par les aspects théoriques peut consulter d'autres ouvrages, par exemple [DJ90, Lal90, Bac91].

Soit $\mathcal{F}$ un ensemble fini ou dénombrable de *symboles fonctionnels* indicé par leur arité. On appelle $\mathcal{F}$ une *signature* non-typée. $\mathcal{F}_0$ dénote les constantes, $\mathcal{F}_1$ les symboles d'arité 1, et ainsi de suite. Ainsi, la fonction $ar: \mathcal{F} \longrightarrow N$ indique l'arité d'un symbole appartenant à $\mathcal{F}$. Soit $\mathcal{X}$ un ensemble dénombrable de *variables*, tel que $\mathcal{F} \cap \mathcal{X} = \emptyset$. Notons par $\mathcal{T}(\mathcal{F}, \mathcal{X})$ l'ensemble de tous les *termes* "bien formés" à partir des variables $\mathcal{X}$ et des symboles $\mathcal{F}$. L'ensemble $\mathcal{T}(\mathcal{F}, \mathcal{X})$ est construit par recurrence:

- toutes les variables appartiennent à $\mathcal{T}(\mathcal{F}, \mathcal{X})$: $\mathcal{X} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$;
- toutes les constantes appartiennent à $\mathcal{T}(\mathcal{F}, \mathcal{X})$: $\mathcal{F}_0 \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$;
- si les termes $t_1, \ldots, t_n$ appartiennent à $\mathcal{T}(\mathcal{F}, \mathcal{X})$ et $ar(f) = n$ pour un symbole $f \in \mathcal{F}$, alors le terme $f(t_1, \ldots, t_n)$ appartient à $\mathcal{T}(\mathcal{F}, \mathcal{X})$.

L'ensemble $\mathcal{T}(\mathcal{F}, \mathcal{X})$ forme l'*algèbre libre* construite sur la signature $\mathcal{F}$. Le symbole à la racine du terme t est noté $\mathcal{H}ead(t)$. Notons $\mathcal{V}ar(t)$ l'ensemble de toutes les variables dans le terme t. Un terme sans occurrence multiple de variables est dit *linéaire*. $\mathcal{G}(\mathcal{F})$ est l'ensemble de tous les *termes clos* (sans variables) construits à partir des symboles $\mathcal{F}$.

Soit N^* l'ensemble de toutes les chaînes d'entiers naturels y compris la chaîne vide notée Λ. La concaténation est une opération binaire sur N^*. En utilisant les éléments de N^* comme étiquettes, les termes peuvent être considérés comme des arbres étiquetés. Un terme t

est, dans ce cadre, une fonction partielle $N^* \longrightarrow \mathcal{F} \cup \mathcal{X}$ telle que son domaine $\mathcal{P}os(t)$ satisfait ces propriétés:

- si $t \in \mathcal{F}_0 \cup \mathcal{X}$ alors $\mathcal{P}os(t) = \{\Lambda\}$,
- si $t = f(t_1, \ldots, t_n)$ alors $\mathcal{P}os(t) = \{\Lambda\} \cup \bigcup_{i=1}^{n} \{i.a \mid a \in \mathcal{P}os(t_i)\}$

$\mathcal{P}os(t)$ est l'ensemble des *positions* du terme t. Le sous-ensemble des positions fonctionnelles (non-variables) de t est noté $\mathcal{FP}os(t)$, celui des positions occupées par des variables de t sont notées $\mathcal{VP}os(t)$. L'expression $a \leq b$ signifie que la position a se trouve *au-dessus* de la position b; autrement dit a est un préfixe de b. L'expression $a \parallel b$ signifie que les positions a et b sont incomparables (ou *parallèles*), autrement dit elles n'ont pas de préfixe commun.

Le *sous-terme* de t à la position $a \in \mathcal{P}os(t)$ est noté $t|_a$. Si $t = f(t_1, \ldots, t_n)$ alors $t|_\Lambda = t$ et $t|_{ia} = t_i|_a$ pour chaque $i = 1, \ldots, n$. Notons $s[t]_a$ un nouveau terme obtenu à partir du terme s par remplacement (greffe) de son sous-terme $s|_a$ par t: $s[t]_\Lambda = t$ et $f(s_1, \ldots, s_n)[t]_{i.a} = f(s_1, \ldots, s_i[t]_a, \ldots, s_n)$. Notons $s[\cdot]_a$ le *contexte* de s avec un "trou" à la position a. Le principe du remplacement (de la greffe) et du contexte s'élargit naturellement aux ensembles des positions A mutuellement parallèles. Les propriétés du remplacement sont étudiées dans l'article de Rosen [Ros73].

Une *substitution* est une fonction $\sigma: \mathcal{X} \longrightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$ telle que $x\sigma \neq x$ est valide pour un sous-ensemble fini de variables. La substitution σ notée $[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$ est la substitution telle que $x_i\sigma = t_i$, habituellement $t_i \neq x_i$, pour $i = 1, \ldots, n$. La substitution vide est notée $[\,]$. Les substitutions sont étendues homomorphiquement sur les termes comme suit:

- $f\sigma = f$ si $f \in \mathcal{F}_0$,
- $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$.

Notons respectivement $\mathcal{D}om(\sigma)$, $\mathcal{VR}an(\sigma)$, $\mathcal{V}ar(\sigma)$ et $\mathcal{TR}an(\sigma)$ le *domaine* (les variables remplacées par la substitution) $\mathcal{D}om(\sigma) = \{x \in \mathcal{X} \mid x\sigma \neq x\}$, le *codomaine des variables* (les nouvelles variables introduites par la substitution) $\mathcal{VR}an(\sigma) = \bigcup_{x \in \mathcal{D}om(\sigma)} \mathcal{V}ar(x\sigma)$, toutes les variables ($\mathcal{V}ar(\sigma) = \mathcal{D}om(\sigma) \cup \mathcal{VR}an(\sigma)$) et le *codomaine des termes* de la substitution σ $\mathcal{TR}an(\sigma) = \{x\sigma \mid x \in \mathcal{D}om(\sigma)\}$. Une substitution σ bijective sur les variables $\mathcal{X}$ s'appelle un *renommage des variables*. Les substitutions ne sont pas obligatoirement idempotentes dans notre approche.

Deux termes s et t sont *unifiables* si et seulement s'il existe une substitution idempotente σ telle que $s\sigma = t\sigma$. Une telle substitution σ s'appelle un *unificateur*. La substitution σ s'appelle l'*unificateur principal* de s et t si pour chaque unificateur φ de s et t il existe une substitution ψ telle que $\varphi = \sigma\psi$. La paire de substitutions $\langle \sigma_1, \sigma_2 \rangle$ est un *unificateur faible* [Ede85, Baa91] des termes s et t si $s\sigma_1 = t\sigma_2$. L'*unificateur principal faible* est une extension directe et naturelle de la notion d'unificateur principal. Il s'agit en fait de calculer la borne supérieure de s et t dans le treillis de termes pour l'ordre de filtrage [Hue80]. L'unificateur principal faible est le couple de substitutions, qui donne la borne supérieure.

La substitution σ est une *substitution dans les variables propres* de t si elle n'introduit pas des nouvelles variables, c-à-d $\mathcal{V}ar(t\sigma) \subseteq \mathcal{V}ar(t)$ et ne contient pas un renommage des variables. On peut étendre cette notion à un ensemble de substitutions.

Une *égalité* est une paire de termes $s \simeq t$ sans distinction d'une partie droite et d'une partie gauche ($s \simeq t$ est la même paire que $t \simeq s$). Une *règle de réécriture* est un couple de termes $s \to t$ telle que $\mathcal{V}ar(t) \subseteq \mathcal{V}ar(s)$. La règle $p^{op} = t \to s$ est l'*inverse* de la règle $p = s \to t$, sous condition que $\mathcal{V}ar(s) = \mathcal{V}ar(t)$. La règle $s \to t$ est *linéaire à gauche* si le terme s est linéaire. La règle $s \to t$ est *linéaire* si les deux termes s et t sont linéaires. Un *système de réécriture* est un ensemble fini de règles $R = \{s \to t \mid s,t \in \mathcal{T}(\mathcal{F},\mathcal{X})\}$. Le système de réécriture R, où $\mathcal{V}ar(s) = \mathcal{V}ar(t)$ est valide pour chaque règle $s \to t \in R$, s'appelle un système de réécriture *préservant les variables*. Le système, construit à partir d'un système préservant les variables par inversion des règles, est l'ensemble de règles $R^{op} = \{t \to s \mid s \to t \in R\}$, appelé le système *inverse* de R. Un système de réécriture est *linéaire à gauche* si chaque règle est linéaire à gauche. Un système de réécriture est *linéaire* si chaque règle est linéaire. Le système de réécriture R est *monadique* s'il est construit à partir d'une signature $\mathcal{F} = \mathcal{F}_0 \cup \mathcal{F}_1$ qui contient uniquement des constantes et des symboles d'arité 1.

La *relation de réécriture* $\longrightarrow_R$ (ou simplement $\longrightarrow$ quand R est univoque) est la plus petite relation contenante R, stable par substitution et par remplacement. Plus précisément, les termes s et t sont reliés par $\longrightarrow_R$ s'il existe une règle $l \to r \in R$, une position $a \in \mathcal{FP}os(s)$ et une substitution σ, appelée *filtre*, telles que $s|_a = l\sigma$ et $t = s[r\sigma]_a$. La relation $\stackrel{*}{\longrightarrow}$ est la fermeture réflexive et transitive de $\longrightarrow$. La relation $\longleftarrow$ indique l'inverse de la relation $\longrightarrow$, c-à-d $s \longleftarrow t$ si et seulement si $t \longrightarrow s$. La relation d'équivalence $\stackrel{*}{\longleftrightarrow}$ est la fermeture réflexive, symétrique et transitive de la relation $\longrightarrow$.

Le terme t est *réductible* par la règle $l \to r$ s'il existe une position $a \in \mathcal{FP}os(t)$ et une substitution σ, telles que $t|_a = l\sigma$. Un terme est *R-réductible* s'il est réductible par une règle qui appartient au système R. Par contre, un terme est *R-irréductible* s'il n'est pas réductible par les règles de R. Un terme t est la *R-forme normale* du terme s si $s \stackrel{*}{\longrightarrow}_R t$ et t est R-irréductible. L'ensemble des R-formes normales du terme t est noté par $R(t)$. Une règle $l \to r \in R$ est *réduite* dans R si r est en R-forme normale et l est en forme normale par rapport à $R - \{l \to r\}$. Le système de réécriture R est *inter-réduit* si toutes ses règles sont réduites.

$\vec{a}$ est soit le vecteur $\langle a_1, \ldots, a_n \rangle$, soit la suite $a_1, \ldots, a_n$, soit l'ensemble $\{a_1, \ldots, a_n\}$. L'objet dont il s'agit est clair à partir du contexte dans lequel cette notation est appliquée. Ainsi, $\vec{f}(\vec{x})$ signifie $f_1(x_1, \ldots, x_k), \ldots, f_n(x_1, \ldots, x_k)$. Notons par $a_\top$ le dernier élément de $\vec{a}$.

Soit A un ensemble. Notons A^n le produit cartésien

$$A^n = \underbrace{A \times \cdots \times A}_{n \ fois}$$

De plus,

$$A^* = \bigcup_{n=0}^{\infty} A^n$$
$$A^+ = \bigcup_{n=1}^{\infty} A^n$$

sont la fermeture réflexive et transitive, (resp. transitive) de l'ensemble A.

0.2 Ordres de réduction

Dans ce paragraphe nous rappelons les notions sur la terminaison et les ordres. Une introduction détaillée se trouve dans [Der87, DJ90].

Un *ordre* (partiel et strict) $\succ$ est une relation irréflexive, anti-symétrique et transitive, sur un ensemble M. La structure $(M, \succ)$ s'appelle un ensemble (partiellement) ordonné. L'ordre $\succ$ est *bien fondé* s'il n'existe pas de chaîne descendante infinie $m_1 \succ m_2 \succ m_3 \succ \cdots$, où $\vec{m} \subseteq M$. La *précédence* est un ordre bien fondé sur une signature $\mathcal{F}$.

On s'intéresse principalement aux ordres sur les termes $\mathcal{T}(\mathcal{F}, \mathcal{X})$. L'ordre $\succ$ est *stable par substitution* si $s \succ t$ implique $s\sigma \succ t\sigma$ pour tous les termes $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ et toute substitution σ. L'ordre $\succ$ est *compatible avec la greffe* si la relation $s \succ t$ implique $u[s]_a \succ u[t]_a$ pour tous les termes $s, t, u \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. L'ordre $\succ$ est *compatible avec la* (relation de) *réécriture* $\longrightarrow_R$ si $\longrightarrow_R \subseteq \succ$. Un *ordre de réduction* est un ordre bien fondé, stable par substitution et compatible avec la greffe.

L'*ordre d'insertion propre* $\triangleright$ est un ordre sur les termes défini par $s \triangleright t$ si et seulement si un sous-terme propre de s est une instance de t. Nous avons besoin aussi d'un ordre particulier sur les règles de réécriture. Soit $\succ$ un ordre de réduction. L'ordre $\succ\!\!\!|$ basé sur $\succ$ est l'ordre bien fondé sur les règles, défini par $(s \to t) \succ\!\!\!| (l \to r)$ si et seulement si

- soit $s \triangleright l$,
- soit $s\rho = l$ pour un renommage ρ et $t \succ r$.

Soient $\succ_1$ et $\succ_2$ deux ordres, la *combinaison lexicographique* de ces deux ordres est un ordre sur les paires, défini par: $\langle s, t\rangle \succ \langle s', t'\rangle$ si soit $s \succ_1 s'$ soit $s = s'$ et $t \succ_2 t'$. La combinaison lexicographique de plusieurs ordres est définie de façon naturelle à partir de la combinaison de deux ordres. Un ordre lexicographique est bien fondé si et seulement si tous ses composantes sont des ordres bien fondés.

Un *multi-ensemble* basé sur un ensemble S est une fonction $M\colon S \longrightarrow \mathcal{N}$ de l'ensemble S dans les entiers naturels. On dit que x appartient à M ($x \in M$) si $M(x) > 0$. L'*union* et l'*intersection* des multi-ensembles sont définies ainsi:

- $[M_1 \cup M_2](x) = M_1(x) + M_2(x)$,
- $[M_1 \cap M_2](x) = \min(M_1(x), M_2(x))$.

Le multi-ensemble M est fini si l'ensemble $\{x \in S \mid M(x) > 0\}$ est fini.

N'importe quel ordre $\succ$ sur un ensemble S peut être étendu à un ordre $\succ_{mul}$ sur les multi-ensembles (finis) basés sur S par la définition suivante: $M \succ_{mul} M'$ si

1. $M \neq M'$ et
2. quand $M'(x) > M(x)$, alors il existe un élément $y \succ x$, tel que $M(y) > M'(y)$.

Autrement dit, chaque élément d'un multi-ensemble peut être remplacé par un nombre fini d'éléments plus petits. Dershowitz et Manna [DM79] ont prouvé que l'ordre $\succ_{mul}$ sur les multi-ensembles finis est bien fondé si et seulement si l'ordre $\succ$ est bien fondé.

A un symbole f d'arité n on peut associer un *statut*, i.e. une extension d'un ordre $\succ$ sur les termes à un ordre $\succ^f$ sur les n-tuples de termes. En particulier, le symbole f possède le *statut multi-ensemble* si $\succ^f$ est défini par $\vec{s} \succ^f \vec{t}$ si $\vec{s} \succ_{mul} \vec{t}$, tandis qu'il possède le *statut lexicographique gauche-droite* (*droite-gauche*) quand $\vec{s} \succ^f \vec{t}$ est défini par

- $\langle s_1, \ldots, s_n \rangle \succ_{lex} \langle t_1, \ldots, t_m \rangle$ pour gauche-droite;
- $\langle s_n, \ldots, s_1 \rangle \succ_{lex} \langle t_m, \ldots, t_1 \rangle$ pour droite-gauche.

La notation $\succ_{lex}$ désigne la combinaison lexicographique de n copies de l'ordre $\succ$.

Soit $\succ$ une précédence sur la signature $\mathcal{F}$ et supposons que chaque symbole $f \in \mathcal{F}$ possède soit le statut multi-ensemble soit le statut lexicographique. L'*ordre récursif sur les chemins* $\succ_{rpo}$ correspondant est défini récursivement par

$$s = f(\vec{s}) \quad \succ_{rpo} \quad g(\vec{t}) = t$$

si $\forall t_i \in \vec{t}$ nous avons $s \succ_{rpo} t_i$ et une des possibilités suivantes est valide:

1. $\exists s_i \in \vec{s}$ tel que $s_i \succ_{rpo} t$ ou $s_i = t$,
2. $f \succ g$,
3. $f \equiv g$ et $\vec{s} \succ^f_{rpo} \vec{t}$.

L'*ordre multi-ensemble sur les chemins*, noté $\succ_{mpo}$, est un ordre récursif sur les chemins où chaque symbole f possède le statut multi-ensemble. De façon analogue, l'*ordre lexicographique sur les chemins*, noté $\succ_{lpo}$, est un ordre récursif sur les chemins où chaque symbole f possède le statut lexicographique.

0.3 Systèmes de réécriture

Les définitions suivantes rappellent les notions de base dans le domaine des systèmes de réécriture.

Un système de réécriture R (et aussi la relation de réécriture $\longrightarrow_R$) est

- **nœthérien** si la relation de réécriture $\longrightarrow_R$ est incluse dans un ordre de réduction $\succ$ (on dit aussi que R **termine**),
- **confluent** si $\overset{*}{\longleftarrow}_R \,.\, \overset{*}{\longrightarrow}_R \;\subseteq\; \overset{*}{\longrightarrow}_R \,.\, \overset{*}{\longleftarrow}_R$,
- **convergent** s'il est confluent et nœthérien,
- **canonique** s'il est convergent et inter-réduit.

Les questions sur la confluence sont abordées dans [Hue80].

La *forme normale* d'un terme t par rapport à une relation de réécriture nœthérienne $\longrightarrow_R$ est notée $t{\downarrow_R}$. Une preuve P est une suite de termes. Une *preuve équationnelle* P de $s = t$ dans R est $P = s \overset{*}{\longleftrightarrow}_R t$. Une *preuve par réécriture* P de $s = t$ dans R est $P = s \overset{*}{\longrightarrow}_R \,.\, \overset{*}{\longleftarrow}_R t$.

La notion de *paire critique* comme des instances de superposition des règles de réécriture a été introduite par Knuth et Bendix [KB70]. La production et l'orientation de ces paires en de nouvelles règles constitue la colonne vertébrale de la procédure de complétion de Knuth et Bendix.

Définition 0.1 *Soient $s_1 \to t_1$ et $s_2 \to t_2$ deux règles de réécriture telles que $s_1|_a\sigma = s_2\sigma$ est valide pour l'unificateur principal σ et la position $a \in \mathcal{FP}os(s_1)$ (nous disons que telles règles sont superposantes, avec $s_1 \to t_1$ comme la règle* majeure *et $s_2 \to t_2$ comme la règle* mineure*). La paire $\langle s_1\sigma[t_2\sigma]_a, t_1\sigma\rangle$ s'appelle une* **paire critique** *de termes. L'ensemble de toutes les paires critiques produites à partir du système de réécriture R est noté $cp(R)$. La paire critique $\langle t, t\rangle$, pour n'importe quel terme t, est dite* **triviale**.

Chapitre 1

Divergence des systèmes de réécriture

1.1 Procédure de complétion

Etant donné un ensemble fini E d'égalités et l'ordre de reduction $\succ$, la *procédure de complétion* déduit des conséquences équationnelles de E dans l'espoir de trouver en un temps fini un système de réécriture convergent R_∞ équivalent dans le sens que les théorèmes de R_∞ sont les théorèmes de E. Bachmair, Dershowitz et Hsiang [BDH86, Bac91] ont placé la complétion dans un cadre abstrait, basé sur la notion de *règles de transition* (voir aussi [DJ90]). Une *règle de transition* (dans le cadre de notre étude) est une relation binaire entre couples $(E; R)$, où E est un ensemble d'égalités et R est un ensemble de règles de réécriture. La procédure de complétion de Knuth et Bendix est fondée sur ces six règles de transition:

$$
\begin{array}{lrll}
\textit{Deduce:} & (E;R) & \vdash (E \cup \{s \simeq t\};R) & \text{si } s \simeq t \in cp(R) - E \\
\textit{Delete:} & (E \cup \{s \simeq s\};R) & \vdash (E;R) & \\
\textit{Simplify:} & (E \cup \{s \simeq t\};R) & \vdash (E \cup \{u \simeq t\};R) & \text{si } s \longrightarrow_R u \\
\textit{Orient:} & (E \cup \{s \simeq t\};R) & \vdash (E;R \cup \{s \to t\}) & \text{si } s \succ t \\
\textit{Compose:} & (E;R \cup \{s \to t\}) & \vdash (E;R \cup \{s \to u\}) & \text{si } t \longrightarrow_R u \\
\textit{Collapse:} & (E;R \cup \{s \to t\}) & \vdash (E \cup \{u \simeq t\};R) & \text{si } s \longrightarrow_R u \text{ par } l \to r \in R \\
 & & & \text{avec } (s \to t) \rhd (l \to r)
\end{array}
$$

Les règles *Compose*, *Collapse*, *Simplify*, *Delete* sont dites "simplificatrices" ou "destructives". Les règles *Deduce* et *Orient* sont dites "créatives". L'ordre $\rhd$ est l'ordre bien fondé sur les règles, où $(s \to t) \rhd (l \to r)$ si et seulement si

- soit $s \triangleright l$,
- soit $s\rho = l$ pour un renommage ρ et $t \succ r$.

On écrit $(E; R) \vdash_{KB} (E'; R')$ si le dernier couple est obtenu à partir du premier par application d'une règle de transition de KB. $\vdash^+_{KB}$ (resp. $\vdash^*_{KB}$) est la fermeture réflexive (resp. réflexive-transitive) de $\vdash_{KB}$.

La **procédure de complétion** est une *stratégie* d'application des règles de transition KB en utilisant un ordre de réduction compatible avec ces règles. Le résultat d'une suite

program *complete*(E) **is**
begin
  while $\exists s\{s \simeq s \in E\}$ **do** *Delete* **od**;
  while $E \neq \emptyset$ **do**
    if $\exists s,t\{s \simeq t \in E \wedge (s \succ t \vee t \succ s)\}$ **then**
      while $\exists s,t\{s \simeq t \in E \wedge (s \succ t \vee t \succ s)\}$ **do**
        Orient;
        while $\exists s,t,u\{s \to t \in R \wedge s \longrightarrow_R u$
            par $l \to r \in R$ avec $s \triangleright l\}$ **do** *Collapse* **od**;
        while $\exists s,t,u\{s \to t \in R \wedge t \longrightarrow_R u\}$ **do** *Compose* **od**;
        while $\exists s,t,u\{s \simeq t \in E \wedge s \longrightarrow_R u\}$ **do** *Simplify* **od**;
        while $\exists s\{s \simeq s \in E\}$ **do** *Delete* **od**
      od
    else *fail* **fi**;
    while $\exists s,t\{s \simeq t \in cp(R) - E\}$ **do** *Deduce* **od**;
    while $\exists s,t,u\{s \simeq t \in E \wedge s \longrightarrow_R u\}$ **do** *Simplify* **od**;
    while $\exists s\{s \simeq s \in E\}$ **do** *Delete* **od**
  od
end

FIG. 1.1 – *Stratégie de complétion génerale:* complete

de complétion (potentiellement infinie) $(E_0; R_0) \vdash_{KB} (E_1; R_1) \vdash_{KB} \cdots$ est l'ensemble $E_\infty = \bigcup_{i\geq 0} \bigcap_{j>i} E_j$ des égalités dites *persistantes* et l'ensemble $R_\infty = \bigcup_{i\geq 0} \bigcap_{j>i} R_j$ des règles dites *persistantes*. La persistance des égalités et des règles s'exprime par l'impossibilité de leur "destruction" par les règles de transition dites destructrices.

Ces règles de transition sont *consistantes* (la plus petite relation engendrée par $E \cup R$ n'est pas modifiée pendant leurs application). En plus, la stratégie de complétion basée sur les règles de transition doit être *équitable* (le traitement de chaque paire critique et l'orientation de chaque égalité ne peuvent pas être évités indéfiniment), *juste* (la stratégie ne doit pas tomber en échec s'il existe un moyen d'éviter cet échec) et *correcte* (lorsque la procédure finit avec succès, elle produit un système de réécriture convergent). Les définitions formelles de ces notions se trouvent dans [Der89] et [Her91].

Il existent, bien sûr, une multitude de stratégies de complétion, basées sur les règles de transition KB, équitables et correctes. Dans notre cadre, nous ne considérons que deux stratégies. La première, *complete* dans la figure 1.1, est est une stratégie de complétion génerale.

La seconde procédure de complétion, *nr-complete* dans la figure 1.2, présente une stratégie sans réduction qui n'utilisent ni *Compose*, ni *Collapse*, ni *Simplify* et qui produit ainsi les conséquences de toutes les paires critiques sans inter-réduction.

Dans les deux cas, on peut montrer que le processus de complétion est consistant, équitable, juste et correct [Her91].

program *nr-complete*(E) **is**
begin
  $E' \leftarrow \emptyset$;
  while $\exists s\{s \simeq s \in E\}$ **do** *Delete* **od**;
  while $E \neq \emptyset$ **do**
    if $\exists s,t\{s \simeq t \in E \wedge (s \succ t \vee t \succ s)\}$ **then**
      while $\exists s,t\{s \simeq t \in E \wedge (s \succ t \vee t \succ s)\}$ **do** *Orient* **od**
    else *fail* **fi**;
    while $\exists s,t\{s \simeq t \in cp(R) - E\}$ **do** *Deduce* **od**;
    if $E = E'$ **then** *stop*
      else $E' \leftarrow E$
    fi;
    while $\exists s\{s \simeq s \in E\}$ **do** *Delete* **od**
  od
end

FIG. 1.2 – *Stratégie sans réduction:* nr-complete

La procédure de complétion peut s'arrêter avec *succès* en produisant un système de réécriture R_∞ fini et convergent ou canonique. Elle peut *échouer* à cause d'une égalité non-orientable par l'ordre de réduction $\succ$, ou bien **diverger** en produisant un système de réécriture R_∞ infini. Dans le Chapitre 2 nous traitons des problèmes de divergence en déduisant des conditions suffisantes pour leur détection.

Les règles de transition qui effectuent la réduction sont primordiales dans le processus de complétion. Sans elles presque chaque système de réécriture deviendrait divergent. D'autre part, l'élimination des règles simplificatrices facilite le développement des schémas de divergence. Pour cette raison je développe d'abord dans le Chapitre 2 la théorie des systèmes divergents pour la stratégie *nr-complete*. La détermination de la divergence d'un système de réécriture, même si des règles simplificatrices sont appliquées, est étudiée ensuite; elle exploite les résultats concernant la stratégie générale.

1.2 Systèmes de réécriture divergents

Pour un ensemble de règles R (ou égalités E) donné, le comportement de la procédure de complétion dépend de la stratégie appliquée, ainsi que de l'ordre $\succ$ utilisé pour orienter les égalités en règles.

Définition 1.1 *Soit R un système de réécriture. R est* **divergent** *pour l'ordre $\succ$ si l'ensemble complete(R) est infini. R est* **faiblement divergent** *pour l'ordre $\succ$ si l'ensemble nr-complete(R) est infini. R est* **divergent** *(resp.* **faiblement divergent***) de façon* **absolue** *si R est divergent (resp. faiblement divergent) pour chaque ordre qui inclut la relation $\xrightarrow{*}_R$.*

La divergence absolue d'un système de réécriture dépend de la structure de ses règles. Chaque système de réécriture divergent est aussi faiblement divergent. Le rapport entre les systèmes divergents et faiblement divergents est établi par la proposition suivante, dont la preuve est évidente.

Proposition 1.2 *Soit R un système de réécriture faiblement divergent et soit $R' \subseteq R$ un sous-système du premier, lui aussi faiblement divergent. Si nr-complete$(R') \subseteq$ complete(R), alors R est divergent.*

1.2.1 Théories équationnelles et décidabilité

Il est clair que les théories équationnelles ayant un problème du mot indécidable ne peuvent pas être complétées en des systèmes de réécriture canoniques et finis. La situation est même pire, car il existe des théories équationnelles avec un problème du mot décidable, pour lesquels il n'existe aucun système de réécriture canonique et fini équivalent. L'un d'eux est $a(b(a(x))) = b(a(b(x)))$ et a été mis en évidence par Kapur et Narendran [KN85].

1.2.2 L'indécidabilité de la divergence

Théorème 1.3 *La divergence d'un système de réécriture, même linéaire, est indécidable.*

Preuve: Comme prouvé par Dershowitz [Der87], Dauchet [Dau88, Dau89] ou Huet avec Lankford [HL78], les machines de Turing peuvent être simulées par les systèmes de réécriture. Soit $DTM = (Q, \Sigma, ?, \delta, q_0, F)$ une machine de Turing déterministe avec une bande infinie à droite, où

Q est l'ensemble fini des états,
Σ est l'alphabet d'entrée,
$? - \Sigma = \{\beta\}$ est le symbole d'espace,
$\delta: Q \times ? \longrightarrow Q \times \Sigma \times \{-1, 0, 1\}$ est la fonction de transition,
$q_0 \in Q$ est l'état initial,
$F \subseteq Q$ est l'ensemble d'états finals.

On impose de plus que la machine de Turing ne peut pas écrire le symbole d'espace, qu'elle ne peut pas sortir de la bande à gauche, et qu'elle ne répète aucune description instantanée pendant son calcul. On peut transformer chaque machine de Turing en une autre machine de Turing, équivalente à la première, qui satisfait ces restrictions [HU79].

On va construire le système de réécriture R_{DTM} qui simule la machine de Turing DTM. Les symboles $Q \cup \Sigma$ sont interprétés comme les symboles fonctionnels monadiques et le symbole d'espace β comme constante (on supprime les parenthèses dans les termes monadiques). Nous introduisons quatre symboles supplémentaires: f, g, et les constantes \$ et $-$. Le symbole f est utilisé pour inhiber les superpositions indésirables, le symbole g pour assurer la terminaison du système, le symbole \$ est la marque gauche de la bande et $-$ est un symbole

quelconque. Pour chaque élément de la fonction de transition δ le système R_{DTM} contient une ou plusieures règles de réécriture. Le tableau suivant indique cette correspondance:

$$\begin{array}{lll}
\delta(q,a) = (p,b,0) & f(x,qay,gz) \to f(x,pby,z) & \\
\delta(q,\beta) = (p,b,0) & f(x,q\beta,gy) \to f(x,pb\beta,y) & \\
\delta(q,a) = (p,b,1) & f(x,qay,gz) \to f(bx,py,z) & \\
\delta(q,\beta) = (p,b,1) & f(x,q\beta,gy) \to f(bx,p\beta,y) & \\
\delta(q,a) = (p,b,-1) & f(sx,qay,gz) \to f(x,psby,z) & \text{pour tout } s \in \Sigma \\
\delta(q,\beta) = (p,b,-1) & f(sx,q\beta,gy) \to f(x,psb\beta,y) & \text{pour tout } s \in \Sigma
\end{array}$$

où $p,q \in Q$, $a,b \in \Sigma$ et $x,y,z \in \mathcal{X}$.

Il n'y a pas de superpositions entre les règles du système R_{DTM} parce que la machine de Turing DTM est déterministe et que le symbole f apparaît seulement à la racine des parties gauches des règles. Il n'y a pas de possibilité d'appliquer la composition parmi les règles de R_{DTM} à caude de la présence du symbole g dans les parties gauches. Le système R_{DTM} est linéaire à gauche et sans superpositions, ce qui implique qu'il est canonique.

Soit $w \in \Sigma^*$ un mot d'entrée et w^r l'image miroir de w. On ajoute la règle $p_0(w) = f(\$, q_0 w\beta, y) \to -$ pour w. Soit $R = R_{DTM} \cup \{p_0(w)\}$. Il est clair que le système de réécriture R est nœthérien.

L'idée principale est d'utiliser le processus de superposition et simuler ainsi les descriptions instantanées successives pendant le calcul de la machine DTM sur l'entrée w. Soit $\mathcal{A}_{DTM}(w) = q_0 w \vdash \cdots \vdash w_1 q w_2 \vdash \cdots$ un calcul (potentiellement infini) de la machine DTM sur le mot w. On va prouver par induction que $complete(R)$ contient la règle

$$p_k(w) \quad = \quad f(w_1^r\$, qw_2\beta, y) \to -$$

si et seulement si la suite $q_0 w \vdash^k w_1 q w_2$ constitue les k premiers pas du calcul $\mathcal{A}_{DTM}(w)$, c.-à.-d. toutes les règles $p_k(w)$ sont persistantes pendant la complétion. La preuve par induction est effectuée sur le nombre k de pas de la suite $q_0 w \vdash^k w_1 q w_2$.

Base: $k = 0$. La description instantanée de départ $q_0 w$ correspond à la règle $p_0(w) = f(\$, q_0 w\beta, y) \to -$, qui est déjà incluse dans R. La règle $p_0(w)$ est irréductible par R_{DTM}, car $p_0(w)$ ne contient pas le symbole g, et la règle $p_0(w)$ ne réduit pas des règles de R_{DTM} car R_{DTM} ne contient pas de symbole \$.

Pas: Soient $q_0 w \vdash^k v_1 p v_2$ les k premiers pas du calcul $\mathcal{A}_{DTM}(w)$. Par hypothèse, $complete(R)$ contient $p_k(w) = f(v_1^r\$, pv_2\beta, y) \to -$. Supposons que le pas $v_1 p v_2 \vdash w_1 q w_2$ soit effectué par la transition $\delta(p,a) = (q,b,m)$, où m présente le déplacement de la tête de la machine DTM. Cela implique que la règle $p_k(w)$ et une seule des règles de R_{DTM} se superposent, produisant la nouvelle règle

$$p_{k+1}(w) \quad = \quad f(w_1^r\$, qw_2\beta, y) \to -$$

Il n'y a pas d'autres superpositions possibles car la machine DTM est déterministe et le symbole f ne permet d'effectuer les superpositions qu'à la racine. De plus, la nouvelle règle $p_{k+1}(w)$ ne peut pas réduire une des règles du système R_{DTM} car ces dernières ne contiennent

pas le symbole \$, de même les règles du système R_{DTM} ne peuvent pas réduire la nouvelle règle $p_{k+1}(w)$ car $p_{k+1}(w)$ ne contient pas le symbole g. Il n'existe pas d'indice $l < k+1$ tel que

$$p_l(w) \quad = \quad f(w_1^r\$, qw_2\beta, y) \to -$$

car la machine DTM ne repète pas la même description instantanée deux fois. Ceci signifie que la nouvelle règle $p_{k+1}(w)$ ne peut pas réduire les règles produites pendant les étapes précédentes. Ceci prouve que

$$complete(R) \quad = \quad R_{DTM} \cup \{p_k(w) \mid k \in N\}$$

Donc le problème de l'arrêt des machines de Turing se réduit au problème de la divergence des systèmes de réécriture. □

Cette preuve est une modification de la méthode utilisée par Paliath Narendran et Johnatan Stillman [NS89]. Avec un codage plus sophistiqué de la fonction de transition δ dans la preuve précédente il est possible de prouver cette indécidabilité aussi pour les systèmes de réécriture monadiques, mais dans ce cas la preuve est plus compliquée et mois élégante. Le seule symbole non-monadique dans la preuve du Théorème 1.3 est le symbole f, qui empèche les superpositions indésirables. Si on efface le symbole f et son troisième argument, en fusionant ses deux premiers arguments en un seul terme, on obtient ainsi un système de réécriture monadique. Il reste alors la question des superpositions indésirables, mais chaque superposition indésirable produit une nouvelle règle $l \to r$ avec au mois deux symboles d'état dans la partie gauche l, ce qui ne correspond plus a une description instantanée d'un calcul de la machine de Turing DTM. Il suffit simplement de ne pas tenir compte de ces règles supplémentaires. Chaque procédure de complétion équitable va produire pour ce nouveau système de réécriture un résultat équivalent à celui écrit dans la preuve précédante, plus les règles issues des superpositions indésirables.

La conclusion du Théorème 1.3 est qu'il est raisonable de chercher seulement des conditions suffisantes pour reconnaître la divergence des systèmes de réécriture. Le Chapitre 2 est consacré au développement des ces schémas de reconnaissance.

Chapitre 2

Moyens pour détecter la divergence

2.1 Opérations et opérateurs sur les substitutions

La divergence du processus de complétion peut donner lieu à plusieurs suites infinies de règles de réécriture. Cet ensemble infini de règles n'est pas produit de façon arbitraire, mais un lien précis entre ces suites de règles peut être observé. La relation entre deux règles dans une suite se manifeste aussi par la production de la deuxième à partir de la première par une substitution spécifique. Cet observation donne lieu à une suite infinie de substitutions qui interviennent au moment de la production de cette suite infinie de règles. Chaque substitution dans cette suite infinie est produite, comme nous allons le voir dans le paragraphe 2.3, comme une itération de certaines substitutions de base. Pour pouvoir exprimer ces itérations de substitutions, il est nécessaire de développer une algèbre sur les substitutions avec ses propres opérateurs.

Dans la première partie, l'opérations de *produit restreint* de substitutions est défini; puis suivent les preuves de ses propriétés. A partir du produit restreint, deux opérateurs (W- et T-opérateur) sont construits. Ces deux opérateurs sont définis par itération du produit restreint des substitutions de base. Comme démontré dans les parties suivantes, les propriétés itératives des opérateurs correspondent parfaitement aux constructions indispensables utilisées dans les preuves des propriétés spécifiques des chaînes de fermeture, ceci permet de plus, de développer, comme un objet itératif, la notion de chaîne de fermeture, qui est l'objet principal de la divergence.

2.1.1 Produit restreint des substitutions

Définition 2.1 *Soit φ et ψ deux substitutions quelconques. Le* **produit restreint** *de φ et ψ est la substitution*

$$\varphi \triangle \psi \quad = \quad (\varphi\psi)|_{\mathcal{D}om(\varphi)}$$

Remarquons qu'on peut écrire la composition des substitutions $\varphi\psi$ par

$$(\varphi \triangle \psi) \cup [x \mapsto x\psi \mid x \in \mathcal{D}om(\psi) - \mathcal{D}om(\varphi)]$$

Rappelons que le domaine $\mathcal{D}om(\varphi)$ d'une substitution φ est l'ensemble fini des variables $x \in \mathcal{X}$ telles que $x\varphi \neq x$. On peut alors produire des cas où $\mathcal{D}om(\varphi \triangle \psi) \neq \mathcal{D}om(\varphi)$ dans le cas où certaines variables sont simplement renommées. Prenons par exemple $\varphi = [x \mapsto f(u), y \mapsto v]$ et $\psi = [u \mapsto g(x), v \mapsto y, w \mapsto h(z)]$; le produit restreint de φ et ψ sera la substitution $\varphi \triangle \psi = [x \mapsto f(g(x))]$. Il faut bien noter que $y \mapsto y$ n'appartient pas à ce produit restreint !

En général, le produit restreint des substitutions n'est ni commutatif ni associatif. Les autres propriétés sont présentées dans le lemme suivant:

Lemme 2.2 *Soient φ, ψ et σ des substitutions.*

1. *$\varphi\psi = \varphi \triangle \psi$ si et seulement si $\mathcal{D}om(\psi) \subseteq \mathcal{D}om(\varphi)$, en particulier dès lors que $\mathcal{D}om([\,]) = \emptyset \subseteq \mathcal{D}om(\varphi)$, $\varphi \triangle [\,] = \varphi[\,] = \varphi$.*

2. *$\varphi \triangle \psi = \varphi$ si et seulement si $\mathcal{D}om(\psi) \cap \mathcal{VR}an(\varphi) = \emptyset$.*

3. *$(\psi \triangle \varphi) \triangle \sigma = \psi \triangle (\varphi \triangle \sigma)$ si $\mathcal{D}om(\sigma) \subseteq \mathcal{D}om(\varphi)$ et σ, φ, ψ ne contiennent pas de renommage de variables.*

Preuve:

1. Nous nous servons de l'égalité $\varphi\psi = (\varphi \triangle \psi) \cup [x \mapsto x\psi \mid x \in \mathcal{D}om(\psi) - \mathcal{D}om(\varphi)$.

   Si $\varphi\psi = \varphi \triangle \psi$, alors $\mathcal{D}om(\psi) - \mathcal{D}om(\varphi) = \emptyset$, ce qui implique l'inclusion $\mathcal{D}om(\psi) \subseteq \mathcal{D}om(\varphi)$.

   Si $\mathcal{D}om(\psi) \subseteq \mathcal{D}om(\varphi)$ alors $\mathcal{D}om(\psi) - \mathcal{D}om(\varphi) = \emptyset$, ce qui implique l'équivalence $\varphi\psi = \varphi \triangle \psi$.

2. Si $\varphi \triangle \psi = \varphi$ alors $\forall x \in \mathcal{D}om(\varphi)$: $x\varphi\psi = x\varphi$, ce qui implique $\mathcal{D}om(\psi) \cap \mathcal{VR}an(\varphi) = \emptyset$.

   Si $\mathcal{D}om(\psi) \cap \mathcal{VR}an(\varphi) = \emptyset$, l'égalité $\varphi \triangle \psi = \varphi$ est impliqué directement à partir de la définition du produit restreint.

3. Si $\psi = [x \mapsto y]$, $\varphi = [x \mapsto f(z), y \mapsto x]$ et $\sigma = [x \mapsto g(z)]$, alors $(\psi \triangle \varphi) \triangle \sigma = [\,]$, tandis que $\psi \triangle (\varphi \triangle \sigma) = [x \mapsto g(z)]$ malgré la validité de l'inclusion $\mathcal{D}om(\sigma) \subseteq \mathcal{D}om(\varphi)$.

□

Le lemme suivant prouve une identité souvent utilisée dans la suite.

Lemme 2.3 *Soient φ et ψ des substitutions. L'identité*

$$\psi(\varphi \triangle \psi) \quad = \quad \varphi\psi$$

est valide si $\mathcal{D}om(\varphi) \cap \mathcal{V}ar(\psi) = \emptyset$.

Preuve: Notons

$$\varphi \sqcup \psi \quad = \quad \varphi \cup [x \mapsto x\psi \mid x \in \mathcal{D}om(\psi) - \mathcal{D}om(\varphi)]$$

l'union des substitutions φ et ψ.

Nous avons

$$\varphi\psi \quad = \quad (\varphi \triangle \psi) \sqcup \psi$$

car

1. si $x \in \mathcal{D}om(\varphi)$ alors $x((\varphi \triangle \psi) \sqcup \psi) = x(\varphi \triangle \psi) = x\varphi\psi$;
2. si $x \in \mathcal{D}om(\psi) - \mathcal{D}om(\varphi)$ alors $x((\varphi \triangle \psi) \sqcup \psi) = x\psi = (x\varphi)\psi$;
3. si $x \notin \mathcal{D}om(\varphi) \cup \mathcal{D}om(\psi)$ alors $x\varphi\psi = x = x((\varphi \triangle \psi) \sqcup \psi)$.

Nous avons

$$(\varphi \triangle \psi) \sqcup \psi = \psi \sqcup (\varphi \triangle \psi)$$

car $\mathcal{D}om(\varphi) \cap \mathcal{D}om(\psi) = \emptyset$. Nous avons

$$\psi \sqcup (\varphi \triangle \psi) = \psi(\varphi \triangle \psi)$$

car $\mathcal{VR}an(\psi) \cap \mathcal{D}om(\varphi) = \emptyset$ et $\mathcal{D}om(\varphi \triangle \psi) \subseteq \mathcal{D}om(\varphi)$. □

La propriété suivante n'est que le raccourci d'une longue notation.

Définition 2.4 *Les substitutions φ et ψ sont* **cohérentes** *(noté par $\varphi - \psi$) si $\mathcal{D}om(\varphi) \cap \mathcal{V}ar(\psi) = \emptyset$ ou $\mathcal{V}ar(\varphi) \cap \mathcal{D}om(\psi) = \emptyset$.*

La cohérence est une relation symétrique, elle signifie que les deux substitutions φ et ψ n'interfèrent pas.

Evidement, les propositions présentées ne couvrent pas entièrement toutes les propriétés des opérations définies, mais sont nécessaires pour pouvoir prouver les propriétés des objets définis dans la partie suivante.

2.1.2 Opérateurs sur les substitutions

Sur la base du produit restreint, nous définissons trois opérateurs itératifs sur les substitutions, notamment l'*exposant*, l'opérateur W et l'opérateur T. L'application de ces opérateurs itératifs sur les substitutions de base engendre une suite de nouvelles substitutions. Les deux premiers opérateurs seront utilisés pour la définition de la fermeture de chaînes, le troisième sera utilisé dans la définition de système croisé.

L'*exposant* appliqué sur une substitution φ signifie le cumul de cette substitution de la façon habituelle: $\varphi^0 = [\,]$ et $\varphi^{n+1} = \varphi^n\varphi$.

Il existe une généralisation du Lemme 2.3 pour l'exposant.

Lemme 2.5 *Soient φ et ψ des substitutions. Si $\mathcal{D}om(\varphi) \cap \mathcal{V}ar(\psi) = \emptyset$ alors pour chaque n nous avons $\psi^n(\varphi \triangle \psi^n) = \varphi\psi^n$.*

Preuve: Par induction sur n à partir du Lemme 2.3 et $\mathcal{D}om(\varphi) \cap \mathcal{V}ar(\psi^n) \subseteq \mathcal{D}om(\varphi) \cap \mathcal{V}ar(\psi) = \emptyset$. □

L'opérateur W présente une généralisation structurelle de l'exposant.

Définition 2.6 *Soit φ et ψ des substitutions. L'opérateur itératif W sur φ et ψ est défini par induction:*

1. $W_0(\varphi, \psi) = [\,]$
2. $W_{n+1}(\varphi, \psi) = (\varphi \triangle W_n(\varphi, \psi)) \triangle \psi$

Pour illustrer cet opérateur, voici les deux premières valeurs de la suite des W:

$$\begin{aligned} W_1(\varphi, \psi) &= (\varphi \triangle [\,]) \triangle \psi = \varphi \triangle \psi \\ W_2(\varphi, \psi) &= (\varphi \triangle (\varphi \triangle \psi)) \triangle \psi \end{aligned}$$

Par suite de la relation entre l'exposant et l'opérateur W, il serait intéressant de savoir quand les deux coïncident.

Lemme 2.7 *Soient φ et ψ des substitutions. Si $\mathcal{D}om(\psi) \cap \mathcal{VR}an(\varphi) = \emptyset$ alors pour chaque n, nous avons $W_n(\varphi, \psi) = \varphi^n$.*

Preuve: Par induction sur n. □

Nous définissons notre troisième opérateur, noté T. Il ne sera pas utilisé dans les chaînes de fermeture mais dans les définitions des systèmes croisés.

Définition 2.8 *Soit φ, ψ et σ des substitutions. L'opérateur itératif T est défini par induction:*

1. $T_0(\sigma, \psi, \varphi) = \sigma \triangle \varphi$
2. $T_{n+1}(\sigma, \psi, \varphi) = (\psi \triangle T_n(\sigma, \psi, \varphi)) \triangle \varphi$

L'opérateur T généralise l'opérateur W. Voici leur coïncidence.

Proposition 2.9 *L'identité $T_n(\psi, \psi, \varphi) = W_{n+1}(\psi, \varphi)$ est valide pour chaque n.*

Preuve: Par induction sur n. □

Le lemme suivant détermine la connexion entre les opérateurs T et exposant sous une condition particulière, retrouvée dans les systèmes croisés.

Lemme 2.10 *Soient φ, ψ et σ des substitutions. Si $\mathcal{D}om(\varphi) \cap (\mathcal{VR}an(\psi) \cup \mathcal{VR}an(\sigma)) = \emptyset$ et $\mathcal{D}om(\sigma) \subseteq \mathcal{D}om(\psi)$, alors $T_n(\sigma, \psi, \varphi) = \psi^n \cdot \sigma$ est valide pour chaque n.*

Preuve: Par induction sur n. □

2.2 Fermetures de règles

Nous avons besoin de manipuler des séquences de règles, qui interviennent dans une suite de superpositions dans des positions dépendantes, comme un seul objet. Ceci est fait par les *fermetures de règles* développées par Lankford et Musser [LM78] ainsi que par Guttag, Kapur et Musser [GKM83].

Définition 2.11 *Soit R un système de réécriture. L'ensemble des* **fermetures en avant** *$FC(R)$ de R est inductivement défini par:*

1. *Chaque règle $s \to t$ de R est une fermeture en avant $s \rhd\!\!\rightarrow t$.*

2. *Soient $s_1 \rhd\!\!\rightarrow t_1$ et $s_2 \rhd\!\!\rightarrow t_2$ deux fermetures en avant sans variable commune. Si $t_1|_a\sigma = s_2\sigma$ est valide pour l'unificateur principal σ et une position $a \in \mathcal{FP}os(t_1)$, alors $s_1\sigma \rhd\!\!\rightarrow t_1\sigma[t_2\sigma]_a$ est aussi une fermeture en avant.*

L'ensemble des **fermetures en arrière** *$BC(R)$ de R est inductivement définie par:*

1. *Chaque règle $s \to t$ de R est une fermeture en arrière $s \rightarrow\!\!\lhd t$.*

2. *Soient $s_1 \rightarrow\!\!\lhd t_1$ et $s_2 \rightarrow\!\!\lhd t_2$ deux fermetures en arrière sans variable commune. Si $t_1\sigma = s_2|_a\sigma$ est valide pour l'unificateur principal σ et une position $a \in \mathcal{FP}os(s_2)$, alors $s_2\sigma[s_1\sigma]_a \rightarrow\!\!\lhd t_2\sigma$ est aussi une fermeture en arrière.*

L'ensemble des **fermetures de superposition** *$OC(R)$ de R est inductivement défini par:*

1. *Chaque règle $s \to t$ de R est une fermeture de superposition $s \mapsto t$.*

2. *Soient $s_1 \mapsto t_1$ et $s_2 \mapsto t_2$ deux fermetures de superposition sans variable commune. Si $t_1|_a\sigma = s_2\sigma$ est valide pour l'unificateur principal σ et une position $a \in \mathcal{FP}os(t_1)$, alors $s_1\sigma \mapsto t_1\sigma[t_2\sigma]_a$ est aussi une fermeture de superposition.*

3. *Soient $s_1 \mapsto t_1$ et $s_2 \mapsto t_2$ deux fermetures de superposition sans variable commune. Si $t_1\sigma = s_2|_a\sigma$ est valide pour l'unificateur principal σ et une position $a \in \mathcal{FP}os(s_2)$, alors $s_2\sigma[s_1\sigma]_a \mapsto t_2\sigma$ est aussi une fermeture de superposition.*

Il faut bien noter que $OC(R) \subseteq \xrightarrow{+}_R$, ce qui indique que les fermetures de superposition sont simplement des réductions potentielles cumulées. De plus, notons $rd_R(s \mapsto t)$ l'ensemble de règles de R produisant la fermeture $s \mapsto t$ par la méthode de la Définition 2.11.

Une fermeture de règles peut être interprétée soit comme une nouvelle règle soit comme une suite de règles enchaînées par superposition. Soit $s_1 \to t_1$ une règle et $l_1 \to r_1$, ..., $l_n \to r_n$ une suite de règles tel que:

1. les règles $s_i \to t_i$ et $l_i \to r_i$ se superposent à la position $a_i \in \mathcal{FP}os(s_i)$, produisant ainsi la nouvelle règle $s_{i+1} \to t_{i+1}$,

2. chaque position a_i est le préfixe de la position a_{i+1}.

Le processus précédent peut être effectué d'une façon différente. D'abord, la fermeture en avant $l \rightarrowtriangle r$ est construite à partir de la suite $l_1 \to r_1, \ldots, l_n \to r_n$. Cette fermeture $l \rightarrowtriangle r$, interprétée pour cette occasion comme une règle, est superposée avec $s_1 \to t_1$, produisant $s_{n+1} \to t_{n+1}$. Cependant toutes les règles intermédiaires $s_2 \to t_2, \ldots, s_n \to t_n$ ne sont pas produites pendant ce nouveau processus, ce qui n'est pas important si nous nous intéressons uniquement à la règle produite à la fin de ce processus de superposition.

Si nous parlons plus loin d'une fermeture de règles, nous allons toujours penser à cette double interprétation. S'il faut parler explicitement de règles qui forment une fermeture, nous utilisons la notation avec rd_R.

Les notions de fermeture sont aussi liées au processus de *surréduction* [Fay79, Hul80, Lan75, Sla74], au processus de superposition généralisée [GKM83] et à la paramodulation. Les fermetures en avant et de superposition engendrent une méthode spécifique de preuve de terminaison des systèmes de réécriture dans [Der87]. En ce qui concerne le lien entre l'utilisation des fermetures de règles pour des preuves de terminaison et leur utilisation pour la divergence, il y a une différence importante. Pour prouver la terminaison d'un système par la méthode des fermetures, il faut prouver qu'il n'y a pas de fermetures infinies. Par contre, nous allons poser la question si l'ensemble de toutes les fermetures, à partir d'un ensemble de règles, est fini ou infini. Cependant chaque fermeture appartenant à cet ensemble est finie. Néanmois, il faut préciser que $s \rightarrowtriangle s$ est une fermeture finie.

Soit $\sqsubset_R$ une relation entre deux fermetures de superposition de $OC(R)$. $p \sqsubset_R q$ signifie que p est un *facteur* de q par rapport aux règles R. La relation $\sqsubset_R$ est définie comme la plus petite relation reflexive et transitive telle que si $s_1 \rightarrowtail t_1$ et $s_2 \rightarrowtail t_2$ sont les deux fermetures qui créent la fermeture $s \rightarrowtail t \in OC(R)$ alors

$$\begin{array}{lll} s_1 \rightarrowtail t_1 & \sqsubset_R & s \rightarrowtail t \quad \text{et} \\ s_2 \rightarrowtail t_2 & \sqsubset_R & s \rightarrowtail t \end{array}$$

Dans la suite, nous allons nous intéresser d'abord à la structure et à la construction des fermetures de règles en général. Ensuite, nous allons introduire des fermetures de règles particulières, intitulées les *chaînes de fermeture*, et présenter leurs propriétés.

2.2.1 Structure et construction des fermetures de règles

Comme c'est visible à partir de la définition 2.11, un principe de dualité détermine la construction des fermetures en avant et en arrière.

Proposition 2.12 (Principe de dualité) *Soit R un système de réécriture préservant les variables. La construction $s \rightarrowtriangle t$ est une fermeture en avant dans $FC(R)$ si et seulement si $t \rightarrowtriangle s$ est une fermeture en arrière dans $BC(R^{op})$.*

Selon ce principe de dualité, nous n'avons pas besoin de distinguer entre les fermetures en avant et en arrière dans le cas général mais simplement dans les cas particuliers. Si un résultat est prouvable pour un type de fermetures de règles alors il est prouvable aussi pour

l'autre type, selon le principe de dualité. Si on parle de fermetures de règles, on pense soit aux fermetures en avant soit aux fermetures en arrière, sans le préciser.

Définition 2.13 *Une* fermeture de règles *sans distinction sera notée par* $s \rhd\!\!\rightarrow t$, *ce qui indique soit* $s \rhd\!\!\rightarrow t$ *soit* $s \blacktriangleright\!\!\rightarrow t$. *L'*ensemble des fermetures *de* R *sans distinction sera noté* $RC(R)$, *ce qui indique soit* $FC(R)$ *soit* $BC(R^{op})$.

Regardons la construction des fermetures de règles de plus près. D'abord, nous introduisons l'opération de la construction des fermetures de règles, suivant la Définition 2.11.

Définition 2.14 *Soient* $p_1 = s_1 \rhd\!\!\rightarrow t_1$, $p_2 = s_2 \rhd\!\!\rightarrow t_2$ *deux fermetures de règles ne partageant pas de variables, telles que* σ *soit l'unificateur principal de* $t_1|_a$ *et* s_2 *pour* $a \in \mathcal{FP}os(t_1)$. *L'opération qui associe la fermeture de règles* $s_1\sigma \rhd\!\!\rightarrow t_1\sigma[t_2\sigma]_a$ *aux fermetures* p_1 *et* p_2 *est notée* $p_1 \rightsquigarrow_a p_2$.

On pourra parfois omettre a dans $\rightsquigarrow_a$ s'il n'y a pas d'ambiguïté.

La Définition 2.11 ne fournit aucune information sur l'associativité de la construction des fermeture. Cependant, l'associativité de cette construction est indispensable pour une raison d'efficacité.

Lemme 2.15 *Soient* p_1, p_2 *et* p_3 *des fermetures de règles de même type*[1]. $p_1 \rightsquigarrow_a (p_2 \rightsquigarrow_b p_3) = (p_1 \rightsquigarrow_a p_2) \rightsquigarrow_{ab} p_3$ *(à renommage près).*

Preuve: Soient $p_1 = s_1 \rhd\!\!\rightarrow t_1$, $p_2 = s_2 \rhd\!\!\rightarrow t_2$ et $p_3 = s_3 \rhd\!\!\rightarrow t_3$ les fermetures de règles de départ. Soit φ la substitution produisant $p_2 \rightsquigarrow p_3$, c'est-à-dire $t_2|_b\varphi = s_3\varphi$. Alors

$$p_2 \rightsquigarrow_b p_3 \quad = \quad s_2 \rhd\!\!\rightarrow t_2\varphi[t_3\varphi]_b$$

est la fermeture de règles produite. Soit ψ la substitution produisante $p_1 \rightsquigarrow_a (p_2 \rightsquigarrow_b p_3)$, c'est-à-dire

$$t_1|_a\psi \quad = \quad s_2\varphi\psi$$

Alors

$$p_1 \rightsquigarrow_a (p_2 \rightsquigarrow_b p_3) \quad = \quad s_1\psi \rhd\!\!\rightarrow t_1\psi[t_2\varphi\psi]_a[t_3\varphi\psi]_{ab} \tag{2.1}$$

est la fermeture construite, appelons-la q.

Les termes $t_1|_a$ et s_2 sont unifiables, ce qui indique que la fermeture de règles $q' = p_1 \rightsquigarrow p_2$ est constructible. Soit σ l'unificateur principal de $t_1|_a$ et s_2 intervenant dans la construction du q'. Selon la définition de l'unificateur principal et comme conséquence de l'identité (2.1), il existe une substitution ρ telle que $\psi \cup (\varphi\psi) = \sigma\rho$. La comparaison de ces substitutions avec (2.1) implique que la fermeture de règles $(p_1 \rightsquigarrow_a p_2) \rightsquigarrow_{ab} p_3$ est un renommage de la fermeture q. □

1. Toutes sont soit des fermetures en avant, soit des fermetures en arrière.

Procédure de production des ensembles de fermetures
Entrée: R, ou bien R^{op}
Sortie: $RC(R)$ en cas d'arrêt
Méthode:

variables A, B, C: *ensemble de couples de termes*;
begin
$B := R$; $A := \emptyset$; $C := \emptyset$;
repeat
for (**all** $p \in B$) **and** (**all** $q \in R$) **and** (**all** $a \in \mathcal{FP}os(p)$) **do**
if $p \rightsquigarrow_a q$ est constructible pour la position a
then $C := C \cup \{p \rightsquigarrow_a q\}$
fi
od;
$A := A \cup B$; $B := C$; $C := \emptyset$
until $B = \emptyset$;
$RC(R) := A$
end

FIG. 2.1 – *Production des ensembles de fermetures*

Le lemme précédent explique la demi-associativité des fermetures de règles. Les fermetures en avant sont associatives à gauche, les fermetures en arrière sont associatives à droite. Ceci rend inutiles les parenthèses autour des constructions de fermetures.

Corollaire 2.16 *Soit R un système de réécriture préservant les variables. Si p est une fermeture de règles appartenant à $RC(R) - R$, ou bien à $RC(R) - R^{op}$ respectivement, alors p est décomposable en $p = q_1 \rightsquigarrow q_2$, où $q_1 \in RC(R)$ et q_2 est une règle de réécriture dans R, ou bien dans R^{op} respectivement.*

Preuve: Application de l'associativité. □

A partir de ce corollaire on peut déduire qu'une nouvelle fermeture de règles peut être produite à partir d'une fermeture déjà existante et d'une règle de réécriture de base. Ce principe est utilisé dans la Procédure de production des ensembles de fermetures (voir la figure 2.1). Cette procédure est la même pour les ensembles de fermetures en avant et en arrière, à l'opération produisant les fermetures près. Pour cette raison, une seule procédure parametrée est présentée au lieu de deux. Le paramètre est l'opération de fabrication des fermetures.

La question, si l'ensemble des fermetures $RC(R)$ est fini, est indécidable pour un système de réécriture R arbitraire. On peut prouver cette proposition par une modification de la preuve d'indécidabilité de la terminaison.

Théorème 2.17 *Il est indécidable si l'ensemble de fermetures $RC(R)$ d'un système de réécriture R est fini, même dans le cas où R ne contient que deux règles de réécriture dont l'une est close.*

Preuve: Comme prouvé par Dauchet [Dau89], une machine de Turing peut être simulée par un système de réécriture d'une seule règle linéaire à gauche et préservant les variables. Soit R un tel système. Il est donc indécidable si la réduction d'un terme t par R se termine, même si R ne contient qu'une seule règle linéaire à gauche et préservant les variables.

Considérons la fermeture en avant $s \rhd\!\!\rightarrow t$ comme un nouveau terme dans une signature élargie. Il est indécidable si la réduction du "terme" $s \rhd\!\!\rightarrow t$, dans le sous-terme t, par le système R est bien fondée. Or la réduction consécutive du terme t dans la fermeture $s \rhd\!\!\rightarrow t$ par les règles du système R n'est rien d'autre que la construction consécutive de nouvelles fermetures à partir de la fermeture $s \rhd\!\!\rightarrow t$ et des règles de R, car le filtrage effectué au cours de la réduction est un cas particulier de l'unification.

Ce raisonnement peut être étendu aux fermetures en arrière utilisant le Principe de Dualité 2.12. Le problème de décidabilité de la finitude de l'ensemble des fermetures se réduit donc au problème de terminaison des systèmes de réécriture. □

Même si seuls les systèmes monadiques et orthogonaux sont pris en compte, cette propriété d'indécidabilité reste valide.

Corollaire 2.18 *Il est indécidable si l'ensemble des fermetures $RC(R)$ d'un système de réécriture R est fini, même si R est monadique et sans superpositions.*

Preuve: Huet et Lankford [HL78] ont prouvé l'indécidabilité de la terminaison sous la restriction aux symboles monadiques et constantes dans un système de réécriture sans superpositions. De plus, le système construit dans leur preuve préserve les variables. L'application à leur proposition des mêmes arguments que dans la preuve du théorème précédent entraîne le résultat désiré. □

2.2.2 Chaînes de fermeture

On définit des fermetures de règles spéciales, nommées les *chaînes de fermeture*. Comme dans le cas des fermetures de règles, deux types de chaînes de fermeture apparaissent: les chaînes en avant et en arrière. Une grande similarité entre leurs constructions est observable. Ensuite, nous introduisons les théorèmes de caractérisation pour les ensembles de fermetures quand ils contiennent une chaîne, montrant que cette propriété est suffisante pour l'infinité de $RC(R)$. Actuellement, notre intérêt est tourné vers les chaînes de fermeture parce que, d'après leur construction, elles sont responsables de la divergence de la procédure de complétion de Knuth et Bendix.

La Section 2.2.1 décrivait la structure et la construction des fermetures de règles $RC(R)$. Maintenant, nous introduisons une classe de fermetures de règles spéciales mais suffisament générale, pour que l'existence de ces règles soient essentielle pour la production d'un ensemble infini des fermetures. Les preuves des propriétés de fermeture de règles sont rendues possibles grâce à l'application de l'algèbre de substitutions développée dans la Section 2.1. Le principe de l'application des substitutions avec variables propres devient plus clair après l'explication suivante.

Considérons une fermeture de règles

$$s \quad \rhd\!\!\rhd \quad t \tag{2.2}$$

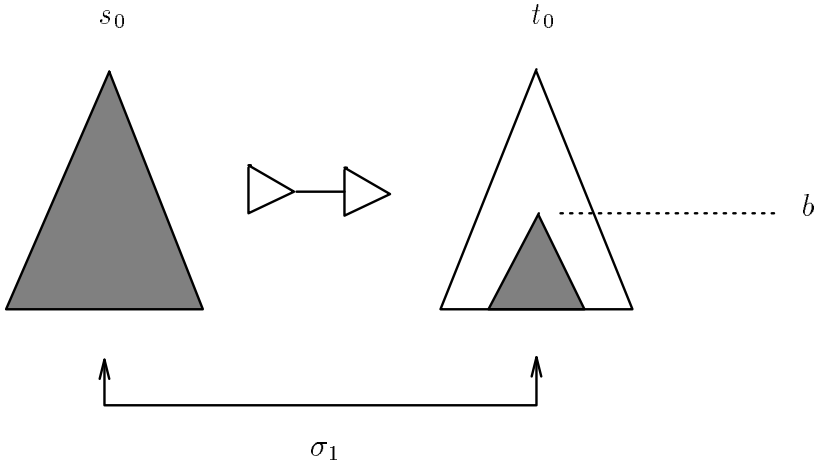


FIG. 2.2 – *Chaîne de fermeture: Etape initiale*

telle qu'il soit possible de l'enchaîner avec elle-même. Ceci nécessite que s soit unifiable avec un sous-terme $t|_b$. En général, cette unification est possible si les termes possèdent des variables disjointes, ce qui n'est pas le cas ici. C'est pouquoi nous sommes obligés de renommer les variables. Si on veut observer l'itération de l'enchaînement, on doit effectuer ce renommage explicitement. Ces réflexions nous amènent à diviser l'unificateur en une partie renommages des variables et une partie substitutions de variables propres.

Nous avons d'abord besoin du renommage des variables $\sigma_0 = [x \mapsto x_0 \mid x \in \mathcal{V}ar(s)]$ pour indicer les variables de la règle (2.2). Considérons maintenant l'enchaînement des fermetures de règles $s\sigma_0 = s_0 \rhd\!\rhd t_0 = t\sigma_0$ et $s \rhd\!\rhd t$. L'unificateur principal σ_1 de cet enchaînement peut être choisi idempotent, tel que

$$t_0|_b\sigma_1 \quad = \quad s\sigma_1 \tag{2.3}$$

(voir figure 2.2). La fermeture de règles obtenue par le premier pas d'itération est donc

$$s_0\sigma_1 \quad \rhd\!\rhd \quad t_0\sigma_1[t\sigma_1]_b \tag{2.4}$$

Supposons de plus que la fermeture de la règle (2.4) et la fermeture de la règle d'origine (2.2) produisent une nouvelle fermeture de règles, et que ce processus puisse continuer indéfiniment: autrement dit, la fermeture de règles, produite pendant la n-ième itération, et la fermeture de règles (2.2) produisent toujours une nouvelle fermeture de règles.

Posons $s_1 = s_0\sigma_1$ et $t_1 = t_0\sigma_1[t\sigma_1]_b$. Pendant la deuxième itération on doit utiliser l'unificateur σ_2 et la position $b' \in \mathcal{FP}os(t_1)$, tels que

$$t_1|_{b'}\sigma_2 \quad = \quad s\sigma_2 \tag{2.5}$$

soit valide. Bien sûr, c'est un simple rappel de la condition (2.3) de la première itération. Il est naturel de considérer b' comme une itération de b, c-à-d $b' = b.b$, parce que cette position existe dans t_1, même si, en général, $t|_b$ n'est pas le seul sous-terme de t qui pourrait s'unifier avec s. Ceci implique qu'on peut transformer l'identité (2.5) en

$$t|_b\sigma_1\sigma_2 \quad = \quad s\sigma_2 \tag{2.6}$$

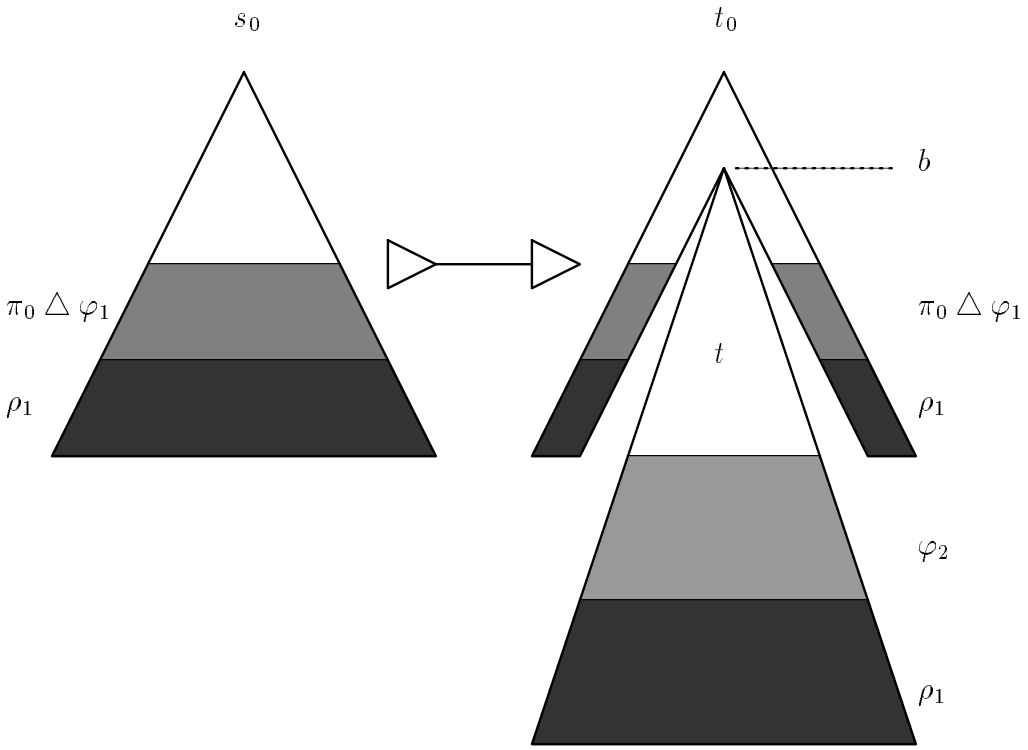


FIG. 2.3 – *Chaîne de fermeture: 1ère étape*

qui engendre la fermeture

$$s_0\sigma_1\sigma_2 \quad \rhd\!\!\rhd \quad t_0\sigma_1\sigma_2[t\sigma_1\sigma_2[t\sigma_1]_b]_b$$

Le problème est donc la décidabilité de l'existence d'une suite infinie de substitutions idempotentes σ_0 (renommage), σ_1, σ_2, ..., σ_n, ... telle que σ_{i+1} soit l'unificateur principal de $t|_b\sigma_i$ et s:

$$\begin{aligned} t|_b\sigma_0\sigma_1 &= s\sigma_1 \\ t|_b\sigma_1\sigma_2 &= s\sigma_2 \\ &\vdots \\ t|_b\sigma_n\sigma_{n+1} &= s\sigma_{n+1} \end{aligned}$$

Ce qui suit ne répond pas exactement à la question qui reste donc ouverte, mais propose une solution sous certaines hypothèses concernant l'expression de σ_{i+1} en fonction des substitutions précédentes σ_0, ..., σ_i, qui aura l'avantage de donner une expression manipulable de ces substitutions. Pour cela, nous allons tout d'abord "découper" chaque substitution σ_i en deux parties. Considérons pour cela l'unificateur σ_1. Il agit à la fois sur le sous-terme $t|_b$ et sur le terme s. On va l'écrire sous la forme:

$$\sigma_1 \quad = \quad ((\pi_0 \triangle \varphi_1) \cup \varphi_2)\rho_1$$

où le renommage π_0 élimine l'effet du renommage précédent, φ_1 agit sur le sous-terme $t|_b$ et φ_2 agit sur le terme s. Les deux substitutions φ_1 et φ_2 sont suivies par renommage ρ_1 pour rendre l'unificateur σ_1 idempotent.

Le même processus de décomposition est appliqué à l'unificateur σ_2 en déduisant les renommages π_1, ρ_2 et les substitutions φ_1', φ_2', correspondant respectivement aux π_0, ρ_1, φ_1 et φ_2.

Les substitutions φ_1' et φ_2' ne sont pas complètement nouvelles, mais il est plus naturel de les considérer comme des fonctions de substitutions d'origine φ_1 et φ_2. Pour cette raison nous introduisons la notion d'une fonction particulière sur les substitutions, dans ce cas

$$\varphi_1' = f_1(\varphi_1, \varphi_2) \tag{2.7}$$
$$\varphi_2' = f_2(\varphi_1, \varphi_2) \tag{2.8}$$

Comparons d'abord les identités (2.3) et (2.6). Il est évident qu'on peut étendre l'identité (2.3) à

$$t|_b\varphi_1\psi' = s\varphi_2\psi'$$

pour chaque substitution ψ' suivant les propriétés de composition, mais cette extension ne donne pas le résultat souhaité et, par conséquent, elle ne correspond pas à nos objectifs. On est obligé de procéder d'une manière plus astucieuse: il faut étendre l'identité (2.3) à

$$t|_b(\varphi_1 \triangle \psi') = s(\varphi_2 \triangle \psi') \tag{2.9}$$

Il est également évident de considérer les nouvelles substitutions de l'extension ψ' en termes de φ_1 et φ_2, introduisant ainsi la troisième fonction sur les substitutions

$$\psi' = f_3(\varphi_1, \varphi_2) \tag{2.10}$$

En comparant les identités (2.6) et (2.9) en termes des fonctions définies sur les substitutions, on obtient les identités suivantes

$$\varphi_2 f_1(\varphi_1, \varphi_2) = \varphi_1 \triangle f_3(\varphi_1, \varphi_2) \tag{2.11}$$
$$f_2(\varphi_1, \varphi_2) = \varphi_2 \triangle f_3(\varphi_1, \varphi_2) \tag{2.12}$$

qui expriment les conditions qui doivent être satisfaites pendant la deuxième itération.

Comme décrit précédemment, on demande qu'il soit possible d'itérer ce processus indéfiniment. Ceci signifie que la suite infinie itérée des fermetures de règles $s_n \rhd\!\!\to t_n$, où $s_n \rhd\!\!\to t_n = (s_{n-1} \rhd\!\!\to t_{n-1}) \rightsquigarrow_{b^n} (s \rhd\!\!\to t)$, doit être constructible à partir de $s \rhd\!\!\to t$ par des suites itérées des substitutions [2]

$$\varphi_1^{(n)} = f_1^{(n)}(\varphi_1, \varphi_2)$$
$$\varphi_2^{(n)} = f_2^{(n)}(\varphi_1, \varphi_2)$$
$$\psi^{(n)} = f_3^{(n)}(\varphi_1, \varphi_2)$$

2. On considère que $\varphi_i^{(1)} = \varphi_i'$ et $f_i^{(1)} = f_i$.

profitant du renommage des variables explicite par la paire des substitutions (π_n, ρ_n) que nous appelons pliage et dépliage

$$\begin{aligned} \pi_n &= [x_n \mapsto x \mid x \in \mathcal{V}ar(t_n|_{b^{n+1}})] \\ \rho_n &= [x \mapsto x_n \mid x \in \mathcal{V}ar(s)] \end{aligned}$$

pour chaque pas d'itération n. L'unificateur de la n-ième itération est construit à partir de $\varphi_1^{(n)}$, $\varphi_2^{(n)}$, π_n et ρ_n:

$$\sigma_n = ((\pi_n \triangle \varphi_1^{(n)}) \cup \varphi_2^{(n)})\rho_{n+1}$$

Les suites itérées des substitutions $\varphi_1^{(n)}$, $\varphi_2^{(n)}$ et $\psi^{(n)}$ sont produites à partir de la notion de fonctions sur les substitutions introduites par les identités (2.7), (2.8) et (2.10). Pour cette raison, les nouvelles fonctions sur les substitutions $f_1^{(n)}$, $f_2^{(n)}$ et $f_3^{(n)}$ sont produites à partir des itérations des fonctions de base f_1, f_2 et f_3. Les conditions (2.11) et (2.12) sont à modifier dans le même sens.

Le théorème suivant prouve que les itérations évoquées dans les paragraphes précédents peuvent être mises en œuvre en utilisant des constructions de l'algèbre de substitutions, en particulier les opérateurs itératifs sur les substitutions, développés dans la Section 2.1. Remarquons enfin qu'on peut élargir la fermeture (2.2) à une structure mixte, dite fermeture de superposition.

Définition 2.19 *Une fermeture de superposition $s \mapsto\!\!\!\!\blacktriangleright\, t$ est une* **chaîne en avant** *s'il existe des substitutions φ_1, φ_2 telles que $\mathcal{D}om(\varphi_i) \subseteq \mathcal{V}ar(s)$ pour $i = 1, 2$ et une position $b \in \mathcal{FP}os(t)$ de t, telles que*

1. *$\langle \varphi_1, \varphi_2 \rangle$ constitue l'unificateur principal faible de $t|_b$ et s: $t|_b\varphi_1 = s\varphi_2$*
2. *$\mathcal{D}om(\varphi_1) \cap \mathcal{V}ar(\varphi_2) = \emptyset$ ou $\mathcal{D}om(\varphi_2) \cap \mathcal{V}ar(\varphi_1) = \emptyset$ (relation de cohérence)*

La présentation schématique d'une chaîne en avant se trouve dans la figure 2.2.

Exemple 2.20 Il existe deux cas particuliers de chaînes en avant dépendant du choix de l'identité dans la relation de cohérence.

1. Considérons la fermeture en avant $(x \oplus y) \otimes y \mapsto\!\!\!\!\blacktriangleright\, k(x \otimes k(y))$. Il existe des substitutions $\varphi_1 = [x \mapsto x \oplus k(y)]$, $\varphi_2 = [y \mapsto k(y)]$ et la position $b = 1$ satisfaisant les conditions $t|_b\varphi_1 = s\varphi_2$ et $\mathcal{D}om(\varphi_1) \cap \mathcal{V}ar(\varphi_2) = \emptyset$. Cette fermeture en avant appartient alors au premier type des chaînes en avant.

2. Considérons la fermeture en avant $g(x) \oplus y \mapsto\!\!\!\!\blacktriangleright\, g(x \oplus (x \oslash y))$. Il existe des substitutions $\varphi_1 = [x \mapsto g(x)]$, $\varphi_2 = [y \mapsto g(x) \oslash y]$ et la position $b = 1$ satisfaisant les conditions $t|_b\varphi_1 = s\varphi_2$ et $\mathcal{D}om(\varphi_2) \cap \mathcal{V}ar(\varphi_1) = \emptyset$. Cette fermeture en avant appartient alors au second type de chaînes en avant.

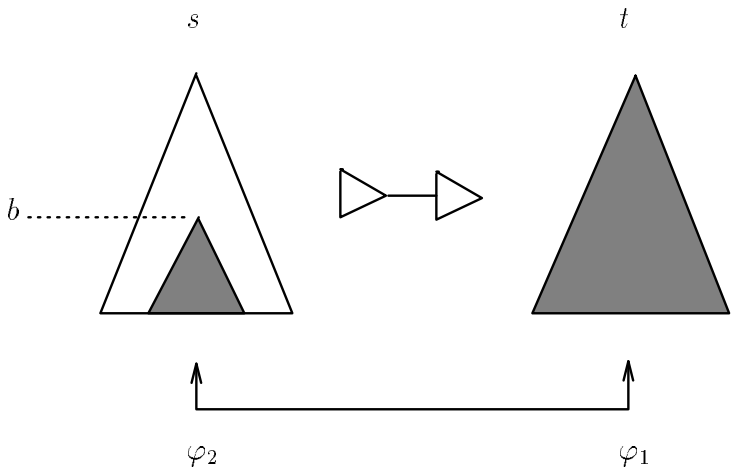


FIG. 2.4 – *Chaîne en arrière*

On pourrait argumenter qu'il existe des fermetures de superposition qui ne satisfont pas la condition de cohérence et qui pourtant produisent une suite infinie itérée, comme par exemple la fermeture en avant

$$f(x \oplus y, y, u, x \otimes z) \quad \rightarrowtail \quad f(g(g(y)) \oplus x, g(y), h(u), z)$$

avec $\varphi_1 = [x \mapsto g(y), z \mapsto x * z]$, $\varphi_2 = [x \mapsto g^2(y), y \mapsto g(y), u \mapsto h(u)]$. Après la première itération on obtient la fermeture

$$f(g(y) + y, y, u, x * (x * z)) \quad \rightarrowtail \quad f(g^3(y) + g^2(y), g^2(y), h^2(u), z)$$

qui satisfait la relation de cohérence avec $\varphi_1' = [z \mapsto x * (x * z)]$, $\varphi_2' = [y \mapsto g^2(y), u \mapsto h^2(u)]$ et, par conséquent, tout rentre dans l'ordre.

Les chaînes en arrière représentent l'image miroir des chaînes en avant. Elles sont construites à partir des fermetures en arrière de la même manière. Une certaine dualité existe aussi entre les chaînes en avant et en arrière.

Définition 2.21 *Une fermeture de superposition $s \rightarrowtail t$ est une* **chaîne en arrière** *s'il existe des substitutions φ_1, φ_2 telles que $\mathcal{D}om(\varphi_i) \subseteq \mathcal{V}ar(s)$ pour $i = 1, 2$ et une position $b \in \mathcal{FP}os(s)$, telles que*

1. *$\langle \varphi_1, \varphi_2 \rangle$ forme l'unificateur principal faible de t et $s|_b$: $t\varphi_1 = s|_b\varphi_2$*
2. *$\mathcal{D}om(\varphi_1) \cap \mathcal{V}ar(\varphi_2) = \emptyset$ ou $\mathcal{D}om(\varphi_2) \cap \mathcal{V}ar(\varphi_1) = \emptyset$ (relation de cohérence)*

La présentation schématique d'une chaîne en arrière se trouve dans la figure 2.4.

Exemple 2.22 Il existe, comme dans le cas des chaînes en avant, deux types des chaînes en arrière dépendant du choix de la condition dans la relation de cohérence.

1. Considérons la fermeture en arrière $d(x \oplus (x \otimes y)) \rightarrowtail d(x) \oplus y$. Il existe des substitutions $\varphi_1 = [y \mapsto d(x) \otimes y]$, $\varphi_2 = [x \mapsto d(x)]$ et la position $b = 1$ telles que les conditions $t\varphi_1 = s|_b\varphi_2$ et $\mathcal{D}om(\varphi_1) \cap \mathcal{V}ar(\varphi_2) = \emptyset$ soient satisfaites. Cette fermeture en arrière appartient alors au premier type de chaînes en arrière.

2. Considérons la fermeture en arrière $(x \otimes h(y)) \oplus y \blacktriangleright\!\triangleright (x \oplus y) \otimes y$. Il existe des substitutions $\varphi_1 = [y \mapsto h(y)]$, $\varphi_2 = [x \mapsto x \oplus h(y)]$ et la position $b = 1$ telles que les conditions $t\varphi_1 = s|_b\varphi_2$ et $\mathcal{D}om(\varphi_2) \cap \mathcal{V}ar(\varphi_1) = \emptyset$ soient satisfaites. Cette fermeture en arrière appartient alors au second type des chaînes en arrière.

La même discussion sur la condition de cohérence, comme dans le cas des chaînes en avant, s'applique aussi aux chaînes en arrière.

Exemple 2.23 Bien sûr, chaque fermeture de règles ne produit pas une chaîne de fermeture. Considérons le système de réécriture

$$\begin{array}{rcl} x \otimes k(y) & \to & k(x \oplus k(y)) \\ (x \oslash y) \oplus y & \to & g(x) \otimes y \end{array}$$

Son ensemble de fermetures en avant contient ces règles de réécriture transformées en fermetures et, en plus, les fermetures en avant composées suivantes

$$\begin{array}{rcl} (x \oslash k(y)) \otimes k(y) & \rhd\!\!\blacktriangleright & k(g(x) \otimes k(y)) \\ (x \oslash k(y)) \oplus k(y) & \rhd\!\!\blacktriangleright & k(g(x) \oplus k(y)) \\ (x \oslash k(y)) \otimes k(y) & \rhd\!\!\blacktriangleright & k^2(g(x) \oplus k(y)) \end{array}$$

On ne trouve pas de chaîne de fermeture parmi elles et cet ensemble de fermetures est fini. Nous prouverons que la finitude de l'ensemble des fermetures nécessite l'absence d'une telle chaîne de fermeture.

Le théorème suivant caractérise les ensembles de fermetures par les chaînes de fermeture. En fait, il s'agit de deux théorèmes, l'un concernant les chaînes en avant, l'autre les chaînes en arrière. La preuve pour les chaînes de fermeture correspond syntaxiquement à celle des chaînes en avant. Le résultat pour les chaînes en arrière peut être prouvé de la même manière. Cet argument s'applique sur tous les autres résultats dans cette sous-section.

Théorème 2.24 *Si $s \rhd\!\!\blacktriangleright t$ est une chaîne de fermeture, où $s \neq t$, alors $RC(\{s \to t\})$ est infini.*

Preuve: Soit $s \rhd\!\!\blacktriangleright t$ une chaîne de fermeture comme dans la Définition 2.19 ou 2.21. Soient

$$\begin{array}{rcl} s_0 & = & s\rho_0 \\ s_{n+1} & = & s_n(\pi_n \triangle (\varphi_1 \triangle W_n(\varphi_2, \varphi_1)))\rho_{n+1} \\ t_0 & = & t\rho_0 \\ t_{n+1} & = & t_n(\pi_n \triangle (\varphi_1 \triangle W_n(\varphi_2, \varphi_1)))\rho_{n+1}[t(\varphi_2 \triangle W_n(\varphi_2, \varphi_1))\rho_{n+1}]_{b^{n+1}} \end{array}$$

des suites des termes s_n et t_n. On va prouver que $s_n \rhd\!\!\blacktriangleright t_n$ appartient à l'ensemble de fermetures $RC(\{s \to t\})$ pour chaque n par induction.

Base: $n = 0$. La fermeture de règles $s_0 \rhd\!\!\blacktriangleright t_0$ est une instance de $s \rhd\!\!\blacktriangleright t$.

Pas: Par hypothèse, les fermetures de règles $s_k \rhd\!\!\rhd t_k$ et $s \rhd\!\!\rhd t$ appartiennent à $RC(\{s \to t\})$. A partir d'elles, on va construire la fermeture de règles $s_{k+1} \rhd\!\!\rhd t_{k+1}$. On est obligé d'analyser deux possibilités selon les définitions des chaînes de fermeture.

1. $\mathcal{D}om(\varphi_1) \cap \mathcal{V}ar(\varphi_2) = \emptyset$.

   Dans ce cas, on a $\varphi_1 \triangle W_n(\varphi_2, \varphi_1) = \varphi_1 \triangle \varphi_2^n$ et $\varphi_2 \triangle W_n(\varphi_2, \varphi_1) = \varphi_2^{n+1}$ d'après le Lemme 2.7. On obtient alors

   $$\begin{aligned} s_{n+1} &= s_n(\pi_n \triangle (\varphi_1 \triangle \varphi_2^n))\rho_{n+1} \\ t_{n+1} &= t_n(\pi_n \triangle (\varphi_1 \triangle \varphi_2^n))\rho_{n+1}[t\varphi_2^{n+1}\rho_{n+1}]_{b^{n+1}} \end{aligned}$$

   A partir de ces identités on obtient $t_k|_{b^{n+1}}\pi_k = t|_b\varphi_2^k$ et à partir du Lemme 2.5 on prouve $t|_b\varphi_2^k(\varphi_1 \triangle \varphi_2^k) = s\varphi_2^{k+1}$. De cette identité on tire l'identité $t_k|_{b^{k+1}}(\pi_k \triangle (\varphi_1 \triangle \varphi_2^k))\rho_{k+1} = s\varphi_2^{k+1}\rho_{k+1}$ qui satisfait la définition de la fermeture de règles avec l'unificateur $((\pi_k \triangle (\varphi_1 \triangle \varphi_2^k)) \cup \varphi_2^{k+1})\rho_{k+1}$. Alors, la construction

   $$s_k(\pi_k \triangle (\varphi_1 \triangle \varphi_2^k))\rho_{k+1} \quad \rhd\!\!\rhd \quad t_k(\pi_k \triangle (\varphi_1 \triangle \varphi_2^k))\rho_{k+1}[t\varphi_2^{k+1}\rho_{k+1}]_{b^{k+1}}$$

   est, elle aussi, une fermeture de règles dans $RC(\{s \to t\})$ et c'est exactement

   $$s_{k+1} \quad \rhd\!\!\rhd \quad t_{k+1}$$

2. $\mathcal{D}om(\varphi_2) \cap \mathcal{V}ar(\varphi_1) = \emptyset$.

   Dans ce cas on a $\varphi_1 \triangle W_n(\varphi_2, \varphi_1) = \varphi_1$ suivant le Lemme 2.2 et $\mathcal{D}om(W_n(\varphi_2, \varphi_1)) \subseteq \mathcal{D}om(\varphi_2)$. On obtient alors

   $$\begin{aligned} s_{n+1} &= s_n(\pi_n \triangle \varphi_1)\rho_{n+1} \\ t_{n+1} &= t_n(\pi_n \triangle \varphi_1)\rho_{n+1}[t(\varphi_2 \triangle W_n(\varphi_2, \varphi_1))\rho_{n+1}]_{b^{n+1}} \end{aligned}$$

   A partir de cela on obtient $t_k|_{b^{k+1}}\pi_k = t|_b(\varphi_2 \triangle W_{k-1}(\varphi_2, \varphi_1))$ pour $k \geq 1$, et utilisant le Lemme 2.3 on prouve que $t|_b(\varphi_2 \triangle W_{k-1}(\varphi_2, \varphi_1))\varphi_1 = s(\varphi_2 \triangle W_k(\varphi_2, \varphi_1))$. Cette identité implique que $t_k|_{b^{k+1}}(\pi_k \triangle \varphi_1)\rho_{k+1} = s(\varphi_2 \triangle W_k(\varphi_2, \varphi_1))\rho_{k+1}$, ce qui satisfait la définition de la fermeture de règles avec l'unificateur $((\pi_k \triangle \varphi_1) \cup (\varphi_2 \triangle W_k(\varphi_2, \varphi_1)))\rho_{k+1}$. Ceci indique que la construction

   $$s_k(\pi_k \triangle \varphi_1)\rho_{k+1} \quad \rhd\!\!\rhd \quad t_k(\pi_k \triangle \varphi_1)\rho_{k+1}[t(\varphi_2 \triangle W_k(\varphi_2, \varphi_1))\rho_{k+1}]_{b^{k+1}}$$

   est, elle aussi, une fermeture de règles dans $RC(\{s \to t\})$, et c'est exactement

   $$s_{k+1} \quad \rhd\!\!\rhd \quad t_{k+1}$$

□

Appelons toute fermeture de règles $s \rhd\!\!\rhd s$ (même terme à gauche et à droite) fermeture *réflexive*.

Corollaire 2.25 *Si $RC(R)$ contient une chaîne de fermeture non-réflexive, alors $RC(R)$ est infini.*

Le corollaire suivant prouve qu'on peut effectuer le test pour déterminer une chaîne de fermeture à n'importe quelle étape de la procédure produisant l'ensemble de fermetures.

Corollaire 2.26 *Chaque fermeture de règles $s_n \rhd\!\!\rightarrow t_n$ de la preuve du Théorème 2.24 est une chaîne de fermeture.*

Preuve: L'existence de la substitution pour la première condition des chaînes de fermeture est une conséquence de la preuve du Théorème 2.24, mélangée avec l'associativité à gauche de la construction des fermetures, présentée dans la proposition suivante:

> *Soient p_1, p_2 et p_3 des fermetures de règles du même type. Si la fermeture de règles $q = (p_1 \rightsquigarrow_a p_2) \rightsquigarrow_{ab} p_3$ est constructible et $b \in \mathcal{FP}os(t_2)$, où $p_2 = s_2 \rhd\!\!\rightarrow t_2$, alors $q = p_1 \rightsquigarrow_a (p_2 \rightsquigarrow_b p_3)$.*

La preuve de cette proposition se trouve dans l'annexe de l'article [GKM83].

Il reste à prouver la propriété de cohérence pour toute la suite de fermetures. On va prouver que la condition cherchée de cohérence est valide dans chaque pas d'itération à partir de la chaîne d'origine.

Pour cette preuve de la propriété de cohérence, on reprend les substitutions

$$\varphi_1 \triangle W_n(\varphi_2, \varphi_1) \qquad \text{et} \qquad \varphi_2 \triangle W_n(\varphi_2, \varphi_1)$$

au lieu de φ_1 et φ_2, respectivement. On est obligé, comme cela a été fait déjà dans la preuve du Théorème 2.24, d'analyser deux possibilités:

1. $\mathcal{D}om(\varphi_1) \cap \mathcal{V}ar(\varphi_2) = \emptyset$.

   A partir des définitions du produit restreint des substitutions et de l'opérateur d'exposant on déduit

   $$\begin{aligned} \mathcal{D}om(\varphi_1 \triangle \varphi_2^n) &\subseteq \mathcal{D}om(\varphi_1) \\ \mathcal{V}ar(\varphi_2^n) &\subseteq \mathcal{V}ar(\varphi_2) \end{aligned}$$

   Le remplacement des substitutions φ_1 et φ_2 par les nouvelles $\varphi_1 \triangle \varphi_2^n$ et φ_2^{n+1} respectivement dans la première partie de la condition de cohérence implique

   $$\mathcal{D}om(\varphi_1 \triangle \varphi_2^n) \cap \mathcal{V}ar(\varphi_2^{n+1}) \subseteq \mathcal{D}om(\varphi_1) \cap \mathcal{V}ar(\varphi_2) = \emptyset$$

2. $\mathcal{D}om(\varphi_2) \cap \mathcal{V}ar(\varphi_1) = \emptyset$.

   Le remplacement de la substitution φ_2 par la nouvelle $\varphi_2 \triangle W_n(\varphi_2, \varphi_1)$ dans la deuxième partie de la condition de cohérence implique

   $$\mathcal{D}om(\varphi_2 \triangle W_n(\varphi_2, \varphi_1)) \cap \mathcal{V}ar(\varphi_1) \subseteq \mathcal{D}om(\varphi_2) \cap \mathcal{V}ar(\varphi_1) = \emptyset$$

□

En utilisant le Corollaire 2.26, on peut classer des chaînes de fermeture selon leur construction initiale.

Définition 2.27 *Une chaîne de fermeture $c = s \rhd\!\!\rightarrow t$ de $RC(R)$ est* **principale** *dans $RC(R)$ s'il n'existe pas d'autre chaîne de fermeture $c' = s' \rhd\!\!\rightarrow t'$ dans $RC(R)$, telles que $c = s'_n \rhd\!\!\rightarrow t'_n$ pour un pas d'itération n sur la chaîne c'.*

Le dernier problème à résoudre dans le contexte des chaînes de fermeture est la question de décider si l'ensemble des fermetures $RC(R)$ contient une chaîne de fermeture.

Théorème 2.28 *Il est indécidable en général si $RC(R)$ contient une chaîne de fermeture donnée.*

Preuve: La preuve constitue une simple modification de la preuve d'indécidabilité de Narendran et Stillman [NS89].

Supposons qu'on a un algorithme pour décider si $RC(R)$ contient une chaîne de fermeture donnée, cet algorithme peut décider l'arrêt de n'importe quelle machine de Turing sur n'importe quelle configuration de départ. Soit M une machine de Turing et w une configuration de départ, soit R_M un système de réécriture qui code M et et s_w le codage associé à la configuration w et t le codage de la configuration d'acceptation, soit a et b deux constantes qui ne figurent pas dans le vocabulaire de R_M. La chaîne $a \mapsto\!\!\!\!\blacktriangleright\, b$ est une chaîne de fermeture du système $R_M \cup \{a \to s_w\} \cup \{t \to b\}$ si et seulement si $s_w \xrightarrow{*}_{R_M} t$, c.-à.-d. si et seulement si la machine M atteint la configuration w, c.-à.-d. si et seulement si M accepte w. Donc la décision d'appartenance d'une chaîne à $RC(R)$ entraîne celle de l'arrêt des machines de Turing. Contradiction. □

2.3 Systèmes de réécriture croisés

Les chaînes de fermeture, introduites dans la section précédente, sont les concepts de base qui permettent de reconnaître la divergence d'un système de réécriture R, mais leur existence dans R sans conditions supplémentaires ne suffit pas pour déclencher la divergence. Pour pouvoir construire un contre-exemple, il faut chercher un système de réécriture $R = \{s_1 \to t_1, s_2 \to t_2\}$, tel que $s_1|_a\sigma_1 = s_2\sigma_2$ et $t_2|_b\varphi_1 = s_2\varphi_2$, où $\varphi_1 - \varphi_2$, mais $\mathcal{D}om(\varphi_1) \cap \mathcal{V}ar(\sigma_2) \neq \emptyset$ et $\mathcal{V}ar(\varphi_1) \cap \mathcal{D}om(\sigma_2) \neq \emptyset$. Un tel système est, par exemple:

$$\begin{aligned} d(f(g(h(x))) \oplus y) &\to x \\ f(g(x)) \oplus k(y) &\to c(f(x) \oplus y) \end{aligned}$$

où la deuxième règle, orientée par $\succ_{lpo}$ utilisant la précédence $\oplus \succ c$, est la *chaîne en avant* avec $\varphi_1 = [x \mapsto g(x), y \mapsto k(y)]$ et $\varphi_2 = [\,]$. Alors $\varphi_1 - \varphi_2$ est valide. Malgré cela, ce système peut être complété dans le système canonique et fini suivant

$$\begin{aligned} d(f(g(h(x))) \oplus y) &\to x \\ f(g(x)) \oplus k(y) &\to c(f(x) \oplus y) \\ d(c(f(h(x)) \oplus y)) &\to x \end{aligned}$$

Ceci est dû au fait que $\sigma_2 = [x \mapsto h(x)]$, alors $\mathcal{D}om(\varphi_1) \cap \mathcal{V}ar(\sigma_2) = \{x\}$ et $\mathcal{V}ar(\varphi_1) \cap \mathcal{D}om(\sigma_2) = \{x\}$.

La suite de cette section explique ce qu'il faut ajouter aux chaînes de fermeture pour obtenir des schémas permettant la reconnaîssance de la divergence d'un système de réécriture. Ces schémas utilisent plutôt des chaînes de fermeture au lieu de simples règles pour pouvoir formaliser des systèmes divergents qui contiennent plus de deux règles.

2.3.1 Systèmes croisés en avant

Avant de présenter la définition, nous allons d'abord étudier le principe de ce schéma de divergence. Le schéma contient deux règles de réécriture, où

$$s_1 \rightarrow t_1 \tag{2.13}$$

est la *règle de départ* (*démarreur*) et

$$s_2 \rightarrow t_2 \tag{2.14}$$

est le *moteur* de la divergence. Supposons que ces règles se superposent avec (2.14) comme règle *mineure*, en utilisant l'unificateur ω_0, produisant une paire critique qui est orientée ensuite dans la règle

$$u_1 = s_1\omega_0[t_2\omega_0]_a \rightarrow t_1\omega_0 = v_1 \tag{2.15}$$

Supposons que les règles (2.14) et (2.15) se superposent à nouveau toujours avec (2.14) comme règle *mineure*, en utilisant l'unificateur ω_1, et produisant la nouvelle règle

$$u_2 = u_1\omega_1[t_2\omega_1]_{a'} \rightarrow v_1\omega_1 = v_2$$

Supposons en outre que ce processus puisse être itéré à l'infini. Ceci signifie qu'on suppose que $u_n \rightarrow v_n$ et $s_2 \rightarrow t_2$ se superposent, en utilisant l'unificateur ω_n, produisant la règles $u_{n+1} \rightarrow v_{n+1}$, pour chaque $n \in N$. Maintenant, il faut résoudre les problèmes suivants:

1. Comment étendre ce schéma à des schémas avec plusieurs règles?
2. Sous quelles conditions est-il possible d'effectuer l'itération précédente à l'infini?
3. Quelle est la description inductive de l'unificateur itéré ω_n et, par conséquent, la forme des règles produites $u_n \rightarrow v_n$?

Le premier problème est résolu assez facilement. Le moteur de la divergence (2.14) est étendu à une fermeture de superpositions $s_2 \rightarrowtail t_2$. Les objets (2.13) et (2.14) ne sont pas forcément distincts, ce qui nous permet de capter aussi les systèmes divergents d'une seule règle par ce schéma.

La condition demandée dans le deuxième problème est satisfaite si la fermeture de superpositions (2.14) est une chaîne de fermeture en avant. En plus, ω_n et φ_1 ne doivent pas se bloquer, ce qui sera expliqué plus tard.

La solution du troisième problème demande un raisonnement non-standard. Normalement, deux règles qui se superposent ont des variables différentes. Puisque nous sommes confrontés à une suite de règles itérées, nous allons assurer explicitement la différenciation des variables dans les règles produites par renommage de variables d'une façon globale. De plus, le désir de pouvoir décrire l'unificateur itéré ω_n inductivement nous incite à utiliser une base invariante, ce sont les variables $\mathcal{V}ar(s_2)$ de la fermeture (2.14) qui jouent ce rôle. Pour cette raison l'unificateur ω_n est décomposé en trois parties: la première partie π_n renomme les variables appropriées de $u_n \rightarrow v_n$ en $\mathcal{V}ar(s_2)$; ensuite le noyau dur — la substitution χ_n

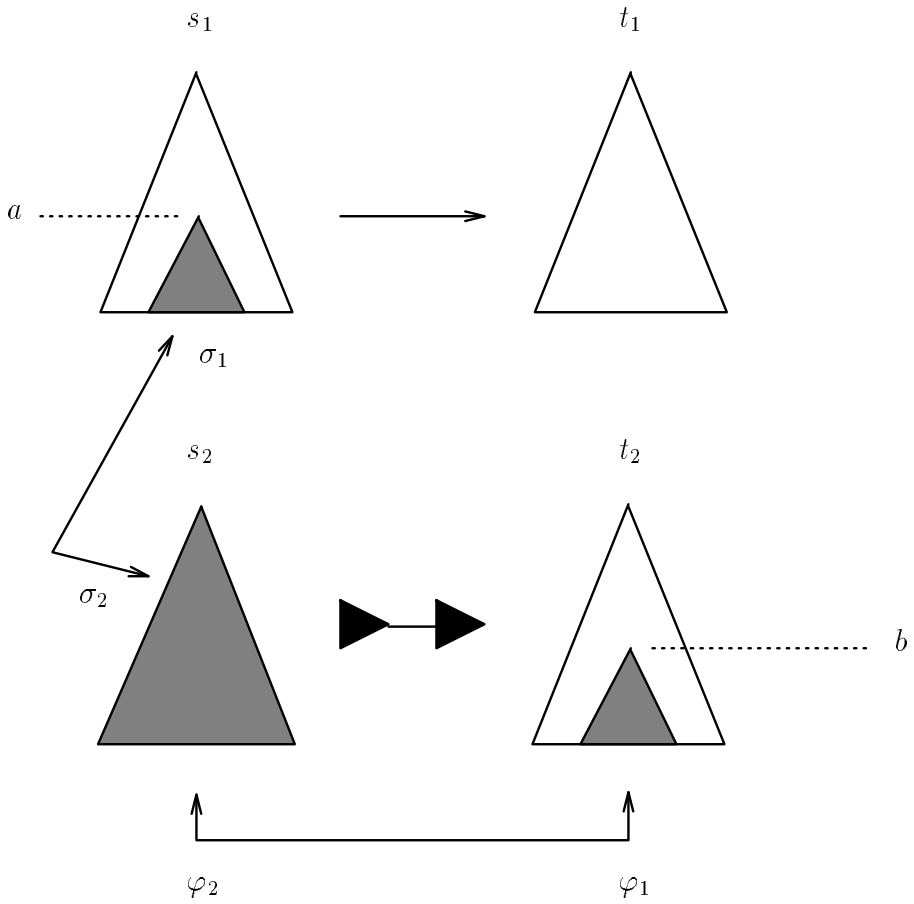


FIG. 2.5 – *Système de réécriture croisé en avant*

de domaine $\mathcal{V}ar(s_2)$ est appliquée, suivie par un renommage de variables ρ_{n+1}, transformant les variables $\mathcal{V}ar(s_2)$ en de nouvelles variables de $u_{n+1} \to v_{n+1}$. Les renommages de variables sont effectués syntaxiquement par enlèvement (π) et étiquetage (ρ) des indices aux variables $\mathcal{V}ar(s_2)$. Dans le même sens, l'unificateur de départ ω_0 est divisé en deux substitutions, σ_1 active sur (2.13) et σ_2 active sur (2.14), et un renommage de variables ρ_1.

Le processus itératif lui-même est exprimé par la substitution noyau χ_n. Comme prouvé plus tard, la substitution χ_n est exprimée comme une fonction des substitutions φ_1, φ_2 et σ_2 qui sont effectives sur la chaîne (2.14). L'opérateur T sur les substitutions de base φ_1, φ_2 et σ_2 présente le formalisme pour décrire cette itération.

La définition suivante formalise les schémas de divergence décrits ci-dessus. Elle comporte les types de divergence (Λ, γ) et $(\Lambda, \Lambda/\gamma)$ décrits par Mong et Purdom [MP87].

Définition 2.29 *La règle de réécriture $s_1 \to t_1$ et la fermeture de superpositions non-réflexive $s_2 \rightarrowtail t_2$ (avec des variables présumées disjointes) constituent un* **système de réécriture croisé en avant** *s'il existe les substitutions σ_2, φ_1, φ_2 de domaine $\mathcal{V}ar(s_2)$, une substitution idempotente σ_1 et des positions $a \in \mathcal{FP}os(s_1)$, $b \in \mathcal{FP}os(t_2)$ telles que*

1. *$\langle \sigma_1, \sigma_2 \rangle$ constitue l'unificateur principal faible de $s_1|_a$ et s_2: $s_1|_a\sigma_1 = s_2\sigma_2$*

2. $\langle \varphi_1, \varphi_2 \rangle$ *constitue l'unificateur principal faible de* $t_2|_b$ *et* s_2*:* $t_2|_b\varphi_1 = s_2\varphi_2$
3. $\mathcal{D}om(\varphi_1) \cap (\mathcal{V}ar(\varphi_2) \cup \mathcal{V}ar(\sigma_2)) = \emptyset$ *ou* $\mathcal{V}ar(\varphi_1) \cap (\mathcal{D}om(\varphi_2) \cup \mathcal{D}om(\sigma_2)) = \emptyset$ *(relation de cohérence élargie)*

Les deux premières conditions expriment les conditions de superposition. Les unificateurs principaux faibles existent, et ils sont uniques car nous travaillons avec les instances restreintes [Baa91] dans le cadre des variables de s_2. La troisième condition (appelée *relation de cohérence élargie*) assure que les substitutions σ_2, φ_1 et φ_2 ne se bloquent pas mutuellement. La présentation schématique d'un système croisé en avant se trouve dans la figure 2.5.

La définition précédente nous livre la partie statique des conditions pour décrire les systèmes divergents en avant. La partie dynamique, introduite dans la définition suivante, établit les conditions sur l'orientation des paires critiques produites.

Définition 2.30 *Le système de réécriture* R *est* **LR-persistant** *dans l'ordre* $\succ$ *si pour chaque paire critique non-triviale de termes* $\langle s_1\sigma[t_2\sigma]_a, t_1\sigma \rangle \in cp(R_\infty)$ *la relation* $s_1\sigma[t_2\sigma]_a \succ t_1\sigma$ *est valide.*

Le préfixe *LR-* indique que les paires critiques produites sont orientées de *gauche* à *droite*[3] du point de vue de la règle majeure.

Il est possible de déduire à partir des conditions de la divergence de la Définition 2.29, en supposant en même temps la LR-persistance, la description générale des schémas de divergence en avant comme une famille infinie de règles itérées pendant le processus de complétion divergent. En parallèle, on exprime la suite des unificateurs itérés intervenant dans le processus de superposition pendant la complétion, où le renommage des variables est effectué explicitement.

Définition 2.31 *Soient* $s_1 \to t_1$ *et* $s_2 \rightarrowtail t_2$ *un système de réécriture croisé comme dans la définition* 2.29*. Soit*

$$\begin{aligned} u_1 \to v_1 &= (s_1\sigma_1[t_2\sigma_2]_a)\rho_1 \to t_1\sigma_1\rho_1 \\ u_{n+1} \to v_{n+1} &= u_n\omega_n[t_2\omega_n]_{ab^n} \to v_n\omega_n \end{aligned}$$

une suite de règles de réécriture $u_n \to v_n$*, où* $\omega_n = \chi_n\rho_{n+1}$ *est l'unificateur de* $u_n|_{ab^n}$ *et* s_2*,*

$$\chi_n = (\pi_n \triangle (\varphi_1 \triangle T_{n-1}(\sigma_2, \varphi_2, \varphi_1))) \cup (\varphi_2 \triangle T_{n-1}(\sigma_2, \varphi_2, \varphi_1))$$

est la substitution noyau itérée et

$$\begin{aligned} \pi_n &= [x_n \mapsto x \mid x_n \in \mathcal{V}ar(u_n|_{ab^n})] \\ \rho_n &= [x \mapsto x_n \mid x \in \mathcal{V}ar(s_2)] \end{aligned}$$

est la paire de substitutions de pliage et dépliage effectuant le renommage des variables, dans chaque pas n *d'itération. Les règles de réécriture* $u_n \to v_n$ *forment une* **famille** *infinie* **itérée en avant**. *La famille itérée en avant, produite à partir d'un système croisé en avant* S*, est notée par* $\mathcal{I}_{FC}^{a,b}(S)$*.*

La classe de tous les systèmes croisés en avant produits pendant la complétion du système R *est notée par* $\coprod_{FC} R$*.*

3. En anglais: "*left* to *right*".

Bien qu'elle soit l'union de deux substitutions, la construction χ_n est bien définie, car $\mathcal{D}om(\pi_n) \cap \mathcal{D}om(\varphi_2) = \emptyset$.

Théorème 2.32 *Soit R en système de réécriture. Si R contient un système croisé en avant et LR-persistant S alors $nr - complete(R)$ contient la famille inifinie itérée en avant de règles $\mathcal{I}_{FC}^{a,b}(S)$.*

Preuve: Par induction sur l'indice n des règles dans la famille infinie itérée en avant de règles.

Soit $s_1 \to t_1$ et $s_2 \rightarrowtail t_2$ un système croisé en avant et LR-persistant S.

Base: $n = 1$.

Suivant la première condition dans la définition des systèmes croisés en avant, $s_1 \to t_1$ et $s_2 \rightarrowtail t_2$ produisent la paire critique $\langle u_1, v_1 \rangle$ qui est orientée dans la règle $u_1 \to v_1$ grâce à la LR-persistance. Aucune réduction n'est effectuée pendant le processus de complétion, ce qui implique que $nr - complete(R)$ contient la règle $u_1 \to v_1$.

Pas: Considérons la règle $u_{k+1} \to v_{k+1}$ et la chaîne en avant $s_2 \rightarrowtail t_2$. Nous sommes obligés de déterminer la substitution χ_{k+1} (et par conséquent l'unificateur ω_{k+1}), dérivée à partir de φ_1, φ_2 et σ_2, qui produit la paire critique par la superposition de $u_{k+1} \to v_{k+1}$ avec $s_2 \rightarrowtail t_2$. Nous allons montrer que la substitution recherchée est de la forme

$$\chi_{k+1} \quad = \quad (\pi_{k+1} \triangle (\varphi_1 \triangle T_k(\sigma_2, \varphi_2, \varphi_1))) \cup (\varphi_2 \triangle T_k(\sigma_2, \varphi_2, \varphi_1))$$

Par hypothèse et à partir de la définition 2.31 on déduit

$$u_{k+1}|_{ab^{k+1}} \quad = \quad t_2|_b(\varphi_2 \triangle T_{k-1}(\sigma_2, \varphi_2, \varphi_1))\rho_{k+1}$$

(le terme t_2 ne contient pas de variables indexées, or la première partie de la substitution χ_k — commençant par π_k — n'est pas opérationnelle et, par conséquent, est écartée). Mantenant, nous devons montrer que $u_{k+1}|_{ab^{k+1}}$ et s_2 sont unifiables par ω_{k+1}. Nous devons distiguer deux cas, d'après les deux possibilités de l'intersection vide dans la troisième condition (cohérence élargie) de la définition 2.29.

1. $\mathcal{D}om(\varphi_1) \cap (\mathcal{V}ar(\sigma_2) \cup \mathcal{V}ar(\varphi_2)) = \emptyset$.

   Dans ce cas, comme prouvé dans le Lemme 2.10, nous avons

   $$\varphi_1 \triangle T_k(\sigma_2, \varphi_2, \varphi_1) \quad = \quad (\varphi_1 \triangle \varphi_2^k) \triangle \sigma_2$$

   et, comme combinaison de plusieures propositions, aussi $\varphi_2 \triangle T_{k-1}(\sigma_2, \varphi_2, \varphi_1) = \varphi_2^k \triangle \sigma_2$. La condition $\mathcal{D}om(\sigma_2) \subseteq \mathcal{D}om(\varphi_2)$, exigée par les deux équations précédentes, n'apparaît pas parmi les conditions de la Définition 2.29, mais elle est dérivée automatiquement après le premier pas d'itération: le paragraphe suivant explique comment.

   Suivant les conditions de la définition 2.29, la procédure de complétion produit la règle $u_1 \to v_1$ décrite dans la définition 2.31. Prenons-la pour la nouvelle règle de départ $s_1' \to t_1' = u_1 \to v_1$ et ensemble avec $s_2 \rightarrowtail t_2$ effectuons le test pour les conditions de la définition 2.29. Il faut chercher les substitutions σ_1', σ_2' et la position $a' \in \mathcal{FP}os(s_1')$ (le

reste est identique car nous n'avons pas changé le moteur de la divergence $s_2 \rightarrowtail t_2$). Si $t_2|_b\varphi_1 = s_2\varphi_2$ alors nous avons

$$t_2|_b\sigma_2(\varphi_1 \triangle (\sigma_2 \triangle \varphi_1)) \quad = \quad s_2(\varphi_2 \triangle (\sigma_2 \triangle \varphi_1))$$

car $\mathcal{D}om(\varphi_1) \cap \mathcal{V}ar(\sigma_2) = \emptyset$. Pour cette raison nous considérons

$$\begin{aligned} \sigma_1' &= \pi_1 \triangle (\varphi_1 \triangle (\sigma_2 \triangle \varphi_1)) \\ \sigma_2' &= \varphi_2 \triangle (\sigma_2 \triangle \varphi_1) \end{aligned}$$

et $a' = ab$, ce qui est impliqué par $s_1'|_{a'} = u_1|_{ab} = t_2|_b\sigma_2\rho_1$. Simplement par inspection de σ_2' nous déduisons $\mathcal{D}om(\sigma_2') \supseteq \mathcal{D}om(\varphi_2)$.

Maintenant, en utilisant les lemmes 2.3 et 2.10 nous obtenons

$$\begin{aligned} & t_2|_b(\varphi_2^k \triangle \sigma_2)((\varphi_1 \triangle \varphi_2^k) \triangle \sigma_2) \\ &= t_2|_b\varphi_1(\varphi_2^k \triangle \sigma_2) \\ &= s_2(\varphi_2^{k+1} \triangle \sigma_2) \end{aligned}$$

ce qui prouve

$$\chi_{k+1} \quad = \quad (\pi_{k+1} \triangle ((\varphi_1 \triangle \varphi_2^k) \triangle \sigma_2)) \cup (\varphi_2^{k+1} \triangle \sigma_2)$$

ainsi que $u_{k+1}|_{ab^{k+1}}$ et s_2 sont unifiables par $\omega_{k+1} = \chi_{k+1}\rho_{k+2}$.

2. $\mathcal{V}ar(\varphi_1) \cap (\mathcal{D}om(\sigma_2) \cup \mathcal{D}om(\varphi_2)) = \emptyset$.

   Dans ce cas nous avons $\varphi_1 \triangle T_k(\sigma_2, \varphi_2, \varphi_1) = \varphi_1$ suivant la définition de l'opération du produit restreint $\triangle$.

   Le lemme 2.3 donne

$$\begin{aligned} & t_2|_b(\varphi_2 \triangle T_{k-1}(\sigma_2, \varphi_2, \varphi_1))\varphi_1 = \\ &= t_2|_b\varphi_1((\varphi_2 \triangle T_{k-1}(\sigma_2, \varphi_2, \varphi_1)) \triangle \varphi_1) = \\ &= s_2(\varphi_2 \triangle T_k(\sigma_2, \varphi_2, \varphi_1)) \end{aligned}$$

   ce qui prouve

$$\chi_{k+1} \quad = \quad (\pi_{k+1} \triangle \varphi_1) \cup (\varphi_2 \triangle T_k(\sigma_2, \varphi_2, \varphi_1))$$

   ainsi que $u_{k+1}|_{ab^{k+1}}$ et s_2 sont unifiables par $\omega_{k+1} = \chi_{k+1}\rho_{k+2}$.

Le raisonnement par cas précédent prouve que la construction de la substitution χ_{k+1} est correcte, ce qui signifie que $\omega_{k+1} = \chi_{k+1}\rho_{k+2}$ est l'unificateur de $u_{k+1}|_{ab^{k+1}}$ et s_2, où

$$\chi_{k+1} \quad = \quad (\pi_{k+1} \triangle (\varphi_1 \triangle T_k(\sigma_2, \varphi_2, \varphi_1))) \cup (\varphi_2 \triangle T_k(\sigma_2, \varphi_2, \varphi_1))$$

Pour cette raison la paire critique $\langle u_{k+2}, v_{k+2}\rangle$ peut être construite et orientée ensuite dans la règle $u_{k+2} \rightarrow v_{k+2}$ à cause de la LR-persistance.

Nous avons prouvé que

- $u_1 \rightarrow v_1 \in \mathcal{I}_{FC}^{a,b}(S)$ et

– si $u_k \to v_k \in \mathcal{I}^{a,b}_{FC}(S)$ alors $u_{k+1} \to v_{k+1} \in \mathcal{I}^{a,b}_{FC}(S)$,

où $S = \{s_1 \to t_1, s_2 \rightarrowtail t_2\}$ est un système croisé. Pour cette raison l'inclusion $S \subseteq R$ implique $\mathcal{I}^{a,b}_{FC}(S) \subseteq nr\text{-}complete(R)$. □

Corollaire 2.33 *Si R contient un système croisé en avant et LR-persistant alors R est faiblement divergent.*

Le corollaire suivant signifie que nous pouvons tester la présence d'un système croisé en avant à n'importe quelle étape de la procédure de complétion. Si un système de réécriture est suspect de divergence par un système croisé en avant, le processus de complétion est interrompu et le test de divergence est exécuté sur l'ensemble des règles produites.

Corollaire 2.34 *Si $s_1 \to t_1$ et $s_2 \rightarrowtail t_2$ forment un système croisé en avant S, alors pour chaque règle p produite dans le cadre de la famille itérée, $\mathcal{I}^{a,b}_{FC}(S)$ p et $s_2 \rightarrowtail t_2$ forment un nouveau système croisé en avant S'. De plus, $\mathcal{I}^{a',b}_{FC}(S') \subseteq \mathcal{I}^{a,b}_{FC}(S)$.*

Preuve: Ceci n'est pas un corollaire du Théorème 2.32 mais plutôt de sa preuve. Nous devons prouver que les substitutions partielles $\theta_n = \varphi_1 \triangle T_{n-1}(\sigma_2, \varphi_2, \varphi_1)$ et $\tau_n = \varphi_2 \triangle T_{n-1}(\sigma_2, \varphi_2, \varphi_1)$ de la substitution χ_n, définies dans la preuve mentionée, satisfont les conditions de la définition 2.29.

La condition $u_n|_{ab^n}\theta_n = s_2\tau_n$, correspondant à la première condition de la définition 2.29, a été prouvée déjà dans le théorème précédent, en utilisant la position $ab^n \in \mathcal{FP}os(u_n)$ comme remplacement de la position a. A partir de l'expression itérative prouvée pour les substitutions τ_n nous déduisons immédiatement l'équation $\mathcal{D}om(\varphi_1) \cap \mathcal{D}om(\tau_n) = \emptyset$. Il reste à prouver les parties correspondant aux deux cas dans la relation de cohérence.

1. $\mathcal{D}om(\varphi_1) \cap (\mathcal{V}ar(\sigma_2) \cup \mathcal{V}ar(\varphi_2)) = \emptyset$.

   Dans ce cas nous avons $\tau_n = \varphi_2^n \triangle \sigma_2$. Ceci implique $\mathcal{V}ar(\tau_n) \subseteq \mathcal{V}ar(\sigma_2) \cup \mathcal{V}ar(\varphi_2)$ et la condition $\mathcal{D}om(\varphi_1) \cap (\mathcal{V}ar(\tau_n) \cup \mathcal{V}ar(\varphi_2)) = \emptyset$ suit automatiquement.

2. $\mathcal{V}ar(\varphi_1) \cap (\mathcal{D}om(\sigma_2) \cup \mathcal{D}om(\varphi_2)) = \emptyset$.

   Nous avons $\mathcal{D}om(\tau_n) = \mathcal{D}om(\varphi_2)$, ce qui implique la validité de la condition $\mathcal{V}ar(\varphi_1) \cap (\mathcal{D}om(\tau_n) \cup \mathcal{D}om(\varphi_2)) = \emptyset$.

□

Considérons maintenant des exemples qui correspondent au schéma des systèmes croisés en avant. Ceci est une collection de systèmes divergents observés dans des cas réels, ainsi que des cas artificiels, construits pour trouver des conditions toujours plus générales qui couvriraient la classe des systèmes divergents la plus large.

Exemple 2.35 Tout le monde sait (cette exemple a été mentionné, entre autres, par Fribourg [Fri89] et Göbel [Göb87]) que si on extrait les règles

$$(x' + y') + z \to x' + (y' + z) \tag{2.16}$$

$$x + s(y) \to s(x + y) \tag{2.17}$$

d'un système spécifiant les entiers naturels avec addition, et si on essaie de compléter ces règles, en utilisant l'ordre $\succ_{lpo}$ avec la précédence $+ \succ s$ et le statut gauche-droite de $+$, alors le résultat sera un processus divergent. La règle (2.17) constitue la chaîne en avant, les positions sont $a = 1$ et $b = 1$, et les substitutions sont $\sigma_1 = [x' \mapsto x, y' \mapsto s(y)]$, $\sigma_2 = [\,]$, $\varphi_1 = [y \mapsto s(y)]$, $\varphi_2 = [\,]$. Ce système divergent correspond à n'importe quel type de systèmes croisés en avant selon la condition de cohérence. La procédure de complétion produit la famille infinie de règles

$$s^n(x + y) + z \quad \to \quad x + (s^n(y) + z)$$

C'est, entre autres, aussi la raison pour laquelle la preuve par récurrence [KM87] de l'associativité à partir du système de réécriture

$$\begin{aligned} x + 0 &\to x \\ x + s(y) &\to s(x + y) \end{aligned}$$

utilisant un ordre inapproprié (ici $\succ_{lpo}$ avec la précédence $+ \succ s$) entraîne une boucle infinie.

Dans [HP85] le système de réécriture

$$\begin{aligned} x + 0 &\to x && (2.18)\\ x + s(y) &\to s(x + y) && (2.19)\\ gcd(x, 0) &\to x \\ gcd(0, x) &\to x \\ gcd(x' + y', y') &\to gcd(x', y') && (2.20) \end{aligned}$$

spécifiant le plus grand commun diviseur de deux entiers naturels, a été prouvé divergent. Il contient un système croisé en avant constitué par les règles (2.19) et (2.20). La règle (2.19) forme la chaîne en avant, les positions sont $a = 1$ et $b = 1$, et les substitutions sont $\sigma_1 = [x' \mapsto x, y' \mapsto s(y)]$, $\sigma_2 = [\,]$, $\varphi_1 = [y \mapsto s(y)]$, $\varphi_2 = [\,]$. La procédure de complétion, utilisant l'ordre $\succ_{lpo}$ avec la précédence $+ \succ s$, produit la famille infinie de règles

$$gcd(s^n(x + y), s^n(y)) \quad \to \quad gcd(x, s^n(y))$$

à partir de règles (2.19) et (2.20), ainsi qu'une autre famille

$$gcd(s^n(x), s^n(0)) \quad \to \quad gcd(x, s^n(0))$$

dérivée à partir de la première par la règle (2.18).

Pour démontrer que la divergence n'est pas provoquée seulement par les spécifications des entiers naturels, regardons aussi d'autres structures algébriques.

Exemple 2.36 Un exemple simple et élégant de système croisé en avant est *l'Associativité & l'Endomorphisme*. Il a été considéré, pour d'autres raisons par Bellegarde [Bel86], BenCherifa avec Lescanne [BL87b] et Martin [Mar87]. Le système de réécriture

$$\begin{aligned} (x' + y') + z &\to x' + (y' + z) \\ f(x) + f(y) &\to f(x + y) \end{aligned}$$

en utilisant l'ordre $\succ_{lpo}$ avec la précédence $+ \succ f$ et le statut gauche-droite de $+$, produit pendant la complétion un système croisé en avant. La règle d'*Endomorphisme* forme la chaîne en avant, les positions sont $a = 1$ et $b = 1$, et les substitutions sont $\sigma_1 = [x' \mapsto f(x), y' \mapsto f(y)]$, $\sigma_2 = [\,]$, $\varphi_1 = [x \mapsto f(x), y \mapsto f(y)]$, $\varphi_2 = [\,]$. A partir de ce système, la procédure de complétion produit la famille infinie de règles

$$f^n(x + y) + z \quad \to \quad f^n(x) + (f^n(y) + z)$$

Un autre système, semblable au précédent, est *l'Associativité & Distributivité*, étudié par Lescanne [Les86], et mentioné aussi par Martin [Mar87] et Mong avec Purdon [MP87]. Le système de réécriture

$$\begin{aligned} (x' + y') + z' &\to x' + (y' + z') \\ (x * y) + (x * z) &\to x * (y + z) \end{aligned}$$

en utilisant l'ordre $\succ_{lpo}$ avec la précédence $+ \succ *$ et le statut gauche-droite de $+$, se revèle divergent pendant la complétion. C'est un système croisé en avant avec la règle de *Distributivité* formant la chaîne en avant et les substitutions $\sigma_1 = [x' \mapsto x * y, y' \mapsto x * z]$, $\sigma_2 = [\,]$, $\varphi_1 = [y \mapsto x * y, z \mapsto x * z]$, $\varphi_2 = [\,]$. C'est, en fait, une variation de l'exemple précédent, où l'opération f est interprétée comme la multiplication "currifiée".

Il n'est pas nécessaire qu'un système divergent soit formé seulement par deux règles, comme dans les exemples précédents. La cardinalité d'un système de réécriture n'est même pas limitée. De l'autre côté, une seule règle suffit à produire un système divergent.

Exemple 2.37 Il existe un système croisé d'une seule règle

$$f(g(f(x))) \quad \to \quad g(f(x))$$

introduit par Ardis [Ard80]. Cette unique règle représente les deux objets de la Définition 2.29. Les positions sont $a = 1$ et $b = 1$, les substitutions sont $\sigma_1 = [x \mapsto g(f(x))]$[4], $\sigma_2 = [\,]$, $\varphi_1 = [x \mapsto g(f(x))]$, $\varphi_2 = [\,]$. A partir de cette règle, la procédure de complétion produit la famille infinie de règles

$$f(g^n(f(x))) \quad \to \quad g^n(f(x))$$

Ce système divergent d'une seule règle contient encore un autre phénomène: si nous voulons assurer la terminaison du système produit, nous ne pouvons pas orienter ces règles dans le sens inverse. Pour cette raison la divergence de ce système est *inhérente*.

Exemple 2.38 Il existe aussi un système de réécriture formé par un ensemble de plusieures règles, où la présence de toutes les règles est nécéssaire pour maintenir la divergence. Comme prouvé dans [HP85], le système de réécriture

$$\begin{aligned} f_{n+1}(f_{n-1}(x)) &\to x \\ f_i(f_n(x)) &\to f_{i-1}(x) \qquad \text{for } i = 2, \ldots, n-1 \\ f_1(f_n(x)) &\to f_0(f_{n-1}(x)) \end{aligned}$$

4. L'ambiguïté dans les variables peut être résolue par doublage de la règle de réécriture et la division de leurs variables.

orienté par $\succ_{lpo}$ basé sur la précédence $f_i \succ f_j$ pour $i > j$, est divergent pour chaque n, mais chaque sous-ensemble propre peut être complété en un système de réécriture canonique et fini. La première règle présente la base de la divergence et le reste forme la chaîne en avant $f_{n-1}(f_n^{n-1}(x)) \rightarrowtail f_0(f_{n-1}(x))$. Les substitutions sont $\sigma_1 = [x' \mapsto f_n^{n-1}(x)]$, $\sigma_2 = [\,]$, $\varphi_1 = [x \mapsto f_n^{n-1}(x)]$, $\varphi_2 = [\,]$. A partir de ce système, la procédure de complétion produit la famille infinie de règles

$$f_{n+1}(f_0^k(f_{n-1}(x))) \quad \rightarrow \quad f_n^{k(n-1)}(x)$$

et en plus toutes les familles intermédiates infinies de règles suivant les règles qui interviennent dans la formation de la chaîne en avant.

Si quelqu'un regarde de près les exemples précédents, il peut se demander pourquoi la définition 2.29 est si compliquée. Ce n'est pas la définition qui est compliquée, mais les exemples qui sont trop faciles. Ils satisfont toutes les conditions simples de divergence, présentées dans [HP86]. Analysons maintenant des systèmes plus compliqués, où toutes les conditions de la Définition 2.29 doivent impérativement être appliquées et où les conditions de [HP86] apparaissent insuffisantes. Le seul défaut de ces exemples est leur construction artificielle, qui ne correspond à aucune structure algébrique connue.

Exemple 2.39 Un exemple plus général de système croisé en avant est[5]

$$\begin{aligned} d(x' \oplus h(y')) &\rightarrow y' \\ (x \otimes y) \oplus y &\rightarrow k_1(x \oplus k_2(y)) \end{aligned}$$

Si nous essayons de compléter ce système en utilisant l'ordre $\succ_{lpo}$ basé sur la précédence $\otimes \succ \oplus$, $\otimes \succ k_1$ et $\otimes \succ k_2$, nous obtenons un processus divergent. La deuxième règle forme la chaîne en avant. Les substitutions sont $\sigma_1 = [x' \mapsto x \otimes h(y), y' \mapsto y]$, $\sigma_2 = [y \mapsto h(y)]$, $\varphi_1 = [x \mapsto x \otimes k_2(y)]$, $\varphi_2 = [y \mapsto k_2(y)]$. A partir de ce système, la procédure de complétion produit la famille infinie de règles

$$d(k_1^n(x \oplus k_2^n(h(y)))) \quad \rightarrow \quad y$$

Un autre cas similaire présente le système croisé en avant

$$\begin{aligned} d(x' \oplus (x' \otimes y')) &\rightarrow y' \\ g(x) \oplus y &\rightarrow g(x \oplus (x \oslash y)) \end{aligned}$$

sauf qu'il s'agit du deuxième type suivant la condition de cohérence. Si nous complétons ce système en utilisant l'ordre $\succ_{lpo}$ basé sur la précédence $\oplus \succ \oslash$, $\oplus \succ g$ et le statut gauche-droite de $\oplus$, nous obtenons un processus divergent. La seconde règle forme la chaîne en avant. Les substitutions sont $\sigma_1 = [x' \mapsto g(x), y' \mapsto y]$, $\sigma_2 = [y \mapsto g(x) \otimes y]$, $\varphi_1 = [x \mapsto g(x)]$, $\varphi_2 = [y \mapsto g(x) \oslash y]$. A partir de ce système, la procédure de complétion produit la famille infinie de règles

$$d(g^n(x \oplus (x \oslash (g(x) \oslash \ldots (g^n(x) \otimes y))))) \quad \rightarrow \quad y$$

5. Les opérateurs binaires doivent être considérés comme des objets syntaxiques, pas comme des opérateurs dans une structure algébrique.

Bien sûr, tous les systèmes ne sont pas si faciles à analyser. Il existe des systèmes avec plus qu'une seule chaîne en avant ou des systèmes avec des superpositions multiples. Regardons un de ces systèmes.

Exemple 2.40 Ce système spécifie la théorie des arbres binaires signés [KK82]. Il est présenté ici comme un extrait du système de réécriture (canonique) spécifiant les groupes:

$$\begin{aligned} i(i(x)) &\to x \\ i(x*y) &\to i(y)*i(x) \\ (y*x)*i(x) &\to y \\ i(x)*(x*y) &\to y \end{aligned}$$

Si nous complétons ce système en utilisant l'ordre $\succ_{lpo}$ basé sur la précédence $i \succ *$, nous obtenons un processus divergent. D'abord, les règles suivantes

$$\begin{aligned} (y*i(x))*x &\to y \\ x*(i(x)*y) &\to y \end{aligned}$$

sont produites; ensuite le processus itératif infini commence. L'astuce ici est que le sous-système

$$\begin{aligned} i(i(x)) &\to x \\ i(x*y) &\to i(y)*i(x) \end{aligned}$$

produit un ensemble infini de chaînes en avant. Pour cette raison un ensemble infini de familles itérées infinies de règles est produit. Même si c'est un peu surprenant, c'est conforme à la théorie présentée. Le système de réécriture, qui produit les chaînes en avant, est canonique, donc décidable, ce qui indique que chaque calcul modulo chaque chaîne en avant est décidable, lui aussi.

2.3.2 Systèmes croisés en arrière

Les principes du second schéma de divergence sont similaires au premier. Pour cette raison nous nous concentrons seulement sur leur différence.

D'abord, dans le second schéma les objets qui interveiennent changent de rôles. La fermeture

$$s_1 \quad \rightarrowtail \quad t_1 \tag{2.21}$$

étant le *moteur* de la divergence, devient la règle *majeure* et

$$s_2 \quad \to \quad t_2 \tag{2.22}$$

devient la règle de *départ*. La règle produite à partir de la superposition de (2.21) et (2.22) est orientée dans le sens

$$u_1 = t_1\omega_0 \quad \to \quad s_1\omega_0[t_2\omega_0]_b = v_1 \tag{2.23}$$

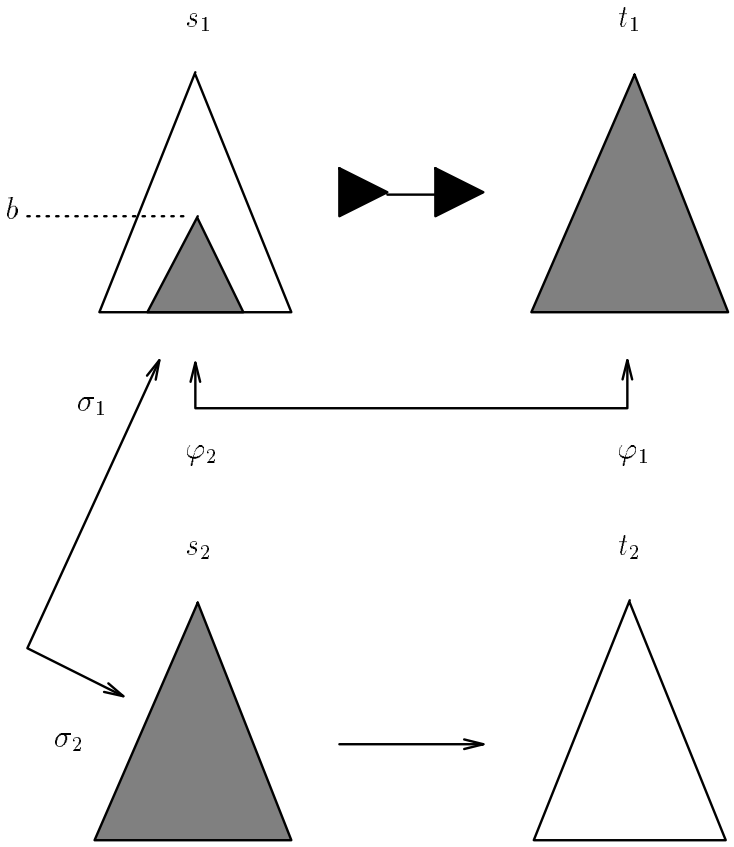


FIG. 2.6 – *Système de réécriture croisé en arrière*

c.-à.-d. exactement opposé à (2.15). Cette superposition est répétée entre (2.21) et (2.23), avec (2.23) comme la règle *mineure*. La règle produite

$$u_2 = t_1\omega_1 \quad \rightarrow \quad s_1\omega_1[v_1\omega_1]_b = v_2$$

est orientée dans le même sens que la règle (2.23). La même hypothèse d'itération infinie, avec $s_1 \rightarrowtail t_1$ et $u_n \rightarrow v_n$ qui se superposent utilisant l'unificateur ω_n, est faite comme dans le cas des systèmes croisés en avant, avec — comme résultat — la règle $u_{n+1} \rightarrow v_{n+1}$.

Maintenant, le moteur de la divergence (2.21) doit être une chaîne en arrière, avec la position $b \in \mathcal{FP}os(s_1)$ et des substitutions φ_1, φ_2, si l'itération infinie doit se maintenir.

Le seul objet invariant dans l'itération est le moteur de la divergence (2.21), alors l'unificateur ω_n et les règles $u_n \rightarrow v_n$ sont calculés par rapport aux variables $\mathcal{V}ar(s_1)$. Puisque le reste des considérations reconstituerait le raisonnement appliqué pendant le développement du premier schéma de divergence, nous ne le répéterons pas.

La définition suivante formalise le schéma de divergence avec les paires critiques orientées en arrière, qui vient d'être décrite. Il couvre les types de divergence (γ, Λ) et $(\Lambda/\gamma, \Lambda)$ introduits par Mong et Purdom [MP87].

Définition 2.41 *La fermeture de superposition $s_1 \rightarrowtail t_1$ et la règle non-réflexive $s_2 \to t_2$ (avec les variables supposées disjointes) présentent un* **système de réécriture croisé en arrière** *si t_1 n'est pas une variable, s'il existe des substitutions σ_1, φ_1, φ_2 telles que $\mathcal{D}om(\sigma_1) \subseteq \mathcal{V}ar(s_1)$, $\mathcal{D}om(\varphi_1) \subseteq \mathcal{V}ar(s_1)$ et $\mathcal{D}om(\varphi_2) \subseteq \mathcal{V}ar(s_1)$, une substitution idempotente σ_2, et une position $b \in \mathcal{FP}os(s_1)$, telles que*

1. *$\langle \sigma_1, \sigma_2 \rangle$ est l'unificateur principal faible de $s_1|_b$ et s_2: $s_1|_b\sigma_1 = s_2\sigma_2$*
2. *$\langle \varphi_1, \varphi_2 \rangle$ est l'unificateur principal faible de t_1 et $s_1|_b$: $t_1\varphi_1 = s_1|_b\varphi_2$*
3. *$\mathcal{D}om(\varphi_1) \cap (\mathcal{V}ar(\varphi_2) \cup \mathcal{V}ar(\sigma_1)) = \emptyset$ ou $\mathcal{V}ar(\varphi_1) \cap (\mathcal{D}om(\varphi_2) \cup \mathcal{D}om(\sigma_1)) = \emptyset$ (relation de cohérence élargie)*

Les deux premières conditions expriment des superpositions. La troisième condition (*cohérence élargie*) assure que les substitutions σ_1, φ_1 d'un coté et φ_2 ne se bloquent pas. La présentation schématique d'un système croisé en arrière se trouve dans la figure 2.6.

La définition précédente fournit la partie statique des conditions pour la reconnaissance des systèmes divergents en arrière. Si nous avions seulement la partie statique dans les deux cas de schémas, nous pourrions établir un *Principe de Dualité* entre les systèmes croisés en avant et en arrière, basé sur le Principe de Dualité 2.12 entre les chaînes en avant et en arrière. La partie dynamique des conditions, introduite dans la définition suivante, pose des conditions sur l'orientation des paires critiques et présente alors la différence primordiale entre les notions introduites dans les définitions 2.29 et 2.41, produisant un autre schéma complètement différent du premier.

Définition 2.42 *Le système de réécriture R is* **RL-persistant** *dans l'ordre $\succ$ si pour chaque paire critique non-triviale de termes $\langle s_1\sigma[t_2\sigma]_a, t_1\sigma\rangle \in cp(R_\infty)$ la relation $t_1\sigma \succ s_1\sigma[t_2\sigma]_a$ est valide.*

Le préfixe *RL-* signifie que les paires critiques produites sont orientées de *droite* à *gauche* du point de vue de la règle majeure.

Il est possible de déduire, à partir des conditions de la divergence de la définition 2.41, en supposant en même temps la *RL*-persistance, la description générale des schémas de divergence en arrière comme une famille infinie de règles itérées pendant le processus de complétion divergent. En parallèle, on exprime la suite des unificateurs itérés intervenant dans le processus de superposition pendant la complétion, où le renommage des variables est effectué explicitement.

Définition 2.43 *Supposons que $s_1 \rightarrowtail t_1$ et $s_2 \to t_2$ forment un système de réécriture croisé en arrière comme dans la Définition 2.41. Soient*

$$\begin{aligned} u_1 \to v_1 &= t_1\sigma_1\rho_1 \to (s_1\sigma_1[t_2\sigma_2]_b)\rho_1 \\ u_{n+1} \to v_{n+1} &= t_1\omega_n \to s_1\omega_n[v_n\omega_n]_b \end{aligned}$$

une suite de règles de réécriture, où $\omega_n = \chi_n\rho_{n+1}$ est l'unificateur de $s_1|_b$ et u_n,

$$\chi_n = (\pi_n \triangle (\varphi_1 \triangle T_{n-1}(\sigma_1, \varphi_2, \varphi_1))) \cup (\varphi_2 \triangle T_{n-1}(\sigma_1, \varphi_2, \varphi_1))$$

est la substitution noyau itérée et

$$\begin{aligned}\pi_n &= [x_n \mapsto x \mid x_n \in Var(v_n|_b)]\\ \rho_n &= [x \mapsto x_n \mid x \in Var(s_1)]\end{aligned}$$

est la paire de substitutions de pliage-dépliage pour le renommage des variables dans chaque étape n. Les règles $u_n \to v_n$ forment une **famille** *infinie* **itérée en arrière**. *La famille de règles itérée en arrière, produite à partir du système croisé S, est noté par $\mathcal{I}^b_{BC}(S)$.*

La classe de tous les systèmes croisés en arrière produits pendant la complétion du système de réécriture R est noté par $\coprod_{BC} R$.

Théorème 2.44 *Soit R un système de réécriture. Si R contient un système croisé en arrière S et s'il est RL-persistant alors nr-complete(R) contient la famille de règles infinie itérée en arrière $\mathcal{I}^b_{BC}(S)$.*

Preuve: La preuve ressemble à celle du théorème 2.32. □

Corollaire 2.45 *Si R contient un système de réécriture croisé en arrière et RL-persistant alors R est faiblement divergent.*

Corollaire 2.46 *Si $s_1 \rightarrowtail t_1$ et $s_2 \to t_2$ forment un système croisé en arrière S alors pour chaque règle de réécriture p appartenant à la famille itérée $\mathcal{I}^b_{BC}(S)$, $s_1 \rightarrowtail t_1$ et p forment un nouveau système croisé S'. De plus, $\mathcal{I}^b_{BC}(S') \subseteq \mathcal{I}^b_{BC}(S)$.*

Considérons maintenant les exemples qui appartiennent au schéma des systèmes croisés en arrière. Ce type de divergence n'est pas observé aussi souvent, autrement dit il existe moins de cas rééls connus aujourd'hui.

Exemple 2.47 Un exemple simple et élégant d'un système croisé en arrière est la théorie (décidable) des semigroupes idempotents

$$\begin{aligned}(x * y) * z &\to x * (y * z)\\ x' * x' &\to x'\end{aligned}$$

étudiée du point de vue de la divergence dans [SS82], [Der89] et [Kir89]. La règle d'*Associativité* produit un nombre infinie des chaînes en arrière indépendantes, d'après le choix multiple des superpositions.

Il faut dire, comme une remarque marginale, qu'il n'existe pas de système canonique, fini et non-conditionnel, pour les semi-groupes idempotents [SS82].

Dans ce schéma, comme dans le précédent, il existe des systèmes divergents de règles multiples, ainsi que des systèmes divergents d'une seule règle. L'exemple cohérent du système d'une seule règle est extrait d'un exemple réél.

Exemple 2.48 En extrayant la règle

$$(x\backslash y)\backslash z \to y\backslash(i(x)\backslash z)$$

du système de réécriture des groupes définis par la division à gauche, étudié par Lescanne [Les83, Les90], et l'orientant par $\succ_{lpo}$ basé sur la précédence $\backslash \succ i$ et le statut gauche-droite de $\backslash$, nous obtenons un système croisé en arrière. Comme dans l'exemple précédent, un nombre infini de chaînes en arrière indépendantes est produit à partir de la règle de départ.

Exemple 2.49 Il existe aussi des systèmes de réécriture croisés en arrière qui contiennent un nombre fini de règles, où la présence de chacune est indispensable pour le maintient de divergence. C'est le système

$$\begin{array}{rcll} f_1(f_0(x')) & \to & x' & \\ f_n(f_{i-1}(x)) & \to & f_i(f_n(x)) & \text{for } i = 2, \ldots, n-1 \\ f_n(f_{n-1}(x)) & \to & f_1(f_n(x)) & \end{array}$$

orienté par $\succ_{lpo}$ basé sur la précédence $f_i \succ f_j$ pour $i > j$. Il est divergent pour chaque n, mais chaque sous-ensemble propre de ce système peut être complété dans un système canonique et fini. La première règle présente la base de divergence et le reste forme la chaîne en arrière $f_n^{n-1}(f_1(x)) \rightarrowtail f_1(f_n^{n-1}(x))$. Les substitutions sont $\sigma_1 = [x \mapsto f_0(x)]$, $\sigma_2 = [x' \mapsto x]$, $\varphi_1 = [\,]$, $\varphi_2 = [x \mapsto f_n^{n-1}(x)]$. A partir de ce système, la procédure de complétion produit la famille infinie de règles

$$f_1(f_n^{k(n-1)}(f_0(x))) \quad \to \quad f_n^{k(n-1)}(x)$$

et, en plus, toutes les familles infinies intermédiaires, suivant les règles qui participent dans la production de la chaîne en arrière.

Comme dans le schéma précédent, des exemple artificiels reflètent mieux les conditions de la Définition 2.41.

Exemple 2.50 Considérons le système de réécriture

$$\begin{array}{rcl} f(x \vee g(y)) & \to & f(x) \vee y \\ (x' \wedge y') \vee y' & \to & y' \end{array}$$

Si nous complétons ce système, en utilisant l'ordre $\succ_{lpo}$ basé sur la précédence $f \succ \vee$, nous aboutissons à un processus divergent. La première règle forme la chaîne en arrière. Les substitutions sont $\sigma_1 = [x \mapsto x \wedge g(y)]$, $\sigma_2 = [y' \mapsto g(y)]$, $\varphi_1 = [y \mapsto g(y)]$, $\varphi_2 = [x \mapsto f(x)]$. La procédure de complétion produit la famille infinie de règles

$$f^n(x \wedge g^n(y)) \vee y \quad \to \quad f^n(g^n(y))$$

Exemple 2.51 Un cas similaire est le système croisé en arrière

$$\begin{array}{rcl} (x \otimes f(y)) \oplus y & \to & (x \oplus y) \otimes y \\ (x' \oslash y') \otimes y' & \to & x' \end{array}$$

La complétion de ce système, en utilisant l'ordre $\succ_{lpo}$ basé sur la précédence $\oplus \succ \otimes$, mène à un processus divergent. Les substitutions sont $\sigma_1 = [x \mapsto x \oslash f(y)]$, $\sigma_2 = [x' \mapsto x, y' \mapsto f(y)]$, $\varphi_1 = [y \mapsto f(y)]$, $\varphi_2 = [x \mapsto x \oplus f(y)]$. La procédure de complétion produit la famille infinie de règles

$$((((x \oslash f^{n+1}(y)) \oplus f^n(y)) \oplus \ldots \oplus f(y)) \oplus y) \otimes y \quad \to \quad ((x \oplus f^n(y))) \oplus \ldots \oplus f(y)) \oplus y$$

2.4 Comportement de la complétion en présence de simplifications

Dans les cas pratiques, la procédure de complétion avec la stratégie *complete* est supposée produire un système de réécriture inter-réduit. Même si beaucoup de systèmes faiblement divergents sont divergents, car ils satisfont la condition de la Proposition 1.2, il subsiste une classe assez large de systèmes faiblement divergents qui ne sont pas divergents du tout.

Exemple 2.52 Un exemple simple d'un système faiblement divergent qui n'est pas divergent est celui-ci:

$$\begin{array}{rcl} x \oplus (x \otimes y) & \to & x \\ f(x) \otimes y & \to & g(x \otimes y) \\ f(f(x)) \oplus g(g(y)) & \to & x \oplus y \end{array}$$

Les deux premières règles forment seules un système divergent, mais la troisième règle entraine l'application d'une étape de simplification sur l'une des paires critiques produites. Alors, on obtient le système canonique:

$$\begin{array}{rcl} x \oplus (x \otimes y) & \to & x \\ f(x) \otimes y & \to & g(x \otimes y) \\ f(x) \oplus g(x \otimes y) & \to & f(x) \end{array} \qquad \begin{array}{rcl} f(f(x)) & \to & x \\ x \oplus g(g(y)) & \to & x \oplus y \\ g(g(x \otimes y)) & \to & x \otimes y \end{array}$$

La tâche principale est de déterminer les circonstances dans lesquelles la divergence des systèmes de réécriture reste invariante, même si les règles de transition *Simplify*, *Compose* et *Collapse* sont utilisées. Cette section présente une collection de propositions pour les systèmes de réécriture où la divergence reste invariante par rapport aux règles de transition destructives.

D'abord, nous montrons que la règle *Compose* appliquée sur un système de réécriture produit ne change rien sur le processus de divergence. Or, toutes les parties droites v_n dans les familles itérées en avant, ainsi qu'en arrière, des règles $u_n \to v_n$ ne jouent aucun rôle. Leur forme actuelle n'est pas significative, à condition que la persistance soit maintenue.

Proposition 2.53 *Soit R un système de réécriture.*

- *Supposons que $s_1 \to t_1$ et $s_2 \rightarrowtail t_2$ forment un système croisé en avant LR-persistant et que $t_1 \xrightarrow{*}_R t$. Alors $s_1 \to t$ et $s_2 \rightarrowtail t_2$ forment un système croisé en avant LR-persistant aussi.*

- *Supposons que $s_1 \rightarrowtail t_1$ et $s_2 \to t_2$ forment un système croisé en arrière RL-persistant et que $t_2 \xrightarrow{*}_R t$. Alors $s_1 \rightarrowtail t_1$ et $s_2 \to t$ forment un système croisé en arrière RL-persistant aussi.*

Preuve: La preuve est effectuée seulement pour le cas des systèmes croisés en avant LR-persistants. La preuve pour l'autre cas est similaire.

Si $t_1 \xrightarrow{*}_R t$ alors soit $t = t_1$ (et il n'y a rien à faire), soit $t_1 \succ t$ dans l'ordre utilisé $\succ$, compatible avec le système R. Par transitivité, nous avons $s_1 \succ t$, ce qui prouve l'existence de la règle de réécriture $s_1 \to t$.

Dans le théorème 2.32 nous avons prouvé que, pour chaque n, la partie droite v_n est une instance du terme t_1 (pour le cas croisé en arrière RL-persistant: v_n contient l'instance de t_2; c'est la seule différence significative entre les cas). Alors $t_1 \xrightarrow{*}_R t$ implique l'existence de v'_n, où $v_n = t_1\beta_n \xrightarrow{*}_R t\beta_n = v'_n$ pour une substitution β_n (en fait, c'est le cumul des substitutions $(\sigma_1 \cup \sigma_2)\rho_1$, ω_1, ..., ω_n), pour chaque $n \in N$. Par transitivité, nous avons $u_n \succ v'_n$, ce qui prouve l'existence de la règle de réécriture $u_n \to v'_n$ sous la présence de la règle $u_n \to v_n$ pour chaque $n \in N$. Ceci prouve aussi que la LR-persistance est maintenue.

Comme il n'y a pas de conditions concernant t_1 dans la définition 2.29, si $s_1 \to t_1$ et $s_2 \rightarrowtail t_2$ forment un système croisé en avant, alors $s_1 \to t$ et $s_2 \rightarrowtail t_2$ forment un système croisé aussi. □

Le lemme suivant présente un résultat concernant l'inter-réduction dans les positions qui ne sont pas concernées par le processus de complétion.

Lemme 2.54 *Supposons que $s_1 \to t_1$ et $s_2 \rightarrowtail t_2$ ($s_1 \rightarrowtail t_1$ et $s_2 \to t_2$) se superposent dans la position $a \in \mathcal{FP}os(s_1)$ (dans la position $b \in \mathcal{FP}os(s_1)$) et qu'elles forment un système croisé en avant (en arrière). Si le terme s_1 est R-irréductible seulement dans les positions c incomparables avec a (avec b), alors $s \to t_1$ et $s_2 \rightarrowtail t_2$ ($s \rightarrowtail t_1$ et $s_2 \to t_2$) forment un système croisé en avant (en arrière) aussi, où $s_1 \xrightarrow{*}_R s$.*

Dans le cas précédent, la combinaison de ce lemme avec le corollaire 2.34 prouve que l'on peut réduire potentiellement chaque terme u_n dans la famille infinie itérée en avant $u_n \to v_n$ dans les positions qui sont incomparables avec la position ab^n et, malgré cela, le système reste croisé en avant (voir la figure 2.7). Dans le cas arrière, ceci signifie qu'on peut réduire potentiellement le terme s_1 dans la chaîne en arrière $s_1 \rightarrowtail t_1$ dans les positions qui ne sont pas comparables avec la position b et, malgré cela, le système reste croisé en arrière.

Preuve: Si $s_1 \xrightarrow{*}_R s$, où les réductions sont effectuées dans les positions incomparables avec a (avec b), alors $s_1|_a = s|_a$ ($s_1|_b = s|_b$). Seulement le sous-terme $s_1|_a$ (le sous-terme $s_1|_b$) du terme s_1 entervient dans les conditions de la Définition 2.29 (Définition 2.41), alors les conditions sont maintenues aussi si l'on remplace $s_1 \to t_1$ ($s_1 \rightarrowtail t_1$) par $s \to t_1$ ($s \rightarrowtail t_1$). □

La dernière étape présente l'analyse de la divergence même sous la présence des réductions dans les positions comparables avec celle de la superposition.

Définition 2.55 *Le système de réécriture R est* **fortement LR-persistant** *dans l'ordre $\succ$ si pour chaque R_n produit pendant une complétion $\vdash_{KB}$, pour chaque paire critique non-triviale de termes $\langle s_1\sigma[t_2\sigma]_a, t_1\sigma\rangle \in cp(R_n)$ la relation*

$$R_n(s_1\sigma[t_2\sigma]_a) \quad \succeq_{mul} \quad R_n(t_1\sigma)$$

est valide.

Le système de réécriture R est **fortement RL-persistant** *dans l'ordre $\succ$ si pour chaque R_n produit pendant une complétion $\vdash_{KB}$, pour chaque paire critique non-triviale de termes $\langle s_1\sigma[t_2\sigma]_a, t_1\sigma\rangle \in cp(R_n)$ la relation*

$$R_n(t_1\sigma) \quad \succeq_{mul} \quad R_n(s_1\sigma[t_2\sigma]_a)$$

est valide.

Dans les deux cas, $R(t)$ est l'ensemble des R-formes normales d'un terme t et $\succeq_{mul}$ est l'extention sur les multi-ensembles de l'ordre $\succeq$.

La persistance forte est un élargissement des notions introduites dans les définitions 2.30 et 2.42. Elle décrit formellement l'idée que les paires critiques restent orientées uniformement même sous l'application des règles de transition destructives. Evidemment, la persistance forte est indécidable en général. Il faut donc trouver des conditions suffisantes *ad hoc* pour les systèmes de réécriture particuliers.

Définition 2.56 *Soit $p = s \rightarrowtail t$ une chaîne de fermeture. La fermeture de superposition $q \in OC(R)$ est* **associée** *avec p dans R si $p \sqsubset_R q$ et pour chaque étape n d'itération sur la chaîne, $s_n \rightarrowtail t_n$ est différente de q. L'ensemble de toutes les fermetures de superposition associées avec p dans R est noté $\mathcal{A}^p_{OC}(R)$.*

La suite itérée $s_n \rightarrowtail t_n$ est construite comme dans la preuve du théorème 2.24.

Nota: Si nous ne distinguons pas entre les familles itérées en avant et en arrière (nous supposons qu'il s'agit de l'une ou l'autre), nous utilisons la notation $\mathcal{I}(S)$ pour une famille itérée produite à partir du système croisé S. De façon similaire, la classe de tous les systèmes croisés produits pendant la complétion du système R est notée

$$\coprod R \;=\; \coprod_{FC} R \;\cup\; \coprod_{BC} R$$

La notion suivante réduit considérablement l'espace de recherche, même si elle n'est pas indispensable dans la formulation du théorème 2.58.

Définition 2.57 *Soit $S \in \coprod R$ un système croisé produit pendant la complétion du système R. Le système S est dit* **saturé** *dans $\coprod R$ s'il n'existe aucun autre système $S' \in \coprod R$, tel que $\mathcal{I}(S) \subseteq \mathcal{I}(S')$.*

La classe de **tous les systèmes croisés en avant (en arrière) saturés** *produits pendant la complétion du système R est notée respectivement $\overline{\coprod_{FC} R}$ et $\overline{\coprod_{BC} R}$. La classe de* **tous les systèmes croisés saturés** *est notée*

$$\overline{\coprod R} \;=\; \overline{\coprod_{FC} R \cup \coprod_{BC} R}$$

Théorème 2.58 *Supposons que $p = s_1 \to t_1$ et $q = s_2 \rightarrowtail t_2$ ($q = s_1 \rightarrowtail t_1$ et $p = s_2 \to t_2$) se superposent dans la position $a \in \mathcal{FP}os(s_1)$ et forment un système croisé en avant (en arrière) fortement LR-persistant (fortement RL-persistant), saturé dans R. Si pour chaque R_n produit pendant la complétion du R*

1. *soit toute fermeture de superposition $q' \in \mathcal{A}^q_{OC}(R_n)$ associée avec q dans R_n ne se superpose pas avec p dans les positions comparables avec a;*

2. *soit il existe un fermeture de superposition $q' \in \mathcal{A}^q_{OC}(R_n)$, qui se superpose avec p et forme le système croisé en avant (en arrière) fortement LR-persistant (fortement RL-persistant) $\{p, q'\}$;*

alors R est divergent.

Preuve: La preuve est effectuée pour les systèmes croisés en avant; elle est analogue à celle pour le cas en arrière.

Suivant le corollaire 2.33, le système de réécriture R est faiblement divergent. Examinons la famille itérée $\mathcal{I}^{a,b}_{FC}(\{p, q\}) = \{u_n \to v_n\}$ en ce qui concerne l'inter-réduction dans R. Les réductions du v_n ne sont pas significatives selon la proposition 2.53. Les réductions du u_n dans les positions incomparables avec ab^n ne sont pas significatives pour la divergence non plus, selon le lemme 2.54 et le corollaire 2.34. La LR-persistance forte garantit que les règles appartenant à la famille itérée $\mathcal{I}^{a,b}_{FC}(\{p, q\})$ ne changent pas la direction en présence de l'inter-réduction.

1. S'il existait une règle $u_n \to v_n$, appartenant à la famille itérée $\mathcal{I}^{a,b}_{FC}(\{p, q\})$, telle que u_n est réductible dans une position comparable avec an^n, alors il existerait une fermeture de superposition $q' \in \mathcal{A}^q_{OC}(R_\infty)$ qui se superposerait avec p dans la position a (Rappelons que les fermetures de superposition sont des réduction cumulatives potentielles). Mais ceci est impossible selon l'hypothèse. Ceci implique la validité de l'égalité $u|_{ab^n} = u_n|_{ab}$ pour chaque étape d'itération n, où $u_n \xrightarrow{*}_{R_n} u$ et R_n est l'approximation de R_∞ produite par la procédure de complétion jusqu'à la considération de la paire critique $\langle u_n, v_n\rangle$. Ceci implique qu'il existe un morphisme injectif ξ, effectuant l'inter-réduction des règles, de la famille itérée $\mathcal{I}^{a,b}_{FC}(\{p, q\})$ dans l'ensemble R_∞. Donc, R est divergent.

2. La seconde condition assure que si l'ancien système croisé, bâti par p et q, ne produit pas une famille itérée infinie de règles, parce qu'il existe des règles de réécriture effectuant la réduction, alors il existe une fermeture de superposition $q' \in \mathcal{A}^q_{OC}(R_n)$, bâti à partir de la fermeture de superposition originale q et la règle de réduction, qui forme un nouveau système croisé avec p. Donc, R est divergent.

□

L'exemple suivant montre l'existence d'une fermeture de superposition, associée avec la chaîne de fermeture dans chaque système croisé indépendant, qui ne satisfait pas la première

condition du théorème précédent. Ceci est impliqué par le fait que le système de réécriture n'est pas divergent, même s'il est faiblement divergent.

Exemple 2.59 (suite de l'Exemple 2.52)
Les seuls systèmes indépendants dans l'Exemple 2.52 sont

$$\begin{aligned} x \oplus (x \otimes y) &\to x \\ f(x) \otimes y &\mapsto g(x \otimes y) \end{aligned} \tag{2.24}$$

et

$$\begin{aligned} x \oplus (x \otimes y) &\to x \\ f(f(x)) \oplus (f(f(y)) \otimes z) &\mapsto x \oplus (y \otimes z) \end{aligned} \tag{2.25}$$

mais l'existence de la fermeture de superposition

$$f(f(x)) \oplus (f(f(x)) \otimes y) \mapsto x$$

qui contient la fermeture (2.24) ainsi que (2.25) entraîne l'arrêt de la divergence à un moment donné de la complétion.

Le comportement de la complétion en présence de réduction peut être plus compliquée, comme dans l'exemple suivant, emprunté de [SK91]. Malgré cela, la seconde condition du théorème 2.58 reste satisfaite, donc le système est divergent.

Exemple 2.60 Le système de réécriture monadique

$$abcx \to ox \tag{2.26}$$
$$cdx \to kcx \tag{2.27}$$
$$bkx \to kbbbx \tag{2.28}$$
$$akx \to abx \tag{2.29}$$
$$odx \to ox \tag{2.30}$$

est divergent avec la famille itérée infinie

$$ab^{f(n)}cx \to ox \tag{2.31}$$

où $f(0) = 1$ et $f(n+1) = 3f(n) + 1$.

Les règles de réécriture (2.26) et (2.27) forment un système croisé en avant. Ceci produirait la règle $abkcx \to odx$, mais à cause de l'existence des règles (2.28), (2.29) et (2.30), en fait la règle $ab^4cx \to ox$ est produite. Toute la famille infinie itérée en avant serait $abk^ncx \to od^n$, mais par contre on a engendré la famille (2.31).

La règle (2.30) a une influence sur les parties droites de règles de la famille infinie engendrée et, selon la proposition 2.53, ne change pas le comportement divergent du système entier. Les règles (2.28) et (2.29) effectuent la simplification des parties gauches de la famille infinie engendrée dans les positions comparables avec a. Mais l'existence de la suite infinie de fermetures de superposition $akcd^nx \mapsto ab^{f(n)}cx$, construite à partir des règles (2.27), (2.28) et (2.29), assure l'existence d'un nouveau système croisé en avant dans chaque étape d'itération après la simplification de la partie gauche de la règle de réécriture produite pendant l'étape précédente.

2.5 Indécidabilité

L'existence d'un système croisé, ainsi qu'une autre condition suffisante de la divergence, comme sous-ensemble d'un système de réécriture est décidable seulement dans des cas explicites. En général, l'existence d'un système croisé dans $complete(R)$ est indécidable. Ceci a été prouvé par Narendran et Stillman.

Théorème 2.61 [NS89] *Il est indécidable si la procédure de complétion avec une stratégie équitable engendre un système croisé en avant ou en arrière.*

Le résultat de Narendran et Stillman a une conséquence immédiate: toutes les conditions suffisantes qui prouvent que le système d'une seule règle

$$f(g(f(x))) \quad \rightarrow \quad g(f(x))$$

est divergent sont indécidables.

2.6 Autres travaux sur la reconnaissance de la divergence

A part les systèmes croisés, il existe une seule condition de reconnaissance des systèmes de réécriture divergents, proposée par Mong et Purdom [MP87].

Mong et Purdom distinguent quatre types de divergence: (Λ, γ), $(\Lambda, \Lambda/\gamma)$, (γ, Λ) et $(\Lambda/\gamma, \Lambda)$. Les deux premiers correspondent à peu près aux systèmes croisés en avant, les deux derniers aux systèmes croisés en arrière.

Dans la notation de Mong et Purdom [MP87], $(\delta, \gamma) \Rightarrow \lambda$ signifie que des règles $\delta: g \rightarrow d$ et $\gamma: l \rightarrow r$ se superposent (ils disent "*clash*") et produisent une nouvelle règle $\lambda: p \rightarrow q$. La notation $(\delta/a, \gamma) \Rightarrow \lambda$ souligne le fait que les règles δ et γ se superposent à la position $a \in \mathcal{FP}os(l)$.

Au lieu d'utiliser le produit restreint $\varphi \triangle \psi$ de substitutions φ et ψ, ils utilisent la *restriction* des substitutions aux variables:

$$x\sigma|_V \quad = \quad \begin{cases} x\sigma & \text{si } x \in V, \\ x & \text{sinon.} \end{cases}$$

Dans la même manière, ils notent $\sigma|_t$ la restriction de la substitution σ aux variables du terme t. Ils exploitent aussi le fait que les renommages de variables sont des substitutions reversibles: si $\nu = [x \mapsto u, y \mapsto v]$ est un renommage, alors son inverse existe et c'est le renommage $\nu^{-1} = [u \mapsto x, v \mapsto y]$.

2.6.1 Type (Λ, γ)

La règle figée γ se superpose avec la règle λ_{i-1} pour produire la règle λ_i, pour chaque $i \geq 1$, où $\lambda_0 = \delta$. L'orientation de chaque paire critique $\langle L, G \rangle$ est $L \rightarrow G$.

Théorème 2.62 [MP87] *Supposons qu'il existe des règles $\delta: g \rightarrow d$ et $\gamma: l \rightarrow r$, des unificateurs σ et τ, des renommages μ et ν, et des positions a et b, tels que:*

1. $(g|_a)\sigma = l\mu\sigma$,

2. $(r|_b)\mu\sigma\tau = l\mu\sigma\nu\tau$,

3. $(\nu\tau\tau)|_{(r|_b)\mu\sigma} = (\tau\nu\tau)|_{(r|_b)\mu\sigma}$,

4. *chaque paire critique* $\langle L_i, G_i\rangle$ *est orientée en la règle* $L_i \to G_i$, *et*

5. *ni* $g|_a$, *ni* l, *ni aucun* L_i *ne sont simplifiables.*

Alors $(\lambda_{i-1}/ab^{i-1}, \gamma) \Rightarrow \lambda_i$ *pour chaque* $i \geq 1$, *où* λ_0 *est la règle* δ.

2.6.2 Type (γ, Λ)

La règle λ_i se superpose avec la règle figée γ, dans une position figée, pour produire la règle λ_{i+1}, pour chaque $i \geq 1$. Plus précisément, $(\gamma/b, \lambda_i) \Rightarrow \lambda_{i+1}$ pour une position b. L'orientation de chaque paire critique $\langle L, G\rangle$ est $G \to L$.

Théorème 2.63 [MP87] *Supposons qu'il existe des règles* $\gamma: l \to r$ *et* $\delta: g \to d$, *des unificateurs* σ *et* τ, *des renommages* μ *et* ν, *et des positions* a *et* b, *tels que:*

1. $(l|_a)\sigma = g\mu\sigma$,

2. $(l|_b)\sigma\tau = r\sigma\nu\tau$,

3. $(\nu\tau\tau)|_{r\sigma} = (\tau\nu\tau)|_{r\sigma}$,

4. *chaque paire critique* $\langle L_i, G_i\rangle$ *est orientable en la règle* $G_i \to L_i$, *et*

5. *ni* g, *ni* l, *ni aucun* G_i *ne sont simplifiés.*

Alors $(\gamma/a, \delta) \Rightarrow \lambda_1$ *et* $(\gamma/b, \lambda_i) \Rightarrow \lambda_{i+1}$ *pour chaque* $i \geq 1$.

2.6.3 Type $(\Lambda, \Lambda/\gamma)$

Chaque règle λ_{i-1} est capable de se superposer avec elle-même en produisant la nouvelle règle λ_i. Les conditions ressemblent à celles du type (Λ, γ). L'orientation de chaque paire critique produite est $L \to G$.

Théorème 2.64 [MP87] *Supposons qu'il existe une règle* $\gamma: l \to r$, *un filtre* σ, *des positions* a, b, c *et une variable* x, *tels que:*

1. $(l|_{ab})\sigma = l$ *où* ab *n'est pas la position à la racine,*

2. $r|_c = l|_{ab}$,

3. $l|_{aba} = r|_{ca} = x$, *où* x *n'apparaît plus ailleurs dans* $r|_c$,

4. *chaque paire critique* $\langle L_i, G_i{\downarrow}\rangle$ *est orientée en la règle* $L_i \to G_i{\downarrow}$, *et*

5. *la seule simplification possible est celle des* G_i.

Alors $(\lambda_{i-1}, \lambda_{i-1}) \Rightarrow \lambda_i$ *pour chaque* $i \geq 1$, *où* λ_0 *est* γ.

2.6.4 Type $(\Lambda/\gamma, \Lambda)$

Dans ce type de divergence, chaque règle λ_{i-1} se superpose avec elle-même en produisant une nouvelle règle λ_i, comme dans le type précédent, sauf que les paires critiques sont orientées dans $G \to L$. Les conditions sont similaires à celles du type (γ, Λ).

Théorème 2.65 [MP87] *Supposons qu'il existe une règle $\gamma: l \to r$, des filtres σ et τ, des positions a, b, c et une variable x, tels que:*

1. $(l|_{ab})\sigma = l$ *où* a *n'est pas à la racine,*
2. $(l|_{ab})\tau = r$,
3. $y\sigma = y\tau$ *pour* $y \in \mathcal{V}ar(l[\cdot]_{ab}) \cap \mathcal{V}ar(l|_{ab})$,
4. $l|_{aba} = r|_{ac} = x$,
5. *chaque paire critique* $\langle L_i, G_i \rangle$ *est orientée dans la règle* $G_i \to L_i{\downarrow}$, *et*
6. *la seule simplification possible est celle des* L_i.

Alors $(\lambda_{i-1}, \lambda_{i-1}) \Rightarrow \lambda_i$ *pour chaque* $i \geq 1$, *où* λ_0 *est* γ.

2.6.5 Implantation

Mong et Purdom ont implanté leurs conditions dans leur laboratoire de réécriture pour pouvoir tester la divergence de la complétion. Ce test est effectué à la demande après que le processus de comlétion (implanté comme un logiciel) soit interrompu.

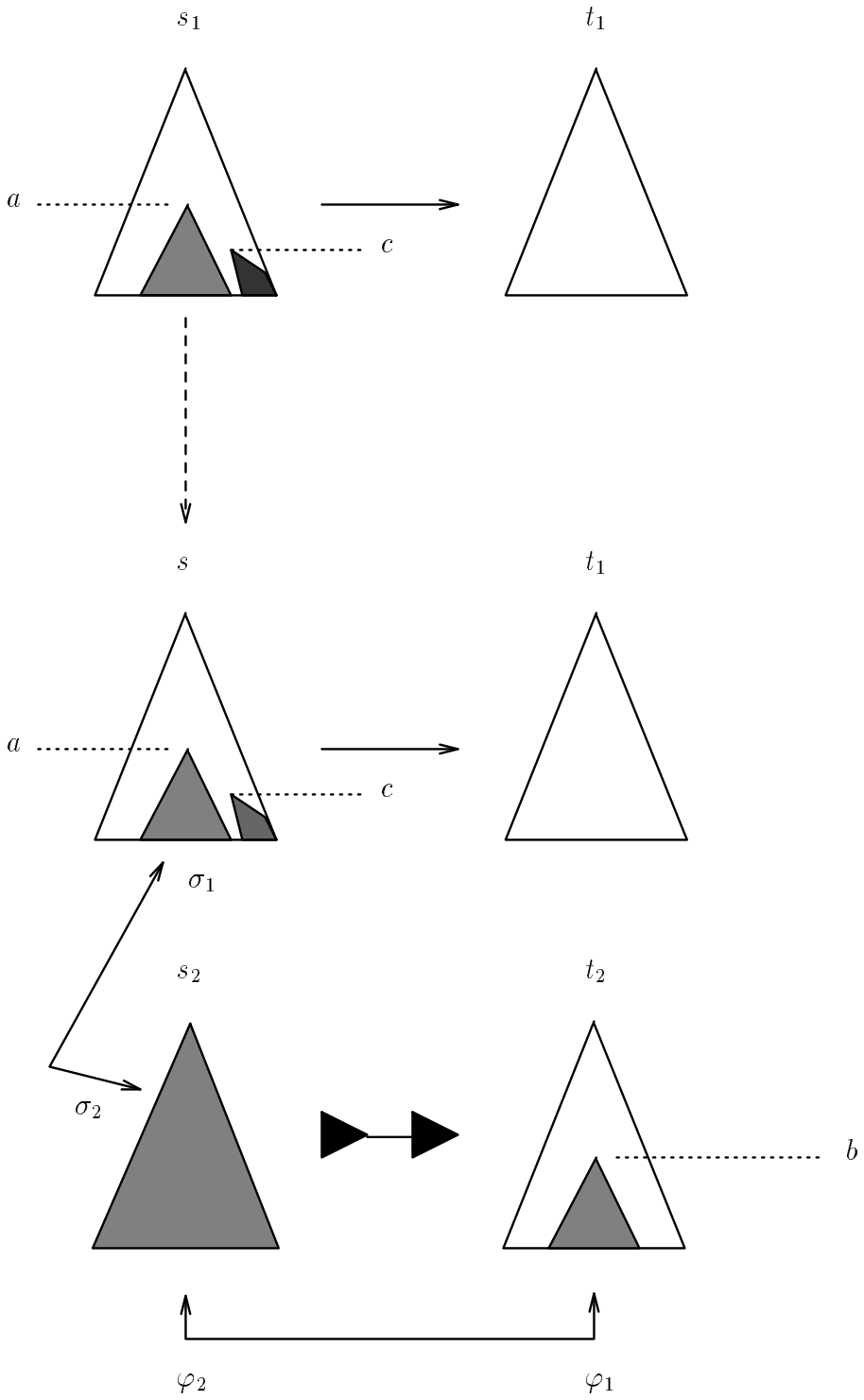


FIG. 2.7 – *Système croisé en avant: Réduction dans une position incomparable*

Chapitre 3

Moyens empiriques pour éviter la divergence

La divergence d'un système de réécriture est un désagrément avec lequel on doit vivre malgré tout. Comme l'objectif est toujours d'obtenir un système de réécriture canonique et fini, il faut agir sur le système original pour éviter sa divergence tout en maintenant sa sémantique. Il existe essentiellement deux approches: l'une théorique et l'autre empirique.

Les approches théoriques, abordés dans le Chapitre 4, présentent des méthodes générales qui permettent de décrire des systèmes divergents (infinis) par des moyens finis, dits schématisations. Ces sont les seules méthodes qui sont capables d'aborder directement des systèmes dont la divergence est inhérente. Par contre, il n'est pas toujours nécessaire d'utiliser ces techniques dès qu'un système divergent est découvert. Il est parfois suffisant d'orienter une ou plusieures règles en sens inverse. Dans d'autres cas, il suffit d'introduire un nouveau symbole lors de la scission d'une équation en deux. Dans le cas des preuves de théorèmes inductifs, la divergence est parfois due au manque d'un ou plusieurs lemmes qui sont, eux-mêmes, des théorèmes inductifs de cette théorie. Pour cette raison, il est parfois suffisant d'utiliser des méthodes empiriques qui sans être aussi sophistiquées que les approches théoriques, peuvent êtres intégrées aux laboratoires de réécriture existant en ne nécessitant qu'une modification minime du code.

Les paragraphes suivants présentent cinq méthodes empiriques pour éviter la divergence de systèmes de réécriture, suivant la complexité d'actions fournis par l'utilisateur. Il existe une méthode supplémentaire pour éviter la divergence: l'utilisation de la réécriture équationnelle et, par conséquent, de la complétion modulo une théorie équationnelle [JK86, BD87, Bac91]. Même si la réécriture équationnelle présente une généralisation puissante qui élimine plusieurs cas d'échec et de divergence, son principe est basé sur l'existence d'un algorithme d'unification équationnelle fini, qui existe rarement pour un ensemble quelconque d'égalités E. De plus, la méthode de la réécriture équationnelle comporte un autre problème potentiel, l'existence d'un ensemble complet infini d'unificateurs, ce qui présente un autre type de divergence [Kir89]. La complétion sans échec ne représente pas un outil pour éviter la divergence. Par contre, le phénomène de la divergence apparaît encore plus souvent en cas de la complétion sans échec.

Les remèdes contre la divergence proposés dans ce chapitre sont donc purement empi-

riques et peuvent être classés en cinq catégories:

1. La *modification* de l'ordre sur les termes à l'intérieur de la même classe d'ordre pour obtenir un nouvel ordre particulier du même type (par exemple, dans la classe des ordres récursifs sur les chemins on change la précédence des opérateurs) pour pouvoir orienter une égalité en sens inverse;

2. Le *choix* d'une autre classe d'ordres (par exemple, au lieu de l'ordre récursif sur les chemins on choisit l'ordre polynomial ou bien l'ordre par transformation) pour pouvoir orienter des règles de réécriture dans le sens désiré;

3. La *division* en deux règles de la règle qui entraîne la divergence: cela nécessite l'introduction d'un nouvel opérateur qui a pour arité le nombre de variables de la règle;

4. La *décomposition* de la règle qui entraîne la divergence en une conjonction d'égalités moins complexes, s'il y a des constructeurs, donc dans le cadre de la complétion inductive;

5. L'*enrichissement* du système par des nouvelles règles qui sont elles-même des théorèmes inductifs valides dans le modèle initial, dans le cadre, là aussi, de la complétion inductive.

Ces actions sont à appliquer pendant une session avec un laboratoire de réécriture du type REVE [Les83], quand une décision doit être prise pour obtenir un système de réécriture canonique. Bien sûr, ces suggestions ne produisent pas obligatoirement l'effet désiré, surtout quand le système de réécriture modifié ne correspond pas aux idées de l'utilisateur.

3.1 Modification de l'ordre à l'intérieur d'une même classe d'ordres

La première méthode empirique pour attaquer la divergence porte sur les changements à l'intérieur d'une même classe d'ordres pour obtenir un nouvel ordre à utiliser. Ceci peut être un changement de la précédence et/ou du statut sous-jacents dans un ordre récursif sur le chemins [Der87], dans un ordre de décomposition (avec statut) [Les90], ou bien dans un autre ordre récursif. Ceci peut être aussi un changement de la précédence et/ou du poids dans un ordre de Knuth et Bendix [KB70, Mar87], ou un changement dans l'interprétation polynomiale dans un ordre polynomial [BL87b].

En principe, il s'agit toujours d'un changement (relativement mineur) sur la structure de base de la classe d'ordres utilisée. Dans le cadre des ordres incrémentaux, il peut être effectué par "backtracking". Le but de ces changements est d'orienter une ou plusieures égalités dans le sens inverse, pour que les superpositions critiques, qui présentent les points de départ de la divergence, disparaissent. Cette méthode remplace des objets satisfaisants la définition 2.29 ou bien la définition 2.41. La relation d'équivalence $\overset{*}{\longleftrightarrow}$, produite à partir du système de réécriture, reste la même, mais les formes normales engendrées par le système réorienté (à condition qu'on puisse le compléter dans un système canonique et fini) ne correspondent pas

obligatoirement aux intentions initiales. Cette méthode possède aussi des limites. En effet, les changements d'orientation des règles peuvent nécessiter de sortir de la classe d'ordres considérée.

Exemple 3.1 Une des possibilités de résoudre la divergence du système de réécriture sur les entiers relatifs est le changement de la précédence à $s \succ +$ dans l'ordre récursif sur les chemins $\succ_{lpo}$ et, alors, on obtient le système canonique suivant

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

où la *deuxième* règle a été orientée en sens inverse. Même si c'est une solution possible, elle ne satisfait pas la condition pour le symbole s d'être un constructeur, si la règle $x+0 \to x$ était ajoutée.

Une autre (et meilleure) possibilité est le seul changement du statut de $+$ à droite-gauche. Dans ce cas on obtient

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

où maintenant c'est la *première* règle orientée dans le sens inverse. Le symbole du successeur s peut être déclaré comme constructeur dans le système élargi.

Le système divergent constitué de l'*Associativité* et l'*Endomorphisme* peut être résolu par le changement de la précédence à $f \succ +$ et, par conséquent, en produisant le système canonique

$$\begin{array}{rcl} (x+y)+z & \to & x+(y+z) \\ f(x+y) & \to & f(x)+f(y) \end{array}$$

où c'est aussi la deuxième règle qui a été orientée dans le sens inverse. Il pourrait s'agir d'une solution acceptable, si l'on ne voulait pas réduire le nombre de symboles f dans les termes par le système de réécriture complété.

Dans les systèmes de réécriture divergents artificiels l'orientation des règles ne joue aucun rôle, cette méthode s'applique donc sans problèmes sur les systèmes des Exemples 2.39, 2.50 et 2.51. Les systèmes canoniques suivants sont alors produits:

$$\begin{array}{rcl} d(x \oplus h(y)) & \to & y \\ k_1(x \oplus k_2(y)) & \to & (x \otimes y) \oplus y \end{array}$$

par le changement de la précédence à $k_1 \succ \otimes$ et $k_2 \succ \oplus$ dans la première partie de l'Exemple 2.39,

$$\begin{array}{rcl} d(x \oplus (x \otimes y)) & \to & y \\ g(x \oplus (x \oslash y)) & \to & g(x) \oplus y \end{array}$$

par le changement de la précédence à $g \succ \oplus$ dans la deuxième partie de l'Exemple 2.39,

$$\begin{aligned} f(x) \vee y &\rightarrow f(x \vee g(y)) \\ (x \wedge y) \vee y &\rightarrow y \end{aligned}$$

par le changement de la précédence à $\vee \succ f$, $\vee \succ g$ et le statut gauche-droite dans l"Exemple 2.50, et à la fin

$$\begin{aligned} (x \oplus y) \otimes &\rightarrow (x \otimes f(y)) \oplus \\ (x \oslash y) \otimes y &\rightarrow x \end{aligned}$$

par le changement de la précédence à[1] $\otimes \succ \oplus$, $\otimes \succ f$ et le statut gauche-droite de $\oplus$ dans l'Exemple 2.51.

Considérons maintenant des systèmes de réécriture orientés par des autres classes d'ordres, en particulier l'ordre de Knuth et Bendix et l'ordre polynomial.

Exemple 3.2 Comme évoqué dans [KB70], la présentation classique de la théorie équationnelle de groupes par le système de réécriture

$$\begin{aligned} e * x &\rightarrow x \\ i(x) * x &\rightarrow e \\ (x * y) * z &\rightarrow x * (y * z) \end{aligned}$$

orienté par l'ordre de Knuth et Bendix avec la précédence $i \succ * \succ e$, le poids $w(*) = 0$ et le poids du symbole d'inverse étant $w(i) > 0$, engendre la divergence. Comme analysé par Mong et Purdom [MP87], la procédure de complétion produit deux sous-systèmes croisés. Ce sont: le système croisé en avant

$$\begin{aligned} x * i(y * x) &\rightarrow i(y) \\ (x * y) * z &\rightarrow x * (y * z) \end{aligned} \tag{3.1}$$

et le système croisé en arrière

$$\begin{aligned} i(x) * i(y) &\rightarrow i(y * x) \\ i(x * i(y)) &\rightarrow y * i(x) \end{aligned} \tag{3.2}$$

La règle (3.2) est le réel coupable comme indiqué dans [KB70], alors elle doit être réorientée. Par conséquent, la règle (3.1) n'est plus produite. La réorientation est achevée par le changement du poids du symbole i à $w(i) = 0$. Après cette intervention, le système de réécriture standard, contenant les dix règles habituelles des groupes, est produite par complétion [Mar87].

1. Il existe aussi des autres possibilités du changement de précédence.

Exemple 3.3 Le système de réécriture composé de règles d'*Associativité* et d'*Endomorphisme* dans l'Exemple 2.36 peut être orienté dans le même sens par l'ordre polynomial avec l'interprétation $[f](X) = 2X$ et $[+](X, Y) = X^2+Y$. Il reste toujours divergent, produisant la même famille infinie de règles [BL87b]. Heureusement, ce système peut être prouvé nœthérien en utilisant une autre interprétation polynomiale $[f](X) = 2X$ et $[+](X, Y) = XY + X$. L'agrément de cette interprétation est qu'elle permet de compléter ces règles en le système canonique

$$\begin{array}{rcl} (x + y) + z & \to & x + (y + z) \\ f(x) + f(y) & \to & f(x + y) \\ f(x) + (f(y) + z) & \to & f(x + y) + z \end{array}$$

où le nombre de symboles f dans les termes décroît pendant leur réduction par le système complété.

La méthode présentée pour éviter la divergence connaît très vite ses limites. Le changement de la précédence à $* \succ i$ dans l'exemple 2.40 ne rapporte rien et dans les deux systèmes d'une seule règle il n'existe qu'un seul choix de précédence dans l'ordre récursif sur les chemins. Ce phénomène est dû à la persistance uniforme d'ordre récursif sur les chemins, alors il oriente par notoriété les paires critiques dans le même sens.

3.2 Choix d'une autre classe d'ordres

La deuxième méthode empirique est constituée du changement de la classe d'ordres. Elle est à appliquer quand la méthode précédente ne résout pas le problème de divergence (la classe originale d'ordres est monotone) ou le système canonique obtenu ne correspond pas aux intentions d'utilisateur. Même s'il existe des hiérarchies d'ordres [Rus87], une classe d'ordres n'est pas forcement "meilleure" que l'autre, car la terminaison des systèmes de réécriture est indécidable en général [Der87, HL78]. Le but de cette méthode est de rompre la persistance inhérente dans certains ordres utilisés.

La méthode proposée falsifie les conditions de la définition 2.30 ou de la définition 2.42. La relation d'équivalence $\overset{*}{\longleftrightarrow}$ reste à nouveau la même. La contribution de l'utilisateur dans cette méthode est essentielle, même si l'ordre est implanté dans le laboratoire de réécriture.

Exemple 3.4 Considérons encore une fois le système de réécriture avec les règles d'*associativité* et d'*endomorphisme*. Supposons que le symbole f correspond à une opération coûteuse et qu'on veuille utiliser ce système de réécriture pour optimiser des expressions où f apparaît. Pour cette raison, l'orientation de la règle

$$f(x) + f(y) \to f(x + y)$$

force l'utilisation de la précédence $+ \succ f$ dans l'ordre lexicographique sur les chemins $\succ_{lpo}$. Cette approche engendre la divergence sous la complétion, comme on l'a déjà constaté. Si on regarde la première règle produite

$$f(x + y) + z \to f(x) + (f(y) + z)$$

on constate qu'elle ne satisfait plus les intentions de départ avec le nombre des symboles f decroissant.

Ni l'ordre récursif sur le chemins, ni l'ordre de décomposition sur les chemins ne sont capables d'orienter ce système de réécriture comme nous le voulons. Comme on l'a vu déjà dans l'exemple 3.3, l'ordre polynomial permet de produire un système de réécriture canonique et fini. On obtient le même système canonique et fini en utilisant l'ordre de Knuth et Bendix, où on pose $w(f) > 0$ [Mar87]. Il existe aussi la possibilité d'utiliser un *ordre de transformation* [BD86, BL87a].

3.3 Division des chaînes de fermeture

La troisième méthode empirique est basée sur une idée présentée dans l'article [KB70]. L'algèbre de termes change, car la signature est élargie pendant la complétion par un nouveau symbole. Cette méthode comporte la division d'une chaîne de fermeture en deux parties différentes, introduisant un nouveau symbole, et cassant ainsi le processus d'enchaînement. On peut l'exprimer formellement par la règle de transition

$$\textit{Divide:} \quad (E \cup \{s \simeq t\}; R) \vdash (E \cup \{s \simeq f(\vec{x}), t \simeq f(\vec{x})\}; R)$$

où f est le nouveau symbole et $\mathcal{V}ar(s) \cap \mathcal{V}ar(t) = \vec{x}$. Cette méthode est spécialemment adoptée aux systèmes avec divergence inhérente et aux systèmes divergents d'une seule règle. Quelques problèmes qui peuvent apparaître en utilisant cette méthode ont été discutés dans [Ott89].

Exemple 3.5 Considérons à nouveau le système croisé en avant d'une seule règle

$$f(g(f(x))) \quad \to \quad g(f(x)) \tag{3.3}$$

observé dans l'Exemple 2.37 comme système avec la divergence inhérente. Cassons maintenant la règle (3.3) en le système de réécriture (non encore réduit)

$$\begin{aligned} f(g(f(x))) &\to h(x) \\ g(f(x)) &\to h(x) \end{aligned}$$

et étendons la précédence avec $g \succ h$. Si nous complétons le système précédent, nous obtenons le système canonique

$$\begin{aligned} f(h(x)) &\to h(x) \\ g(f(x)) &\to h(x) \\ g(h(x)) &\to h(h(x)) \end{aligned}$$

Même si cette méthode produit parfois le résultat souhaité (disparition du processus divergent), elle n'est pas un outil systématique pour arrêter la divergence. Son application à l'Exemple 2.47 avec le semi-groupe idempotent ou au système d'une seule règle

$$(x\backslash y)\backslash \quad \to \quad y\backslash(i(x)\backslash z)$$

n'a pas l'effet désiré. La division de la chaîne de fermeture donne lieu ici à un processus divergent encore plus compliqué qu'auparavant.

3.4 Décomposition des chaînes de fermeture

La procédure de complétion peut être utilisée aussi pour les preuves inductives. Les théorèmes inductifs sont valides dans le modèle initial, mais peuvent changer la théorie équationnelle. Cette observation entraîne l'utilisation de méthodes particulières pour éviter la divergence des preuves inductives, qui changent la théorie équationnelle, tout en conservant intact le modèle initial. Ce changement de la théorie équationnelle du départ peut résulter de deux mécanismes différents: soit en simplifiant un axiome de la théorie, soit en y ajoutant des axiomes nouveaux.

La première méthode utilisable en cas de divergence sur les preuves des théorèmes inductifs est applicable dans la procédure de Huet et Hullot [HH82], qui prend l'avantage de la déclaration explicite des constructeurs. La méthode propose la décomposition de la chaîne de fermeture en une conjonction d'égalités moins complexes. Formellement, on peut la décrire par la règle de transition

$$\textit{Decompose:} \quad (E \cup \{f(\vec{s}) \simeq f(\vec{t})\}; R) \vdash (E \cup \{s_i \simeq t_i \mid i \in N\}; R)$$

si f est un constructeur. Comme on peut le constater en regardant la règle de transition précédente, cette transformation est applicable uniquement aux chaînes de fermeture qui ont le même symbole à la racine de deux parties. On suppose que le processus d'enchaînement disparaît après cette transformation. La relation d'équivalence $\overset{*}{\longleftrightarrow}$ sur les termes, produite par le système original, reste la même sous la condition que f est un constructeur.

Exemple 3.6 Considérons les systèmes croisés en arrière d'une seule règle

$$f(g(f(x))) \quad \rightarrow \quad f(h(x)) \tag{3.4}$$

orienté par l'ordre récursif sur les chemins basé sur la précédence $f \succ h$ (ou $g \succ h$). La complétion de ce système engendre la famille infinie de règles

$$f(h^n(g(f(x)))) \quad \rightarrow \quad f(h^{n+1}(x))$$

Maintenant, si f est un constructeur, la règle (3.4) (considérée comme égalité) est décomposée en une nouvelle égalité, qui est orientable en la règle

$$g(f(x)) \quad \rightarrow \quad h(x)$$

en choisissant la précédence $g \succ h$. Cette nouvelle règle présente un système canonique.

3.5 Enrichissement du système par un lemme inductif

La dernière méthode proposée et probablement la plus puissante parmi les méthodes empiriques pour éviter la divergence comporte l'enrichissement du système original par des nouvelles règles. Bien sûr, il n'est pas utile d'ajouter des conséquences équationnelles des équations de départ, car ces équations sont soit éliminées au cours du processus divergent après plusieurs pas de simplification, soit elles produisent un échec. Il faut donc ajouter des

nouvelles règles qui changent la théorie équationnelle. Cette observation limite l'utilisation de cette méthode au processus divergent d'une preuve inductive, en y ajoutant des lemmes inductifs. Cette méthode s'attaque aux conditions de la Proposition 1.2, où le système enrichi reste faiblement divergent mais cesse d'être divergent. La nouvelle règle est ajoutée pour provoquer une inter-réduction ultérieure, ce qui entraîne la disparition de l'objet de la divergence à cet instant.

L'application de cette méthode se déroule en quatre étapes:

1. Enlever la(les) règle(s) de base entrainant la divergence du système R, produisant ainsi le système R'.

2. Compléter l'ensemble résiduel de règles R' dans un système canonique et fini R_2.

3. Trouver un théorème inductif $l = r$, dérivé de la structure de la famille infinie de règles produite; prouver qu'il est un théorème par rapport à R_2 par récurrence [HH82, KM87] ou bien par réducibilité inductive [JK89, KNZ87], avec utilisation possible de la méthode de Fribourg [Fri89] ou de sa généralisation [Bac88, Bac91]; et l'ajouter comme une règle au système existant R_2, produisant ainsi le système R'_2.

4. Ajouter les règles enlevées dans l'étape 1 au système R'_2, formant ainsi un système enrichi R_e. Compléter R_e.

Il existe une méthode pour prouver des théorèmes inductifs, proposée dans [HK88], qui ne demande pas que le système de base soit confluent, même sur les termes clos. Pour cette raison, l'étape 2 n'est pas indispensable si cette méthode est utilisée.

Il est parfois nécessaire d'itérer plusieurs fois ce processus pour aboutir à la solution désirée avec un système canonique et fini. Avenhaus a utilisé cette méthode dans [Ave91].

La question ouverte est l'inférence de la nouvelle règle $l \to r$, qui doit être prouvée comme un théorème inductif du système original R, avec laquelle on élargit ce système. Les approches le plus connues pour inférer un tel théorème inductif ont été présentées par Jantke avec Thomas [JT88, TJ89], par Lange et Jantke [Lan89, LJ89, LJ90], par Dershowitz et Pinchover [DP90] et par Thomas et Watson [TW90, TW93].

Si on veut prouver un théorème inductif dans le modèle initial, ce modèle ne peut pas être vide. Ceci demande l'existence au moins d'une constante. Si elle n'est pas présente explicitement, nous introduisons une constante fictive.

Exemple 3.7 Considérons un système de réécriture spécifiant des listes, avec les symboles $-$ pour la liste vide, $[\_]$ pour la liste d'un seul élément et le symbole @ pour coller deux liste ensemble, l'une après l'autre. Nous définissons le symbole *flatten* pour aplatir les liste structurées dans les listes simples[2]. Le système de réécriture est [LLT90]

$$\begin{aligned} - \ @ \ x &\to x \\ x \ @ \ - &\to x \\ (x \ @ \ y) \ @ \ z &\to x \ @ \ (y \ @ \ z) \end{aligned}$$

2. Cet exemple est construit à partir d'une preuve inductive divergente.

$$\begin{aligned} \mathit{flatten}(-) &\to - \\ \mathit{flatten}([x]) &\to \mathit{flatten}(x) \\ \mathit{flatten}([x] \mathbin{@} y) &\to \mathit{flatten}(x) \mathbin{@} \mathit{flatten}(y) \quad (3.5) \\ \mathit{flatten}(\mathit{flatten}(x)) &\to \mathit{flatten}(x) \quad (3.6) \end{aligned}$$

orienté par l'ordre récursif sur les chemins basé sur la précédence $\mathit{flatten} \succ @$ et le statut gauche-droite de @. Si on complète ce système, on obtient un processus divergent. Les règles (3.5) et (3.6) forment un système croisé en avant avec la chaîne en avant (3.5).

On enlève l'involution (3.6) et le système résiduel est déjà canonique. Nous sommes capables de prouver la règle d'endomorphisme

$$\mathit{flatten}(x \mathbin{@} y) \to \mathit{flatten}(x) \mathbin{@} \mathit{flatten}(y)$$

comme son théorème inductif. Nous l'ajoutons au système et reprenons la règle d'involution (3.6), formant ainsi le système élargi. Ce système élargi peut être complété en le système canonique et fini

$$\begin{aligned} - \mathbin{@} x &\to x \\ x \mathbin{@} - &\to x \\ (x \mathbin{@} y) \mathbin{@} z &\to x \mathbin{@} (y \mathbin{@} z) \\ \mathit{flatten}(-) &\to - \\ \mathit{flatten}([x]) &\to \mathit{flatten}(x) \\ \mathit{flatten}(x \mathbin{@} y) &\to \mathit{flatten}(x) \mathbin{@} \mathit{flatten}(y) \\ \mathit{flatten}(\mathit{flatten}(x)) &\to \mathit{flatten}(x) \end{aligned}$$

La règle à enlever doit être choisie avec précaution. Si on enlève la mauvaise règle, elle peut être regénérée au cours de la complétion du système résiduel (la tête du dragon).

Exemple 3.8 Considérons encore une fois le système introduit dans l'exemple 2.40, qu'on ne pouvait pas résoudre par les méthodes empiriques précédentes. Si l'on enlève la règle d'antimorphisme

$$i(x * y) \to i(y) * i(x)$$

et que l'on essaie de compléter le système résiduel, la règle enlevée sera regénérée à partir des autres règles.

Dans l'approche correcte, il faut enlever les règles

$$\begin{aligned} (y * x) * i(x) &\to y \\ i(x) * (x * y) &\to y \end{aligned}$$

et le système résiduel

$$\begin{aligned} i(i(x)) &\to x \\ i(x * y) &\to i(y) * i(x) \end{aligned}$$

est déjà canonique. Il n'y a pas de constante dans le système, alors il faut inventer une constante fictive, notée par e. Nous sommes capables de prouver la règle d'associativité

$$(x * y) * z \quad \rightarrow \quad x * (y * z)$$

comme étant un théorème inductif. On l'ajoute au système et reprend les règles enlevées, formant ainsi le système élargi. En complétant ce système élargi, la paire critique

$$x * i(x) \quad = \quad i(y) * y$$

est produite, qui n'est pas orientable car elle possède des variables différentes dans les parties gauche et droite de l'égalité. Ceci est considéré comme un appel pour un nouveau symbole. Alors, on divise la paire critique en les règles

$$\begin{aligned} x * i(x) &\rightarrow e \\ i(y) * y &\rightarrow e \end{aligned}$$

en utilisant le symbole e déjà introduit comme l'élément neutre avec l'élargissement de la précédence avec $* \succ e$. Après cette action, les règles sont complétables dans le système standard de groupe qui contient dix règles.

Cette méthode connaît ses limites, elle aussi, par exemple dans le cas des systèmes divergents d'une seule règle.

Exemple 3.9 Considérons encore une fois la règle

$$(x\backslash y)\backslash z \quad \rightarrow \quad y\backslash(i(x)\backslash z)$$

extraite à partir une spécification des groupes avec division à gauche. Choisissons le symbole d'inverse i comme non-constructeur. Les règles[3], qui peuvent paraître arbitraires mais sont satisfaites dans la théorie des groupes,

$$\begin{aligned} i(x\backslash y) &\rightarrow y\backslash x \\ i(i(x)) &\rightarrow x \\ i(e) &\rightarrow e \end{aligned}$$

définissent complètement le symbole i. Pour rendre possible l'orientation des nouvelles règles dans le sens désiré, la précédence doit être d'une telle façon que i et $\backslash$ soient équivalents dans cette précédence. Le système enrichi est canonique. En ajoutant encore la règle

$$x\backslash e \quad \rightarrow \quad i(x)$$

on obtient le système canonique fini

$$\begin{aligned} (x\backslash y)\backslash z &\rightarrow y\backslash(i(x)\backslash z) \\ i(x\backslash y) &\rightarrow y\backslash x \\ i(i(x)) &\rightarrow x \\ i(e) &\rightarrow e \\ x\backslash e &\rightarrow i(x) \\ e\backslash x &\rightarrow x \end{aligned}$$

3. Il est indispensable d'y ajouter l'élément neutre e comme constante.

3.6 Extensions des systèmes de réécriture

La divergence peut être évitée en étendant le système de réécriture de base par des règles supplémentaires. Il y a deux types d'extensions possibles. La première étend la signature originale $\mathcal{F}$ à une nouvelle signature $\mathcal{F}'$, et étand le système de réécriture original R en un nouveau système R' en y ajoutant des règles concernant les nouveaux symboles de $\mathcal{F}' - \mathcal{F}$. Le but est de préserver la théorie équationnelle sur les termes de la signature originale $\mathcal{F}$, plus formellement on veut que:

$$\forall s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X}): \quad s \overset{*}{\longleftrightarrow}_R t \quad \equiv \quad s \overset{*}{\longleftrightarrow}_{R'} t$$

La deuxième extension est basée sur la méthode décrite informellement dans la section 3.5 de ce document. Les élargissements du deuxième type de système de réécriture original R changent la théorie équationnelle $\overset{*}{\longleftrightarrow}_R$ mais préservent le modèle initial. Plus précisement, si R' est le système étendu et R le système original, alors

$$\forall s, t \in \mathcal{G}(\mathcal{F}): \quad s \overset{*}{\longleftrightarrow}_R t \quad \equiv \quad s \overset{*}{\longleftrightarrow}_{R'} t$$

Dans les deux cas, les nouvelles règles $R' - R$ sont synthétisées à partir de suites de règles produites pendant la divergence de la complétion grâce à l'*inférence inductive* (Jantke et Thomas [JT88, TJ89], Lange et Jantke [Lan89, LJ89, LJ90], Dershowitz et Pinchover [DP90], Thomas et Watson [TW90, TW93].

3.7 Stratégies particulières

L'un des caractéristiques des systèmes de réécriture est l'absence de stratégies d'application des règles de réécriture au cours de la réduction des termes (indéterminisme "don't care"). Cette absence de stratégie est contrebalancée par la confluence du système de réécriture qui rend toutes les réductions possibles d'un terme opérationnellement équivalentes. Or la confluence des systèmes de réécriture est une propriété difficile à atteindre, surtout à cause de la divergence. Afin de pouvoir utiliser un système de réécriture nœthérien mais non-confluent pour calculer des formes normales, il suffit parfois de définir une stratégie d'application des règles de réécriture. Bien sûr, il faut prouver la complétude de la stratégie utilisée. Une telle approche a été présentée par Paola Inverardi et Monica Nesi pour vérifier la congruence observationnelle [IN90]. Cette approche par les stratégies a été généralisée à tous les systèmes de réécriture par les mêmes auteurs dans [IN92]. Si une stratégie de réécriture est prouvée complète, il n'est plus nécessaire de compléter le système de réécriture et courir le risque d'engendrer un processus divergent.

Chapitre 4

Formalismes pour maîtriser la divergence: travaux antérieurs

Le raisonnement équationnel exige souvent l'existence d'un ensemble fini d'objets, par exemple: un ensemble fini d'axiomes, un ensemble fini d'équations, un ensemble fini de règles de réécriture, un ensemble fini d'unificateurs principaux, un ensemble fini de clauses de Horn, etc. Cette exigence est souvent imposée par la nécessité de décider si un élément de l'ensemble satisfait un prédicat donné. L'autre exigence, aussi importante que la première, est la nécessité de manipuler ces ensembles dans divers logiciels. Or, il existe des ensembles infinis d'objets qui répondent aux exigences précédentes: les ensembles récursifs. C'est le cas par exemple des ensembles infinis d'unificateurs principaux d'unification modulo l'associativité. Pour contourner cette exigence trop restrictive d'existence d'ensembles finis, nous avons besoin d'un outil qui nous permette d'exprimer finiment des ensembles infinis d'objets. Nous pourrons ainsi incorporer cet outil dans différents développements théoriques, ainsi que dans des logiciels d'application.

La *schématisation* est un formalisme pour la description finie de familles infinies de termes, règles, équations, substitutions, etc. Cette branche de recherche connaît actuellement un développement rapide. Contrairement aux extensions (la signature est étendue par des nouveaux symboles et des nouveaux axiomes sont ajoutés), les schématisations ont pour but d'exprimer, de façon finie, une infinité d'objets engendrée par un processus de déduction automatique, ceci afin de pouvoir les manipuler d'une manière précise. Différentes schématisations dans le domaine du raisonnement équationnel pour les termes du premier ordre sont apparues au cours des dernières années. Selon nos connaissances, jusqu'à maintenant il existe deux classes de schématisations en rapport avec la divergence.

1. *Schématisations par contraintes*

   Les premiers travaux de schématisation, entamés à partir de 1985, ont donné lieu aux formalismes de cette classe sans, bien entendu, à cette époque parler de contraintes. La dernière contribution à cette classe, formalisée déjà dans le cadre de contraintes, date de 1992. Le défaut principal de toutes les schématisations de cette classe est que la décidabilité de l'unification des objets schématisés n'est pas claire a priori.

Les formalismes de cette classe sont:

- les *méta-règles*, parues la première fois dans [Kir85], dont la version définitive à été publiée dans [Kir89],
- les *schémas de termes* [Gra88],
- les *contraintes d'appartenance* avec variables de contexte [Com92].

2. *Schématisations récurrentes*

   Cette classe est "plus récente" que la précédente. Chaque formalisme, qui appartient à cette classe, schématise une sous-classe des fonctions primitives récursives. Comme plusieurs formalismes ont été développés comme généralisation d'un formalisme précédent, il existe une hierarchie stricte entre les schématisations récurrentes. Leur avantage est l'existence d'un algorithme d'unification, ce qui implique la décidabilité de l'unification des objets schématisés par cette classe de formalismes.

   Les formalismes de cette classe sont:

   - les *termes récurrents*, originellement appelés les *hyper-termes* [CHK90], avec leur sous-classe, les ω-*termes* [CH91a] parfois appelés aussi les ρ-*termes* [CH91b],
   - les *termes avec des exposants entiers* [Com95],
   - le formalisme de Salzer [Sal92],
   - les *grammaires primales* [Her92].

Les *grammaires de congruence* de David McAllester [McA92] ne sont, à notre avis, rien d'autre que l'application des grammaires régulières d'arbres [GS84] à la schématisation.

Les méta-règles et les grammaires primales ont été choisies pour schématiser les familles de règles itérées infinies produites à partir des systèmes croisés pour plusieures raisons. D'abord, parce que les systèmes croisés d'une part, les méta-règles et les grammaires primales de l'autre sont très proches. En effet, les systèmes croisés et les schématisations présentent respectivement les aspects opérationnel et synthétique du même problème, tandis que le calcul des systèmes croisés fournit le moyen d'engendrer automatiquement les méta-règles et les grammaires primales.

La schématisation par des méta-règles en présence d'un nombre fini des systèmes croisés indépendants ne pose pas de problèmes particuliers. Ceci est vrai aussi pour la schématisation par des grammaires primales. De plus, il est possible de schématiser par des méta-règles, sous certaines conditions, des familles produites à partir d'un ensemble infini de systèmes croisés. Une schématisation par des grammaires primales de telles familles n'est pas encore connue. Au contraire, la schématisation par des méta-règles se révèle laborieuse en présence de systèmes croisés formés par une seule règle, car dans ce cas, la méthode standard de transformation en méta-règles est inapplicable. Au contraire, la schématisation de tels systèmes par des grammaires primales ne présente aucun problème particulier.

4.1 Méta-règles

Nota: Dans ce paragraphe, l'expression "schématisation" sans autre précision signifie une "schématisation par des méta-règles".

La notion formelle de schématisation par méta-règles à été proposée par Hélène Kirchner dans [Kir85, Kir89], mais sans proposer de moyen systématique de leur construction à partir d'un système croisé. Les méta-règles sont en général des règles de réécriture conditionnelles, où les conditions expriment que les méta-variables appartiennent à des domaines récursivement énumérables. La réécriture avec méta-règles est donc une approche voisine de la réécriture avec contraintes d'appartenance [Com92, Toy88]. Dans [Kir89], Hélène Kirchner a prouvé que, sous certaines conditions sur les domaines des méta-variables, la réécriture avec les méta-règles peut être interprétée par la réécriture à sortes ordonées, c'est-à-dire avec contraintes d'appartenance. La méthode de construction systématique des méta-règles à partir des systèmes croisés a été présentée dans [KH90].

L'exemple suivant donne une idée intuitive des notations introduites dans la suite.

Exemple 4.1 (Suite de l'exemple 2.36)
Considérons l'équivalence $\overset{*}{\longleftrightarrow}_Q$ engendrée par le système de réécriture

$$Q \quad = \quad \{f(x) + f(y) \to f(x + y)\}$$

La suite des règles produites $(l_n \to r_n)$ peut se généraliser modulo Q par

$$(\dot{x} + \dot{y}) + \dot{z} \quad \to \quad \dot{x} + (\dot{y} + \dot{z})$$

en utilisant les substitutions qui associent aux vecteur $\langle \dot{x}, \dot{y}, \dot{z} \rangle$ chaque valeur dans l'ensemble des vecteurs de termes

$$P_\Psi(\langle \dot{x}, \dot{y}, \dot{z} \rangle) \quad = \quad \{\langle x_1, x_2, z \rangle, \langle f(x_1), f(x_2), z \rangle, \ldots, \langle f^n(x_1), f^n(x_2), z \rangle, \ldots\}$$

La méta-règle est la règle d'associativité avec ses variables transformées en méta-variables:

$$(\dot{x} + \dot{y}) + \dot{z} \quad \to \quad \dot{x} + (\dot{y} + \dot{z}) \quad \| \quad \langle \dot{x}, \dot{y}, \dot{z} \rangle \in^? P_\Psi(\langle \dot{x}, \dot{y}, \dot{z} \rangle)$$

Le symbole $\in^?$ signifie la contrainte d'appartenance à un ensemble.

Nota: Les paragraphes 4.1.1, 4.1.2 et 4.1.3 sont repris complétement à partir de l'article [Kir89].

4.1.1 Schématisation

Des variables génériques, appelées *méta-variables*, sont introduites pour pouvoir schématiser des ensembles infinis de règles par des moyens finis. L'ensemble des instances possibles est toujours associé à ces variables.

Notons par $\dot{\mathcal{X}}$ l'ensemble des *méta-variables*, où $\mathcal{X} \cap \dot{\mathcal{X}} = \emptyset$. Notons par $\mathcal{T}(\mathcal{F}, \dot{\mathcal{X}})$ l'ensemble des *méta-termes* avec méta-variables dans $\dot{\mathcal{X}}$ et symboles dans $\mathcal{F}$. En principe, des identificateurs pointés sont utilisés pour les méta-objets.

Définition 4.2 *Une* **instance principale** *est une application* $\dot{\psi}: \dot{\mathcal{X}} \longrightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$.

Dans cette définition, une méta-variable peut s'instancier en un terme réduit à une seule variable. Ceci permet de considérer, par exemple dans l'exemple 4.1, la méta-variable $\dot{z}$ avec, comme instance, l'unique variable z. Cette approche évite l'introduction d'un autre ensemble de variables pour les images des instances principales.

Il est parfois plus avantageux de présenter l'ensemble d'instances principales Ψ du vecteur des méta-variables $\vec{\dot{x}} = \langle \dot{x}_1, \ldots, \dot{x}_n \rangle$ implicitement par l'ensemble $P_\Psi(\vec{\dot{x}}) = \{\langle \dot{x}_1\dot{\psi}, \ldots, \dot{x}_n\dot{\psi} \rangle \mid \dot{\psi} \in \Psi\}$.

Définition 4.3 *Une* **contrainte** *associée à l'ensemble d'instances principales Ψ est l'expression $\vec{\dot{x}} \in^? P_\Psi(\vec{\dot{x}})$ où $\vec{\dot{x}} = \langle \vec{x}_1, \ldots, \vec{x}_n \rangle$ est un vecteur de méta-variables et*

$$P_\Psi(\vec{\dot{x}}) \quad = \quad \{\langle \dot{x}_1\dot{\psi}, \ldots, \dot{x}_n\dot{\psi} \rangle \mid \dot{\psi} \in \Psi\}$$

l'ensemble d'instances principales de $\vec{\dot{x}}$.

Une **solution** *d'une contrainte associée $\vec{\dot{x}} \in^? P_\Psi(\vec{\dot{x}})$ est une méta-substitution $\dot{\sigma}: \dot{\mathcal{X}} \longrightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$ telle que $\mathcal{D}om(\dot{\sigma}) = \vec{\dot{x}}$ et $\vec{\dot{x}}\dot{\sigma} = \langle t_1\gamma, \ldots, t_n\gamma \rangle$ pour un vecteur de termes $\langle t_1, \ldots, t_n \rangle \in P_\Psi(\vec{\dot{x}})$ et une substitution $\gamma: \mathcal{X} \longrightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$.*

L'ensemble des solutions de la contrainte $\vec{\dot{x}} \in^? P_\Psi(\vec{\dot{x}})$ est noté $[\![P_\Psi(\vec{\dot{x}})]\!]$.

Définition 4.4 *Une* **méta-règle** *est une règle contrainte*

$$\dot{g} \quad \rightarrow \quad \dot{d} \quad \| \quad \vec{\dot{x}} \in^? P_\Psi(\vec{\dot{x}})$$

où $\dot{g}, \dot{d} \in \mathcal{T}(\mathcal{F}, \dot{\mathcal{X}})$ et $\vec{\dot{x}} = \mathcal{V}ar(\dot{g})$.

Une **schématisation par méta-règles** *$\mathcal{S} = (MR, \Psi, Q)$ est définie par un ensemble* fini *MR de* méta-règles*, un ensemble d'*instances principales *Ψ et un système de réécriture convergent Q qui produit la* **congruence de schématisation** *$\overset{*}{\longleftrightarrow}_Q$ sur $\mathcal{T}(\mathcal{F}, \mathcal{X})$.*

Si Ψ contient la méta-substitution $[\dot{x}_1 \mapsto x_1, \ldots, \dot{x}_n \mapsto x_n]$ pour chaque vecteur de méta-variables $\vec{\dot{x}}$, la schématisation $\mathcal{S} = (MR, \Psi, Q)$ est dite **sans contrainte**.

Pour pouvoir reproduire l'ensemble infini de règles à partir d'une méta-règle, les méta-variables doivent être instanciées par les instances principales. La notion d'instance principale exprime le fait que la méta-règle est la généralisation de toutes les règles dans la famille infinie de règles. La notion de congruence de schématisation permet d'appliquer la relation d'équivalence évoquée dans l'exemple 4.1 et de généraliser modulo une théorie.

La schématisation $\mathcal{S} = (MR, \Psi, Q)$ représente la famille infinie de règles

$$\{\dot{g}\dot{\psi}\downarrow_Q \rightarrow \dot{d}\dot{\psi}\downarrow_Q \mid \dot{\psi} \in \Psi, (\dot{g} \rightarrow \dot{d} \parallel \vec{\dot{x}} \in^? P_\Psi(\vec{\dot{x}})) \in MR\}$$

Cette famille est engendrée par l'application, sur chaque méta-règle $\dot{g} \rightarrow \dot{d} \parallel \vec{\dot{x}} \in^? P_\Psi(\vec{\dot{x}})$, de toutes instances principales $\dot{\psi} \in \Psi$, suivie par la réduction aux formes normales des termes instanciés par rapport à Q, notées par $\downarrow_Q$.

4.1.2 Réécriture et schématisation

Il faut définir une notion adéquate de réécriture dans une schématisation donnée et trouver des propriétés qui doivent être satisfaites par cette relation de réécriture.

La réécriture associée à une schématisation tient compte de la congruence de schématisation ainsi que des instances principales.

Définition 4.5 *La* **relation de réécriture associée à la schématisation** $\mathcal{S} = (MR, \Psi, Q)$ *est définie sur* $\mathcal{T}(\mathcal{F}, \mathcal{X})$ *par:*

$$t \quad \Longrightarrow_{\mathcal{S}}^{u, \dot{g} \to \dot{d} \parallel \dot{\vec{x}} \in^? P_\Psi(\dot{\vec{x}})} \quad t'$$

s'il existe une méta-règle $(\dot{g} \to \dot{d} \parallel \dot{\vec{x}} \in^? P_\Psi(\dot{\vec{x}})) \in MR$ *et une solution* $\dot{\sigma}$ *de la contrainte* $\dot{\vec{x}} \in^? P_\Psi(\dot{\vec{x}})$ *telles que* $t|_u \overset{*}{\longleftrightarrow}_Q \dot{g}\dot{\sigma}$ *et* $t' = t[\dot{d}\dot{\sigma}]_u$.

On écrit parfois $\Longrightarrow$ au lieu de $\Longrightarrow_{\mathcal{S}}$ si le contexte détermine sans ambiguïté une schématisation donnée. Soit $\overset{*}{\Longleftrightarrow}^{Q}$ la notation pour la fermeture réflexive, symétrique et transitive de la relation $(\Longrightarrow \cup \longrightarrow_Q)$.

Dans une schématisation, les méta-variables sont typées par leurs instances principales.

Définition 4.6 *Soit* $\mathcal{S} = (MR, \Psi, Q)$ *une schématisation. La* **contrainte de sorte** *sur une méta-variable* $\dot{x} \in \dot{\mathcal{X}}$ *est l'expression* $\dot{x}{:}\mathcal{S}ort(\dot{x})$ *où*

$$\mathcal{S}ort(\dot{x}) \quad = \quad \{t \mid t \overset{*}{\longleftrightarrow}_Q \dot{x}\dot{\sigma},\ \dot{\sigma} \in [\![P_\Psi(\dot{x})]\!]\}$$

Une interprétation de la relation de réécriture $\Longrightarrow$ dans les sortes ordonnées est développée dans [Kir89]. Informellement, une méta-variable $\dot{x}$ est identifiée avec une variable typée $\dot{x}{:}\mathcal{S}ort(\dot{x})$. L'ensemble de sortes est ordonné par inclusion, avec une sorte principale, notamment $\mathcal{T} = \mathcal{T}(\mathcal{F}, \mathcal{X})$. Chaque opérateur $f \in \mathcal{F}$ d'arité n est associé à une déclaration de sorte $f{:}\mathcal{T}^n \to \mathcal{T}$. Les méta-règles deviennent alors des règles à sortes ordonnés. La schématisation, pour laquelle une telle interprétation est possible, s'appelle *bien sortée*.

4.1.3 Décision de l'équivalence avec schématisation

Nous voulons utiliser l'ensemble de méta-règles MR pour décider l'équivalence engendrée par R_∞, c.-à.-d. par la fermeture réflexive, symétrique et transitive de la relation de réécriture $\longrightarrow_{R_\infty}$, notée $\overset{*}{\longleftrightarrow}_{R_\infty}$. Pour cette raison, nous introduisons deux propriétés de schématisation, à savoir, la *consistance* et la *complétude* par rapport à R_∞. La consistance signifie que toutes les déductions effectuées par les méta-règles sont des conséquences équationnelles de R_∞.

Définition 4.7 *La schématisation* $\mathcal{S} = (MR, \Psi, Q)$ *est* **consistante** *par rapport à un système de réécriture infini* R_∞ *si, pour tous les termes* t *et* t', *la relation* $t \overset{*}{\Longleftrightarrow}^{Q}_{MR} t'$ *implique la relation* $t \overset{*}{\longleftrightarrow}_{R_\infty} t'$.

La consistance d'une schématisation est assurée si toutes instances principales d'une méta-règle sont des conséquences équationnelles du système de réécriture infini R_∞.

Proposition 4.8 [Kir89] *$\mathcal{S} = (MR, \Psi, Q)$ est une schématisation consistante de R_∞ si la relation $\overset{*}{\longleftrightarrow}_Q$ est incluse dans la congruence $\overset{*}{\longleftrightarrow}_{R_\infty}$: ($\overset{*}{\longleftrightarrow}_Q \subseteq \overset{*}{\longleftrightarrow}_{R_\infty}$), et pour chaque méta-règle $(\dot{g} \to \dot{d} \parallel \vec{\dot{x}} \in^? P_\Psi(\vec{\dot{x}})) \in MR$ et pour chaque instance principale $\dot{\psi}$ nous avons $\dot{g}\dot{\psi} \overset{*}{\longleftrightarrow}_{R_\infty} \dot{d}\dot{\psi}$.*

Il est important d'exiger l'existence d'un ensemble *fini* de méta-règles dans la schématisation, car dans le cas contraire l'ensemble infini R_∞ pourrait bien tenir lieu de schématisation consistante de lui-même avec l'égalité syntaxique comme congruence de schématisation Q. Evidemment, seules les schématisations *finies* sont intéressantes.

Une condition suffisante simple peut être proposée pour assurer la deuxième condition de la proposition 4.8 pour une méta-règle $\dot{g} \to \dot{d} \parallel \vec{\dot{x}} \in^? P_\Psi(\vec{\dot{x}})$. Si l'instance $\dot{\psi}_0$ qui transforme chaque méta-variable $\dot{x}$ en une variable $x \in \mathcal{X}$ est une instance principale, la deuxième condition de la proposition 4.8 est équivalente à la condition suivante: pour chaque méta-règle $(\dot{g} \to \dot{d} \parallel \vec{\dot{x}} \in^? P_\Psi(\vec{\dot{x}})) \in MR$: $\dot{g}\dot{\psi}_0 \overset{*}{\longleftrightarrow}_{R_\infty} \dot{d}\dot{\psi}_0$. Une condition suffisante plus générale à été proposée par Hélène Kirchner dans [Kir89].

La consistance n'est pas suffisante pour décider l'équivalence modulo l'ensemble de méta-règles dans l'algèbre quotientée $\mathcal{T}(\mathcal{F}, \mathcal{X})/R_\infty$. La schématisation doit en outre satisfaire la propriété de complétude, qui exprime le fait que la réécriture avec une schématisation est une méthode complète pour prouver des théorèmes équationnels dans $\mathcal{T}(\mathcal{F}, \mathcal{X})$.

Définition 4.9 *La schématisation $\mathcal{S} = (MR, \Psi, Q)$ est* **complète** *par rapport à un système de réécriture infini R_∞ si la relation $\overset{*}{\longleftrightarrow}_{R_\infty}$ est incluse dans la relation $\overset{*}{\Longrightarrow}_{\mathcal{S}} . \overset{*}{\longleftrightarrow}_Q . \overset{*}{\Longleftarrow}_{\mathcal{S}}$:*

$$\overset{*}{\longleftrightarrow}_{R_\infty} \quad \subseteq \quad \overset{*}{\Longrightarrow}_{\mathcal{S}} . \overset{*}{\longleftrightarrow}_Q . \overset{*}{\Longleftarrow}_{\mathcal{S}}$$

Chaque fois que $\Longrightarrow_{\mathcal{S}}$ est une réécriture modulo Q, la preuve de complétude est décomposable en deux parties:

Proposition 4.10 [Kir89] *La schématisation $\mathcal{S} = (MR, \Psi, Q)$ est complète par rapport à un système de réécriture infini R_∞ si la relation $\overset{*}{\longleftrightarrow}_{R_\infty}$ est incluse dans la relation $\overset{*}{\Longleftrightarrow}^Q_{\mathcal{S}}$ et si l'ensemble de méta-règles MR est convergent modulo Q.*

La première condition est assurée par la méthode de production des méta-règles à partir des systèmes croisés. Par contre, l'ensemble des méta-règles d'une schématisation ne satisfait pas toujours la propriété de Church-Rosser et, par conséquent, une procédure de complétion des méta-règles est indispensable. Les techniques présentées dans [Kir89] donnent une procédure de complétion des méta-règles pour transformer une schématisation consistante en une schématisation consistante et complète. Cette procédure s'appuie sur la complétion à sortes ordonnées modulo Q [GKK90] dans le cas des schématisations bien sortées, et sur la complétion modulo Q [Bac91, JK86, PS81] pour les schématisations sans contraintes.

Nota: Ici s'arrête la partie reprise de l'article [Kir89].

4.1.4 Production systématique des méta-règles

Nous présentons ici une méthode systématique de production des méta-règles pour la construction des schématisations consistantes en deux étapes. La première présente la schématisation d'une seule famille itérée produite à partir d'un système croisé. Bien sûr, le système infini R_∞ peut contenir plusieures familles itérées sans inclusion mutuelle. Pour cette raison, l'étape suivante présente la schématisation du système infini R_∞ entier, fondée sur la synthèse des schématisations de toutes les familles itérées. Cette technique est illustrée par un exemple assez complexe qui contient un ensemble infini de systèmes croisés.

Il faut se rappeler que les systèmes croisés produisent des familles itérées en avant et en arrière. A partir de la description de ces types de familles, une méta-règle peut se construire systématiquement dans tous les cas à condition que la fermeture de superpositions soit elle-même un système de réécriture convergent. En fait, la méta-règle est la règle elle-même avec des variables spéciales devenues des méta-variables et la fermeture de superpositions constitue la congruence de schématisation.

Systèmes croisés en avant

Proposition 4.11 *Supposons que $\mathcal{I}^{a,b}_{FC}(C) = \{u_n \to v_n\}$ forme une famille infinie itérée en avant produite à partir du système croisé en avant $C = \{s_1 \to t_1, s_2 \mapsto t_2\}$ comme dans la Définition 2.31, où $rd_C(s_2 \mapsto t_2)$ constitue lui-même un système de réécriture convergent. Considérons la schématisation sans contrainte $\mathcal{S} = (MR, \Psi, Q)$ suivante:*

- *la méta-règle $\dot{s}_1 \to \dot{t}_1 \parallel \dot{\vec{x}} \in^? P_\Psi(\dot{\vec{x}})$ est formée à partir de la règle $s_1 \to t_1$ par la transformation de chaque variable $x \in \mathcal{V}ar(s_1)$ en une méta-variable $\dot{x}$, et $MR = \{\dot{s}_1 \to \dot{t}_1 \parallel \dot{\vec{x}} \in^? P_\Psi(\dot{\vec{x}})\}$;*
- *$P_\Psi(\dot{\vec{x}}) = \{\vec{x}\} \cup \{\vec{x}\sigma_1\rho_1\omega_1 \ldots \omega_n \mid n \geq 0\}$, où $\vec{x} = \mathcal{V}ar(s_1)$;*
- *$Q = rd_C(s_2 \mapsto t_2)$.*

Alors pour chaque n, $u_n \overset{*}{\Longleftrightarrow}^{Q}_{\mathcal{S}} v_n$. De plus, cette schématisation est consistante par rapport à la famille $\mathcal{I}^{a,b}_{FC}(C)$.

Preuve: Il est facile de déduire par induction à partir des expressions pour u_{n+1} et v_{n+1} que $v_{n+1} = t_1\sigma_1\rho_1\omega_1 \ldots \omega_n$ et $u_{n+1} \overset{*}{\longleftrightarrow}_Q s_1\sigma_1\rho_1\omega_1 \ldots \omega_n$ car $u_{n+1} = u_n\omega_n[t_2\omega_n]_{ab^n}$ et $t_2\omega_n \overset{*}{\longleftrightarrow}_Q s_2\omega_n = u_n|_{ab^n}\omega_n$ implique $u_{n+1} \overset{*}{\longleftrightarrow}_Q u_n\omega_n$. La schématisation est sans contrainte car le vecteur des variables $\vec{x} = \mathcal{V}ar(s_1)$ devient un vecteur de méta-variables $\dot{\vec{x}}$ avec ses instances principales

$$P_\Psi(\dot{\vec{x}}) \quad = \quad \{\vec{x}, \vec{x}\sigma_1\rho_1, \vec{x}\sigma_1\rho_1\omega_1, \ldots, \vec{x}\sigma_1\rho_1\omega_1 \ldots \omega_{n-1}\omega_n, \ldots\}$$

où $\langle x_1, \ldots, x_n\rangle\sigma = \langle x_1\sigma, \ldots, x_n\sigma\rangle$. Naturellement, $u_{n+1} \Longrightarrow v_{n+1}$. En plus, la schématisation est consistante car la méta-règle est uniquement une règle du système divergent avec des variables transformées en méta-variables. □

Exemple 4.12 (Suite de l'exemple 2.36).
Les variables x', y' et z' de la règle d'*Associativité* deviennent des méta-variables $\dot{x}$, $\dot{y}$ et $\dot{z}$. Les

instances principales du vecteur des méta-variables $\langle \dot{x}, \dot{y}, \dot{z} \rangle$ sont présentées par l'ensemble $P_\Psi(\langle \dot{x}, \dot{y}, \dot{z} \rangle) = \{\langle f^n(x), f^n(y), z \rangle \mid n \geq 0\}$. La méta-règle est

$$(\dot{x} + \dot{y}) + \dot{z} \quad \rightarrow \quad \dot{x} + (\dot{y} + \dot{z}) \quad \| \quad \langle \dot{x}, \dot{y}, \dot{z} \rangle \in^? \{\langle f^n(x), f^n(y), z \rangle \mid n \geq 0\}$$

et le système de réécriture convergent Q est $\{f(x) + f(y) \rightarrow f(x + y)\}$.

Les systèmes dans les exemples 2.35, 2.38 et 2.39 peuvent être traités de la même façon.

Systèmes croisés en arrière

Proposition 4.13 *Supposons que $\mathcal{I}^b_{BC}(C) = \{u_n \rightarrow v_n\}$ forme une famille infinie itérée en arrière produite à partir du système croisé en arrière $C = \{s_1 \rightarrowtail t_1, s_2 \rightarrow t_2\}$ comme dans la définition 2.43, où $rd_C(s_1 \rightarrowtail t_1)$ représente lui-même un système de réécriture convergent. Considérons la schématisation sans contrainte $\mathcal{S} = (MR, \Psi, Q)$ suivante:*

- *la méta-règle $\dot{s}_2 \rightarrow \dot{t}_2 \parallel \dot{\vec{x}} \in^? P_\Psi(\dot{\vec{x}})$ est formée à partir de la règle $s_2 \rightarrow t_2$ par transformation de chaque variable $x \in \mathcal{V}ar(s_2)$ en une méta-variable $\dot{x}$, et $MR = \{\dot{s}_2 \rightarrow \dot{t}_2 \parallel \dot{\vec{x}} \in^? P_\Psi(\dot{\vec{x}})\}$;*
- *$P_\Psi(\dot{\vec{x}}) = \{\vec{x}\} \cup \{\vec{x}\sigma_2\rho_1\omega_1 \ldots \omega_n \mid n \geq 0\}$, où $\vec{x} = \mathcal{V}ar(s_2)$;*
- *$Q = rd_C(s_1 \rightarrowtail t_1)$.*

Alors pour chaque n, $u_n \overset{*}{\Longleftrightarrow}^Q_{\mathcal{S}} v_n$. De plus, cette schématisation est consistante par rapport à la famille $\mathcal{I}^b_{BC}(C)$.

Preuve: Il est facile de déduire par induction à partir des expressions pour u_{n+1} et v_{n+1} que

$$u_{n+1} \quad \overset{*}{\longleftrightarrow}_Q . \Longrightarrow \quad v_{n+1}$$

Pour $n = 0$ nous avons $u_1 \overset{*}{\longleftrightarrow}_Q . \Longrightarrow v_1$, car $u_1 = t_1\sigma_1\rho_1 \overset{*}{\longleftrightarrow}_Q s_1\sigma_1\rho_1 = (s_1\sigma_1[s_2\sigma_2]_b)\rho_1 \Longrightarrow (s_1\sigma_1[t_2\sigma_2]_b)\rho_1 = v_1$.

Supposons que la relation soit valide pour n. Nous allons la prouver pour $n + 1$. Maintenant $v_{n+1} = s_1\omega_n[v_n\omega_n]_b$ et $u_{n+1} = t_1\omega_n \overset{*}{\longleftrightarrow}_Q s_1\omega_n = s_1\omega_n[u_n\omega_n]_b$. Naturellement, si $u_n \overset{*}{\longleftrightarrow}_Q . \Longrightarrow v_n$, alors $u_{n+1} \overset{*}{\longleftrightarrow}_Q . \Longrightarrow v_{n+1}$.

La schématisation est sans contrainte car le vecteur des variables $\vec{x} = \mathcal{V}ar(s_2)$ est transformé en le vecteur des méta-variables $\dot{\vec{x}}$ avec ses instances principales données par

$$P_\Psi(\dot{\vec{x}}) \quad = \quad \{\vec{x}, \vec{x}\sigma_2\rho_1, \vec{x}\sigma_2\rho_1\omega_1, \ldots, \vec{x}\sigma_2\rho_1\omega_1 \ldots \omega_{n-1}\omega_n, \ldots\}$$

La consistance de la schématisation est prouvée comme dans le cas des systèmes croisés en avant. □

Il faut bien noter que dans cette schématisation des systèmes croisés en arrière, l'ensemble infini de règles

$$\{\dot{g}\dot{\psi}\downarrow_Q \rightarrow \dot{d}\dot{\psi}\downarrow_Q \mid \dot{\psi} \in \Psi, (\dot{g} \rightarrow \dot{d} \parallel \dot{\vec{x}} \in^? P_\Psi(\dot{\vec{x}})) \in MR\}$$

ne coïncide pas précisément avec la famille de règles produites par le processus de complétion divergent, mais il présente la même congruence modulo Q.

Exemple 4.14 (Suite de l'exemple 2.50).
Les variables x' et y' dans la deuxième règle sont devenues des méta-variables $\dot{x}$ et $\dot{y}$. Les instances principales du vecteur des méta-variables $\langle \dot{x}, \dot{y} \rangle$ sont données par

$$P_\Psi(\langle \dot{x}, \dot{y} \rangle) \quad = \quad \{\langle x, g^n(y) \rangle \mid n \geq 0\}$$

La méta-règle est $(\dot{x} \wedge \dot{y}) \vee \dot{y} \to \dot{y} \parallel \langle \dot{x}, \dot{y} \rangle \in^? \{\langle x, g^n(y) \rangle \mid n \geq 0\}$ et le système de réécriture convergent Q est $f(x \vee g(y)) \to f(x) \vee y$.

Exemple 4.15 (Suite de l'exemple 2.51).
Les variables x' et y' dans la deuxième règle sont devenues des méta-variables $\dot{x}$ et $\dot{y}$. Les instances principales du vecteur des méta-variables $\langle \dot{x}, \dot{y} \rangle$ sont données par

$$P_\Psi(\langle \dot{x}, \dot{y} \rangle) \quad = \quad \{\langle x, f^n(y) \rangle \mid n \geq 0\}$$

La méta-règle est $(\dot{x} \oslash \dot{y}) \otimes \dot{y} \to \dot{x} \parallel \langle \dot{x}, \dot{y} \rangle \in^? \{\langle x, f^n(y) \rangle \mid n \geq 0\}$ et le système de réécriture convergent Q est $\{(x \otimes f(y)) \oplus y \to (x \oplus y) \otimes y\}$. La preuve de la relation

$$\begin{array}{l} ((((x \oslash f^{n+1}(y)) \oplus f^n(y)) \oplus \ldots \oplus f(y)) \oplus y) \otimes y \\ \overset{*}{\longleftrightarrow}_Q \quad ((x \oplus f^n(y))) \oplus \ldots \oplus f(y)) \oplus y \end{array}$$

est laissée au lecteur.

Les systèmes à une seule règle

La schématisation d'une famille infinie issue d'un système réduit à une seule règle n'est pas clair.

Exemple 4.16 La famille infinie $\{f(g^n(f(x))) \to g^n(f(x))\}$, produite par le système réduit à une seule règle $f(g(f(x))) \to g(f(x))$, est présentée par la méta-règle

$$f(g(\dot{x})) \to g(\dot{x}) \parallel \dot{x} \in^? P_\Psi(\dot{x})$$

Le domaine de $\dot{x}$ est présenté par les instances de l'ensemble

$$P_\Psi(\dot{x}) \quad = \quad \{f(x), g(f(x)), g(g(f(x))), \ldots\}$$

Le problème est que la congruence de la schématisation $\overset{*}{\longleftrightarrow}_Q$, ainsi que la méta-règle, sont produites à partir du même objet. Dans le cas d'un système divergent à une seule règle, une schématisation sans contrainte ne peut pas être construite. Il faut alors utiliser toute la puissance du formalisme des méta-règles avec les contraintes.

Proposition 4.17 *Supposons que $\mathcal{I}^{a,b}_{FC}(C) = \{u_n \to v_n\}$ forme une famille infinie itérée en avant produite à partir du système croisé en avant $C = \{s \to t, s \mapsto t\}$ comme dans la définition 2.31. Supposons en plus que $s|_a = t|_b$ et $\sigma_2 = \varphi_2 = [\,]$. Considérons la schématisation $\mathcal{S} = (MR, \Psi, Q)$ suivante:*

- *la méta-règle est $s[\dot{x}]_a \to t[\dot{x}]_b \parallel \dot{x} \in^? P_\Psi(\dot{x})$ où $\dot{x}$ est une méta-variable avec son ensemble d'instances principales $P_\Psi(\dot{x}) = \{t|_b, t, c[t], \ldots, c^n[t], \ldots\}$, où $c[\cdot]$ est le contexte $t[\cdot]_b$, et $MR = \{s[\dot{x}]_a \to t[\dot{x}]_b \parallel \dot{x} \in^? P_\Psi(\dot{x})\}$;*

– $Q = \emptyset$ *(et, par conséquent, la congruence de schématisation* $\overset{*}{\longleftrightarrow}_Q$ *est l'égalité syntaxique).*

Alors pour chaque n, $u_n \overset{*}{\Longleftrightarrow}_{\mathcal{S}}^{Q} v_n$. *De plus, cette schématisation est consistante par rapport à la famille* $\mathcal{I}_{FC}^{a,b}(C)$.

Preuve: A partir des hypothèses $s|_a = t|_b$, $\sigma_2 = \varphi_2 = [\,]$, on déduit que pour chaque n, l'unificateur ω_n de $u_n|_{ab^n}$ et s est constant. De plus, $\mathcal{D}om(\omega_n) \cap \mathcal{V}ar(s) = \emptyset$ et $\mathcal{VR}an(\omega_n) \subseteq \mathcal{V}ar(s)$. Donc $s\omega_n = s$ et $t\omega_n = t$.

Nous prouvons par récurrence sur n les propriétés suivantes:

$$\begin{array}{lcl} u_n & = & s[c^n[t|_b]]_a \\ v_n & \overset{*}{\longrightarrow}_{\{s\to t\}} & t[c^n[t|_b]]_b \end{array}$$

Pour $n = 0$ on obtient $u_0 = s[t|_b]_a = s[s|_a]_a = s$ et $v_0 = t[t|_b]_b = t$.

Par construction, car $u_n|_{ab^n}\omega_n = s\omega_n = s$, on obtient $u_{n+1} = u_n\omega_n[t]_{ab^n}$. Par hypothèse, on obtient $u_{n+1} = s[c^n[t|_b]]_a\omega_n[t]_{ab^n} = s[c^n[t]]_a = s[c^{n+1}[t|_b]]_a$.

Par construction encore une fois on obtient $v_{n+1} = v_n\omega_n$. En utilisant la deuxième hypothèse de récurrence, on obtient $v_{n+1} \overset{*}{\longrightarrow}_{\{s\to t\}} t[c^n[t|_b]]_b\omega_n = t[c^n[t|_b\omega_n]]_b$. Par application de la première hypothèse de récurrence, on obtient maintenant $t|_b\omega_n = u_n|_{ab^n}\omega_n = s\omega_n = s \to t$. Alors, $v_{n+1} \overset{*}{\longrightarrow}_{\{s\to t\}} t[c^{n+1}[t|_b]]_b$.

En choisissant l'instance principale $\dot\tau = [\dot x \mapsto t|_b]$, la schématisation de $\mathcal{I}_{FC}^{a,b}(C)$ satisfait la proposition suivante:

> Pour chaque instance principale $\dot\psi$ il existe un entier j et un contexte $c[\cdot]$ tels que $\dot x\dot\psi = c^j[\dot x\dot\tau]$.

La schématisation est consistante car $s[\dot x]_a\dot\tau = s[t|_b]_a = s[s|_a]_a = s \overset{*}{\longleftrightarrow}_{R_\infty} t = t[t|_b]_b = t[\dot x]_b\dot\tau$.

A partir de $s[\dot x]_a\dot\tau = s$ et $t[\dot x]_b\dot\tau = t$ on déduit $s \Longrightarrow t$. Finalement $u_n \overset{*}{\Longleftrightarrow} v_n$, car le choix de $\dot\psi_n = [\dot x \mapsto c^n[t|_b]]$ comme l'instance principale de $\dot x$ implique $u_n = s[\dot x]_a\dot\psi_n \Longrightarrow t[\dot x]_b\dot\psi_n \overset{*}{\Longleftrightarrow} v_n$ ($\overset{*}{\Longleftrightarrow}$ est équivalent à $\overset{*}{\Longleftrightarrow}^{Q}$ parce que Q est vide). □

Un autre problème apparaît quand la schématisation introduit des méta-variables corrélées entre elles. Ceci est illustré par un exemple simple d'un système croisé en arrière consistant en une seule règle.

Exemple 4.18 La complétion du système croisé en arrière d'une seule règle

$$f(g(f(x))) \quad \to \quad f(h(x))$$

aboutit à la famille infinie $f(h^n(g(f(x)))) \to f(h^{n+1}(x))$. Les parties gauches de règles de cette famille sont schématisées par le méta-terme $f(\dot x)$, alors que les parties droites sont schématisées par le méta-terme $f(h(\dot y))$. Les instances principales du vecteur des méta-variables $\langle \dot x, \dot y\rangle$ sont données par l'ensemble

$$P_\Psi(\langle \dot x, \dot y\rangle) \quad = \quad \{\langle g(f(x)), x\rangle, \langle h(g(f(x))), h(x)\rangle, \ldots, \langle h^n(g(f(x))), h^n(x)\rangle, \ldots\}$$

La schématisation est donnée par une règle contrainte

$$f(\dot x) \quad \to \quad f(h(\dot y)) \quad \| \quad \langle \dot x, \dot y\rangle \in^? P_\Psi(\langle \dot x, \dot y\rangle)$$

Les conditions suffisantes qui mènent à ce type de schématisations contraintes sont exprimées dans la proposition suivante.

Proposition 4.19 *Soit $\mathcal{I}^b_{BC}(C) = \{u_n \to v_n\}$ une famille infinie itérée en arrière, construite à partir du système croisé en arrière $C = \{s \to t, s \rightarrowtail t\}$ comme dans la définition 2.43. Supposons de plus que $\sigma_2 = \varphi_1 = [\,]$ et $\mathcal{V}ar(t) \cap \mathcal{D}om(\sigma_1) = \{x\}$. Cette famille de règles possède la schématisation suivante:*

- *la méta-règle est $\dot{g} \to \dot{d} \parallel \langle \dot{x}, \dot{y} \rangle \in^? P_\Psi(\langle \dot{x}, \dot{y} \rangle)$, où*
  - *$\dot{g}$ est obtenu à partir de $s|_b$ par la transformation de $x \in \mathcal{D}om(\sigma_1)$ en une méta-variable $\dot{x}$,*
  - *$\dot{d}$ est obtenu à partir de t par la transformation de $x \in \mathcal{V}ar(t)$ en une méta-variable $\dot{y}$,*
  - *$\langle \dot{x}, \dot{y} \rangle$ est un vecteur de méta-variables dont l'ensemble des instances principales est $P_\Psi(\langle \dot{x}, \dot{y} \rangle) = \{\langle x\sigma_1, x \rangle, \langle x\varphi_2\sigma_1, x\varphi_2 \rangle, \ldots, \langle x\varphi_2^i\sigma_1, x\varphi_2^i \rangle, \ldots\}$*
- *$Q = \emptyset$.*

Alors pour chaque n, $u_n \overset{*}{\underset{\mathcal{S}}{\overset{Q}{\Longleftrightarrow}}} v_n$. De plus, cette schématisation est consistante par rapport à la famille $\mathcal{I}^b_{BC}(C)$.

Preuve: Pour chaque n, l'unificateur ω_n de u_{n+1} et $s|_b$ satisfait $\mathcal{D}om(\omega_n) \cap \mathcal{V}ar(u_n) = \emptyset$ et $\mathcal{D}om(\omega_n) \cap \mathcal{V}ar(v_n) = \emptyset$. Alors $u_n\omega_n = u_n$ et $v_n\omega_n = v_n$. De plus, $\omega_{n+1} = \varphi_2^n\sigma_1$ pour chaque n.

Nous prouvons par récurrence sur n les propriétés suivantes:

$$\begin{array}{lcl} u_{n+1} & = & s|_b\varphi_2^n\sigma_1 \\ v_{n+1} & \overset{*}{\longrightarrow}_{\{s\to t\}} & t\varphi_2^n \end{array}$$

Pour $n = 0$, nous avons $u_1 = s|_b\sigma_1 = s$, $\omega_1 = \sigma_1$ et $v_1 = t$.

Par construction, nous avons $u_{n+1} = t\omega_n$. Par hypothèse de $\omega_n = \varphi_2^{n-1}$ et $t = s|_b\varphi_2$, nous obtenons $u_{n+1} = t\varphi_2^{n-1}\sigma_1 = s|_b\varphi_2\varphi_2^{n-1}\sigma_1 = s|_b\varphi_2^n\sigma_1$.

Par construction aussi nous avons $v_{n+1} = s\omega_n[v_n\omega_n]_b = s[v_n]_b$. Par hypothèse, nous obtenons $v_{n+1} \overset{*}{\longrightarrow}_{\{s\to t\}} s[t\varphi_2^{n-1}]_b = s[s|_b\varphi_2^n]_b = s\varphi_2^n \to_{\{s\to t\}} t\varphi_2^n$.

Définissons maintenant $\dot{\psi}_n = [\dot{x} \mapsto x\varphi_2^n\sigma_1, \dot{y} \mapsto x\varphi_2^n]$ et nous obtenons $\dot{g}\dot{\psi}_n = u_{n+1} \overset{*}{\longleftrightarrow}_{R_\infty} v_{n+1} \overset{*}{\longleftrightarrow}_{R_\infty} \dot{d}\dot{\psi}_n$. Donc, la schématisation est consistante.

Parce que $\dot{g}\dot{\psi}_0 = s$ et $\dot{d}\dot{\psi}_0 = t$, nous avons $s \Longrightarrow t$. Finalement $u_n \overset{*}{\Longleftrightarrow} v_n$, car $u_n = \dot{g}\dot{\psi}_n \Longrightarrow \dot{d}\dot{\psi}_n \overset{*}{\Longleftrightarrow} v_n$. □

Une classe des familles itérées

On considère un exemple plus complexe pour illustrer le cas général où les familles itérées différentes sont présentes en même temps. Chaque système de réécriture divergent R est divisible en deux parties disjointes: le sous-système R_{diverg} dont toutes les règles participent au processus divergent; et le sous-système $R_{fix} = R - R_{diverg}$. Il est assez clair, selon les considérations précédentes, que les règles de R_{fix} sont transformées en méta-règles simplement par la transformation de chaque variable standard x en la méta-variable $\dot{x}$ avec la valeur x comme ensemble des instances principales. Pour cette raison, nous allons toujours supposer que $R = R_{diverg}$ sans perte de généralité.

Clairement, un système divergent peut contenir plusieur familles itérées de règles.

Exemple 4.20 (Suite de l'exemple 2.40).
La procédure de complétion produit deux familles infinies de règles:

$$F_1 = \left\{ \begin{array}{rcl} i(x_1) * (x_1 * y) & \to & y \\ (i(x_2) * i(x_1)) * ((x_1 * x_2) * y) & \to & y \\ (i(x_2) * x_1) * ((i(x_1) * x_2) * y) & \to & y \\ (x_2 * i(x_1)) * ((x_1 * i(x_2)) * y) & \to & y \\ \multicolumn{3}{c}{\ldots} \end{array} \right\}$$

$$F_2 = \left\{ \begin{array}{rcl} (y * x_1) * i(x_1) & \to & y \\ (y * (x_1 * x_2)) * (i(x_2) * i(x_1)) & \to & y \\ (y * (i(x_1) * x_2)) * (i(x_2) * x_1) & \to & y \\ (y * (x_1 * i(x_2))) * (x_2 * i(x_1)) & \to & y \\ \multicolumn{3}{c}{\ldots} \end{array} \right\}$$

Le sous-ensemble de règles

$$Q = \left\{ \begin{array}{rcl} i(i(x)) & \to & x \\ i(x * y) & \to & i(y) * i(x) \end{array} \right\}$$

est confluent et nœthérien. La suite de règles $(l_n \to r_n)$ de la première famille peut se généraliser modulo Q par $i(\dot{x}) * (\dot{x} * \dot{y}) \to \dot{y}$, où les instances principales du vecteur des méta-variables $\langle \dot{x}, \dot{y} \rangle$ est donné par l'ensemble

$$\{\langle x_1, y\rangle, \langle x_1 * x_2, y\rangle, \langle i(x_1) * x_2, y\rangle, \langle x_1 * i(x_2), y\rangle, \langle i(x_1) * i(x_2), y\rangle, \langle x_1 * (x_2 * x_3), y\rangle, \ldots\}$$

Les méta-règles associées respectivement à chaque famille de règles sont

$$\begin{array}{rcl} i(\dot{x}) * (\dot{x} * \dot{y}) & \to & \dot{y} \\ (\dot{y} * \dot{x}) * i(\dot{x}) & \to & \dot{y} \end{array}$$

L'exemple précédent contient deux familles infinies de règles, mais un nombre infini de familles itérées dû au fait que l'ensemble de fermetures de superposition $OC(Q)$ contient un ensemble infini de chaînes en avant indépendantes.

Nous allons introduire des définitions pour pouvoir classifier les règles dans les familles avec un ensemble générateur de systèmes croisés saturés. Notons $\mho(S)$ la *fermeture de superposition* et par $\aleph(S)$ la *règle de départ* du système croisé S. Notons par suite $\aleph(\mathcal{C}) = \bigcup_{S\in\mathcal{C}} \aleph(S)$ ses extensions à une classe $\mathcal{C}$ de systèmes croisés l'ensemble de toutes les règles de départ, et $\mho(\mathcal{C}) = \bigcup_{S\in\mathcal{C}} \mho(S)$ l'ensemble de toutes les fermetures de superposition présentes dans la classe $\mathcal{C}$. L'*ensemble de règles de départ de tous les systèmes croisés saturés* produits pendant la complétion du système de réécriture R est alors, avec cette notation, $\aleph(\overline{\coprod R})$.

Exemple 4.21 (Suite de l'exemple 2.40).
L'ensemble de toutes les règles de départ $\aleph(\overline{\coprod R})$ est

$$\{(y * x) * i(x) \to y, i(x) * (x * y) \to y\}$$

Deux règles appartiennent à la même famille si elles sont produites à partir de la même règle de départ p dans le système croisé et saturé S. Chaque famille est associée à un ensemble des systèmes croisés et saturés produits à partir de la même règle de départ p.

Définition 4.22 *Notons*

$$\mathcal{G}en_p(R) \quad = \quad \{S \in \overline{\coprod R} \mid \aleph(S) = p\}$$

le **générateur fondé sur** $\boldsymbol{p}$ *dans* R_∞, *et*

$$\mathcal{F}am_p(R) \quad = \quad \{p\} \cup \{q \in \mathcal{I}(S) \mid S \in \mathcal{G}en_p(R)\}$$

la **famille** *(infinie) de règles* **fondée sur** $\boldsymbol{p}$ *dans* R_∞. *Notons encore*

$$\mathcal{G}en_*(R) \quad = \quad \{\mathcal{G}en_p(R) \mid p \in \aleph(\overline{\coprod R})\}$$

la classe de **tous les générateurs** *dans* R_∞, *et*

$$\mathcal{F}am_*(R) \quad = \quad \{\mathcal{F}am_p(R) \mid p \in \aleph(\overline{\coprod R})\}$$

la classe de **toutes les familles infinies** *de règles dans* R_∞.

Exemple 4.23 (Suite de l'exemple 2.40).
La classe $\mathcal{F}am_*(R)$ contient deux familles: $\mathcal{F}am_p(R) = F_1$ où $p = i(x) * (x * y)) \to y$ et $\mathcal{F}am_q(R) = F_2$ où $q = (y * x) * i(x) \to y$, selon l'ensemble $\aleph(\overline{\coprod R})$ qui contient les deux règles p et q.

La classe $\mathcal{G}en_*(R)$ contient deux générateurs, $\mathcal{G}en_p(R)$ et $\mathcal{G}en_q(R)$, qui sont tous deux infinis. Le générateur $\mathcal{G}en_p(R)$ contient par exemple le système croisé

$$\begin{array}{rcl} i(x) * (x * y) & \to & y \\ i(i(x) * y) & \rightarrowtail & i(y) * x \end{array}$$

Dans l'exemple 2.40, un ensemble infini de systèmes croisés et saturés correspond à chaque famille. Le problème posé par cet ensemble infini est la description de la congruence de schématisation $\overset{*}{\longleftrightarrow}_{\mathfrak{V}(\mathcal{B})}$ pour chaque générateur $\mathcal{B} \in \mathcal{G}en_*(R)$ en utilisant un ensemble fini d'axiomes. La difficulté est résolue grâce à l'existence du système de réécriture fini et convergent Q.

Exemple 4.24 (Suite de l'exemple 2.40).
La relation d'équivalence produite par le système convergent Q contient la relation d'équivalence produite par $\mathfrak{V}(\overline{\coprod R})$, ainsi que par $\mathfrak{V}(\mathcal{G}en_p(R))$ et $\mathfrak{V}(\mathcal{G}en_q(R))$.

Notons que Q contient la règle de réécriture $i(i(x)) \to x$ qui n'appartient à aucun de ces ensembles de fermetures.

Théorème 4.25 *Soit R un système de réécriture divergent. S'il existe un système de réécriture fini et convergent Q tel que pour chaque générateur $\mathcal{B} \in \mathcal{G}en_*(R)$ la congruence engendrée par $\mathfrak{V}(\mathcal{B})$ est incluse dans $\overset{*}{\longleftrightarrow}_Q$ et son complémentaire par rapport à Q est inclus dans $\overset{*}{\longleftrightarrow}_R$, alors R_∞ possède la schématisation sans contrainte $\mathcal{S} = (MR, \Psi, Q)$, où l'ensemble de méta-règles MR est obtenu à partir de $\aleph(\overline{\coprod R})$ par la transformation de chaque variable x en une méta-variable $\dot{x}$.*

Cette schématisation est consistante par rapport à R_∞. De plus, si MR est convergent modulo Q, la schématisation est complète par rapport à R_∞.

Preuve: Il est clair que la schématisation est consistante car les méta-règles MR sont des règles du système de réécriture divergent avec leurs variables transformées en méta-variables.

Pour la complétude, il suffit de prouver que $\overset{*}{\longleftrightarrow}_{R_\infty}$ implique $\overset{*}{\Longleftrightarrow}{}^{Q}_{\mathcal{S}}$ quand MR est convergent modulo Q, d'après la proposition 4.10. Si $t \longrightarrow_{R_\infty} t'$, le pas de réécriture est effectué soit par une règle de $\mathfrak{V}(\mathcal{B})$, soit par une règle de $Q - \mathfrak{V}(\mathcal{B})$, soit par une règle de $R_\infty - (Q \cup \mathfrak{V}(\mathcal{B}))$, où $\mathcal{B}$ est un générateur de $\mathcal{G}en_*(R)$.

Si le pas de réécriture est effectué par une règle de $\mathfrak{V}(\mathcal{B})$ alors $t \overset{*}{\longleftrightarrow}_Q t'$, puisque $\overset{*}{\longleftrightarrow}_{\mathfrak{V}(\mathcal{B})} \subseteq \overset{*}{\longleftrightarrow}_Q$.

Si le pas de réécriture est effectué par une règle de $Q - \mathfrak{V}(\mathcal{B})$, cette règle appartient à Q et $t \overset{*}{\longleftrightarrow}_Q t'$.

Si le pas de réécriture est effectué par une règle $(l \to r) \in R_\infty - (Q \cup \mathfrak{V}(\mathcal{B}))$, alors la règle $l \to r$ appartient à la famille $\mathcal{F}am_p(R) \in \mathcal{F}am_*(R)$ fondée sur p, pour un $p \in \aleph(\overline{\coprod R})$. Ceci signifie que $l \to r$ est soit une règle d'une famille itérée $\mathcal{I}(C)$ soit c'est la règle $p = \aleph(C)$, pour un système croisé $C \in \overline{\coprod R}$. D'après les propositions 4.11 et 4.13, il existe une règle $(g \to d) = \aleph(C)$, transformée (par renommage des variables) en une méta-règle $(\dot{g} \to \dot{d} \parallel \vec{x} \in^? P_\Psi(\vec{x})) \in MR$, et

$$l \quad \overset{*}{\Longleftrightarrow}{}^{Q}_{\mathcal{S}} \quad r$$

Ceci prouve que $\longrightarrow_{R_\infty}$ implique $\overset{*}{\Longleftrightarrow}{}^{Q}_{\mathcal{S}}$.

En appliquant l'induction sur la longueur de $\overset{*}{\longleftrightarrow}_{R_\infty}$ nous prouvons immédiatement que $\overset{*}{\longleftrightarrow}_{R_\infty}$ implique $\overset{*}{\Longleftrightarrow}{}^{Q}_{\mathcal{S}}$. □

Notons que quand chaque générateur $\mathcal{B} \in \mathcal{G}en_*(R)$ est fini, Q peut être choisi comme $\mathfrak{V}(\mathcal{B})$, comme dans le cas des familles itérées uniques, sous condition que $\mathfrak{V}(\mathcal{B})$ soit convergent.

Bien sûr, pour obtenir une schématisation raisonnable, l'ensemble de toutes les règles de départ de tous les systèmes croisés $\aleph(\overline{\coprod R})$ doit être fini.

4.2 Schémas de termes

Les schémas de termes (*term schemes* en anglais) ont été dévélopés par Bernhard Gramlich [Gra88]. Il n'est pas acquis que le problème d'unification des schémas de termes soit décidable, parce que cette schématisation est fondée sur les termes avec des variables de second ordre.

Bernhard Gramlich a proposé en 1988 une méthode pour résoudre certains ensembles infinis de problèmes d'unification du premier ordre, présentés par des schémas de termes. Dans le cadre de la logique équationnelle du second ordre, la solution de tels problèmes d'unification de schémas se réduit exactement à la solution des problèmes d'unification avec des contraintes sur les variables. Il a donc généralisé au second ordre une méthode connue pour la solution de problèmes d'unification du premier ordre avec contraintes sur les variables. Essentiellement, il propose la transformation d'un problème d'unification contrainte à un problème sans contrainte, suivi par la solution de celui-ci et la retransformation de la solution ainsi obtenue. Les résultats sur l'unification contrainte du second ordre sont alors utilisés pour résoudre le problème original, plus précisément pour décider l'existence des solutions des problèmes d'unification des schémas et – dans le cas positif – pour calculer les unificateurs principaux correspondants. Finalement, il déduit des conditions suffisantes pour la propriété d'*unifiabilité répétitive* qui est primordiale pour l'analyse de la divergence de la complétion. Alors que la divergence de la complétion était le point de départ des travaux sur les schémas de termes, les résultats obtenus sont applicables en dehors de l'analyse stricte de la divergence et ont bien d'autres applications.

4.2.1 Syntaxe

Définition 4.26 *Soit $\mathcal{V}_n$, pour chaque $n \in \mathcal{N}$, l'ensemble dénombrable des* variables *d'arité n et $\mathcal{V} = \bigcup_i \mathcal{V}_i$ la signature de toutes les variables. Notamment, $\mathcal{V}_0 = \mathcal{X}$. Soit $\mathcal{K}$ la signature des* constantes *stratifiée par leur arité, où $\mathcal{K} \cap \mathcal{V} = \emptyset$.*

*L'***algèbre de termes de second ordre restreint** *$\mathcal{T}(\mathcal{K}, \mathcal{V})$ est le plus petit ensemble tel que si $A \in \mathcal{V} \cup \mathcal{K}$, $ar(A) = n$ et $\vec{t} \in \mathcal{T}(\mathcal{K}, \mathcal{V})^n$ alors $A(\vec{t}) \in \mathcal{T}(\mathcal{K}, \mathcal{V})$.*

Les variables $\mathcal{V}ar(t)$ d'un terme $t \in \mathcal{T}(\mathcal{K}, \mathcal{V})$ sont définies par

$$\mathcal{V}ar(A(\vec{t})) \quad = \quad (\{A\} \cap \mathcal{V}) \cup \bigcup_{i=1}^{ar(A)} \mathcal{V}ar(t_i)$$

Pour obtenir l'algèbre des termes de second ordre sans restriction, qui contient aussi des objets fonctionnels, il faut compléter $\mathcal{T}(\mathcal{K}, \mathcal{V})$ en $\overline{\mathcal{T}}(\mathcal{K}, \mathcal{V})$ par introduction des variables liées.

Définition 4.27 *L'algèbre $\overline{\mathcal{T}}(\mathcal{K}, \mathcal{V})$ est le plus petit ensemble tel que*

– $\mathcal{T}(\mathcal{K}, \mathcal{V}) \subset \overline{\mathcal{T}}(\mathcal{K}, \mathcal{V})$

– *si* $t \in \overline{\mathcal{T}}(\mathcal{K},\mathcal{V})$ *et* $x \in \mathcal{V}_0$ *alors* $\lambda x.t \in \overline{\mathcal{T}}(\mathcal{K},\mathcal{V})$

Les variables $\mathcal{V}ar(t)$ d'un terme $t \in \overline{\mathcal{T}}(\mathcal{K},\mathcal{V})$ sont définies par

$$\begin{aligned}\mathcal{V}ar(A(\vec{t})) &= (\{A\} \cap \mathcal{V}) \cup \bigcup_{i=1}^{ar(A)} \mathcal{V}ar(t_i)\\ \mathcal{V}ar(\lambda x.t) &= \mathcal{V}ar(t) - \{x\}\end{aligned}$$

Comme d'habitude dans le λ-calcul, $\lambda x_1 \ldots x_n.t$ est un raccourci pour $\lambda x_1 \ldots \lambda x_n.t$ et $\eta[A] = \lambda\vec{x}.A(\vec{x})$ note la forme normale de A par rapport à la η-conversion.

Une **substitution de second ordre** est une application $\sigma\colon \mathcal{V} \longrightarrow \overline{\mathcal{T}}(\mathcal{K},\mathcal{V})$ notée par un ensemble *fini* $[X_1 \mapsto t_1, \ldots, X_n \mapsto t_n]$ tel que $t_1 \neq \eta[X_i]$ pour chaque $i = 1, \ldots, n$. Le *domaine* d'une substitution σ est l'ensemble

$$\mathcal{D}om(\sigma) = \{X \in \mathcal{V} \mid X\sigma \neq \eta[X]\}$$

Les *variables introduites* par σ sont notées

$$\mathcal{VR}an(\sigma) = \{\mathcal{V}ar(t) \mid (X \mapsto t) \in \sigma\}$$

L'application d'une substitution σ aux termes $\overline{\mathcal{T}}(\mathcal{K},\mathcal{V})$ est définie récursivement par

$$\begin{array}{rcll} A(t_1,\ldots,t_n)\sigma &=& A(t_1\sigma,\ldots,t_n\sigma) & \text{pour } A \in \mathcal{K} \cup (\mathcal{V} - \mathcal{D}om(\sigma))\\ X(t_1,\ldots,t_n)\sigma &=& t\phi & \text{pour } X \in \mathcal{V} \cap \mathcal{D}om(\sigma) \text{ où } (X \mapsto \lambda\vec{x}.t) \in \sigma\\ & & & \text{et } \phi = [x_i \mapsto t_i\sigma \mid x_i \in \vec{x}]\\ (\lambda x.t)\sigma &=& \lambda x.(t\sigma) & \text{pour } x \notin \mathcal{D}om(\sigma)\end{array}$$

La *restriction* d'une substitution σ à l'ensemble (fini) des variables $\vec{V}$ est

$$\sigma|_{\vec{V}} = \{(X \mapsto t) \in \sigma \mid X \in \vec{V} \cap \mathcal{D}om(\sigma)\}$$

Une substitution σ est dite *stricte* si pour chaque variable $X \in \mathcal{D}om(\sigma)$ nous avons $X\sigma = \lambda\vec{x}.t$ où $\vec{x} \subseteq \mathcal{V}ar(t)$.

Définition 4.28 *Un* **schéma de termes** *est un terme de second ordre restreint* $t \in \mathcal{T}(\mathcal{K},\mathcal{V})$ *avec deux sortes de variables: des* variables ordinaires *du premier ordre et des* variables de schémas *du second ordre. Une* équation de schémas *est une paire* $s \simeq t$ *de schémas de termes.*

Un **problème d'unification des schémas** *est présenté par un ensemble* $E = \{s_i \stackrel{?}{=} t_i \mid i = 1,\ldots,n\}$ *des équations de schémas noté* $E \dashv \mathcal{W}$*, où* $\mathcal{W}$ *est l'ensemble de toutes les variables ordinaires de* E *et* $\mathcal{W}^c = \mathcal{V}ar(E) - \mathcal{W}$ *est l'ensemble des variables de schémas de* E*.*

Une *solution* du problème d'unification des schémas $\{s_i \stackrel{?}{=} t_i \mid i = 1,\ldots,n\} \dashv \mathcal{W}$ est équivalente à une solution du problème d'unification du premier ordre $\{s_i\psi \stackrel{?}{=} t_i\psi \mid i =$

$1, \ldots, n\}$ où ψ est une substitution de second ordre telle que $\mathcal{D}om(\psi) \subseteq \mathcal{W}^c$ et $\mathcal{VR}an(\psi) \subseteq \mathcal{V}_0 - \mathcal{W}$.

Définition 4.29 *Un* **problème d'unification restreint** *aux variables $\mathcal{W}$ est un ensemble fini E des formules d'unification noté*

$$E \dashv \mathcal{W} \quad = \quad \{s_i \stackrel{?}{=} t_i \mid i = 1, \ldots, n\} \dashv \mathcal{W}$$

où $\mathcal{W} \subseteq \mathcal{V}ar(E)$.

Une *solution* d'un problème d'unification restreint $E \dashv \mathcal{W}$ est une substitution σ telle que $\mathcal{D}om(\sigma) \cap \mathcal{V}ar(E) \subseteq \mathcal{W}$ et $s_i\sigma = t_i\sigma$ pour chaque $(s_i \stackrel{?}{=} t_i) \in E$. L'*ensemble de* toutes les *solutions* de $E \dashv \mathcal{W}$ est noté $\mathcal{U}(E \dashv \mathcal{W})$. Car $\sigma \in \mathcal{U}(E \dashv \mathcal{W})$ si et seulement si $\sigma|_{\mathcal{W}} \in \mathcal{U}(E \dashv \mathcal{W})$, notre intérêt est focalisé sur l'ensemble

$$\mathcal{U}_{\mathcal{W}}(E \dashv \mathcal{W}) \quad = \quad \{\sigma \in \mathcal{U}(E \dashv \mathcal{W}) \mid \mathcal{D}om(\sigma) \subseteq \mathcal{W}\}$$

Le problème $E \dashv \mathcal{W}$ a une solution si $\mathcal{U}(E \dashv \mathcal{W}) \neq \emptyset$.

Par le choix $\mathcal{W} = \mathcal{V}ar(E)$ nous obtenons le problème d'unification habituel de termes de second ordre. Intuitivement, $\mathcal{W}$ est l'ensemble des variables sur lesquelles la substitution est active, tandis que $\mathcal{W}^c = \mathcal{V}ar(E) - \mathcal{W}$ est l'ensemble des *variables interdites* sur lesquelles la substitution est inactive.

Définition 4.30 *Un problème d'unification restreint $E \dashv \mathcal{W}$ est en* **forme résolue** *si pour chaque $(s_i \stackrel{?}{=} t_i) \in E$, s_i est égale à $\eta[X]$ où $X \in \mathcal{W}$ et X n'apparaît nulle part ailleurs dans E.*

Pour chaque problème d'unification restreint $E \dashv \mathcal{W}$ en forme résolue, on définit la *substitution réalisante*

$$\sigma_E \quad = \quad \{X \mapsto t \mid (\eta[X] \stackrel{?}{=} t) \in E\}$$

et pour chaque substitution de second ordre σ on définie la forme résolue sous-jacente

$$E_\sigma \quad = \quad \{\eta[X] \stackrel{?}{=} t \mid (X \mapsto t) \in \sigma\}$$

4.2.2 Sémantique

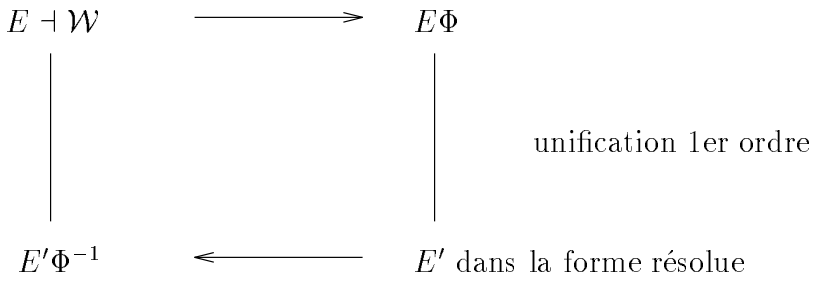
Nous montrons que chaque problème d'unification restreint $E \dashv \mathcal{W}$ peut être ramené en un problème d'unification ordinaire E' tel que les solutions de $E \dashv \mathcal{W}$ peuvent être obtenues à partir des solutions de E' par la transformation inverse. La fonction de transformation interprète les variables interdites $\mathcal{W}^c$ en des nouvelles constantes d'arité correspondante. Plus précisemment, pour chaque $X_i \in \mathcal{W}^c = \vec{X}$ soit K_i une nouvelle constante d'arité $ar(X_i) = ar(K_i)$. Soit $\mathcal{K}^* = \mathcal{K} \uplus \vec{K}$ et $\mathcal{V}^* = \mathcal{V} - \mathcal{W}^c$, les algèbres de termes transformés corresponantes sont $\mathcal{T}(\mathcal{K}^*, \mathcal{V}^*)$ et $\overline{\mathcal{T}}(\mathcal{K}^*, \mathcal{V}^*)$ respectivement. La fonction de transformation $\Phi{:}\mathcal{T}(\mathcal{K}^*, \mathcal{V}^*) \longrightarrow \overline{\mathcal{T}}(\mathcal{K}^*, \mathcal{V}^*)$ est définie comme l'extension homomorphique de $\Phi{:}\mathcal{V} \longrightarrow$

$\overline{\mathcal{T}}(\mathcal{K}^*, \mathcal{V}^*)$ où $X_i\Phi = \eta[K_i]$ pour chaque $X_i \in \mathcal{W}^c$ et $X\Phi = \eta[X]$ si $X \notin \mathcal{W}^c$. Elle peut aussi être étendue aux substitutions qui ne touchent pas les variables $\mathcal{W}^c$.

Lemme 4.31 *Soit* $E = \{s_i \stackrel{?}{=} t_i \mid i = 1, \ldots, n\}$ *et* $E\Phi = \{s_i\Phi \stackrel{?}{=} t_i\Phi \mid i = 1, \ldots, n\}$. *La substitution* σ *est l'unificateur principal du problème d'unification restreint* $E \dashv \mathcal{W}$ *si et seulement si* $\sigma \triangle \Phi$ *est l'unificateur principal du problème* $E\Phi$.

Nous sommes intéressés par les problèmes d'unification restreints dont les seules variables interdites sont, du second ordre. Dans ce cas, le problème transformé est du premier ordre. Résoudre un problème d'unification non-restreint E, avec $E \subseteq \mathcal{T}(\mathcal{K}, \mathcal{V}_0) \times \mathcal{T}(\mathcal{K}, \mathcal{V}_0)$, dans $\overline{\mathcal{T}}(\mathcal{K}, \mathcal{V})$ est essentiellement la même chose que de résoudre E dans $\mathcal{T}(\mathcal{K}, \mathcal{V}_0)$ comme un problème d'unification du premier ordre.

En utilisant le lemme 4.31, nous déduisons que chaque problème d'unification restreint $E \dashv \mathcal{W}$, avec $E \subseteq \mathcal{T}(\mathcal{K}, \mathcal{V}) \times \mathcal{T}(\mathcal{K}, \mathcal{V})$ et $\mathcal{W} \subseteq \mathcal{V}_0$, soit n'a pas de solution, soit possède un unificateur principal (unique modulo la relation de $\equiv_{\mathcal{V}ar(E)}$-équivalence) et il peut être calculé par l'algorithme classique d'unification du premier ordre.



Les transformations explicites Φ et Φ^{-1} peuvent être économisées par la skolémisation des variables interdites $\mathcal{W}^c$ pendant le processus d'unification du premier ordre.

Si le problème d'unification restreint $E \dashv \mathcal{W}$, avec $E \subseteq \mathcal{T}(\mathcal{K}, \mathcal{V}) \times \mathcal{T}(\mathcal{K}, \mathcal{V})$ et $\mathcal{W} \subseteq \mathcal{V}_0$, possède une solution, l'unificateur principal σ, alors le problème instancié $E\psi \dashv \mathcal{W}$ possède une solution aussi, pour chaque substitution ψ qui ne touche pas les variables $\mathcal{W}^c$ et qui n'introduit aucune nouvelle variable. De plus, si ψ est stricte, l'unificateur principal peut être calculé à partir de σ et ψ sans avoir unifié à nouveau explicitement.

Théorème 4.32 *Soit* $E \dashv \mathcal{W}$, *avec* $E \subseteq T(K, V) \times T(K, V)$ *et* $\mathcal{W} \subseteq \mathcal{V}_0$, *un problème d'unification restreint et soit* σ *son unificateur principal. Supposons de plus que* ψ *est une substitution avec* $\mathcal{D}om(\psi) \subseteq \mathcal{V}ar(E) - \mathcal{W}$ *et* $\mathcal{VR}an(\psi) \cap \mathcal{W} = \emptyset$. *Alors* $\sigma \triangle \psi$ *est l'unificateur du problème restreint* $E\psi \dashv \mathcal{W}$. *Cet unificateur est principal si* ψ *est stricte.*

Revenons maintenant aux problèmes d'unification des schémas. Il faut décider pour $E \dashv \mathcal{W}$, avec $\mathcal{W} \subseteq \mathcal{V}_0$, si tous les problèmes d'unification du premier ordre sans restriction $E\psi$, avec $\mathcal{D}om(\psi) \subseteq \mathcal{V}ar(E) - \mathcal{W} \cap \mathcal{V}_0$ et $\mathcal{VR}an(\psi) \cap \mathcal{W} = \emptyset$, possèdent une solution et dans le cas positif calculer les unificateurs principaux.

Théorème 4.33 *Soit* $E \subseteq \mathcal{T}(\mathcal{K}, \mathcal{V}) \times \mathcal{T}(\mathcal{K}, \mathcal{V})$ *avec* $\mathcal{W} \subseteq \mathcal{V}_0$ *un système donné. Supposons de plus que la signature contient au moins une constante fonctionnelle d'arité* ≥ 1 *et une*

autre d'arité ≥ 2. *Le problème d'unification restreint* $E \dashv \mathcal{W}$ *possède une solution si et seulement si le problème d'unification des schémas* $E \dashv \mathcal{W}$ *possède une solution, c.-à.-d.* $E\psi$ *possède une solution pour chaque* ψ *avec* $\mathcal{D}om(\psi) \subseteq \mathcal{V}ar(E) - \mathcal{W}$, $\mathcal{VR}an(\psi) \cap \mathcal{W} = \emptyset$ *et* $\mathcal{VR}an(\psi) \subseteq \mathcal{V}_0$. *Si la réponse est positive, pour chaque* ψ *stricte, l'unificateur principal de* $E\psi$ *est* $\sigma \triangle \psi$, *où* σ *est l'unificateur principal de* $E \dashv \mathcal{W}$.

4.3 Contraintes d'appartenance avec variables de contexte

Hubert Comon [Com92] a étudié une logique équationnelle contrainte dans laquelle les contraintes sont des conditions d'appartenance $t \in \underline{s}$ où $\underline{s}$ est interprété comme un langage régulier d'arbres. La logique considérée contient un fragment de la logique équationnelle du second ordre, les variables de second ordre parcourant des ensembles réguliers de contextes. Le problème essentiel de la logique équationnelle contrainte est l'absence de lemme de paires critiques. Comon a proposé de nouvelles règles de déductions permettant de rétablir ce résultat. Mais le calcul des paires critiques utilise l'unification de certains termes de second ordre. Il a montré que le fragment nécessaire de l'unification de second ordre est décidable.

Comme les signatures avec sortes ordonnées ne sont rien d'autre que des automates ascendants d'arbres, la logique équationnelle avec sortes ordonnées rentre dans le cadre de ce travail. Les résultats présentés ici ne supposent ni la régularité de la signature, ni la propriété de décroissance des sortes.

4.3.1 Syntaxe

Soit $\mathcal{CX}$ l'alphabet infini des *symboles variables contextuels*, disjoints de $\mathcal{F} \cup \mathcal{X}$. Les variables qui appartiennent à $\mathcal{CX}$ sont notées par les majuscules: X, X_1, X', Y, Z, etc. $\mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{CX})$ est l'ensemble des termes finis construits sur la signature $\mathcal{F}$ et les variables $\mathcal{X} \cup \mathcal{CX}$, où chaque variable de $\mathcal{X}$ est d'arité 0 et chaque variable contextuelle de $\mathcal{CX}$ est d'arité 1.

Exemple 4.34 $f(X(x), Y(X(a))) \in \mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{CX})$ si $Y, X \in \mathcal{CX}$, $x \in \mathcal{X}$ et $a, f \in \mathcal{F}$ avec $ar(a) = 0$ et $ar(f) = 2$.

Soit $C(\mathcal{F})$ l'ensemble de tous les *contextes clos*. Le contexte $t[\cdot]_p \in C(\mathcal{F})$ peut être considéré aussi comme la λ-expression $\lambda u.t[u]_p : \mathcal{G}(\mathcal{F}) \longrightarrow \mathcal{G}(\mathcal{F})$. Soit $C(\mathcal{F}, \mathcal{X}, \mathcal{CX})$ l'ensemble de tous les termes du second ordre (monadiques) $t[X]_p$, où $X \in \mathcal{CX}$ et $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{CX})$. L'expression $X(Y) \in C(\mathcal{F}, \mathcal{X}, \mathcal{CX})$ est écrite aussi par $X \cdot Y$ ou bien $\lambda x.X(Y(x))$.

Soit Q un ensemble fini de *symboles des sortes*, disjoints de $\mathcal{F} \cup \mathcal{X} \cup \mathcal{CX}$. Soient $-_S$, $\top_S$, ϵ, $\top_C$, $-_C$ des symboles particuliers qui n'appartiennent pas aux $Q \cup \mathcal{F} \cup \mathcal{X} \cup \mathcal{CX}$. L'ensemble $\mathcal{SE}$ des *expressions de sortes* et l'ensemble $\mathcal{CE}$ des *expressions de contexte* sont définis comme les plus petits ensembles qui satisfont:

- $Q \cup \{\top_S, -_S\} \subseteq \mathcal{SE}$
- $\{\epsilon, \top_C, -_C\} \subseteq \mathcal{CE}$

- $q \wedge q', q \vee q', \neg q \in \mathcal{SE}$ si $q, q' \in \mathcal{SE}$
- $C \wedge C', C \vee C', \neg C \in \mathcal{CE}$ si $C, C' \in \mathcal{CE}$
- $f(\vec{q}) \in \mathcal{SE}$ si $f \in \mathcal{F}$ et $\vec{q} \subseteq \mathcal{SE}$
- $f(\vec{q})[C]_i \in \mathcal{CE}$ si $1 \leq i \leq |\vec{q}|$, $\vec{q} \subseteq \mathcal{SE}$ et $C \in \mathcal{CE}$
- $C \cdot q \in \mathcal{SE}$ si $C \in \mathcal{CE}$ et $q \in \mathcal{SE}$
- $C \cdot C' \in \mathcal{CE}$ si $C, C' \in \mathcal{CE}$
- $C^* \in \mathcal{CE}$ si $C \in \mathcal{CE}$.

L'ensemble des *formules* est le langage construit sur les formules atomiques:

- "$t[X]_p \in C$" où $t[X]_p \in C(\mathcal{F}, \mathcal{X}, \mathcal{CX})$
- "$t \in q$" où $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{CX})$ et $q \in \mathcal{SE}$
- "$s = t$" où $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{CX})$
- "$X = t[Y]_p$" où $X, Y \in \mathcal{CX}$ et $t[Y]_p \in C(\mathcal{F}, \mathcal{X}, \mathcal{CX})$

et les connecteurs logiques $\wedge$, $\vee$, $\exists x$, $\exists X$. De plus, $-$ est la disjonction vide et $\top$ est la conjonction vide. Soit Φ l'ensemble de toutes les formules construites ainsi. Une *contrainte d'appartenance* est une formule $\phi \in \Phi$ qui ne contient aucune égalité. Une *forme résolue d'appartenance* est une conjonction de contraintes de la forme $X \in C$ et $x \in q$, où X et x sont des variables qui apparaissent une seule fois.

Une *égalité contrainte* est une paire $(\phi, s = t)$ où ϕ est une contrainte d'appartenance $(e \in A)$ et $s = t$ est une égalité sur les termes de $\mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{CX})$. Une telle paire est notée $\phi : s = t$. Une *règle contrainte* est une égalité contrainte orientée, écrite $\phi : s \to t$. Il n'est pas nécessaire de supposer $\mathcal{V}ar(t) \subseteq \mathcal{V}ar(s)$.

4.3.2 Sémantique

Soit $\mathcal{A}$ un automate d'arbre fini ascendant dont Q est l'ensemble des états [GS84]. Notons $\mathcal{P}(M)$ l'ensemble de tous les sous-ensembles de M et $C(\mathcal{F}) = \{f \mid f : \mathcal{G}(\mathcal{F}) \longrightarrow \mathcal{G}(\mathcal{F})\}$ l'ensembles de tous les endomorphismes sur $\mathcal{G}(\mathcal{F})$. L'interprétation sémantique surchargée $[\![\cdot]\!]_{\mathcal{A}} : \mathcal{SE} \longrightarrow \mathcal{P}(\mathcal{G}(\mathcal{F}))$ et $[\![\cdot]\!]_{\mathcal{A}} : \mathcal{CE} \longrightarrow \mathcal{P}(\mathcal{C}(\mathcal{F}))$ est définie par:

- $[\![q]\!]_{\mathcal{A}} = \{\alpha \mid q \vdash_{\mathcal{A}} \alpha\}$ est le langage reconnu par $\mathcal{A}$ dans l'état q, si $q \in Q$.
- $[\![-_S]\!]_{\mathcal{A}} = \emptyset$, $[\![\top_S]\!]_{\mathcal{A}} = \mathcal{G}(\mathcal{F})$, $[\![\top_C]\!]_{\mathcal{A}} = C(\mathcal{F})$, $[\![-_C]\!]_{\mathcal{A}} = \emptyset$, $[\![\epsilon]\!]_{\mathcal{A}} = \{id_{\mathcal{G}(\mathcal{F})}\}$ où $id_{\mathcal{G}(\mathcal{F})}$ est l'identité sur $\mathcal{G}(\mathcal{F})$.
- $[\![\cdot]\!]_{\mathcal{A}}$ est un morphisme de $(\mathcal{SE}, \wedge, \vee, \neg)$ dans $(2^{\mathcal{G}(\mathcal{F})}, \cap, \cup, \bar{\cdot})$ où $\bar{A}$ est le complément de A dans $\mathcal{G}(\mathcal{F})$. Ceci signifie, par exemple, que $[\![q \wedge q']\!]_{\mathcal{A}} = [\![q]\!]_{\mathcal{A}} \cap [\![q']\!]_{\mathcal{A}}$.
- $[\![\cdot]\!]_{\mathcal{A}}$ est un morphisme de $(\mathcal{CE}, \wedge, \vee, \neg)$ dans $(2^{C(\mathcal{F})}, \cap, \cup, \bar{\cdot})$

- $[\![f(\vec{q})]\!]_{\mathcal{A}} = \{f(\vec{t}) \in \mathcal{G}(\mathcal{F}) \mid t_i \in [\![q_i]\!]_{\mathcal{A}}\}$
- $[\![f(\vec{q})[C]_i]\!]_{\mathcal{A}} = \{\lambda x.f(\vec{t})[u[x]_p]_i \in C(\mathcal{F}) \mid \lambda x.u[x]_p \in [\![C]\!]_{\mathcal{A}}, \forall i,\, t_i \in [\![q_i]\!]_{\mathcal{A}}\}$
- $[\![C \cdot q]\!]_{\mathcal{A}} = \{t[u]_p \in \mathcal{G}(\mathcal{F}) \mid u \in [\![q]\!]_{\mathcal{A}}, t[\cdot]_p \in [\![C]\!]_{\mathcal{A}}\}$
- $[\![C \cdot C']\!]_{\mathcal{A}} = \{\lambda x.t[u[x]_p]_q \in C(\mathcal{F}) \mid t[\cdot]_q \in [\![C]\!]_{\mathcal{A}}, u[\cdot]_p \in [\![C']\!]_{\mathcal{A}}\}$
- $[\![C^*]\!]_{\mathcal{A}} = \bigcup_{n\geq 0}[\![C^n]\!]_{\mathcal{A}}$ où $C^0 = \{id_{\mathcal{G}(\mathcal{F})}\}$ et $C^n = C \cdot C^{n-1}$ pour $n \geq 1$.

Une *affectation close* est une interprétation σ de $\mathcal{X} \cup \mathcal{C}\mathcal{X}$ dans $\mathcal{G}(\mathcal{F}) \cup C(\mathcal{F})$, telle que $\sigma|_{\mathcal{X}}$ est une interprétation de $\mathcal{X}$ dans $\mathcal{G}(\mathcal{F})$ et $\sigma|_{\mathcal{C}\mathcal{X}}$ est une interprétation de $\mathcal{C}\mathcal{X}$ dans $C(\mathcal{F})$. Une telle affectation est élargie par l'homomorphisme sur les $\mathcal{F}$-algèbres de $\mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{C}\mathcal{X})$ dans $\mathcal{G}(\mathcal{F})$.

Une *solution* d'une équation $s = t$ est une affectation close σ telle que $s\sigma \equiv t\sigma$. Une *solution* d'une contrainte d'appartenance $t \in q$ (ou bien $X \in C$) par rapport à l'automate $\mathcal{A}$ est une affectation close σ telle que $t\sigma \in [\![q]\!]_{\mathcal{A}}$ (ou bien $X\sigma \in [\![C]\!]_{\mathcal{A}}$). Ces définitions des solutions sont étendues de façon naturelle aux formules Φ. L'expression $[\![\phi]\!]_{\mathcal{A}}$ désigne l'ensemble de toutes les solutions de $\phi \in \Phi$ par rapport à l'automate $\mathcal{A}$.

Hubert Comon considère une "sémantique engendrée par les termes" de sa logique avec des contraintes (comme dans [KKR90]): une égalité contrainte (ou bien une règle contrainte) est une représentation d'un ensemble infini d'égalités (... de règles).

Soit $\mathcal{A}$ un automate dont Q est l'ensemble des états, alors

$$[\![\phi\colon s = t]\!]_{\mathcal{A}} \quad = \quad \{s\sigma = t\sigma \mid \sigma \in [\![\phi]\!]_{\mathcal{A}}\}$$

Une définition similaire est valide pour les règles.

Toutes les définitions pour les systèmes de réécriture s'appliquent sur le cas contraint de la façon naturelle. Par exemple, la relation $\longleftrightarrow_{E,\mathcal{A}}$ est interprétée comme la relation $\longleftrightarrow_{[\![E]\!]_{\mathcal{A}}}$.

L'ensemble des égalités contraintes et des règles contraintes dans [Com92] schématisent uniquement des systèmes *clos*. Malgré cela, il s'intéresse à ces systèmes pour deux raisons:

1. son but est de prouver des propriétés des langages de spécification équationnelle avec contraintes. La sémantique déclarative de tels programmes utilise l'ensemble des instances closes de tous les axiomes sans besoin de constructions supplémentaires.

2. pour prouver par réfutation $E \models \forall \vec{x}\colon s = t$, on a besoin de considérer uniquement toutes les instances de E appartenant à $\mathcal{T}(\mathcal{F}, \vec{x})$: on n'a besoin que des instances closes des axiomes.

4.3.3 Utilisation

Sur ce formalisme, Hubert Comon bâtit une procédure de complétion d'ensembles infinis d'égalités (closes), présentées par des égalités contraintes. Les règles de transition classiques ne sont pas complètes, c'est-à-dire qu'on peut avoir des cas où aucune des règles classiques

ne s'applique et où pourtant il existe des égalités (closes) valides qui n'ont pas de preuve de réécriture. Comon présente trois règles supplémentaires de déduction et reconstruit ainsi le lemme des paires critiques.

Ensuite, il développe l'algorithme d'unification dans le nouveau formalisme et présente la méthode de solution des contraintes, mais aucune application explicite à la divergence n'est développée, aucune méthode de construction des systèmes avec contraintes d'appartenance pour les systèmes divergents n'est présentée.

4.4 Termes récurrents

Les termes récurrents, appelés originellement *hyper-termes*, avec leurs variantes, les ω-termes ou ρ-termes [1], ont été dévéloppés par Hong Chen, Jieh Hsiang et H.-C. Kong [CHK90, CH91a, CH91b].

4.4.1 Syntaxe

Définition 4.35 *Soient Φ un symbole spécial d'arité 3 et $\mathcal{C}$ l'ensemble de variables représentant les entiers naturels, appelées* variables compteurs. *L'ensemble de* **termes récurrents** *est le plus petit ensemble $\mathcal{R}(\mathcal{F},\mathcal{X},\mathcal{C})$ tel que:*

1. $\mathcal{X} \subset \mathcal{R}(\mathcal{F},\mathcal{X},\mathcal{C})$
2. *Si $f \in \mathcal{F}$ avec $ar(f) = k$ et $\vec{t} \in \mathcal{R}(\mathcal{F},\mathcal{X},\mathcal{C})^k$, alors $f(\vec{t}) \in \mathcal{R}(\mathcal{F},\mathcal{X},\mathcal{C})$.*
3. *Si $h[\cdot]_p$ est un contexte avec $p \neq \Lambda$, $n \in \mathcal{C} \cup \mathcal{N}$, et $l \in \mathcal{R}(\mathcal{F},\mathcal{X},\mathcal{C})$, alors $\Phi(h[\cdot]_p, n, l) \in \mathcal{R}(\mathcal{F},\mathcal{X},\mathcal{C})$.*

L'expression $\Phi(h[\cdot]_p, n, l)$ s'appelle un générateur *ou bien un n-*générateur *si l'on a besoin de mettre n en évidence.*

Définition 4.36 *Si dans la définition 4.35 on remplace l'expression "*terme récurrent*" par "***ρ*-terme***", $\mathcal{R}(\mathcal{F},\mathcal{X},\mathcal{C})$ par $\mathcal{R}^\rho(\mathcal{F},\mathcal{X},\mathcal{C})$ et la condition 3 par*

3. *Si $h[l]_p \in \mathcal{G}(\mathcal{F})$ est un terme clos avec une position $p \neq \Lambda$, et $n \in \mathcal{C}$, alors $\Phi(h[\cdot]_p, n, l) \in \mathcal{R}^\rho(\mathcal{F},\mathcal{X},\mathcal{C})$.*

*on obtient la sous-classe des ρ-*termes *$\mathcal{R}^\rho(\mathcal{F},\mathcal{X},\mathcal{C})$.*

Il n'est pas indispensable que $h[l]_p$ soit un terme clos dans la définition 4.36. Nous pouvons prendre aussi un terme $h[l]_p \in \mathcal{T}(\mathcal{F},\mathcal{X})$ et skolémiser ses variables $\mathcal{V}ar(h[l]_p)$ en des nouvelles constantes.

1. Les auteurs ont changé de terminologie d'un article à l'autre.

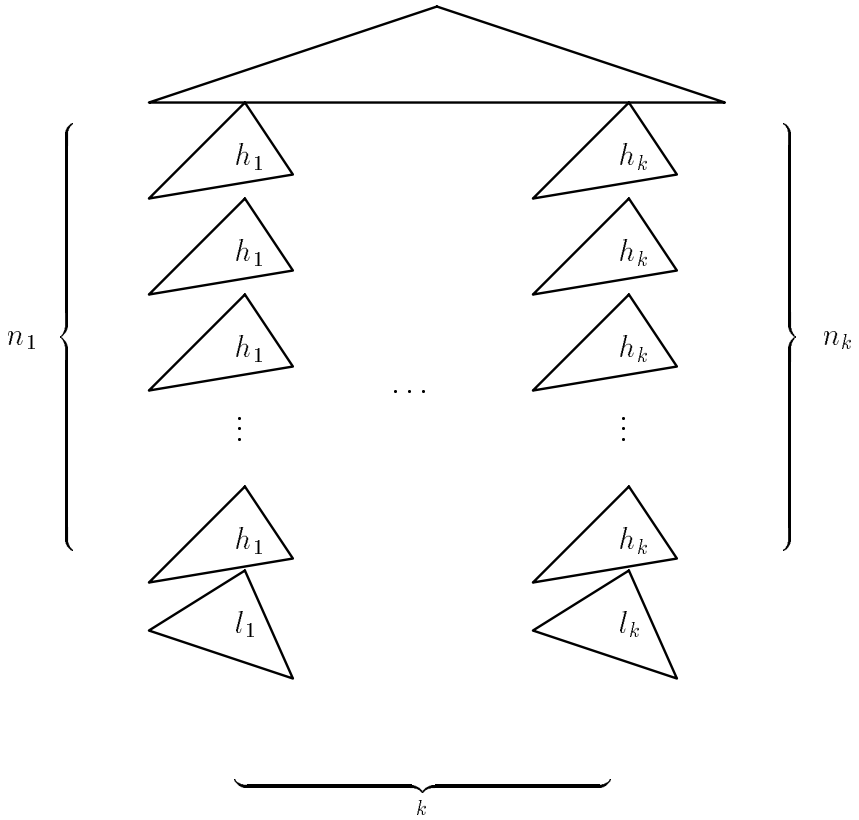


FIG. 4.1 – *Un ρ-terme typique*

4.4.2 Sémantique

Les termes récurrents (ρ-termes) sont utilisés pour schématiser des suites infinies de termes qui partagent des structures répétitives communes. Ces structures sont décrites par les *générateurs* $\Phi(h[\cdot]_p, n, l)$, où $h[\cdot]_p$ est un contexte répétitif déplié n-fois et l est utilisé pour boucher le trou du contexte à la fin du processus de dépliage. Si n est une variable compteur de $\mathcal{C}$, elle est instanciée par un entier naturel avant l'évaluation du générateur. Un terme récurrent (ρ-terme) avec des variables compteurs instanciées est évalué par le *dépliage* des générateurs:

$$\begin{aligned} \mathit{unfold}(\Phi(h[\cdot]_p, s(n), l)) &\rightarrow h\sigma[\mathit{unfold}(\Phi(h[\cdot]_p, n, l))]_p \\ \mathit{unfold}(\Phi(h[\cdot]_p, 0, l)) &\rightarrow l \end{aligned}$$

où σ est un renommage des variables (pour les termes récurrents) ou bien la substitution vide (pour les ρ-termes).

L'ensemble $\Omega(t)$ de termes représentés par le terme récurrent (le ρ-terme) t est

$$\Omega(t) = \{\mathit{unfold}(t[N_1 \mapsto n_1, \ldots, N_k \mapsto n_k]) \mid \vec{n} \in \mathcal{N}^k, \vec{N} = \mathcal{CV}ar(t)\}$$

Chaque élement de $\Omega(t)$ s'appelle un *constituant* de t.

4.4.3 Utilisation

Les termes récurrents sont utilisés comme outils de schématisation de certains systèmes de réécriture divergents. Une méthode empirique de construction des termes récurrents à partir de suites de termes a été présentée dans [CHK90], avec quelques exemples intéressants de systèmes divergents dont la schématisation est déduite par ce moyen. Ensuite, l'algorithme de filtrage des termes récurrents (avec les preuves de la complétude, consistance et terminaison) et la définition des systèmes de réécriture récurrents sont présentés. Une représentation intéressante des termes récurrents par le biais de l'unification équationnelle est donnée par Giegerich et Ohlenbusch [GO91].

[CH91a, CH91b] traitent les aspects d'unification et de programmation en logique des ρ-termes. En particulier, l'algorithme d'unification des ρ-termes est présenté avec ses preuves de complétude, de consistance et de terminaison, ainsi qu'avec un calcul de sa complexité dans le pire cas. Cet algorithme d'unification a été implanté par Amaniss [Ama92].

4.5 Généralisation de Comon

Hubert Comon a proposé dans [Com95] de généraliser la définition 4.36 de la façon suivante:

4.5.1 Syntaxe

Définition 4.37 *L'ensemble $\mathcal{R}^\eta(\mathcal{F},\mathcal{X},\mathcal{C})$ de* ***HC*-termes** *est le plus petit ensemble tel que:*

- $\mathcal{X} \subset \mathcal{R}^\eta(\mathcal{F},\mathcal{X},\mathcal{C})$
- *Si $f \in \mathcal{F}$ avec $ar(f) = n$ et $\vec{t} \in \mathcal{R}^\eta(\mathcal{F},\mathcal{X},\mathcal{C})^n$, alors $f(\vec{t}) \in \mathcal{R}^\eta(\mathcal{F},\mathcal{X},\mathcal{C})$.*
- *Si $t, t' \in \mathcal{R}^\eta(\mathcal{F},\mathcal{X},\mathcal{C})$, $p \in \mathcal{P}os(t)$ et $n \in \mathcal{C}$ alors $t[t']_p^n \in \mathcal{R}^\eta(\mathcal{F},\mathcal{X},\mathcal{C})$.*

où $\mathcal{P}os(t[t']_p^n) = \mathcal{P}os(t) - \{p\}$ est l'extension des positions sur $\mathcal{R}^\eta(\mathcal{F},\mathcal{X},\mathcal{C})$.

4.5.2 Sémantique

Les variables compteurs $\mathcal{C}$ sont instanciées par des entiers naturels. Un HC-terme avec des variables compteurs instanciées est évalué par le *dépliage*:

$$\begin{aligned} unfold(f(t_1,\ldots,f_k)) &\rightarrow f(unfold(t_1),\ldots,unfold(t_k)) \\ unfold(t[t']_p^{s(n)}) &\rightarrow unfold(t)[unfold(t[t']_p^n)]_p \\ unfold(t[t']_p^0) &\rightarrow unfold(t') \end{aligned}$$

4.5.3 Rapport avec les ρ-termes

Comon [Com95] a prouvé que la classe de ρ-termes est strictement incluse dans la classe de HC-termes.

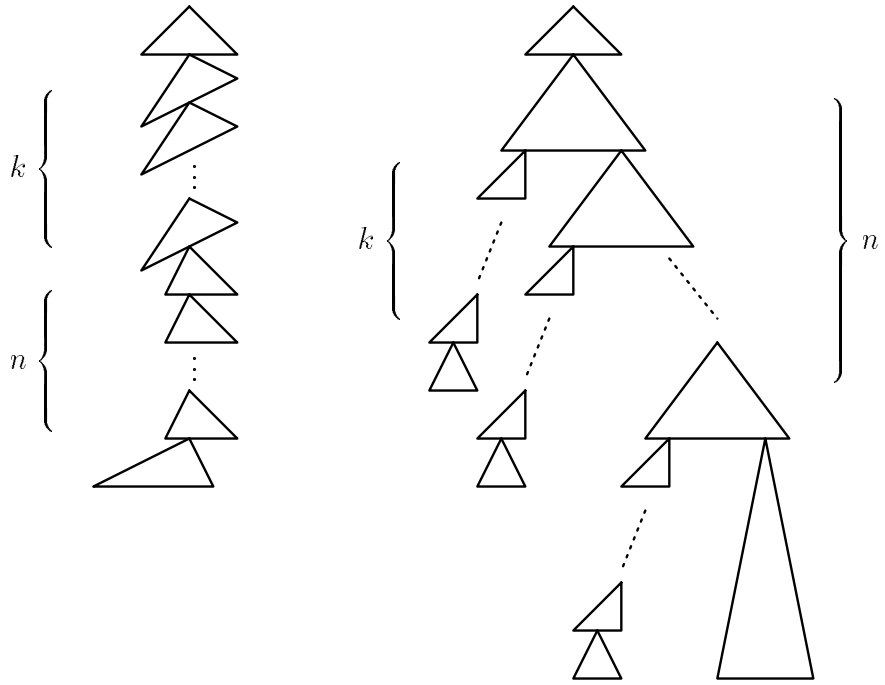


FIG. 4.2 – *Des GC/HC-termes typiques qui ne sont pas des ρ-termes*

4.6 Généralisation de Salzer

Gernot Salzer a proposé dans [Sal92] de généraliser la définition 4.36 de la façon suivante:

4.6.1 Syntaxe

L'*algèbre additive de Presburger* $\mathcal{PA}_+(\mathcal{C})$ est le plus petit ensemble tel que:

- $\mathcal{C} \subset \mathcal{PA}_+(\mathcal{C})$
- $\mathcal{N} \subset \mathcal{PA}_+(\mathcal{C})$
- Si $e, e' \in \mathcal{PA}_+(\mathcal{C})$ alors $e + e' \in \mathcal{PA}_+(\mathcal{C})$.
- Si $n \in \mathcal{N}$ et $e \in \mathcal{PA}_+(\mathcal{C})$ alors $n * e \in \mathcal{PA}_+(\mathcal{C})$.

Chaque élément e dans l'algèbre additive de Presburger $\mathcal{PA}_+(\mathcal{C})$ représente une expression linéaire $(n_1 * e_1) + \cdots + (n_k * e_k)$ où $\vec{n} \in \mathcal{N}^k$ et $\vec{e} \in \mathcal{C}^k$ pour chaque $k = 0, 1, \ldots$

Définition 4.38 *L'ensemble* $\mathcal{R}^\gamma(\mathcal{F}, \mathcal{X}, \mathcal{C})$ *de* ***GS*-termes** *est le plus petit ensemble tel que:*

- $\mathcal{T}(\mathcal{F}, \mathcal{X}) \subset \mathcal{R}^\gamma(\mathcal{F}, \mathcal{X}, \mathcal{C})$
- *Si* $h, l \in \mathcal{R}^\gamma(\mathcal{F}, \mathcal{X}, \mathcal{C})$, *la variable* $x \in \mathcal{X}$ *est un sous-terme stricte de* h *et* $n \in \mathcal{PA}_+(\mathcal{C})$, *alors* $\Phi(h[x \mapsto \diamond], n, l) \in \mathcal{R}^\gamma(\mathcal{F}, \mathcal{X}, \mathcal{C})$.

– *Si* $t, t' \in \mathcal{R}^{\gamma}(\mathcal{F}, \mathcal{X}, \mathcal{C})$ *et* $x \in \mathcal{X}$ *alors* $t[x \mapsto t'] \in \mathcal{R}^{\gamma}(\mathcal{F}, \mathcal{X}, \mathcal{C})$.

Le symbole $\diamond$, représentant un trou dans les contextes, est interprété comme une nouvelle variable.

4.6.2 Sémantique

Les GS-termes sont utilisés pour schématiser des suites infinies de termes qui partagent des structures répétitives communes. Ces structures sont décrites par les *générateurs* $\Phi(h, e, l)$, où h est un contexte répétitif déplié avec des trous $\diamond$, le terme additif de Presburger e dirige le dépliage du contexte dans tous les trous et le terme l est utilisé à la fin du processus de dépliage. Le processus de dépliage d'un générateur est défini récursivement:

$$\begin{array}{rcll} \mathit{unfold}(\Phi(h, v, l)) & \rightarrow & \Phi(h, v, l) & \text{si } v \in \mathcal{C} \\ \mathit{unfold}(\Phi(h, n, l)) & \rightarrow & \diamond[\diamond \mapsto h]^n[\diamond \mapsto l] & \text{si } n \in \mathcal{N} \\ \mathit{unfold}(\Phi(h, n * e, l)) & \rightarrow & \mathit{unfold}(\Phi(\diamond[\diamond \mapsto h]^n, e, l)) & \text{si } n \in \mathcal{N} \text{ et } e \in \mathcal{PA}_+(\mathcal{C}) \\ \mathit{unfold}(\Phi(h, e + e', l)) & \rightarrow & \mathit{unfold}(\Phi(h, e, \Phi(h, e', l))) & \text{si } e, e' \in \mathcal{PA}_+(\mathcal{C}) \end{array}$$

Salzer définit la sémantique de tous les GS-termes par dépliage, permettant ainsi des évaluations partielles de générateurs. Il suffit de considérer seulement la deuxième règle de dépliage

$$\mathit{unfold}(\Phi(h, n, l)) \rightarrow \diamond[\diamond \mapsto h]^n[\diamond \mapsto l] \quad \text{si } n \in \mathcal{N}$$

pour les GS-termes énumérés pendant le calcul d'un constituant. Avant le dépliage d'un GS-terme énuméré, chacun de ses termes additifs de Presburger énumérés et chacun de ses termes additifs de Presburger clos peuvent être réduits à leur forme normale qui appartient aux entiers naturels $\mathcal{N}$ par le système de réécriture canonique suivant:

$$\begin{array}{rcl} x + 0 & \rightarrow & x \\ x + s(y) & \rightarrow & s(x + y) \\ x * 0 & \rightarrow & 0 \\ x * s(y) & \rightarrow & (x * y) + x \end{array}$$

La pseudo-substitution $[\diamond \mapsto h]$ remplace chaque occurrence du trou $\diamond$ par h. Car chaque trou $\diamond$ est localement défini pour un générateur, la substitution $[\diamond \mapsto h]$ ne peut pas remplacer d'autres trous appartenant aux autres générateurs.

Le dépliage unfold^* d'un GS-terme est défini comme le dépliage de tous ces générateurs.

4.6.3 Rapport avec les HC-termes

Amaniss, Lugiez et moi-même [AHL93], nous avons prouvé que la classe des HC-termes est strictement incluse dans la classe des GS-termes.

Chapitre 5

Grammaires primales

Nota: Dans ce chapitre, "schématisation" sans plus de précision signifie une "schématisation par des grammaires primales".

5.1 Construction des grammaires primales

Dans la schématisation par les grammaires primales, la signature $\mathcal{F}$ est constituée des *constructeurs* $\mathcal{K}$, des *symboles définis* $\mathcal{D}$ tels que $\mathcal{K} \cap \mathcal{D} = \emptyset$, ainsi que du symbole sucesseur s et de la constante zéro 0, qui ne sont inclus ni dans $\mathcal{K}$ ni dans $\mathcal{D}$. Les symboles définis $\mathcal{D}$ sont surmontés d'un accent circonflexe pour les distinguer des symboles constructeurs $\mathcal{K}$. De plus, nous considérons l'ensemble des *variables compteurs* $\mathcal{C}$, telles que $\mathcal{X} \cap \mathcal{C} = \emptyset$.

Les arguments des symboles définis $\hat{f} \in \mathcal{D}$ sont divisés en deux parties par un point-virgule (cf. [Sim88]). Ceux *avant* le point-virgule sont appelés des *compteurs*, ou *variables-compteurs* s'ils sont représentés par une variable. Le premier compteur d'un symbole auxiliaire est dit le *compteur principal*, les autres sont dits des *compteurs secondaires*. Chaque symbole défini $\hat{f}$ possède une *arité de compteurs*, noté $ar_c(\hat{f})$, qui indique le nombre de ses compteurs.

La manipulation des compteurs nécessite la définition d'une algèbre des *expressions compteurs* $\mathcal{N}(\mathcal{C})$, qui est le plus petit ensemble tel que:

- $0 \in \mathcal{N}(\mathcal{C})$
- $\mathcal{C} \subset \mathcal{N}(\mathcal{C})$
- si $n \in \mathcal{N}(\mathcal{C})$ alors $s(n) \in \mathcal{N}(\mathcal{C})$

Au lieu d'écrire $s(c)$ et $s^k(c)$, nous allons utiliser la notation naturelle, c.-à.-d. $c+1$ et $c+k$. Ainsi, dans la suite $s(0)$ sera souvent noté 1, $s(s(0))$ ou $s(1)$ noté 2, $s(2)$ noté 3, etc.

L'algèbre de *termes primaux* $\mathcal{TP}(\mathcal{K}, \mathcal{D}; \mathcal{X}, \mathcal{C})$ est le plus petit ensemble tel que:

- $\mathcal{X} \subset \mathcal{TP}(\mathcal{K}, \mathcal{D}; \mathcal{X}, \mathcal{C})$

- si $\vec{n} \in \mathcal{N}(\mathcal{C})^k$, $\vec{t} \in \mathcal{TP}(\mathcal{K},\mathcal{D};\mathcal{X},\mathcal{C})^l$ et $\hat{f} \in \mathcal{D}$, où $ar_c(\hat{f}) = k$ et $ar(\hat{f}) = k+l$, alors $\hat{f}(\vec{n};\vec{t}) \in \mathcal{TP}(\mathcal{K},\mathcal{D};\mathcal{X},\mathcal{C})$
- si $\vec{t} \in \mathcal{TP}(\mathcal{K},\mathcal{D};\mathcal{X},\mathcal{C})^k$ et $f \in \mathcal{K}$, où $ar(f) = k$, alors $f(\vec{t}) \in \mathcal{TP}(\mathcal{K},\mathcal{D};\mathcal{X},\mathcal{C})$

Sans que cela soit précisé explicitement, l'algèbre $\mathcal{TP}(\mathcal{K},\mathcal{D};\mathcal{X},\mathcal{C})$ est typée: les expressions avant le point-virgule représentent des entiers naturels, tandis que celles après le point-virgule sont des termes primaux, et $\hat{f}(\vec{n};\vec{t})$ est lui aussi un terme primal.

L'ensemble $\mathcal{CP}os(t) = \{a.\mathtt{n} \mid \mathcal{H}ead(t|_a) = \hat{f} \in \mathcal{D}, \mathtt{n} \leq ar_c(\hat{f})\}$ s'appelle l'ensemble des *positions compteurs* dans un terme primal t. Ce sont les positions dans t immédiatement en-dessous d'un symbole défini $\hat{f}$, avant le point-virgule. L'ensemble des variables compteurs d'un terme t est noté

$$\mathcal{CV}ar(t) \quad = \quad \{t|_a \mid a \in \mathcal{CP}os(t) \cap \mathcal{VP}os(t)\}$$

L'ensemble des *positions redex* d'un terme primal t est noté

$$\mathcal{DP}os(t) \quad = \quad \{a \in \mathcal{FP}os(t) \mid \mathcal{H}ead(t|_a) \in \mathcal{D}\}$$

Une position redex est la position à laquelle on peut réduire le terme t. L'ensemble des *positions redex extérieures* d'un terme primal t est noté

$$\begin{aligned} \mathcal{OP}os(t) &= \{a \in \mathcal{DP}os(t) \mid a \leq b \text{ ou } a \parallel b \text{ pour chaque } b \in \mathcal{DP}os(t)\} \\ &= \liminf_{\leq} \mathcal{DP}os(t) \end{aligned}$$

Un terme primal t est dit *régulier* (non-encapsulé) si ses positions redex $\mathcal{DP}os(t)$ sont mutuellement parallèles. Autrement dit, il n'existe pas deux positions redex différentes a et b appartenant à $\mathcal{DP}os(t)$, telles que a soit le préfixe de b. Un terme primal t est dit *encapsulé* s'il existe deux positions redex différentes a et b, telles que le redex $t|_a$ est un sous-terme propre de $t|_b$.

L'*approximation* d'un terme primal $\hat{f}(n, c_1, \ldots, c_k; t_1, \ldots, t_n)$ définie pour chaque expression compteur n et en respectant la précédence $\succ$ entre les symboles définis, est l'ensemble des termes primaux:

$$\mathcal{A}px(\hat{f}(n,\vec{c};\vec{t})) \quad = \quad \{\hat{g}(\vec{c'};\vec{t'}) \mid \hat{f} \succ \hat{g},\ \vec{c'} \sqsubseteq \vec{c},\ \vec{t'} \sqsubseteq \vec{t}\}$$

L'*emballage* d'un terme primal t noté $\mathcal{W}rp(t)$, est le contexte $t[\cdot]_{\mathcal{DP}os(t)}$ tel que tout sous-terme commençant par un symbole défini est remplacé par le "trou". Donc tout emballage ne doit contenir que des constructeurs $\mathcal{K}$ et des variables ordinaires $\mathcal{X}$.

Exemple 5.1 Soient $\mathcal{K} = \{*, F, G, a, b\}$ et $\mathcal{D} = \{\hat{e}, \hat{f}, \hat{g}, \hat{h}, \hat{k}\}$ respectivement l'ensemble des constructeurs et l'ensemble des symboles définis avec la précédence $\hat{f} \succ \hat{g} \succ \hat{h} \succ \hat{k}$.

$$\hat{f}(c_1+3, c_2+5; F(x), y) * (\hat{f}(c_3+1, c_4+2; a, F(b)) * \hat{f}(c_3, c_1+10; F(\hat{g}(c_2)), \hat{h}(c_4; a*b)))$$

est un terme primal si c_1, c_2, c_3 et c_4 appartiennent à $\mathcal{C}$, mais ce terme n'est pas régulier. En effet, le redex $\hat{g}(c_2)$ est encapsulé dans le redex $\hat{f}(c_3, c_1+10; F(\hat{g}(c_2)))$. Par contre, le terme $\hat{g}(c_2+1; F(x))$ est un terme primal régulier.

L'approximation du terme primal $t = \hat{f}(c_1, c_2; x, y)$ est représentée par l'ensemble des termes primaux suivants:

$$\mathcal{A}px(t) \quad = \quad \{\hat{g}(c_1), \hat{g}(c_2), \hat{h}(c_1; x), \hat{h}(c_2; x), \hat{h}(c_1; y), \hat{h}(c_2; y), \hat{k}(c_1, c_2; x, y)\}$$

Par contre, les termes primaux $\hat{k}(c_2, c_1; x, y)$ et $\hat{k}(c_1, c_2; y, x)$ n'appartiennent pas à l'approximation de t car dans le premier cas $\langle c_2, c_1 \rangle$ n'est pas une sous-suite de $\langle c_1, c_2 \rangle$ et dans le deuxième cas $\langle y, x \rangle$ n'est pas une sous-suite de $\langle x, y \rangle$. Le terme primal $\hat{e}(c_1; x)$ n'appartient pas non plus à l'approximation $\mathcal{A}px(t)$ car $\hat{f} \not\succ \hat{e}$.

L'emballage du terme primal $(\hat{h}(c_1; x) + \hat{h}(c_2; y)) * F(\hat{k}(c_1, c_2; x, y))$ est le contexte

$$(\cdot + \cdot) * F(\cdot)$$

5.1.1 Systèmes de réécriture Presburger

Si nous regardons la structure générale des règles d'une famille itérée, il est clair qu'il nous faut un appareil assez complexe pour la schématiser. Malgré leur complexité structurelle, les termes d'une famille itérée partagent des caractéristiques communes:

Répétition des radicaux: Les mêmes radicaux sont itérés dans chaque terme de la suite. Si un tel radical est déplié n fois sur un chemin donné du terme t_n, alors il est déplié $n + 1$ fois sur le même chemin du terme t_{n+1}.

Hiérarchie des radicaux: Dans chaque étape de dépliage d'un radical (dit *principal*), d'autres radicaux (dits *secondaires*) peuvent commencer à se déplier sur des branches parallèles.

Séquence des radicaux: Si le dépliage d'un radical est terminé, il peut donner naissance au dépliage d'un autre radical (dit *subordonné*) sur le même chemin.

Rapport entre les radicaux: Le numéro d'étape de dépliage d'un radical principal peut — mais pas obligatoirement — être en rapport avec le nombre de dépliages autorisés pour les radicaux secondaires ou subordonnés.

Ces quatre caractéristiques indiquent qu'il faut trouver un moyen de schématisation à l'aide de compteurs (des compteurs font partie des paramètres des symboles définis), avec la possibilité de produire les radicaux (chaque symbole défini donne naissance à un radical répété), avec la possibilité de hiérarchiser et séquentialiser des radicaux (on l'a en définissant une précédence sur les symboles définis) et enfin avec la possibilité d'associer deux radicaux (on l'aura en reliant deux compteurs comme des vases communicants: si l'un décroît, l'autre augmente). Ces exigences impliquent l'utilisation de systèmes primitifs récursifs spéciaux [Pét67]. En revanche, il faut éviter de recourir à la récursion primitive dans sa totalité car on veut garder de bonnes propriétés de décision comme l'unification par exemple. Ainsi, on ne doit pas pouvoir simuler par ces systèmes la multiplication des entiers (donc le 10^e problème de Hilbert) ou le problème de Post.

Le système de réécriture Presburger cible uniquement les symboles définis $\mathcal{D}$, d'où la nécessité d'une précédence sur $\mathcal{D}$. A chaque symbole défini $\hat{f}$ est associé une paire de règles de réécriture.

Définition 5.2 *Supposons qu'il existe une précédence $\succ$ sur les symboles définis $\mathcal{D}$. Le* **système de réécriture Presburger** $\mathcal{R}$ *contient pour chaque symbole défini $\hat{f}$ une paire de règles de réécriture:*

– *une règle de* base

$$\hat{f}(0, \vec{c}; \vec{x}) \quad \rightarrow \quad r_1^{\hat{f}}$$

– *une règle* inductive *qui peut avoir l'une des deux formes suivantes:*

$$\hat{f}(n+1, \vec{c}; \vec{x}) \quad \rightarrow \quad r_2^{\hat{f}}[\hat{f}(n, \vec{c}; \vec{x})]_A \tag{5.1}$$

$$\hat{f}(n+1, \vec{c}; \vec{x}) \quad \rightarrow \quad r_2^{\hat{f}}[\hat{f}(n, c_1, \ldots, c_{i-1}, c_i + 1, c_{i+1}, \ldots, c_k; \vec{x})]_A \tag{5.2}$$

tels que

– *$\vec{c}$ est le vecteur des variables compteurs, $\vec{x}$ le vecteur des variables ordinaires,*

– *A est l'ensemble fini des positions mutuellement parallèles, autres que la racine de $r_2^{\hat{f}}$,*

– *$r_2^{\hat{f}}[\hat{f}(n, \vec{c}; \vec{x})]_A$ et $r_2^{\hat{f}}[\hat{f}(n, c_1, \ldots, c_{i-1}, c_i + 1, c_{i+1}, \ldots, c_k; \vec{x})]_A$ sont des termes réguliers,*

– *tous les redex de $r_1^{\hat{f}}$ et $r_2^{\hat{f}}$ appartiennent à l'approximation $\mathcal{A}px(\hat{f}(n, \vec{c}; \vec{x}))$,*

– *le symbole racine de $r_2^{\hat{f}}$ est un constructeur,*

– *chaque variable ordinaire doit appartenir à $r_1^{\hat{f}}$ ou à $r_2^{\hat{f}}$.*

n est appelé le compteur actif. Si la règle inductive a la forme (5.2) alors le compteur c_i est dit lié avec le compteur actif n. Si par contre la règle inductive apparaît sous la première forme (5.1), les variables compteurs $\vec{c}$ sont considérées indépendantes de n.

Pour chaque symbole défini $\hat{f}$ le terme $r_2^{\hat{f}}$ contient le radical à répéter. Les symboles définis dans le terme $r_2^{\hat{f}}$ donnent naissance aux radicaux secondaires, ceux dans le terme $r_1^{\hat{f}}$ aux radicaux subordonnés. Les compteurs secondaires du symbole défini $\hat{f}$ peuvent être diffusés aux symboles définis $\hat{g}_i$ dans les termes $r_1^{\hat{f}}$ et $r_2^{\hat{f}}$, établissant ainsi le lien direct entre le nombre de répétitions du radical du symbole $\hat{f}$ et des radicaux des symboles $\hat{g}_i$. Les "vases communicants" sont établis entre les compteurs n et c_i par la règle inductive de la forme (5.2).

Les systèmes Presburger sont des systèmes de réécriture primitifs récursifs un peu particuliers. En effet, la récursion primitive est restreinte à une sous-classe de termes réguliers donc sans aucune possibilité de simuler la multiplication; d'où la restriction des termes dans le membre droit des règles inductives à des termes réguliers. En fait, ceci permet de rendre

impossible la simulation du 10^e problème de Hilbert par l'unification de deux termes primaux modulo un système de réécriture Presburger.

Ainsi la nécessité d'avoir des termes réguliers et des approximations dans le membre droit des règles de réécriture nous garantit la décidabilité de l'unification de termes primaux modulo le système de réécriture Presburger.

L'existence d'un symbole constructeur à la racine de $r_2^{\hat{f}}$ permet d'éviter des règles inductives du type: $\hat{f}(n+1,\vec{c};\vec{x}) \to \hat{f}(n,\vec{c};\vec{x})$ qui pourraient faire disparaître le symbole défini $\hat{f}$. En effet, les deux dernières conditions de la définition 5.2 ne sont pas nécessaires car chaque système Presburger qui ne satisfait pas ces deux dernières conditions peut être transformé en un système Presburger équivalent qui les satisfait.

Exemple 5.3 Supposons que $\mathcal{D} = \{\hat{f},\hat{g},\hat{h}\}$ et $\hat{f} \succ \hat{g} \succ \hat{h}$. Le système de réécriture

$$\begin{aligned}
\hat{f}(0,v,w;x,y) &\to \hat{g}(v,w;x,y)\\
\hat{f}(u+1,v,w;x,y) &\to \hat{f}(u,v+1,w;x,y) + (\hat{f}(u,v+1,w;x,y) + \hat{f}(u,v+1,w;x,y))\\
\hat{g}(0,w;x,y) &\to \hat{h}(w;x,y)\\
\hat{g}(v+1,w;x,y) &\to \hat{g}(v,w;x,y) * \hat{g}(v,w;x,y)\\
\hat{h}(0;x,y) &\to A(x)\\
\hat{h}(w+1;x,y) &\to B(y_w).\hat{h}(w;x,y)
\end{aligned}$$

est un système Presburger. Par contre, chacun des systèmes suivants est un contre-exemple à la définition 5.2:

- $\hat{f}(u+1,v,w) \to \hat{f}(u,v+1,w) * \hat{f}(u,v,w+1)$ ne correspond pas à la partie droite d'un système primal car $\hat{f}(u,v+1,w)$ et $\hat{f}(u,v,w+1)$ sont des termes différents.
- $\hat{f}(u+1;x) \to F(\hat{g}(u;\hat{f}(u;x)))$ contredit le fait que les symboles définis ne peuvent pas être encapsulés.
- $\{\hat{f}(u+1) \to \hat{g}(u) * \hat{f}(u),\ \ \hat{g}(u+1) \to \hat{f}(u) + \hat{g}(u)\}$ implique la précédence $\hat{f} \succ \hat{g} \succ \hat{f}$ sur les symboles définis, ce qui est impossible, c.-à.-d. la récursion croisée est interdite.

Les systèmes Presburger sont confluents car ils sont orthogonaux et linéaires à gauche. Ils sont nœthériens puisqu'on peut construire un ordre lexicographique $\succ_{lpo}$ pour chaque système. La précédence sur les symboles définis $\mathcal{D}$ peut être étendue aux constructeurs $\mathcal{K}$ de manière à ce que pour chaque symbole défini $\hat{f} \in \mathcal{D}$ et pour chaque constructeur $g \in \mathcal{K}$ on ait $\hat{f} \succ g$. Cette extension et le statut gauche-droit pour tous les symboles définis garantissent la terminaison de chaque système Presburger.

5.1.2 Générateurs et formes pliées

Si toutes les positions compteurs d'un terme primal t sont occupées par des variables compteurs $\mathcal{C}$, autrement dit $\mathcal{CP}os(t) \subseteq \mathcal{VP}os(t)$, alors le terme t est dit être un **générateur**. On dit aussi qu'un générateur est un terme avec *compteurs ouverts*, non encore instanciés avec des entiers naturels.

Notons $\mathcal{N} = \mathcal{N}(\emptyset) = \{s^i(0) \mid i \in N\}$ l'ensemble infini de termes présentant des *entiers naturels*.

Définition 5.4 *Un* **énumérateur** (partiel) *d'un terme primal t est une substitution close $\xi{:}\mathcal{C} \longrightarrow \mathcal{N}$, telle que $\mathcal{D}om(\xi) = \mathcal{CV}ar(t)$ ($\mathcal{D}om(\xi) \subset \mathcal{CV}ar(t)$). Un énumérateur permet d'instancier toutes les variables compteurs d'un terme primal par une expression compteur close, c'est-à-dire par un entier naturel.*

*L'***énumération** $\Xi(t)$ *d'un terme primal t est l'ensemble de tous les énumérateurs possibles de t. L'instance d'un terme primal t, déterminée par un énumérateur ξ, est dite un* terme énuméré.

Un générateur énuméré par un énumérateur partiel est dit un **axiome**.

Création de nouvelles variables

La création systématique de variables nouvelles est un problème difficile de la schématisation d'une suite infinie de termes. Il faut, en effet, trouver un moyen pour concevoir de nouvelles variables qui apparaissent dans les éléments successifs de la suite. Ceci est fait dans les systèmes primaux par le marquage des variables.

Nous avons besoin d'un symbole défini particulier qui sera évalué d'une façon différente de la définition 5.2. Ce symbole $\hat{\imath}$ présente l'indexation d'une variable ordinaire: le terme $\hat{\imath}(m; x)$ avec l'expression compteur m et la variable ordinaire x est interprété comme x_m. L'expression compteur m est la **marque** de la variable *marquée* x_m. Les systèmes Presburger sans variable marquée sont des systèmes Presburger *plats*.

Supposons que l'on puisse réduire le terme t à la position a par la règle $l \to r$ du système Presburger $\mathcal{R}$, et que la partie droite r de cette règle contienne des variables marquées. La substitution σ, qui filtre la partie gauche l de la règle vers le sous-terme $t|_a$, instancie les variables ainsi que leurs marques. Ce mécanisme permet d'engendrer des termes plus riches en variables qu'avec la réduction à la forme normale par la relation de réécriture ordinaire. Les marques ont été introduites pour pouvoir produire de nouvelles variables au cours de chaque étape de réécriture.

Exemple 5.5 Considérons l'unification équationnelle [FH86] dont les symboles sont $\mathcal{F}_0 = \{a, b\}$, $\mathcal{F}_1 = \{g\}$, $\mathcal{F}_2 = \{f\}$ et l'ensemble d'égalités est

$$E \;=\; \left\{ \begin{array}{rcl} f(b,x) & = & x \\ g(f(x,y)) & = & g(y) \end{array} \right\}$$

L'unification $g(x) \stackrel{?}{=}_E g(a)$ possède, comme solution, la suite infinie d'unificateurs

$$[x \mapsto a],\ [x \mapsto f(y_0, a)],\ [x \mapsto f(y_1, f(y_0, a))],\ \ldots,\ [x \mapsto f(y_n, \ldots, f(y_0, a)\ldots)],\ \ldots$$

Cette suite peut être produite à partir de l'axiome $x \mapsto \hat{h}(z; y)$ en utilisant le système Presburger

$$\begin{array}{rcl} \hat{h}(0; y) & \to & a \\ \hat{h}(z+1; y) & \to & f(y_z, \hat{h}(z; y)) \end{array}$$

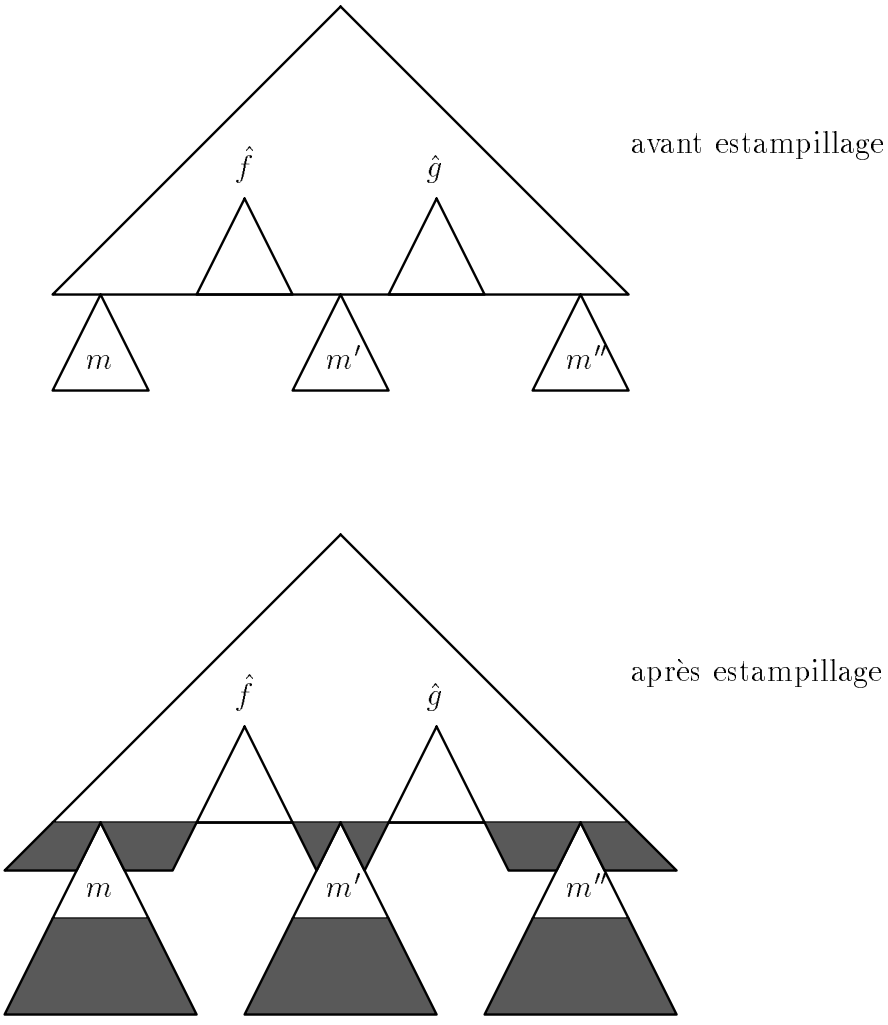


FIG. 5.1 – *Estampillage d'un terme*

à condition que nous sachions renommer à chaque décrémentation du compteur z la variable y_z qui apparaît dans le terme $f(y_z, \cdot)$.

Lors d'une étape de réécriture, ces variables doivent être renommées; ceci est effectué par des *estampilleurs*, ce qui signifie qu'on leur attache une *estampille* identifiée en fonction de la règle $l \to r$ appliquée. Plus précisément, cette estampille est la valeur d'un compteur de $\hat{h}$. Cette relation de réécriture avec des estampilles ajoutées s'appelle la *réécriture estampillée*.

Estampiller un terme t par un estampilleur κ signifie appliquer la substitution κ sur tout le terme t y compris sur les marques des variables à l'exception de la partie du terme t sous

les symboles définis. L'estampillage par κ est noté $t \bullet_{\mathcal{D}} \kappa$ et est formellement défini par

$$\begin{array}{rcll} f(\vec{t}) \bullet_{\mathcal{D}} \kappa & = & f(\vec{t} \bullet_{\mathcal{D}} \kappa) & \text{si } f \notin \mathcal{D}, \\ f(\vec{t}) \bullet_{\mathcal{D}} \kappa & = & f(\vec{t}) & \text{si } f \in \mathcal{D}, \\ x_m \bullet_{\mathcal{D}} \kappa & = & x\kappa & \text{si } x \in \mathcal{D}om(\kappa) \text{ et } x_m \text{ est une variable marquée}, \\ x_m \bullet_{\mathcal{D}} \kappa & = & x_{(m\kappa)} & \text{si } x \notin \mathcal{D}om(\kappa) \text{ et } x_m \text{ est une variable marquée}, \\ x \bullet_{\mathcal{D}} \kappa & = & x\kappa & \text{si } x \text{ est une variable non-marquée}, \end{array}$$

pour chaque vecteur de termes $\vec{t}$. Chaque estampilleur κ est divisé en deux parties, μ et ν. La première est une substitution sur les variables compteurs $\mu{:}\mathcal{C} \longrightarrow \mathcal{N}(\mathcal{C})$ qui instancie les marques. L'autre est une substitution $\nu{:}\mathcal{X} \longrightarrow \mathcal{TP}(\mathcal{K}, \mathcal{D}; \mathcal{X}, \mathcal{C})$ qui instancie des variables ordinaires. En cas de conflit entre μ et ν sur une variable marquée x_m (quand $x \in \mathcal{D}om(\nu)$) seule la substitution ν est appliquée et la marque est perdue. En gros, cela signifie que si ν modifie la variable x alors ν modifie de la même manière toutes les variables marquées x_m qui lui sont sœurs. Cependant un tel conflit signifie qu'un générateur qui utilise cet estampilleur pour filtrer la partie gauche d'une règle du système primal n'est pas adapté aux besoins de la schématisation et doit être évité.

Définition 5.6 (Réécriture estampillée) *Soit $\mathcal{R}$ un système de réécriture Presburger et soient $t, t' \in \mathcal{TP}(\mathcal{K}, \mathcal{D}; \mathcal{X}, \mathcal{C})$ deux termes primaux. La relation $t \Longrightarrow_{\mathcal{R}} t'$ est définie ainsi:*

- *il existe une position $a \in \mathcal{OP}os(t)$, une règle de réécriture $l \to r \in \mathcal{R}$ et un estampilleur κ, tels que $t|_a = l\kappa$; et*
- *$t' = t[r \bullet_{\mathcal{D}} \kappa]_a$*

Notons $t{\Downarrow_{\mathcal{R}}}$ la forme normale de t par rapport à la relation $\Longrightarrow_{\mathcal{R}}$.

Notons cependant que la relation $\Longrightarrow_{\mathcal{R}}$ est équivalente à la relation $\longrightarrow_{\mathcal{R}}$ si $\mathcal{R}$ est un système plat, car $r \bullet_{\mathcal{D}} \kappa = r\kappa$ parce qu'il n'y a pas de variable marquée dans le terme r et l'estampilleur κ devient alors un simple filtre.

Les propositions suivantes déterminent les propriétés de la relation de réécriture estampillée.

Lemme 5.7 *La relation $\Longrightarrow_{\mathcal{R}}$ est nœthérienne pour chaque système Presburger.*

Preuve: Supposons qu'il existe un terme primal t et un système Presburger $\mathcal{R}$, pour lequel il existe une réduction infinie

$$t = t_0 \Longrightarrow_{\mathcal{R}} t_1 \Longrightarrow_{\mathcal{R}} t_2 \Longrightarrow_{\mathcal{R}} \cdots \tag{5.3}$$

Une règle $l \to r \in \mathcal{R}$ est appliquée à la position $a_i \in \mathcal{P}os(t_i)$ lors de l'étape $t_i \Longrightarrow_{P_{\mathcal{H}}} t_{i+1}$. L'ensemble $\{\mathcal{H}ead(t_i|_{a_i}) \mid i = 0, 1, \ldots\}$ des symboles définis apparaissant à la racine des sous-termes $t_i|_{a_i}$ est fini car il existe une précédence $\succ$ sur les symboles définis $\mathcal{D}$. Cela implique qu'à partir d'un terme primal t_k de la réduction infinie (5.3) nous aurons toujours le même

symbole défini $\hat{f} = \mathcal{H}ead(t_i|_{a_i})$ à la racine du terme $t_i|_{a_i}$ pour chaque $i \geq k$. Or ni $r_1^{\hat{f}}$ ni $r_2^{\hat{f}}$ dans la paire de règles

$$\begin{aligned} \hat{f}(0, \vec{x}; \vec{y}) &\rightarrow r_1^{\hat{f}} \\ \hat{f}(z+1, \vec{x}; \vec{y}) &\rightarrow r_2^{\hat{f}}[\hat{f}(z, \vec{x}; \vec{y})]_A \end{aligned} \tag{5.4}$$

de $\mathcal{R}$ ne contiennent de symbole défini $\hat{f}$, ce qui implique nécessairement l'utilisation exclusive de la règle (5.4) dans la réduction (5.3) à partir du terme t_k.

Les sous-termes filtrés vers la partie gauche de la règle (5.4) lors d'une étape de réduction (5.3) sont

$$t_j|_{a_j} = \hat{f}(n_j + 1, \vec{m}; \vec{p})$$

et

$$t_{j+1}|_{a_{j+1}} = \hat{f}(n_j, \vec{m}; \vec{p})$$

pour une expression compteur $n_j \in \mathcal{N}(\mathcal{C})$ pour chaque $j \geq k$. Ceci indique que le premier compteur du symbole $\hat{f}$ décroît: $n_j + 1$ est remplacé par n_j, ce qui ne peut se faire qu'un nombre fini de fois et donc la réduction infinie, définie en (5.3), ne peut pas exister. □

Corollaire 5.8 *La forme normale $t{\Downarrow_{\mathcal{R}}}$ pour chaque terme primal t et chaque système Presburger $\mathcal{R}$ existe et elle est unique.*

Preuve: La relation $\Longrightarrow_{\mathcal{R}}$ est nœthérienne, donc il n'y a pas de réduction infinie. Chaque système Presburger $P_{\mathcal{H}}$ est orthogonal (il n'y a pas de superpositions entre les règles) et linéaire à gauche. Ceci implique qu'il existe uniquement des étapes parallèles dans la réduction selon le lemme de paires critiques. Alors, l'unicité et l'existence de la forme normale sont garanties. □

Proposition 5.9 *Soit $\xi \in \Xi(t)$ un énumérateur d'un terme primal t et $\mathcal{R}$ un système Presburger. La forme normale par rapport au système $\mathcal{R}$ de l'instance $t\xi$ ne contient pas de symbole défini: $t\xi{\Downarrow_{\mathcal{R}}} \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.*

Preuve: Le terme énuméré $t\xi$ ne contient plus de variables compteurs. Chaque symbole défini $\hat{f} \in \mathcal{D}$ qui apparaît dans $t\xi$ est complètement défini par une paire de règles dans $\mathcal{R}$. L'existence de la précédence $\succ$ sur les symboles définis garantit que chaque symbole défini disparaît après un nombre fini d'étapes de réduction de l'instance $t\xi$, ce qui est prouvé par induction structurelle. □

Selon le choix de la marque d'une variable, nous avons des marquages *croissants*, *décroissants* et *stables* des variables dans la relation de réécriture estampillée $\Longrightarrow_{\mathcal{R}}$.

Exemple 5.10 Prenons le terme énuméré $t = a + \hat{f}(3, 2, 0; x)$. Si nous appliquons le système Presburger $\mathcal{R}$, dont les règles sont

$$\begin{aligned} \hat{f}(0, u, v; x) &\rightarrow b \\ \hat{f}(z+1, u, v; x) &\rightarrow x_{z+1} * \hat{f}(z, u+1, v; x) \end{aligned}$$

sur le terme t, nous obtenons $t' = a + (x_3 * \hat{f}(2,3,0;x))$. Si nous changeons la deuxième règle du système Presburger en

$$\hat{f}(z+1,u,v;x) \quad \rightarrow \quad x_u * \hat{f}(z,u+1,v;x)$$

nous obtenons $t' = a + (x_2 * \hat{f}(2,3,0;x))$. Si finalement la deuxième règle est changée en

$$\hat{f}(z+1,u,v;x) \quad \rightarrow \quad x_v * \hat{f}(z,u+1,v;x)$$

nous obtenons $t' = a + (x_0 * \hat{f}(2,3,0;x))$ dans la relation de réécriture indexée $t \Longrightarrow_{\mathcal{D}} t'$. Suivant la forme des règles, la forme normale $t\Downarrow_{\mathcal{R}}$ du terme t sera

$a + (x_3 * (x_2 * (x_1 * b)))$	pour $\hat{f}(z+1,u,v;x) \rightarrow x_{z+1} * \hat{f}(z,u+1,v;x)$	(décroissant),
$a + (x_2 * (x_3 * (x_4 * b)))$	pour $\hat{f}(z+1,u,v;x) \rightarrow x_u * \hat{f}(z,u+1,v;x)$	(croissant),
$a + (x_0 * (x_0 * (x_0 * b)))$	pour $\hat{f}(z+1,u,v;x) \rightarrow x_v * \hat{f}(z,u+1,v;x)$	(stable).

Construction des grammaires primales

Nous pouvons utiliser les axiomes pour schématiser des ensembles infinis de termes de $\mathcal{T}(\mathcal{K},\mathcal{X})$. L'axiome $t^\star$ représente la *forme pliée* de tous les termes d'un ensemble A. Les systèmes Presburger fournissent le moyen de calcul d'un terme $e \in A$ à partir de l'axiome $t^\star$: l'axiome $t^\star$ est d'abord instancié par un énumérateur $\xi \in \Xi(t^\star)$ et cette instance $t^\star\xi$ est réduite à la forme normale $t^\star\xi\Downarrow_{\mathcal{R}} = e$ par rapport au système Presburger $\mathcal{R}$. Suivant ce modèle, il suffit d'avoir une structure qui contient un ensemble de constructeurs $\mathcal{K}$, un ensemble de symboles définis $\mathcal{D}$, un système Presburger $\mathcal{R}$ et un axiome $t^\star$ pour pouvoir schématiser un ensemble infini de termes. Une telle structure est dite une *grammaire primale de termes*.

Définition 5.11 *Une* **grammaire primale de termes** *(ou, plus simplement,* grammaire primale*) G est un quadruplet $(\mathcal{K},\mathcal{D},\mathcal{R},t^\star)$, où $\mathcal{K}$ est l'ensemble des* constructeurs*, $\mathcal{D}$ est l'ensemble des* symboles définis*, $\mathcal{R}$ est un système de réécriture* Presburger *et $t^\star$ est un* axiome*.*

Le langage *produit par une grammaire primale de termes $G = (\mathcal{K},\mathcal{D},\mathcal{R},t^\star)$ est un ensemble de termes $L(G) = \{t^\star\xi\Downarrow_{\mathcal{R}} \mid \xi \in \Xi(t^\star)\}$. L'axiome $t^\star$ s'appelle la* **forme pliée** *de $L(G)$.*

Cette définition reste toujours valable pour les équations et les règles utilisant des axiomes, il suffit d'étendre l'ensemble des constructeurs respectivement à $\mathcal{K} \cup \{=\}$ et $\mathcal{K} \cup \{\rightarrow\}$.

Les atouts principaux des grammaires primales par rapport aux autres schématisations récurrentes sont la possibilité d'exprimer la *croissance exponentielle* des constructeurs dans la suite infinie des termes et la possibilité de *diagonalisation*. Les grammaires primales sont capables de schématiser des ensembles de termes $Q = \{t_i \mid i \in \mathcal{N}\}$ où le nombre de symboles du terme t_n est d'ordre 2^n pour chaque n.

Exemple 5.12 Soit $G = (\{a,+\},\{\hat{f}\},\mathcal{R},t^\star)$ une grammaire primale où

$$\mathcal{R} \quad = \quad \left\{ \begin{array}{rcl} \hat{f}(0) & \rightarrow & a \\ \hat{f}(k+1) & \rightarrow & \hat{f}(k) + \hat{f}(k) \end{array} \right\}$$

et $t^\star = \hat{f}(k)$.

Pour chaque $n \in \mathcal{N}$, $t^\star[k \mapsto n]\Downarrow_{P_\mathcal{H}}$ contient 2^n occurrences du symbole a.

Si $\xi = [k \mapsto 0]$, $t^\star[k \mapsto 0] = \hat{f}(0) \Longrightarrow_\mathcal{R} a$.

Si $\xi = [k \mapsto n+1]$, $t^\star[k \mapsto n+1] = \hat{f}(n+1) \Longrightarrow_\mathcal{R} \hat{f}(n) + \hat{f}(n)$. Par hypothèse, $\hat{f}(n)\Downarrow_\mathcal{R}$ contient 2^n occurrences du symbole a. Par conséquent, $t^\star[k \mapsto n+1]\Downarrow_\mathcal{R} = (\hat{f}(n) + \hat{f}(n))\Downarrow_{P_\mathcal{H}} = \hat{f}(n)\Downarrow_\mathcal{R} + \hat{f}(n)\Downarrow_\mathcal{R}$ contient $2 * 2^n = 2^{n+1}$ occurrences du symbole a.

S'il existe deux ou plusieurs variables compteurs, par exemple k et n, dans un axiome $t^\star$, alors il existe une grammaire primale $G = (\mathcal{K}, \mathcal{D}, \mathcal{R}, t^\star)$ avec un langage $L(G)$ qui ne représente pas toute la "matrice" des termes $t_{i,j} = t^\star[k \mapsto i, n \mapsto j]\Downarrow_\mathcal{R}$ mais uniquement la diagonale $t_{i,i+j} = t^\star[k \mapsto i, n \mapsto j]\Downarrow_\mathcal{R}$. Si on pose $j = 0$, on obtient ainsi la diagonale principale. Cet effet de diagonalisation est obtenu en reliant les compteurs k et n comme des vases communicants.

Exemple 5.13 Soit $G = (\{a, b, c, +\}, \{\hat{f}, \hat{g}\}, \mathcal{R}, t^\star)$ une grammaire primale où

$$\mathcal{R} \quad = \quad \left\{ \begin{array}{rcl} \hat{f}(0,n) & \rightarrow & c \\ \hat{f}(k+1,n) & \rightarrow & \hat{g}(n) + \hat{f}(k,n+1) \\ \hat{g}(0) & \rightarrow & a \\ \hat{g}(n+1) & \rightarrow & b(\hat{g}(n)) \end{array} \right\}$$

et $t^\star = \hat{f}(k,n)$.

Cette grammaire primale schématise le langage

$$L(G) \quad = \quad \{b^j(a) + (b^{j+1}(a) + (\cdots + (b^{i+j}(a) + c)\cdots)) \mid i \in \mathcal{N}, j \in \mathcal{N}\}$$

Supposons l'existence de deux grammaires primales, l'une $G_1 = (\mathcal{K}, \mathcal{D}, \mathcal{R}, t_1^\star)$ et l'autre $G_2 = (\mathcal{K}, \mathcal{D}, \mathcal{R}, t_2^\star)$, qui diffèrent seulement par leur axiome[1]. Nous pouvons alors envisager l'*unification* de deux grammaires primales G_1 et G_2, ce qui revient au problème d'unification des termes $t_1^\star$ et $t_2^\star$ modulo la théorie équationnelle présentée par le système Presburger $\mathcal{R}$.

Avant d'aborder les questions d'unification des grammaires primales (Section 5.2), nous allons étudier le rapport entre les grammaires primales et les autres schématisations récurrentes, puis le rapport entre les grammaires primales et les familles itérées de règles.

5.1.3 Rapport avec les autres schématisations récurrentes

Une question intéressante pour connaître la puissance des grammaires primales est leur rapport avec la classe des ρ-termes, avec la classe des GS-termes et avec la classe des HC-termes. Les grammaires primales présentent la schématisation la plus puissante parmi ces quatre schématisations.

Proposition 5.14 *La classe des ρ-termes est strictement incluse dans la classe des grammaires primales.*

1. N'importe quelle paire de grammaires primales peut être ainsi transformée à condition que les symboles définis communs aux deux grammaires définissent exactement les mêmes fonctions par les mêmes paires de règles dans les deux systèmes Presburger.

Preuve: Chaque terme $t \in \mathcal{T}(\mathcal{K}, \mathcal{X})$ peut être représenté par les règles

$$\begin{array}{rcl} \hat{f}(0; \vec{x}) & \rightarrow & t \\ \hat{f}(n+1; \vec{x}) & \rightarrow & \hat{f}(n; \vec{x}) \end{array}$$

où $\vec{x} = \mathcal{V}ar(t)$.

Si le ρ-terme $\Phi(h[\cdot]_p, n, l)$ est construit à partir du terme $h[l]_p \in \mathcal{T}(\mathcal{K}, \mathcal{X})$, la grammaire primale correspondante va simuler le dépliage de ce ρ-terme par les règles

$$\begin{array}{rcl} \hat{f}(0; \vec{x}) & \rightarrow & l \\ \hat{f}(n+1; \vec{x}) & \rightarrow & h[\hat{f}(n; \vec{x})]_p \end{array}$$

où $\vec{x} = \mathcal{V}ar(h[l]_p)$.

Supposons que le ρ-terme $t = f(\vec{t})$ est construit à partir des ρ-termes $\vec{t}$ dont chaque ρ-terme t_i correspond à la grammaire primale $G_i = (\mathcal{K}, \mathcal{D}, \mathcal{R}, t_i^\star)$. Alors la grammaire primale $G = (\mathcal{K}, \mathcal{D}, \mathcal{R}, t^\star)$ avec l'axiome $t^\star = f(\vec{t^\star})$ correspond au ρ-terme t.

Comme il n'y a pas de traitement de variables défini pour les ρ-termes, il n'y a pas de variable marquée non plus.

Considérons l'ensemble infini de termes $A = \{f^n(g^n(-)) \mid n = 0, 1, \ldots\}$, schématisé par la grammaire primale $G = (\{f, g\}, \{\hat{f}, \hat{g}\}, \mathcal{R}, t^\star)$ où

$$\mathcal{R} \quad = \quad \left\{ \begin{array}{rcl} \hat{f}(0, n) & \rightarrow & \hat{g}(n) \\ \hat{f}(k+1, n) & \rightarrow & f(\hat{f}(k, n+1)) \\ \hat{g}(0) & \rightarrow & - \\ \hat{g}(n+1) & \rightarrow & g(\hat{g}(n)) \end{array} \right\}$$

et $t^\star = \hat{f}(k, 0)$.

Supposons que l'ensemble A peut être schématisé par le ρ-terme $t = \Phi(h[\cdot]_b, n, l)$. Nous pouvons exprimer t aussi par $h^n[l]_{b^n}$. Le contexte $h[\cdot]_b$ doit être construit à partir des symboles f et g, car l ne peut pas contenir lui-même un autre ρ-terme. Posons $h[\cdot]_b = f^i(g^j(\cdot))$. Si $i = 0$ (resp. $j = 0$), le ρ-terme t ne schématise pas les symboles f (resp. g). Si $i \neq 0$ et $j \neq 0$, alors

$$\Phi(h[\cdot]_b, 2, l) \quad = \quad f^i(g^j(f^i(g^j(l)))) \quad \notin \quad A$$

Par conséquent, l'ensemble A ne peut être schématisé par aucun ρ-terme. □

Proposition 5.15 *La classe de HC-termes est strictement incluse dans la classe des grammaires primales.*

Preuve: Chaque terme $t \in \mathcal{T}(\mathcal{K}, \mathcal{X})$ peut être représenté par les règles

$$\begin{array}{rcl} \hat{f}(0; \vec{x}) & \rightarrow & t \\ \hat{f}(n+1; \vec{x}) & \rightarrow & \hat{f}(n; \vec{x}) \end{array}$$

où $\vec{x} = \mathcal{V}ar(t)$.

Supposons que le HC-terme $t = f(\vec{t})$ est construit à partir des HC-termes $\vec{t}$ dont chaque HC-terme t_i correspond à la grammaire primale $G_i = (\mathcal{K}, \mathcal{D}, \mathcal{R}, t_i^\star)$. Alors la grammaire primale $G = (\mathcal{K}, \mathcal{D}, \mathcal{R}, t^\star)$ avec l'axiome $t^\star = f(\vec{t^\star})$ correspond au HC-terme t.

Supposons que le HC-terme $t[t']_p^n$ est construit à partir des HC-termes t et t' où les règles

$$\begin{array}{rcl} \hat{f}(0, \vec{c}; \vec{x}) & \rightarrow & r_1^{\hat{f}} \\ \hat{f}(k+1, \vec{c}; \vec{x}) & \rightarrow & r_2^{\hat{f}}[\hat{f}(k, \vec{c}; \vec{x})]_q \end{array}$$

correspondent au dépliage du terme $t[z]_p$ avec la nouvelle variable $z \notin \mathcal{V}ar(t)$ et

$$\begin{array}{rcl} \hat{g}(0, \vec{c}; \vec{x}) & \rightarrow & r_1^{\hat{g}} \\ \hat{g}(n+1, \vec{c}; \vec{x}) & \rightarrow & r_2^{\hat{g}}[\hat{g}(n, \vec{c}; \vec{x})]_{q'} \end{array}$$

correspondent au dépliage de t'. Les règles

$$\begin{array}{rcl} \hat{h}(0, n, \vec{c}; \vec{x}) & \rightarrow & r_1^{\hat{f}} \bullet_{\mathcal{H}} [z \mapsto \hat{g}(n, \vec{c}; \vec{x})] \\ \hat{h}(k+1, n, \vec{c}; \vec{x}) & \rightarrow & r_2^{\hat{f}} \bullet_{\mathcal{D}} [z \mapsto \hat{g}(n, \vec{c}; \vec{x})][\hat{h}(k, n, \vec{c}; \vec{x})]_q \end{array}$$

avec un nouveau symbole défini $\hat{h}$ et la précédence $\hat{h} \succ \hat{f}$, $\hat{h} \succ \hat{g}$ correspondent au dépliage du HC-terme $t[t']_p^n$.

Considérons l'ensemble infini de termes $Q = \{q_i \mid i \in \mathcal{N}\}$ schématisé par la grammaire primale $G = (\{a, +\}, \{\hat{f}\}, \mathcal{R}, t^\star)$ où

$$\mathcal{R} = \left\{ \begin{array}{rcl} \hat{f}(0) & \rightarrow & a \\ \hat{f}(n+1) & \rightarrow & \hat{f}(n) + \hat{f}(n) \end{array} \right\}$$

et $t^\star = \hat{f}(n)$.

Pour chaque n, l'élément $q_n \in Q$ contient 2^n occurrences du symbole a. Ceci est facilement prouvé par récurrence structurelle sur les règles de $\mathcal{R}$.

Supposons que l'ensemble Q puisse être schématisé par un HC-terme t. Si le HC-terme t' exprime la croissance $p(k)$, alors $t = t'[t'']_p^n$ exprime la croissance $n * p(k)$ des symboles. Or les HC-termes, par construction, ne peuvent exprimer qu'une croissance *polynomiale* des symboles. Par conséquent, il n'y a aucun HC-terme pour schématiser l'ensemble Q. □

Proposition 5.16 *La classe des GS-termes est strictement incluse dans la classe des grammaires primales.*

Preuve: La première partie de la preuve est effectuée par récurrence structurelle sur la construction des GS-termes.

Chaque terme $t \in \mathcal{T}(\mathcal{K}, \mathcal{X})$ peut être représenté par les règles

$$\begin{array}{rcl} \hat{f}(0; \vec{x}) & \rightarrow & t \\ \hat{f}(n+1; \vec{x}) & \rightarrow & \hat{f}(n; \vec{x}) \end{array}$$

où $\vec{x} = \mathcal{V}ar(t)$.

Soit $\Phi(h[x \mapsto \diamond], e, l)$ un GS-terme. Soit D l'ensemble de toutes les occurrences du trou $\diamond$ dans le terme $h[x \mapsto \diamond]$, c.-à.-d. $h[x \mapsto \diamond] = h[\diamond]_D$. Il faut analyser le terme additif de Presburger $e \in \mathcal{PA}_+(\mathcal{C})$ pour construire la grammaire primale correspondante, en se servant du dépliage des GS-termes par *unfold*.

Si $e \in \mathcal{C}$, la grammaire primale correspondante va représenter le dépliage du GS-terme $\Phi(h[v \mapsto \diamond], e, l)$ par les règles

$$\begin{aligned} \hat{f}(0; \vec{x}) &\rightarrow l \\ \hat{f}(n+1; \vec{x}) &\rightarrow h[\hat{f}(n; \vec{x})]_D \end{aligned}$$

où $\vec{x} = \mathcal{V}ar(h[v \mapsto \diamond])$.

Si $e \in \mathcal{N}$ alors

$$\Phi(h[v \mapsto \diamond], e, l) = \diamond[\diamond \mapsto h]^e[\diamond \mapsto l]$$

est un autre GS-terme pour lequel, par hypothèse structurelle, il existe déjà un dépliage par les règles de grammaire primale.

Soit $e = n * e'$ où $n \in \mathcal{N}$ et $e' \in \mathcal{PA}_+(\mathcal{C})$ et supposons que le dépliage du terme $\Phi(h[v \mapsto \diamond], e', l)$ est représenté par les règles [2]

$$\begin{aligned} \hat{f}(0, \vec{c}; \vec{x}) &\rightarrow r_1^{\hat{f}} \\ \hat{f}(k+1, \vec{c}; \vec{x}) &\rightarrow r_2^{\hat{f}}[\hat{f}(k, , \vec{c}; \vec{x})]_D \end{aligned}$$

Par hypothèse structurelle, il existe un terme primal $\bar{h}$, correspondant au GS-terme h, pour lequel il existe déjà un dépliage par les règles de grammaire primale. Le dépliage du GS-terme $\Phi(h[v \mapsto \diamond], n * e', l)$ est égal au dépliage du GS-terme $\Phi(\bar{h}[v \mapsto \diamond], n * e', l)$. Il est alors représenté par les règles

$$\begin{aligned} \hat{f}_1(0, c, \vec{c}; \vec{x}) &\rightarrow \diamond[\diamond \mapsto \bar{h}]^n[\diamond \mapsto \hat{f}(c, \vec{c}; \vec{x})] \\ \hat{f}_1(k+1, c, \vec{c}; \vec{x}) &\rightarrow \hat{f}_1(k, c, \vec{c}; \vec{x}) \end{aligned}$$

avec un nouveau symbole défini $\hat{f}_1$ et la précédence $\hat{f}_1 \succ \hat{f}$.

Soit $e = e' + e''$ où $e', e'' \in \mathcal{PA}_+(\mathcal{C})$. Supposons que le dépliage du GS-terme $\Phi(h[v \mapsto \diamond], e', l)$ est représenté par les règles

$$\begin{aligned} \hat{f}(0, \vec{c}; \vec{x}) &\rightarrow r_1^{\hat{f}} \\ \hat{f}(k+1, \vec{c}; \vec{x}) &\rightarrow r_2^{\hat{f}}[\hat{f}(k, \vec{c}; \vec{x})]_D \end{aligned}$$

et que le dépliage du GS-terme $\Phi(h[v \mapsto \diamond], e'', l)$ est représenté par les règles

$$\begin{aligned} \hat{g}(0, \vec{c}; \vec{x}) &\rightarrow r_1^{\hat{g}} \\ \hat{g}(n+1, \vec{c}; \vec{x}) &\rightarrow r_2^{\hat{g}}[\hat{g}(n, \vec{c}; \vec{x})]_{D'} \end{aligned}$$

2. La substitution δ n'est pas utilisée car les GS-termes ne permettent pas de lier les compteurs comme des vases communicants.

Le dépliage du GS-terme $\Phi(h[v \mapsto \diamond], e' + e'', l)$ est alors représenté par les règles

$$\begin{aligned} \hat{h}(0, n, \vec{c}; \vec{x}) &\rightarrow \hat{g}(n, \vec{c}; \vec{x}) \\ \hat{h}(k+1, n, \vec{c}; \vec{x}) &\rightarrow r_2^{\hat{f}}[\hat{h}(k, n, \vec{c}; \vec{x})]_D \end{aligned}$$

avec le nouveau symbole $\hat{h}$ et la précédence $\hat{h} \succ \hat{f}$, $\hat{h} \succ \hat{g}$.

Supposons que le GS-terme $t[x \mapsto t']$ est construit à partir des GS-termes t et t' où les règles

$$\begin{aligned} \hat{f}(0, \vec{c}; \vec{x}) &\rightarrow r_1^{\hat{f}} \\ \hat{f}(k+1, \vec{c}; \vec{x}) &\rightarrow r_2^{\hat{f}}[\hat{f}(k, \vec{c}; \vec{x})]_D \end{aligned}$$

correspondent au dépliage de t et

$$\begin{aligned} \hat{g}(0, \vec{c}; \vec{x}) &\rightarrow r_1^{\hat{g}} \\ \hat{g}(n+1, \vec{c}; \vec{x}) &\rightarrow r_2^{\hat{g}}[\hat{g}(n, \vec{c}; \vec{x})]_{D'} \end{aligned}$$

correspondent au dépliage de t'. Les règles

$$\begin{aligned} \hat{h}(0, n, \vec{c}; \vec{x}) &\rightarrow r_1^{\hat{f}} \bullet_{\mathcal{D}} [x \mapsto \hat{g}(n, \vec{c}; \vec{x})] \\ \hat{h}(k+1, n, \vec{c}; \vec{x}) &\rightarrow r_2^{\hat{f}} \bullet_{\mathcal{D}} [x \mapsto \hat{g}(n, \vec{c}; \vec{x})][\hat{h}(k, n, \vec{c}; \vec{x})]_D \end{aligned}$$

avec un nouveau symbole défini $\hat{h}$ et la précédence $\hat{h} \succ \hat{f}$, $\hat{h} \succ \hat{g}$ correspondent au dépliage du GS-terme $t[x \mapsto t']$.

Pour l'inclusion stricte, considérons l'ensemble infini de termes

$$L(G) = \{a + (b(a) + (\cdots + (b^i(a) + c)\cdots)) \mid i \in \mathcal{N}\}$$

schématisé par la grammaire primale $G = (\{a, b, c, +\}, \{\hat{f}, \hat{g}\}, \mathcal{R}, t^\star)$ où

$$\mathcal{R} = \left\{ \begin{array}{rcl@{\qquad}rcl} \hat{f}(0, n) & \rightarrow & c & \hat{g}(0) & \rightarrow & a \\ \hat{f}(k+1, n) & \rightarrow & \hat{g}(n) + \hat{f}(k, n+1) & \hat{g}(n+1) & \rightarrow & b(\hat{g}(n)) \end{array} \right\}$$

et $t^\star = \hat{f}(k, 0)$.

Le terme $t^\star[k \mapsto s^i(0)]\Downarrow$ contient i occurrences du symbole $+$ ainsi que le sous-terme $b^i(a)$, donc pour schématiser $L(G)$ il faudrait un GS-terme du type

$$\Phi(\Phi(b(\diamond), k, a) + \diamond, n, c).$$

Or ce GS-terme schématise l'ensemble

$$Q = \{b^k(a) + (b^k(a) + (\cdots + (b^k(a) + c)\cdots))\}$$

car il n'existe aucun moyen de faire interagir les deux expressions de Presburger k et n. □

5.1.4 Grammaires primales pour des familles itérées

Soit $\mathcal{I}(S)$ une famille itérée de règles produite par complétion à partir d'un système croisé S. Nous montrerons comment produire une grammaire primale G telle que $L(G) = \mathcal{I}(S)$. L'utilité des compteurs intervenant dans des grammaires primales $G = (\mathcal{K}, \mathcal{D}, \mathcal{R}, t^\star)$ est mise en évidence dans les exemples suivants. Les compteurs secondaires, initialisés à zéro dans l'axiome $t^\star$, interconnectent les symboles définis du système Presburger $\mathcal{R}$. Le compteur principal sert d'indice des éléments de $\mathcal{I}(S)$. Plus précisément, l'instanciation du compteur principal de l'axiome $t^\star$ par n, suivie par la réduction à la forme normale par rapport à la relation de réécriture $\Longrightarrow_{\mathcal{R}}$, aboutit au n-ième élément de la famille itérée $\mathcal{I}(S)$.

Avant de présenter le théorème reliant les grammaires primales et les systèmes croisés, considérons quelques exemples qui expliquent les constructions qui seront utilisées.

Exemple 5.17 (Suite de l'exemple 2.39, deuxième partie)
Les règles de la famille itérée sont de la forme

$$d(g^n(x \oplus (x \oslash (g(x) \oslash \ldots (g^n(x) \otimes y))))) \quad \rightarrow \quad y \tag{5.5}$$

Nous avons $t_2|_b = x \oplus (x \oslash y)$ et $t_2[\cdot]_b = g(\cdot)$. Le premier radical répétitif est le contexte $t_2[\cdot]_b$. Son itération est assurée par le symbole défini $\hat{f}$. Cette itération de radical finit avec la production d'une instance de $t_2|_b$. Comme $\mathcal{D}om(\varphi_2) = \{y\}$, cette instance est produite à partir d'un radical répétitif au lieu de la variable y. Ce radical subordonné est produit à partir du symbole défini $\hat{f}_y$. Il y a autant d'itérations du radical subordonné que d'itérations du radical $t[\cdot]_b$. Ceci indique que le compteur principal de $\hat{f}_y$ doit être l'un des compteurs secondaires de $\hat{f}$. De plus, le compteur principal de $\hat{f}$ et le compteur principal de $\hat{f}_y$ doivent être reliés comme des vases communicants, à condition que le compteur principal de $\hat{f}_y$ soit instancié par 0 dans l'axiome. Ces conditions nous donnent la première partie du système Presburger:

$$\begin{aligned} \hat{f}(0, z_y, z_x; x, y) &\rightarrow x \oplus (x \oslash \hat{f}_y(z_y, z_x; x, y)) \\ \hat{f}(z+1, z_y, z_x; x, y) &\rightarrow g(\hat{f}(z, z_y + 1, z_x; x, y)) \end{aligned}$$

La forme pliée de la famille itérée est l'axiome $d(g(\hat{f}(z, 0, 0; x, y))) \rightarrow y$.

Le radical représenté par le symbole défini $\hat{f}_y$, encadrant les instances itérées de la variable y, est construit à partir de la substitution $\varphi_2 \triangle T_n(\sigma_2, \varphi_2, \varphi_1)$. En effet, il faut construire des règles du système primal qui assurent la production de l'opérateur $T_n(\sigma_2, \varphi_2, \varphi_1)$ pour chaque n à partir de la forme pliée $\hat{f}_y(n, 0; x, y)$. D'après la construction récurrente d'opérateur T, la substitution σ_2 est à l'origine de la partie droite de la règle de base, tandis que la substitution φ_2 fournit le corps de la partie droite de la règle de successeur. Cependant à chaque étape d'itération de φ_2, ainsi qu'à la fin sur σ_2, intervient aussi la substitution φ_1. En effet, si le niveau d'itération dans le dépliage d'opérateur T est k, il faut appliquer k-fois la substitution φ_1. Comme $\mathcal{D}om(\varphi_1) \cap (\mathcal{VR}an(\sigma_2) \cup \mathcal{VR}an(\varphi_2)) = \{x\}$, l'application itérée de la substitution φ_1 est produite à partir d'un radical répétitif au lieu de la variable x dans les corps des substitutions σ_2 et φ_2. Ce radical, à la fois secondaire et subordonné, est produit à partir du symbole défini $\hat{f}_x$. Les itérations de la substitution φ_2 sont assurées par le symbole

défini $\hat{f}_y$ car $\mathcal{D}om(\varphi_2) \cap \mathcal{VR}an(\varphi_2) = \{y\}$. Il y a autant d'itérations du radical secondaire ou subordonné que d'itérations du radical à partir des corps de φ et σ_2. Ceci indique que le compteur principal de $\hat{f}_x$ doit être l'un des compteurs secondaires de $\hat{f}_y$ et, par transitivité, aussi l'un des compteurs secondaires de $\hat{f}$. De plus, le compteur principal de $\hat{f}_y$ et le compteur principal de $\hat{f}_x$ doivent, eux aussi, être reliés comme des vases communicants. Ces conditions nous donnent la deuxième partie du système Presburger:

$$\begin{aligned} \hat{f}_y(0, z_x; x, y) &\rightarrow g(\hat{f}_x(z_x; x)) \otimes y \\ \hat{f}_y(z+1, z_x; x, y) &\rightarrow g(\hat{f}_x(z_x; x)) \oslash \hat{f}_y(z, z_x + 1; x, y) \end{aligned}$$

produite à partir des substitutions φ_2 et σ_2.

Le radical représenté par le symbole défini $\hat{f}_x$, encadrant les instances itérées de la variable x, est construit à partir de la substitution $\varphi_1 \triangle T_n(\sigma_2, \varphi_2, \varphi_1)$. Or dans notre exemple nous avons $\mathcal{VR}an(\varphi_1) \cap (\mathcal{D}om(\sigma_1) \cup \mathcal{D}om(\varphi_2)) = \emptyset$, seule la substitution φ_1 intervient dans l'itération. La substitution φ_1 fournit le corps de la partie droite de la règle de successeur pour $\hat{f}_x$, tandis que la partie droite de la règle de base contient uniquement la variable x, car $x \notin \mathcal{D}om(\sigma_1)$. Ces conditions nous donnent la troisième partie du système Presburger:

$$\begin{aligned} \hat{f}_x(0; x) &\rightarrow x \\ \hat{f}_x(z+1; x) &\rightarrow g(\hat{f}_x(z; x)) \end{aligned}$$

La condition $\mathcal{V}ar(\varphi_1) \cap (\mathcal{D}om(\varphi_2) \cup \mathcal{D}om(\sigma_2)) = \emptyset$ implique l'impossibilité de produire des itérations sur la variable y à partir du symbole défini $\hat{f}_x$ et, par conséquent, assure la validité de la précédence $\hat{f}_y \succ \hat{f}_x$.

La nécessité d'introduire des variables marquées est illustrée dans l'exemple suivant.

Exemple 5.18 La preuve par récurrence du théorème $rev(rev(x)) = x$ où rev est défini par le système

$$\begin{aligned} rev_1(nil, y) &\rightarrow nil \\ rev_1(xa.xb, y) &\rightarrow rev_1(xb, xa.y) \\ rev(x) &\rightarrow rev_1(x, nil) \end{aligned}$$

entraîne la divergence et produit la famille itérée

$$\begin{aligned} rev_1(rev_1(xb, xa.nil), nil) &\rightarrow xa.xb \\ rev_1(rev_1(xb, xa_1.(xa.nil)), nil) &\rightarrow xa.(xa_1.xb) \\ rev_1(rev_1(xb, xa_2.(xa_1.(xa.nil))), nil) &\rightarrow xa.(xa_1.(xa_2.xb)) \\ &\vdots \end{aligned}$$

résultant du système croisé en avant

$$\begin{aligned} rev_1(rev_1(x, nil), nil) &\rightarrow x \\ rev_1(xa.xb, y) &\rightarrow rev_1(xb, xa.y) \end{aligned}$$

où $a = 1$, $b = \Lambda$, $\sigma_1 = [x \mapsto xa.xb]$, $\sigma_2 = [y \mapsto nil]$, $\varphi_1 = [xb \mapsto xa.xb]$, $\varphi_2 = [y \mapsto xa.y]$.

Au lieu de paraphraser le raisonnement de l'exemple précédent, nous nous focalisons uniquement sur les principales différences.

Il faut noter que $t_2[\cdot]_b = \cdot$ est un contexte vide et par conséquent la deuxième règle associée au symbole défini $\hat{f}$ n'itère aucun radical. Comme $\mathcal{D}om(\varphi_1) \cap \mathcal{VR}an(\varphi_1) = \{xb\}$ et $\mathcal{D}om(\varphi_2) \cap \mathcal{VR}an(\varphi_2) = \{y\}$, il y a des radicaux à itérer: sur la variable xb par le symbole $\hat{f}_{xb}$ et sur la variable y par $\hat{f}_y$. Cette fois-ci nous n'avons pas de lien entre $\hat{f}_y$ et $\hat{f}_{xb}$ car $\mathcal{D}om(\varphi_1) \cap (\mathcal{V}ar(\varphi_2) \cup \mathcal{V}ar(\sigma_2)) = \emptyset$ et $\mathcal{V}ar(\varphi_1) \cap (\mathcal{D}om(\varphi_2) \cup \mathcal{D}om(\sigma_2)) = \emptyset$.

La différence principale par rapport à l'exemple précédent est la nécessité de maîtriser le renommage de la variable xa en utilisant les marques. La nécessité d'ajouter des marques à la variable xa dans les règles pour $\hat{f}_y$ et $\hat{f}_{xb}$ se manifeste, entre autre, par

$$(\mathcal{VR}an(\varphi_1) - \mathcal{D}om(\varphi_1)) \cap (\mathcal{VR}an(\varphi_2) - \mathcal{D}om(\varphi_2)) \quad = \quad \{xa\}$$

La présence de la variable xa dans l'intersection précédente est aussi la raison de son renommage dans la famille itérée.

La marque de la variable xa dans les règles pour le symbole $\hat{f}_y$ varie avec la profondeur d'itération: l'indice ajouté à la variable xa diminue avec la profondeur du niveau d'itération de $\hat{f}_y$. Par contre, la marque de la variable xa dans les règles pour le symbole $\hat{f}_{xb}$ varie en sens inverse de la profondeur d'itération: l'indice ajouté à la variable xa augmente avec la profondeur du niveau d'itération de $\hat{f}_{xb}$. Ceci est dû à la construction de $T_n(\sigma_2, \varphi_2, \varphi_1)$. Par conséquent, la marque de la variable xa dans les règles pour $\hat{f}_y$ est construite à partir du compteur principal de $\hat{f}_y$, tandis que la marque de la variable xa dans les règles pour $\hat{f}_{xb}$ est construite à partir d'un compteur secondaire de $\hat{f}_{xb}$ qui est lui-même relié au compteur principal de $\hat{f}_{xb}$ par le principe des vases communicants.

Le système Presburger résultant sera

$$\begin{aligned}
\hat{f}(0, z_y, z_a, z_b; y, xa, xb) &\rightarrow rev_1(xb, xa_{z_y}.\hat{f}_y(z_y, z_a; y, xa)) \\
\hat{f}(z+1, z_y, z_a, z_b; y, xa, xb) &\rightarrow \hat{f}(z, z_y+1, z_a, z_b; xa, xb) \\
\hat{f}_y(0, z_a; y, xa) &\rightarrow nil \\
\hat{f}_y(z_y+1, z_a; y, xa) &\rightarrow xa_{z_y}.\hat{f}_y(z_y, z_a; y, xa) \\
\hat{f}_{xb}(0, z_a; xa, xb) &\rightarrow xb \\
\hat{f}_{xb}(z_b+1, z_a; xa, xb) &\rightarrow xa_{z_a+1}.\hat{f}_{xb}(z_b, z_a+1; xa, xb)
\end{aligned}$$

et l'axiome est $rev_1(\hat{f}(v, 0, 0, 0; y, xa, xb)) \rightarrow xa.\hat{f}_{xb}(v, 0; xa, xb)$.

Une situation similaire est observée dans le cas de systèmes croisés en arrière.

Exemple 5.19 (Suite de l'exemple 2.51)
Les règles de la famille itérée sont de la forme

$$((((x \oslash f^{n+1}(y)) \oplus f^n(y)) \oplus \ldots \oplus f(y)) \oplus y) \otimes y \quad \rightarrow \quad ((x \oplus f^n(y))) \oplus \ldots \oplus f(y)) \oplus y$$

Nous avons $t_2\sigma_2 = x$ et $s_1[\cdot]_b = (\cdot \oplus y)$. Le premier radical répétitif est le contexte $s_1[\cdot]_b$. Son itération est assurée par le symbole $\hat{g}$. Cette itération finit avec la production de $t_2\sigma_2$.

Car $\mathcal{D}om(\varphi_1) = \{y\}$ et la variable y est présente dans le radical $s_1[\cdot]_b$, un radical secondaire à partir du symbole défini $\hat{g}_y$ est itéré au lieu de la variable y dans le contexte $s_1[\cdot]_b$. Ceci indique que le compteur principal de $\hat{g}_y$ doit être l'un des compteurs secondaires de $\hat{g}$. Le niveau d'itération du contexte $s_1[\cdot]_b$ varie en sens inverse du nombre d'itérations requises pour le radical secondaire: le nombre d'itérations du contexte secondaire à partir de $\hat{g}_y$ augmente avec la profondeur du niveau d'itération de $\hat{g}$, du fait de l'itération de $\varphi_1 \triangle T_n(\sigma_1, \varphi_2, \varphi_1)$ sur la variable y. Par conséquent, le compteur principal de $\hat{g}$ et le compteur principal de $\hat{g}_y$ doivent être reliés comme des vases communicants, à condition que le compteur principal de $\hat{g}_y$ soit instancié par 0 dans l'axiome. Ces conditions nous donnent la première partie du système Presburger:

$$\begin{aligned} \hat{g}(0, z_y, z_x; x, y) &\rightarrow x \\ \hat{g}(z+1, z_y, z_x; x, y) &\rightarrow \hat{g}(z, z_y + 1, z_x; x, y) \oplus \hat{g}_y(z_y; y) \end{aligned}$$

En utilisant les règles précédentes pour $\hat{g}$, nous pouvons présenter un produit intermédiaire d'axiomes à partir de la famille itérée, en schématisant les parties droites:

$$((((x \oslash f^{n+1}(y)) \oplus f^n(y)) \oplus \ldots \oplus f(y)) \oplus y) \otimes y \rightarrow \hat{g}(s^n(0), 0, 0, x \oplus f^n(y), y)$$

Le symbole défini $\hat{f}$ intervient dans la partie *gauche* de l'axiome de l'exemple 5.17, grâce à la LR-persistance. Or la RL-persistance est appliquée sur les systèmes croisés en arrière, ce qui entraîne l'application du symbole $\hat{g}$ pour la schématisation dans la partie *droite* de l'axiome.

Les règles pour le symbole défini $\hat{g}_x$, encadrant les instances itérées de la variable $x \in \mathcal{D}om(\varphi_2) \cup \mathcal{D}om(\sigma_1)$, sont construites à partir des substitutions σ_1 et $\varphi_2 \triangle T_n(\sigma_1, \varphi_2, \varphi_1)$. La substitution σ_1 est à l'origine de la partie droite de la règle de base, tandis que la substitution φ_2 fournit le corps de la partie droite de la règle inductive car $\mathcal{D}om(\varphi_2) \cap \mathcal{VR}an(\varphi_2) = \{x\}$. Cependant à chaque étape d'itération de φ_2 dans l'opérateur T, ainsi qu'à la fin sur σ_1, intervient aussi la substitution φ_1. Comme $\mathcal{D}om(\varphi_1) \cap (\mathcal{VR}an(\sigma_1) \cup \mathcal{VR}an(\varphi_2)) = \{y\}$, l'itération de la substitution φ_1 s'applique sur la variable y dans le corps des substitutions σ_1 et φ_2. Le radical, à la fois secondaire et subordonné, est produit à partir du symbole défini $\hat{g}_y$. Le niveau d'itération de la substitution φ_2 varie en sens inverse du nombre d'itérations requises pour le radical à partir de $\hat{g}_y$: le nombre d'itérations à partir de $\hat{g}_y$ augmente avec la profondeur du niveau d'itération de $\hat{g}_x$. Ceci est dû à la construction de l'opérateur $T_n(\sigma_1, \varphi_2, \varphi_1)$. Ces conditions nous donnent la deuxième partie du système Presburger:

$$\begin{aligned} \hat{g}_x(0, z_y; x, y) &\rightarrow x \oslash f(\hat{g}_y(z_y; y)) \\ \hat{g}_x(z+1, z_y; x, y) &\rightarrow \hat{g}_x(z, z_y + 1; x, y) \oplus f(\hat{g}_y(z_y; y)) \end{aligned}$$

Les règles pour le symbole défini $\hat{g}_y$, encadrant les instances itérées de la variable $y \in \mathcal{D}om(\varphi_1)$, sont construites à partir de la substitution $\varphi_1 \triangle T_n(\sigma_1, \varphi_2, \varphi_1)$. Or dans notre exemple nous avons $\mathcal{V}ar(\varphi_1) \cap (\mathcal{D}om(\sigma_1) \cup \mathcal{D}om(\varphi_2)) = \emptyset$, seule la substitution φ_1 intervient dans l'itération. La substitution φ_1 fournit le corps de la partie droite de la règle de successeur pour $\hat{g}_y$, tandis que la partie droite de la règle de base contient uniquement la variable y, car $y \notin \mathcal{D}om(\sigma_2)$. Ces conditions nous donnent la troisième partie du système Presburger:

$$\begin{aligned} \hat{g}_y(0; y) &\rightarrow y \\ \hat{g}_y(z+1; y) &\rightarrow f(\hat{g}_y(z; y)) \end{aligned}$$

La condition $\mathcal{V}ar(\sigma_1) \cap (\mathcal{D}om(\sigma_1) \cup \mathcal{D}om(\varphi_2)) = \emptyset$ implique l'impossibilité de la production des itérations sur la variable x à partir du symbole $\hat{g}_y$ et, par conséquent, assure la validité de la précédence $\hat{g}_x \succ \hat{g}_y$.

Après avoir considéré les règles de réécriture précédentes pour $\hat{g}_x$ et $\hat{g}_y$, la famille itérée dans cet exemple peut être produite à partir de l'axiome

$$(\hat{g}_x(z, 0; x, y) \oplus y) \otimes y \quad \rightarrow \quad \hat{g}(z, 0; x \oplus \hat{g}_y(z; y), y)$$

Cet axiome est tiré de la fermeture $s_1 \rightarrowtail t_1$ orientée dans le sens inverse, où le contexte $s_1[\cdot]_b$ est remplacé par le symbole $\hat{g}$, la variable x dans t_1 est remplacée par le symbole $\hat{g}_x$ car $x \in \mathcal{D}om(\varphi_2)$ et le sous-terme $s_1|_b$ est remplacé par l'itération de $x \oplus \hat{g}_y(z; y)$ tirée de $\varphi_2 \triangle T_n(\sigma_1, \varphi_2, \varphi_1)$.

Théorème 5.20 *Pour chaque famille itérée $\mathcal{I}(S)$, produite à partir d'un système croisé S, il existe une grammaire primale $G = (\mathcal{K}, \mathcal{D}, \mathcal{R}, t^\star)$ telle que $L(G) = \mathcal{I}(S)$.*

Preuve: Tout d'abord, nous introduisons quelques notations:

$$\begin{array}{rclcrcl}
\vec{w}_f & = & \mathcal{V}ar(t_2) & & \vec{w}_b & = & \mathcal{V}ar(s_1) \\
\vec{x}_1 & = & \mathcal{D}om(\varphi_1) & & \vec{y}_1 & = & \mathcal{VR}an(\varphi_1) \\
\vec{x}_2 & = & \mathcal{D}om(\varphi_2) & & \vec{y}_2 & = & \mathcal{VR}an(\varphi_2) \\
\vec{x}_{12} & = & \vec{x}_1 \cup \vec{x}_2 & & \vec{y}_{12} & = & \vec{y}_1 \cup \vec{y}_2 \\
\vec{c}_1 & = & \{z_x \in \mathcal{X} \mid x \in \vec{x}_1\} & & \alpha_1(\vec{u}; \vec{v}) & = & [x \mapsto \hat{f}_x(\vec{u}; \vec{v}) \mid x \in \vec{x}_1] \\
\vec{c}_2 & = & \{z_x \in \mathcal{X} \mid x \in \vec{x}_2\} & & \alpha_2(\vec{u}; \vec{v}) & = & [x \mapsto \hat{f}_x(\vec{u}; \vec{v}) \mid x \in \vec{x}_2] \\
\vec{c}_{12} & = & \{z_x \in \mathcal{X} \mid x \in \vec{x}_{12}\} & & \alpha_{12}(\vec{u}; \vec{v}) & = & [x \mapsto \hat{f}_x(\vec{u}; \vec{v}) \mid x \in \vec{x}_{12}] \\
\vec{d}_1 & = & \{z_x \in \mathcal{X} \mid x \in \vec{y}_1\} & & \gamma_1 & = & [z_x \mapsto z_x + 1 \mid z_x \in \vec{c}_1] \\
\vec{d}_2 & = & \{z_x \in \mathcal{X} \mid x \in \vec{y}_2\} & & \gamma_2 & = & [z_x \mapsto z_x + 1 \mid z_x \in \vec{c}_2] \\
\vec{d}_{12} & = & \{z_x \in \mathcal{X} \mid x \in \vec{y}_{12}\} & & \gamma_{12} & = & [z_x \mapsto z_x + 1 \mid z_x \in \vec{c}_{12}] \\
\epsilon_1 & = & [z_x \mapsto z_x + 1 \mid x \in \vec{y}_1 - \vec{x}_1] & & \epsilon_2 & = & [z_x \mapsto z_x + 1 \mid x \in \vec{y}_2 - \vec{x}_2] \\
q_1 & = & (\vec{d}_1 - \vec{c}_1)\epsilon_1 & & q_2 & = & (\vec{d}_2 - \vec{c}_2)\gamma \\
V & = & \mathcal{V}ar(t_2[\cdot]_b) - \mathcal{V}ar(t_2|_b) & & W & = & (\vec{y}_1 - \vec{x}_1) \cap (\vec{y}_2 - \vec{x}_2)
\end{array}$$

Les $\vec{w}$, $\vec{x}$ et $\vec{y}$ sont des variables ordinaires, les $\vec{c}$ et $\vec{d}$ sont des compteurs, les α sont des substitutions paramétrées particulières introduisant les symboles de support $\hat{f}_x$ et les γ sont des substitutions incrémentant les compteurs. L'expression Θ_x signifie soit Θ_1 si $x \in \mathcal{D}om(\varphi_1)$ soit Θ_2 si $x \in \mathcal{D}om(\varphi_2)$, où Θ représente un des symboles $\vec{x}$, $\vec{y}$, $\vec{c}$, $\vec{d}$, α et γ. Toutes les variables sont considérées comme globales, par exemple $\vec{c}_1 \cap \vec{d}_2 = \{z_x \in \mathcal{X} \mid x \in \vec{x}_1 \cap \vec{y}_2\}$.

De plus, soit $\tau_{\mathcal{V}}(v) = [u \mapsto u_v \mid u \in \mathcal{V}]$ la substitution qui ajoute la marque v aux variables $\mathcal{V}$.

Supposons que $S = \{s_1 \rightarrow t_1, s_2 \rightarrowtail t_2\}$ est le système croisé en avant de la définition 2.29.

L'ensemble $\mathcal{D}$ contient le symbole principal $\hat{f}$ pour l'itération du contexte $t_2[\cdot]_b$ en ajoutant le sous-terme $t_2|_b$ à la fin, ainsi que les symboles de support $\hat{f}_x$ pour l'itération des expressions $\varphi_1 \triangle T_n(\sigma_2, \varphi_2, \varphi_1)$ et $\varphi_2 \triangle T_n(\sigma_2, \varphi_2, \varphi_1)$ sur chaque variable $x \in \vec{w}_f$. Le système Presburger $\mathcal{R}$ contient les règles de réécriture

$$\begin{array}{rcl}
\hat{f}(0, \vec{d}_{12}; \vec{w}_f) & \rightarrow & t_2|_b \alpha_2(\vec{d}_2; \vec{y}_2) \bullet_{\mathcal{D}} \tau_{V \cup W}(\vec{c}_2 \cap \vec{d}_{12}) \\
\hat{f}(z+1, \vec{d}_{12}; \vec{w}_f) & \rightarrow & t_2 \alpha_1(\vec{d}_1; \vec{y}_1)[\hat{f}(z, \vec{d}_{12}\gamma_2; \vec{w}_f)]_b \bullet_{\mathcal{D}} \tau_{V \cup W}(\vec{c}_2 \cap \vec{d}_{12})
\end{array}$$

pour le symbole principal $\hat{f}$ et les règles de réécriture

$$\begin{aligned}\hat{f}_x(0,\vec{d}_x-\{z_x\};\vec{y}_x) &\rightarrow x(\sigma_2 \triangle \alpha_1(\vec{d}_1\epsilon_1;\vec{y}_1)) \bullet_{\mathcal{D}} \tau_{V\cup W}(q_x)\\ \hat{f}_x(z+1,\vec{d}_x-\{z_x\};\vec{y}_x) &\rightarrow x(((\varphi_1\cup\varphi_2)\triangle \alpha_1(\vec{d}_1\epsilon_1;\vec{y}_1)) \triangle \alpha_2(\vec{d}_2\gamma_1;\vec{y}_2)) \bullet_{\mathcal{D}} \tau_{V\cup W}(q_x)\end{aligned}$$

pour chaque variable $x \in \vec{x}_{12}$ et, par conséquent, aussi pour chaque symbole de support $\hat{f}_x$. L'union $\varphi_1 \cup \varphi_2$ est une substitution car $\mathcal{D}om(\varphi_1) \cap \mathcal{D}om(\varphi_2) = \emptyset$ d'après la définition 2.29.

L'axiome t est la règle

$$s_1\sigma_1\alpha_1(z,\vec{0};\vec{y}_1)[t_2\sigma_2\alpha_2(z,\vec{0};\vec{y}_2)]_a[\hat{f}(z,\vec{0};\vec{w}_f)]_{ab} \rightarrow t_1\sigma_1\alpha_1(z,\vec{0};\vec{y}_1)$$

On prouve ensuite, par induction sur n, que $t[z \mapsto s^n(0)]\Downarrow_{\mathcal{R}}$ est le n-ième élément de $\mathcal{I}(S)$.

La preuve pour les systèmes croisés en arrière est très similaire. □

Puissance calculatoire de la divergence

Andrea Sattler-Klein a utilisé la divergence comme un outil de production de nouvelles règles de réécriture. Elle a démontré ainsi qu'il est possible de coder par des systèmes divergents toutes les fonctions primitives récursives [SK91]. En généralisant ces résultats, elle a pu produire par la divergence avec interréduction des ensembles de règles qui ne sont même pas récursivement énumérables [SK92].

La construction de la grammaire primale permet de prouver le résultat principal de [SK91] par d'autres moyens. Andrea Sattler-Klein a prouvé, en utilisant les systèmes de réécriture de mots, qu'on peut simuler le calcul de toute fonction primitive récursive par un système divergent. Ce résultat est, dans un certain sens, une réciproque du théorème 5.20.

5.2 Unification des grammaires primales

L'unification de deux grammaires primales $G_1 = (\mathcal{K},\mathcal{D},\mathcal{R},t_1^*)$ et $G_2 = (\mathcal{K},\mathcal{D},\mathcal{R},t_2^*)$ correspond à l'unification de deux axiomes $t_1^\star$ et $t_2^\star$ modulo la théorie équationnelle présentée par le système Presburger $\mathcal{R}$. Néanmoins, il existe une sémantique plus intéressante pour un problème d'unification de deux termes primaux. Un problème d'unification de termes primaux modulo un système Presburger peut être interprété comme un problème d'unification syntaxique de deux ensembles infinis de termes, où chaque ensemble infini est schématisé par une grammaire primale.

Définition 5.21 *Un* **problème d'unification primale** $\mathcal{P}$ *défini sur les termes primaux* $\mathcal{TP}(\mathcal{K},\mathcal{D};\mathcal{X},\mathcal{C})$ *et le système de réécriture Presburger* $\mathcal{R}$ *est une conjonction finie dont les atomes sont l'*identité $\top$*, l'*échec $-$ *et des équations* $t \stackrel{?}{=} t'$ *où* t *et* t' *sont des termes primaux.*

Remarque: Tous les problèmes d'unification considérés jusqu'à la fin de ce chapitre sont des problèmes primaux. Pour nous faciliter la tâche, nous allons considérer uniquement les problèmes d'unification de systèmes Presburger *plats*. Les relations de réécriture $\Longrightarrow_{\mathcal{R}}$ et $\longrightarrow_{\mathcal{R}}$ sont équivalentes pour les systèmes Presburger plats.

Définition 5.22 *L'***unificateur** *local du problème $\mathcal{P}$ est constitué de la paire $\langle \mu;\nu \rangle$, où $\mu{:}\mathcal{C} \longrightarrow \mathcal{N}$ est un énumérateur et $\nu{:}\mathcal{X} \longrightarrow \mathcal{T}(\mathcal{K},\mathcal{X})$ est une substitution idempotente, tels que pour chaque équation $t_i \stackrel{?}{=} t'_i$ du problème $\mathcal{P}$ on ait:*

$$t_i\nu\mu\Downarrow_{\mathcal{R}} \quad = \quad t'_i\nu\mu\Downarrow_{\mathcal{R}} \tag{5.6}$$

L'ensemble des unificateurs du problème $\mathcal{P}$ est noté $\mathcal{U}(\mathcal{P})$. L'unificateur $\langle \mu;\nu \rangle$ est *plus général* que l'unificateur $\langle \mu';\nu' \rangle$ s'il existe des substitutions σ et τ telles que $\mu' = \mu\sigma|_V$ et $\nu' = \nu\tau|_V$, où V est l'ensemble des variables du problème $\mathcal{P}$. L'unificateur $\langle \mu;\nu \rangle$ est *strictement plus général* que l'unificateur $\langle \mu';\nu' \rangle$ si $\langle \mu;\nu \rangle$ est plus général que $\langle \mu';\nu' \rangle$ et ni σ ni τ ne sont des renommages des variables. L'unificateur $\langle \mu;\nu \rangle$ dans $\mathcal{U}(\mathcal{P})$ est dit *principal* si $\mathcal{U}(\mathcal{P})$ ne contient aucun unificateur strictement plus général que $\langle \mu;\nu \rangle$ sur les variables de $\mathcal{P}$. L'*ensemble minimal et complet des unificateurs* du problème $\mathcal{P}$ est l'ensemble $\mu CSU(\mathcal{P})$ tel que

- chaque unificateur $\langle \mu;\nu \rangle \in \mu CSU(\mathcal{P})$ est un unificateur principal dans $\mathcal{U}(\mathcal{P})$, et
- chaque unificateur principal de $\mathcal{U}(\mathcal{P})$ appartient à $\mu CSU(\mathcal{P})$.

Une *expression linéaire* est un polynôme linéaire $n_0 + n_1 * c_1 + \cdots + n_k * c_k$ sur les variables compteurs $c_1, \ldots, c_k$ et les coefficients entiers $n_0, n_1, \ldots, n_k$. Nous allons utiliser aussi la notation abrégée $n_0 + n_1c_1 + \cdots + n_kc_k$. La classe des expressions linéaires, définies sur les variables compteurs $\mathcal{C}$, est notée $\mathcal{L}in(\mathcal{C})$. Une *équation diophantienne linéaire* est une équation $l \stackrel{?}{=} l'$ où l et l' sont des expressions linéaires.

Exemple 5.23 Les polynômes $2 + 5c_1 - 3c_2$ et $3 - 2c_1 + 7c_2$ sont des expressions linéaires, tandis que $5c_1c_2$ et $2c_1^3$ ne le sont pas.

Une expression linéaire l est évaluée par un énumérateur ξ suivi par l'exécution des opérations arithmétiques présentes dans l. L'interprétation des opérations arithmétiques étant naturelle, nous ne faisons pas de distinction entre une expression linéaire énumérée $l\xi$ et sa valeur. Néanmoins, nous sommes obligés de nous assurer que les expressions linéaires s'évaluent sur des valeurs entières positives. Ceci est garanti par les *énumérateurs admissibles* de l. Un énumérateur ξ est admissible pour l si $l\xi$ est un entier positif. Un énumérateur est admissible pour un ensemble L d'expressions linéaires s'il est admissible pour chaque expression linéaire $l \in L$.

Exemple 5.24 L'énumérateur $[c_1 \mapsto 2, c_2 \mapsto 1]$ est admissible pour $2+5c_1-3c_2$. Par contre, l'énumérateur $[c_1 \mapsto 10, c_2 \mapsto 1]$ n'est pas admissible pour $3 - 2c_1 + 7c_2$.

Définition 5.25 *L'***unificateur global** *d'un problème $\mathcal{P}$ est constitué de la paire $\langle \mu;\nu \rangle$ de substitutions idempotentes $\mu{:}\mathcal{C} \longrightarrow \mathcal{L}in(\mathcal{C})$ et $\nu{:}\mathcal{X} \longrightarrow \mathcal{TP}(\mathcal{K},\mathcal{D};\mathcal{X},\mathcal{C})$, telles que pour chaque équation $t_i \stackrel{?}{=} t'_i$ du problème $\mathcal{P}$ et pour chaque énumérateur $\xi_j{:}\mathcal{C} \longrightarrow \mathcal{N}$ on ait:*

$$t_i\nu\mu\xi_j\Downarrow_{\mathcal{R}} \quad = \quad t'_i\nu\mu\xi_j\Downarrow_{\mathcal{R}} \tag{5.7}$$

Dans le cas des unificateurs locaux, l'application de μ et ν dans (5.6) peut se faire dans n'importe quel ordre, car ν ne contient pas de variable compteur et μ est une substitution close. Tandis que dans le cas des unificateurs globaux, l'ordre d'application de ν avant μ dans (5.7) est important, car ν peut contenir des variables compteurs appartenant au domaine de μ.

L'ensemble U des unificateurs locaux est dit *linéaire* s'il existe un unificateur global $\langle \mu; \nu \rangle$ tel que chaque unificateur local $\langle \mu_i; \nu_i \rangle$ appartient à U si et seulement si $x\nu_i = x\nu\mu_i\Downarrow_{\mathcal{R}}$ et $c\mu_i = c\mu\xi_i$ pour chaque variable ordinaire $x \in \mathcal{X}$ et chaque variable compteur $c \in \mathcal{C}$ et pour chaque énumérateur admissible $\xi_i : \mathcal{C} \longrightarrow \mathcal{N}$. On dit que l'unificateur global $\langle \mu; \nu \rangle$ subsume l'ensemble des unificateurs locaux U.

Un ensemble d'unificateurs U est dit *semi-linéaire* s'il est une union finie d'ensembles linéaires d'unificateurs, c'est-à-dire qu'une disjonction finie d'unificateurs globaux subsume l'ensemble U. Nous allons démontrer que l'ensemble complet d'unificateurs principaux de chaque problème d'unification primale est semi-linéaire.

Exemple 5.26 Soit

$$\begin{array}{rcl@{\qquad}rcl} \hat{f}(0;x) & \to & x & \hat{f}(c+1;x) & \to & F(\hat{f}(c;x)) \\ \hat{g}(0) & \to & a & \hat{g}(c+1) & \to & F(\hat{g}(c)) \end{array}$$

le système Presburger pour le problème d'unification $\hat{f}(c_1;x) \stackrel{?}{=} \hat{g}(c_2)$. Son ensemble minimal et complet d'unificateurs est

$$\begin{array}{lll} \langle [c_1 \mapsto 0, c_2 \mapsto 0]; [x \mapsto a] \rangle, & \langle [c_1 \mapsto 0, c_2 \mapsto 1]; [x \mapsto F(a)] \rangle, & \ldots \\ \langle [c_1 \mapsto 1, c_2 \mapsto 1]; [x \mapsto F(a)] \rangle, & \langle [c_1 \mapsto 1, c_2 \mapsto 2]; [x \mapsto F^2(a)] \rangle, & \ldots \\ \vdots & \vdots & \vdots \end{array}$$

Cet ensemble minimal et complet d'unificateurs est linéaire car l'unificateur global

$$\langle [c_1 \mapsto c'_1, c_2 \mapsto c'_1 + c'_2]; [x \mapsto \hat{g}(c_2)] \rangle$$

est linéaire et subsume l'ensemble précédent.

5.2.1 Algorithme d'unification

L'algorithme d'unification présenté ici transforme tout problème d'unification $\mathcal{P}$ en un problème plus simple $\mathcal{P}'$ et en un ensemble fini (ou conjonction) d'équations diophantiennes linéaires $\mathcal{L}$. Au cours de l'algorithme, il est parfois nécessaire de séparer une équation du reste du problème $\mathcal{P}$, d'appliquer plusieurs opérations sur ce problème séparé, et d'incorporer ce résultat partiel au reste du problème original.

Définition 5.27 *Un problème d'unification* **mixte** *est constitué d'une paire* $(\mathcal{L}; \mathcal{P})$*, où* $\mathcal{P}$ *est un problème d'unification primale et* $\mathcal{L}$ *une conjonction d'équations diophantiennes linéaires.*

Un problème d'unification **séparé** *est un problème mixte* $(\mathcal{L}; \mathcal{P})$*, où* $\mathcal{P}$ *est atomique.*

L'algorithme d'unification présenté ici est un mélange d'unification syntaxique et d'un processus de surréduction. Cet algorithme est composé de deux procédures dites principale et subordonnée; toutes deux consistent en un ensemble de règles de transition. L'implantation de cet algorithme a été entrepris par Lamia Djerid au cours de son stage de DEA [Dje93].

Procédure principale

La procédure principale est une extension de l'unification syntaxique aux termes du premier ordre [JK91]. Elle traite les problèmes mixtes en commençant toujours par le problème $(\top; \mathcal{P})$. Elle est constituée de l'ensemble de règles de la figure 5.2.

Les six premières règles ne diffèrent pas tellement des règles de l'unification syntaxique. Elles réalisent les mêmes opérations sur les constructeurs et les variables ordinaires et n'interviennent pas du tout dans la partie diophantienne $\mathcal{L}$.

La règle Check déclare un échec uniquement dans le cas où la variable x est présente dans l'emballage du terme t. La variable x peut apparaître sous un symbole défini sans que cela soit un cas d'*occur check* (test d'occurrence). En fait, ce cas est traité par une des règles de la procédure subordonnée.

Par contre, les cinq dernières règles préparent le traitement des symboles définis pour la procédure subordonnée et récupèrent le résultat de celle-ci.

La règle Linearize permet de linéariser l'équation primale $t \stackrel{?}{=} t'$ avant qu'elle soit traitée par la procédure subordonnée. La linéarisation consiste à éliminer les variables compteurs communes entre t et t', car la décision d'arrêt de l'arbre de surréduction ne peut être possible que si les problèmes séparés sont linéarisés.

Les termes à unifier peuvent contenir dans leurs positions compteurs des expressions et pas nécessairement des variables. Or, la procédure subordonnée n'est valide que si toutes les positions compteurs sont occupées par des variables. D'où la nécessité de la règle Lift, qui justement permet de remplacer, dans une position compteur, une expression n par une nouvelle variable compteur c et de rajouter l'équation $c \stackrel{?}{=} n$ dans $\mathcal{L}$.

Pour des raisons d'efficacité, il est préférable d'appliquer la règle Simplify avant la règle Lift. La règle Simplify permet de réduire les termes d'une équation primale à l'aide du système Presburger $\mathcal{R}$.

La règle Separate permet de déclencher la procédure subordonnée qui crée autant de branches alternatives que de paires $(\mathcal{L}_i; \mathcal{P}_i)$ qui apparaissent dans la solution du problème séparé $t \stackrel{?}{=} t'$.

La règle Dio peut utiliser l'algorithme de Contejean et Devie [CD91] ou celui de Domenjoud [Dom91] pour résoudre les équations diophantiennes $\mathcal{L}$.

Procédure subordonnée

La procédure subordonnée est appelée par la règle Separate avec le problème linéarisé $(\top; t \stackrel{?}{=} t')$. Elle n'appelle jamais récursivement la procédure principale, mais elle revient à son point d'appel. Elle est constituée de l'ensemble de règles de la figure 5.3, ainsi que de la figure 5.4.

L'obtention d'un problème d'unification $(\mathcal{L}'; \mathcal{P}')$ à partir d'un autre problème $(\mathcal{L}; \mathcal{P})$ par application d'une des règles de transition est notée $(\mathcal{L}; \mathcal{P}) \vdash (\mathcal{L}'; \mathcal{P}')$. La notation $\vdash_Q$ indique explicitement l'utilisation de la règle de transition Q à cette étape de la déduction. $\vdash^*$ est la fermeture réflexive et transitive de la relation $\vdash$.

Les mêmes remarques que celles faites pour la procédure principale peuvent être faites pour les quatre premières règles. L'absence de la règle Eliminate est due au fait qu'elle peut

Delete:	$\dfrac{\mathcal{L};\mathcal{P} \wedge t \stackrel{?}{=} t}{\mathcal{L};\mathcal{P}}$	
Decompose:	$\dfrac{\mathcal{L};\mathcal{P} \wedge f(\vec{t}) \stackrel{?}{=} f(\vec{t'})}{\mathcal{L};\mathcal{P} \wedge t_1 \stackrel{?}{=} t'_1 \wedge \ \ldots \ \wedge t_n \stackrel{?}{=} t'_n}$	si $f \in \mathcal{K}$
Conflict:	$\dfrac{\mathcal{L};\mathcal{P} \wedge f(\vec{t}) \stackrel{?}{=} g(\vec{t'})}{-}$	si $f, g \in \mathcal{K}$ et $f \neq g$
Coalesce:	$\dfrac{\mathcal{L};\mathcal{P} \wedge x \stackrel{?}{=} y}{\mathcal{L};\mathcal{P}[x \mapsto y] \wedge x \stackrel{?}{=} y}$	si $x \in \mathcal{V}ar(\mathcal{P})$, $y \in \mathcal{X}$ et $x \neq y$
Check:	$\dfrac{\mathcal{L};\mathcal{P} \wedge x \stackrel{?}{=} t}{-}$	si $\mathcal{H}ead(t) \in \mathcal{K}$ et $x \in \mathcal{V}ar(\mathcal{W}rp(t))$
Eliminate:	$\dfrac{\mathcal{L};\mathcal{P} \wedge x \stackrel{?}{=} t}{\mathcal{L};\mathcal{P}[x \mapsto t] \wedge x \stackrel{?}{=} t}$	si $x \notin \mathcal{V}ar(t)$, $t \notin \mathcal{X}$ et $x \in \mathcal{V}ar(\mathcal{P})$
Linearize:	$\dfrac{\mathcal{L};\mathcal{P} \wedge t \stackrel{?}{=} t'}{\mathcal{L} \wedge c \stackrel{?}{=} c';\mathcal{P} \wedge t \stackrel{?}{=} t'[c \mapsto c']}$	si $c \in \mathcal{CV}ar(t) \cap \mathcal{CV}ar(t')$ et c' est une nouvelle variable compteur
Simplify:	$\dfrac{\mathcal{L};\mathcal{P} \wedge t \stackrel{?}{=} t'}{\mathcal{L};\mathcal{P} \wedge t \stackrel{?}{=} t''}$	si $t' \longrightarrow_{\mathcal{R}} t''$
Lift:	$\dfrac{\mathcal{L};\mathcal{P} \wedge t'[\hat{f}(\ldots, n, \ldots; \vec{t}) \stackrel{?}{=} t''}{\mathcal{L} \wedge c \stackrel{?}{=} n;\mathcal{P} \wedge t'[\hat{f}(\ldots, c, \ldots; \vec{t})] \stackrel{?}{=} t''}$	si $\hat{f} \in \mathcal{D}$, $n \in \mathcal{N}(\mathcal{C}) - \mathcal{C}$ et c est une nouvelle variable compteur
Separate:	$\dfrac{\mathcal{L};\mathcal{P} \wedge t \stackrel{?}{=} t'}{(\mathcal{L} \wedge \mathcal{L}_1;\mathcal{P} \wedge \mathcal{P}_1) \vee \cdots \vee (\mathcal{L} \wedge \mathcal{L}_n;\mathcal{P} \wedge \mathcal{P}_n)}$	si $\mathcal{H}ead(t) \in \mathcal{D}$; t, t' sont irréductibles et $\bigvee_i(\mathcal{L}_i;\mathcal{P}_i)$ est le résultat produit par la procédure subordonnée sur le problème linéarisé $(\top; t \stackrel{?}{=} t')$
Dio:	$\dfrac{\mathcal{L};\mathcal{P}}{solve(\mathcal{L});\mathcal{P}}$	si $solve(\mathcal{L})$ est la forme résolue des équations diophantiennes linéaires $\mathcal{L}$

FIG. 5.2 – *Unification primale: Procédure principale*

Delete: $$\frac{\mathcal{L}; t \stackrel{?}{=} t}{\mathcal{L}; \top}$$

Decompose: $$\frac{\mathcal{L}; f(\vec{t}) \stackrel{?}{=} f(\vec{t'})}{\mathcal{L}; t_1 \stackrel{?}{=} t'_1 \quad \wedge \cdots \wedge \quad \mathcal{L}; t_n \stackrel{?}{=} t'_n} \quad \text{si } f \in \mathcal{K}$$

Conflict: $$\frac{\mathcal{L}; f(\vec{t}) \stackrel{?}{=} g(\vec{t'})}{-} \quad \text{si } f, g \in \mathcal{K} \text{ et } f \neq g$$

Check: $$\frac{\mathcal{L}; x \stackrel{?}{=} t}{-} \quad \text{si } \mathcal{H}ead(t) \in \mathcal{K} \text{ et } x \in \mathcal{V}ar(\mathcal{W}rp(t))$$

FIG. 5.3 – *Unification primale: Procédure subordonnée, partie 1*

Fork: $$\frac{\mathcal{L}; \hat{f}(c, \vec{n}; \vec{t}) \stackrel{?}{=} t'}{\mathcal{L} \wedge c \stackrel{?}{=} 0; \hat{f}(0, \vec{n}; \vec{t}) \stackrel{?}{=} t' \quad \vee \quad \mathcal{L} \wedge c \stackrel{?}{=} c' + 1; \hat{f}(c' + 1, \vec{n}; \vec{t}) \stackrel{?}{=} t'}$$

si $\hat{f} \in \mathcal{D}$, $c \in \mathcal{C}$, $\vec{n}$ sont des variables compteurs, t' est irréductible, $t' \notin \mathcal{X}$, et c' est une nouvelle variable compteur

Bang: $$\frac{\mathcal{L}; x \stackrel{?}{=} t}{\mathcal{L} \wedge c \stackrel{?}{=} 0; x \stackrel{?}{=} t[c \mapsto 0]}$$

si t est irréductible, $x \in \mathcal{X}$, $x \notin \mathcal{V}ar(\mathcal{W}rp(t))$ et $\exists p \in \mathcal{DP}os(t)$ tel que $t|_p = \hat{f}(c, \vec{n}; \vec{t})$ où $c \in \mathcal{C}$ et $x \in \mathcal{V}ar(\vec{t})$

Simplify: $$\frac{\mathcal{L}; t \stackrel{?}{=} t'}{\mathcal{L}; t \stackrel{?}{=} t''} \quad \text{si } t' \longrightarrow_{\mathcal{R}} t''$$

Relax: $$\frac{\mathcal{L}; \hat{f}(\ldots, c + 1, \ldots; \vec{t}) \stackrel{?}{=} t'}{\mathcal{L} \wedge c \stackrel{?}{=} c' - 1; \hat{f}(\ldots, c', \ldots; \vec{t}) \stackrel{?}{=} t'}$$

si $\hat{f} \in \mathcal{D}$, $c \in \mathcal{C}$, $k \in \mathcal{N}$ et c' est une nouvelle variable compteur

FIG. 5.4 – *Unification primale: Procédure subordonnée, partie 2*

créer des équations non linéaires. En effet, si par exemple nous avions au niveau de la procédure subordonnée la conjonction $x \stackrel{?}{=} t \wedge x \stackrel{?}{=} t'$, l'application de la règle Eliminate au niveau de la procédure subordonnée nous donnerait $x \stackrel{?}{=} t \wedge t \stackrel{?}{=} t'$. Or t et t' peuvent avoir des compteurs en commun et il n'existe aucun moyen de linéariser l'équation $t \stackrel{?}{=} t'$ au niveau de la procédure subordonnée. Il faut donc renvoyer la conjonction $x \stackrel{?}{=} t \wedge x \stackrel{?}{=} t'$ telle quelle à la procédure principale. Cette dernière se chargera alors d'appliquer Eliminate (ou Coalesce) et de linéariser si nécessaire.

La règle Fork provoque un branchement alternatif dans le processus de déduction. En effet, sa conclusion est une disjonction de deux parties: la première est la branche *basique*, la seconde est la branche *inductive*. Le problème séparé $(\mathcal{L}; \hat{f}(c, \vec{n}; \vec{t}) \stackrel{?}{=} t')$ sur lequel s'applique la règle Fork s'appelle un *problème fourchu* et l'équation $\hat{f}(c, \vec{n}; \vec{t}) \stackrel{?}{=} t'$ est dite l'*équation fourchue*. Le terme $\hat{f}(c, \vec{n}; \vec{t})$ dans l'équation est dit *terme fourchu* et le symbole $\hat{f}$ est dit *symbole fourchu*.

La règle Decompose provoque un branchement conjonctif dans le processus de déduction. En effet, sa conclusion est une conjonction de problèmes séparés. Le nombre de branches créées par Decompose dépend de l'arité du constructeur sur lequel s'applique la règle.

Tout au long du traitement de la procédure subordonnée, on peut voir apparaître des expressions compteurs de la forme $c + 1$. Elles sont dues à l'application de Simplify après un Fork. En effet, la réécriture du terme $\hat{f}(n + 1, \vec{c}; \vec{t})$ peut créer un ou plusieurs sous-termes $\hat{f}(n, c_1, \ldots, c_{i-1}, c_i + 1, c_{i+1}, \ldots, c_k; \vec{t})$ par une réduction avec une règle de réécriture inductive. Le rôle de Relax est justement d'éliminer toute expression $c + 1$ pour la remplacer par une nouvelle variable compteur c' et d'ajouter l'équation $c \stackrel{?}{=} c' - 1$ dans $\mathcal{L}$.

Comme il a été mentionné dans la description de la procédure principale, une équation $x \stackrel{?}{=} t$ où x appartient à l'emballage de t est un cas d'échec. Par contre, le cas d'une équation $x \stackrel{?}{=} \hat{f}(c; x)$ où x apparaît sous le symbole défini $\hat{f}$, est traité par la règle Bang. En effet, si la règle basique du système Presburger $\mathcal{R}$ est $\hat{f}(0; x) \to x$, le problème à résoudre étant $(\top; x \stackrel{?}{=} \hat{f}(c; x))$, la déduction correcte est:

$$\top; x \stackrel{?}{=} \hat{f}(c; x) \vdash_{\mathsf{Bang}} c \stackrel{?}{=} 0; x \stackrel{?}{=} \hat{f}(0; x) \vdash_{\mathsf{Simplify}} c \stackrel{?}{=} 0; x \stackrel{?}{=} x \vdash_{\mathsf{Delete}} c \stackrel{?}{=} 0; \top$$

et l'unificateur $([c \mapsto 0]; \top)$ est la solution du problème $(\top; x \stackrel{?}{=} \hat{f}(c; x))$.

Les règles de transition construisent une sorte d'*arbre complet de surréduction*, où les feuilles ne représentent cependant pas des formes résolues dans le sens standard.

Définition 5.28 *Une* **forme pré-résolue** *est une paire de conjonctions d'équations*

$$\left(\bigwedge_{i=1}^{p} c_i \stackrel{?}{=} l_i; \bigwedge_{j=1}^{q} x_j \stackrel{?}{=} t_j \right)$$

telle que

- $\forall i \forall j \ (c_i \notin \mathcal{CV}ar(l_j))$
- $\forall i \forall j \ (x_i \notin \mathcal{V}ar(t_j))$

où les c_i sont des variables compteurs, les l_i sont des expressions linéaires, les x_j sont des variables ordinaires et les t_j sont des termes primaux.

Une **forme résolue** *est une forme pré-résolue augmentée de*

– $\forall i \forall j \ (c_i = c_j) \supset (i = j)$

– $\forall i \forall j \ (x_i = x_j) \supset (i = j)$

La conjonction *des formes pré-résolues $(\mathcal{L}_1; \mathcal{P}_1)$, ..., $(\mathcal{L}_n; \mathcal{P}_n)$ est la forme pré-résolue*

$$\left(\bigwedge_{i=1}^{n} \mathcal{L}_i ; \bigwedge_{j=1}^{n} \mathcal{P}_j \right)$$

La disjonction *des formes pré-résolues $(\mathcal{L}_1; \mathcal{P}_1)$, ..., $(\mathcal{L}_n; \mathcal{P}_n)$ est la formule*

$$(\mathcal{L}_1; \mathcal{P}_1) \vee \cdots \vee (\mathcal{L}_n; \mathcal{P}_n)$$

Une **quasi-solution** *est une disjonction finie de formes pré-résolues. Une* **solution** *est une disjonction finie de formes résolues.*

La procédure subordonnée construit un arbre de surréduction dont les feuilles sont des formes pré-résolues et non des formes résolues. Cela est dû au fait qu'au niveau de la procédure subordonnée, nous n'appelons ni Eliminate, ni Coalesce, ni Dio.

Partant des feuilles, nous devons récupérer la quasi-solution du problème $(\top; \mathcal{P})$ de bas en haut sous forme normale disjonctive. Donc les formules récupérées au niveau de chaque nœud doivent — elles aussi — être sous cette forme. Il reste alors à résoudre le problème des branchements qui peuvent être de deux sortes: Decompose et Fork.

Dans le cas de Decompose

$$\frac{\mathcal{L}; \mathcal{P}}{\mathcal{L}; \mathcal{P}_1 \quad \wedge \cdots \wedge \quad \mathcal{L}; \mathcal{P}_n} \ (\mathsf{Decompose})$$

chaque nœud $(\mathcal{L}; \mathcal{P}_i)$ est décoré par une formule $\mathcal{A}_i$, le nœud $(\mathcal{L}; \mathcal{P})$ est donc décoré par la forme normale disjonctive de la formule $\mathcal{A}_1 \wedge \cdots \wedge \mathcal{A}_n$.

Dans le cas de Fork

$$\frac{\mathcal{L}; \mathcal{P}}{\mathcal{L}_1; \mathcal{P}_1 \quad \vee \quad \mathcal{L}_2; \mathcal{P}_2} \ (\mathsf{Fork})$$

chacun des deux nœuds $(\mathcal{L}_1; \mathcal{P}_1)$ et $(\mathcal{L}_2; \mathcal{P}_2)$ est décoré par une formule $\mathcal{A}_i$ $(i = 1, 2)$. Le nœud $(\mathcal{L}; \mathcal{P})$ est alors décoré par la formule $\mathcal{A}_1 \vee \mathcal{A}_2$.

Cette reconstitution est, bien sûr, purement hypothétique car l'arbre construit est potentiellement infini. De plus, la procédure subordonnée ne peut pas restituer une disjonction infinie à la procédure principale. Nous allons d'abord voir comment élaguer cet arbre potentiellement infini pour obtenir un arbre fini. Ensuite une extension de la reconstitution va nous donner une méthode pour construire des quasi-solutions pour des problèmes séparés.

5.2.2 Similarité des problèmes

La procédure subordonnée pouvant engendrer une disjonction infinie, nous allons donc, tout d'abord, étudier l'origine de ce problème puis montrer que la notion de similarité est un moyen efficace pour arrêter la divergence du processus de surréduction en permettant de récupérer une quasi-solution pour la procédure principale.

Origine des branches infinies

La divergence de la procédure subordonnée est due à l'utilisation de la règle Fork suivie des règles Simplify, Relax et Decompose. Nous allons voir qu'en fait un déduction infinie ne peut être causée que par l'utilisation répétée de la règle Fork.

Lemme 5.29 *Chaque déduction* $(\mathcal{L}_1;\mathcal{P}_1) \vdash (\mathcal{L}_2;\mathcal{P}_2) \vdash \cdots \vdash (\mathcal{L}_i;\mathcal{P}_i)$ *résultant de l'application des règles* Delete, Decompose, Conflict, Check, Bang, Simplify *et* Relax *est finie.*

Preuve: L'application d'une des règles Delete, Conflict ou Check termine évidemment la déduction. Pour le reste, nous introduisons pour chaque règle de transition un ordre bien fondé dans lequel la prémisse d'une règle est plus grande que chaque feuille de la conclusion. L'extension lexicographique de ces ordres prouve alors la terminaison de chaque déduction.

Soit $(\mathcal{L};t_1 \stackrel{?}{=} t'_1) \vdash_{\mathsf{Decompose}} (\mathcal{L};t_2 \stackrel{?}{=} t'_2)$, alors t_2 est un sous-terme strict de t_1 et t'_2 est un sous-terme strict de t'_1. L'ordre pour Decompose est alors le produit cartésien $\rhd \times \rhd$ de la relation de sous-terme strict $\rhd$.

Soit $(\mathcal{L};t \stackrel{?}{=} t') \vdash_{\mathsf{Simplify}} (\mathcal{L};t \stackrel{?}{=} t'')$ alors $t' \longrightarrow_{\mathcal{R}} t''$. Chaque système Presburger $\mathcal{R}$ est nœthérien, alors la relation de réécriture $\longrightarrow_{\mathcal{R}}$ est bien fondée. L'ordre pour Simplify est alors la relation de réécriture $\longrightarrow_{\mathcal{R}}$. Un pas de Simplify est incomparable dans l'ordre pour Decompose car t n'est pas un sous-terme strict de t.

Soit $(\mathcal{L};x \stackrel{?}{=} t) \vdash_{\mathsf{Bang}} (\mathcal{L}';x \stackrel{?}{=} t')$, alors $t' = t[c \mapsto 0]$ pour une variable compteur $c \in \mathcal{CV}ar(t)$. Le terme primal t contient plus de variables compteurs que t'. L'ordre pour Bang est alors le nombre de variables compteurs dans un terme primal. Un pas de Bang est incomparable par rapport aux deux ordres précédents.

Soit $(\mathcal{L};t_1 \stackrel{?}{=} t') \vdash_{\mathsf{Relax}} (\mathcal{L}';t_2 \stackrel{?}{=} t')$ alors t_1 contient plus d'expressions compteurs non-variables que t_2. L'ordre pour Relax est le nombre d'expressions compteurs, qui ne sont pas des variables, dans un terme primal. Un pas de Relax est incomparable aux ordres précédents. □

L'application systématique de la règle Fork a pour but d'instancier les variables compteurs par toutes les valeurs possibles. Ce qui peut, en général, mener à un développement infini d'une branche. Cependant, le développement d'un nœud peut être arrêté si ce dernier est similaire à un problème ancêtre (donc déjà développé).

Similarité entre deux problèmes

Une *translation* est une substitution idempotente

$$\kappa : \mathcal{C} \longrightarrow \mathcal{L}in(\mathcal{C})$$

telle que $\kappa = [c_1 \mapsto c'_1 + k_1, \ldots, c_n \mapsto c'_n + k_n]$ avec $k_1, \ldots, k_n$ des constantes entières et $c'_1, \ldots, c'_n$ des variables compteurs. Un *renommage* $\rho{:}\mathcal{C} \longrightarrow \mathcal{C}$ est une application injective, admettant comme réciproque le renommage ρ^{-1}.

Exemple 5.30 La substitution $[c_1 \mapsto c'_1 + 3, c_2 \mapsto c'_2 - 5]$ est une translation si c_1, c_2, c'_1 et c'_2 sont des variables compteurs. Par contre, $[c_1 \mapsto c'_1, c_2 \mapsto c'_1 + c'_2]$ n'en est pas une.

Définition 5.31 *Un problème d'unification séparé $(\mathcal{L}';\mathcal{P}')$ est similaire à un autre problème $(\mathcal{L};\mathcal{P})$ s'il existe une translation κ et un renommage ρ tels que $\mathcal{L}'\kappa = \mathcal{L}\rho$ et $\mathcal{P}' = \mathcal{P}\rho$.*

Les conditions de similarité peuvent être exprimées par $\mathcal{L} = \mathcal{L}'\kappa\rho^{-1}$ et $\mathcal{P} = \mathcal{P}'\rho^{-1}$ si cela est plus approprié. Il y a aussi un problème d'interprétation de l'identité. Nous ne faisons pas de différence entre l'équation diophantienne $l \stackrel{?}{=} l$ et l'identité $\top$ car les deux sont isomorphes.

Exemple 5.32 Le problème séparé

$$(\mathcal{L}';\mathcal{P}') \quad = \quad (k \stackrel{?}{=} k' + 1 \wedge n \stackrel{?}{=} n' + 1; \hat{f}(k';x) \stackrel{?}{=} \hat{g}(n'))$$

est similaire au problème

$$(\mathcal{L};\mathcal{P}) \quad = \quad (\top; \hat{f}(k;x) \stackrel{?}{=} \hat{g}(n))$$

En effet, la translation κ et le renommage ρ étant respectivement

$$\kappa \quad = \quad [k \mapsto k' + 1, n \mapsto n' + 1] \qquad \text{et} \qquad \rho \quad = \quad [k' \mapsto k, n' \mapsto n]$$

nous avons d'une part $\mathcal{L} = \mathcal{L}'\kappa\rho^{-1}$ puisque

$$(k \stackrel{?}{=} k' + 1 \wedge n \stackrel{?}{=} n' + 1)\kappa\rho^{-1} \quad = \quad (k + 1 \stackrel{?}{=} k + 1 \wedge n + 1 \stackrel{?}{=} n + 1)$$

qui est équivalent à $\top$ et d'autre part $\mathcal{P} = \mathcal{P}'\rho^{-1}$ puisque

$$\mathcal{P}'\rho^{-1} \quad = \quad (\hat{f}(k';x) \stackrel{?}{=} \hat{g}(n'))\rho^{-1} \quad = \quad \mathcal{P}$$

La translation κ est unique modulo un renommage de variables compteurs. En effet, supposons qu'il existe deux translations différentes κ et κ' avec respectivement deux renommages différents ρ et ρ'. Prouver que le problème $(\mathcal{L}';\mathcal{P}')$ est similaire au problème $(\mathcal{L};\mathcal{P})$ revient à vérifier les deux conditions suivantes: $\mathcal{L}'\kappa = \mathcal{L}\rho$ et $\mathcal{L}'\kappa' = \mathcal{L}\rho'$ ou encore $\mathcal{L} = \mathcal{L}'\kappa\rho^{-1}$ et $\mathcal{L} = \mathcal{L}'\kappa'\rho'^{-1}$, d'où nous déduisons $\kappa\rho^{-1} = \kappa'\rho'^{-1}$. Donc la translation κ' est égale à la translation κ modulo un renommage de variables.

Définition 5.33 *Soit T l'arbre construit par la procédure subordonnée pour le problème séparé $(\mathcal{L};\mathcal{P})$.*

Un problème d'unification séparé $(\mathcal{L}';\mathcal{P}')$ est un **descendant similaire** *à un problème $(\mathcal{L};\mathcal{P})$ si $(\mathcal{L}';\mathcal{P}')$ est un descendant de $(\mathcal{L};\mathcal{P})$ et $(\mathcal{L}';\mathcal{P}')$ est similaire à $(\mathcal{L};\mathcal{P})$.*

Un problème d'unification $(\mathcal{L}';\mathcal{P}')$ est **le plus proche descendant similaire** (PPDS) *de $(\mathcal{L};\mathcal{P})$ s'il n'existe pas d'autre descendant similaire entre $(\mathcal{L};\mathcal{P})$ et $(\mathcal{L}';\mathcal{P}')$.*

L'ensemble complet des PPDS d'un problème $(\mathcal{L};\mathcal{P})$, noté $CSCSD(\mathcal{L};\mathcal{P})$, est l'ensemble de tous les PPDS de $(\mathcal{L};\mathcal{P})$.

Un problème d'unification séparé peut avoir 0, 1 ou plusieurs PPDS.

Le théorème suivant présente l'outil principal pour arrêter le développement d'un arbre potentiellement infini. Sa preuve est basée sur la linéarité des variables compteurs dans le problème d'unification séparé et sur l'existence d'une précédence $\succ$ sur les symboles définis $\mathcal{D}$.

Théorème 5.34 *Soit T l'arbre construit par la procédure subordonnée pour le problème séparé $(\mathcal{L};\mathcal{P})$. Chaque branche infinie de T contient deux problèmes séparés $(\mathcal{L}';\mathcal{P}')$ et $(\mathcal{L}'';\mathcal{P}'')$ tels que $(\mathcal{L}'';\mathcal{P}'')$ est un descendant similaire de $(\mathcal{L}';\mathcal{P}')$.*

Preuve: Soit β une branche infinie de l'arbre T. Une telle branche ne peut avoir qu'un nombre infini d'applications des règles choisies parmi Decompose, Fork, Simplify et Relax.

Aucune branche de T ne contient un nombre infini de branchements basiques de Fork car chaque branchement basique diminue le nombre de variables compteurs présentes dans le problème séparé. Le même argument s'applique à la règle Bang. Considérons alors uniquement de telles branches infinies β de T qui ne contiennent aucun branchement basique de Fork et aucune application de la règle Bang. D'après le lemme 5.29, il existe un nombre infini de branchements inductifs de Fork sur une telle branche infinie β.

Considérons la suite infinie $\vec{F} = (\mathcal{L}_1; t_1 \stackrel{?}{=} t'_1), \ldots, (\mathcal{L}_i; t_i \stackrel{?}{=} t'_i), \ldots$ de tous les problèmes fourchus sur la branche β. Appelons t-termes la partie gauche des équations $t_i \stackrel{?}{=} t'_i$ (c'est-à-dire la suite de termes primaux t_1, t_2, ..., t_i, ...) et t'-termes la partie droite. Soit $\vec{G}$ la sous-suite de $\vec{F}$ où les termes fourchus sont les t-termes et $\vec{D}$ la sous-suite de $\vec{F}$ où les termes fourchus sont les t'-termes.

Les deux suites, $\vec{G}$ et $\vec{D}$, sont infinies. Si l'une des deux, par exemple $\vec{D}$, était finie alors à partir d'une position p de la branche β tous les termes fourchus seraient des t-termes et les t'-termes seraient uniquement décomposés. Cela est impossible car nous ne pouvons pas décomposer infiniment un terme fini.

Pour chaque problème fourchu $(\mathcal{L}_i; t_i \stackrel{?}{=} t'_i) \in \vec{F}$, notons $\hat{g}_i$ son symbole fourchu si t_i est le terme fourchu et $\hat{d}_i$ si t'_i est le terme fourchu. Nous avons construit respectivement des suites $\hat{G}$ et $\hat{D}$ de symboles fourchus pour les problèmes fourchus $\vec{G}$ et $\vec{D}$.

Analysons ce qui se passe avec les t-termes sur la branche β entre deux problèmes fourchus consécutifs $(\mathcal{L}_j; t_j \stackrel{?}{=} t'_j)$ et $(\mathcal{L}_{j+1}; t_{j+1} \stackrel{?}{=} t'_{j+1})$ dans la suite $\vec{G}$. La même analyse s'applique aussi aux t'-termes entre deux problèmes fourchus consécutifs dans la suite $\vec{D}$. Le symbole à la racine du terme fourchu t_j est le symbole fourchu $\hat{g}_j \in \hat{G}$, le symbole à la racine du terme fourchu t_{j+1} est le symbole fourchu $\hat{g}_{j+1} \in \hat{G}$. Chaque branchement inductif d'un Fork est suivi par une application de Simplify: le t-terme $\hat{g}_j(n+1, \vec{c}; \vec{t})$ est réduit à la racine à un autre t-terme par une règle de réécriture inductive pour le symbole défini $\hat{g}_j$. Au cours de la déduction $(\mathcal{L}_j; t_j \stackrel{?}{=} t'_j) \vdash^* (\mathcal{L}_{j+1}; t_{j+1} \stackrel{?}{=} t'_{j+1})$ la règle Fork ne peut être appliquée qu'aux t'-termes. La règle Relax ne change pas les symboles définis dans les t-termes. Un branchement de la règle Decompose contient le sous-ensemble des symboles définis présents dans l'équation décomposée, alors elle ne peut pas introduire de nouveaux symboles définis. Simplify est la seule règle de transition qui change des symboles définis dans les t-termes.

Soit $\hat{g}_j(n+1, \vec{c}; \vec{x}) \to r_j$ la règle de réécriture inductive pour le symbole défini $\hat{g}_j$. Chaque symbole défini $\hat{f}$ présent dans r_j est inférieur au symbole $\hat{g}_j$ dans la précédence $\succ$. Un terme

primal ne contient qu'un nombre fini de redex, alors il n'existe qu'un nombre fini d'indices i tels que $\hat{g}_i \prec \hat{g}_{i+1}$ pour les symboles fourchus $\hat{G}$. Ceci implique qu'à partir d'un indice n_0 la suite des symboles fourchus dans $\hat{G}$ (ainsi que dans $\hat{D}$) est non-croissante par rapport à la précédence $\succ$.

A partir d'un indice n_0, les suites $\hat{G}$ et $\hat{D}$ sont non-croissantes et infinies, la précédence $\succ$ est bien-fondée, donc il existe un autre indice n_1 à partir duquel le symbole fourchu dans $\hat{G}$, ainsi que dans $\hat{D}$, reste le même. A partir de cet indice n_1, un seul symbole défini $\hat{g} \in \hat{G}$ est fourchu dans les t-termes et un seul symbole défini $\hat{d} \in \hat{D}$ est fourchu dans les t'-termes sur la branche β.

Un t-terme (resp. t'-terme) n'a qu'un nombre fini de redex qui commencent par le symbole $\hat{g}$ (resp. par le symbole $\hat{d}$). Alors à partir d'un indice n_2, le symbole fourchu $\hat{g}$ (resp. le symbole fourchu $\hat{d}$) dans le terme fourchu t'_{i+1} (resp. le terme fourchu t_{j+1}) est introduit par une application antérieure de Simplify, lorsque le terme t_i (resp. le terme t'_j) a été fourchu, par la règle inductive pour le symbole défini $\hat{g}$ (resp. pour le symbole défini $\hat{d}$).

Considérons la sous-branche β_{n_2} de β à partir de la position n_2 et considérons les suites $\vec{G}$ et $\vec{D}$ à partir de l'indice n_2. Pour chaque problème fourchu $(\mathcal{L}_i; \hat{g}(c, \vec{n}; \vec{q}) \stackrel{?}{=} t'_i) \in \vec{G}$ nous avons la déduction suivante

$$\begin{array}{lll} (\mathcal{L}_i; \hat{g}(c, \vec{n}; \vec{p}) \stackrel{?}{=} t'_i) & \vdash_{\mathsf{Fork}} & (\mathcal{L}_i \wedge c \stackrel{?}{=} c' + 1; \hat{g}(c' + 1, \vec{n}; \vec{p}) \stackrel{?}{=} t'_i) \\ & \vdash_{\mathsf{Simplify}} & (\mathcal{L}_i \wedge c \stackrel{?}{=} c' + 1; r_i \stackrel{?}{=} t'_i) \end{array}$$

Selon la construction des règles de réécriture inductives pour le symbole défini $\hat{g}$, les redex du terme r_i dans les positions A sont soit des t-termes $\hat{g}(c, \vec{n}; \vec{p})$ au renommage de variables près, soit ils diffèrent de ce renommage dans une position compteur dans le cas des compteurs liés: au lieu de n_j nous avons $n_j + 1$ pour une variable compteur $n_j \in \vec{n}$. Chaque expression compteur doit être échangée contre une nouvelle variable n'_j par Relax avant la prochaine application de la règle Fork à un t-terme. Après l'application de tous les Relax, tous les redex dans les positions A sont occupés par le t-terme $\hat{g}(c, \vec{n}; \vec{p})$ au renommage près. Quelques-uns de ces redex peuvent être perdus au cours de la décomposition mais il en reste au moins un. Ces redex ne peuvent pas être modifiés par Fork, Simplify et Relax appliqués aux t'-termes. Ceci implique que pour chaque paire de t-termes t_i et t_{i+1} (t'-termes t'_j et t'_{j+1}) de deux problèmes fourchus consécutifs dans $\vec{G}$ (dans $\vec{D}$) il existe un renommage ρ_i (un renommage ρ'_i) tel que $t_{i+1} = t_i\rho_i$ ($t'_{i+1} = t'_i\rho'_i$).

Considérons les emballages des t'-termes. Seule la règle Simplify peut augmenter la taille d'un emballage. Si un t'-terme t'_i est décomposé en un terme t'_{i+1} alors l'emballage de t'_{i+1} est un sous-terme propre de $\mathcal{W}rp(t'_i)$. Les règles de transition Fork et Relax ne changent pas les emballages. Les emballages des t'-termes dans $\vec{D}$ sont vides. Alors l'emballage maximum $\bar{w}$ (par rapport à la relation de sous-terme) des t'-termes sur la branche β_{n_2} est l'emballage introduit par la réduction par la règle de réécriture inductive pour le symbole défini $\hat{d}$ au cours de l'application de Simplify.

Considérons les t'-termes dans la suite des problèmes fourchus $\vec{G}$. Les emballages des t'-termes dans deux problèmes consécutifs de $\vec{G}$ peuvent être différents. Pourtant, il n'existe qu'un nombre fini de sous-termes propres de l'emballage maximal $\bar{w}$, contrairement au nombre de t'-termes dans $\vec{G}$ qui est infini. Alors, par le principe de Dirichlet, il existe deux

problèmes fourchus différents $(\mathcal{L}_k; t_k \stackrel{?}{=} t'_k)$ et $(\mathcal{L}_l; t_l \stackrel{?}{=} t'_l)$ dans $\vec{G}$, où $n_2 < k < l$, tels que $\mathcal{W}rp(t'_k) = \mathcal{W}rp(t'_l)$. Nous avons déjà prouvé que t'_k et t'_l ne diffèrent que dans des variables compteurs. Le t-terme t_l est obtenu à partir du terme fourchu t_k par un renommage aussi, comme nous l'avons prouvé auparavant. Les variables compteurs de t_k et t'_k ainsi que de t_l et t'_l sont disjointes, alors nous pouvons combiner les deux renommages en un seul. Ainsi, l'équation $t_l \stackrel{?}{=} t'_l$ n'est rien d'autre que l'équation $t_k \stackrel{?}{=} t'_k$ à un renommage près.

Fork et Relax sont les seules règles qui ajoutent de nouvelles équations à la partie diophantienne $\mathcal{L}$. Pour la déduction

$$(\mathcal{L}; \mathcal{P}) \quad \vdash_{\mathsf{Fork}} \quad (\mathcal{L} \wedge c \stackrel{?}{=} c' + 1; \mathcal{P}[c \mapsto c' + 1]) \quad = \quad (\mathcal{L}'; \mathcal{P}')$$

nous avons $\mathcal{L}[c \mapsto c'] = \mathcal{L}'[c \mapsto c' + 1]$. Pour la déduction

$$(\mathcal{L}; \mathcal{P}) \quad \vdash_{\mathsf{Relax}} \quad (\mathcal{L} \wedge c \stackrel{?}{=} c' - 1; \mathcal{P}[c \mapsto c' - 1]) \quad = \quad (\mathcal{L}'; \mathcal{P}')$$

nous avons $\mathcal{L}[c \mapsto c'] = \mathcal{L}'[c \mapsto c' - 1]$. Les deux substitutions $[c \mapsto c' + 1]$ et $[c \mapsto c' - 1]$ sont des translations. La composition $\kappa\kappa'$ de deux translations κ et κ' est une translation si $\mathcal{D}om(\kappa) \cap \mathcal{VR}an(\kappa') = \emptyset$. Les règles Fork et Relax introduisent toujours de nouvelles variables, la condition précédente est donc toujours satisfaite. Par conséquent, si $\mathcal{L}\rho = \mathcal{L}'\kappa$ et $\mathcal{L}'\rho' = \mathcal{L}''\kappa'$ pour les translations κ, κ' et pour les renommages ρ, ρ', alors il existe une translation $\kappa'' = \kappa\kappa'$ et un renommage $\rho'' = \rho\rho'$ tels que $\mathcal{L}\rho'' = \mathcal{L}''\kappa''$, et une translation κ et un renommage ρ tels que $\mathcal{L}_k\rho = \mathcal{L}_l\kappa$ pour les problèmes $(\mathcal{L}_k; t_k \stackrel{?}{=} t'_k)$ et $(\mathcal{L}_l; t_l \stackrel{?}{=} t'_l)$.

Fork et Relax sont les seules règles qui changent des variables compteurs en des équations $\mathcal{P}$. Si le branchement inductif de Fork ajoute une nouvelle équation diophantienne $c \stackrel{?}{=} c' + 1$ ou si la règle Relax ajoute une nouvelle équation diophantienne $c \stackrel{?}{=} c' - 1$ alors le renommage correspondant est $\rho_i = [c \mapsto c']$. Le renommage résultant entre deux problèmes fourchus consécutifs $(\mathcal{L}_i; t_i \stackrel{?}{=} t'_i)$ et $(\mathcal{L}_{i+1}; t_{i+1} \stackrel{?}{=} t'_{i+1})$ dans $\vec{G}$ est la substitution $\rho = \rho_1 \ldots \rho_n$ à condition que $\rho_1, \ldots, \rho_n$ soit la suite complète de renommages appliqués au cours de la déduction $(\mathcal{L}_i; t_i \stackrel{?}{=} t'_i) \vdash^* (\mathcal{L}_{i+1}; t_{i+1} \stackrel{?}{=} t'_{i+1})$. La composition des renommages est un renommage. Alors, il existe un renommage ρ pour les problèmes $(\mathcal{L}_k; t_k \stackrel{?}{=} t'_k)$ et $(\mathcal{L}_l; t_l \stackrel{?}{=} t'_l)$ tel que $(t_l \stackrel{?}{=} t'_l) = (t_k \stackrel{?}{=} t'_k)\rho$.

Ceci implique que le problème $(\mathcal{L}_l; t_l \stackrel{?}{=} t'_l)$ est similaire au problème $(\mathcal{L}_k; t_k \stackrel{?}{=} t'_k)$. □

L'existence d'un problème similaire sur une branche de l'arbre de surréduction nous permet d'élaguer celle-ci. Par élagage systématique des branches nous obtenons un arbre dit *élagué* qui a la bonne caractéristique d'être fini.

5.2.3 Fermeture d'un arbre élagué

Pour tout arbre infini T d'un problème séparé $(\mathcal{L}; \mathcal{P})$ il existe un arbre équivalent fini $\bar{T}$, obtenu par élagage des branches présentant une paire de problèmes similaires. Une branche est élaguée dès qu'un problème similaire à un problème antérieur est déduit. L'arbre fini $\bar{T}$ est dit l'arbre *élagué* du problème $(\mathcal{L}; \mathcal{P})$.

La quasi-solution du problème $(\mathcal{L}; \mathcal{P})$ est reconstituée en parcourant l'arbre élagué $\bar{T}$ de bas en haut. Soit $(\mathcal{L}; \mathcal{P})$ un problème fourchu et $\bar{T}$ son arbre élagué. La fermeture de l'arbre $\bar{T}$

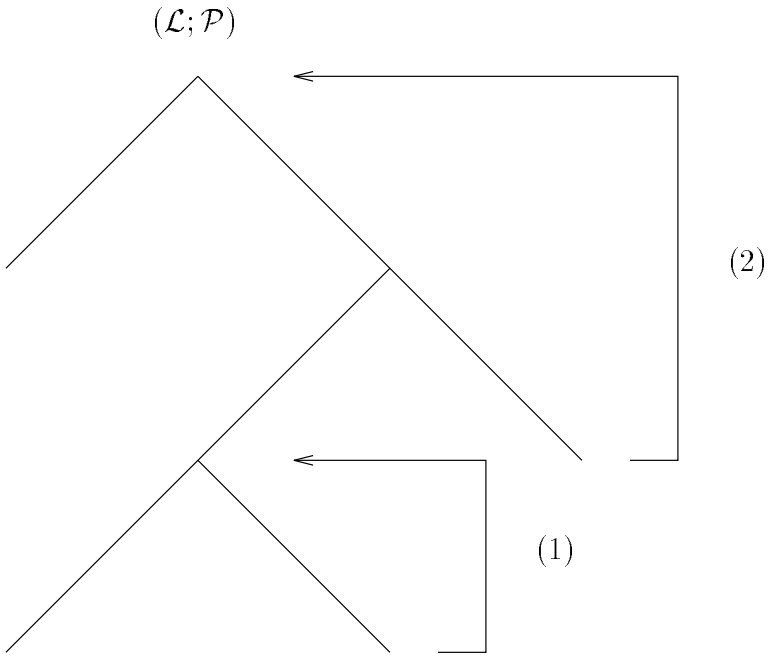


FIG. 5.5 – *Fermeture de l'arbre élagué; ordre: (1) puis (2)*

suppose que toutes les fermetures se trouvant à un niveau inférieur ont été déjà effectuées. Les quasi-solutions récupérées de ces fermetures vont servir d'éléments pour la fermeture du problème $(\mathcal{L};\mathcal{P})$. La figure 5.5 permet de mieux illustrer cela.

Les quasi-solutions provenant de la partie basique de tous les branchements Fork dans $\bar{T}$ s'appellent la *base* du problème $(\mathcal{L};\mathcal{P})$. On appelle *multiplicande* de $(\mathcal{L};\mathcal{P})$ les quasi-solutions qui proviennent des branches de la règle Decompose dont les feuilles ne sont pas des problèmes similaires à $(\mathcal{L};\mathcal{P})$, à partir de la branche inductive du dernier Fork. La définition suivante permet de définir la base et le multiplicande d'un problème de manière plus formelle.

Définition 5.35 *Soit $(\mathcal{L};\mathcal{P})$ un problème fourchu dont l'ensemble $CSCSD(\mathcal{L};\mathcal{P})$ est non-vide. Soit T l'arbre de surréduction de ce problème et $\bar{T}$ l'arbre élagué de $(\mathcal{L};\mathcal{P})$. Soit Δ l'ensemble de toutes les positions de l'ensemble $CSCSD(\mathcal{L};\mathcal{P})$ dans $\bar{T}$.*

La **base** *du problème $(\mathcal{L};\mathcal{P})$ est la formule $\mathcal{B}$ reconstituée à partir de l'arbre $\bar{T}[-]_\Delta$ dans lequel les plus proches descendants similaires à $(\mathcal{L};\mathcal{P})$ ont été remplacés par l'échec $-$.*

Le **multiplicande** *de $(\mathcal{L};\mathcal{P})$ est la formule $\neg\mathcal{B}\wedge\mathcal{G}$ en forme normale disjonctive, où $\mathcal{G}$ est la formule reconstituée à partir de $\bar{T}[\top]_\Delta$ dans lequel les plus proches descendants similaires à $(\mathcal{L};\mathcal{P})$ ont té remplacé par l'identité $\top$.*

Exemple 5.36 Soit le système Presburger $\mathcal{R}$ suivant

$$\begin{array}{rcl@{\qquad}rcl} \hat{f}(0;x) & \to & x & \hat{g}(0;x) & \to & x \\ \hat{f}(c+1;x) & \to & x * \hat{f}(c;x) & \hat{g}(c'+1;x) & \to & \hat{g}(c';x) * \hat{g}(c';x) \end{array}$$

et soit $(\top;\hat{f}(c;x) \stackrel{?}{=} \hat{g}(c';y))$ le problème séparé. La *base* de ce problème est la quasi-solution

$$(c \stackrel{?}{=} 0; x \stackrel{?}{=} \hat{g}(c';y)) \vee (c \stackrel{?}{=} c_1 + 1 \wedge c' \stackrel{?}{=} 0; y \stackrel{?}{=} x * \hat{f}(c_1;x))$$

et le *multiplicande* est la quasi-solution

$$c \stackrel{?}{=} c_1 + 1 \wedge c' \stackrel{?}{=} c'_1 + 1; x \stackrel{?}{=} \hat{g}(c'_1; y)$$

Nous avons besoin de deux opérateurs sur la base et le multiplicande pour mieux maîtriser la notation. Soit $(\mathcal{L}; \mathcal{P})$ un problème fourchu et $\kappa = [c_1 \mapsto c'_1 + k_1, \ldots, c_p \mapsto c'_p + k_p]$ une translation. Soit

$$\mathcal{B} \quad = \quad \bigvee_{j=1}^{m} \left(\bigwedge_{i=1}^{p} c_i \stackrel{?}{=} l_{ij}; \mathcal{P}_j \right)$$

la *base* et

$$\mathcal{M} \quad = \quad \bigvee_{h=1}^{m'} \left(\bigwedge_{i=1}^{p} c_i \stackrel{?}{=} l'_{ih}; \mathcal{P}'_h \right)$$

le *multiplicande* du problème $(\mathcal{L}; \mathcal{P})$. Notons

$$\mathcal{B}^n_\kappa(\mathcal{L}; \mathcal{P}) \quad = \quad \bigvee_{j=1}^{m} \left(\bigwedge_{i=1}^{p} c_i \stackrel{?}{=} l_{ij} + k_i * n; \mathcal{P}_j \right)$$

la n-ème *itération de la base* $\mathcal{B}$ par rapport à la translation κ et

$$\mathcal{M}^n_\kappa(\mathcal{L}; \mathcal{P}) \quad = \quad \bigvee_{h=1}^{m'} \left(\bigwedge_{i=1}^{p} c_i \stackrel{?}{=} l'_{ih} + k_i * n; \mathcal{P}'_h \right)$$

la n-ème *itération du multiplicande* $\mathcal{M}$ par rapport à la translation κ, où n est une nouvelle variable compteur jouant le rôle de paramètre.

La règle Decompose peut engendrer le long d'une branche un ou plusieurs PPDS. Nous allons d'abord traiter le cas d'un seul PPDS, nous verrons ensuite le cas de deux PPDS. Ce dernier se généralise directement au cas de plusieurs PPDS.

Cas du plus proche descendant similaire unique

Soit $(\mathcal{L}; \mathcal{P})$ le problème fourchu et $(\mathcal{L}'; \mathcal{P}')$ son unique PPDS. La branche se trouvant entre $(\mathcal{L}; \mathcal{P})$ et $(\mathcal{L}'; \mathcal{P}')$ peut être refermée au niveau de $(\mathcal{L}'; \mathcal{P}')$ (voir figure 5.6).

Lors de la fermeture du problème $(\mathcal{L}; \mathcal{P})$, deux cas peuvent se produire.

Cas 1. Soit le *multiplicande* de $(\mathcal{L}; \mathcal{P})$ est vide. La quasi-solution de $(\mathcal{L}; \mathcal{P})$ est le résultat de la combinaison de la *base* de $(\mathcal{L}; \mathcal{P})$ avec la translation κ entre $(\mathcal{L}; \mathcal{P})$ et son plus proche descendant similaire $(\mathcal{L}'; \mathcal{P}')$. Cette translation κ est déduite de la similarité entre $(\mathcal{L}; \mathcal{P})$ et $(\mathcal{L}'; \mathcal{P}')$. Le théorème suivant permet d'expliquer plus formellement ce cas.

Théorème 5.37 *Soit $(\mathcal{L}; \mathcal{P})$ le problème fourchu avec son unique PPDS $(\mathcal{L}'; \mathcal{P}')$, tel que $\mathcal{L} = \mathcal{L}'\kappa\rho^{-1}$ et $\mathcal{P} = \mathcal{P}'\kappa\rho^{-1}$, où κ est la translation et ρ le renommage. Si la quasi-solution $\mathcal{B}$ est la base de $(\mathcal{L}; \mathcal{P})$ et le multiplicande de $(\mathcal{L}; \mathcal{P})$ est vide, alors $\mathcal{B}^n_\kappa(\mathcal{L}; \mathcal{P})$ est la quasi-solution de $(\mathcal{L}; \mathcal{P})$.*

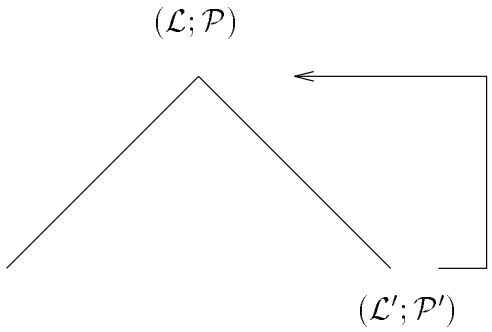


FIG. 5.6 – *Fermeture dans le cas du PPDS unique*

Preuve: Soit T l'arbre de surréduction produit par la procédure subordonnée à partir du problème fourchu $(\mathcal{L};\mathcal{P})$. Soit $(\mathcal{L}_1;\mathcal{P}_1)$ une feuille de l'arbre T déduite entre le problème $(\mathcal{L};\mathcal{P})$ et son PPDS unique $(\mathcal{L}';\mathcal{P}')$. Alors il existe une autre feuille $(\mathcal{L}'_1;\mathcal{P}'_1)$ de T, déduite à partir du PPDS $(\mathcal{L}';\mathcal{P}')$, telle que $\mathcal{L}_1\rho = \mathcal{L}'_1\kappa$ et $\mathcal{P}_1\rho = \mathcal{P}'_1\kappa$. Ceci est dû au fait qu'on peut répéter à un renommage près la déduction $(\mathcal{L};\mathcal{P}) \vdash^* (\mathcal{L}_1;\mathcal{P}_1)$ comme la déduction $(\mathcal{L}';\mathcal{P}') \vdash^* (\mathcal{L}'_1;\mathcal{P}'_1)$. De plus, les problèmes $(\mathcal{L};\mathcal{P})$ et $(\mathcal{L}';\mathcal{P}')$ sont similaires, d'où l'on déduit la similarité de $(\mathcal{L}_1;\mathcal{P}_1)$ et $(\mathcal{L}'_1;\mathcal{P}'_1)$.

La base $\mathcal{B} = \mathcal{B}^0_\kappa(\mathcal{L};\mathcal{P})$ est reconstituée à partir de toutes les feuilles déduites entre $(\mathcal{L};\mathcal{P})$ et son PPDS unique $(\mathcal{L}';\mathcal{P}')$. Le problème $(\mathcal{L}';\mathcal{P}')$ possède aussi le PPDS unique $(\mathcal{L}'';\mathcal{P}'')$. On peut répéter la même reconstitution pour $(\mathcal{L}';\mathcal{P}')$, récupérant ainsi la base de $(\mathcal{L}';\mathcal{P}')$. Comme nous l'avons démontré, pour chaque feuille déduite entre $(\mathcal{L}';\mathcal{P}')$ et $(\mathcal{L}';\mathcal{P}')$ il existe une feuille similaire déduite entre $(\mathcal{L};\mathcal{P})$ et $(\mathcal{L}';\mathcal{P}')$. Ceci implique que la base de $(\mathcal{L}';\mathcal{P}')$ est similaire à la base de $(\mathcal{L};\mathcal{P})$, le décalage est exprimé par la translation κ. Alors la base de $(\mathcal{L}';\mathcal{P}')$ est $\mathcal{B}^1_\kappa(\mathcal{L};\mathcal{P})$. Par récurrence, on obtient ainsi la suite infinie de bases

$$\mathcal{B}^0_\kappa(\mathcal{L};\mathcal{P}), \mathcal{B}^1_\kappa(\mathcal{L};\mathcal{P}), \ldots, \mathcal{B}^i_\kappa(\mathcal{L};\mathcal{P}), \ldots$$

Comme le multiplicande de $(\mathcal{L};\mathcal{P})$ est vide, aucune conjonction ne s'ajoute à la base de $(\mathcal{L}';\mathcal{P}')$ pendant la reconstitution de la quasi-solution de $(\mathcal{L};\mathcal{P})$. Par récurrence on récupère la disjonction infinie

$$\mathcal{B}^0_\kappa(\mathcal{L};\mathcal{P}) \vee \mathcal{B}^1_\kappa(\mathcal{L};\mathcal{P}) \vee \cdots \vee \mathcal{B}^i_\kappa(\mathcal{L};\mathcal{P}) \vee \cdots$$

pour $(\mathcal{L};\mathcal{P})$. Cette disjonction infinie est équivalente à $\mathcal{B}^n_\kappa(\mathcal{L};\mathcal{P})$ où n est une nouvelle variable jouant le rôle du paramètre. □

Cas 2. Soit le *multiplicande* de $(\mathcal{L};\mathcal{P})$ n'est pas vide. La quasi-solution de $(\mathcal{L};\mathcal{P})$ est la conjonction des combinaisons de la base et du multiplicande avec la translation κ. Le théorème suivant permet d'expliquer plus formellement ce cas.

Théorème 5.38 *Soit $(\mathcal{L};\mathcal{P})$ le problème fourchu avec son unique PPDS $(\mathcal{L}';\mathcal{P}')$, tel que $\mathcal{L} = \mathcal{L}'\kappa\rho^{-1}$ et $\mathcal{P} = \mathcal{P}'\kappa\rho^{-1}$, où κ est la translation et ρ le renommage. Soient les quasi-solutions $\mathcal{B}$ et $\mathcal{M}$ (respectivement la* base *et le* multiplicande *de $(\mathcal{L};\mathcal{P})$), alors*

$$\mathcal{B}^0_\kappa(\mathcal{L};\mathcal{P}) \vee (\mathcal{B}^{n+1}_\kappa(\mathcal{L};\mathcal{P}) \wedge \mathcal{M}^n_\kappa(\mathcal{L};\mathcal{P}))$$

en forme normale disjonctive est la quasi-solution de $(\mathcal{L};\mathcal{P})$.

Preuve: La reconstitution des multiplicandes s'effectue de la même façon que la reconstruction des bases. On obtient ainsi la suite des multiplicandes

$$\mathcal{M}^0_\kappa(\mathcal{L};\mathcal{P}), \mathcal{M}^1_\kappa(\mathcal{L};\mathcal{P}), \ldots, \mathcal{M}^i_\kappa(\mathcal{L};\mathcal{P}), \ldots$$

de façon analogue à la suite des bases (voir la preuve du théorème 5.37).

Tous les multiplicandes $\mathcal{M}^0_\kappa(\mathcal{L};\mathcal{P}), \ldots, \mathcal{M}^i_\kappa(\mathcal{L};\mathcal{P})$ doivent être ajoutés à la base $\mathcal{B}^{i+1}_\kappa(\mathcal{L};\mathcal{P})$ au cours de la reconstruction de la quasi-solution. Par récurrence, on obtient ainsi la formule

$$\mathcal{B}^0_\kappa(\mathcal{L};\mathcal{P}) \vee \left(\mathcal{B}^{n+1}_\kappa(\mathcal{L};\mathcal{P}) \wedge \bigwedge_{i=0}^{n} \mathcal{M}^i_\kappa(\mathcal{L};\mathcal{P})\right)$$

comme quasi-solution de $(\mathcal{L};\mathcal{P})$.

Nous allons démontrer que la conjonction $\bigwedge_{i=0}^{n} \mathcal{M}^i_\kappa(\mathcal{L};\mathcal{P})$ est équivalente à $\mathcal{M}^n_\kappa(\mathcal{L};\mathcal{P})$. Soit $c_i \stackrel{?}{=} l_i$ une équation diophantienne dans la $\mathcal{L}$-partie du multiplicande $\mathcal{M}$ où c_j est une variable compteur. La translation κ contient la substitution $c_j \mapsto c'_j + k_j$. Par conséquent, nous obtenons la suite d'équations

$$c_j \stackrel{?}{=} c'_j + k_j,\ c'_j \stackrel{?}{=} c''_j + k_j,\ \ldots, c_j^{(n-1)} \stackrel{?}{=} \tilde{c}_j + k_j \tag{5.8}$$

en construisant respectivement les multiplicandes $\mathcal{M}^1_\kappa(\mathcal{L};\mathcal{P})$, $\mathcal{M}^2_\kappa(\mathcal{L};\mathcal{P})$, ..., $\mathcal{M}^n_\kappa(\mathcal{L};\mathcal{P})$. Or la conjonction des équations (5.8) est équivalente à la conjonction

$$(c_j \stackrel{?}{=} \tilde{c}_j + n * k_j) \wedge (c'_j \stackrel{?}{=} \tilde{c}_j + (n-1) * k_j) \wedge \cdots \wedge (c_j^{(n-1)} \stackrel{?}{=} \tilde{c}_j + k_j)$$

Les variables compteurs intermédiaires c'_j, ..., $c_j^{(n-1)}$ sont introduites au cours de la production des multiplicandes et leur solution est indépendante de la solution de la variable c_j. L'équation $c_j \stackrel{?}{=} \tilde{c}_j + n * k_j$ appartient à la $\mathcal{L}$-partie du multiplicande $\mathcal{M}^n_\kappa(\mathcal{L};\mathcal{P})$. La conjonction $\bigwedge_{i=0}^{n} \mathcal{M}^i_\kappa(\mathcal{L};\mathcal{P})$ est donc équivalente à $\mathcal{M}^n_\kappa(\mathcal{L};\mathcal{P})$. □

Cas où plusieurs descendants similaires sont les plus proches

Si la règle Decompose engendre un autre PPDS $(\mathcal{L}'';\mathcal{P}'')$ le long de la branche entre le problème fourchu $(\mathcal{L};\mathcal{P})$ et son PPDS $(\mathcal{L}';\mathcal{P}')$, la fermeture de la branche au niveau de $(\mathcal{L}';\mathcal{P}')$ dépend du problème $(\mathcal{L}'';\mathcal{P}'')$. Si $(\mathcal{L}';\mathcal{P}')$ et $(\mathcal{L}'';\mathcal{P}'')$ apparaissent respectivement aux positions p' et p'', il est évident que ces deux positions sont parallèles (voir figure 5.7).

Il faut d'abord remplacer le PPDS $(\mathcal{L}'';\mathcal{P}'')$ dans l'arbre élagué $\bar{T}$ par l'identité $\top$ et calculer la fermeture $\mathcal{A}_1$ pour $(\mathcal{L}';\mathcal{P}')$ en utilisant l'unicité du PPDS. Ensuite, il faut remplacer le PPDS $(\mathcal{L}';\mathcal{P}')$ dans l'arbre élagué $\bar{T}$ par l'identité et calculer la fermeture $\mathcal{A}_2$ pour $(\mathcal{L}'';\mathcal{P}'')$ en utilisant la méthode du PPDS unique. La quasi-solution du problème $(\mathcal{L};\mathcal{P})$ sera alors la conjonction des deux quasi-solutions calculées, transformée en forme normale disjonctive.

Théorème 5.39 *Soit* $(\mathcal{L};\mathcal{P})$ *un problème fourchu avec deux PPDS* $(\mathcal{L}';\mathcal{P}')$ *et* $(\mathcal{L}'';\mathcal{P}'')$. *Soit* $\mathcal{A}_1$ *la quasi-solution à partir de l'arbre élagué où nous avons remplacé* $(\mathcal{L}'';\mathcal{P}'')$ *par l'identité* $\top$. *Soit* $\mathcal{A}_2$ *la quasi-solution à partir de l'arbre élagué où nous avons remplacé* $(\mathcal{L}';\mathcal{P}')$ *par l'identité* $\top$. *Alors la quasi-solution de* $(\mathcal{L};\mathcal{P})$ *est la forme normale disjonctive de* $\mathcal{A}_1 \wedge \mathcal{A}_2$.

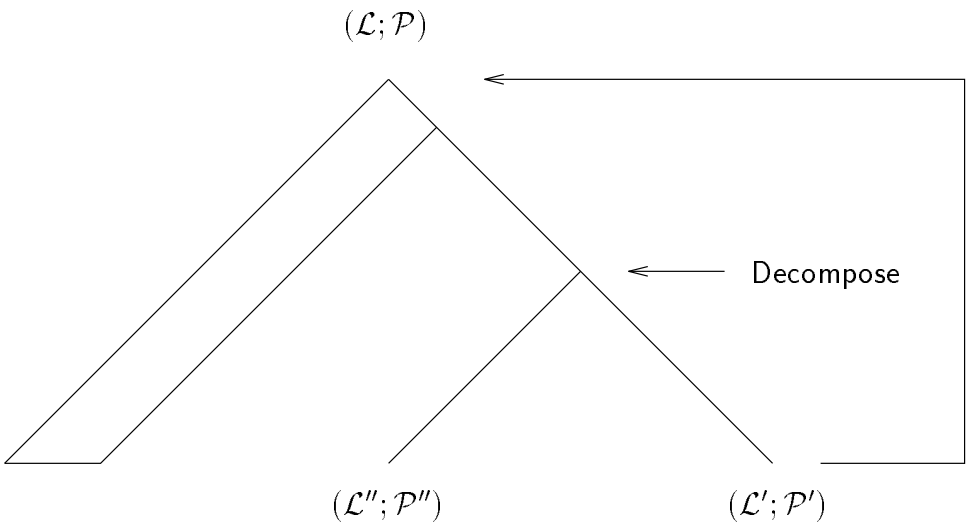


FIG. 5.7 – *Fermeture dans le cas de deux PPDS*

Preuve: Supposons que la fermeture n'est pas effectuée au niveau du problème $(\mathcal{L};\mathcal{P})$, mais nous avons construit deux fermetures aux niveaux inférieurs: l'une au niveau du problème $(\mathcal{L}';\mathcal{P}')$, l'autre au niveau du problème $(\mathcal{L}'';\mathcal{P}'')$. Soit $\mathcal{A}_1$ la quasi-solution obtenue par la fermeture de $(\mathcal{L}';\mathcal{P}')$ et $\mathcal{A}_2$ la quasi-solution obtenue par la fermeture de $(\mathcal{L}'';\mathcal{P}'')$.

Il n'y a plus de branchement alternatif entre le Decompose après le dernier Fork et les problèmes $(\mathcal{L}';\mathcal{P}')$ et $(\mathcal{L}'';\mathcal{P}'')$ respectivement. Soit $\mathcal{B}_1$ (resp. $\mathcal{B}_2$) la formule récupérée sur les branches secondaires par la reconstruction entre $(\mathcal{L}';\mathcal{P}')$ (resp. $(\mathcal{L}'';\mathcal{P}'')$) et le Decompose distingué. La quasi-solution reconstituée au niveau du Decompose est alors $(\mathcal{A}_1 \wedge \mathcal{B}_1) \wedge (\mathcal{A}_2 \wedge \mathcal{B}_2)$ ce qui est logiquement équivalent à

$$((\mathcal{A}_1 \wedge \mathcal{B}_1) \wedge (\top \wedge \mathcal{B}_1)) \wedge ((\top \wedge \mathcal{B}_1) \wedge (\mathcal{A}_2 \wedge \mathcal{B}_2))$$

Or $(\mathcal{A}_1 \wedge \mathcal{B}_1) \wedge (\top \wedge \mathcal{B}_1)$ est la quasi-solution récupérée dans le cas où nous avons remplacé $(\mathcal{L}'';\mathcal{P}'')$ par l'identité et $(\top \wedge \mathcal{B}_1) \wedge (\mathcal{A}_2 \wedge \mathcal{B}_2)$ est la quasi-solution récupérée dans le cas où nous avons remplacé $(\mathcal{L}';\mathcal{P}')$ par l'identité. □

L'opération de conjonction étant associative, le théorème précédent s'étend naturellement au cas où plusieurs descendants similaires sont les plus proches. Il faut à chaque fois retenir un des PPDS, remplacer les autres par l'identité et calculer la fermeture individuelle. La quasi-solution du problème $(\mathcal{L};\mathcal{P})$ est alors la conjonction des fermetures individuelles, transformée en sa forme normale disjonctive.

Soit κ' et κ' les translations du problème $(\mathcal{L};\mathcal{P})$ au PPDS $(\mathcal{L}';\mathcal{P}')$ et $(\mathcal{L}'';\mathcal{P}'')$, respectivement. Nous ne sommes obligés de considérer que des cas où les translations κ' et κ'' sont différentes. Si $\kappa'' = \kappa'\rho$ pour un renommage ρ, les deux PPDS $(\mathcal{L}';\mathcal{P}')$ et $(\mathcal{L}'';\mathcal{P}'')$ sont équivalents modulo le renommage ρ et leurs fermetures seront, elles aussi, équivalentes au renommage près.

Nota: Il n'est pas nécessaire de transformer la formule en forme normale disjonctive au cours de chaque étape de la reconstitution de la quasi-solution. Cependant, il est indispen-

sable de rendre la quasi-solution en forme normale disjonctive à la procédure principale. Il suffit donc d'effectuer une seule transformation à la fin de la reconstruction.

Pour bien comprendre l'algorithme d'unification présenté, considérons l'exemple suivant.

Exemple 5.40 (Suite de l'Exemple 5.36)
Le problème

$$\hat{f}(c;x) \stackrel{?}{=} \hat{g}(c';y)$$

est directement présenté à la procédure subordonnée. Un seul PPDS est rencontré en utilisant la translation

$$\kappa = [c \mapsto c_1 + 1, c' \mapsto c'_1 + 1]$$

et le renommage

$$\rho = [c \mapsto c_1, c' \mapsto c'_1]$$

La fermeture de l'arbre élagué produit, d'après le théorème 5.38, la formule

$$\begin{array}{l}(c \stackrel{?}{=} 0; x \stackrel{?}{=} \hat{g}(c';y)) \vee \\ (c \stackrel{?}{=} c_1 + 1 \wedge c' \stackrel{?}{=} 0; y \stackrel{?}{=} x * \hat{f}(c_1;x)) \vee \\ ((c \stackrel{?}{=} n+1; x \stackrel{?}{=} \hat{g}(c';y)) \vee (c \stackrel{?}{=} c_2 + n + 2 \wedge c' \stackrel{?}{=} n+1; y \stackrel{?}{=} x * \hat{f}(c_2;x))) \wedge \\ (c \stackrel{?}{=} c_1 + n + 1 \wedge c' \stackrel{?}{=} c'_1 + n + 1; x \stackrel{?}{=} \hat{g}(c'_1;y))\end{array}$$

où n est la nouvelle variable compteur jouant le rôle du paramètre. Cette formule est transformée en forme normale disjonctive et rendue à la procédure principale.

La partie

$$(c \stackrel{?}{=} 0; x \stackrel{?}{=} \hat{g}(c';y)) \quad \vee \quad (c \stackrel{?}{=} c_1 + 1 \wedge c' \stackrel{?}{=} 0; y \stackrel{?}{=} x * \hat{f}(c_1;x)) \tag{5.9}$$

de la quasi-solution récupérée est déjà en forme résolue. Il reste à résoudre les clauses

$$(c \stackrel{?}{=} n+1 \wedge c \stackrel{?}{=} c_1 + n + 1 \wedge c' \stackrel{?}{=} c'_1 + n + 1; x \stackrel{?}{=} \hat{g}(c';y) \wedge x \stackrel{?}{=} \hat{g}(c'_1;y)) \tag{5.10}$$

et

$$(c \stackrel{?}{=} c_1+n+1 \wedge c \stackrel{?}{=} c_2+n+2 \wedge c' \stackrel{?}{=} c'_1+n+1 \wedge c' \stackrel{?}{=} n+1; x \stackrel{?}{=} \hat{g}(c'_1;y) \wedge y \stackrel{?}{=} x * \hat{f}(c_2;x)) \tag{5.11}$$

Par Eliminate à partir de la clause (5.11) dans la $\mathcal{P}$-partie la procédure principale produit la formule

$$x \stackrel{?}{=} \hat{g}(c'_1; x * \hat{f}(c_2;x)) \quad \wedge \quad y \stackrel{?}{=} x * \hat{f}(c_2;x)$$

Le problème séparé $x \stackrel{?}{=} \hat{g}(c'_1; x * \hat{f}(c_2;x))$ envoyé à la procédure subordonnée conduit après un Fork à l'échec, donc il n'y a pas de solution pour la clause (5.11).

Par Eliminate à partir de la clause (5.10) la procédure principale produit la formule

$$(c \stackrel{?}{=} n+1 \wedge c \stackrel{?}{=} c_1+n+1 \wedge c' \stackrel{?}{=} c'_1+n+1; x \stackrel{?}{=} \hat{g}(c';y) \wedge \hat{g}(c';y) \stackrel{?}{=} \hat{g}(c'_1;y)) \qquad (5.12)$$

Le problème séparé $\hat{g}(c';y) \stackrel{?}{=} \hat{g}(c'_1;y)$ est envoyé à la procédure subordonnée qui rend la quasi-solution $(c' \stackrel{?}{=} k \wedge c'_1 \stackrel{?}{=} k; \top)$ où k est la nouvelle variable compteur. La formule (5.12) après le retour de la procédure subordonnée devient

$$(c \stackrel{?}{=} n+1 \wedge c \stackrel{?}{=} c_1+n+1 \wedge c' \stackrel{?}{=} c'_1+n+1 \wedge c' \stackrel{?}{=} k \wedge c'_1 \stackrel{?}{=} k; x \stackrel{?}{=} \hat{g}(c';y))$$

L'equation $x \stackrel{?}{=} \hat{g}(c';y)$ est irréductible, il faut donc résoudre le système d'équations diophantiennes suivant

$$\begin{array}{lll} c & \stackrel{?}{=} & n+1 \\ c & \stackrel{?}{=} & c_1+n+1 \\ c' & \stackrel{?}{=} & c'_1+n+1 \end{array} \qquad \begin{array}{lll} c' & \stackrel{?}{=} & k \\ c'_1 & \stackrel{?}{=} & k \end{array}$$

On en déduit l'équation $k \stackrel{?}{=} k+n+1$ qui n'a pas de solution en entiers non-négatifs. La solution du problème d'unification primale $\hat{f}(c;x) \stackrel{?}{=} \hat{g}(c';y)$ est la disjonction (5.9).

5.2.4 Terminaison et correction de l'algorithme d'unification

Théorème 5.41 (Correction subordonnée) *La procédure subordonnée combinée avec la fermeture de l'arbre élagué calcule la quasi-solution complète de tout problème d'unification séparé et linéarisé.*

Preuve: Les règles de transition Delete, Decompose, Conflict et Check ont déjà été prouvées correctes pour l'unification syntaxique.

La linéarité des compteurs dans chaque équation $t \stackrel{?}{=} t'$ reste invariante pendant l'application de toutes les règles de transition.

Les problèmes séparés $t \stackrel{?}{=} t'$ sont linéarisés, il est alors correct d'appliquer Fork seulement à un côté des équations; l'autre côté reste inchangé car il n'y a pas de variable compteur commune.

Il existe une règle de réécriture basique et une règle inductive pour chaque symbole défini, alors les branches, basique et inductive, du Fork sont suffisantes pour le filtrage exhaustif des cas.

L'application de chaque règle de transition préserve les unificateurs, en rappellant implicitement les conjonctions et disjonctions produites par Decompose et Fork.

Soit $t \rightsquigarrow_{\mathcal{R}} t'$ un pas de surréduction avec le système Presburger $\mathcal{R}$. Si $t \rightsquigarrow_{\mathcal{R}} t'$ est un pas extérieur de surréduction, alors il existe une position redex $p \in \mathcal{DP}os(t)$ et une règle de réécriture $\hat{f}(e,\vec{c};\vec{x}) \to r$ telles que $t|_p = \hat{f}(c,\vec{n};\vec{t})$ avec le filtre $\sigma = [c \mapsto e, \vec{c} \mapsto \vec{n}; \vec{x} \mapsto \vec{t}]$ et $t' = t\sigma[r\sigma]_p$, où $e = 0$ ou $e = k+1$ pour une variable compteur k. Le filtre σ peut être restreint à la variable c s'il est appliqué au contexte $t[\cdot]_p$. Ceci implique $t'[\cdot]_p = t[c \mapsto e][\cdot]_p$. Or p est une position redex et c est une variable compteur, alors les emballages de t et t' sont égaux: $\mathcal{W}rp(t) = \mathcal{W}rp(t')$.

En utilisant ces résultats, si $t \leadsto_{\mathcal{R}} t'$ alors pour chaque problème d'unification séparé $t \stackrel{?}{=} t''$ nous avons

– soit

$$(t \stackrel{?}{=} t'') \vdash^*_{\mathsf{Decompose}} (t|_p \stackrel{?}{=} t''|_p) \vdash_{\mathsf{Fork}} (t|_p[c \mapsto e] \stackrel{?}{=} t''|_p) \vdash_{\mathsf{Simplify}} (r\sigma \stackrel{?}{=} t''|_p)$$

si $p \in \mathcal{FP}os(t'')$,

– soit

$$\begin{array}{llll}(t \stackrel{?}{=} t'') & \vdash^*_{\mathsf{Decompose}} & (t|_a \stackrel{?}{=} x) & \vdash_{\mathsf{Bang}} (t|_a[t|_p]_b[c \mapsto 0] \stackrel{?}{=} x) \vdash_{\mathsf{Simplify}} \\ & \vdash_{\mathsf{Simplify}} & (t|_a[c \mapsto 0][r\sigma]_b \stackrel{?}{=} x) & \end{array}$$

si $p \notin \mathcal{FP}os(t'')$, où $p = a.b$ et $t''|_a = x$.

Par conséquent, la procédure subordonnée construit l'arbre complet de surréduction T pour chaque problème séparé et linéarisé $t \stackrel{?}{=} t'$. Pour chaque arbre complet de surréduction T il existe un arbre élagué unique $\bar{T}$, d'après le théorème 5.34. Chaque arbre élagué est fermé itérativement de bas en haut suivant les théorèmes 5.37, 5.38 et 5.39. Cette reconstruction en utilisant les fermetures entre les problèmes fourchus et leurs PPDS rend toujours une formule finie, car l'arbre élagué ne contient qu'un nombre fini de feuilles. La formule rendue, transformée en forme normale disjonctive, est la quasi-solution du problème séparé $t \stackrel{?}{=} t'$. □

Lemme 5.42 (Terminaison principale) *L'algorithme d'unification primale termine pour tout problème d'unification primale.*

Preuve: Chaque appel de la procédure subordonnée, avec un problème séparé et linéarisé, combiné avec la fermeture de l'arbre élagué, termine par le Théorème 5.34, grâce aux branchements finis dans l'arbre élagué et grâce au nombre fini de fermetures de l'arbre élagué. Il est clair aussi que la procédure principale, sans appel de la procédure subordonnée, termine car

1. l'algorithme d'unification syntaxique termine,
2. $\mathcal{R}$ est un système nœthérien alors la relation $\longrightarrow_{\mathcal{R}}$ est bien fondée, et
3. l'algorithme de résolution d'un système d'équations diophantiennes linéaires termine.

La seule possibilité de non-terminaison de l'algorithme est une possibilité de va-et-viens infini entre la procédure principale et la procédure subordonnée.

Pour chaque problème fourchu $t \stackrel{?}{=} t'$ la procédure subordonnée construit l'arbre élagué $\bar{T}$. Cet arbre élagué contient un ensemble fini de feuilles $(\mathcal{L}_1; x_1 \stackrel{?}{=} t_1)$, ..., $(\mathcal{L}_k; x_k \stackrel{?}{=} t_k)$ en provenance des branches non-élaguées. Soit $\mathcal{A}$ la quasi-solution engendrée par la procédure subordonnée pour le problème fourchu $(\top; t \stackrel{?}{=} t')$. L'ensemble $\mathcal{A}_{\mathcal{P}}$ est formé des seconds éléments des pairs de l'ensemble $\mathcal{A}$. $\mathcal{A}_{\mathcal{P}}$ est une formule propositionnelle en forme normale

disjonctive dont les atomes sont les $\mathcal{P}$-parties des feuilles de l'arbre élagué, c'est-à-dire $x_1 \stackrel{?}{=} t_1, \ldots, x_k \stackrel{?}{=} t_k$. Nous allons démontrer que le problème fourchu $t \stackrel{?}{=} t'$ est "plus complexe" que la formule $\mathcal{A}_\mathcal{P}$. Pour prouver ceci, il suffit de démontrer que $t \stackrel{?}{=} t'$ est supérieur à chaque feuille $x_i \stackrel{?}{=} t_i$.

Considérons le problème séparé $t \stackrel{?}{=} t'$ comme le terme t_0 dans la signature enrichie par le symbole $\stackrel{?}{=}$. Chaque variable ordinaire x_i est présente dans le terme t_0, alors pour chaque i, $t_0 \succ x_i$ pour n'importe quel ordre de simplification.

La construction de l'arbre élagué pour t_0 implique

$$\forall i \in \{1, \ldots, k\} \quad \exists \xi_i \quad t_0\xi_i \xrightarrow{*}_\mathcal{R} \vdash^*_{\mathsf{Decompose}} (x_i \stackrel{?}{=} t_i) \tag{5.13}$$

Soit $\succ$ l'ordre de simplification utilisé pour démontrer la terminaison du système Presburger $\mathcal{R}$. Alors, (5.13) implique

$$\forall i \in \{1, \ldots, k\} \quad \exists \xi_i \quad t_0\xi_i \succ t_i \tag{5.14}$$

Les énumérateurs ξ_i instancient des variables compteurs de t_0 par les entiers naturels. L'ordre des variables compteurs dans t_0 est fixé alors on considère le même ordre de variables compteurs instanciées dans chaque ξ_i. Ceci permet de trier les énumérateurs $\xi_1 < \cdots < \xi_k$ et par conséquent aussi les instances de t_0:

$$t_0\xi_1 < \cdots < t_0\xi_k \tag{5.15}$$

La combinaison de (5.14) et (5.15) indique que

1. pour chaque énumérateur ξ_i, l'ensemble des termes t_j tels que $t_0\xi_i \not\succ t_j$ est fini;
2. pour chaque énumérateur ξ_i, il existe un énumérateur $\xi_j > \xi_i$ tel que $t_0\xi_i > t_0\xi_j$.

On en déduit que $t_0 \succ' t_i$ pour chaque terme t_i, pour un ordre de simplification $\succ'$, et par conséquent, $t_0 \succ' \mathcal{A}_\mathcal{P}$, ce qui est étendu à

$$(t \stackrel{?}{=} t') \quad \succ' \quad \mathcal{A} \tag{5.16}$$

Le multi-ensemble des variables ordinaires dans le problème fourchu $t \stackrel{?}{=} t'$ et sa quasi-solution $\mathcal{A}$ sont invariants. Par contre, chaque application de la règle Eliminate diminue la taille du multi-ensemble des variables ordinaires du problème d'unification. Or, seule la règle Eliminate peut créer de nouveaux problèmes séparés à traiter par la procédure subordonnée. L'ordre de simplification $\succ'$ est bien fondé et le multi-ensemble des variables ordinaires décroît pendant l'application de la règle Eliminate, donc il n'y a pas un nombre infini de va-et-viens entre la procédure principale et la procédure subordonnée. Par conséquent, l'algorithme d'unification primale termine pour chaque problème primal. □

Théorème 5.43 (Correction principale) *L'algorithme d'unification primale calcule la solution complète de chaque problème d'unification primale $\mathcal{P}$. Cette solution présente la disjonction finie de l'ensemble complet d'unificateurs globaux du problème $\mathcal{P}$.*

Preuve: Les formes résolues sont les formes normales des problèmes d'unification primale traités par la procédure principale.

D'après le Théorème 5.41, chaque appel de la procédure subordonnée calcule une quasi-solution avec un nombre fini de formes pré-résolues. Selon le lemme 5.42, il n'y a qu'un nombre fini d'appels de la procédure subordonnée. Alors la solution de chaque problème d'unification primale, engendrée par la procédure principale, contient un nombre fini de formes résolues. Chaque forme résolue

$$\left(\bigwedge_{i=1}^{p} c_i \stackrel{?}{=} l_i;\ \bigwedge_{j=1}^{q} x_j \stackrel{?}{=} t_j \right)$$

correspond à l'unificateur global

$$[c_1 \mapsto l_1, \ldots, c_p \mapsto l_p;\quad x_1 \mapsto t_1, \ldots, x_q \mapsto t_q]$$

Chaque système Presburger $\mathcal{R}$ est canonique, donc la surréduction par $\mathcal{R}$ est complète. Selon le théorème 5.41, la quasi-solution de chaque problème fourchu, calculée par la procédure subordonnée, est complète pour ce problème fourchu. Les règles de transition pour l'unification syntaxique sont complètes pour l'unification des termes avec constructeurs. L'algorithme de résolution des systèmes d'équations diophantiennes linéaires engendre la solution complète de chaque système. Alors, l'algorithme d'unification principale calcule la solution complète de chaque problème d'unification primal. □

Corollaire 5.44 *L'unification primale est décidable.*

Conclusion

Dans ce document, nous avons abordé différents aspects de la divergence de la procédure de complétion des systèmes de réécriture. Le premier chapitre a introduit le problème et présente la preuve de son indécidabilité.

Le deuxième chapitre a été consacré à la reconnaissance de systèmes de réécriture divergents. Il commençait par une étude d'une opération spéciale, intitulée produit restreint, utilisée tout au long de ce chapitre. Le chapitre poursuivait avec l'étude des fermetures de règles, un outil permettant de manipuler une suite de superpositions comme un seul objet. Les fermetures répétitives, intitulées chaînes de fermeture, sont déduites comme l'objet-clé pour la reconnaissance de la divergence. Sur sa base sont déduits les systèmes de réécriture croisés, les outils principaux pour la reconnaissance de la divergence. Le chapitre s'achève par l'étude de la divergence de la complétion en présence de simplifications entre les règles de réécriture produites. Y a été ajouté le résumé du travail de Mong et Purdom sur la reconnaissance de la divergence.

Le troisième chapitre a présenté différents moyens empiriques pour éviter la divergence. Il s'agit notamment de la modification de l'ordre à l'intérieur d'une même classe d'ordres, du choix d'une autre classe d'ordres, de la division des chaînes de fermeture, de la décomposition des chaînes de fermeture et enfin de l'enrichissement du système divergent par un lemme inductif. Les trois premières méthodes sont applicables au cours de chaque procédure de complétion, tandis que les deux dernières ne peuvent être utilisées qu'au cours de preuves de théorèmes inductifs.

Le quatrième chapitre a abordé le vaste sujet des formalismes pour maîtriser la divergence, appelés les schématisations. Plusieurs schématisations ont été présentées, notamment les méta-règles d'Hélène Kirchner, les schémas de termes de Bernhard Gramlich, les contraintes d'appartenance avec variables de contexte d'Hubert Comon, les termes récurrents de Jieh Hsiang et Hong Chen, avec leurs généralisations d'Hubert Comon et de Gernot Salzer, et en fin les grammaires primales de Miki Hermann. Les quatre dernières appartiennent á la classe des schématisations récurrentes. En particulier, une méthode pour engendrer une schématisation par des méta-règles et une autre pour engendrer une schématisation par des grammaires primales à partir d'un système divergent ont été présentées. Les schématisations ont acquis un statut en logique équationnelle et ont été reconnues comme un problème important. Ceci est vrai surtout pour les schématisations récurrentes dont le problème d'unification est décidable. L'étude d'un algorithme d'unification des grammaires primales, les plus puissantes schématisations récurrentes connues au moment de l'écriture de ce document, a clos le dernier chapitre.

Néanmois, plusieurs aspects n'ont pourtant pas été abordés dans ce document. Tout

d'abord, la complétion sans échec. Malgré des différences entre les deux procédures de complétion, nous ne pouvons pas espérer obtenir de résultats bien différents de ceux présentés ici. Un autre axe de recherche, beaucoup plus intéressant sur le plan de la divergence, est celui de la complétion modulo une théorie équationnelle. Nous sommes confrontés dans ce cas à deux types de divergence: la divergence propre du système de réécriture liée à l'explosion du nombre d'unificateurs principaux dans les théories finitaires d'une part, et à la divergence due à l'infinité d'unificateurs principaux dans les théories infinitaires d'autre part. Le dernier sujet que nous n'avons pas abordé est celui de la divergence de la surréduction. Cependant, la première partie du deuxième chapitre en prépare le terrain. Des travaux sur ce sujet ont été entrepris par Stefan Kurtz [Kur92].

Bibliographie

[AHL93] A. Amaniss, M. Hermann, and D. Lugiez. Etude comparative des méthodes de schématisation de séquences infinies de termes du premier ordre. Research report 93-R-114, Centre de Recherche en Informatique de Nancy, 1993. France.

[Ama92] A. Amaniss. Unification des termes récurrents. Master's thesis, Centre de Recherche en Informatique de Nancy, September 1992.

[Ard80] M.A. Ardis. Data abstraction transformations. Technical report TR-925, University of Maryland, Maryland (USA), 1980.

[Ave91] J. Avenhaus. Proving equational and inductive theorems by completion and embedding techniques. In R.V. Book, editor, *Proceedings 4th Conference on Rewriting Techniques and Applications (RTA'91), Como (Italy)*, volume 488 of *Lecture Notes in Computer Science*, pages 361–373. Springer-Verlag, April 1991.

[Baa91] F. Baader. Unification, weak unification, upper bound, lower bound, and generalization problems. In R.V. Book, editor, *Proceedings 4th Conference on Rewriting Techniques and Applications (RTA'91), Como (Italy)*, volume 488 of *Lecture Notes in Computer Science*, pages 86–97. Springer-Verlag, April 1991.

[Bac88] L. Bachmair. Proof by consistency in equational theories. In *Proceedings 3rd IEEE Symposium on Logic in Computer Science (LICS'88), Edinburgh (Scotland)*, pages 228–233, July 1988.

[Bac91] L. Bachmair. *Canonical equational proofs*. Birkhäuser, Boston, 1991.

[BD86] L. Bachmair and N. Dershowitz. Commutation, transformation and termination. In J.H. Siekmann, editor, *Proceedings 8th International Conference on Automated Deduction (CADE'86), Oxford (England)*, volume 230 of *Lecture Notes in Computer Science*, pages 5–20. Springer-Verlag, July 1986.

[BD87] L. Bachmair and N. Dershowitz. Completion for rewriting modulo a congruence. In P. Lescanne, editor, *Proceedings 2nd Conference on Rewriting Techniques and Applications (RTA'87), Bordeaux (France)*, volume 256 of *Lecture Notes in Computer Science*, pages 192–203. Springer-Verlag, May 1987.

[BDH86] L. Bachmair, N. Dershowitz, and J. Hsiang. Orderings for equational proofs. In *Proceedings 1st IEEE Symposium on Logic in Computer Science (LICS'86), Cambridge, (Massachusetts, USA)*, pages 346–357, June 1986.

[Bel86] F. Bellegarde. Rewriting systems on FP expressions to reduce the number of sequences yielded. *Science of Computer Programming*, 6(1):11–34, January 1986.

[BL87a] F. Bellegarde and P. Lescanne. Transformation ordering. In H. Ehrig, R. Kowalski, G. Levi, and U. Montanari, editors, *Proceedings of CAAP '87, Pisa (Italy)*, volume 249 of *Lecture Notes in Computer Science*, pages 69–80. TAPSOFT '87, volume 1, Springer-Verlag, March 1987.

[BL87b] A. BenCherifa and P. Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *Science of Computer Programming*, 9(2):137–159, October 1987.

[CD91] E. Contejean and H. Devie. Résolution de systèmes linéaires d'équations diophantiennes. *Compte-rendus de l'Académie des Sciences de Paris*, 1991.

[CH91a] H. Chen and J. Hsiang. Logic programming with recurrence domains. In J. Leach Albert, B. Monien, and M. Rodríguez Artalejo, editors, *Proceedings 18th ICALP Conference, Madrid (Spain)*, volume 510 of *Lecture Notes in Computer Science*, pages 20–34. Springer-Verlag, July 1991.

[CH91b] H. Chen and J. Hsiang. Recurrence domains: Their unification and application to logic programming. Technical report, Departement of Computer Science, SUNY at Stony Brook, 1991.

[CHK90] H. Chen, J. Hsiang, and H.-C. Kong. On finite representations of infinite sequences of terms. In S. Kaplan and M. Okada, editors, *Proceedings 2nd International Workshop on Conditional and Typed Rewriting Systems (CTRS'90), Montreal (Canada)*, volume 516 of *Lecture Notes in Computer Science*, pages 100–114. Springer-Verlag, June 1990.

[Com92] H. Comon. Completion of rewrite systems with membership constraints. In W. Kuich, editor, *Proceedings 19th ICALP Conference, Wien (Austria)*, volume 623 of *Lecture Notes in Computer Science*, pages 392–403. Springer-Verlag, July 1992. Exists also as Research report 699, LRI, Orsay.

[Com95] H. Comon. On unification of terms with integer exponents. *Mathematical Systems Theory*, 28(1):67–88, 1995.

[Dau88] M. Dauchet. Termination of rewriting is undecidable in the one rule case. In M. Chytil, L. Janiga, and V. Koubek, editors, *Proceedings of the 14th Matematical Foundations of Computer Science, Carlsbad (Czechoslovakia)*, volume 324 of *Lecture Notes in Computer Science*, pages 262–268. Springer-Verlag, 1988.

[Dau89] M. Dauchet. Simulation of Turing machines by a left-linear rewrite rule. In N. Dershowitz, editor, *Proceedings 3rd Conference on Rewriting Techniques and Applications (RTA'89), Chapel Hill, (North Carolina, USA)*, volume 355 of *Lecture Notes in Computer Science*, pages 109–120. Springer-Verlag, April 1989.

[Der87] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1 & 2):69–116, 1987. Special issue on Rewriting Techniques and Applications.

[Der89] N. Dershowitz. Completion and its applications. In H. Aït-Kaci and M. Nivat, editors, *Proceedings of Resolution of Equations in Algebraic Structures, Lakeway, (Texas, USA); Volume 2: Rewriting Techniques*, pages 31–86. MCC Corporation & INRIA, Academic Press, 1989.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science B: Formal Methods and Semantics*, chapter 6, pages 243–309. Elsevier, Amsterdam, 1990.

[DJ91] N. Dershowitz and J.-P. Jouannaud. Notations for rewriting. *Bulletin of the European Association for Theoretical Computer Science*, 43:162–172, February 1991.

[Dje93] L. Djerid. Etude et implantation de l'unification des grammaires primales. Master's thesis, Université de Nancy I, 1993.

[DM79] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of ACM*, 22:465–476, 1979.

[Dom91] E. Domenjoud. Solving systems of linear diophantine equations: An algebraic approach. In A. Tarlecki, editor, *Proceedings 16th International Symposium on Mathematical Foundations of Computer Science (MFCS'91), Kazimierz Dolny (Poland)*, volume 520 of *Lecture Notes in Computer Science*, pages 141–150. Springer-Verlag, September 1991.

[DP90] N. Dershowitz and E. Pinchover. Inductive synthesis of equational programs. In *Proceedings 8th National Conference on AI*, pages 234–239. American Association for Artificial Intelligence, 1990.

[Ede85] E. Eder. Properties of substitutions and unifications. *Journal of Symbolic Computation*, 1(1):31–46, 1985.

[Fay79] M. Fay. First-order unification in an equational theory. In S. Sickel, editor, *Proceedings of the 4th Workshop on Automated Deduction, Austin (Texas, USA)*, pages 161–167, February 1979.

[FH86] F. Fages and G. Huet. Complete sets of unifiers and matchers in equational theories. *Theoretical Computer Science*, 43(1):189–200, 1986.

[Fri89] L. Fribourg. A strong restriction of the inductive completion procedure. *Journal of Symbolic Computation*, 8(3):253–276, September 1989.

[GH92] R. Galbavý and M. Hermann. Unification of infinite sets of terms schematized by primal grammars. Research report 92-R-220, Centre de Recherche en Informatique de Nancy, Nancy, France, 1992.

[GKK90] G. Gnaedig, C. Kirchner, and H. Kirchner. Equational completion in order-sorted algebras. *Theoretical Computer Science*, 72:169–202, 1990.

[GKM83] J.V. Guttag, D. Kapur, and D.R. Musser. On proving uniform termination and restricted termination of rewrite systems. *SIAM Journal on Computing*, 12(1):189–214, February 1983.

[GO91] R. Giegerich and E. Ohlebusch. An implicite representation of infinite sequences of terms. *Bulletin of the European Association for Theoretical Computer Science*, 43:173–182, February 1991.

[Göb87] R. Göbel. Ground confluence. In P. Lescanne, editor, *Proceedings 2nd Conference on Rewriting Techniques and Applications (RTA'87), Bordeaux (France)*, volume 256 of *Lecture Notes in Computer Science*, pages 156–167. Springer-Verlag, May 1987.

[Gra88] B. Gramlich. Unification of term schemes - theory and applications. SEKI Report SR-88-18, Universität Kaiserslautern, Germany, 1988.

[GS84] M. Gécseg and M. Steinby. *Tree automata*. Akademiai kiadó, Budapest, Hungary, 1984.

[Her89] M. Hermann. Chain properties of rule closures. In B. Monien and R. Cori, editors, *Proceedings 6th Symposium on Theoretical Aspects of Computer Science (STACS'89), Paderborn (Germany)*, volume 349 of *Lecture Notes in Computer Science*, pages 339–347. Springer-Verlag, February 1989.

[Her90a] M. Hermann. Chain properties of rule closures. *Formal Aspects of Computing*, 2(3):207–225, 1990.

[Her90b] M. Hermann. Divergent term rewriting systems. Research report 90-R-037, Centre de Recherche en Informatique de Nancy, February 1990. Submitted.

[Her90c] M. Hermann. Vademecum of divergent term rewriting systems. In *"Avancées en Programation" – Journées AFCET-GROPLAN, Nice (France)*, volume 70, pages 148–164. BIGRE, January 1990.

[Her91] M. Hermann. On proving properties of completion strategies. In R.V. Book, editor, *Proceedings 4th Conference on Rewriting Techniques and Applications (RTA'91), Como (Italy)*, volume 488 of *Lecture Notes in Computer Science*, pages 398–410. Springer-Verlag, April 1991.

[Her92] M. Hermann. On the relation between primitive recursion, schematization, and divergence. In H. Kirchner and G. Levi, editors, *Proceedings 3rd Conference on Algebraic and Logic Programming (ALP'92), Volterra (Italy)*, volume 632 of *Lecture Notes in Computer Science*, pages 115–127. Springer-Verlag, September 1992.

[HG93] M. Hermann and R. Galbavý. Unification of infinite sets of terms schematized by primal grammars. In W. Snyder, editor, *UNIF '93, Boston (MA, USA)*, June 1993.

[HH82] G. Huet and J.-M. Hullot. Proofs by induction in equational theories with constructors. *Journal of Computer and System Science*, 25(2):239–266, October 1982.

[HK88] D. Hofbauer and R.-D. Kutsche. Proving inductive theorems based on term rewriting systems. In J. Grabowski, P. Lescanne, and W. Wechler, editors, *Proceedings of the International Workshop on Algebraic and Logic Programming, Gaussig (Germany)*, volume 343 of *Lecture Notes in Computer Science*, pages 180–190. Springer-Verlag, November 1988.

[HL78] G. Huet and D.S. Lankford. On the uniform halting problem for term rewriting systems. Rapport de recherche 283, Institut de Recherche en Informatique et en Automatique, Le Chesnay, France, 1978.

[HP85] M. Hermann and I. Prívara. On nontermination of Knuth-Bendix algorithm. Research report VUSEI-AR-OPS-3/85, Institute of Socio-Economic Information and Automation in Management, Bratislava, Czechoslovakia, November 1985.

[HP86] M. Hermann and I. Prívara. On nontermination of Knuth-Bendix algorithm. In L. Kott, editor, *Proceedings 13th ICALP Conference, Rennes (France)*, volume 226 of *Lecture Notes in Computer Science*, pages 146–156. Springer-Verlag, July 1986.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, Mass., USA, 1979.

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

[Hue81] G. Huet. A complete proof of correctness of the Knuth-Bendix completion algorithm. *Journal of Computer and System Science*, 23(1):11–21, August 1981. Also as: Rapport 25, INRIA, 1980.

[Hul80] J.-M. Hullot. Canonical forms and unification. In W. Bibel and R. Kowalski, editors, *Proceedings 5th International Conference on Automated Deduction (CADE'80), Les Arcs (France)*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334. Springer-Verlag, July 1980.

[IN90] P. Inverardi and M. Nesi. A rewriting strategy to verify observational congruence. *Information Processing Letters*, 35:191–199, August 1990.

[IN92] P. Inverardi and M. Nesi. A strategy to deal with divergent rewrite systems. In M. Rusinowitch and J.-L. Rémy, editors, *Proceedings 3rd International Workshop on Conditional Term Rewriting Systems (CTRS'92), Pont-à-Mousson (France)*, volume 656 of *Lecture Notes in Computer Science*, pages 458–467. Springer-Verlag, July 1992.

[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, November 1986.

[JK89] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82:1–33, 1989.

[JK91] J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: A rule-based survey of unification. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic. Essays in honor of Alan Robinson*, chapter 8, pages 257–321. MIT Press, Cambridge (MA, USA), 1991.

[JT88] K.P. Jantke and M. Thomas. Inductive inference for solving divergence in Knuth-Bendix completion. Research report 88/R6, University of Glasgow, Department of Computer Science, Glasgow, UK, September 1988.

[KB70] D.E. Knuth and P.B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1970.

[KH90] H. Kirchner and M. Hermann. Meta-rule synthesis from crossed rewrite systems. In S. Kaplan and M. Okada, editors, *Proceedings 2nd International Workshop on Conditional and Typed Rewriting Systems (CTRS'90), Montreal (Canada)*, volume 516 of *Lecture Notes in Computer Science*, pages 143–154. Springer-Verlag, June 1990.

[Kir85] H. Kirchner. *Preuves par complétion dans les variétés d'algèbres.* PhD thesis, Université de Nancy I, France, 1985.

[Kir89] H. Kirchner. Schematization of infinite sets of rewrite rules generated by divergent completion process. *Theoretical Computer Science*, 67(2-3):303–332, 1989.

[KK82] C. Kirchner and H. Kirchner. Résolution d'équations dans les algèbres libres et les variétés équationelles d'algèbres. Master's thesis, Université de Nancy I, 1982.

[KKR90] C. Kirchner, H. Kirchner, and M. Rusinowitch. Deduction with symbolic constraints. *Revue Française d'Intelligence Artificielle*, 4(3):9–52, 1990.

[KM87] D. Kapur and D.R. Musser. Proof by consistency. *Artificial Intelligence*, 31(2):125–157, February 1987.

[KN85] D. Kapur and P. Narendran. A finite Thue system with decidable word problem and without equivalent finite canonical system. *Theoretical Computer Science*, 35(2 & 3):337–344, 1985.

[KNZ87] D. Kapur, P. Narendran, and H. Zhang. On sufficient completeness and related properties of term rewriting systems. *Acta Informatica*, 24(4):395–415, August 1987.

[Kur92] S. Kurtz. Narrowing and basic forward closures. Research report 5, Technische Fakultät, Abteilung Informatik, Universität Bielefeld, Germany, 1992.

[Lal90] R. Lalement. *Logique, réduction, résolution*. Etudes et recherches en informatique. Masson, Paris, 1990.

[Lan75] D.S. Lankford. Canonical inference. Research report ATP-32, Department of Mathematics and Computer Science, University of Texas, Austin, Texas (USA), December 1975.

[Lan89] S. Lange. Towards a set of inference rules for solving divergence in Knuth-Bendix completion. In K.P. Jantke, editor, *Proceedings of the International Workshop on Analogical and Inductive Inference, Reinhardsbrunn Castle (Germany)*, volume 397 of *Lecture Notes in Computer Science (in Artificial Intelligence)*, pages 304–316. Springer-Verlag, October 1989.

[Les83] P. Lescanne. Computer experiments with the REVE term rewriting system generator. In *Proceedings of the 10th ACM POPL Symposium, Austin, (Texas, USA)*, pages 99–108, January 1983.

[Les86] P. Lescanne. Divergence of the Knuth-Bendix completion procedure and termination orderings. *Bulletin of the European Association for Theoretical Computer Science*, 30:80–83, October 1986.

[Les90] P. Lescanne. On the recursive decomposition ordering with lexicographical status and other related orderings. *Journal of Automated Reasoning*, 6:39–49, 1990.

[LJ89] S. Lange and K.P. Jantke. Towards a learning calculus for solving divergence in Knuth-Bendix completion. Communications of the algorithmic learning group, Leipzig University of Technology, Germany, November 1989.

[LJ90] S. Lange and K.P. Jantke. Inductive completion for transformation of equational specifications. In H. Ehrig, K.P. Jantke, F. Orejas, and H. Reichel, editors, *Recent Trends in Data Type Specification; 7th Workshop on Specification of Abstract Data Types, Wusterhausen/Dosse (Germany)*, volume 534 of *Lecture Notes in Computer Science*, pages 117–140. Springer-Verlag, April 1990.

[LLT90] A. Lazrek, P. Lescanne, and J.-J. Thiel. Tool for proving inductive equalities, relative completeness, and ω-completeness. *Information and Computation*, 84(1):47–70, January 1990.

[LM78] D.S. Lankford and D.R. Musser. A finite termination criterion. Unpublished draft, Information Sciences Institute, University of Southern California, Marina-del-Rey, CA, 1978.

[Mar87] U. Martin. How to choose the weights in the Knuth-Bendix ordering. In P. Lescanne, editor, *Proceedings 2nd Conference on Rewriting Techniques and Applications (RTA'87), Bordeaux (France)*, volume 256 of *Lecture Notes in Computer Science*, pages 42–53. Springer-Verlag, May 1987.

[McA92] D. McAllester. Grammar rewriting. In D. Kapur, editor, *Proceedings 11th International Conference on Automated Deduction (CADE'92), Saratoga Springs (New York, USA)*, volume 607 of *Lecture Notes in Computer Science (in Artificial Intelligence)*, pages 124–138. Springer-Verlag, June 1992.

[MP87] C.-T. Mong and P.W. Purdom. Divergence in the completion of rewriting systems. Technical report, Dept. of Comp. Science, Indiana University, 1987.

[NS89] P. Narendran and J. Stillman. It is undecidable whether the Knuth-Bendix completion procedure generates a crossed pair. In B. Monien and R. Cori, editors, *Proceedings 6th Symposium on Theoretical Aspects of Computer Science (STACS'89), Paderborn (Germany)*, volume 349 of *Lecture Notes in Computer Science*, pages 348–359. Springer-Verlag, February 1989.

[Ott89] F. Otto. Restrictions of congruences generated by finite canonical string-rewriting systems. In N. Dershowitz, editor, *Proceedings 3rd Conference on Rewriting Techniques and Applications (RTA'89), Chapel Hill, (North Carolina, USA)*, volume 355 of *Lecture Notes in Computer Science*, pages 359–370. Springer-Verlag, April 1989.

[Pét67] R. Péter. *Recursive functions.* Academic Press, 1967.

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

[Ros73] B. K. Rosen. Tree-manipulating systems and Church-Rosser theorems. *Journal of the Association for Computing Machinery*, 20(1):160–187, January 1973.

[Rus87] M. Rusinowitch. Path of subterms ordering and recursive decomposition ordering revisited. *Journal of Symbolic Computation*, 3(1 & 2):117–131, 1987.

[Sal92] G. Salzer. The unification of infinite sets of terms and its applications. In A. Voronkov, editor, *Proceedings 3rd International Conference on Logic Programming and Automated Reasoning (LPAR'92), St. Petersburg (Russia)*, volume 624 of *Lecture Notes in Computer Science (in Artificial Intelligence)*, pages 409–420. Springer-Verlag, July 1992.

[Sim88] H. Simmons. The realm of primitive recursion. *Archive for Mathematical Logic*, 27:177–188, 1988.

[SK91] A. Sattler-Klein. Divergence phenomena during completion. In R.V. Book, editor, *Proceedings 4th Conference on Rewriting Techniques and Applications (RTA'91), Como (Italy)*, volume 488 of *Lecture Notes in Computer Science*, pages 374–385. Springer-Verlag, April 1991.

[SK92] A. Sattler-Klein. Infinite canonical string rewriting systems generated by completion. In A. Voronkov, editor, *Proceedings 3rd International Conference on Logic Programming and Automated Reasoning (LPAR'92), St. Petersburg (Russia)*, volume 624 of *Lecture Notes in Computer Science (in Artificial Intelligence)*, pages 433–444. Springer-Verlag, July 1992.

[Sla74] J.R. Slagle. Automated theorem-proving for theories with simplifiers, commutativity, and associativity. *Journal of the Association for Computing Machinery*, 21:622–642, 1974.

[SS82] J. Siekmann and P. Szabó. A Nœtherian and confluent rewrite system for idempotent semigroups. *Semigroup Forum*, 25:83–110, 1982.

[TJ89] M. Thomas and K.P. Jantke. Inductive inference for solving divergence in Knuth-Bendix completion. In K.P. Jantke, editor, *Proceedings of the International Workshop on Analogical and Inductive Inference, Reinhardsbrunn Castle (Germany)*, volume 379 of *Lecture Notes in Computer Science (in Artificial Intelligence)*, pages 288–303. Springer-Verlag, October 1989.

[Toy88] Y. Toyama. Confluent term rewriting systems with membership conditions. In J.-P. Jouannaud and S. Kaplan, editors, *Proceedings 1st International Workshop on Conditional Term Rewriting Systems (CTRS'88), Orsay (France)*, volume 308 of *Lecture Notes in Computer Science*, pages 228–241. Springer-Verlag, July 1988.

[TW90] M. Thomas and P. Watson. Generalising diverging sequences of rewrite rules by synthesising new sorts. In S.L. Peyton Jones, G. Hutton, and C. Kehler Holst, editors, *Proceedings 3rd Glasgow Workshop on Functional Programming, Ullapool (United Kingdom)*, Workshops in Computing Science, pages 268–273. Springer-Verlag, 1990.

[TW93] M. Thomas and P. Watson. Solving divergence in Knuth-Bendix completion by enriching signatures. *Theoretical Computer Science*, 112(1):145–185, 1993.

Annexes

Chapitre 6

On Proving Properties of Completion Strategies

Chapitre 7

The Complexity of Counting Problems in Equational Matching