Contributions à l'étude des Lambda-calculs avec Substitutions Explicites

A. RIOS

Laboratoire d'Informatique, URA 1327 du CNRS
Département de Mathématiques et d'Informatique
Ecole Normale Supérieure

LIENS - 93 - 24

Décembre 1993

THESE DE DOCTORAT

présentée à

l'UNIVERSITE PARIS VII

Spécialité : Informatique

présentée par

Alejandro RIOS

Sujet de la thèse :

Contributions à l'étude des Lambda-calculs avec Substitutions Explicites

Soutenue le 8 janvier 1993, devant la Comission
d'examen composée de :

MM.	P.-L. Curien	Président
	H. Barendregt	
	S. Grigorieff	
	T. Hardin	Examinateurs
	H. Kirchner	
	T. Streicher	

Contributions à l'étude des λ-calculs avec des substitutions explicites

Alejandro Ríos

21 janvier 1993

A la mémoire de mes grand-parents
Herminia et Rafael.

Je veux témoigner toute ma gratitude à Pierre-Louis Curien, mon directeur de thèse et président de ce jury. Il a suggéré cette voie de recherche et a toujours été disponible pour discuter. Ses encouragements et ses critiques ont été essentiels. Ce travail lui doit beaucoup. Je le remercie aussi de m'avoir initié à la recherche.

Je suis particulièrement reconnaissant à Henk Barendregt qui a accepté la tâche d'être rapporteur. Je le remercie vivement des remarques qu'il m'a faites et des futures voies de recherche qu'il m'a suggérées.

Thomas Streicher a aussi été rapporteur et il est venu de München pour cette soutenance. Je lui adresse tous mes remerciements pour la lecture attentive de la version préliminaire et pour être présent aujourd'hui.

Serge Grigorieff, Thérèse Hardin et Hélène Kirchner m'ont fait l'honneur d'être membres de ce jury. Qu'ils en soient chaleureusement remerciés.

Je tiens à dire ici tout ce que je dois à Thérèse Hardin. Les discussions amicales que nous avons eues, les lectures attentives des versions préliminaires des chapitres 3 et 4, sa participation active au chapitre 2, les suggestions et critiques qu'elle a faites m'ont été précieuses. J'ai beaucoup appris en lisant ses travaux et en discutant avec elle. Je lui en serai toujours profondément reconnaissant.

Je tiens également à remercier vivement Guy Cousineau pour son accueil chaleureux au sein du Projet Formel (ENS-INRIA) et je veux dire à tous ses membres combien j'ai apprécié de travailler avec eux. Je tiens tout particulièrement à remercier mes collègues des Toits du DMI : Roberto Bellucci, François Bouladoux, Giuseppe Castagna, Roberto Di Cosmo et Carolina Lavatelli.

Mes derniers remerciements sont pour Susana Berestevoy. Elle sait tout ce que je lui dois.

Enfin, je tiens à dire que j'ai bénéficié d'une bourse du Gouvernement Français et d'une allocation de recherche de CAIMENS, sans lequelles ce travail n'aurait pas vu le jour.

Table des matières

Table des figures

Introduction

Ce travail se place dans un domaine où convergent la théorie de la réécriture et le λ-calcul. Son but principal est d'approfondir les connaissances sur une famille de calculs, les $\lambda\sigma$-calculs, qui ont vu le jour récemment et dont la caractéristique essentielle est le traitement explicite de la substitution. On a également abordé deux nouveaux calculs avec des substitutions explicites, pour lesquels certains résultats partiels ont été obtenus : le $\lambda\nu$-calcul et le $\lambda\tau$-calcul.

Le λ-calcul a été fondé par Church dans les années 30 afin de développer une théorie générale des fonctions et d'étendre cette théorie avec des notions logiques en vue de fournir une fondation de la logique et des mathématiques. Malheureusement, les tentatives fondationnelles ont échoué et seule la sous-théorie correspondant à la partie fonctionnelle a survécu. C'est dans le cadre de cette théorie que Church a formalisé la notion de "fonction effectivement calculable" (λ-définissable).

Dans le λ-calcul, les fonctions sont considérées en tant que procédés. L'interêt de cette conception est de souligner leur aspect calculatoire. Ce point de vue dynamique diffère de la notion ensembliste, plus statique, de fonction en tant que graphe, c'est-à-dire ensemble de paires ordonnées.

Une fonction en tant que règle de calcul peut être considérée comme un programme, exécutable en suivant des instructions, pour donner un résultat à partir de certains arguments. Le λ-calcul permet donc la manipulation des programmes et présente lui-même plusieurs aspects des langages de programmation et de leurs implémentations. On sait que ce calcul, même en ayant une syntaxe très simple, est suffisamment puissant pour décrire toutes les fonctions calculables et devient ainsi un outil efficace pour la conception et l'analyse d'une large famille de langages de programmation : les langages fonctionnels.

Le λ-calcul contient une seule règle, appelée β : une application de cette règle consiste essentiellement à remplacer certaines occurrences d'une variable par un terme. Cette opération de substitution est décrite dans le méta-langage du λ-calcul. Ceci peut s'avérer génant dans la modélisation des implémentations. En effet, la variable doit être remplacée seulement quand elle est libre dans un certain sous-terme et, en plus, il faut éviter que, après la substitution, les variables libres du terme introduit soient capturées par des abstractions préexistantes. On est donc obligé de faire les renommages adéquats pour résoudre ce problème.

Plusieurs solutions ont été proposées pour traiter ces inconvénients, dont deux

nous intéressent particulièrement :

- Celle proposée dans les années 70 par de Bruijn qui introduit une notation où les noms des variables sont remplacés par des entiers naturels indiquant la hauteur de liaison de chaque variable, i.e. le nombre de λ's à franchir quand on remonte de l'occurrence examinée jusqu'à l'abstracteur qui la lie.

- Celle où, au lieu de définir la substitution dans le méta-langage, on introduit dans le langage un nouvel opérateur pour dénoter la substitution à effectuer.

Dans les $\lambda\sigma$-calculs et dans le $\lambda\tau$-calcul ces deux ingrédients sont combinés, tandis que pour le $\lambda\nu$-calcul on ne retient que le second.

Une caractéristique commune aux calculs mentionnés précédemment est la présence d'une règle *(Beta)* destinée à déclencher la substitution, le reste des règles servant à la propager jusqu'aux variables concernées. Les calculs obtenus par suppression de *(Beta)* sont dénotés : σ-calculs, ν-calcul et τ-calcul, respectivement, et appelés génériquement calculs de substitutions.

Les $\lambda\sigma$-calculs ont leur origine dans la Logique Combinatoire Catégorique (CCL), introduite par P.-L. Curien [Cur86] comme un modèle syntaxique des Catégories Cartesiennes Fermées. Sur les Combinateurs Catégoriques, deux théories équationnelles non équivalentes ont été développées :

- La Logique Combinatoire Catégorique Faible qui est à l'origine de l'implémentation du langage CAML (cf [MS86]). Celle-ci repose sur la Machine Abstraite Catégorique (CAM) décrite dans [CCM87].

- La Logique Combinatoire Catégorique Forte équivalente à la théorie du λ-calcul avec couples (cf. [Cur86] et [Har87]).

Dans [Har87], T. Hardin étudie les propriétés de confluence du système de réécriture obtenu en orientant les équations de la Logique Combinatoire Catégorique Forte. Elle montre que le système n'est pas confluent, mais qu'en revanche, un sous-système, appelé $\mathcal{E}+$*(Beta)*, capable de simuler la β-réduction, est confluent sur un sous-ensemble de termes clos. La définition de ce sous-ensemble n'est pas simple [Har89] et il contient une traduction des termes du λ-calcul.

CCL est le premier système de réécriture connu, capable d'implémenter complètement la substitution, mais il n'est pas tout à fait satisfaisant. Premièrement, pour étudier les λ-termes il faut se restreindre au sous-ensemble mentionné plus haut, dont la manipulation est difficile. Deuxièmement, fonctions et environnements sont représentés par des combinateurs au même niveau, tandis que leur sémantique est fort différente.

Pour essayer de résoudre ces problèmes, P.-L. Curien conçut en 1988 un calcul, appelé $\lambda\rho$ [Cur88], qui contient deux sortes : `terme` et `substitution`, et qui a été étendu ultérieurement au $\lambda\sigma$-calcul [ACCL90]. Ce dernier calcul s'est avéré être une

version à deux sortes du système $\mathcal{E}+(Beta)$ et il répond favorablement aux problèmes qui nous intéressent :

- Il est confluent sur les termes/substitutions clos. De plus, les termes clos contiennent les termes du λ-calcul en notation de de Bruijn, qui peuvent être décrits d'une manière très simple : les σ-formes normles.

- On dispose d'une distinction sémantique entre fonctions et environnements.

Pourtant, sur l'ensemble total des termes/substitutions, le $\lambda\sigma$-calcul, tout comme sa version combinatoire catégorique, n'est même pas localement confluent. L'étude des paires critiques suggère l'ajout de quatre règles, grâce auxquelles la confluence locale est atteinte. Le calcul ainsi obtenu est appelé $\lambda\sigma'$-calcul. L'une de ces règles est non-linéaire à gauche et rappelle la règle dite de "surjective pairing" du λ-calcul avec couples. J. W. Klop a montré que la confluence de ce calcul est détruite par cette règle [Klo80]. C'est aussi le cas pour le $\lambda\sigma'$-calcul : la non-confluence, déjà conjecturée dans [ACCL90], a été établie récemment [CHL91].

Existe-t-il un calcul totalement confluent qui possède les bonnes propriétés de $\lambda\sigma$? La réponse affirmative a été donnée dans [HL89]. Le $\lambda\sigma_{\Uparrow}$-calcul est un calcul confluent sur un ensemble de termes à deux sortes et il peut être interprété dans $\lambda\sigma$. L'idée centrale pour l'obtention de ce calcul est l'introduction d'un opérateur unaire afin d'eviter la paire critique qui conduit à l'ajout de la règle de "surjective pairing". Cet opérateur, noté $\Uparrow$, affecte la substitution après la traversée d'un abstracteur.

On peut donc comprendre l'évolution de CCL à $\lambda\sigma_{\Uparrow}$ comme une suite d'améliorations qui ont pour but la confluence. L'une des contributions principales de cette thèse se situe à mi-chemin dans cette ascension vers la confluence totale. Etant donné que le $\lambda\sigma'$-calcul n'est pas confluent sur la totalité des termes/substitutions, et que le contre-exemple donné dans [CHL91] utilise de manière cruciale les variables de sorte `substitution`, nous nous demandons quelle est la limite jusqu'à laquelle on peut pousser la confluence dans le cadre du $\lambda\sigma'$-calcul. La réponse donnée dans le chapitre 3 est la suivante : au moins jusqu'aux termes *semi-clos*. Ce sont les termes qui admettent des variables de sorte `terme`, mais pas de sorte `substitution`. La même réponse est valide pour le $\lambda\sigma'$-calcul extensionnel (chapitre 4).

Le $\lambda\sigma'$-calcul devient donc fort intéressant, non seulement du point de vue théorique, mais aussi dans le domaine des applications, puisque la possession des variables de sorte `terme` (rappelons que les termes du λ-calcul sont codés dans cette sorte) permet de se placer dans une théorie plus générale de la réduction où l'on peut formaliser une notion de modularité : une variable étant considérée comme une sous-expression non spécifiée, un terme du $\lambda\sigma'$-calcul semi-clos joue le rôle d'un contexte.

La méthode proposée pour arriver à ces résultats est la méthode d'interprétation, identifiée dans [Har89], où elle a été utilisée pour les combinateurs catégoriques. C'est une méthode simple et effective qui consiste à réduire le problème de confluence, via σ'-normalisation dans ce cas, à la confluence d'un calcul plus simple (une extension du λ-calcul) qui peut être établie par des méthodes classiques.

L'existence et l'unicité des σ'-formes normales, et plus généralement la noethérianité et la confluence des calculs de substitutions, deviennent donc indispensables. Grâce au lemme de Newman [New42], il suffit d'établir la noethérianité et la confluence locale, et celle-ci peut être vérifiée, dans le cas des σ-calculs et du τ-calcul, par analyse automatique des paires critiques [KB70]. Par conséquent, c'est le problème de la noetherianité de ces calculs qu'il faut résoudre.

Une seule démonstration de la noetherianité du σ-calcul était connue. Elle est fondée sur la terminaison du sous-système de CCL correspondant à σ' : SUBST. La preuve de la noetherianité de SUBST [HL86] est complètement différente de celle que nous proposons pour σ dans le deuxième chapitre[1]. Etant donné que la noetherianité du $\lambda\sigma$-calcul typé est encore un problème ouvert, nous considérons qu'aucune technique développée pour établir la terminaison de σ n'est à écarter.

Tous les calculs discutés précédemment sont non-typés et leurs versions typées restent encore peu étudiées. Dans [ACCL90], on introduit le $\lambda\sigma$-calcul typé du premier ordre et on montre sa correction par rapport au λ-calcul : si a est typable dans le $\lambda\sigma$-calcul, alors sa σ-forme normale est typable dans le λ-calcul. La réciproque (complétude) n'est pas vraie à cause des σ-règles qui perdent de l'information. Notre contribution à ce sujet est la formulation d'une traduction autre que la σ-normalisation, pour laquelle on peut montrer la correction et la complétude (chapitre 5).

Dans sa thèse [Ehr88], T. Ehrhard présente une théorie équationnelle provenant d'une définition catégorique générale du calcul des constructions de T. Coquand et G. Huet [CH88]. Une possible orientation du fragment correspondant au premier ordre non-typé de cette théorie, donne naissance à un système de réécriture que nous appelons $\lambda\tau$. Le calcul ainsi obtenu est très proche de $\lambda\sigma_{\Uparrow}$: il n'introduit qu'une modification syntaxique dans la sorte `substitution`; on peut donc le considérer comme un nouveau membre de la famille des $\lambda\sigma$-calculs. Par les mêmes raisons que celles que nous avons évoquées plus haut, la terminaison de τ est essentielle pour étudier ce calcul. Notre apport dans ce domaine est la preuve de terminaison faible (chapitre 7). La terminaison forte reste encore un problème ouvert et, étant donnée sa diffuculté, nous croyons que c'est un vrai défi pour les spécialistes de la réécriture.

Le traitement explicite de la substitution, combiné avec le formalisme de de Bruijn, offre un cadre adéquat pour décrire et prouver des propriétés mathématiques des substitutions. Dès qu'on abandonne la notation de de Bruijn pour revenir à la notation usuelle des variables, en s'éloignant ainsi des systèmes de réécriture classiques, les difficultés augmentent de sorte qu'on n'est pas encore arrivé à formuler de manière satisfaisante un calcul de substitutions explicites avec des noms de

[1] Le contenu de ce chapitre est le fruit d'un travail en collaboration avec P.-L. Curien et T. Hardin et a déjà été publié comme Rapport de Recherche du LIENS [CHR91] et dans les Proceedings de MFCS'92 [CHR92].

Récemment une troisième preuve de la noethérianité de σ a été fournie par H. Zantema [Zan92]. Cette preuve est moins complexe que les deux énoncées précédemment et s'intègre dans une méthode plus générale.

variables. Le $\lambda\nu$-calcul que nous introduisons dans le chapitre 6 est un essai dans cette direction. Ce calcul possède deux opérateurs de renommage qui permettent la mise à jour des variables des abstracteurs pour éviter les captures. La correction de ce calcul ne peut être obtenue qu'en imposant une condition sur les variables du terme de départ. Malgré nos efforts nous n'avons pas trouvé une condition qui soit préservée par les dérivations. En revanche, nous avons obtenu une condition qui est préservée en suivant une certaine stratégie de réduction pour laquelle le calcul est correct. Ceci ajouté au fait qu'on n'a pas trouvé d'exemple où il y ait une capture, nous a conduit à conjecturer une propriété de confluence pour le $\lambda\nu$-calcul, qui reste encore un problème ouvert.

Plan

Dans le premier chapitre nous rappelons les notions utiles de la théorie des algèbres universelles et de la théorie de la réécriture. Nous introduisons le λ-calcul dans ses deux versions : classique et à la de Bruijn, et présentons le concept d'extensionalité pour ces deux versions. Le reste de ce chapitre est consacré à exhiber la famille des $\lambda\sigma$-calculs en même temps que nous donnons l'état de l'art de chacun d'eux, en incluant les résultats obtenus dans cette thèse. Dans la dernière section nous définissons la version typée du $\lambda\sigma$-calcul.

Les chapitres 2, 3, et 4 constituent une unité logique concernant essentiellement le $\lambda\sigma'$-calcul.

Dans le chapitre 2 nous donnons la preuve de normalisation forte de σ, obtenue en collaboration avec P.-L. Curien et T. Hardin, et nous étendons ce résultat à σ'. C'est dans ce chapitre que les rapports entre les $\lambda\sigma$-calculs et la logique combinatoire catégorique sont examinés.

Le chapitre 3 est consacré à la confluence du $\lambda\sigma'$-calcul semi-clos. La méthode utilisée est celle d'interprétation sur un calcul qui s'avère une extension du λ-calcul et que nous appelons λ'. La confluence de ce dernier calcul est établie grâce à une preuve à la Tait-Martin-Löf (définition d'une parallélisation et égalité des clôtures réflexives transitives).

Dans le chapitre 4 nous montrons la confluence du $\lambda\sigma'$-calcul extensionnel. La méthode utilisée est toujours celle d'interprétation. La confluence est ramenée maintenant à celle du λ'-calcul extensionnel, et celle-ci est montrée suivant la méthode classique : commutation de la règle extensionnelle avec β.

Le chapitre 5, le seul qui s'occupe de typage, étudie une interprétation différente de la σ-normalisation qui permet d'établir un résultat de complétude pour le $\lambda\sigma$-calcul typé. Nous étendons ensuite ce résultat au $\lambda\sigma_{\Uparrow}$-calcul typé du premier ordre.

Les chapitres 6 et 7 regroupent des résultats qui restent encore partiels.

Dans le chapitre 6, nous présentons et motivons le $\lambda\nu$-calcul où nous abandonnons la notation de de Bruijn pour revenir à la notation classique avec les noms des variables. Pour que ce calcul soit correct il faut se restreindre à opérer sur un

sous-ensemble de termes qui satisfont un certain invariant. Nous montrons que, malheureusement, cet invariant n'est pas préservé par $\lambda\nu$-dérivation. Cependant, nous trouvons une stratégie qui conserve l'invariant et la correction du $\lambda\nu$-calcul est ainsi garantie pour cette stratégie. Ce résutat de correction permet d'obtenir la conflunce, modulo α-congruence, de notre stratégie. Le chapitre finit avec deux questions ouvertes concernant la confluence.

Dans le chapitre 7 nous introduisons le $\lambda\tau$-calcul. Un outil indispensable pour l'étude de ce calcul est l'existence et unicité des τ-formes normales. Malgré nos efforts, la τ-noetherianité reste encore un problème ouvert. Cependant, nous avons franchi un premier pas : nous établissons la normalisation faible en montrant que la stratégie "leftmost innermost" est normalisante.

Pour finir cette thèse nous énonçons d'autres problèmes encore ouverts en rapport avec les substitutions explicites, les plus importants étant les développements finis et la noethérianité du $\lambda\sigma$-calcul typé.

Enfin, en annexe, nous rappelons quelques sous-systèmes de la logique combinatoire catégorique, énonçons des résultats du λ-calcul classique qui ont été utilisés et transcrivons les sessions avec le logiciel KB rendant compte des résultats de confluence locale utilisés dans cette thèse.

Chapitre 1

Présentation des formalismes et des résultats principaux

Dans ce chapitre nous présentons les formalismes qui seront utilisés dans la suite. Les résultats principaux de ce travail, concernant les $\lambda\sigma$-calculs, sont énoncés dans les sections 1.6 et 1.7.

Dans la section 1.1 nous décrivons le formalisme des algèbres de termes. Ce formalisme connaît diverses dénominations: Algèbres Universelles, selon les algébristes; Théories du Premier Ordre, selon les logiciens; Types Abstraits Algébriques, selon les informaticiens. Les notions introduites dans cette section seront sous-jacentes à l'étude des $\lambda\sigma$-calculs et du $\lambda\tau$-calcul.

Dans la section 1.2 nous rappelons les définitions essentielles et quelques propriétés des systèmes de réduction en général, et de réécriture sur des algèbres de termes, en particulier. Ils sont importants dans plusieurs domaines : spécification de types de données abstraits, implémentation de langages fonctionnels, déduction automatique. Réécrire un terme consiste à remplacer une sous-expression donnée du terme par une autre expression plus simple dans un certain sens. Deux propriétés importantes de ces systèmes sont la confluence et la noethérianité (ou normalisation forte). La confluence assure l'unicité du résultat d'un calcul par réécriture, tandis que la noethérianité garantit l'existence de ce résultat.

Le λ-calcul, dont nous avons déjà donné un apperçu dans l'Introduction, est présenté formellement dans la section 1.3. Ce calcul a été introduit par le logicien Alonzo Church dans les années 30. Il s'agit d'un système formel permettant la construction des fonctions éventuellement partielles qui peuvent accepter n'importe quel terme du calcul comme argument. L'un des traits intéressants du λ-calcul est que tout algorithme peut être vu comme une expression du calcul. Le λ-calcul contient une seule règle, appelée β : une étape de simplification consiste essentiellement à remplacer certaines occurrences d'une variable par un terme. Cette opération de substitution est décrite dans le méta-langage du λ-calcul.

Un inconvénient du λ-calcul est le problème bien connu des captures de variables. Dans les années 70, une notation spécialement adéquate pour traiter ce problème a

été introduite par de Bruijn. Nous consacrons la section 1.4 à la présentation de ce formalisme qui sera sous-jacent à la plupart des calculs introduits ultérieurement.

La β-égalité engendrée par la β-réduction du λ-calcul n'est pas extensionnelle, mais on peut obtenir un calcul extensionnel ($\lambda\eta$-calcul) par ajout d'une règle connue dans la littérature sous le nom de η-règle. Dans la section 1.5 nous présentons ce calcul et sa version dans le formalisme de de Bruijn.

Le reste de ce chapitre est consacré à l'introduction de la famille des $\lambda\sigma$-calculs :

Le $\lambda\sigma$-calcul (section 1.6) est un développement du λ-calcul où les substitutions sont gérées de manière explicite, i.e. elles ont une représentation syntaxique dans le langage du $\lambda\sigma$-calcul. On trouve donc dans ce calcul le cadre cherché pour étudier la théorie des substitutions, ainsi qu'un intermédiaire entre le λ-calcul classique et les implémentations concrètes des langages fonctionnels. Par conséquent, le $\lambda\sigma$-calcul s'avère un outil efficace pour la dérivation de machines abstraites pour le λ-calcul et pour prouver la correction de ces machines.

Le $\lambda\sigma$-calcul est un système de réécriture sur une algèbre à deux sortes : `terme` et `substitution`. Il est facile de montrer que le $\lambda\sigma$-calcul n'est pas localement confluent sur les termes ouverts (ceux contenant des variables des deux sortes). L'ajout de quatre règles permet d'obtenir la confluence locale. Nous appelons cette extension du $\lambda\sigma$-calcul, le $\lambda\sigma'$-calcul. Nous présentons ce dernier calcul dans la section 1.7, où nous énonçons les propriétés essentielles.

Le $\lambda\sigma'$-calcul ne s'avère pas confluent sur la totalité des termes ouverts, mais une autre version du $\lambda\sigma$-calcul, appelée ici $\lambda\sigma_{\Uparrow}$-calcul, a cette propriété. La section 1.8 est consacrée à exposer succinctement ce calcul.

Enfin, dans la section 1.9 nous introduisons les versions typées du premier ordre du λ-calcul et du $\lambda\sigma$-calcul. Nous énonçons le théorème de correction (soundness) et nous posons le problème de la complétude, qui sera traité dans le chapitre 5.

1.1 Algèbres de Termes

L'un des buts de l'Algèbre Universelle est d'extraire les éléments communs à divers types de structures algébriques. Cette généralisation et unification de concepts, constructions et résultats permet de se placer à un niveau plus élevé d'abstraction.

Nous ne donnons ici que les définitions premières de la théorie des Algèbres Universelles, pour une étude approfondie de cette théorie se référer à [BS81] et [Gra79].

1.1.1 Algèbres homogènes

Définition 1.1 *Soit $\mathcal{F}$ un ensemble de symboles de fonctions et $d : \mathcal{F} \to \mathbb{N}$. Soit $f \in \mathcal{F}$, on appelle $d(f)$ l'*arité *de f. Soit V un ensemble de variables.*
La $\mathcal{F}$-algèbre libre homogène *engendrée par V, notée $T_{\mathcal{F}}(V)$, est le plus petit en-*

semble tel que:

1. $V \subset T_{\mathcal{F}}(V)$
2. *si* $f \in \mathcal{F}$, $d(f) = n$ *et* $t_1,...,t_n \in T_{\mathcal{F}}(V)$, *alors* $f(t_1, \cdots, t_n) \in T_{\mathcal{F}}(V)$.

Les éléments de $T_{\mathcal{F}}(V)$ *sont appelés* termes *et* $\mathcal{F}$, *la* signature *de l'algèbre. L'ensemble des variables présentes dans* $t \in T_{\mathcal{F}}(V)$ *sera noté* $V(t)$.

Définition 1.2 *La* $\mathcal{F}$-algèbre initiale, *notée* $T_{\mathcal{F}}$, *est l'algèbre engendrée par un ensemble vide de variables. Elle est non vide si et seulement si* $\mathcal{F}$ *possède des symboles d'arité 0, c'est-à-dire des constantes. Les termes de* $T_{\mathcal{F}}$ *sont dits* clos.

Définition 1.3 *L'ensemble des* occurrences $O(t)$ *du terme* $t \in T_{\mathcal{F}}(V)$ *et le* sous-terme t/γ *de* t *à l'occurrence* γ *sont définis ainsi:*

1. *si* $t \in V$, *alors* $O(t) = \{\epsilon\}$ *et* $t/\epsilon = t$ *où* ϵ *désigne le mot vide.*
2. *si* $t = f(t_1, \cdots, t_n)$, *alors* $O(t) = \{\epsilon\} \cup \{i\gamma \ : \ 1 \leq i \leq n \ , \ \gamma \in O(t_i)\}$ *et* $t/i\gamma = t_i/\gamma$.

L'ensemble $O(t)$ est ordonné par l'ordre préfixe. Un terme peut être représenté par un arbre dont la racine est le symbole à l'occurrence ϵ et une branche est une suite totalement ordonnée d'occurrences.

Exemple 1.1 *Soient* $\uparrow$, $\Uparrow$ *et* $\circ$ *des symboles d'arité 0 (constante), 1 et 2, respectivement. Le terme* $\circ(t_1, t_2)$ *sera noté* $t_1 \circ t_2$. *Le terme*

$$t = (\Uparrow (\uparrow \circ \uparrow) \circ \uparrow) \circ (\Uparrow (\Uparrow (\uparrow)) \circ \Uparrow (\uparrow))$$

est un terme clos dont l'ensemble d'occurrences est donnée par:

$$O(t) = \{\epsilon, 1, 11, 111, 1111, 1112, 12, 2, 21, 211, 2111, 22, 221\}.$$

Voici quelques sous-termes du terme t*:*

$$t/11 = \Uparrow (\uparrow \circ \uparrow) \ , \quad t/1111 = t/1112 = \uparrow \ , \quad t/22 = \Uparrow (\uparrow).$$

Définition 1.4 *Le* remplacement *du sous-terme* t/γ *par le terme* u *dans le terme* t *sera noté* $t\{\gamma \leftarrow u\}$ *et est défini par:*

1. *si* $\gamma = \epsilon$, *alors* $t\{\gamma \leftarrow u\} = u$.
2. $f(t_1, \cdots, t_n)\{i\gamma \leftarrow u\} = f(t_1, \cdots, t_{i-1}, t_i\{\gamma \leftarrow u\}, t_{i+1}, \cdots, t_n)$.

1.1.2 Algèbres hétérogènes

Définition 1.5 *Soient $\mathcal{K}$ un ensemble de sortes et $\mathcal{F}$ un ensemble de symboles. Soit $d : \mathcal{F} \to \mathbb{N}$, comme dans le cas homogène, on appelle $d(f)$ l'*arité *de f.*
A tout symbole $f \in \mathcal{F}$ on associe une suite de $d(f)+1$ sortes, appelée le type *de f. On note le type de f ainsi :*

$$f \ : \ k_1, \ldots, k_n \Rightarrow k_{n+1} \quad \text{où} \quad d(f) = n \ \text{ et } \ k_i \in \mathcal{K} \, .$$

Soient V un ensemble de variables et $v : V \to \mathcal{K}$. On dira que $v(x)$ est le type de la variable $x \in V$.
*On définit simultanément l'ensemble de termes de la $\mathcal{F}$-*algèbre libre hétérogène *engendrée par V, notée encore $T_{\mathcal{F}}(V)$, et le type de chaque terme ainsi:*

$$t \in T_{\mathcal{F}}(V) \ \text{ et la sorte de } t \ \text{ est } k \, , \text{ noté } t : k \ \text{ ssi}$$

1. $t \in V$ *et* $k = v(t)$.

2. $t = f(t_1, \cdots, t_n)$ *où* $d(f) = n$, $f : k_1, \ldots, k_n \Rightarrow k$ *et* $t_i : k_i$ *pour* $1 \leq i \leq n$.

Exemple 1.2 *Soit $\mathcal{K} = \{\tau, \sigma\}$ et soient $\mathtt{1}$, $\uparrow$, λ et $[\,]$ des symboles de fonctions d'arité 0, 0, 1 et 2, respectivement. On abrège $[\,](t, s)$ par $t[s]$.*
Les types sont donnés par :

$$\mathtt{1} : \tau \ , \quad \uparrow : \sigma \ , \quad \lambda : \tau \Rightarrow \tau \ , \quad [\,] : \tau, \sigma \Rightarrow \tau \, .$$

Un exemple de terme clos de cette algèbre hétérogène est

$$t = \lambda \, (\ (\ (\lambda \mathtt{1})\, [\uparrow]\ )\, [\uparrow]\ ) \, ,$$

et on vérifie aisément que sa sorte est τ.

Les notions de *sous-terme* et *occurrence* sont définies de la même façon que dans le cas homogène.

1.2 Réécriture

Nous présentons dans cette section les notions essentielles des systèmes de réduction et systèmes de réécriture et ne donnons que les résultats qui seront nécessaires dans la suite. Un excellent recueil des résultats les plus importants obtenus jusqu'à présent dans ce domaine est [Klo90].

1.2.1 Systèmes de réduction

Définition 1.6 *Soient A un ensemble et R une relation binaire dans A, i.e. un sous-ensemble de $A \times A$. On notera le fait $(a, b) \in R$ par $a \to_R b$ ou $a \to b$ quand le contexte le permettra. On appelle* réduction *cette relation et* système de réduction, *la paire (A, R). On note :*

1. *R^ϵ ou $\xrightarrow{\epsilon}_R$ ou simplement $\xrightarrow{\epsilon}$ la clôture réflexive de R.*
2. *R^+ ou $\xrightarrow{+}_R$ ou simplement $\xrightarrow{+}$ la clôture transitive de R.*
3. *R^* ou $\twoheadrightarrow_R$ ou simplement $\twoheadrightarrow$ la clôture réflexive et transitive de R. Quand $a \twoheadrightarrow b$ on dit qu'il y a une* dérivation *de a à b.*
4. *$a \xrightarrow{n}_R b$ ou simplement $a \xrightarrow{n} b$ le fait que a se réduit par b en $n \geq 0$ pas de réduction, i.e.*
   - *si $n \geq 2$, il existe $b_1, \ldots, b_{n-1}$ tels que $a \to b_1 \to \cdots \to b_{n-1} \to b$,*
   - *$a \xrightarrow{1}_R b$ exprime $a \to b$,*
   - *$a \xrightarrow{0}_R b$ équivaut à $a = b$.*

   On appelle n la longueur de la dériavation.

Définition 1.7 *Soient R et S deux réductions dans A. On dit qu'elles* commutent *ssi R et S satisfont la condition suivante (voir la figure 1.2(i) [1]) :*

$$\forall a, b, c \in A \ \exists d \in A \ ((a \twoheadrightarrow_R b \wedge a \twoheadrightarrow_S c) \Rightarrow (b \twoheadrightarrow_S d \wedge c \twoheadrightarrow_R d)).$$

Définition 1.8 *Soit R une réduction dans A.*

1. *R est* localement confluente *ou WCR (*weakly Church-Rosser*) sii R satisfait la condition suivante (voir la figure 1.2(ii)) :*

$$\forall a, b, c \in A \ \exists d \in A \ ((a \to b \wedge a \to c) \Rightarrow (b \twoheadrightarrow d \wedge c \twoheadrightarrow d)).$$

2. *R est* confluente *ou CR (*Church-Rosser*) ssi elle commute avec elle même (voir la figure 1.2(iii)), i.e.*

$$\forall a, b, c \in A \ \exists d \in A \ ((a \twoheadrightarrow b \wedge a \twoheadrightarrow c) \Rightarrow (b \twoheadrightarrow d \wedge c \twoheadrightarrow d)).$$

3. *R est* quasi-fortement confluente *ou QFC ssi (voir la figure 1.2(iv))*

$$\forall a, b, c \in A \ \exists d \in A \ ((a \to b \wedge a \to c) \Rightarrow (b \xrightarrow{\epsilon} d \wedge c \xrightarrow{\epsilon} d)).$$

[1] Toutes les figures de ce chapitre, sauf la figure 1.1, se trouvent à la fin de celui-ci.

4. *R est* fortement confluente *ou FC ssi (voir la figure 1.2(v))*

$$\forall a, b, c \in A \ \exists d \in A \ ((a \to b \wedge a \to c) \Rightarrow (b \to d \wedge c \to d)) .$$

Définition 1.9 *Soit R une réduction dans A.*

1. *On dit que $a \in A$ est une* R-forme normale *s'il n'existe pas de $b \in A$ tel que $a \to b$ et on dit que b* a une forme normale *s'il existe une forme normale a telle que $b \twoheadrightarrow a$.*

2. *R est* faiblement noethérienne *ou WN (*weakly normalizing*) si tout $a \in A$ a une R-forme normale.*

3. *R est* noethérienne *ou SN (*strongly normalizing*) s'il n'existe pas de suite infinie $(a_i)_{i \geq 0}$ telle que $a_i \to a_{i+1}$ pour tout $i \geq 0$.*

Lemme 1.1 *Soient R une réduction dans un ensemble A.*

1. *Si R est FC, alors R^* est FC.*

2. *Si R est QFC, alors R est CR.*

Démonstration

1. Voir [Bar84], lemme 3.3.2.

2. Si R est QFC, alors R^ϵ est FC. D'après le premier item, $(R^\epsilon)^*$ est FC. Or, $(R^\epsilon)^* = R^*$ et R^* est FC ssi R est confluente.

□

Lemme 1.2 (Commutativity Lemma) *Soient R et S deux réductions dans un ensemble A. Si la conditon*

$$(\forall a, b, c \in A)[(\, a \to_R b \ \wedge \ a \to_S c \,) \Rightarrow (\, \exists d \in A \,)(\, b \twoheadrightarrow_S d \ \wedge \ c \stackrel{\epsilon}{\to}_R d \,)]$$

est satisfaite, alors R et S commutent. (Voir la figure 1.3.)

Démonstration
Voir [Ros73], lemme 3.6.
□

Théorème 1.1 (Hindley-Rosen) *Si R et S sont deux réductions confluentes dans un ensemble A et si elles commutent, alors $R \cup S$ est une réduction confluente.*

Démonstration
C'est un cas particulier du théorème 3.5, [Ros73].
□

Lemme 1.3 (Newman) *Toute relation noethérienne et localement confluente est confluente.*

Démonstration
Voir [Bar84], proposition 3.1.25.
G. Huet en donne une autre démonstration en utilisant le Principe de récurrence noethérienne (cf. [Hue80]).
□

Les résultats de confluence que nous présentons dans les chapitres 3 et 4 son montrés en utilisant une méthode qui s'avère très efficace pour les calculs sur lesquels nous travaillons. Cette méthode a été identifiée dans [Har89], où elle a été utilisée pour les combinateurs catégoriques. Dans [CHL91], on l'utilise pour montrer la confluence du $\lambda\sigma$-calcul faible, celle du $\lambda\sigma$-calcul sur les termes clos et la non-confluence du $\lambda\sigma'$-calcul sur les termes ouverts. Nous l'appelons *méthode d'interprétation* et la présentons ici puisqu'elle peut être formulée pour un système de réduction quelconque.

Lemme 1.4 (Méthode d'interprétation) *Soit $R = R_1 \cup R_2$ où R_1 est une réduction confluente et noethérienne et R_2 une réduction quelconque. Notons $R_1(f)$ la R_1-forme normale de f et supposons l'existence d'une réduction R' sur les R_1-formes normales qui satisfait :*

$$R' \subseteq R^* \quad et \quad (f \to_{R_2} g \quad \Rightarrow \quad R_1(f) \twoheadrightarrow_{R'} R_1(g)) .$$

Alors R' est confluente ssi R est confluente.

Démonstration
La preuve est simple (cf. [CHL91], lemme 1.1). La figure 1.1 illustre la partie ($\Rightarrow$) de la démonstration. C'est dans ce sens qu'on utilise en fait le lemme.
□

1.2.2 Systèmes de réécriture

Dans cette sous-section nous nous restreignons à $A = T_{\mathcal{F}}(V)$ une algèbre de termes de signature $\mathcal{F}$ sur l'ensemble de variables V.

Définition 1.10 *Un* contexte *est un terme qui contient exactement une occurrence d'un symbole spécial $\Box$, qui dénote un espace vide que nous appellerons un* trou. *Un contexte est noté $C[\,]$ ou simplement C. Le remplacement du trou par le terme $t \in T_{\mathcal{F}}(V)$ est noté $C[t]$.*

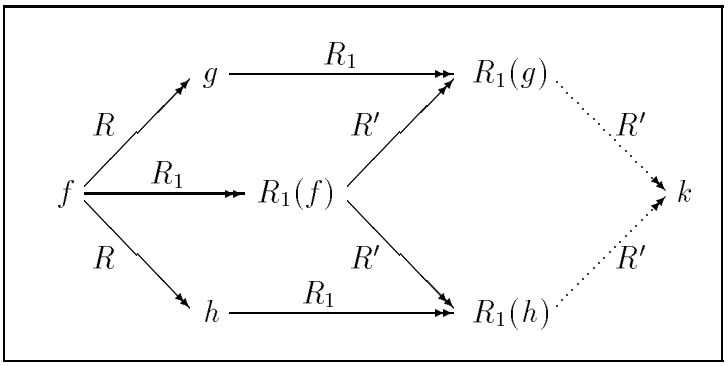


FIG. 1.1 - Méthode d'interprétation

Définition 1.11 *Une fonction* $\sigma : T_{\mathcal{F}}(V) \to T_{\mathcal{F}}(V)$ *qui satisfait*

$$\sigma(f(t_1, \cdots, t_n)) = f(\sigma(t_1), \cdots, \sigma(t_n))$$

pour tout $f \in \mathcal{F}$ *n-aire* ($n \geq 0$) *est appelée* substitution. *De sorte que* σ *est déterminée par sa restriction à* V.
On écrit $\sigma \leq \tau$ *s'il existe une substitution* ρ *telle que* $\tau = \sigma \circ \rho$ *(composition fonctionnelle), dans ce cas on dit que* σ *est* plus générale *que* τ.

Définition 1.12 *Une* règle de réduction *(ou règle de réécriture)* r *est une paire* (t, s) *de termes dans* $T_{\mathcal{F}}(V)$, *souvent écrite* $t \to_r s$, *telle que*

1. $t \notin V$,
2. $V(s) \subseteq V(t)$.

Définition 1.13 *Un* système de réécriture *est une paire* $(T_{\mathcal{F}}(V), \mathcal{R})$ *où* $\mathcal{R}$ *est un ensemble de règles de réécriture.*
La réduction R engendrée *par* $\mathcal{R}$ *est définie ainsi :*

$$t \to_R s \text{ ssi il existe } r = (t_r, s_r) \in \mathcal{R}, \text{ une substitution } \sigma \text{ et un contexte } C$$
$$\text{tels que } t = C[\sigma(t_r)] \text{ et } s = C[\sigma(s_r)].$$

Dans ce cas on dit que $\sigma(t_r)$ *est le* radical *de la réduction et* $s = C[\sigma(s_r)]$ *le* réduit *de* t.

Nous donnons maintenant la notion de *paire critique* et énonçons le critère de Knuth-Bendix pour établir la confluence locale.

Définition 1.14 *On dit que la substitution* σ *est un* unificateur *des termes* t *et* s *si* $\sigma(t) = \sigma(s)$. *Dans ce cas on dit que* t *et* s *sont* unifiables.

Théorème 1.2 (Unification) *Soient t et s deux termes unifiables. Il existe un unificateur τ qui est le plus général (upg), i.e. pour tout unificateur σ de t et s, on a $\tau \leq \sigma$.*

Démonstration
Voir [MM82], où un algorithme élégant est présenté pour trouver le upg.
□

Définition 1.15 *Soient $s \to t$ et $u \to v$ deux règles de réécriture. Soit w un sous-terme de u qui n'est pas une variable et C le contexte tel que $u = C[w]$.*
Supposons que s et w sont unifiables et soit σ un upg.
Le terme $\sigma(u)$ peut donc être réduit de deux manières :

$$\sigma(u) = \sigma(C[w]) \to \sigma(C[t]) \quad et \quad \sigma(u) \to \sigma(v)$$

La paire de réduits $(\sigma(C[t]), \sigma(v))$ est appelée une paire critique *obtenue par la* superposition *des règles $s \to t$ et $u \to v$.*
Dans le cas où $s \to t$ et $u \to v$ sont la même règle on demande que s soit unifiable avec un sous-terme propre *de u.*
Une paire critique (s, t) est convergente *s'il existe u tel que $s \twoheadrightarrow u$ et $t \twoheadrightarrow u$.*

Théorème 1.3 (Knuth-Bendix) *Un système de réécriture est WCR (localement confluent) ssi toute paire critique est convergente.*

Démonstration
Voir [KB70] ou [Hue80].
□

Dans l'annexe B nous reproduisons quelques sessions avec le logiciel KB, développé à l'INRIA, qui examine les paires critiques d'un système donné et teste leur convergence.

1.3 Le λ-calcul

L'étude de la fonctionalité se base principalement sur le λ-calcul, fort étudié depuis les travaux de A. Church (cf. [Chu41]). Rappelons le trait essentiel de ce calcul : c'est une théorie qui étudie plutôt l'aspect applicatif des fonctions, et pas celui compositionnel, comme dans le cas de la théorie des catégories. Une notion complémentaire à celle d'application est celle d'abstraction : à partir d'un terme a, qui contient éventuellement la variable x, la syntaxe fournit un terme $\lambda x.a$ dont l'interprétation informelle est la fonction a vue comme fonction de x.

Nous ne donnons ici que les définitions de base et les résultats qui seront utilisés dans la suite. La référence classique est [Bar84]. Cependant, nous suivrons la présentation de [HS86] en ce qui concerne la définition de la substitution. Nous choisissons

ce point de vue car, en évitant de travailler modulo α-conversion, on est plus proche des implémentations.

Définition 1.16 *Soit V un ensemble de variables. L'ensemble de termes du λ-calcul, noté Λ_V, est le plus petit ensemble tel que :*

1. $V \subset \Lambda_V$.
2. *Si $a \in \Lambda_V$ et $x \in V$, alors $\lambda x.a \in \Lambda_V$.*
   Intuitivement, $\lambda x.a$ représente la fonction qui à la variable x associe la valeur a. Un terme $\lambda x.a$ sera appelé une abstraction *et λx, l'*abstracteur.
3. *Si $a, b \in \Lambda_V$, alors $ab \in \Lambda_V$.*
   *On appellera ab l'*application *du terme a au terme b et on peut l'interpréter comme la valeur rendue par "la fonction" a appliquée à "l'argument" b.*

Dans la suite on abrégera ce type de définitions en présentant l'ensemble de termes à définir, dans ce cas Λ_V, ainsi :

$$a ::= x \mid ab \mid \lambda x.a \qquad (x \in V)$$

Définition 1.17 *On définit l'* ensemble de variables libres *d'un terme $a \in \Lambda_V$, noté $FV(a)$, par récurrence sur la structure de a :*

1. $FV(x) = \{x\}$
2. $FV(ab) = FV(a) \cup FV(b)$
3. $FV(\lambda x.a) = FV(a) - \{x\}$.

On dit qu'une variable x est liée *dans un terme quand celui-ci contient un sous-terme de la forme $\lambda x.a$.*

Définition 1.18 *Soient $a, b \in \Lambda_V$ et $x \in V$, on définit $a[\![b/x]\!]$ comme la* substitution *par b de toutes les occurrences libres de x dans a, avec éventuel changement de variables pour éviter des captures. Plus précisément :*

1. $x[\![b/x]\!] = b$
2. $y[\![b/x]\!] = y$ *pour toute variable $y \neq x$*
3. $(a_1 a_2)[\![b/x]\!] = (a_1[\![b/x]\!])(a_2[\![b/x]\!])$
4. $(\lambda x.a_1)[\![b/x]\!] = \lambda x.a_1$
5. $(\lambda y.a_1)[\![b/x]\!] = \lambda y.(a_1[\![b/x]\!])$ *si $y \neq x$, et $y \notin FV(b)$ ou $x \notin FV(a_1)$*

6. $(\lambda y.a_1)[\![b/x]\!] = \lambda z.(a_1[\![z/y]\!][\![b/x]\!])$ *si* $y \neq x$, $y \in FV(b)$ *et* $x \in FV(a_1)$.

Dans 6, on choisit z *comme la première variable qui n'appartient pas à* $FV(a_1 b)$.

La clause 6 est introduite pour éviter la capture de la variable y de b. En effet, par exemple, le terme $\lambda y.x$ représente la fonction constante dont la valeur est x. On souhaiterait que $(\lambda y.x)[\![y/x]\!]$ représente la fonction constante dont la valeur est y. Si on évaluait $(\lambda y.x)[\![y/x]\!]$ d'après la clause 5, on obtiendrait

$$(\lambda y.x)[\![y/x]\!] = \lambda y.y\,,$$

qui représente la fonction identité et non la fonction constante à valeur y. Tandis que si on fait l'évaluation d'après la clause 6, notre expectative est satisfaite. En effet,

$$(\lambda y.x)[\![y/x]\!] = \lambda z.(x[\![z/y]\!][\![y/x]\!]) = \lambda z.(x[\![y/x]\!]) = \lambda z.y\,,$$

qui est bien la fonction constante à valeur y.

Définition 1.19 *Soit* a *un terme qui contient une occurrence de* $\lambda x.c$, *et soit* $y \notin FV(c)$. *Le remplacement de ce* $\lambda x.c$ *par* $\lambda y.(c[\![y/x]\!])$ *est appelé* α-conversion *ou* changement de variable liée.
On dit que a *est* α-congruent *à* b, *noté* $a \equiv_\alpha b$ *si* b *est obtenu à partir de* a *par une série finie (éventuellement vide) de* α*-conversions.*

Deux termes α-congruents ont des interprétations identiques et ils jouent le même rôle dans les applications du λ-calcul. En fait, plusieurs auteurs identifient l'α-congruence avec l'identité syntaxique.

Un terme de la forme $(\lambda x.a)b$ représente un opérateur $\lambda x.a$ appliqué à un argument b. D'après l'interprétation informelle de $\lambda x.a$, sa valeur en b est calculée en substituant x par b dans a, de sorte que $(\lambda x.a)b$ peut être "simplifié" en $a[\![b/x]\!]$. Ce processus de simplification sera précisé dans la définition suivante.

Remarque 1.1 *Un* contexte dans le λ-calcul *est défini de manière analogue à un contexte dans une algèbre de termes (voir définition 1.10).*

Définition 1.20 *Un terme de la forme* $(\lambda x.c)d$ *est appelé un* β-radical *et le terme* $c[\![d/x]\!]$, *son* réduit.
On définit la β-réduction, *appelée aussi* β*-contraction et notée* $\rightarrow_\beta$, *par*

$$a \rightarrow_\beta b \quad \textit{ssi il existe un contexte } C \textit{ et des termes } c, d \textit{ tels que}$$
$$a = C[(\lambda x.c)d] \textit{ et } b = C[c[\![d/x]\!]]\,.$$

Définition 1.21 *Le* λ-calcul *est le système de réduction* $(\Lambda_V, (\beta \cup \equiv_\alpha))$.
Les dérivations de ce calcul sont notées $\twoheadrightarrow_\beta$ *. Il y a donc abus de notation en sous-entendant les* α*-conversions.*

La β-réduction n'est pas symétrique, mais elle engendre la rélation d'équivalence suivante :

Définition 1.22 *Un terme* a *est* β*-égal à un terme* b*, noté* $a =_\beta b$*, ssi* b *est obtenu à partir de* a *par une série finie (ou vide) de* β*-réductions, des* β*-réductions inversées et des changements de variables liées. Plus précisément,* $a =_\beta b$ *ssi il existe* $n \geq 0$ *et des termes* $c_0, \cdots, c_n$ *tels que*

- $c_0 = a$ *et* $c_n = b$.
- $(\forall i \leq n-1)(c_i \rightarrow_\beta c_{i+1}$ *ou* $c_{i+1} \rightarrow_\beta c_i$ *ou* $c_i \equiv_\alpha c_{i+1})$.

Le théorème suivant est un résultat classique du λ-calcul (cf. [Bar84] et [HS86]).

Théorème 1.4 *Le* λ*-calcul est confluent mais il n'est pas noethérien.*

A condition de travailler modulo α-conversion on peut remplacer les clauses 4, 5 et 6 de la définition 1.18 par la clause suivante (cf [Bar84], p. 26) :

$$(\lambda y.a_1)[\![b/x]\!] = \lambda y.(a_1[\![b/x]\!]) .$$

Pour opérer ainsi, il faut établir la convention suivante : toutes les variables liées des termes intervenant dans une définition, preuve, etc., doivent être choisies de sorte qu'elles diffèrent des variables libres.

Ce choix simplifie notablement les démonstrations mais cette solution n'est pas directement utilisable dans une implémentation.

On retrouve donc le problème bien connu des λ-calculistes : celui de la capture des variables par l'opération de substitution. Etant donné que la solution apportée par l'α-conversion n'est pas satisfaisante du point de vue implémentation, car elle est coûteuse et difficile à gérer, d'autres solutions ont été proposées :

1. La compilation des expressions du λ-calcul dans la Logique Combinatoire, théorie sans variables décrite par Schönfinkel et Curry (cf [CF58] et [CHS72]).
2. La notation de de Bruijn, dont nous donnons les notions essentielles dans la section suivante.
3. La Logique Combinatoire Catégorique développée par J. Lambek puis P.-L. Curien (cf [Cur86]), à l'origine comme modèle syntaxique des Catégories Cartésiennes Fermés.

On peut trouver une discussion sur ces différentes approches dans la thèse de T. Hardin ([Har87]).

Revenons maintenant au formalisme de de Bruijn.

1.4 Le formalisme de de Bruijn

Ce formalisme (cf. [dB72] et [dB78]) est basé sur le remplacement des noms des variables du λ-calcul par des numéros, appelés *indices de de Bruijn*, rendant compte de la portée de la variable ainsi remplacée. Plus précisément ce numéro exprime la *hauteur de liaison* de l'occurrence de la variable correspondante, i.e. le nombre de λ's à franchir quand on remonte de l'occurrence examinée jusqu'à l'abstracteur qui la lie. Par exemple,

$$\lambda x.\lambda y.xy \quad \text{devient} \quad \lambda\lambda(\mathtt{21})$$

$$\lambda x.\lambda y.(x(\lambda z.zx))y \quad \text{devient} \quad \lambda(\lambda(\mathtt{2}(\lambda(\mathtt{13}))\mathtt{1}))$$

Remarquons que, dans le dernier exemple, la même variable x a été traduite par $\mathtt{2}$ et $\mathtt{3}$ selon ses différentes positions, tandis que les variables z et y sont devenues le même numéro de de Bruijn $\mathtt{1}$. Il n'y aura donc pas de correspondance directe entre numéros et noms de variables.

Même si la notation de de Bruijn est difficile à lire, elle s'avère très efficace pour un traitement formel de la substitution. Nous commençons donc par introduire l'ensemble de termes du λ-calcul à la de Bruijn.

Définition 1.23 *L'ensemble des termes, noté Λ, du λ-calcul décrit avec la notation de de Bruijn est donné par :*

$$a ::= \mathtt{n} \mid ab \mid \lambda a \qquad \textit{où} \quad \mathtt{n} \in \mathbb{N}.$$

Pour exprimer un terme a du λ-calcul avec des variables libres $x_1, \ldots, x_k$ dans cette notation il suffit de considérer a comme sous-terme de $\lambda x_1 \ldots x_k.a$. Par exemple :

$$\lambda x.xy \quad \text{devient} \quad \lambda(\mathtt{12})$$

$$(\lambda x.xy)y \quad \text{devient} \quad (\lambda(\mathtt{12}))\mathtt{1}$$

Pour définir la β-réduction à la de Bruijn, nous devons définir la substitution d'une variable par un terme, disons b, dans un terme, disons a. Il faudra donc identifier, parmi les numéros du terme a, ceux qui correspondent à la variable qu'on est en train de substituer. On peut calculer de façon récursive ces hauteurs de liaison à l'aide d'une famille de compteurs, décrite informellement ainsi :

1. On démarre la traversée de a avec un seul compteur initialisé à 1.
2. Au passage d'un nœud application, on crée deux exemplaires du compteur de façon à en obtenir un pour chaque branche.
3. Au passage d'un nœud abstraction on incrémente le compteur.

4. Lorsqu'on arrive à l'occurrence d'un numéro :

   4.i) S'il est supérieur à la valeur du compteur, on le diminue de 1, puisqu'il existe un λ de moins sur le chemin entre ce numéro et l'abstracteur qui le lie.

   4.ii) Si ce numéro est égal à la valeur du compteur, disons n, il doit être remplacé par b, qui va donc se trouver ainsi sous $n-1$ λ's. Il faudra donc ajuster les numéros de b de façon à ne pas modifier les liaisons à l'intérieur de b. Nous faisons ceci à l'aide d'une autre famille de fonctions que nous appellerons *fonctions d'actualisation.*

   4.iii) Si le numéro est inférieur à la valeur du compteur, alors il est lié par un abstracteur interne à a et il ne doit pas être modifié.

Nous commençons par définir formellement les fonctions d'actualisation :

Définition 1.24 *Les* fonctions d'actualisation $U_i^n : \Lambda \to \Lambda$ *pour* $i \geq 0$ *et* $n \geq 1$ *sont définies par récurrence ainsi :*

$$\begin{aligned} U_i^n(ab) &= U_i^n(a)\, U_i^n(b) \\ U_i^n(\lambda a) &= \lambda(U_{i+1}^n(a)) \\ U_i^n(\mathtt{m}) &= \begin{cases} \mathtt{m+n-1} & si \quad m > i \\ \mathtt{m} & si \quad m \leq i\,. \end{cases} \end{aligned}$$

Nous utiliserons dans le chapitre 3 la remarque suivante, dont la preuve par récurrence sur n n'offre aucune difficulté :

Remarque 1.2 *Soient* $b \in \Lambda$ *et* $n \geq 0$, *alors* $U_n^1(b) = b$.

Nous pouvons définir maintenant la famille de fonctions qui effectuent la substitution :

Définition 1.25 *Les* substitutions à la hauteur n, *pour* $n \geq 1$, *du terme* b *dans le terme* a, *notées* $a\{\!\{\mathtt{n} \leftarrow b\}\!\}$, *sont définies par récurrence sur* a *ainsi :*

$$\begin{aligned} (a_1 a_2)\{\!\{\mathtt{n} \leftarrow b\}\!\} &= (a_1\{\!\{\mathtt{n} \leftarrow b\}\!\})\,(a_2\{\!\{\mathtt{n} \leftarrow b\}\!\}) \\ (\lambda a_1)\{\!\{\mathtt{n} \leftarrow b\}\!\} &= \lambda(a_1\{\!\{\mathtt{n+1} \leftarrow b\}\!\}) \\ \mathtt{m}\{\!\{\mathtt{n} \leftarrow b\}\!\} &= \begin{cases} \mathtt{m-1} & si \quad m > n \\ U_0^n(b) & si \quad m = n \\ \mathtt{m} & si \quad m < n\,. \end{cases} \end{aligned}$$

Nous avons déjà les éléments nécessaires pour définir la β-réduction, dans le formalisme de de Bruijn :

Définition 1.26 *La* β-réduction [2] *est définie sur* Λ *ainsi :*

$$a \to_\beta b \quad \textit{ssi il existe un contexte } C \textit{ et des termes } c, d \in \Lambda \textit{ tels que}$$
$$a = C[(\lambda c)\, d] \quad \textit{et} \quad b = C[c\{\!\{1 \leftarrow d\}\!\}]$$

Le λ-calcul à la de Bruijn, *abrégé* $\lambda\beta$-calcul, *ou* λ-calcul *quand le contexte sera suffisamment clair, est le système de réduction* (Λ, β) .

Théorème 1.5 *Le* $\lambda\beta$*-calcul est confluent mais il n'est pas noethérien.*

Démonstration
Le $\lambda\beta$-calcul et le λ-calcul classique sont isomorphes (cf. [Mau85]). La confluence et la non-noethérianité du λ-calcul (cf. théorème 1.4) sont donc transmises au $\lambda\beta$-calcul.

Une preuve de la confluence qui n'utilise pas l'isomorphisme mentionné précédemment, est donnée dans le chapitre 3 (cf. corollaire 3.6).
□

Nous remarquons que l'opération de substitution dans Λ est définie, comme pour le λ-calcul classique, dans le méta-langage. Ceci est une désavantage au niveau des implémentations. Les calculs que nous introduisons à partir de la section 1.6 sont formulés dans un langage qui intègre les substitutions, tout en gardant une notation à la de Bruijn pour mieux gérer les problèmes de captures.

1.5 Extensionalité

Le concept d'égalité fonctionelle utilisé généralement en mathématique est appelé "extensionnel". Ceci veut dire que deux fonctions sur le même domaine sont considérées égales quand elles prennent la même valeur pour chaque argument :

$$((\forall x) F(x) = G(x)) \Rightarrow F = G\,.$$

1.5.1 Dans le λ-calcul classique

La β-égalité introduite dans la définition 1.22 est plutôt "intensionnelle", i.e. il existe des termes $a, b \in \Lambda_V$ tels que pour toute variable $x \in V$, on a $ax =_\beta bx$ mais $a \neq_\beta b$. Par exemple, pour $y, z \in V$ prendre :

$$a = y \qquad \text{et} \qquad b = \lambda z.yz\,.$$

[2] Le symbole $\to_\beta$ est donc surchargé : même notation que pour le λ-calcul classique. Le contexte sera toujours suffisamment clair pour donner à $\to_\beta$ l'interprétation correcte.

Cependant, on peut obtenir une notion d'égalité extensionnelle en ajoutant une autre règle au λ-calcul et en définissant l'égalité correspondant à ce nouveau calcul.

Définition 1.27 *Un terme de la forme $\lambda x.cx$ où $x \notin FV(c)$ est appelé un* η-radical *et le terme c, son* réduit.
On définit la η-réduction, *notée $\rightarrow_\eta$, par*

$$a \rightarrow_\eta b \text{ ssi il existe un contexte } C, \text{ une variable } x \text{ et un terme } c \text{ tels que } a = C[\lambda x.cx] \ , \ b = C[c] \text{ et } x \notin FV(c) .$$

Définition 1.28 *Le* $\lambda\eta$-calcul *est le système de réduction $(\Lambda_V, (\beta \cup \eta \cup \equiv_\alpha))$.*
Les dérivations de ce calcul sont notées $\twoheadrightarrow_{\beta\eta}$ (α-conversions sous-entendues).

De même que la β-réduction (cf. définition 1.22), la $\beta\eta$-réduction s'étend à une relation d'équivalence :

Définition 1.29 *Un terme a est $\beta\eta$-égal à un terme b, noté $a =_{\beta\eta} b$, ssi b est obtenu à partir de a par une série finie (ou vide) de $\beta\eta$-réductions, des $\beta\eta$-réductions inversées et des changements de variables liées.*

Les résultats suivants sont classiques (cf. [Bar84] ou [HS86]) :

Lemme 1.5 *La $\beta\eta$-égalité est extensionnelle.*

Théorème 1.6 *Le $\lambda\eta$-calcul est confluent.*

1.5.2 Dans le λ-calcul à la de Bruijn

Dans cette sous-section nous traduisons la η-règle dans le λ-calcul avec notation à la de Bruijn. Remarquons d'abord que cette règle est conditionnelle : pour l'appliquer il faut vérifier qu'une certaine variable n'est pas libre dans un certain terme.

Nous devons donc traduire la η-règle et la condition sur la variable. La règle aura cette forme :

$$\lambda(a\mathtt{1}) \rightarrow a' \quad \text{si } \mathtt{1} \text{ n'est pas libre dans } a \ .$$

Appelons cette condition $C\eta_1$. Pour l'interpréter dans le contexte de de Bruijn, nous précisons d'abord le concept de λ-hauteur :

Définition 1.30 *Soit $a \in \Lambda$ et γ une occurrence dans a. La* λ-hauteur *de γ dans a, notée $h(\gamma, a)$ est le nombre de λ's que l'on trouve à des occurrences préfixes de γ. Plus précisément :*

$$\begin{aligned} h(1, \mathtt{m}) &= 0 \\ h(1\gamma, ab) &= h(\gamma, a) \\ h(2\gamma, ab) &= h(\gamma, b) \\ h(1\gamma, \lambda a) &= h(\gamma, a) + 1 \end{aligned}$$

Par abus de langage nous écrivons $h(a/\gamma, a)$, ou simplement $h(a)$, au lieu de $h(\gamma, a)$ quand l'occurrence γ et le terme a sont clairs d'après le contexte.

Exemple 1.3 *Considérons le terme* $a = \lambda(\mathtt{23})$, *qui correspond à* $\lambda x.yz$ *dans la notation classique. Nous affirmons que l'occurrence de* $\mathtt{2}$ *dans* a *"joue le rôle" de* $\mathtt{1}$ *en tant que variable libre. En effet, si nous affectons le terme* a *par une méta-substitution de la forme* $\{\!\{\mathtt{1} \leftarrow b\}\!\}$, *c'est bien la variable* $\mathtt{2}$ *qui sera concernée par cette substitution :*

$$(\lambda(\mathtt{23}))\{\!\{\mathtt{1} \leftarrow b\}\!\} = \lambda(\mathtt{2}\{\!\{\mathtt{2} \leftarrow b\}\!\}\,\mathtt{3}\{\!\{\mathtt{2} \leftarrow b\}\!\}) = \lambda(U_0^2(b)\,\mathtt{2})\,.$$

Signalons que $h(\mathtt{2}, a) = h(\mathtt{3}, a) = 1$.

Cet exemple suggère la remarque suivante :

Remarque 1.3 *Soit* $\mathtt{m}$ *un indice de de Bruijn et* γ *une occurrence de* $\mathtt{m}$ *dans un terme* a.

1. $\mathtt{m}$ *est libre dans* a, *à l'occurrence* γ *ssi* $m > h(\gamma, a)$.
2. $\mathtt{m}$ *joue le rôle de* $\mathtt{1}$ *en tant que variable libre ssi* $m = h(\gamma, a) + 1$.

Il sera utile de disposer, non seulement de la condition $C\eta_1$, mais d'une famille de conditions $C\eta_n$, où $n \geq 1$.

Définition 1.31 *Un terme* $a \in \Lambda$ *satisfait la* condition $C\eta_n$ *ssi pour tout indice de de Bruijn* $\mathtt{m}$ *et toute occurrence* γ *de* $\mathtt{m}$ *dans* a, *on a* $m \neq h(\gamma, a) + n$.

Il nous reste encore à étudier le réduit a' de $\lambda(a\mathtt{1})$. Comme la η-réduction efface un λ, les indices qui jouent le rôle de variables libres doivent être diminués de 1. Nous proposons donc :

Définition 1.32 *La* η*-réduction*[3] *est définie sur* Λ *par :*

$a \to_\eta b$ *ssi il existe un contexte* C *et un terme* c *qui satisfait* $C\eta_1$ *tels que*

$$a = C[\lambda(c\mathtt{1})] \quad , \quad b = C[c^{\downarrow}]\,.$$

où $c^{\downarrow}$ *est obtenu à partir de* c *en remplçant tout indice de de Bruijn* $\mathtt{m}$ *à chaque occurrence* γ *qui vérifie* $m > h(\gamma, c)$ *par l'indice* $\mathtt{m} - \mathtt{1}$.
Le $\lambda\beta\eta$-calcul *est le système de réduction* $(\Lambda, (\beta \cup \eta))$.

Pour finir cette section nous proposons une présentation plus opérationnelle de la η-règle.

Définition 1.33 *Soient* $a \in \Lambda$ *et* $n \geq 1$. *On définit partiellement le terme* a_n *par récurrence sur* a *ainsi :*

$$\begin{array}{rcl}
(b\,c)_n & = & b_n\,c_n \\
(\lambda b)_n & = & \lambda b_{n+1} \\
\mathtt{m}_{\,n} & = & \begin{cases} \mathtt{m} - \mathtt{1} & si \quad m > n \\ \text{indéfini} & si \quad m = n \\ \mathtt{m} & si \quad m < n\,. \end{cases}
\end{array}$$

[3] Même remarque que celle faite pour la β-réduction dans la note précédente, page 15.

On sous-entend que chaque membre gauche est défini quand tous les a_n apparaissant dans le membre droit correspondant sont définis.

On montre aisément par récurrence sur a le lemme suivant :

Lemme 1.6 *Soient $a \in \Lambda$ et $n \geq 1$.*

1. *a satisfait $C\eta_n$ ssi a_n est défini.*
2. *Si a satisfait $C\eta_1$, alors $a_1 = a^{\downarrow}$.*

Le résultat suivant est classique :

Théorème 1.7 *Le $\lambda\beta\eta$-calcul est confluent.*

Démonstration
Nous obtenons ce théorème comme corollaire d'un résultat plus général (cf. théorème 4.3 et corollaire 4.5).
□

Dans le chapitre 4 nous abordons l'extentionnalité dans les calculs que nous introduisons dans les sections suivantes.

1.6 Le $\lambda\sigma$-calcul

Le $\lambda\sigma$-calcul, introduit dans [ACCL90], ainsi que la logique combinatoire catégorique [Cur86], sont des formalismes permettant de rendre compte de manière explicite du calcul de la substitution. Nous avons vu que la β-règle du λ-calcul est définie à partir d'une substitution implicite et que cette opération de substitution est définie par récurrence sur la structure du terme où la substitution doit être effectuée. Dans le $\lambda\sigma$-calcul au contraire, la substitution est gerée par un constructeur explicite de la syntaxe : un terme est soit une variable, soit une application, soit une abstraction, soit enfin une clôture (ou fermeture) $a[s]$, où, en simplifiant, la substitution s est une liste de termes. Les difficultés liées à la capture des variables ont conduit les auteurs de [ACCL90] à opter pour une notation à la de Bruijn sans variables dans le λ-calcul (et donc dans le $\lambda\sigma$-calcul).

Des essais de description de la substitution à l'intérieur même du langage apparaissent déjà dans [Sta78] et [Sle85]. Le premier traitement systématique de la substitution basé sur le langage des Catégories est fourni par la théorie des Combinateurs Catégoriques, CCL (cf. [Cur86]), où il est assuré par le sous-système nommé SUBST (cf. Annexe C).

Le langage du $\lambda\sigma$-calcul est un langage à deux sortes : sorte `terme` des *termes proprement dits* et sorte `substitution` des *substitutions explicites*. Celles-ci peuvent être interprétées comme des suites des termes, et le résultat d'effectuer une

substitution dans un terme, comme le terme obtenu par le remplacement des occurrences du n-ième indice de de Bruijn dans le terme par le n-ième terme de la suite. Cette interprétation intuitive est developpée et illustrée avec plusieurs exemples dans [ACCL90].

Avant de poursuivre la discussion, nous donnons la syntaxe et les règles du calcul :

Définition 1.34 *La syntaxe du* $\lambda\sigma$-calcul *est donnée par :*

$$\begin{array}{ll} \textbf{Termes} & a ::= \mathtt{1} \mid ab \mid \lambda a \mid a[s] \\ \textbf{Substitutions} & s ::= id \mid \uparrow \mid a \cdot s \mid s \circ t\,. \end{array}$$

L'ensemble, noté $\lambda\sigma$*, des règles du* $\lambda\sigma$*-calcul est donné dans la figure 1.5. Nous appelons* σ *l'ensemble de règles* $\lambda\sigma - \{(Beta)\}$ *. Le* σ-calcul *est le calcul dont l'ensemble de règles est* σ*.*

Notation 1.1 *On note* $\Lambda\sigma^t$ *l'ensemble des termes proprement dits et* $\Lambda\sigma^s$ *l'ensemble des substitutions du* $\lambda\sigma$*-calcul. Enfin,* $\Lambda\sigma = \Lambda\sigma^t \cup \Lambda\sigma^s$ *.*

Définition 1.35 *Pour toute substitution* s *on définit l'* itération de la composition *par récurrence ainsi :*

$$\begin{array}{lcl} s^1 & = & s \\ s^{n+1} & = & s \circ s^n \end{array}$$

On convient $s^0 = id$ *.*

Remarquons d'abord que le seul indice de de Bruijn qui apparaît explicitement dans le langage est $\mathtt{1}$, mais on peut coder l'indice $\mathtt{n}$ par le terme $\mathtt{1}[\uparrow^{n-1}]$. Ce faisant, on a $\Lambda \subset \Lambda\sigma^t$.

Remarquons aussi que, en plus des opérateurs classiques d'application et d'abstraction, le langage possède :

- Un opérateur binaire appelé *clôture* ou *fermeture* dont la notation semi-infixe est $-[-]$, et qui représente l'opération de substitution. Le terme $a[s]$ est donc interprété comme le terme a où la substitution s doit être effectuée.
- Une substitution id , appelée *identité*, et qui sera interprétée comme la suite des entiers de de Bruijn $(\mathtt{n})_{n\geq 1}$.
- Une substitution $\uparrow$, appelée *shift* et interprétée comme la suite $(\mathtt{n}+\mathtt{1})_{n\geq 1}$.
- Un opérateur binaire appelé *cons*, dont la notation infixe est "$\cdot$". La substitution $a \cdot s$ est interpretée comme la suite obtenue en ajoutant le terme a au commencement de la suite représentant s .

– Un opérateur binaire appelé *composition*, dont la notation infixe est $\circ$. Si la substitution s est interprétée par la suite $(a_n)_{n\geq 1}$ alors la substitution $s \circ t$ est interprétée par la suite $(a_n[t])_{n\geq 1}$.

La β-réduction du λ-calcul est simulée dans le $\lambda\sigma$-calcul en deux étapes. La première, obtenue par application de la règle nommée *(Beta)*, consiste à déclencher la substitution. La seconde étape effectue la propagation de cette substitution jusqu'aux variables concernées, par réécriture avec l'ensemble de règles σ.

Exemple 1.4 *Le terme* $a = \lambda z.((\lambda x.(\lambda y.xy))z)$ *se traduit par* $a' = \lambda((\lambda\lambda(2\,1))1)$. *Dans le* λ*-calcul on a la réduction suivante :*

$$a \rightarrow_{\beta} \lambda z.\lambda y.zy$$

Cette réduction se traduit dans la $\lambda\sigma$*-dérivation :*

$$a' \rightarrow_{(Beta)} \lambda((\lambda(2\,1))[1 \cdot id]) \rightarrow_{(Abs)} \lambda(\lambda((2\,1)[1 \cdot ((1 \cdot id) \circ \uparrow)])) \twoheadrightarrow_{(Map),(IdL)}$$

$$\twoheadrightarrow \lambda(\lambda((2\,1)[1 \cdot 2 \cdot \uparrow])) \rightarrow_{(App)} \lambda(\lambda(2[1 \cdot 2 \cdot \uparrow]\ 1[1 \cdot 2 \cdot \uparrow])) \twoheadrightarrow_{(Clos),(VarCons)}$$

$$\twoheadrightarrow \lambda(\lambda(1[\uparrow \circ (1 \cdot 2 \cdot \uparrow)]\ 1)) \rightarrow_{(ShiftCons)} \lambda(\lambda(1[2 \cdot \uparrow)]\ 1)) \rightarrow_{(VarCons)} \lambda(\lambda(2\,1))$$

□

Remarquons que, d'après la syntaxe que nous avons donnée, les termes/substitutions que nous avons définis, sont en fait des termes/substitutions clos, puisque nous n'avons pas inclu des variables de sorte `terme` ni de sorte `substitution`. Nous appellerons explicitement *termes ouverts* ceux décrits par la syntaxe correspondante, incluant, en plus, des variables des deux sortes. Ces variables seront parfois appelées *méta-variables* pour les différencier de celles codées par les indices de de Bruijn.

Définition 1.36 *On définit, par récurrence simultanée sur la structure, les termes et substitutions ouverts par :*

Termes ouverts	$a ::= X \mid \mathtt{1} \mid ab \mid \lambda a \mid a[s]$,
Substitutions ouvertes	$s ::= \mathbf{x} \mid id \mid \uparrow \mid a \cdot s \mid s \circ t$

où X *parcourt un ensemble dénombrable de variables de type* `terme` *et* $\mathbf{x}$ *parcourt un ensemble dénombrable de variables de type* `substitution`.
On note Λ_ς *l'ensemble des termes et substitutions ouverts.*

Nous résumons maintenant les propriétés essentielles du $\lambda\sigma$-calcul et du σ-calcul :

Proposition 1.1 *Le* σ*-calcul est localement confluent sur* Λ_ς.

Démonstration
Remarquons d'abord que les règles préservent les types. On montre que le système non typé est localement confluent en utilisant le logiciel KB, développé à l'INRIA, qui est une implémentation de l'algorithme de complétion de Knuth-Bendix (cf. théorème 1.3).

Nous transcrivons le compte-rendu de la session montrant la confluence locale de σ dans l'annexe B.1.
□

Théorème 1.8 *Le σ-calcul est noethérien sur $\Lambda\varsigma$.*

Démonstration
Le chapitre 2 sera dédié à la normalisation forte de σ sur $\Lambda\sigma$. La noethérianité s'étend facilement à $\Lambda\varsigma$. En effet, une dérivation infinie dans $\Lambda\varsigma$ produit une dérivation infinie dans $\Lambda\sigma$ par remplacement de toutes les variables de type terme par, disons $\mathtt{1}$, et de celles de type substitution par, disons id.
□

Corollaire 1.1 *Le σ-calcul est confluent sur $\Lambda\varsigma$.*

Démonstration
C'est une conséquence immédiate du lemme de Newman (cf. lemme 1.3) et des deux résultats précédents.
□

Nous avons donc existence et unicité des formes normales pour le σ-calcul sur $\Lambda\varsigma$ et donc sur $\Lambda\sigma$.

Notation 1.2 *Soit $f \in \Lambda\varsigma$, on note $\sigma(f)$ la σ-forme normale de f.*

Théorème 1.9 *Le $\lambda\sigma$-calcul est confluent sur $\Lambda\sigma$.*

Démonstration
Voir [ACCL90], théorème 3.2. Cette preuve est basée sur la confluence de σ, celle du λ-calcul et la technique d'interprétation (cf. lemme 1.4).
□

1.7 Le $\lambda\sigma'$-calcul

Le $\lambda\sigma$-calcul n'est pas localement confluent sur les termes ouverts. En effet, considérons par exemple la paire critique suivante :

$$\begin{array}{lcll} ((\lambda X)Y)[\mathbf{x}] & \to & X[Y\cdot id][\mathbf{x}] & \twoheadrightarrow\ X[Y[\mathbf{x}]\cdot\mathbf{x}] \\ ((\lambda X)Y)[\mathbf{x}] & \to & ((\lambda X)[\mathbf{x}])\,(Y[\mathbf{x}]) & \twoheadrightarrow\ X[Y[\mathbf{x}]\cdot(\mathbf{x}\circ id)]\,. \end{array}$$

Même en se restreignant aux termes semi-clos (ceux qui admettent des variables de sorte terme, mais pas de sorte substitution) on n'obtient pas la confluence locale.

Avant d'introduire les règles qui doivent être ajoutées pour avoir la confluence locale, nous définissons formellement l'ensemble des termes semi-clos. C'est l'ensemble des termes qui nous intéresse et sur lequel porteront nos résultats de confluence (cf. chapitres 3 et 4).

Définition 1.37 *L'ensemble des* termes semi-clos, *noté* ΛX, *est défini par récurrence simultanée ainsi :*

$$\begin{array}{ll} \textbf{Termes} & a ::= X \mid \mathtt{1} \mid ab \mid \lambda a \mid a[s] \\ \textbf{Substitutions} & s ::= id \mid \uparrow \mid a \cdot s \mid s \circ t\,, \end{array}$$

où X parcourt un ensemble dénombrable de variables. On note ΛX^t l'ensemble des termes proprement dits semi-clos. On note de même ΛX^s l'ensemble des substitutions semi-closes.

Le contre-exemple que nous venons de donner peut être adapté facilement aux cas semi-clos. En effet :

$$\begin{array}{lcll} ((\lambda X)\mathtt{1})[Y \cdot id] & \to & X[\mathtt{1} \cdot id][Y \cdot id] & \twoheadrightarrow \quad X[Y \cdot (Y \cdot id)] \\ ((\lambda X)\mathtt{1})[Y \cdot id] & \to & ((\lambda X)[Y \cdot id])\,(\mathtt{1}[Y \cdot id]) & \twoheadrightarrow \quad X[Y \cdot (Y[id] \cdot id)]\,. \end{array}$$

Pour que cette paire critique devienne convergente nous proposons d'ajouter la règle : $a[id] \to a$. Avec cette règle deux paires critiques apparaissent :

Avec la règle *(Map)* :

$$\begin{array}{lcll} (a \cdot s)[id] & \to & a[id] \cdot (s \circ id) & \to \quad a \cdot (s \circ id) \\ (a \cdot s)[id] & \to & a \cdot s\,, & \end{array}$$

qui suggère l'ajout de la règle $s \circ id \to s$.

Avec la règle *(Abs)* :

$$\begin{array}{lcll} (\lambda a)[id] & \to & \lambda(a[\mathtt{1} \cdot (id \circ \uparrow)]) & \to \quad \lambda(a[\mathtt{1} \cdot \uparrow]) \\ (\lambda a)[id] & \to & \lambda a\,, & \end{array}$$

qui suggère l'ajout de la règle $\mathtt{1} \cdot \uparrow \to id$.

Cette dernière règle crée encore une autre paire critique avec la règle *(Map)* :

$$\begin{array}{lcll} (\mathtt{1} \cdot \uparrow)[s] & \to & id \circ s & \to \quad s \\ (\mathtt{1} \cdot \uparrow)[s] & \to & \mathtt{1}[s] \cdot (\uparrow \circ s)\,, & \end{array}$$

qui suggère d'introduire enfin la règle $\mathtt{1}[s] \cdot (\uparrow \circ s) \to s$.

Ces quatre règles sont suffisantes pour établir la confluence locale (cf. proposition 1.2). Nous définissons donc :

Définition 1.38 *Le* σ'-calcul *est obtenu par l'ajout au* σ*-calcul de l'ensemble, noté* $SPID$*, des quatre règles suivantes :*

$$\begin{array}{lrcl} (Id) & a[id] & \longrightarrow & a \\ (IdR) & s \circ id & \longrightarrow & s \\ (VarShift) & \mathtt{1} \cdot \uparrow & \longrightarrow & id \\ (SCons) & \mathtt{1}[s] \cdot (\uparrow \circ s) & \longrightarrow & s \end{array}$$

et l'élimination de (VarId) et (ShiftId) qui sont des instances de (Id) et (IdR), respectivement. Le $\lambda\sigma'$-calcul *est le calcul obtenu par l'ajout de la règle (Beta) au* σ'*-calcul. L'ensemble des règles du* $\lambda\sigma'$*-calcul est presenté dans la figure 1.6.*

Même si ces règles additionnelles sont justifiées d'un point de vue théorique, la confluence sur les termes clos, come on vient de le voir, peut être établie sans elles. En plus, pour les termes et substitutions clos on peut montrer (cf. [ACCL90]) que les règles *(Id)* et *(IdR)* sont admissibles dans le σ-calcul, i.e. que toute instance de ces règles peut être simulée dans le σ-calcul. Ceci n'est pas le cas pour le calcul semi-clos (i.e. le calcul sur ΛX) : $X[id] \to X$ ne peut pas être obtenu dans le σ-calcul.

La règle *(SCons)* suggère que le cons du premier terme d'une substitution avec le reste réduit à la substitution originale. Elle est donc en rapport avec la règle dite de "surjective pairing" du λ-calcul avec couples[4]. J. W. Klop a montré que cette règle détruit la confluence du λ-calcul [Klo80].

Dans les chapitres 3 et 4 nous utiliserons fréquemment les $SPID$-formes normales, dont l'existence et unicité sont garanties par le lemme suivant.

Lemme 1.7 *La* $SPID$*-réduction est confluente et noethérienne.*

Démonstration
On vérifie la confluence locale à l'aide du logiciel KB (cf. Annexe B.3). La terminaison est garantie par le fait que toutes les règles diminuent la taille du terme. On conclut donc grâce au lemme de Newman.
□

Notation 1.3 *La* $SPID$*-forme normale d'un terme* $f \in \Lambda X$ *est notée* $S(f)$.

[4] La syntaxe du λ-calcul avec couples est donnée par :

$$a ::= x \mid ab \mid \lambda x.a \mid \langle a, b \rangle \mid fst(a) \mid snd(a) .$$

L'ensemble de règles est obtenu en ajoutant à la règle β les regles suivantes :

$$\begin{array}{ll} (Fst) & fst(\langle a, b \rangle) \to a \\ (Snd) & snd(\langle a, b \rangle) \to b \\ (SP) & \langle fst(a), snd(a) \rangle \to a . \end{array}$$

Nous énonçons maintenant quelques propriétés fondamentales du σ'-calcul et du $\lambda\sigma'$-calcul qui seront démontrées dans cette thèse. Nous renvoyons le lecteur au chapitre correspondant à chacune d'elles.

Proposition 1.2 *Le σ'-calcul et le $\lambda\sigma'$-calcul sont localement confluents sur les termes et substitutions ouverts.*

Démonstration
Comme nous l'avons déjà fait pour le σ-calcul, nous testons la confluence locale à l'aide du logiciel KB. Nous transcrivons dans l'annexe B.2 la partie de la session où les nouvelles règles entrent en jeu.
□

Théorème 1.10 *Le σ'-calcul est noethérien sur $\Lambda\varsigma$.*

Démonstration
La section 2.8 du chapitre 2 sera dédiée à la démonstration de la noethérianité sur $\Lambda\sigma$ en utilisant essentiellement la noethérianité du σ-calcul et un résultat de commutation entre les anciennes règles et les règles ajoutées (cf. lemme 2.4).
De même que pour le σ-calcul (cf. théorème 1.8), le résultat s'étend à $\Lambda\varsigma$.
□

Corollaire 1.2 *Le σ'-calcul est confluent sur $\Lambda\varsigma$.*

Démonstration
C'est une conséquence immédiate du lemme de Newman (cf. lemme 1.3) et des deux résultats précédents.
□

Quant à la confluence du $\lambda\sigma'$-calcul, nous avons le résultat négatif suivant :

Théorème 1.11 *Le $\lambda\sigma'$-calcul n'est pas confluent sur l'ensemble des termes et substitutions ouverts $\Lambda\varsigma$.*

Démonstration
Voir [CHL91], lemme 3.9.
□

Mais sur les termes semi-clos, et donc sur les termes clos, la confluence peut être établie :

Théorème 1.12 *Le $\lambda\sigma'$-calcul est confluent sur ΛX.*

Démonstration
Le chapitre 3 sera consacré à la démonstration de ce résultat (cf.théorème 3.3). Nous utiliserons essentiellement la méthode d'interprétation.
□

Théorème 1.13 *Le $\lambda\sigma'$-calcul extensionnel est confluent sur ΛX .*

Démonstration
Nous montrons ce résultat dans le chapitre 4 (cf. théorème 4.4) en utilisant encore la méthode d'interprétation.
□

1.8 Le $\lambda\sigma_{\Uparrow}$-calcul

La seule contribution de notre part au calcul introduit dans cette section est le résultat de complétude de la section 5.4. Cependant nous voulons l'inclure dans ce chapitre pour donner une vue d'ensemble de l'état de l'art des $\lambda\sigma$-calculs. En plus, le $\lambda\tau$-calcul présenté dans le chapitre 7 est très proche du $\lambda\sigma_{\Uparrow}$-calcul.

Dans [HL89], T. Hardin et J.-J. Lévy ont montré qu'en introduisant un nouvel opérateur unaire appelé *lift*, et noté $\Uparrow$, on obtient une version du $\lambda\sigma$-calcul confluente sur l'ensemble des termes ouverts.

L'idée sous-jacente dans l'introduction de ce nouvel opérateur est de simplifier la règle *(Abs)* pour faire disparaître une paire critique. Dans la section précédente nous avons vu qu'il faut ajouter à $\lambda\sigma$ l'ensemble de règles $SPID$ pour obtenir un système localement confluent. La règle *(SCons)*, n'étant pas linéaire à gauche, n'est pas bonne du point de vue de la confluence. Cette règle apparaît pour compléter une paire critique engendrée par *(VarShift)*, et celle-ci à son tour provient d'une paire critique entre *(Abs)* et *(IdR)*.

On peut éviter cette paire critique en remplaçant le sous-terme $\mathtt{1}\cdot(s\circ\uparrow)$ du membre droit de *(Abs)* par $\Uparrow(s)$. Il faut bien sûr ajouter un paquet de règles pour gérer la confluence avec ce nouvel opérateur.

Les indices de de Bruijn dans le $\lambda\sigma_{\Uparrow}$-calcul sont des vrais numéros de de Bruijn, i.e. on travaille avec $\mathtt{n}$ directement au lieu de le coder par $\mathtt{1}[\uparrow^{n-1}]$. De toutes façons, le fait de ne pas coder les indices de de Bruijn n'est pas essentiel, tandis que l'introduction du *lift* est fondamentale.

Voici la définition du $\lambda\sigma_{\Uparrow}$-calcul :

Définition 1.39 *La syntaxe du* $\lambda\sigma_{\Uparrow}$-calcul *est donnée par :*

Termes	$a ::= X \mid \mathtt{n} \mid ab \mid \lambda a \mid a[s]$
Substitutions	$s ::= \mathbf{x} \mid id \mid \uparrow \mid a\cdot s \mid s\circ t \mid \Uparrow(s)$

L'ensemble, noté $\lambda\sigma_{\Uparrow}$, des règles du $\lambda\sigma_{\Uparrow}$-calcul est donné dans la figure 1.7.
Nous appelons $\sigma_{\Uparrow}$ l'ensemble de règles $\lambda\sigma_{\Uparrow} - \{(Beta)\}$.

Lemme 1.8 *Le $\sigma_{\Uparrow}$-calcul est localement confluent.*

Démonstration
La confluence locale est obtenue à l'aide du logiciel KB.
□

Lemme 1.9 *Le $\sigma_{\Uparrow}$-calcul est fortement normalisable.*

Démonstration
Un ordre de type polynomial est présenté dans [HL89] . Nous soulignons la simplicité de cette démonstration par rapport à celle de la terminaison forte de σ .
□

Proposition 1.3 *Le $\sigma_{\Uparrow}$-calcul est confluent.*

Démonstration
C'est une conséquence immédiate des deux lemmes précédents et du lemme 1.3.
□

Théorème 1.14 *Le $\lambda\sigma_{\Uparrow}$-calcul est confluent.*

Démonstration
La preuve, donnée dans [HL89], utilise la méthode d'interprétation (cf. lemme 1.4) par $\sigma_{\Uparrow}$-normalisation.
□

Nous synthétisons les résultats de confluence mentionnés dans ces dernières sections dans la figure 1.4.

1.9 Les calculs typés

Une version simplement typée du $\lambda\sigma$-calcul a été introduite dans [ACCL90]. Le système σ permet de relier le typage dans le $\lambda\sigma$-calcul au typage dans le λ-calcul. Nous commençons donc par définir formellement le λ-calcul simplement typé, dans la notation de de Bruijn.

Les environnements sont utilisés dans les règles d'inférence de types comme d'habitude, i.e. pour enregistrer les types des variables libres. Dans le formalisme choisi, les environnements sont donc indexés par des indices de de Bruijn. Par exemple, l'environnement $A_1, A_2, \cdots, A_n, nil$ associe à l'indice $\mathtt{i}$ le type A_i.

Définition 1.40 *La syntaxe du λ-calcul simplement typé est donnée par:*

$$\begin{array}{lrcl} \textbf{Types} & A & ::= & K \mid A \to B \\ \textbf{Environnements} & E & ::= & nil \mid A, E \\ \textbf{Termes} & a & ::= & \mathtt{n} \mid ab \mid \lambda A.a \end{array}$$

Les règles de typage sont données par la théorie **L1** *suivante:*

$$(\mathbf{L1} - var) \qquad A, E \vdash \mathtt{1} : A$$

$$(\mathbf{L1} - varn) \qquad \frac{E \vdash \mathtt{n} : B}{A, E \vdash \mathtt{n+1} : B}$$

$$(\mathbf{L1} - lambda) \qquad \frac{A, E \vdash b : B}{E \vdash \lambda A.b : A \to B}$$

$$(\mathbf{L1} - app) \qquad \frac{E \vdash b : A \to B \quad E \vdash a : A}{E \vdash ba : B}$$

Dans le $\lambda\sigma$-calcul, il faut typer les termes proprement dits et les substitutions. Les environnements, étant une liste de types, ils joueront le rôle de "types" pour les substitutions, qui sont interprétées comme une liste de termes. Nous utilisons la notation $s \triangleright E$ pour dire que la substitution s a pour environnement E.

Définition 1.41 *La syntaxe du $\lambda\sigma$-calcul simplement typé est donnée par:*

$$\begin{array}{lrcl} \textbf{Types} & A & ::= & K \mid A \to B \\ \textbf{Environnements} & E & ::= & nil \mid A, E \\ \textbf{Termes} & a & ::= & \mathtt{1} \mid ab \mid \lambda A.a \mid a[s] \\ \textbf{Substitutions} & s & ::= & id \mid \uparrow \mid a : A \cdot s \mid s \circ t \end{array}$$

Les règles de typage sont données par la théorie **S1** *suivante :*

$$(\mathbf{S1} - var) \qquad A, E \vdash \mathtt{1} : A$$

$$(\mathbf{S1} - lambda) \qquad \frac{A, E \vdash b : B}{E \vdash \lambda A.b : A \to B}$$

$$(\mathbf{S1} - app) \qquad \frac{E \vdash b : A \to B \quad E \vdash a : A}{E \vdash ba : B}$$

$$(\mathbf{S1} - clos) \qquad \frac{E \vdash s \triangleright E' \quad E' \vdash a : A}{E \vdash a[s] : A}$$

$$(\mathbf{S1} - id) \qquad E \vdash id \triangleright E$$

$$(\mathbf{S1} - shift) \qquad A, E \vdash \uparrow \triangleright E$$

$$(\mathbf{S1} - cons) \qquad \frac{E \vdash a : A \quad E \vdash s \triangleright E'}{E \vdash a : A \cdot s \ \triangleright \ A, E'}$$

$$(\mathbf{S1} - comp) \qquad \frac{E \vdash s'' \triangleright E'' \quad E'' \vdash s' \triangleright E'}{E \vdash s' \circ s'' \ \triangleright \ E'}$$

Dans [ACCL90], la correction (soundness) du $\lambda\sigma$-calcul simplement typé a été établie. Avant d'énoncer ce résultat, dans lequel la notion de σ-forme normale intervient, il faut préciser la signification de σ-réduction dans la version typée. Afin de réduire les termes typés nous utilisons une version typée de l'ensemble σ de règles. Voici celles qui sont affectées :

$$\begin{array}{ll} (VarCons) & \mathtt{1}[a : A \cdot s] \to a \\ (Abs) & (\lambda A.a)[s] \to \lambda A.(a[\mathtt{1} : A \cdot (s \circ \uparrow)]) \\ (ShiftCons) & \uparrow \circ (a : A \cdot s) \to s \\ (Map) & (a : A \cdot s) \circ t \to a[t] : A \cdot (s \circ t) \end{array}$$

Remarquons que la version typée de σ a les mêmes propriétés (noethérianité et confluence) que la version non typée.

Nous énonçons maintenant le résultat de correction que nous avons annoncé plus haut :

Théorème 1.15 (Soundness) *Si* $E \vdash_{\mathbf{S1}} a : A$ *, alors* $E \vdash_{\mathbf{L1}} \sigma(a) : A$ *.*

On peut trouver sans difficulté un contrexemple montrant que la réciproque (complétude) n'est pas vrai. Dans le chapitre 5, nous établirons un cadre adéquat pour un résultat de complétude.

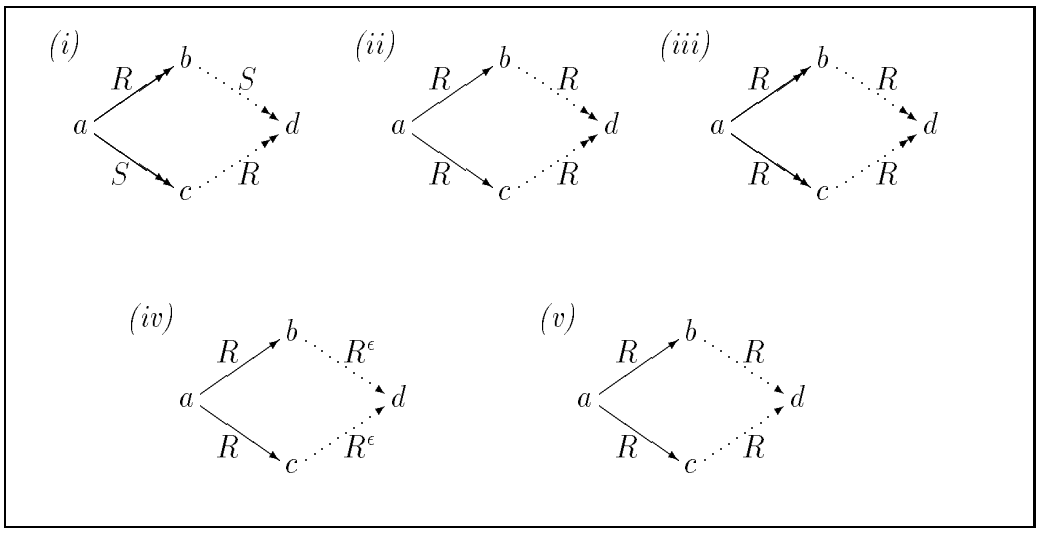


FIG. 1.2 - Propriétés des réductions

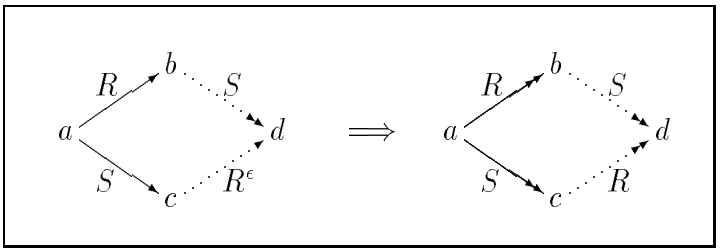


FIG. 1.3 - Diagramme du "Commutativity Lemma"

	sur l'ensemble des termes		
	clos	semi-clos	ouverts
$\lambda\sigma$	C	¬ LC	¬ LC
$\lambda\sigma'$	C	C	LC , ¬ C
$\lambda\sigma_{\Uparrow}$	C	C	C

C : confluent
¬ C : non confluent
LC : loc. conf.
¬ LC : non loc. conf.

FIG. 1.4 - Schéma de confluence des $\lambda\sigma$-calculs

(Beta)	$(\lambda a)\, b$	$\longrightarrow$	$a\,[b \cdot id]$
(VarId)	$\mathtt{1}\,[id]$	$\longrightarrow$	$\mathtt{1}$
(VarCons)	$\mathtt{1}\,[a \cdot s]$	$\longrightarrow$	a
(App)	$(a\, b)[s]$	$\longrightarrow$	$(a\,[s])\,(b\,[s])$
(Abs)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a\,[\mathtt{1} \cdot (s \circ \uparrow)])$
(Clos)	$(a\,[s])[t]$	$\longrightarrow$	$a\,[s \circ t]$
(IdL)	$id \circ s$	$\longrightarrow$	s
(ShiftId)	$\uparrow \circ id$	$\longrightarrow$	$\uparrow$
(ShiftCons)	$\uparrow \circ (a \cdot s)$	$\longrightarrow$	s
(Map)	$(a \cdot s) \circ t$	$\longrightarrow$	$a\,[t] \cdot (s \circ t)$
(Ass)	$(s_1 \circ s_2) \circ s_3$	$\longrightarrow$	$s_1 \circ (s_2 \circ s_3)$

FIG. 1.5 - Le système de réécrirure $\lambda\sigma$

	(Beta)	$(\lambda a)\, b$	$\longrightarrow$	$a\,[b \cdot id]$
$\star$	*(Id)*	$a\,[id]$	$\longrightarrow$	a
	(VarCons)	$\mathtt{1}\,[a \cdot s]$	$\longrightarrow$	a
	(App)	$(a\, b)[s]$	$\longrightarrow$	$(a\,[s])\,(b\,[s])$
	(Abs)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a\,[\mathtt{1} \cdot (s \circ \uparrow)])$
	(Clos)	$(a\,[s])[t]$	$\longrightarrow$	$a\,[s \circ t]$
	(IdL)	$id \circ s$	$\longrightarrow$	s
$\star$	*(IdR)*	$s \circ id$	$\longrightarrow$	s
	(ShiftCons)	$\uparrow \circ (a \cdot s)$	$\longrightarrow$	s
	(Map)	$(a \cdot s) \circ t$	$\longrightarrow$	$a\,[t] \cdot (s \circ t)$
	(Ass)	$(s_1 \circ s_2) \circ s_3$	$\longrightarrow$	$s_1 \circ (s_2 \circ s_3)$
$\star$	*(VarShift)*	$\mathtt{1} \cdot \uparrow$	$\longrightarrow$	id
$\star$	*(SCons)*	$\mathtt{1}\,[s] \cdot (\uparrow \circ s)$	$\longrightarrow$	s

Les règles précédées de $\star$ sont les $SPID$-règles,
donc les règles ajoutées à $\lambda\sigma$ pour obtenir $\lambda\sigma'$.

FIG. 1.6 - Le système de réécrirure $\lambda\sigma'$

(Beta)	$(\lambda a)b$	$\longrightarrow$	$a\,[b \cdot id]$
(App)	$(a\,b)[s]$	$\longrightarrow$	$a\,[s]\,b\,[s]$
(Lambda)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a\,[\Uparrow(s)])$
(Clos)	$(a\,[s])[t]$	$\longrightarrow$	$a\,[s \circ t]$
(Varshift1)	$\mathtt{n}\,[\uparrow]$	$\longrightarrow$	$\mathtt{n}+\mathtt{1}$
(Varshift2)	$\mathtt{n}\,[\uparrow \circ s]$	$\longrightarrow$	$\mathtt{n}+\mathtt{1}\,[s]$
(FVarCons)	$\mathtt{1}\,[a \cdot s]$	$\longrightarrow$	a
(FVarLift1)	$\mathtt{1}\,[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{1}$
(FVarLift2)	$\mathtt{1}\,[\Uparrow(s) \circ t]$	$\longrightarrow$	$\mathtt{1}\,[t]$
(RVarCons)	$\mathtt{n}+\mathtt{1}\,[a \cdot s]$	$\longrightarrow$	$\mathtt{n}\,[s]$
(RVarLift1)	$\mathtt{n}+\mathtt{1}\,[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{n}\,[s \circ \uparrow]$
(RVarLift2)	$\mathtt{n}+\mathtt{1}\,[\Uparrow(s) \circ t]$	$\longrightarrow$	$\mathtt{n}\,[s \circ (\uparrow \circ t)]$
(AssEnv)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$
(MapEnv)	$(a \cdot s) \circ t$	$\longrightarrow$	$a\,[t] \cdot (s \circ t)$
(ShiftCons)	$\uparrow \circ (a \cdot s)$	$\longrightarrow$	s
(ShiftLift1)	$\uparrow \circ \Uparrow(s)$	$\longrightarrow$	$s \circ \uparrow$
(ShiftLift2)	$\uparrow \circ (\Uparrow(s) \circ t)$	$\longrightarrow$	$s \circ (\uparrow \circ t)$
(Lift1)	$\Uparrow(s) \circ \Uparrow(t)$	$\longrightarrow$	$\Uparrow(s \circ t)$
(Lift2)	$\Uparrow(s) \circ (\Uparrow(t) \circ u)$	$\longrightarrow$	$\Uparrow(s \circ t) \circ u$
(LiftEnv)	$\Uparrow(s) \circ (a \cdot t)$	$\longrightarrow$	$a \cdot (s \circ t)$
(IdL)	$id \circ s$	$\longrightarrow$	s
(IdR)	$s \circ id$	$\longrightarrow$	s
(LiftId)	$\Uparrow(id)$	$\longrightarrow$	id
(Id)	$a\,[id]$	$\longrightarrow$	a

FIG. 1.7 - Le système de réécriture $\lambda\sigma_{\Uparrow}$

Chapitre 2

Normalisation forte du σ-calcul

Jusqu'à présent[1] une seule démonstration de la noethérianité du σ-calcul était connue (cf. [HL86]). Elle est fondée sur la terminaison de SUBST[2]. Il est facile de prouver la terminaison de SUBST privé de la règle *(DΛ)*. Cette règle est essentielle : elle sert à propager la substitution sous les abstractions et assure la gestion des variables du terme substitué afin d'éviter toute capture. La preuve donnée ici est nouvelle et complètement différente de celle proposée par Hardin et Laville [HL86].

Après la présentation d'un système plus économique en syntaxe et dont la noethérianité nous permettra détablir celle de σ (section 2.1), nous montrons que les termes sans λ (w-termes) sont fortement normalisables (section 2.2). L'extension du résultat aux termes contenant λ nécessite un lemme, dont la démonstration s'avère difficile et demande l'introduction d'une nouvelle notion, celle d'*inflation* (section 2.3). Le lemme est obtenu grâce à un théorème de préservation : les résultats des certaines inflations des termes fortement normalisables sont fortement normalisables. La section 2.4 sera consacrée à un plan de démonstration de ce théorème. Dans les sections 2.5 et 2.6 nous montrons des résultats préliminaires et dans la section 2.7 nous donnons la démonstration complète du théorème de préservation.

Le $\lambda\sigma$-calcul (σ + *(Beta)*) n'est pas localement confluent, mais il est confluent sur les termes clos, donc sur les termes du λ-calcul. Pour obtenir la confluence locale, il faut ajouter quatre règles à σ. Le système ainsi obtenu est appelé σ' (cf. définition 1.38). La section 2.8 sera consacrée à la preuve de terminaison de σ'. Or, σ' et SUBST sont deux systèmes très voisins [3]. Nous obtenons ainsi une deuxième démonstration de la terminaison de SUBST.

---

[1] Récemment une troisième preuve de la noethérianité de σ a été fournie par H. Zantema [Zan92]. Voir la note de la page x.

[2] Nous donnons les règles de SUBST ainsi que l'idée générale de cette preuve de terminaison dans l'annexe C.

[3] Les systèmes σ' et SUBST contiennent tous deux une règle dite de Surjective Pairing, non linéaire à gauche. Le λ-calcul augmenté de cette règle n'est pas confluent. Il en est de même de *(Beta)* + SUBST [Har87] et de *(Beta)* + σ' [CHL91].

(VarId)	$\mathtt{1} \circ id$	$\longrightarrow$	$\mathtt{1}$
(VarCons)	$\mathtt{1} \circ (s \cdot t)$	$\longrightarrow$	s
(Abs)	$(\lambda s) \circ t$	$\longrightarrow$	$\lambda(s \circ (\mathtt{1} \cdot (t \circ \uparrow)))$
(IdL)	$id \circ s$	$\longrightarrow$	s
(ShiftId)	$\uparrow \circ id$	$\longrightarrow$	$\uparrow$
(ShiftCons)	$\uparrow \circ (s \cdot t)$	$\longrightarrow$	t
(Map)	$(s \cdot t) \circ u$	$\longrightarrow$	$(s \circ u) \cdot (t \circ u)$
(Ass)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$

FIG. 2.1 - Le système de réécrirure σ_0

2.1 Le σ_0-calcul

Nous rappelons que le $\lambda\sigma$-calcul a été introduit dans le chapitre 1 (cf. définition 1.34). La règle *(DΛ)* de SUBST, dont nous avons parlé plus haut, correspond à la règle *(Abs)* de notre calcul.

Etant donné que les opérateurs de substitution et composition ont un comportement semblable en ce qui concerne la démonstration de la normalisation forte, nous allons confondre les deux opérateurs. Ce faisant, l'opérateur application devient redondant et peut être confondu avec l'opérateur cons. Donc, nous proposons un calcul plus économique en syntaxe que nous appelons σ_0-calcul. Voici sa définition :

Définition 2.1 *La syntaxe du* σ_0-calcul *est donnée par :*

$$\textbf{Termes} \quad s ::= \mathtt{1} \mid id \mid \uparrow \mid \lambda s \mid s \circ t \mid s \cdot t$$

L'ensemble, noté σ_0*, des règles de ce calcul est donné dans la figure 2.1.*

Dans la suite nous utiliserons à plusieures reprises des fonctions d'interprétation d'un calcul dans un autre calcul. Nous appelons *strictes* les interprétations qui amènent chaque étape de réduction du premier calcul sur une étape de réduction du deuxième. La noethérianité de ce dernier calcul et l'existence d'une interprétation stricte garantissent la noethérianité du calcul de départ.

On remarque la ressemblance du σ_0-calcul avec le sous-calcul $\mathcal{E}$ de SUBST introduit dans [Har89] et donné ici en annexe C. En effet, il y a une interprétation stricte de σ_0 dans $\mathcal{E}$ (l'interprétation inverse de celle donnée dans la proposition 2.6).

Notation 2.1 *On note* $\Lambda\sigma_0$ *l'ensemble des termes du* σ_0*-calcul.*

Définition 2.2 *Soit* $F : \Lambda\sigma \to \Lambda\sigma_0$ *l'interprétation suivante :*

$$\begin{array}{ll} F(\mathtt{1}) = \mathtt{1} & F(id) = id \\ F(ab) = F(a) \cdot F(b) & F(\uparrow) = \uparrow \\ F(\lambda a) = \lambda F(a) & F(a \cdot s) = F(a) \cdot F(s) \\ F(a[s]) = F(a) \circ F(s) & F(s \circ t) = F(s) \circ F(t) \end{array}$$

Puisque cette interprétation est stricte, on déduit la

Proposition 2.1 *Si* σ_0 *est noethérien, alors* σ *est noethérien.*

2.2 Les w-termes

Dans cette section nous présentons les w-termes et démontrons quelques propriétés, en particulier la normalisation forte. Il est facile de définir un R.P.O. montrant la terminaison de $\sigma_0 - \{(Abs)\}$. Mais nous n'avons pas réussi, malgré plusieurs tentatives, à étendre ce R.P.O. à σ_0. Nous allons tenter une preuve de terminaison par récurrence structurelle qui ne sera pas menée jusqu'au bout dans cette section, car la règle *(Abs)* posera des problèmes. Nous définissons les w-termes comme des termes sans λ's et montrons que tout w-terme est fortement normalisable. Enfin, nous formulons le résultat qui sera démontré dans la suite du chapitre et qui permettra d'étendre cette preuve en une preuve de terminaison de σ_0.

Notation 2.2 SN_0 *est le sous-ensemble des termes fortement normalisables de* $\Lambda\sigma_0$, *i.e. ceux à partir desquels toute réduction termine.*

Comme aucune règle du σ_0-calcul n'a ni "λ" ni "$\cdot$" comme symbole de tête, on montre sans difficulté le

Lemme 2.1 *Pour tous* $s, t, u \in \Lambda\sigma_0$ *on a les équivalences suivantes :*

1. $\lambda s \in SN_0$ *ssi* $s \in SN_0$.
2. $s \cdot t \in SN_0$ *ssi* $s \in SN_0$ *et* $t \in SN_0$.

Donc, pour prouver que tous les termes sont fortement normalisables, il reste à démontrer :

$$\text{Si } s, t \in SN_0, \text{ alors } s \circ t \in SN_0 .$$

On est conduit à discuter suivant la règle applicable au sommet de $s \circ t$. Mais, comme nous ne savons pas conclure directement lorsqu'il s'agit de *(Abs)*, nous allons d'abord faire la preuve pour les termes ne contenant pas de λ's.

Définition 2.3 *On définit l'ensemble des* w-termes, *noté* $W\sigma$, *par récurrence :*

$$\mathbf{w-termes} \quad w ::= \mathtt{1} \mid id \mid \uparrow \mid w_1 \circ w_2 \mid w_1 \cdot w_2$$

Comme aucune règle de σ_0 n'introduit le symbole λ, on constate :

Lemme 2.2 *Les w-termes sont stables par σ_0-réduction.*

Proposition 2.2 *Soient $t, s \in W\sigma$. Si $t, s \in SN_0$, alors $t \circ s \in SN_0$.*

Démonstration
Récurrence sur $(\text{prof}(t), \lg(t), \text{prof}(s), \lg(s))$ où $\text{prof}(x)$ est la profondeur de x, i.e. la longueur de la réduction la plus longue à partir de x et $\lg(x)$ est la longueur de x, i.e. la quantité des symboles dans x ($\lg(\mathtt{1}) = 1$, $\lg(a \cdot b) = \lg(a) + \lg(b) + 1$, $\lg(\lambda a) = \lg(a) + 1$, etc.).
Si $t \circ s$ est en forme normale, la proposition est évidente. Considérons donc une réduction à partir de $t \circ s$. Il y a trois possibilités :

i) La première étape réduit dans t . On a donc que $t \circ s \to t' \circ s$, et on peut appliquer l'hypothèse de récurrence (H.R. dans la suite) car $\text{prof}(t') < \text{prof}(t)$.

ii) La première ètape réduit dans s . On a donc que $t \circ s \to t \circ s'$, et on peut appliquer l'H.R. car $(\text{prof}(t), \lg(t), \text{prof}(s')) < (\text{prof}(t), \lg(t), \text{prof}(s))$.

iii) La première étape réduit à la racine. Il y a plusieurs sous-cas selon la règle utilisée :

(VarId) **ou** *(ShiftId)* **:** Le réduit est $\mathtt{1}$ ou $\uparrow$ qui sont en forme normale.

(IdL) **:** Par hypothèse le réduit est fortement normalisable.

(VarCons) **ou** *(ShiftCons)* **:** Alors $s = s_1 \cdot s_2$ et le réduit étant s_1 ou s_2, respectivement, est fortement normalisable, car $s \in SN_0$.

(Map) **:** Alors $(t_1 \cdot t_2) \circ s \to (t_1 \circ s) \cdot (t_2 \circ s)$.
Comme $\text{prof}(t_1 \cdot t_2) \geq \text{prof}(t_1)$ et $\lg(t_1 \cdot t_2) > \lg(t_1)$, par H.R. on peut conclure que $t_1 \circ s \in SN_0$ et de manière analogue on a $t_2 \circ s \in SN_0$, d'où, par la remarque 2.1, $(t_1 \circ s) \cdot (t_2 \circ s) \in SN_0$.

(Ass) **:** Alors $(t_1 \circ t_2) \circ s \to t_1 \circ (t_2 \circ s)$.
Or, comme $\text{prof}(t_1 \circ t_2) \geq \text{prof}(t_2)$ et $\lg(t_1 \circ t_2) > \lg(t_2)$, l'H.R. garantit que $t_2 \circ s \in SN_0$. D'autre part $\text{prof}(t_1 \circ t_2) \geq \text{prof}(t_1)$ et $\lg(t_1 \circ t_2) > \lg(t_1)$. On peut donc appliquer l'H.R. à t_1 et $t_2 \circ s$ pour conclure que $t_1 \circ (t_2 \circ s) \in SN_0$.

□

Pour étendre cette preuve à $\Lambda\sigma$, il faudrait traiter le cas de la règle *(Abs)* :

$$(\lambda t_1) \circ s \ \to \ \lambda(t_1 \circ (\mathtt{1} \cdot (s \circ \uparrow)))$$

Nous ne savons pas, avec l'ordre ci-dessus, montrer que $s \circ \uparrow \in SN_0$. La démonstration de cette propriété se révèle non évidente.
Le seul résultat obtenu jusque lors est donc :

Proposition 2.3 *Les w-termes sont fortement normalisables.*

Remarque 2.1 *Pour montrer la terminaison de $\Lambda\sigma$, il reste donc à prouver :*

$$\star \quad \textit{Si } s \in SN_0, \textit{ alors } \quad s \circ \uparrow \in SN_0 .$$

En effet, l'appartenance à $W\sigma$ ne joue aucun rôle dans les preuves précédentes, si ce n'est pour éviter *(Abs)*.

La démonstration d'un résultat plus général que l'affirmation $\star$ sera donnée dans les sections 2.3 - 2.7.

2.3 Contextes et Inflation

Nous présentons dans cette section la notion de contexte à plusieurs trous. Après avoir introduit formellement la notion d'inflation, nous donnons, en les motivant les définitions de bon et très bon contexte, et montrons quelques propriétés.

Donnons d'abord une idée intuititive de ces notions. Nous voulons étudier le comportement de $s \circ \uparrow$ par rapport à celui de s. Analysons donc quelques réduits de $s \circ \uparrow$ selon la forme de s.

1. Soit $s = \lambda t$. Examinons l'étape de réduction suivante :

$$(\lambda t) \circ \uparrow \rightarrow \lambda(t \circ (\mathtt{1} \cdot (\uparrow \circ \uparrow))).$$

   Nous avons donc besoin d'un résultat plus général que $\star$, car nous devons traiter $t \circ (\mathtt{1} \cdot (\uparrow \circ \uparrow))$ au lieu de $t \circ \uparrow$. Essayons la généralisation suivante :

$$\star\star \quad \text{Si } s \in SN_0 \text{, alors } \quad s \circ w \in SN_0 .$$

   où w est un w-terme quelconque. $\star\star$ suffit ici, car on sait par la partie 1 du lemme 1 que $\lambda(t \circ (\mathtt{1} \cdot (\uparrow \circ \uparrow))) \in SN_0$ dès que $t \circ (\mathtt{1} \cdot (\uparrow \circ \uparrow)) \in SN_0$. Mais ce n'est pas le cas pour d'autres opérateurs. Nous continuons notre analyse avec les réduits de $s \circ w$.

2. Soit $s = a \circ b$. Examinons l'étape de réduction suivante :

$$(a \circ b) \circ w \ \rightarrow \ a \circ (b \circ w).$$

Même en établissant $b \circ w \in SN_0$ grâce à une hypothèse de récurrence, nous n'avons pas de moyen d'en déduire $a \circ (b \circ w) \in SN_0$. Il nous faut une généralisation plus radicale de $\star\star$, dont nous donnons maintenant l'idée. Soit t un terme et u un sous-terme de t. On peut noter $t = C[u]$, où $C[\,]$ est le contexte obtenu en remplaçant le sous-terme u par un trou (voir définitions 2.4 et 2.5 plus loin). Apppelons *augmenté* de t un terme de la forme $C[u \circ w]$, où w est un w-terme. Un tel terme est donc obtenu en remplaçant le sous-terme u par $u \circ w$ dans t. Par exemple, $(a \circ b) \circ w$ est un augmenté de $s = a \circ b$ en prenant le contexte $C[\,] = [\,]$. Le terme $a \circ (b \circ w) = D[b \circ w]$ est aussi un augmenté de $s = a \circ b = D[b]$, où $D[\,] = a \circ [\,]$.

La généralisation de $\star\star$ que nous proposons est, essentiellement :

$\star\star\star$ Si $s \in SN_0$, alors tout augmenté de s appartient à SN_0.

En fait, $\star\star\star$ est la version informelle de ce que nous appellerons le théorème de préservation.

Nous revenons sur cette discussion dans la section suivante. Donnons un dernier exemple :

3. Soit $s = a \cdot b$. Examinons l'étape de réduction suivante :

$$(a \cdot b) \circ w \;\rightarrow\; (a \circ w) \cdot (b \circ w)$$

Ce dernier exemple nous suggère l'introduction de contextes à plusieurs trous, à cause de la duplication de w.

Un contexte pour s est donc un préfixe de s, à partir duquel on peut reconstruire s en remplissant les trous par les sous-termes correspondants de s. Il faudra bien choisir les contextes de façon à obtenir de bonnes propriétés de récurrence sur les sous-termes.

Nous définissons maintenant les notions que nous venons d'introduire informellement.

Définition 2.4 *On définit les* contextes à plusieurs trous *par récurrence :*

$$C ::= \Box_n \mid \mathtt{1} \mid id \mid \uparrow \mid \lambda C \mid C \cdot D \mid C \circ D$$

où $n \geq 1$ *et* $\Box_n$ *dénote un trou.*

Notation 2.3 *Soit* γ *une occurrence du contexte* C. *On notera* C/γ *le sous-contexte de* C *à l'occurrence* γ *et* $C\{\gamma \leftarrow s\}$ *le contexte obtenu en remplaçant dans* C *le sous-contexte* C/γ *par le terme* s.
Dans le cas particulier où C/γ *est un trou, on écrira* $C[s]_\gamma$ *au lieu de* $C\{\gamma \leftarrow s\}$.
La notation $\Box_k \in C$ *exprimera qu'il existe une occurrence* γ *dans l'ensemble d'occurrences de* C *tel que* $C/\gamma = \Box_k$.

Exemple 2.1 *Soient* $C = ((\lambda\Box_4)\cdot(\Box_2\circ\uparrow))\cdot(1\cdot\Box_4)$ *un contexte, et les termes* $s = (\lambda(1\circ\uparrow))\cdot(1\cdot id)$ *et* $t = \uparrow\circ\lambda 1$ *. Alors*

- $C/1 = (\lambda\Box_4)\cdot(\Box_2\circ\uparrow)$
- $C/12 = \Box_2\circ\uparrow$
- $C\{12\leftarrow t\} = ((\lambda\Box_4)\cdot(\uparrow\circ\lambda 1))\cdot(1\cdot\Box_4)$
- $C[t]_{121} = ((\lambda\Box_4)\cdot((\uparrow\circ\lambda 1)\circ\uparrow))\cdot(1\cdot\Box_4)$
- $C[s]_{11} = ((\lambda((\lambda(1\circ\uparrow))\cdot(1\cdot id)))\cdot(\Box_2\circ\uparrow))\cdot(1\cdot\Box_4)$

Définition 2.5 *Soient* C *un contexte,* $n_C = \max\{m : \Box_m \in C\}$, $n \geq n_C$ *et* $\mathbf{u} = (u_1,\cdots,u_n)$ *un n-uplet de termes. Alors*

$$C[\mathbf{u}] = C[u_1,\cdots,u_n]$$

est le terme obtenu en plaçant u_k *dans tous les trous* $\Box_k$ *de* C *pour* $1\leq k\leq n$ *.*

Chaque u_k remplit donc tous les trous $\Box_k$. Comme nous exigeons $n \geq n_C$, tous les trous du contexte seront remplis.

Notation 2.4 *Soient* C *un contexte et* $q \geq 0$ *. On notera* C_q *le contexte obtenu par décalage à partir de* C *en renommant les trous* $\Box_k$ *par* $\Box_{k+q}$*.*
Soient $\mathbf{u} = (u_1,\cdots,u_{n_1})$ *et* $\mathbf{v} = (v_1,\cdots,v_{n_2})$*. La juxtaposition de* $\mathbf{u}$ *et* $\mathbf{v}$ *sera notée* $\mathbf{u}@\mathbf{v} = (u_1,\cdots,u_{n_1},v_1,\cdots,v_{n_2})$ *.*

Remarque 2.2 $C[\mathbf{u}]$ *ne dépend que des composantes de* $\mathbf{u}$ *qui correspondent à des trous de* C *. Plus précisément, si* $\mathbf{u}$ *et* $\mathbf{v}$ *sont tels que* $u_k = v_k$ *pour tout* k *tel que* $\Box_k \in C$*, alors* $C[\mathbf{u}] = C[\mathbf{v}]$ *.*
Si $\mathbf{u} = \mathbf{s}@\mathbf{t}$ *et* $\lg(\mathbf{s}) = q$*, alors* $C_q[\mathbf{u}] = C[\mathbf{t}]$ *.*

Exemple 2.2 Si $C = (\Box_2\circ(1\cdot\Box_3))\cdot(\Box_2\cdot\Box_5)$, alors

- $C_3 = (\Box_5\circ(1\cdot\Box_6))\cdot(\Box_5\cdot\Box_8)$
- $C\,[\,1\,,\uparrow,id\,,\uparrow\circ\uparrow,\lambda 1\,,1\cdot id] = (\uparrow\circ(1\cdot id))\cdot(\uparrow\cdot(\lambda 1))$
- $C_1\,[\,\uparrow,id\,,\uparrow\cdot\uparrow,1\cdot id\,,1\,,1\circ 1] = ((\uparrow\cdot\uparrow)\circ(1\cdot(1\cdot id)))\cdot((\uparrow\cdot\uparrow)\cdot(1\circ 1))$

Définition 2.6 *Un* contexte C relatif à s *est un contexte tel que* $s = C[\mathbf{u}]$ *pour un certain* $\mathbf{u}$ *. Si* $\Box_k \in C$*, on dira que le trou* $\Box_k$ *est un* w-trou *si* u_k *est un w-terme. On appellera* λ-trous *les trous qui ne sont pas des w-trous.*

Définition 2.7 *Une* inflation *de s est une paire $(C, \mathbf{w})$ où C est un contexte et $\mathbf{w}$ un n-uplet de w-termes tel qu'il existe un n-uplet de termes $\mathbf{u}$ qui satisfait $s = C[\mathbf{u}]$.*
Le résultat de l'inflation *est $s' = C[\mathbf{u}']$ où $\mathbf{u}'$ est donné par*

- $u'_k = w_k$ *si u_k est un w-terme*
- $u'_k = u_k \circ w_k$ *autrement.*

On peut considérer que $'$ est un opérateur qui prend $\mathbf{u}$ et $\mathbf{w}$ pour rendre $\mathbf{u}'$. Il y aura donc un abus de notation, puisque la dépendence de $\mathbf{w}$ ne sera pas explicitée dans $\mathbf{u}'$. Mais le contexte déterminera toujours $\mathbf{w}$.

Convention La phrase "$(C, \mathbf{w})$ est une inflation de $s = C[\mathbf{u}]$" abrégera la phrase "$(C, \mathbf{w})$ est une inflation de s et $\mathbf{u}$ est un $\lg(\mathbf{w})$-uplet tel que $s = C[\mathbf{u}]$".

Remarque 2.3 *Si $(C, \mathbf{w})$ est une inflation de $s = C[\mathbf{u}] = C[\mathbf{v}]$, alors $C[\mathbf{u}'] = C[\mathbf{v}']$.*

Nous allons maintenant imposer quelques restrictions aux contextes avec lesquels nous allons travailler. Examinons un exemple pour comprendre la nécessité de ces restrictions. Cet exemple sera traité de manière plus générale dans l'étude de la règle *(Ass)* au cas III) du théorème de préservation.

Soient $C = \Box_1 \circ \Box_2$ et $s = C[t, \lambda u]$ où t n'est pas un w-terme. Considérons l'inflation $(C, (w_1, w_2))$ de s de résultat $(t \circ w_1) \circ ((\lambda u) \circ w_2)$. Une réduction possible est

$$(t \circ w_1) \circ ((\lambda u) \circ w_2) \rightarrow t \circ (w_1 \circ ((\lambda u) \circ w_2)).$$

Nous voulons interdire cette situation, car nous ne pouvons pas traiter le réduit comme le résultat d'une inflation de s, puisque $w_1 \circ ((\lambda u) \circ w_2)$ n'est pas un w-terme.

La solution que nous proposons est d'interdire que $\Box_2$ soit un λ-trou. La définition suivante précise cette discusion.

Définition 2.8 *Un* bon contexte *relatif à s est un contexte C relatif à s vérifiant la condition suivante:*
Pour chaque occurrence δ d'une composition dans C, s'il existe un trou à une occurrence de la forme $\delta 1\alpha$, donc s'il existe un trou dans un descendant gauche de cette composition, alors à l' occurrence $\delta 2$ on ne peut trouver qu'un w-trou. Autrement dit, le fils droit de cette composition est obligatoirement un w-trou. La figure 2.3 illustre cette situation.

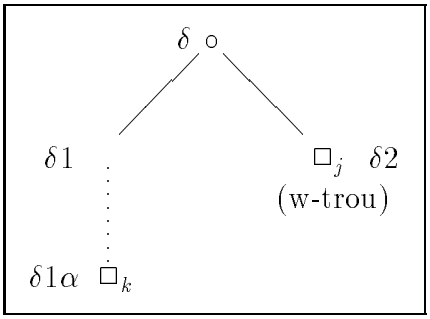


FIG. 2.2 - Un bon contexte

Nous imposons maintenant une autre restriction plus technique, qui sera utile pour restreindre le cas III) du théorème de préservation.

Définition 2.9 *Un* très bon contexte C *relatif à* s *est un bon contexte relatif à* s *tel que si* $s = C[\mathbf{u}]$ *et* u_k *est un w-terme, alors pour toute occurrence* γ *telle que* $C/\gamma = \Box_k$ *et pour tout* δ *préfixe propre de* γ, s/δ *n'est pas un w-terme. On abrégera cette situation en disant que si* u_k *est un w-terme, alors il est* w-maximal *dans* s.

Exemple 2.3 Soit $s = (\mathtt{1}\cdot\uparrow)\cdot((\uparrow\cdot id)\circ(\lambda\mathtt{1}))$, alors

- $C = (\mathtt{1}\cdot\Box_2)\cdot((\uparrow\cdot\Box_4)\circ\Box_1)$ est un contexte relatif à s, car

$$s = C\,[\,\lambda\mathtt{1}\,,\,\uparrow,\,\mathtt{1}\,,\,id\,]\,.$$

  C n'est pas bon, car en partant de $\Box_4$ vers la racine on arrive à une composition par la gauche et, même si on trouve comme fils droit de cette composition un trou ($\Box_1$), ce trou n'est pas un w-trou puisque $u_1 = \lambda\mathtt{1}$ n'est pas un w-terme.

- $D = (\mathtt{1}\cdot\Box_2)\cdot\Box_3$ est un contexte relatif à s, car

$$s = D\,[\,\mathtt{1}\,,\,\uparrow,\,(\uparrow\cdot id)\circ(\lambda\mathtt{1})\,]\,.$$

  D est bon car D ne contient pas de compositions.
  D n'est pas très bon, car $u_2 = \uparrow$ est un w-terme qui n'est pas w-maximal. En effet, $D/12 = \Box_2$ et $s/1 = \mathtt{1}\cdot\uparrow$, qui est un w-terme.

- $E = \Box_5\cdot\Box_6$ est un contexte relatif à s, car

$$s = E\,[\,\mathtt{1}\,,\mathtt{1}\,,\mathtt{1}\,,\mathtt{1}\,,\,\mathtt{1}\cdot\uparrow,\,(\uparrow\cdot id)\circ(\lambda\mathtt{1})\,]\,.$$

  On vérifie aisément que E est un très bon contexte pour s.

La remarque suivante sera utilisée à plusieurs reprises dans les propositions 2.4 et 2.5.

Remarque 2.4 (Stabilité par passage au sous-terme) *Si C est un (très) bon contexte relatif à s et si C' est un sous-contexte de C, alors C' est un (très) bon contexte relatif au sous-terme correspondant de s.*

Le lemme suivant sera utilisé dans le traitement de la règle *(Ass)* du cas III) du théorème de préservation.

Lemme 2.3 *Soit C un bon contexte relatif à s tel que C/γ soit un λ-trou et soit $s' = s\{\gamma \leftarrow t\}$, alors*

1. *Si $t \in \Lambda\sigma$, $C[t]_\gamma$ est un bon contexte relatif à s'.*

2. *Si $t \in \Lambda\sigma_0 - W\sigma$, i.e. si t contient au moins un λ, et si C est très bon relativement à s, alors $C[t]_\gamma$ est un très bon contexte relatif à s'.*

Démonstration

1. Immédiat, car les w-trous de C sont les mêmes que ceux de $C[t]_\gamma$.

2. Soit $\mathbf{u}$ tel que $s' = (C[t]_\gamma)[\mathbf{u}]$, et supposons que u_k soit un w-terme et ne soit pas w-maximal dans s', on a donc un w-terme w à une occurrence δ préfixe de celle de u_k. Alors δ n'est pas préfixe de γ car t n'est pas un w-terme, et cela entraîne que w est un sous-terme de s, ce qui contredit le fait que C est un très bon contexte relatif à s.

□

Définition 2.10 *Une* bonne (très bonne) inflation *de s est définie comme une inflation de s dont le contexte est bon (très bon) relativement à s.*

Exemple 2.4 *Soit $s \in \Lambda\sigma_0 - W\sigma$, i.e. un terme contenant au moins un λ, alors $s \circ \uparrow$ est le résultat d'une très bonne inflation de s.*

Démonstration
Prendre l'inflation $(\Box_1, \uparrow)$ qui est très bonne, $s \circ \uparrow$ est le résultat de cette inflation.
□

2.4 Plan de démonstration

Nous avons déjà les éléments nécessaires pour donner l'énoncé précis du théorème de préservation et faire un plan de démonstration.

Théorème de préservation
Soient $s \in SN_0$ *et* s' *le résultat d'une très bonne inflation* $(C, \mathbf{w})$ *de* $s = C[\mathbf{u}]$. *Alors* $s' \in SN_0$.

Nous devons étudier les réduits possibles de s'. Soit t' tel que

$$s' = C[\mathbf{u}'] \rightarrow_{\sigma_0} t'.$$

Il suffira de montrer l'existence d'un ordinal associé à s, C et $\mathbf{w}$, appelons-le $\theta_{s,s'}$ pour éclaircir la discussion informelle suivante, et d'un terme t tel que t' soit le résultat d'une très bonne inflation de t, de sorte que $\theta_{t,t'} < \theta_{s,s'}$.

Examinons la réduction $s' \rightarrow t'$. Il y a trois cas selon l'occurrence du radical :

I) La réduction a lieu dans le contexte, i.e. $C \rightarrow D$ et $t' = D[\mathbf{u}']$.

Posons $t = D[\mathbf{u}]$. Pour régler ce cas, il suffira de montrer que t' est le résultat d'une très bonne inflation de t et, comme $s \rightarrow t$ on pourra tenter $\theta_{s,s'} = \mathrm{prof}(s)$.

Nous consacrons les sections 2.5 et 2.6 à la démonstration de ce résultat. Dans la section 2.5 nous le montrons pour le cas où il s'agit d'une bonne inflation (proposition 2.4). Dans la section 2.6 nous montrons que pour toute bonne inflation on peut trouver une très bonne inflation de même résultat (proposition 2.5). Ce dernier résultat sera utile aussi pour régler un sous-cas du deuxième cas (II.i.3.2).

II) La réduction a lieu dans u_k'.

Il y aura deux sous-cas à considérer:

i) Si u_k n'est pas un w-terme, alors $u_k' = u_k \circ w_k$. On trouve ici la partie cruciale de la démonstration. Il faut raisonner selon l'emplacement du radical :

i.1) Le radical est $u_k \circ w_k$. Il faut étudier la règle utilisée. Le cas 2 de la discusion informelle de la section précedente est un bon exemple pour comprendre cette situation dans le cas de la règle *(Ass)*. Reprenons donc cette discusion. Nous avions

$$C[u'] = C[u \circ w] \rightarrow D[b \circ w] = D[b'] = t'$$

où $u = s = a \circ b$, $C = \Box$ et $D = a \circ \Box$. Le membre droit t' doit être vu comme le résultat d'une inflation d'un terme, disons t. Seul $t = s$ paraît être un choix raisonnable. Nous posons donc $t = s$, mais alors $\mathrm{prof}(t) = \mathrm{prof}(s)$. Il faut donc trouver un poids qui décroisse.

On voit que $\lg(D) > \lg(C)$, donc $\lg(s) - \lg(C) > \lg(s) - \lg(D)$, ce qui justifie l'introduction d'une deuxième composante dans l'ordinal $\theta_{s,s'}$. Nous proposons donc

$$\theta_{s,s'} = (\mathrm{prof}(s), \lg(s) - \lg(C)).$$

i.2) Le radical est contenu dans w_k.
Les deux composantes de l'ordinal sont inchangés, mais on pourra faire la récurrence sur la profondeur des w_k (on a montré que les w-termes sont fortement normalisables dans la section 2.2). Ceci nous oblige à ajouter une dernière composante à l'ordinal de récurrence. Nous proposons donc

$$\theta_{s,s'} = (\mathrm{prof}(s)\ ,\ \lg(s) - \lg(C)\ ,\ \textstyle\sum \mathrm{prof}(w_k))\ .$$

i.3) Le radical est contenu dans u_k. On montrera que la première composante décroît.

ii) Si u_k est un w-terme, alors ${u_k}' = w_k$ et $w_k \to w$. On montrera que la troisième composante décroît.

III) Il y a interaction entre C et $\mathbf{u}'$.

Grâce à la condition de très bon contexte, nous montrerons qu'il ne peut s'agir que d'une application de la règle *(Ass)* et que la deuxième composante décroît.

2.5 Réduction de contextes

Nous allons montrer que la notion de bonne inflation est stable par réduction du contexte.

Définition 2.11 *La* longueur d'un contexte C, *notée aussi* $\lg(C)$, *est définie par récurrence sur la complexité de* C *en étendant la définition de longueur d'un terme aux trous par* $\lg(\Box_n) = 0$.

Proposition 2.4 *Soit* $s' = C[\mathbf{u}']$ *le résultat d'une bonne inflation* $(C, \mathbf{w})$ *de* $s = C[\mathbf{u}]$. *Soit* D *un contexte tel que* $C \to D$, *et soient* $t = D[\mathbf{u}]$ *et* $t' = D[\mathbf{u}']$ *(donc,* $s \to t$ *et* $s' \to t'$*).*
Alors il existe une bonne inflation $(D', \mathbf{w}')$ *de* t *dont le résultat est* t'.

Démonstration
On le montrera par récurrence sur $\lg(C)$.

$\lg(C) \leq 1$: Il n'y a pas de réduction possible.

$C = A \cdot B$: La réduction a lieu dans A ou dans B. Analysons le premier cas (le deuxième est analogue).
Le terme $s' = (A \cdot B)[\mathbf{u}'] = A[\mathbf{u}'] \cdot B[\mathbf{u}']$ est donc le résultat de la bonne

inflation $(A \cdot B, \mathbf{w})$ de $s = (A \cdot B)[\mathbf{u}] = A[\mathbf{u}] \cdot B[\mathbf{u}]$. Soit E tel que $A \to E$; alors $A \cdot B \to E \cdot B = D$. Donc, $t = (E \cdot B)[\mathbf{u}] = E[\mathbf{u}] \cdot B[\mathbf{u}]$ et $t' = (E \cdot B)[\mathbf{u}'] = E[\mathbf{u}'] \cdot B[\mathbf{u}']$.
Posons $a = A[\mathbf{u}]$, $a' = A[\mathbf{u}']$, $b = B[\mathbf{u}]$, $b' = B[\mathbf{u}']$, $e = E[\mathbf{u}]$ et $e' = E[\mathbf{u}']$.
D'après la remarque 2.4, $(A, \mathbf{w})$ est une bonne inflation de a, alors par H.R. il existe une bonne inflation $(E', \mathbf{w}')$ de e de résultat e', i.e. si $e = E'[\mathbf{r}]$, alors $e' = E'[\mathbf{r}']$.
Soit $(E' \cdot B_q, \mathbf{w}'@\mathbf{w})$ où $q = \lg(\mathbf{r})$, et montrons qu'elle est une bonne inflation de $e \cdot b$ de résultat $e' \cdot b'$. Soit $\mathbf{v} = \mathbf{r}@\mathbf{u}$, alors

$$(E' \cdot B_q)[\mathbf{v}] = E'[\mathbf{v}] \cdot B_q[\mathbf{v}] = E'[\mathbf{r}] \cdot B[\mathbf{u}] = e \cdot b\,,$$

la dernière égalité étant justifiée par la remarque 2.2. Le résultat de cette inflation est

$$(E' \cdot B_q)[\mathbf{v}'] = E'[\mathbf{v}'] \cdot B_q[\mathbf{v}'] = E'[\mathbf{r}'] \cdot B[\mathbf{u}'] = e' \cdot b'\,,$$

avec la même justification pour la dernière égalité. Enfin, $(E' \cdot B_q, \mathbf{w}'@\mathbf{w})$ est une bonne inflation de $e \cdot b$, car $(E', \mathbf{w}')$ est une bonne inflation de e et $(B, \mathbf{w})$ est une bonne inflation de b.

$C = \lambda A$: La réduction a lieu dans A. On a donc une bonne inflation $(\lambda A, \mathbf{w})$ de $s = (\lambda A)[\mathbf{u}] = \lambda(A[\mathbf{u}])$ de résultat $s' = (\lambda A)[\mathbf{u}'] = \lambda(A[\mathbf{u}'])$. Soit E tel que $A \to E$, alors $\lambda A \to \lambda E = D$.
Posons $a = A[\mathbf{u}]$, $a' = A[\mathbf{u}']$, $e = E[\mathbf{u}]$ et $e' = E[\mathbf{u}']$.
D'après la remarque 2.4, $(A, \mathbf{w})$ est une bonne inflation de a, alors par H.R. il existe une bonne inflation $(E', \mathbf{w}')$ de e de résultat e'. On vérifie aisément que $(\lambda E', \mathbf{w}')$ est une bonne inflation de λe de résultat $\lambda e'$.

$C = A \circ B$: Il y a trois possibilités :

i) Si la réduction a lieu dans A, le raisonnement est analogue à celui du cas $C = A \cdot B$, mais il faut vérifier la condition de bon contexte. En effet, si $A \to E$, le candidat pour une bonne inflation de $t = (E \circ B)[\mathbf{u}]$, sera $(E' \circ B_q, \mathbf{w}'@\mathbf{w})$ (même notation que celle du cas $C = A \cdot B$). Deux possibilités :

   i.1) Si A contient au moins un trou, alors B est un w-trou car $A \circ B$ est bon. Mais alors B_q est aussi un w-trou et $E' \circ B_q$ est bon.

   i.2) Si A ne contient pas de trous, on prend $E' = E$ qui est un bon contexte, car le réduit d'un contexte sans trous est évidemment un contexte sans trous. Alors $E' \circ B_q$ est bon.

ii) Si la réduction a lieu dans B, alors B n'est pas un w-trou et, $A \circ B$ étant bon, A n'a pas de trous. Soit E tel que $B \to E$, alors $A \circ E'$ sera un bon contexte.

iii) Si la réduction a lieu à la racine, il faut considérer la règle utilisée.

(Map) : D étant un bon contexte, on prend l'inflation $(D, \mathbf{w})$.

(Abs) : Alors $C = (\lambda E) \circ B$ et $D = \lambda(E \circ (\mathtt{1} \cdot (B \circ \uparrow)))$. Il y a deux cas à considérer :

a) Si E contient au moins un trou, alors B est un w-trou $\Box_j$ et on prend la bonne inflation $(\lambda(E \circ \Box_{n_0}), \mathbf{w}')$ où $n_0 = n_C + 1$, $w'_k = w_k$ pour $k \neq n_0$ et $w'_{n_0} = \mathtt{1} \cdot (w_j \circ \uparrow)$. Vérifions que $(\lambda(E \circ \Box_{n_0}), \mathbf{w}')$ est une inflation de $t = D[\mathbf{u}]$: on a bien $t = (\lambda(E \circ \Box_{n_0}))[\mathbf{v}]$ où $\mathbf{v}$ est tel que $v_k = u_k$ pour $k \neq n_0$ et $v_{n_0} = \mathtt{1} \cdot (u_j \circ \uparrow)$, et le résultat est bien $t' = D[\mathbf{u}']$, car

$$D[\mathbf{u}'] = \lambda(E[\mathbf{u}'] \circ (\mathtt{1} \cdot (w_j \circ \uparrow))) = (\lambda(E \circ \Box_{n_0}))[\mathbf{v}']$$

puisque $v'_k = u'_k$ si $k \neq n_0$ et $v'_{n_0} = w'_{n_0} = \mathtt{1} \cdot (w_j \circ \uparrow)$.

b) Si E ne contient pas de trous, on prend comme bonne inflation $(\lambda(E \circ (\mathtt{1} \cdot (B \circ \Box_{n_0}))), \mathbf{w}')$ où $n_0 = n_C + 1$, $w'_k = w_k$ pour $k \neq n_0$ et $w'_{n_0} = \uparrow$. Remarquons que le contexte est bon, car on a pris la précaution de transformer $\uparrow$ en trou pour que la condition de bon contexte soit vraie pour les trous de B. Cette inflation est bien une inflation de $t = D[\mathbf{u}]$ qui a pour résultat $t' = D[\mathbf{u}']$.

(Ass) : Alors $C = (E \circ F) \circ B$. On a, comme pour *(Abs)*, deux cas à considérer :

a) Si E contient au moins un trou, alors F est un w-trou $\Box_{j_1}$ et B est un w-trou $\Box_{j_2}$. On prend la bonne inflation $(E \circ \Box_{n_0}, \mathbf{w}')$ où $n_0 = n_C + 1$, $w'_k = w_k$ pour $k \neq n_0$ et $w'_{n_0} = w_{j_1} \circ w_{j_2}$. On vérifie aisément qu'elle satisfait les conditions de la proposition.

b) Si E ne contient pas de trous, le réduit est un bon contexte et on prend la bonne inflation $(E \circ (F \circ B), \mathbf{w})$.

Autres règles : D étant un sous-contexte de C, on prend la bonne (remarque 2.4) inflation $(D, \mathbf{w})$.

□

2.6 Très bonnes inflations de même résultat

Dans cette section nous montrerons que pour toute bonne inflation on peut trouver une très bonne inflation de même résultat.

Notation 2.5 *Si C est un contexte on notera $\kappa_C = \mathrm{card}\,\{\gamma : C/\gamma \text{ est un trou}\}$. Autrement dit, κ_C est la quantité de trous du contexte C.*

Proposition 2.5 *Si $(C, \mathbf{w})$ est une bonne inflation de s de résultat s', il existe une très bonne inflation $(C', \mathbf{w}')$ de s de résultat s' telle que $\kappa_{C'} \leq \kappa_C$.*

Démonstration

On montrera la proposition par récurrence sur la complexité de C. Soit $\mathbf{u}$ tel que $s = C[\mathbf{u}]$ et $s' = C[\mathbf{u}']$.

$\lg(C) \leq 1$: Le résultat est immédiat car C est très bon et on prend l'inflation $(C', \mathbf{w}') = (C, \mathbf{w})$.

$C = A \cdot B$: Alors $s = (A \cdot B)[\mathbf{u}] = A[\mathbf{u}] \cdot B[\mathbf{u}]$ et $s' = (A \cdot B)[\mathbf{u}'] = A[\mathbf{u}'] \cdot B[\mathbf{u}']$. Posons $a = A[\mathbf{u}]$, $b = B[\mathbf{u}]$, $a' = A[\mathbf{u}']$ et $b' = B[\mathbf{u}']$, alors $s = a \cdot b$ et $s' = a' \cdot b'$.
Par la remarque 2.4, $(A, \mathbf{w})$ et $(B, \mathbf{w})$ sont des bonnes inflations de a et b, respectivement. Donc, par H.R., il existe des très bonnes inflations $(A', \mathbf{w_1})$ et $(B', \mathbf{w_2})$ de a et b dont les résultats sont a' et b', respectivement.
Soient $\mathbf{u_1}$ et $\mathbf{u_2}$ tels que $a = A'[\mathbf{u_1}]$, $a' = A'[\mathbf{u_1}']$, $b = B'[\mathbf{u_2}]$ et $b' = B'[\mathbf{u_2}']$. Et soit $q = \lg(\mathbf{u_1})$.
Prenons l'inflation $I = (A' \cdot B'_q, \mathbf{w_1} @ \mathbf{w_2})$. Elle est une inflation de s. En effet,

$$(A' \cdot B'_q)[\mathbf{u_1} @ \mathbf{u_2}] = A'[\mathbf{u_1}] \cdot B'[\mathbf{u_2}] = a \cdot b = s\,,$$

et son résultat est s', en effet

$$(A' \cdot B'_q)[(\mathbf{u_1} @ \mathbf{u_2})'] = (A' \cdot B'_q)[\mathbf{u_1}' @ \mathbf{u_2}'] = A'[\mathbf{u_1}'] \cdot B'[\mathbf{u_2}'] = a' \cdot b' = s'\,.$$

De plus, I est une bonne inflation car A' et B' sont bons.

Considérons l'hypothèse :

†) Soit A' est un w-trou et b est un w-terme, soit B'_q est un w-trou et a est un w-terme.

Démontrons maintenant l'affirmation suivante :

a) Si l'hypothèse †) n'est pas satisfaite, I est une très bonne inflation de s.

Supposons que $\mathbf{u_{1_k}}$ est un w-terme, A' étant très bon, $\mathbf{u_{1_k}}$ est w-maximal dans a. Il y a deux possibilités :

1. si $A' \neq \Box_k$, alors $\mathbf{u_{1_k}}$ est w-maximal dans $s = a \cdot b$.
2. si $A' = \Box_k$, alors comme la condition †) n'est pas satisfaite, b n'est pas un w-terme, et on conclut que $\mathbf{u_{1_k}}$ est w-maximal dans s.

On procède de même pour analyser les $\mathbf{u_{2_m}}$ qui sont des w-termes, et l'affirmation a) est donc démontrée.

Pour compléter l'étude du cas $C = A \cdot B$, posons $I' = (\Box_1, s')$. Démontrons l'affirmation suivante:

b) Si l'hypothèse †) est satisfaite, I' est une très bonne inflation de s dont le résultat est s'.

D'abord on constate que s' est un w-terme car, la condition †) étant satisfaite, s est un w-terme, et le résultat d'une inflation d'un w-terme est un w-terme. On vérifie aisément que le contexte $\Box_1$ est très bon.

Remarquons que pour les deux inflations I et I', la condition sur le cardinal de l'ensemble de trous est satisfaite. En effet, pour I on a

$$\kappa_{A'\cdot B'_q} = \kappa_{A'} + \kappa_{B'} \overset{HR}{\leq} \kappa_A + \kappa_B = \kappa_C$$

Puisqu'on choisit I' quand la condition †) est satisfaite, alors A' est un trou ou B' est un trou. Donc, $1 = \kappa_{A'} \leq \kappa_A$ ou $1 = \kappa_{B'} \leq \kappa_B$ et alors $\kappa_{\Box_1} = 1 \leq \kappa_C$.

$C = \lambda A$: Alors $s = (\lambda A)[\mathbf{u}] = \lambda(A[\mathbf{u}])$ et $s' = (\lambda A)[\mathbf{u}'] = \lambda(A[\mathbf{u}'])$.
Appelons $a = A[\mathbf{u}]$ et $a' = A[\mathbf{u}']$, alors $s = \lambda a$ et $s' = \lambda a'$.
Par la remarque 2.4, $(A, \mathbf{w})$ est une bonne inflation de a. Donc, par H.R., il existe une très bonne inflation $(A', \mathbf{w}')$ de a dont le résultat est a'.
On vérifie aisément que $(\lambda A', \mathbf{w}')$ est une très bonne inflation de s dont le résultat est s'. En plus $\kappa_{\lambda A'} = \kappa_{A'} \leq \kappa_A = \kappa_{\lambda A}$.

$C = A \circ B$: Alors $s = a \circ b$ et $s' = a' \circ b'$ (même notation que dans le cas $C = A \cdot B$).
Prenons l'inflation $I = (A' \circ B'_q, \mathbf{w_1}@\mathbf{w_2})$ (même notation que dans le cas $C = A \cdot B$ et vérification analogue du fait qu'elle est une inflation de s dont le résultat est s').

Maintenant nous nous servirons de la condition $\kappa_{C'} \leq \kappa_C$.

Si le contexte A' possède au moins un trou ($\kappa_{A'} \geq 1$), alors, par H.R., A possède au moins un trou ($\kappa_A \geq 1$). C étant bon, B ne peut être qu'un w-trou, alors $B' = B$ (voir le premier cas étudié : $\lg(C) \leq 1$). Donc B'_q est aussi un w-trou, et on conclut que I est bonne.

Considérons maintenant l'hypothèse :

‡) Soit A' est un w-trou et b est un w-terme, soit B'_q est un w-trou et a est un w-terme.

et le raisonnement se poursuit exactement comme dans le cas $C = A \cdot B$.

□

2.7 Le théorème de préservation

Nous finissons maintenant le travail des sections précédentes en démontrant le théorème de préservation.

Définition 2.12 *La* multiplicité *du trou* $\Box_k$ *dans le contexte* C *est l'entier*

$$\mu_k = \text{card}\,\{\gamma \,:\, C/\gamma = \Box_k\}$$

i.e. le nombre de fois où le trou $\Box_k$ *apparaît dans* C *.*

Théorème 2.1 (Préservation) *Soient* $s \in SN_0$ *et* s' *le résultat d'une très bonne inflation* $(C, \mathbf{w})$ *de* s *. Alors* $s' \in SN_0$ *.*

Démonstration
On montrera le théorème par récurrence sur l'ordinal

$$\theta = (\text{prof}(s)\,,\ \lg(s) - \lg(C)\,,\ \textstyle\sum \mu_k \text{prof}(w_k))\,.$$

Remarquons que $\text{prof}(w_k)$ est bien définie pour chaque w_k grâce à la proposition 2.3, et que $\lg(s) - \lg(C) \geq 0$, car C est un contexte relatif à s .

Soit $\mathbf{u}$ tel que $s = C[\mathbf{u}]$ et $s' = C[\mathbf{u}']$. On veut démontrer que toute réduction à partir de s' termine. Supposons donc que

$$s' = C[\mathbf{u}'] \rightarrow_{\sigma_0} t'.$$

et montrons que $t' \in SN$. On considère trois cas selon la position du radical :

I) La réduction a lieu dans le contexte, i.e. $C \rightarrow D$ et $t' = D[\mathbf{u}']$.

Posons $t = D[\mathbf{u}]$, alors par la proposition 2.4 il existe une bonne inflation $(D', \mathbf{w}')$ de t dont le résultat est t' . La proposition 2.5 nous fournit une très bonne inflation $(D'', \mathbf{w}'')$ de t avec résultat t' .

Or, comme $s \rightarrow t$, on a que $\text{prof}(t) < \text{prof}(s)$ et on peut appliquer l'H.R. pour conclure $t' \in SN$.

II) La réduction a lieu dans ${u_k}'$.

Appelons γ l'occurrence correspondant à ${u_k}'$, donc $C/\gamma = \Box_k$. Soit $q = \lg(\mathbf{u}) + 1$ et soit $C' = C\{\gamma \leftarrow \Box_q\}$. C' est un contexte relatif à s (car $s = C'[\mathbf{u}@u_k]$) qui est, en plus, très bon.

Il y a deux possibilités à considérer :

i) Si u_k n'est pas un w-terme, alors ${u_k}' = u_k \circ w_k$. Il y a trois sous-cas à considérer selon la position du radical.

i.1) Le radical est $u_k \circ w_k$.

Comme u_k n'est pas un w-terme, il n'y a que trois règles possibles pour réduire $u_k \circ w_k$ à la racine :

(Abs) : Alors $u_k = \lambda a$. On a :

$$s = C'[\mathbf{u}@(\lambda a)] \quad \text{et} \quad s' = C'[\mathbf{u}'@((\lambda a) \circ w_k)] .$$

Donc, $s' \to C'[\mathbf{u}'@(\lambda(a \circ (\mathtt{1} \cdot (w_k \circ \uparrow))))] = t'$.
Soit $C'' = C\{\gamma \leftarrow \lambda\Box_q\}$ et prenons l'inflation

$$I = (C'', \mathbf{w}@w_q)$$

où

$$w_q = \begin{cases} \mathtt{1} \cdot (w_k \circ \uparrow) & \text{si } a \text{ n'est pas un w-terme} \\ a \circ (\mathtt{1} \cdot (w_k \circ \uparrow)) & \text{autrement.} \end{cases}$$

Alors $s = C''[\mathbf{u}@a]$ et $t' = C''[\mathbf{u}'@a']$. Donc, t' est le résultat de l'inflation I de s .
En plus, I est bonne, car $C/\gamma = \Box_k$ n'est pas un w-trou, et très bonne car, dans le cas où a est un w-terme, a est w-maximal (λa n'est pas un w-terme).
Donc, on peut utiliser l'H.R. car la deuxième composante de θ décroît ($\lg(C'') > \lg(C)$).

(Map) : Alors $u_k = a \cdot b$. On a :

$$s = C'[\mathbf{u}@(a \cdot b)] \quad \text{et} \quad s' = C'[\mathbf{u}'@((a \cdot b) \circ w_k)] .$$

Donc, $s' \to C'[\mathbf{u}'@((a \circ w_k) \cdot (b \circ w_k))] = t'$.
Soit $C'' = C\{\gamma \leftarrow \Box_q \cdot \Box_{q+1}\}$ et prenons l'inflation

$$I = (C'', \mathbf{w}@(w_q, w_{q+1}))$$

où

$$w_q = \begin{cases} w_k & \text{si } a \text{ n'est pas un w-terme} \\ a \circ w_k & \text{autrement} \end{cases}$$

$$w_{q+1} = \begin{cases} w_k & \text{si } b \text{ n'est pas un w-terme} \\ b \circ w_k & \text{autrement.} \end{cases}$$

Alors $s = C''[\mathbf{u}@(a, b)]$ et $t' = C''[\mathbf{u}'@(a', b')]$. Donc, t' est le résultat de l'inflation I de s .
En plus, I est bonne, car $C/\gamma = \Box_k$ n'est pas un w-trou, et très bonne car $a \cdot b$ n'est pas un w-terme.
Donc, on peut utiliser l'H.R., car la deuxième composante de θ décroît ($\lg(C'') > \lg(C)$).

(Ass) : Alors $u_k = a \circ b$. On a :

$$s = C'[\mathbf{u}@(a \circ b)] \quad \text{et} \quad s' = C'[\mathbf{u}'@((a \circ b) \circ w_k)] .$$

Donc, $s' \to C'[\mathbf{u}'@(a \circ (b \circ w_k))] = t'$.
Soit $C'' = C\{\gamma \leftarrow a \circ \Box_q\}$ et prenons l'inflation

$$I = (C'', \mathbf{w}@w_q)$$

où

$$w_q = \begin{cases} w_k & \text{si } b \text{ n'est pas un w-terme} \\ b \circ w_k & \text{autrement.} \end{cases}$$

Alors $s = C''[\mathbf{u}@b]$ et $t' = C''[\mathbf{u}'@b']$. Donc, t' est le résultat de l'inflation I de s.
En plus, I est bonne car $C/\gamma = \Box_k$ n'est pas un w-trou et le nouveau trou est à droite de la composition qu'on vient d'ajouter. Et très bonne car $a \circ b$ n'est pas un w-terme.
Donc, on peut utiliser l'H.R., car la deuxième composante de θ décroît ($\lg(C'') > \lg(C)$).

i.2) Le radical est contenu dans w_k. Soit w tel que $w_k \to w$. Prenons l'inflation $(C', \mathbf{w}@w)$ de s dont le résultat est $C'[\mathbf{u}'@w] = t'$.
Le lemme 2.2 garantit que w est un w-terme, donc l'inflation est bien définie.
Comme $\text{prof}(w) < \text{prof}(w_k)$ et $\lg(C) = \lg(C')$, la troisième composante de l'ordinal θ décroît et on utilise l'H.R. pour conclure $t' \in SN$.

i.3) Le radical est contenu dans u_k. Soit u tel que $u_k \to u$. Deux possibilités :

i.3.1) Si u n'est pas un w-terme, on prend l'inflation $(C', \mathbf{w}@w_k)$. Alors

$$s' = C'[\mathbf{u}'@(u_k \circ w_k)] \to C'[\mathbf{u}'@(u \circ w_k)] = t'$$

Donc, t' est le résultat de l'inflation

$$(C', \mathbf{w}@w_k)$$

de $t = C'[\mathbf{u}@u]$. Mais $s \to t$, alors on peut utiliser l'H.R. (la première composante de θ ayant décru) pour obtenir $t' \in SN$.
Remarquons que C' est aussi très bon pour t car, u n'étant pas un w-terme, le trou $\Box_q$ continue à être un λ-trou.

i.3.2) Si u est un w-terme, on prend, comme dans le cas précédent, l'inflation $(C', \mathbf{w}@w_k)$.
C' n'est pas nécessairement très bon pour t car il y a eu mue de λ-trou en w-trou. Mais ce qui est sûr, c'est que C' est un bon contexte pour t (aucun w-trou n'a disparu).
La proposition 2.5 assure l'existence d'une très bonne inflation $(C'', \mathbf{w}'')$ de t dont le résultat est t', et puisque $s \to t$, on utilise l'H.R. (la première composante de θ ayant décru) pour obtenir $t' \in SN$.

ii) Si u_k est un w-terme, alors ${u_k}' = w_k$. Soit w tel que $w_k \to w$.
On prend l'inflation $(C', \mathbf{w}@w)$ de s, et on raisonne comme dans le cas i.2).

III) Il y a interaction entre C et $\mathbf{u}'$. On étudiera pour chaque règle de σ_0 toutes les interactions possibles. Soit γ l'occurrence du radical.

(VarId) : Trois possibilités :

i) $C/\gamma = \mathtt{1} \circ \Box_k$ et ${u_k}' = id$. Alors u_k est un w-terme qui n'est pas w-maximal. Ce cas est impossible car C est très bon

ii) $C/\gamma = \Box_k \circ id$ et ${u_k}' = \mathtt{1}$. Idem.

iii) $C/\gamma = \Box_j \circ \Box_k$, ${u_j}' = \mathtt{1}$ et ${u_k}' = id$. Idem.

(ShiftId) : Analogue au cas précédent.

(VarCons) : $C[\mathbf{u}']/\gamma = \mathtt{1} \circ (a \cdot b)$. Donc, $C/\gamma = C/\gamma 1 \circ C/\gamma 2$. Il y a deux possibilités :

i) $C/\gamma 1 = \mathtt{1}$, alors $C/\gamma 2 = \Box_k$ et ${u_k}' = a \cdot b$ (alors u_k est un w-terme).
Ce cas est impossible car C est très bon et u_k n'est pas w-maximal.

ii) $C/\gamma 1 = \Box_k$, alors, C étant bon, $C/\gamma 2 = \Box_j$ est un w-trou. Mais ${u_k}' = \mathtt{1}$ entraîne que u_k est un w-terme ce qui contredit la w-maximalité de u_j . Donc, ce cas est aussi impossible.

(ShiftCons) : Analogue au cas précédent.

(Map) : $C[\mathbf{u}']/\gamma = (a \cdot b) \circ c$. Donc, $C/\gamma = C/\gamma 1 \circ C/\gamma 2$ et $C/\gamma 1 = \Box_k$. Or, C étant bon, $C/\gamma 2 = \Box_j$ est un w-trou. Mais ${u_k}' = a \cdot b$ entraîne que u_k est un w-terme .
Donc, ce cas est aussi impossible car C est très bon, et u_j n'est pas w-maximal.

(Abs) : $C[\mathbf{u}']/\gamma = (\lambda a) \circ c$. Donc, $C/\gamma = C/\gamma 1 \circ C/\gamma 2$, $C/\gamma 1 = \Box_k$ et ${u_k}' = \lambda a$.
Ce cas est impossible car, par définition d'inflation, ${u_k}'$ est soit un w-terme, soit une composition.

(IdL) : $C[\mathbf{u}']/\gamma = id \circ a$. Donc, $C/\gamma = C/\gamma 1 \circ C/\gamma 2$, $C/\gamma 1 = \Box_k$ et ${u_k}' = id$. Or, C étant bon, $C/\gamma 2 = \Box_j$ est un w-trou, et alors $a = u_j$ est un w-terme. Donc, ce cas est aussi impossible car C est très bon, et u_j n'est pas w-maximal.

(Ass) : $C[\mathbf{u}']/\gamma = (a \circ b) \circ c$. Donc, $C/\gamma = C/\gamma 1 \circ C/\gamma 2$, $C/\gamma 1 = \Box_k$ et ${u_k}' = (a \circ b)$.
Comme C est bon, $C/\gamma 2 = \Box_j$ doit être un w-trou. Donc, $a \circ b$ n'est pas un w-terme (autrement u_k serait un w-terme et on contredirait la condition de très bon contexte). Alors, nécessairement, $u_k = a$, $w_k = b$ et a n'est pas un w-terme. Donc,

$$s'/\gamma = (u_k \circ w_k) \circ w_j \quad \text{et} \quad t' = s'\{\gamma \leftarrow u_k \circ (w_k \circ w_j)\}.$$

Soit $D = C\{\gamma 2 \leftarrow \Box_q\}$ où $q = \lg(\mathbf{u}) + 1$. Et considérons le contexte $D[u_k]_{\gamma 1}$.
C'est un contexte relatif à s, car $s = (D[u_k]_{\gamma 1})[\mathbf{u}@u_j]$, qui est en plus très bon d'après le lemme 2.3.
Considérons maintenant l'inflation $(D[u_k]_{\gamma 1}, \mathbf{w}@(w_k \circ w_j))$ de s.
Son résultat est

$$(D[u_k]_{\gamma 1})[\mathbf{u}'@(w_k \circ w_j)] = t' ,$$

car u_j est un w-terme.
Or, $\lg(C) = \lg(D) < \lg(D[u_k]_{\gamma 1})$. Donc, par H.R. (la deuxième composante de l'ordinal θ ayant décru), on conclut que t' est fortement normalisable.

□

Théorème 2.2 *Le σ_0-calcul et le σ-calcul sont noethériens.*

Démonstration
D'après le théorème précédent et l'exemple 2.4, on conclut que si $s \in \Lambda\sigma_0 - \Lambda\sigma_w$ et si $s \in SN_0$, alors $s \circ \uparrow \in SN_0$. Mais si $s \in \Lambda\sigma_w$, alors $s \circ \uparrow \in \Lambda\sigma_w$ et d'après la proposition 2.3, $s \circ \uparrow \in SN_0$. Donc,

$$\star \quad \text{si } s \in SN_0, \text{ alors } s \circ \uparrow \in SN_0.$$

Par la remarque 2.1 on a la normalisation forte pour le σ_0-calcul. Enfin, la proposition 2.1 fournit la noethérianité du σ-calcul.
□

2.8 Normalisation forte du σ'-calcul

Dans cette section nous étendons nos résultats de terminaison au σ'-calcul (cf. définition 1.38). Comme nous l'avons déjà remarqué, ce calcul possède une règle (*SCons*) qui fait penser à la règle de "surjective pairing" du λ-calcul avec couples. Nous avons aussi vérifié que ce calcul plus la règle *(Beta)*, i.e. le $\lambda\sigma'$-calcul, est localement confluent (cf. proposition 1.2) sur tous les termes, tandis que le $\lambda\sigma$-calcul ne l'est que sur les termes clos.

De même que dans la section 2.1, nous plongeons le σ'-calcul dans un calcul plus économique en syntaxe, appelé σ_0'. Voici sa définition :

Définition 2.13 *Le* σ_0'-calcul *est obtenu par l'ajout au σ_0-calcul des trois règles suivantes :*

$$\begin{array}{ll} (IdR) & s \circ id \to s \\ (VarShift) & \mathtt{1} \cdot \uparrow \to id \\ (SCons) & (\mathtt{1} \circ s) \cdot (\uparrow \circ s) \to s \end{array}$$

et l'élimination de (VarId) et (ShiftId) qui sont des instances de (IdR).
On appellera ISP le calcul dont les seules règles sont (IdR), (VarShift) et (SCons).

Remarque 2.5 *La fonction F introduite dans la définition 2.2 est aussi une interprétation stricte du σ'-calcul dans le ${\sigma_0}'$-calcul.*

On remarque la ressemblance du ${\sigma_0}'$-calcul avec le calcul SUBST étudié dans [Har89] (voir Annexe C). En effet, il existe une interprétation stricte de ${\sigma_0}'$ dans SUBST. Il y a de même une interprétation stricte de SUBST dans ${\sigma_0}'$. La noethérianité de ${\sigma_0}'$ entraîne donc la noethérianité de SUBST :

Proposition 2.6 *Si ${\sigma_0}'$ est noethérien, alors SUBST est noethérien.*

Démonstration
La seule difficulté (mineure) est la présence de la constante *App*.
Grâce à l'interprétation stricte G, donnée par

$$\begin{array}{lcl} Id & \mapsto & id \\ Fst & \mapsto & \uparrow \\ Snd & \mapsto & \mathtt{1} \\ \Lambda x & \mapsto & \lambda G(x) \\ x \circ y & \mapsto & G(x) \circ G(y) \\ < x\,,\, y > & \mapsto & G(y) \cdot G(x)\,, \end{array}$$

on déduit que tout terme ne contenant pas *App* est fortement normalisable.
Or, *App* étant une constante inerte pour SUBST, i.e. n'apparaissant dans aucune règle de SUBST, on montre facilement que tout terme de CCL est fortement normalisable.
□

Soulignons que, bien que les règles ajoutées à σ_0 soient des règles simplifiantes, le théorème de préservation ne s'étend pas au système ${\sigma_0}'$.

En effet, un nouveau sous-cas apparaît dans le cas d'interaction entre contexte et $\mathbf{u}'$ (cas III)) :

(IdR) : $C[\mathbf{u}']/\gamma = a \circ id$. Donc, $C/\gamma = C/\gamma 1 \circ C/\gamma 2$, $C/\gamma 2 = \Box_k$ et ${u_k}' = id$.

Or, u_k est un w-terme et $\mathbf{u}'$ ne contient aucune information dessus. Si on enlève id aux w-termes, on pourrait régler ce cas. Mais ce faisant on perd la stabilité par ${\sigma_0}'$-réduction pour les w-termes, à cause de la règle *(VarShift)*. Donc, on ne sait pas traiter ce cas.

Nous montrons que tout terme est ${\sigma_0}'$-fortement normalisable (théorème 2.3) par récurrence sur (σ_0-profondeur, longueur). Puisque la ISP-réduction fait diminuer

strictement la longueur, il suffit d'obtenir le résultat préliminaire suivant : la ISP-réduction n'augmente pas la σ_0-profondeur (proposition 2.7). Enfin, pour démontrer ce dernier résultat, il faut un lemme qui dit essentiellement qu'un pas de ISP-réduction suivi d'un pas de σ_0-réduction peut être simulé par une σ_0'-réduction commençant par un pas de σ_0-réduction, i.e. une sorte de lemme de commutation (lemme 2.4).

Nous commençons donc par ce lemme, dont nous donnons une démonstration très détaillée en analysant tous les cas possibles.

Lemme 2.4 *Soient* $s, s_1, s_2 \in \Lambda\sigma_0$ *tels que* $s \to_{ISP} s_1 \to_{\sigma_0} s_2$, *alors il existe* $s' \in \Lambda\sigma_0$ *tel que* $s \to_{\sigma_0} s' \twoheadrightarrow_{\sigma_0'} s_2$.

Démonstration
On montrera le lemme par récurrence sur la structure de s .

$\lg(s) = 1$: Le lemme est immédiat.

$s = t \cdot u$: La ISP-réduction peut avoir lieu dans t, dans u ou à la racine.

i) La ISP-réduction a lieu dans t ou dans u , disons dans t (l'autre cas est symétrique). Donc, $t \cdot u \to_{ISP} t_1 \cdot u$. Il y a deux sous-cas à considérer selon l'emplacement du σ_0-radical :

i.1) La σ_0-réduction a lieu dans t_1 . Posons t_2 tel que $t_1 \to_{\sigma_0} t_2$. Par H.R., il existe t' tel que $t \to_{\sigma_0} t' \twoheadrightarrow_{\sigma_0'} t_2$. Alors

$$t \cdot u \to_{\sigma_0} t' \cdot u \twoheadrightarrow_{\sigma_0'} t_2 \cdot u$$

et on prend $s' = t' \cdot u$.

i.2) La σ_0-réduction a lieu dans u . Posons u_1 tel que $u \to_{\sigma_0} u_1$. Alors il suffit de prendre $s' = t \cdot u_1$, car $t \cdot u \to_{\sigma_0} t \cdot u_1 \twoheadrightarrow_{\sigma_0'} t_1 \cdot u_1$.

ii) La ISP-réduction a lieu à la racine. Il y a deux possibilités :

ii.1) $t = \mathtt{1}$ et $u = \uparrow$, alors $s_1 = id$ qui est en σ_0-forme normale et il n'y a rien à démontrer.

ii.2) $t = \mathtt{1} \circ v$, $u = \uparrow \circ v$ et $s_1 = v$. Alors on prend, par exemple, $s' = (\mathtt{1} \circ s_2) \cdot (\uparrow \circ v)$, car

$$(\mathtt{1} \circ v) \cdot (\uparrow \circ v) \to_{\sigma_0} (\mathtt{1} \circ s_2) \cdot (\uparrow \circ v) \to_{\sigma_0} (\mathtt{1} \circ s_2) \cdot (\uparrow \circ s_2) \to_{ISP} s_2$$

$s = \lambda t$: La ISP-réduction a lieu dans t, de sorte que $\lambda t \to_{ISP} \lambda t_1$. Alors la σ_0-réduction a lieu dans t_1 . Posons t_2 tel que $t_1 \to_{\sigma_0} t_2$. Par H.R., il existe t' tel que $t \to_{\sigma_0} t' \twoheadrightarrow_{\sigma_0'} t_2$. Alors $\lambda t \to_{\sigma_0} \lambda t' \twoheadrightarrow_{\sigma_0'} \lambda t_2$ et on prend $s' = \lambda t'$.

$s = t \circ u$: La ISP-réduction peut avoir lieu dans t, dans u ou à la racine et la σ_0-réduction peut aussi avoir lieu à la racine. Donc, ce cas requiert une étude plus approfondie.

Si la ISP-réduction et la σ_0-réduction n'ont pas lieu à la racine, le raisonnement est analogue au sous-cas i) du cas $s = t \cdot u$.

Examinons les autres cas selon l'emplacement du ISP-radical. Pour chacun d'eux étudions quelles sont les σ_0-règles qui peuvent être utilisées pour réduire s_1.

i) La ISP-réduction a lieu à la racine, alors $u = id$ et $s_1 = t$. On prend $s' = s_2 \circ id$ qui vérifie les conditions du théorème.

ii) $t \to_{ISP} t_1$, alors $s_1 = t_1 \circ u$. Supposons donc que la σ_0-réduction a lieu à la racine. Avant d'analyser les σ_0-règles utilisées, nous étudierons les possibilités pour t. Dans deux cas nous pourrons conclure directement. Remarquons d'abord que si $t_1 \neq id$, il n'y a que trois possibilités :

P.1) Le symbole de tête de t est le même que celui de t_1 et la ISP-réduction est interne.

P.2) $t = (\mathtt{1} \circ t_1) \cdot (\uparrow \circ t_1)$

P.3) $t = t_1 \circ id$

Tandis que si $t_1 = id$, on a aussi le cas :

P.4) $t = \mathtt{1} \cdot \uparrow$

On peut traiter P.2) et P.3) indépendamment de la σ_0-règle utilisée :

- Si $t = (\mathtt{1} \circ t_1) \cdot (\uparrow \circ t_1)$, alors
$$s = ((\mathtt{1} \circ t_1) \cdot (\uparrow \circ t_1)) \circ u \to_{ISP} t_1 \circ u \to_{\sigma_0} s_2 .$$
Et il suffit de prendre
$$s' = (((\mathtt{1} \circ t_1) \circ u) \cdot ((\uparrow \circ t_1) \circ u))$$
car $s \to_{\sigma_0} s' \twoheadrightarrow_{\sigma_0} ((\mathtt{1} \circ (t_1 \circ u)) \cdot (\uparrow \circ (t_1 \circ u))) \to_{ISP} t_1 \circ u \to_{\sigma_0} s_2$.
- Si $t = t_1 \circ id$, alors
$$s = (t_1 \circ id) \circ u \to_{ISP} t_1 \circ u \to_{\sigma_0} s_2 .$$
Et il suffit de prendre
$$s' = t_1 \circ (id \circ u)$$
car $s \to_{\sigma_0} s' \to_{\sigma_0} t_1 \circ u \to_{\sigma_0} s_2$

Supposons donc qu'on est dans la situation P.1) ou P.4). Examinons le σ_0-radical qui est, rappelons-le, à la racine :

(VarId) **ou** *(VarCons)* : Donc, $t_1 = \mathtt{1}$. Ceci est impossible.

(ShiftId) **ou** *(ShiftCons)* : Analogue au cas précédent.

(Map) : $t_1 = v_1 \cdot w_1$ et $s_2 = (v_1 \circ u) \cdot (w_1 \circ u)$. Il y a deux possibilités :

a) $t = v \cdot w_1$ et $v \to_{ISP} v_1$, alors on prend $s' = (v \circ u) \cdot (w_1 \circ u)$, car $(v \cdot w_1) \circ u \to_{\sigma_0} (v \circ u) \cdot (w_1 \circ u) \to_{ISP} (v_1 \circ u) \cdot (w_1 \circ u)$.

b) $t = v_1 \cdot w$ et $w \to_{ISP} w_1$, analogue au sous-cas a).

(Abs) : $t_1 = \lambda v_1$ et $s_2 = \lambda(v_1 \circ (\mathtt{1} \cdot (u \circ \uparrow)))$. Donc, $t = \lambda v$ avec $v \to_{ISP} v_1$. On prend $s' = \lambda(v \circ (\mathtt{1} \cdot (u \circ \uparrow)))$ qui vérifie

$$(\lambda v) \circ u \to_{\sigma_0} s' \to_{ISP} s_2 .$$

(IdL) : $t_1 = id$ et $s_2 = u$. Donc, $t = \mathtt{1} \cdot \uparrow$, et il suffit de prendre $s' = (\mathtt{1} \circ u) \cdot (\uparrow \circ u)$.

(Ass) : Analogue à l'étude de la règle *(Map)*.

iii) Si $u \to_{ISP} u_1$, alors $s_1 = t \circ u_1$. La discussion est menée suivant la σ_0-règle utilisée dans la réduction $s_1 \to_{\sigma_0} s_2$ supposée à la racine.

(VarId) : $t = \mathtt{1}$, $u_1 = id$ et $s_2 = \mathtt{1}$. On a trois cas :

a) $u = (\mathtt{1} \circ id) \cdot (\uparrow \circ id)$, on prend $s' = \mathtt{1} \circ id$, car
$$\mathtt{1} \circ ((\mathtt{1} \circ id) \cdot (\uparrow \circ id)) \to_{\sigma_0} \mathtt{1} \circ id \to_{\sigma_0} \mathtt{1}.$$

b) $u = id \circ id$, on prend $s' = \mathtt{1} \circ id$, car
$$\mathtt{1} \circ (id \circ id) \to_{\sigma_0} \mathtt{1} \circ id \to_{\sigma_0} \mathtt{1} .$$

c) $u = \mathtt{1} \cdot \uparrow$, on prend $s' = \mathtt{1}$, car
$$\mathtt{1} \circ (\mathtt{1} \cdot \uparrow) \to_{\sigma_0} \mathtt{1} \twoheadrightarrow_{\sigma_0'} \mathtt{1} .$$

(ShiftId) : On raisonne comme dans le cas précédent.

(VarCons) : $t = \mathtt{1}$, $u_1 = v_1 \cdot w_1$ et $s_2 = v_1$. Il y a quatre possibilités :

a) $u = v \cdot w_1$ et $v \to_{ISP} v_1$, on prend $s' = v$.

b) $u = v_1 \cdot w$ et $w \to_{ISP} w_1$, on prend $s' = v_1$, car $v_1 \twoheadrightarrow_{\sigma_0'} v_1$.

c) $u = (\mathtt{1} \circ (v_1 \cdot w_1)) \cdot (\uparrow \circ (v_1 \cdot w_1))$, alors on prend
$$s' = \mathtt{1} \circ (v_1 \cdot ((\uparrow \circ (v_1 \cdot w_1)))) ,$$
puisque $\mathtt{1} \circ u \to_{\sigma_0} s' \to_{\sigma_0} v_1$.

d) $u = (v_1 \cdot w_1) \circ id$, alors on prend
$$s' = \mathtt{1} \circ ((v_1 \circ id) \cdot (w_1 \circ id)),$$
car $\mathtt{1} \circ u \to_{\sigma_0} s' \to_{\sigma_0} v_1 \circ id \to_{ISP} v_1$.

(ShiftCons) : On raisonne comme dans le cas précédent.

(Map) : $t = v \cdot w$ et $s_2 = (v \circ u_1) \cdot (w \circ u_1)$. Alors il suffit de prendre
$$s' = (v \circ u) \cdot (w \circ u) .$$

(Abs) : $t = \lambda v$ et $s_2 = \lambda(v \circ (\mathtt{1} \cdot (u_1 \circ \uparrow)))$. Prendre
$$s' = \lambda(v \circ (\mathtt{1} \cdot (u \circ \uparrow))) .$$

(IdL) : $t = id$ et $s_2 = u_1$. Alors on prend $s' = u$.

(Ass) : Analogue à l'étude de la règle *(Map)*.

□

Notation 2.6 *Pour $s \in \Lambda\sigma_0$, on notera* pr(s) *la* profondeur *de s par rapport à la réduction σ_0, i.e. la longueur de la σ_0-réduction la plus longue à partir de s.*

Proposition 2.7 *Soient* $s, t \in \Lambda\sigma_0$ *tels que* $s \to_{ISP} t$, *alors* $\mathrm{pr}(t) \leq \mathrm{pr}(s)$.

Démonstration
La preuve se fait par récurrence sur $\mathrm{pr}(s)$, en étudiant $\mathrm{pr}(t)$.

Si $\mathrm{pr}(t) = 0$, il n'y a rien à démontrer. Supposons donc que $\mathrm{pr}(t) > 0$. Soit $u \in \Lambda\sigma_0$ tel que $t \to_{\sigma_0} u$ et $\mathrm{pr}(u) + 1 = \mathrm{pr}(t)$ (i.e. on prend u sur un chemin de réduction de longueur maximale à partir de t).

D'après le lemme 2.4 il existe $s_1 \in \Lambda\sigma_0$ tel que $s \to_{\sigma_0} s_1 \twoheadrightarrow_{\sigma_0'} u$. On montre que

$*$ Pour tout $v \in \Lambda\sigma_0$ tel que $s_1 \twoheadrightarrow_{\sigma_0'} v$, on a $\mathrm{pr}(v) \leq \mathrm{pr}(s_1)$.

La preuve de $*$ est faite par récurrence sur la longueur, notée L, de la réduction $s_1 \twoheadrightarrow_{\sigma_0'} v$. Rappelons que la notation $s_1 \overset{n}{\twoheadrightarrow}_{\sigma_0'} s_2$ signifie que s_1 se réduit par s_2 en n pas de réduction.

Si $L = 0$, alors $v = s_1$ et le résultat est immédiat. Supposons le résultat vrai pour $L = n$ et appelons cette hypothèse L-H.R.. Soit $s_2 \in \Lambda\sigma_0$ tel que $s_1 \overset{n}{\twoheadrightarrow}_{\sigma_0'} s_2 \to_{\sigma_0'} v$. Par L-H.R. on a

$$\mathrm{pr}(s_2) \leq \mathrm{pr}(s_1) \tag{2.1}$$

et, comme $s \to_{\sigma_0} s_1$, on a aussi $\mathrm{pr}(s_1) < \mathrm{pr}(s)$. Donc, $\mathrm{pr}(s_2) < \mathrm{pr}(s)$. Il y a deux possibilités :

- Si $s_2 \to_{\sigma_0} v$, alors $\mathrm{pr}(v) < \mathrm{pr}(s_2)$ et, par (2.1) et transitivité, on conclut $\mathrm{pr}(v) \leq \mathrm{pr}(s_1)$.
- Si $s_2 \to_{ISP} v$, alors par H.R. (car $\mathrm{pr}(s_2) < \mathrm{pr}(s)$), $\mathrm{pr}(v) \leq \mathrm{pr}(s_2)$ et, d'après (2.1), $\mathrm{pr}(v) \leq \mathrm{pr}(s_1)$.

On a donc démontré $*$, et on peut conclure que $\mathrm{pr}(u) \leq \mathrm{pr}(s_1)$. Alors

$$\mathrm{pr}(t) - 1 = \mathrm{pr}(u) \leq \mathrm{pr}(s_1) < \mathrm{pr}(s).$$

Donc, $\mathrm{pr}(t) \leq \mathrm{pr}(s)$.
□

La proposition précédente nous permet de démontrer aisément le théorème de normalisation forte pour les σ_0' et σ'-calculs.

Théorème 2.3 *Le σ_0'-calcul, et donc le σ'-calcul, sont noethériens.*

Démonstration
On montrera que tout $s \in \Lambda\sigma_0$ est fortement normalisable par récurrence sur $(\mathrm{pr}(s), \mathrm{lg}(s))$.
Si s est en σ_0'-forme normale, s est fortement normalisable. Soit donc $t \in \Lambda\sigma_0$ tel que $s \to_{\sigma_0'} t$ et montrons que t est fortement normalisable :

- Si $s \to_{\sigma_0} t$, alors $\mathrm{pr}(t) < \mathrm{pr}(s)$, et par H.R. t est fortement normalisable.

– Si $s \rightarrow_{ISP} t$, alors par la proposition précédente $\mathrm{pr}(t) \leq \mathrm{pr}(s)$. Mais la ISP-réduction fait diminuer la taille du terme réduit, donc la deuxième composante de l'ordinal décroît strictement, et par H.R. on conclut que t est fortement normalisable.

La noethérianité de ${\sigma'}_0$ étant établie, la remarque 2.5 assure la noethérianité de σ'.
□

Chapitre 3

Confluence du $\lambda\sigma'$-calcul semi-clos

Dans ce chapitre nous montrons la confluence du $\lambda\sigma'$-calcul (cf. définition 1.38) sur les termes semi-clos (on admet des variables de type terme proprement dit, mais pas de type substitution). Nous appelons ce calcul le $\lambda\sigma'$-calcul semi-clos. Ce résultat de confluence a été conjecturé dans [ACCL90] ainsi que la non-confluence du $\lambda\sigma'$-calcul ouvert (variables de type terme et substitution).

Cette dernière conjecture a été prouvée (cf. lemme 3.9 dans [CHL91]) par un contrexemple qui contient essentiellement des variables de type substitution. Le problème de la confluence du calcul semi-clos restait donc ouvert. La preuve que nous présentons ici utilise la méthode d'interprétation (cf. lemme 1.4). Cette méthode était déjà suggérée dans [ACCL90] pour étudier la confluence du calcul semi-clos.

La confluence du $\lambda\sigma_{\Uparrow}$-calcul sur la totalité des termes/substitutions a aussi été montrée en utilisant la méthode d'interprétation (cf. [HL89] et [Har92]). Cependant, la technique que nous employons pour $\lambda\sigma'$ diffère de celle présentée dans les articles cités. Notre outil essentiel est la récurrence structurelle, tandis que pour le $\lambda\sigma_{\Uparrow}$-calcul cette technique ne s'avère pas efficace (voir [Har92], section 1.2). C'est l'étude des résidus des *(Beta)*-radicaux qui fournit les résultats désirés dans le $\lambda\sigma_{\Uparrow}$-calcul.

La remarque que nous venons de faire sur la différence de techniques est aussi valide pour le $\lambda\sigma'$-calcul extensionnel, que nous étudierons dans le chapitre suivant, et le $\lambda\sigma_{\Uparrow}$-calcul extensionnel.

Le plan de ce chapitre est le suivant :

Nous commençons par caractériser les ensembles des σ-formes normales et des σ'-formes normales, dont nous étudions quelques propriétés (section 3.1). Nous introduisons ensuite le λ'-calcul (section 3.2) sur les σ'-formes nomales, dans lequel nous interprétons le $\lambda\sigma'$-calcul. Le λ'-calcul contient une seule règle appelée β', qui peut être décrite comme *(Beta)* suivie de σ'-normalisation du terme entier. D'après cette description, on constate aisément que la règle β' peut être simulée par une $\lambda\sigma'$-dérivation. Le reste de la section 3.2 est consacré à montrer que le λ'-calcul est une extension du $\lambda\beta$-calcul (λ-calcul à la de Bruijn). Ce résultat (corollaire 3.1.2) ne sera utilisé que pour donner une nouvelle preuve d'un résultat déjà connu : la confluence du $\lambda\beta$-calcul. Cependant les résultats de cette section sont intéressants car ils ren-

seignent sur les rapports entre les substitutions explicites et les méta-substitutions du λ-calcul classique.

Pour appliquer la méthode d'interprétation il faut vérifier que la règle *(Beta)* du $\lambda\sigma'$-calcul se traduit en une β'-dérivation sur les σ'-formes normales, plus précisément :

$$\text{Si } f \to_{(Beta)} g \text{ , alors } \sigma'(f) \twoheadrightarrow_{\beta'} \sigma'(g) .$$

Nous consacrons la section 3.3 à la preuve de ce résultat et signalons les difficultés que l'on trouve quand on essaye d'étendre ces résultats au calcul ouvert.

Nous montrons la confluence du λ'-calcul en adaptant la preuve de confluence du λ-calcul classique donnée dans [Bar84] et due à Tait-Martin-Löf. Nous définissons donc une parallélisation de $\to_{\beta'}$ de sorte que leurs clôtures reflexives transitives coïncident (section 3.4).

Pour obtenir la confluence de la parallélisation, un lemme de substitution est nécessaire. Nous consacrons la section 3.5 à ce résultat qui est le corollaire 3.5.

Nous avons enfin les élements nécessaires pour montrer la confluence de la parallélisation qui fournit à son tour celle du λ'-calcul et permet donc d'utiliser la méthode d'interprétation afin d'obtenir la confluence du $\lambda\sigma'$-calcul (section 3.6).

3.1 Description des formes normales

Rappelons d'abord que l'ensemble des termes semi-clos (termes proprement dits ouverts et substitutions closes) ΛX est donné par :

Termes	$a ::= X \mid \mathtt{1} \mid ab \mid \lambda a \mid a[s]$
Substitutions	$s ::= id \mid \uparrow \mid a \cdot s \mid s \circ t$

où X parcourt un ensemble dénombrable de variables de type `terme`. Rappelons aussi que ΛX^t dénote l'ensemble des termes proprement dits et ΛX^s dénote l'ensemble des substitutions.

Comme d'habitude, nous identifions les termes $\mathtt{1}[\uparrow^n]$ avec les indices de de Bruijn $\mathtt{n+1}$. Nous convenons de poser $\uparrow^0 = id$.

Remarque 3.1 *Puisque aucun membre gauche des σ-règles n'est ni une application, ni une abstraction, ni de la forme $X[s]$, ni un cons, pour $a, b \in \Lambda X^t$, $s \in \Lambda X^s$, on a :*

1. $\sigma(ab) = \sigma(a)\,\sigma(b)$.
2. $\sigma(\lambda a) = \lambda\sigma(a)$.
3. $\sigma(X[s]) = X[\sigma(s)]$.
4. $\sigma(a \cdot s) = \sigma(a) \cdot \sigma(s)$.

Les termes en σ-forme normale qui sont des clôtures sont restreints aux termes de la forme $1[\uparrow^n]$ et $X[s]$, où s est en σ-forme normale. Les seules compositions en σ-f.n. sont de la forme $\uparrow^n$. Ceci est un aspect essentiel du calcul semi-clos par rapport au calcul ouvert. Nous revenons sur cette différence radicale qui permet d'établir la confluence du calcul semi-clos à la fin de la section 3.3. La description précise des σ-f.n. sur ΛX est la suivante:

Proposition 3.1 *Les termes et substitutions en σ-forme normale dans ΛX sont donnés récursivement par:*

$$\begin{array}{ll} \textbf{Termes} & a ::= \mathtt{1} \mid ab \mid \lambda a \mid 1[\uparrow^n] \mid X \mid X[s] \\ \textbf{Substitutions} & s ::= id \mid \uparrow^n \mid a \cdot s \end{array}$$

Rappelons que $\uparrow^n$ est défini par récurrence sur n ainsi: $\uparrow^1 = \uparrow$ et $\uparrow^{n+1} = \uparrow \circ \uparrow^n$.

Démonstration
D'abord on remarque que les termes et substitutions ainsi définies sont en fait des formes normales. On montrera qu'il n'y en a pas d'autres.

Supposons que $a[s]$ soit en forme normale, alors a est nécessairement $\mathtt{1}$ ou X, autrement on aurait un *(App)*, un *(Abs)* ou un *(Clos)*-radical. Analysons maintenant les possibilités pour s, en supposant $a = \mathtt{1}$. On voit que s ne peut être qu'une composition ou $\uparrow$, autrement on aurait un *(VarId)* ou un *(VarCons)*-radical. Si $s = t \circ u$, alors $t = \uparrow$ nécessairement, autrement on aurait un *(IdL)*, un *(Map)* ou un *(Ass)*-radical. Démontrons l'affirmation suivante:

Affirmation *Si $\uparrow \circ u$ est en σ-forme normale, il existe $n \geq 1$ tel que $u = \uparrow^n$.*

On la montre par récurrence sur la complexité de u. Si $\lg(u) = 1$, alors $u = \uparrow$ ($u = id$ est impossible par *(ShiftId)*). Supposons donc que $\uparrow \circ u$ est en f.n. et $\lg(u) > 1$, u ne peut être qu'une composition, autrement on aurait un *(ShiftId)* ou un *(ShiftCons)*-radical. Alors $u = u_1 \circ u_2$, mais nécessairement $u_1 = \uparrow$, autrement on aurait un *(IdL)*, un *(Map)* ou un *(Ass)*-radical. On conclut par H.R. que $u_2 = \uparrow^m$, et donc $u = \uparrow^{m+1}$. L'affirmation est donc démontrée. □

Alors $s = \uparrow^n$ et $a[s] = \mathtt{1}[\uparrow^n]$. D'autre part, si $a = X$, il n'y a pas de restriction sur la substitution s.

Remarquons que si ab est en f.n., alors a et b sont en f.n., et que si λa est en f.n., alors a est en f.n. (cf. remarque précédente). On a donc démontré que les termes en forme normale sont exactement ceux décrits dans l'enoncé de la proposition.

Etudions maintenant les substitutions en forme normale. Il sufit d'étudier les compositions car $a \cdot s$ est en f.n. ssi a et s sont en f.n. (cf. remarque précédente). Soit donc $s \circ t$ en f.n. et analysons les possibilités pour s et t. On constate que s ne peut être que $\uparrow$, autrement on aurait un *(IdL)*, un *(Map)* ou un *(Ass)*-radical. L'affirmation précédente nous permet de conclure que $t = \uparrow^n$, ce qui démontre la

proposition.
□

Notation 3.1 *L'ensemble des termes en σ-forme normale est noté ΛX^t_σ et l'ensemble des substitutions en σ-forme normale est noté ΛX^s_σ, tandis que $\Lambda X_\sigma = \Lambda X^t_\sigma \cup \Lambda X^s_\sigma$.*

Nous décrivons maintenant les σ'-formes normales. Il faut ajouter deux restrictions imposées par les nouvelles règles : s ne peut pas être id dans $X[s]$ et il ne doit pas exister $n \in \mathbb{N}$ tel que $a = \mathtt{n}$ et $s = \uparrow^n$ dans $a \cdot s$.

Proposition 3.2 *Les termes et substitutions en σ'-forme normale sont donnés récursivement par :*

Termes	$a ::= \mathtt{1} \mid ab \mid \lambda a \mid 1[\uparrow^n] \mid X \mid X[s]$ *$(s \neq id)$*
Substitutions	$s ::= id \mid \uparrow^n \mid a \cdot s$ *$(a \neq \mathtt{n}$ ou $s \neq \uparrow^n)$*

Démonstration
La preuve est analogue à celle de la proposition précédente. Seule l'étude de $X[s]$ et de $a \cdot s$ deviennent différentes :

- Si $X[s]$ est en σ'-f.n., $s \neq id$ (autrement on aurait un *(Id)*-radical) et celle-ci est la seule restriction.
- Si $a \cdot s$ est en f.n. et $a = \mathtt{1}$, alors $s \neq \uparrow$ (autrement on aurait un *(VarShift)*-radical).
  Si $a = \mathtt{n}$, avec $n > 1$, alors $s \neq \uparrow^n$ (autrement on aurait un *(SCons)*-radical).

□

Notation 3.2 *L'ensemble des termes en σ'-forme normale est noté $\Lambda X^t_{\sigma'}$ et l'ensemble des substitutions en σ'-forme normale est noté $\Lambda X^s_{\sigma'}$, tandis que $\Lambda X_{\sigma'} = \Lambda X^t_{\sigma'} \cup \Lambda X^s_{\sigma'}$.*

Les deux propositions précédentes nous permettent de décrire les substitutions en forme normale ainsi :

Remarque 3.2 *Soit $s \in \Lambda\sigma^s$.*

1. *$s \in \Lambda X^s_\sigma$ ssi il existe $k \geq 0$, $n \geq 0$ et $a_1, \ldots, a_k \in \Lambda X^t_\sigma$ tels que*

$$s = a_1 \cdot \ldots \cdot a_k \cdot \uparrow^n .$$

2. *$s \in \Lambda X^s_{\sigma'}$ ssi il existe $k \geq 0$, $n \geq 0$ et $a_1, \ldots, a_k \in \Lambda X^t_{\sigma'}$ tels que*

$$s = a_1 \cdot \ldots \cdot a_k \cdot \uparrow^n \text{ et } a_k \neq \mathtt{n} .$$

Dans les deux cas, nous appelons k *la* complexité *de* s *et* n *sa* hauteur.
Pour $k = 0$ *nous convenons* $a_1 \cdot \ldots \cdot a_k \cdot \uparrow^n = \uparrow^n$.

La remarque suivante, dont la démonstration se fait par récurrence sur m , sera utilisée constamment au cours de ce chapitre.

Remarque 3.3 *Soient* $a_1, \ldots, a_k \in \Lambda X^t$ *et* $k, m, n \geq 0$, *alors*

$$\uparrow^m \circ (a_1 \cdot \ldots \cdot a_k \cdot \uparrow^n) \twoheadrightarrow_\sigma \begin{cases} a_{m+1} \cdot \ldots \cdot a_k \cdot \uparrow^n & si \quad m < k \\ \uparrow^{n+m-k} & si \quad m \geq k \, . \end{cases}$$

Nous faisons maintenant quelques remarques sur les formes normales. Ces remarques seront utilisées dans la suite sans mention explicite.

Rappelons d'abord que l'ensemble de règles σ' est obtenu en ajoutant à σ l'ensemble des règles

$$SPID = \{(Id), (IdR), (VarShift), (SCons)\}$$

introduit dans la définition 1.38. La confluence et la noethérianité du σ'-calcul (cf. corollaire 1.2 et théorème 2.3) et du $SPID$-calcul (cf. lemme 1.7) assurent l'existence et l'unicité des formes normales notées $\sigma'(f)$ et $S(f)$, respectivement.

Remarque 3.4 *Puisque aucun membre gauche des* σ'*-règles n'est ni une abstraction ni une application, pour* $a, b \in \Lambda X^t$, *on a :*

1. $\sigma'(a\, b) = \sigma'(a)\, \sigma'(b)$ *et* $S(a\, b) = S(a)\, S(b)$.
2. $\sigma'(\lambda a) = \lambda \sigma'(a)$ *et* $S(\lambda a) = \lambda S(a)$.

Remarque 3.5 *Soient* $a \in \Lambda X^t$ *et* $s \in \Lambda X^s$. *Puisque* $X[s] \twoheadrightarrow_{SPID} X[S(s)]$ *et* $a \cdot s \twoheadrightarrow_{SPID} S(a) \cdot S(s)$, *on a :*

1. $S(X[s]) = X[S(s)]$ *ssi* $S(s) \neq id$.
2. $S(X[s]) = X$ *ssi* $S(s) = id$.
3. $S(a \cdot s) = S(a) \cdot S(s)$ *ssi* $\neg(\exists n \geq 1)(S(a) = \mathtt{n} \wedge S(s) = \uparrow^n)$.
4. $S(a \cdot s) = \uparrow^{n-1}$ *ssi* $S(a) = \mathtt{n} \wedge S(s) = \uparrow^n$.

Les énoncés précédents sont encore vrais si l'on substitue partout S *par* σ' .

On peut montrer par récurrence sur $f \in \Lambda X_\sigma$ l'égalité $\sigma'(f) = S(f)$ et donc on peut conclure que tout $g \in \Lambda X$ satisfait $\sigma'(g) = S(\sigma(g))$, i.e. que la mise en σ'-f.n. peut se faire en deux étapes : d'abord, réduction à σ-f.n., et ensuite mise en $SPID$-f.n..

Cependant nous n'utiliserons qu'un résultat plus faible qui décrit les σ'-f.n. des clôtures et des consings :

Lemme 3.1 *Soient $a \in \Lambda X^t$ et $s \in \Lambda X^s$.*

1. $\sigma'(X[s]) = S(X[\sigma'(s)])$.
2. $\sigma'(a \cdot s) = S(\sigma'(a) \cdot \sigma'(s))$.

Démonstration
Démontrons par exemple le premier résultat. La remarque précédente assure :
Si $\sigma'(s) \neq id$, alors $\sigma'(X[s]) = X[\sigma'(s)] = S(X[\sigma'(s)])$.
Si $\sigma'(s) = id$, alors $\sigma'(X[s]) = X = S(X[\sigma'(s)])$.
□

3.2 Le λ'-calcul

Nous allons réduire le problème de confluence du $\lambda\sigma'$-calcul à la confluence d'un calcul sur les σ'-formes normales : le λ'-calcul, qui s'avère une extension du λ-calcul.

M. Abadi avait suggéré la méthode suivante pour prouver la confluence d'un calcul qui opérerait sur les σ'-f.n. : essayer de traduire les termes du type $X[a \cdot b \cdot \ldots]$ en une forme normale de tête du λ-calcul classique $x\, a\, b \ldots$, où x est une variable fixée de ce dernier calcul. Cette méthode informelle pose bien sûr le problème de la traduction de $\uparrow^n$, mais elle donne quand même une idée intuitive de la raison pour laquelle le calcul, que nous allons traiter formellement, est confluent.

La seule règle du λ'-calcul consiste en une application de la règle *(Beta)* suivie d'une réduction totale (i.e. de tout le terme) à sa σ'-forme normale. Plus précisément :

Définition 3.1 *Le λ'-calcul est défini sur les σ'-formes normales par la réduction β' donnée par :*

$$f \to_{\beta'} g \quad ssi \quad il\ existe\ h \in \Lambda X\ tel\ que\ f \to_{(Beta)} h \quad et \quad g = \sigma'(h).$$

Rappelons que $f \to_{(Beta)} h$ signifie : il existe un contexte C est des termes a, b tels que $f = C[(\lambda a)b]$ et $h = C[\,a[b \cdot id]\,]$.

Evidemment, d'après cette définition, toute β'-dérivation peut être simulée dans le $\lambda\sigma'$-calcul :

Remarque 3.6 *Soient $f, g \in \Lambda X_{\sigma'}$ tels que $f \twoheadrightarrow_{\beta'} g$, alors $f \twoheadrightarrow_{\lambda\sigma'} g$.*

Signalons enfin que la β'-réduction est une extension de la β-réduction du λ-calcul classique à la de Bruijn. Rappelons-la :

$$(\lambda a)b \rightarrow_\beta a\{\!\{\mathtt{1} \leftarrow b\}\!\}$$

où les méta-substitutions $-\{\!\{\mathtt{k} \leftarrow -\}\!\}$ sont définies à l'aide des fonctions d'actualisation U_i^n (cf. définitions 1.24, 1.25 et 1.26).

Le reste de cette section est dédié à montrer cette extension. Comme nous l'avons dit au commencement de ce chapitre, ce résultat ne sera pas utilisé dans la suite, sauf dans la preuve du corollaire 3.6, mais nous voulons les inclure ici car ils montrent le rapport entre les substitutions explicites et les méta-substitutions du λ-calcul classique.

Nous montrons d'abord $\sigma'(a[b \cdot id]) = a\{\!\{\mathtt{1} \leftarrow b\}\!\}$, si $a, b \in \Lambda$.

Définition 3.2 *Pour $i \geq 0$ et $n \geq 1$ nous définissons la famille des substitutions $s_{i\,n}$ par*

$$s_{i\,n} = \mathtt{1} \cdot \mathtt{2} \cdot \ldots \cdot \mathtt{i} \cdot \uparrow^{i+n-1} .$$

On convient $s_{0\,n} = \uparrow^{n-1}$ et $s_{0\,1} = id$.

Grâce aux règles *(Clos)* et *(Ass)*, pour tout indice de De Bruijn $\mathtt{n}$, on a que $\mathtt{n}[\uparrow] \twoheadrightarrow_\sigma \mathtt{n}+\mathtt{1}$. Donc, les règles *(IdL)* et *(Map)* assurent :

Remarque 3.7 *Soient $i \geq 0$ et $n \geq 1$, alors $\mathtt{1} \cdot (s_{i\,n} \circ \uparrow) \twoheadrightarrow_\sigma s_{i+1\,n}$.*

Lemme 3.2 *Soit $b \in \Lambda$, $i \geq 0$ et $n \geq 1$, alors $b[s_{i\,n}] \twoheadrightarrow_\sigma U_i^n(b)$.*
En particulier, $b[\uparrow^n] \twoheadrightarrow_\sigma U_0^{n+1}(b)$.

Démonstration
Nous montrons le lemme par récurrence sur b.

$b = \mathtt{1}$: Deux possibilités:

$i = 0$: Si $n = 1$, alors $\mathtt{1}[s_{01}] = \mathtt{1}[id] \rightarrow \mathtt{1} = U_0^1(\mathtt{1})$.
Si $n > 1$, alors $\mathtt{1}[\uparrow^{n-1}] = \mathtt{n} = U_0^n(\mathtt{1})$.

$i \geq 1$: $\mathtt{1}[s_{i\,n}] = \mathtt{1} = U_i^n(\mathtt{1})$.

$b = c\,d$: $(c\,d)[s_{i\,n}] \rightarrow c[s_{i\,n}]\,d[s_{i\,n}] \overset{HR}{\twoheadrightarrow}_\sigma U_i^n(c)\,U_i^n(d) = U_i^n(c\,d)$.

$b = \lambda c$: $(\lambda c)[s_{i\,n}] \rightarrow \lambda(c[\mathtt{1} \cdot (s_{i\,n} \circ \uparrow)]) \overset{R3.7}{\twoheadrightarrow} \lambda(c[s_{i+1\,n}]) \overset{HR}{\twoheadrightarrow}_\sigma \lambda(U_{i+1}^n(c)) = U_i^n(\lambda c)$.

$b = \mathtt{k} > \mathtt{1} : \mathtt{k}[s_{i\,n}] = \mathtt{1}[\uparrow^{k-1}][s_{i\,n}] \to \mathtt{1}[\uparrow^{k-1} \circ s_{i\,n}]$. D'après la remarque 3.3 :

$$\uparrow^{k-1} \circ (\mathtt{1} \cdot \mathtt{2} \cdot \ldots \cdot \mathtt{i} \cdot \uparrow^{n+i-1}) \twoheadrightarrow_\sigma \begin{cases} \mathtt{k} \cdot \ldots \cdot \mathtt{i} \cdot \uparrow^{n+i-1} & \text{si} \quad k-1 < i \\ \uparrow^{n+k-2} & \text{si} \quad k-1 \geq i\,. \end{cases}$$

Donc,

$$\mathtt{1}[\uparrow^{k-1} \circ s_{i\,n}] \twoheadrightarrow_\sigma \begin{cases} \mathtt{k} & \text{si} \quad k \leq i \\ \mathtt{n+k-1} & \text{si} \quad k > i\,. \end{cases}$$

Le lemme est donc démontré car, par définition 1.24, on a :

$$U_i^n(\mathtt{k}) = \begin{cases} \mathtt{k} & \text{si} \quad k \leq i \\ \mathtt{n+k-1} & \text{si} \quad k > i\,. \end{cases}$$

□

Si l'on veut montrer $a[b \cdot id] \twoheadrightarrow_\sigma a\{\!\{\mathtt{1} \leftarrow b\}\!\}$ par récurrence sur a, quand on considère le cas $a = \lambda c$, à cause de la règle *(Abs)*, on se rend compte qu'une hypothèse de récurrence plus forte est nécessaire. La définition suivante est introduite pour gérer cette difficulté.

Définition 3.3 *A partir d'un terme $b \in \Lambda X_\sigma^t$ nous définissons une famille de substitutions $(b_k)_{k\geq 1}$ ainsi:*

$$\begin{array}{lcl} b_1 & = & b \cdot id \\ b_2 & = & \mathtt{1} \cdot b[\uparrow] \cdot \uparrow \\ \vdots & & \\ b_{k+1} & = & \mathtt{1} \cdot \mathtt{2} \cdot \ldots \cdot \mathtt{k} \cdot b[\uparrow^k] \cdot \uparrow^k\ . \end{array}$$

Les règles *(Map)*, *(Clos)* et *(Ass)* permettent de constater :

Remarque 3.8 *Soit $k \geq 1$, alors $\mathtt{1} \cdot (b_k \circ \uparrow) \twoheadrightarrow_\sigma b_{k+1}$.*

Lemme 3.3 *Soient $a, b \in \Lambda$ et $k \geq 1$, alors $a[b_k] \twoheadrightarrow_\sigma a\{\!\{\mathtt{k} \leftarrow b\}\!\}$.*
En particulier, $a[b \cdot id] \twoheadrightarrow_\sigma a\{\!\{\mathtt{1} \leftarrow b\}\!\}$.

Démonstration
On montre le lemme par récurrence sur $a \in \Lambda$.

$a = \mathtt{1}$: Si $k = 1$, alors $\mathtt{1}[b_1] = \mathtt{1}[b \cdot id] \to b \overset{R1.2}{=} U_0^1(b) = \mathtt{1}\{\!\{\mathtt{1} \leftarrow b\}\!\}$.
Si $k > 1$, alors $\mathtt{1}[b_k] = \mathtt{1} = \mathtt{1}\{\!\{\mathtt{k} \leftarrow b\}\!\}$.

$a = c\,d$: $(c\,d)[b_k] \to c\,[b_k]\,d\,[b_k] \overset{HR}{\twoheadrightarrow} c\,\{\!\{\mathtt{k} \leftarrow b\}\!\}\,d\,\{\!\{\mathtt{k} \leftarrow b\}\!\} = (c\,d)\{\!\{\mathtt{k} \leftarrow b\}\!\}$.

$a = \lambda c :\ (\lambda c)[b_k] \to \lambda(c\,[\mathtt{1}\cdot(b_k \circ \uparrow)]) \overset{R3.8}{\twoheadrightarrow} \lambda(c\,[b_{k+1}]) \overset{HR}{\twoheadrightarrow} \lambda(c\,\{\!\{\mathtt{k}+\mathtt{1} \leftarrow b\}\!\}) =$
$= (\lambda c)\{\!\{\mathtt{k} \leftarrow b\}\!\}\,.$

$a = \mathtt{m} > \mathtt{1} :\ \mathtt{m}\,[b_k] = \mathtt{1}[\uparrow^{m-1}][b_k] \to \mathtt{1}[\uparrow^{m-1} \circ b_k]$ .D'après le lemme 3.3 :

$$\uparrow^{m-1} \circ b_k = \uparrow^{m-1} \circ (\mathtt{1}\cdot\mathtt{2}\cdot\ldots\cdot\mathtt{k}-\mathtt{1}\cdot b\,[\uparrow^{k-1}]\cdot\uparrow^{k-1}) \twoheadrightarrow_\sigma$$

$$\twoheadrightarrow_\sigma \begin{cases} \mathtt{m}\cdot\ldots\cdot\mathtt{k}-\mathtt{1}\cdot b\,[\uparrow^{k-1}]\cdot\uparrow^{k-1} & \text{si} \quad m < k \\ b\,[\uparrow^{k-1}]\cdot\uparrow^{k-1} & \text{si} \quad m = k \\ \uparrow^{m-2} & \text{si} \quad m > k\,. \end{cases}$$

Donc,

$$\mathtt{1}[\uparrow^{m-1} \circ b_k] \twoheadrightarrow_\sigma \begin{cases} \mathtt{m} & \text{si} \quad m < k \\ b\,[\uparrow^{k-1}] & \text{si} \quad m = k \\ \mathtt{m}-\mathtt{1} & \text{si} \quad m > k\,. \end{cases}$$

Le lemme est donc démontré car, par définition 1.25, on a :

$$\mathtt{m}\{\!\{\mathtt{k} \leftarrow b\}\!\} = \begin{cases} \mathtt{m}-\mathtt{1} & \text{si} \quad m > k \\ U_0^k(b) & \text{si} \quad m = k \\ \mathtt{m} & \text{si} \quad m < k \end{cases}$$

et, d'après le lemme précédent, $b[\uparrow^{k-1}] \twoheadrightarrow_\sigma U_0^k(b)$.

□

Corollaire 3.1 *Soient* $a, b \in \Lambda$.

1. $\sigma'(a[b\cdot id]) = \sigma(a[b\cdot id]) = a\{\!\{\mathtt{1} \leftarrow b\}\!\}$.
2. $a \to_{\beta'} b$ *ssi* $a \to_\beta b$.

Démonstration

1. Le terme $a\{\!\{\mathtt{1} \leftarrow b\}\!\}$ appartient à Λ , étant ainsi en σ-f.n. et en σ'-f.n..
2. Par récurrence sur la structure de a. Si la réduction est interne, l'H.R. permet de conclure. Quand la réduction a lieu à la racine, on utilise l'item précédent.

□

3.3 Interprétation de *(Beta)*

L'objectif de cette section est de montrer (cf. proposition 3.3) que pour tous $f, g \in \Lambda X$ on a :

$$\text{Si } f \to_{(Beta)} g \text{ , alors } \sigma'(f) \twoheadrightarrow_{\beta'} \sigma'(g) .$$

Ceci est l'une des conditions du lemme d'interprétation. Si la réduction a lieu à la racine, on conclut facilement, tandis que si elle est interne, pour conclure nous avons besoin d'une série de lemmes qui garantissent le passage au contexte des β'-dérivations.

Nous commençons par étudier ce passage quand la β'-dérivation est constituée d'un seul pas de réduction.

Remarque 3.9 *Soient* $a, b, c \in \Lambda X^t_{\sigma'}$ *et* $s, t, u \in \Lambda X^s_{\sigma'}$.

1. *Si* $a \to_{\beta'} b$, *alors* $a\,c \to_{\beta'} b\,c$.
2. *Si* $a \to_{\beta'} b$, *alors* $c\,a \to_{\beta'} c\,b$.
3. *Si* $a \to_{\beta'} b$, *alors* $\lambda a \to_{\beta'} \lambda b$.
4. *Si* $s \to_{\beta'} t$, *alors* $S(X[s]) \to_{\beta'} S(X[t])$.
5. *Si* $a \to_{\beta'} b$, *alors* $S(a \cdot s) \to_{\beta'} S(b \cdot s)$.
6. *Si* $s \to_{\beta'} t$, *alors* $S(a \cdot s) \to_{\beta'} S(a \cdot t)$.

Démonstration
Les trois premiers sont presque immédiats. Nous montrons, par exemple, le premier énoncé.
Il existe $d \in \Lambda X^t$ tel que $a \to_{(Beta)} d$ et $b = \sigma'(d)$. Alors $ac \to_{(Beta)} dc$ et $\sigma'(dc) = \sigma'(d)\sigma'(c) = b\,c$. Donc, $ac \to_{\beta'} b\,c$.
Pour montrer les trois derniers, il faut remarquer d'abord que la réduction qu'on a par hypothèse assure que S est superflue dans le terme de départ de la β'-réduction à démontrer. Nous avons choisi cette formulation (avec S) pour des raisons techniques qui deviendront claires dans la suite.
Montrons, par exemple, le quatrième.
Il existe $u \in \Lambda X^s$ telle que $s \to_{(Beta)} u$ et $t = \sigma'(u)$. Donc, $s \neq id$ (autrement il n'y a pas de *(Beta)*-radical). Alors

$$S(X[s]) = X[s] \to_{(Beta)} X[u] \quad \text{et} \quad \sigma'(X[u]) = S(X[\sigma'(u)]) = S(X[t]) ,$$

i.e. $S(X[s]) \to_{\beta'} S(X[t])$.
Pour le cinquième, $a \to_{\beta'} b$, garantit $a \neq \mathtt{m}$. Donc $S(a \cdot s) = a \cdot s$, et le raisonnement se poursuit comme dans le cas précédent.

Pour le sixième, $s \to_{\beta'} t$, garantit $s \neq \uparrow^m$. Donc $S(a \cdot s) = a \cdot s$, etc..
□

Lemme 3.4 *Soient* $a \in \Lambda X^t_{\sigma'}$, $t \in \Lambda X^s_{\sigma'}$, $b \in \Lambda X^t$ *et* $s, u \in \Lambda X^s$.

1. *Si* $a \to_{(Beta)} b$, *alors* $\sigma'(a[s]) \to_{\beta'} \sigma'(b[s])$.
2. *Si* $t \to_{(Beta)} u$, *alors* $\sigma'(t \circ s) \to_{\beta'} \sigma'(u \circ s)$.

Démonstration
On montre les deux résultats par récurrence simultanée sur a et t. On n'étudie que les cas où il y a des radicaux.

$a = cd$: Deux sous-cas selon l'emplacement du radical :

Interne : Par exemple, $c \to_{(Beta)} e$ et $b = ed$.
Par H.R. on a $\sigma'(c[s]) \to_{\beta'} \sigma'(e[s])$. Donc,

$$\sigma'((cd)[s]) = \sigma'(c[s])\sigma'(d[s]) \overset{R.3.9.1}{\to_{\beta'}} \sigma'(e[s])\sigma'(d[s]) = \sigma'(ed[s]).$$

A la racine : Alors $c = \lambda e$ et $b = e[d \cdot id]$.

$$\sigma'(((\lambda e)d)[s]) = (\lambda\sigma'(e[\mathtt{1} \cdot (s \circ \uparrow)]))\sigma'(d[s]) \to_{\beta'}$$

$$\to_{\beta'} \sigma'(\sigma'(e[\mathtt{1} \cdot (s \circ \uparrow)])[\sigma'(d[s]) \cdot id]) = \sigma'(e[\mathtt{1} \cdot (s \circ \uparrow)][d[s] \cdot id]) =$$
$$= \sigma'(e[d[s] \cdot (s \circ id)]) = \sigma'(e[d[s] \cdot s]) = \sigma'(e[d \cdot id][s]).$$

$a = \lambda c$: Alors $b = \lambda d$ où $c \to_{(Beta)} d$.

$$\sigma'((\lambda c)[s]) = \lambda\sigma'(c[\mathtt{1} \cdot (s \circ \uparrow)]) \overset{HR\&R3.9.3}{\to_{\beta'}} \lambda\sigma'(d[\mathtt{1} \cdot (s \circ \uparrow)]) = \sigma'((\lambda d)[s]).$$

$a = X[v]$: Alors $b = X[w]$ où $v \to_{(Beta)} w$.

$$\sigma'(X[v][s]) = S(X[\sigma'(v \circ s)]) \overset{HR\&R3.9.4}{\to_{\beta'}} S(X[\sigma'(w \circ s)]) = \sigma'(X[w][s]).$$

$t = c \cdot v$: Par exemple, $c \to_{(Beta)} d$ et $u = d \cdot v$.

$$\sigma'((c \cdot v)[s]) = S(\sigma'(c[s]) \cdot \sigma'(v[s])) \overset{HR\&R3.9.5}{\to_{\beta'}}$$
$$\to_{\beta'} S(\sigma'(d[s]) \cdot \sigma'(v[s])) = \sigma'((d \cdot v)[s]).$$

□

Corollaire 3.2 *Soient* $a, b \in \Lambda X^t_{\sigma'}$, $t, u \in \Lambda X^s_{\sigma'}$ *et* $s \in \Lambda X^s$.

1. *Si* $a \to_{\beta'} b$, *alors* $\sigma'(a[s]) \to_{\beta'} \sigma'(b[s])$.

2. *Si* $t \to_{\beta'} u$, *alors* $\sigma'(t \circ s) \to_{\beta'} \sigma'(u \circ s)$.

3. *Si* $a \twoheadrightarrow_{\beta'} b$, *alors* $\sigma'(a[s]) \twoheadrightarrow_{\beta'} \sigma'(b[s])$.

4. *Si* $t \twoheadrightarrow_{\beta'} u$, *alors* $\sigma'(t \circ s) \twoheadrightarrow_{\beta'} \sigma'(u \circ s)$.

Démonstration
Montrons le premier résultat. Par hypothèse, il existe $c \in \Lambda X^t$ tel que $a \to_{(Beta)} c$ et $\sigma'(c) = b$. Et d'après le lemme 3.4.1 on a $\sigma'(a[s]) \to_{\beta'} \sigma'(c[s]) = \sigma'(b[s])$.
La preuve du deuxième est analogue.

Les troisième et quatrième résultats sont montrés par récurrence sur la longueur des dérivations $a \twoheadrightarrow_{\beta'} b$ et $t \twoheadrightarrow_{\beta'} u$, respectivement, et en utilisant les items 1 et 2.
□

Lemme 3.5 *Soient* $a, b, c, d \in \Lambda X^t_{\sigma'}$ *et* $s, t \in \Lambda X^t_{\sigma'}$.

1. *Si* $a \twoheadrightarrow_{\beta'} b$ *et* $c \twoheadrightarrow_{\beta'} d$, *alors* $ac \twoheadrightarrow_{\beta'} bd$.

2. *Si* $a \twoheadrightarrow_{\beta'} b$ *et* $s \twoheadrightarrow_{\beta'} t$, *alors* $S(a \cdot s) \twoheadrightarrow_{\beta'} S(b \cdot t)$.

3. *Si* $a \twoheadrightarrow_{\beta'} b$, *alors* $\lambda a \twoheadrightarrow_{\beta'} \lambda b$.

4. *Si* $s \twoheadrightarrow_{\beta'} t$, *alors* $S(X[s]) \twoheadrightarrow_{\beta'} S(X[t])$.

Démonstration

1. Récurrence sur la longueur de la dérivation $c \twoheadrightarrow_{\beta'} d$. Si elle est nulle, il faut montrer $ac \twoheadrightarrow_{\beta'} bc$. Ceci se fait par récurrence sur la longueur de la dérivation $a \twoheadrightarrow_{\beta'} b$ en utilisant la remarque 3.9.1.

   Supposons donc $c \twoheadrightarrow_{\beta'} e \to_{\beta'} d$. Par H.R. $ac \twoheadrightarrow_{\beta'} be$ et la remarque 3.9.2 permet de conclure.

2. Analogue à la preuve précédente. Utiliser maintenant les remarques 3.9.5 et 3.9.6.

3. Récurrence sur la longueur de la dérivation. Utiliser la remarque 3.9.3.

4. Récurrence sur la longueur de la dérivation. Utiliser la remarque 3.9.4.

□

Lemme 3.6 *Soient* $a \in \Lambda X^t_{\sigma'}$, $s, u \in \Lambda X^s_{\sigma'}$ *et* $t \in \Lambda X^s$. *Si* $s \to_{(Beta)} t$, *alors* $\sigma'(a[s]) \twoheadrightarrow_{\beta'} \sigma'(a[t])$ *et* $\sigma'(u \circ s) \twoheadrightarrow_{\beta'} \sigma'(u \circ t)$.

Démonstration
Récurrence simultanée sur a et u.
Puisque $s \to_{(Beta)} t$, il existe $k \geq 1$ tel que $s = a_1 \cdot \ldots \cdot a_i \cdot \ldots \cdot a_k \cdot \uparrow^n$ et $t = a_1 \cdot \ldots \cdot a_i{}' \cdot \ldots \cdot a_k \cdot \uparrow^n$, où $a_i \to_{(Beta)} a_i{}'$.

$a = bc$: $\sigma'((bc)[s]) = \sigma'(b[s])\sigma'(c[s]) \overset{H.R.\ \&\ L.3.5.1}{\twoheadrightarrow_{\beta'}} \sigma'(b[t])\sigma'(c[t]) = \sigma'((bc)[t])$.

$a = \lambda b$: Puisque $s \to_{(Beta)} t$, le lemme 3.4.2 assure $\sigma'(s \circ \uparrow) \to_{\beta'} \sigma'(t \circ \uparrow)$, i.e. il existe $v \in \Lambda X^s$ telle que $\sigma'(s \circ \uparrow) \to_{(Beta)} v$ et $\sigma'(v) = \sigma'(t \circ \uparrow)$. D'où, $\mathtt{1} \cdot \sigma'(s \circ \uparrow) \to_{(Beta)} \mathtt{1} \cdot v$. En plus, comme $\sigma'(s \circ \uparrow)$ contient un *(Beta)*-radical, $\sigma'(s \circ \uparrow) \neq \uparrow$, et alors $\mathtt{1} \cdot \sigma'(s \circ \uparrow)$ est en σ'-forme normale. On peut donc appliquer l'H.R. pour obtenir :

$$\sigma'(b[\mathtt{1} \cdot \sigma'(s \circ \uparrow)]) \twoheadrightarrow_{\beta'} \sigma'(b[\mathtt{1} \cdot v]) = \sigma'(b[\mathtt{1} \cdot \sigma'(v)]) = \sigma'(b[\mathtt{1} \cdot \sigma'(t \circ \uparrow)]).$$

Donc,

$$\sigma'((\lambda b)[s]) = \lambda\sigma'(b[\mathtt{1} \cdot \sigma'(s \circ \uparrow)]) \overset{L3.5.3}{\twoheadrightarrow_{\beta'}} \lambda\sigma'(b[\mathtt{1} \cdot \sigma'(t \circ \uparrow)]) = \sigma'((\lambda b)[t]).$$

$a = \mathtt{m}$: Si $m \leq k$ et $m \neq i$, alors $\sigma'(\mathtt{m}[s]) = a_m = \sigma'(\mathtt{m}[t])$.
Si $m = i$, alors $\sigma'(\mathtt{m}[s]) = a_i \to_{\beta'} \sigma'(a'_i) = \sigma'(\mathtt{m}[t])$.
Si $m \geq k$, alors $\sigma'(\mathtt{m}[s]) = \mathtt{m} + \mathtt{n} - \mathtt{k} = \sigma'(\mathtt{m}[t])$.

$a = X$: Puisque $s \to_{(Beta)} t$, on a $s \to_{\beta'} \sigma'(t)$.
Donc (cf. remarque 3.9.4), $S(X[s]) \to_{\beta'} S(X[\sigma'(t)])$. Alors

$$\sigma'(X[s]) = S(X[s]) \to_{\beta'} S(X[\sigma'(t)]) = \sigma'(X[t]).$$

$a = X[v]$: Par H.R. $\sigma'(v \circ s) \twoheadrightarrow_{\beta'} \sigma'(v \circ t)$.

$$\sigma'(X[v][s]) = S(X[\sigma'(v \circ s)]) \overset{L.3.5.4}{\twoheadrightarrow_{\beta'}} S(X[\sigma'(v \circ t)]) = \sigma'(X[v][t]).$$

$u = id$: $\sigma'(id \circ s) = s \to_{\beta'} \sigma'(t) = \sigma'(id \circ t)$.

$u = \uparrow^m$: D'après la remarque 3.3 :

$$\uparrow^m \circ s = \uparrow^m \circ (a_1 \cdot \ldots \cdot a_i \cdot \ldots \cdot a_k \cdot \uparrow^n) \twoheadrightarrow_\sigma \begin{cases} a_{m+1} \cdot \ldots \cdot a_k \cdot \uparrow^n & \text{si} \quad m < k \\ \uparrow^{n+m-k} & \text{si} \quad m \geq k. \end{cases}$$

Un calcul analogue pour $\uparrow^m \circ t$ nous permet de conclure :

$m \leq i - 1$: $\sigma'(\uparrow^m \circ s) = a_{m+1} \cdot \ldots \cdot a_i \cdot \ldots \cdot a_k \cdot \uparrow^n \to_{(Beta)}$
$\to_{(Beta)} a_{m+1} \cdot \ldots \cdot a'_i \cdot \ldots \cdot a_k \cdot \uparrow^n \twoheadrightarrow_{\sigma'} \sigma'(\uparrow^m \circ t)$.
Donc, $\sigma'(\uparrow^m \circ s) \to_{\beta'} \sigma'(\uparrow^m \circ t)$.

$m > i-1$: $\sigma'(\uparrow^m \circ s) = \sigma'(\uparrow^m \circ t)$.

$u = c \cdot v$: $\sigma'((c \cdot v) \circ s) = S(\sigma'(c[s]) \cdot \sigma'(v \circ s)) \overset{HR\&L.3.5.2}{\twoheadrightarrow_{\beta'}} S(\sigma'(c[t]) \cdot \sigma'(v \circ t)) = \sigma'((c \cdot v) \circ t)$.

□

Corollaire 3.3 *Soient* $a \in \Lambda X^t_{\sigma'}$ *et* $s, t, u \in \Lambda X^s_{\sigma'}$.

1. *Si* $s \to_{\beta'} t$, *alors* $\sigma'(a[s]) \twoheadrightarrow_{\beta'} \sigma'(a[t])$ *et* $\sigma'(u \circ s) \twoheadrightarrow_{\beta'} \sigma'(u \circ t)$.
2. *Si* $s \twoheadrightarrow_{\beta'} t$, *alors* $\sigma'(a[s]) \twoheadrightarrow_{\beta'} \sigma'(a[t])$ *et* $\sigma'(u \circ s) \twoheadrightarrow_{\beta'} \sigma'(u \circ t)$.

Démonstration
Le premier résultat est une conséquence immédiate du lemme précédent. On montre le deuxième par récurrence sur la longueur de la réduction et en utilisant le premier.
□

Nous avons maintenant les outils nécessaires pour simuler la règle *(Beta)* sur ΛX par la dérivation $\twoheadrightarrow_{\beta'}$ sur les σ'-formes normales.

Proposition 3.3 *Soient* $f, g \in \Lambda X$ *tels que* $f \to_{(Beta)} g$, *alors* $\sigma'(f) \twoheadrightarrow_{\beta'} \sigma'(g)$.

Démonstration
Récurrence sur la structure de f.

$f = a\,b$: On considère l'occurrence du radical :

$a \to_{(Beta)} c$: Alors $g = c\,b$.

$$\sigma'(a\,b) = \sigma'(a)\,\sigma'(b) \overset{HR\&L3.5.1}{\twoheadrightarrow_{\beta'}} \sigma'(c)\,\sigma'(b) = \sigma'(c\,b).$$

$b \to_{(Beta)} d$: Analogue au sous-cas précédent.

A la racine : Alors $a = \lambda c$ et $g = c\,[b \cdot id]$.

$$\sigma'((\lambda c)\,b) = (\lambda\sigma'(c))\,\sigma'(b) \to_{\beta'} \sigma'(\sigma'(c)[\sigma'(b) \cdot id]) = \sigma'(c[b \cdot id]).$$

$f = \lambda a$: Alors $a \to_{(Beta)} b$ et $g = \lambda b$.

$$\sigma'(\lambda a) = \lambda\sigma'(a) \overset{HR\&L3.5.3}{\twoheadrightarrow_{\beta'}} \lambda\sigma'(b) = \sigma'(\lambda b).$$

$f = a\,[s]$: Considérons l'occurrence du radical :

$a \to_{(Beta)} b$: Alors $g = b\,[s]$.

$$\sigma'(a\,[s]) = \sigma'(\sigma'(a)[\sigma'(s)]) \overset{HR\&C3.2.3}{\twoheadrightarrow_{\beta'}} \sigma'(\sigma'(b)[\sigma'(s)]) = \sigma'(b\,[s]).$$

$s \to_{(Beta)} t$: Alors $g = a\,[t]$.

$$\sigma'(a\,[s]) = \sigma'(\sigma'(a)[\sigma'(s)]) \overset{HR\&C3.3.2}{\twoheadrightarrow_{\beta'}} \sigma'(\sigma'(a)[\sigma'(t)]) = \sigma'(a\,[t])\,.$$

$f = a \cdot s$: Selon l'occurrence du radical :

$a \to_{(Beta)} b$: Alors $g = b \cdot s$.

$$\sigma'(a \cdot s) = S(\sigma'(a) \cdot \sigma'(s)) \overset{HR\&L3.5.2}{\twoheadrightarrow_{\beta'}} S(\sigma'(b) \cdot \sigma'(s)) = \sigma'(b \cdot s)\,.$$

$s \to_{(Beta)} t$: Alors $g = a \cdot t$.

$$\sigma'(a \cdot s) = S(\sigma'(a) \cdot \sigma'(s)) \overset{HR\&L3.5.2}{\twoheadrightarrow_{\beta'}} S(\sigma'(a) \cdot \sigma'(t)) = \sigma'(a \cdot t)\,.$$

$f = s \circ t$: Selon l'occurrence du radical :

$s \to_{(Beta)} u$: Alors $g = u \circ t$.

$$\sigma'(s \circ t) = \sigma'(\sigma'(s) \circ \sigma'(t)) \overset{HR\&C3.2.4}{\twoheadrightarrow_{\beta'}} \sigma'(\sigma'(u) \circ \sigma'(t)) = \sigma'(u \circ t)\,.$$

$t \to_{(Beta)} v$: Alors $g = s \circ v$.

$$\sigma'(s \circ t) = \sigma'(\sigma'(s) \circ \sigma'(t)) \overset{HR\&C3.3.2}{\twoheadrightarrow_{\beta'}} \sigma'(\sigma(s) \circ \sigma'(v)) = \sigma'(s \circ v)\,.$$

□

Corollaire 3.4 *Soient $f, g \in \Lambda X$ tels que $f \twoheadrightarrow_{\lambda\sigma'} g$, alors $\sigma'(f) \twoheadrightarrow_{\beta'} \sigma'(g)$.*

Démonstration
Par récurrence sur la longueur de la dérivation. Si elle est nulle, le résultat est immédiat. Supposons donc $f \twoheadrightarrow_{\lambda\sigma'} h \to_{\lambda\sigma'} g$, et étudions la dernière règle. Si elle est une σ'-règle, $\sigma'(h) = \sigma'(g)$. Si elle est la règle *(Beta)*, la proposition précédente assure $\sigma'(h) \twoheadrightarrow_{\beta'} \sigma'(g)$.
□

Avant de finir cette section, nous voulons remarquer que la proposition 3.3 n'est pas vraie pour les termes ouverts. En effet, voici un contre-exemple, où x est une variable de type substitution :
Soit $f = (\mathtt{1}[x \circ (((\lambda\mathtt{1})\mathtt{1}) \cdot id)]) \cdot (\uparrow \circ (x \circ (((\lambda\mathtt{1})\mathtt{1}) \cdot id)))$, alors

$$f \to_{(Beta)} (\mathtt{1}[x \circ ((\mathtt{1}[\mathtt{1} \cdot id]) \cdot id)]) \cdot (\uparrow \circ (x \circ (((\lambda\mathtt{1})\mathtt{1}) \cdot id))) =_{\mathrm{def}} g\,.$$

D'autre part, $\sigma'(f) = x \circ (((\lambda\mathtt{1})\mathtt{1}) \cdot id)$, et comme $\sigma'(f)$ contient un seul *(Beta)*-radical, la seule β'-réduction possible est la suivante :

$$\sigma'(f) \to_{\beta'} x \circ (\mathtt{1} \cdot id) \neq \sigma'(g)\,.$$

Remarquons aussi que ce contre-exemple l'est aussi pour la remarque 3.9.5 en prenant $a = \mathtt{1}[x \circ (((\lambda\mathtt{1})\mathtt{1}) \cdot id)]$, $b = \mathtt{1}[x \circ ((\mathtt{1}[\mathtt{1} \cdot id]) \cdot id)]$ et $s = \uparrow \circ (x \circ (((\lambda\mathtt{1})\mathtt{1}) \cdot id))$.

La confluence peut être établie pour les termes semi-clos, car les σ'-f.n. sont très simples par rapport à celles des termes ouverts. En fait, on peut montrer de manière analogue à la preuve de la proposition 3.1, le résultat suivant :

Remarque 3.10 *Les termes et substitutions en σ-forme normale dans $\Lambda\varsigma$ sont donnés récursivement par :*

Termes	$a ::= \mathtt{1} \mid ab \mid \lambda a \mid X \mid X[s] \mid \mathtt{1}[s]\,(s \neq id \wedge s \neq a \cdot t)$
Substitutions	$s ::= id \mid \uparrow \mid a \cdot s \mid x \mid x \circ s \mid \uparrow \circ s\,(s \neq id \wedge s \neq a \cdot t)$

Les termes et substitutions en σ'-forme normale dans $\Lambda\varsigma$ sont donnés récursivement par :

T.	$a ::= \mathtt{1} \mid ab \mid \lambda a \mid X \mid X[s]\,(s \neq id) \mid \mathtt{1}[s]\,(s \neq id \wedge s \neq a \cdot t)$
S.	$s ::= id \mid \uparrow \mid a \cdot s\,(\neg\exists t : a = \mathtt{1}[t] \wedge s = \uparrow \circ t) \mid x \mid x \circ s\,(s \neq id) \mid$ $\uparrow \circ s\,(s \neq id \wedge s \neq a \cdot t)$

La différence radicale est l'apparition des compositions $x \circ s$ et $\uparrow \circ s$ dans les formes normales des substitutions, tandis que dans le cas semi-clos les seules compositions qui interviennent sont de la forme $\uparrow^n$. Ceci limite la création de *(SCons)*-radicaux à ceux de la forme $\mathtt{n} \cdot \uparrow^n$, ce qui n'interfère pas avec la *(Beta)*-réduction. Le contre-exemple que nous venons de donner se fonde justement sur la possibilité de détruire un *(SCons)*-radical par une *(Beta)*-réduction, ce qui est impossible dans le cas semi-clos.

3.4 La parallélisation

Nous utilisons la méthode de Tait et Martin-Löf pour montrer la confluence du λ'-calcul, i.e. nous définissons une parallélisation de $\rightarrow_{\beta'}$, notée $\Rightarrow$, dont nous montrons la confluence forte dans la section suivante.

Cette section est consacrée à introduire la parallélisation et à montrer que sa clôture reflexive transitive coïncide avec $\twoheadrightarrow_{\beta'}$, de sorte que, dès que la confluence forte de $\Rightarrow$ sera établie, la confluence de $\rightarrow_{\beta'}$ s'ensuivra.

Nous commençons par définir la parallélisation et remarquons ensuite qu'une définition équivalente sera plus facile à manipuler.

Définition 3.4 *Soient $a, b, c, d \in \Lambda X^t_{\sigma'}$ et $s, t \in \Lambda X^s_{\sigma'}$. La réduction $f \Rightarrow g$ est*

définie sur $\Lambda X_{\sigma'}$ *par les règles suivantes:*

$$(REFL)\quad f \Rightarrow f \qquad (Clos)\quad \frac{s \Rightarrow t}{X[s] \Rightarrow S(X[t])}$$

$$(ABST)\quad \frac{a \Rightarrow b}{\lambda a \Rightarrow \lambda b} \qquad (Cons)\quad \frac{a \Rightarrow b \quad s \Rightarrow t}{a \cdot s \Rightarrow S(b \cdot t)}$$

$$(APPL)\quad \frac{a \Rightarrow c \quad b \Rightarrow d}{a\, b \Rightarrow c\, d} \qquad (BETA)\quad \frac{a \Rightarrow c \quad b \Rightarrow d}{(\lambda a)\, b \Rightarrow \sigma'(a[b \cdot id])}$$

On sous-entend que les règles (Clos) et (Cons) peuvent être appliquées seulement quand $X[s] \in \Lambda X^t_{\sigma'}$ *et* $a \cdot s \in \Lambda X^s_{\sigma'}$ *, respectivement; i.e.*

1. *(Clos) peut être appliquée ssi* $s \neq id$ *.*
2. *(Cons) peut être appliquée ssi* $(\neg\exists n \geq 1)\,(a = \mathtt{n} \ \wedge \ s = \uparrow^n)$ *.*

Evidemment, il est plus commode d'avoir une définition où il n'y ait pas de conditions à vérifier pour pouvoir appliquer les règles *(Clos)* et *(Cons)*.

Lemme 3.7 *La réduction qu'on obtient en remplaçant les règles (Clos) et (Cons) par les règles*

$$(CLOS)\quad \frac{s \Rightarrow t}{S(X[s]) \Rightarrow S(X[t])} \qquad (CONS)\quad \frac{a \Rightarrow b \quad s \Rightarrow t}{S(a \cdot s) \Rightarrow S(b \cdot t)}$$

respectivement, et sans aucune condition pour l'application de ces règles, est équivalente à la réduction $\Rightarrow$ *, introduite dans la définition 3.4.*

Démonstration
Il suffit de constater que id, $\mathtt{n}$ et $\uparrow^n$ n'ont qu'un seul $\Rightarrow$-réduit, à savoir : id, $\mathtt{n}$ et $\uparrow^n$, respectivement. On peut donc utiliser la remarque 3.5 pour vérifier que dans les cas critiques *(CLOS)* et *(CONS)* fonctionnent comme *(REFL)*.
□

Dans le reste de ce chapitre, nous utiliserons toujours cette dernière version de la parallélisation.

Nous finissons cette section en montrant que les clôtures réflexives transitives de $\Rightarrow$ et de $\rightarrow_{\beta'}$ coïncident.

Lemme 3.8 *Soient* $f, g \in \Lambda X_{\sigma'}$ *tels que* $f \Rightarrow g$ *, alors* $f \twoheadrightarrow_{\beta'} g$ *.*

Démonstration
Récurrence sur la dérivation de $f \Rightarrow g$. On étudie donc la dernière règle utilisée.

(REFL) : Immédiat.

(APPL) : On utilise l' H.R. et le lemme 3.5.1.

(BETA) : Alors $f = (\lambda a)b$ et $g = \sigma'(c[d \cdot id])$ où $a \Rightarrow c$ et $b \Rightarrow d$.
Par H.R. $a \twoheadrightarrow_{\beta'} c$ et $b \twoheadrightarrow_{\beta'} d$. Signalons que, d'après la remarque 3.5.3, $b \in \Lambda X^t_{\sigma'}$, entraîne $b \cdot id \in \Lambda X^t_{\sigma'}$.
D'où, le lemme 3.5.2 garantit $b \cdot id \twoheadrightarrow_{\beta'} d \cdot id$. Donc,

$$(\lambda a)b \to_{\beta'} \sigma'(a[b \cdot id]) \overset{C3.2.3}{\twoheadrightarrow_{\beta'}} \sigma'(c[b \cdot id]) \overset{C3.3.2}{\twoheadrightarrow_{\beta'}} \sigma'(c[d \cdot id]).$$

(ABST) : Utiliser H.R. et lemme 3.5.3.

(CLOS) : Utiliser H.R. et lemme 3.5.4.

(CONS) : Utiliser H.R. et lemme 3.5.2.

□

Proposition 3.4 *Les clôtures réflexives transitives de* $\to_{\beta'}$ *et de sa parallélisation coïncident sur* $\Lambda X_{\sigma'}$, *i.e.* $\twoheadrightarrow_{\beta'} = \Rightarrow^*$.

Démonstration
On montre sans difficulté par récurrence sur f que si $f \to_{(Beta)} g$ alors $f \Rightarrow \sigma'(g)$.
En effet, la règle *(BETA)* permet de traiter les réductions à la racine, tandis que les réductions internes sont gérées par les autres règles.
Ceci permet de conclure $\to_{\beta'} \subseteq \Rightarrow$.
Le lemme précédent assure $\Rightarrow \subseteq \twoheadrightarrow_{\beta'}$.
Alors $\to_{\beta'} \subseteq \Rightarrow \subseteq \twoheadrightarrow_{\beta'}$. Donc, $\twoheadrightarrow_{\beta'} = \Rightarrow^*$.
□

3.5 Le lemme de substitution pour $\Rightarrow$

Pour montrer la confluence de la parallélisation nous avons besoin d'un lemme de substitution (cf. corollaire 3.5). Pour établir ce lemme nous montrons une série de résultats préliminaires sur $\Rightarrow$ du même genre de ceux que nous avons obtenu pour $\twoheadrightarrow_{\beta'}$ dans la section 3.3. Malheureusement, l'égalité des clôtures reflexives transitives montrée dans la section précédente et les anciens résultats sur $\twoheadrightarrow_{\beta'}$ ne fournissent pas les résultats désirés pour $\Rightarrow$, car nous envisageons la confluence *forte* de $\Rightarrow$, et par conséquent, c'est un lemme de substitution pour $\Rightarrow$, et pas pour $\Rightarrow^*$, dont nous avons besoin.

Si $s \in \Lambda X^s_{\sigma'}$, la remarque 3.2 explicite sa structure ainsi : $s = a_1 \cdot \ldots \cdot a_k \cdot \uparrow^n$.
Nous avons appelé k la *complexité* de s et n sa *hauteur*.

Le lemme suivant dit que la complexité de tout $\Rightarrow$-réduit de s n'augmente pas par rapport à la complexité de s. En plus, si la complexité du réduit a diminué de p, alors la hauteur aussi.

Lemme 3.9 *Soient* $k, n \geq 0$ *et* $s, t \in \Lambda X^s_{\sigma'}$ *tels que* $s = a_1 \cdot \ldots \cdot a_k \cdot \uparrow^n$ *et* $s \Rightarrow t$. *Alors il existe* $q \geq 0$ *tel que* $q \leq k$ *et* $b_1, \ldots, b_{k-q} \in \Lambda X^t_{\sigma'}$ *tels que*

- $t = b_1 \cdot \ldots \cdot b_{k-q} \cdot \uparrow^{n-q}$.
- $a_{k-q+j} \Rightarrow \mathtt{n} - \mathtt{q} + \mathtt{j}$ *pour* $1 \leq j \leq q$.
- $a_i \Rightarrow b_i$ *pour* $1 \leq i \leq k-q$.

Pour le cas limite $q = k$, *le premier item doit être lu* $t = \uparrow^{n-q}$, *et le troisième doit être omis.*

Démonstration
Par récurrence sur la longueur de l'inférence $s \Rightarrow t$. Remarquons que si $k = 0$, nécessairement $t = s$, et il suffit de prendre $q = 0$.
Supposons donc $k \geq 1$. Si la dernière règle est *(REFL)* on prend $q = 0$ et $b_i = a_i$ pour $1 \leq i \leq k$.
Si la dernière règle est *(CONS)*, puisque $S(a_1 \cdot \ldots \cdot a_k \cdot \uparrow^n) = a_1 \cdot \ldots \cdot a_k \cdot \uparrow^n$, car $s \in \Lambda X^s_{\sigma'}$, on a :

$$\frac{a_1 \Rightarrow b_1 \quad a_2 \cdot \ldots \cdot a_k \cdot \uparrow^n \Rightarrow u}{a_1 \cdot a_2 \cdot \ldots \cdot a_k \cdot \uparrow^n \Rightarrow S(b_1 \cdot u)}.$$

Dans ce cas on peut appliquer l'H.R. pour obtenir $q' \leq k-1$ et $b_2, \ldots b_{k-q'} \in \Lambda X^t_{\sigma'}$ tels que $u = b_2 \cdot \ldots \cdot b_{k-q'} \cdot \uparrow^{n-q'}$.

- Si $k - q' \geq 2$, alors $t = S(b_1 \cdot u) = b_1 \cdot u$ et il suffit de prendre $q = q'$.
- Si $k - q' = 1$, alors $u = \uparrow^{n-q'}$, et on étudie la valeur de b_1 :
  - Si $b_1 \neq n - q'$, on prend $q = q'$.
  - Si $b_1 = n - q'$, on prend $q = q' + 1 = k$.

□

Par récurrence sur n on montre sans dificulté :

Remarque 3.11 *Soient* $n \geq 0$ *et* $m \geq 0$, *alors*

$$\mathtt{m+1} \cdot \ldots \cdot \mathtt{m+n} \cdot \uparrow^{m+n} \twoheadrightarrow_{SPID} \uparrow^m .$$

En particulier, $s_{i1} \twoheadrightarrow_{SPID} id$.

Le lemme suivant est un cas particulier du lemme 3.12. Or, il faut d'abord établir ce cas particulier pour prouver le cas général.

Lemme 3.10 *Soient* $a, b \in \Lambda X^t_{\sigma'}$, $s, t \in \Lambda X^s_{\sigma'}$, $i \geq 0$ *et* $n \geq 1$. *Alors*

1. *Si* $a \Rightarrow b$, *alors* $\sigma'(a[s_{in}]) \Rightarrow \sigma'(b[s_{in}])$.

2. Si $s \Rightarrow t$, *alors* $\sigma'(s \circ s_{in}) \Rightarrow \sigma'(t \circ s_{in})$.

Démonstration
On montre les deux résultats par récurrence simultanée sur les longueurs des inférences $a \Rightarrow b$ et $s \Rightarrow t$.
Analysons donc la dernière règle appliquée :

(REFL) : Immédiat.

(ABST) : Alors $a = \lambda c$, $b = \lambda d$ et $c \Rightarrow d$.

$$\sigma'((\lambda c)[s_{in}]) = \sigma'(\lambda(c[\mathtt{1} \cdot (s_{in} \circ \uparrow)])) = \lambda(\sigma'(c[\mathtt{1} \cdot (s_{in} \circ \uparrow)])) \overset{R3.7}{=}$$

$$= \lambda(\sigma'(c[s_{i+1\,n}])) \overset{HR}{\Rightarrow} \lambda(\sigma'(d[s_{i+1\,n}])) = \lambda(\sigma'(d[\mathtt{1} \cdot (s_{in} \circ \uparrow)])) =$$

$$= \sigma'(\lambda(d[\mathtt{1} \cdot (s_{in} \circ \uparrow)])) = \sigma'((\lambda d)[s_{in}]).$$

(APPL) : Alors $a = cd$ et $b = c'\,d'$, où $c \Rightarrow c'$ et $d \Rightarrow d'$.

$$\sigma'((cd)[s_{in}]) = \sigma'(c[s_{in}])\sigma'(d[s_{in}]) \overset{HR}{\Rightarrow} \sigma'(c'[s_{in}])\,\sigma'(d'[s_{in}]) = \sigma'((c'\,d')[s_{in}]).$$

(BETA) : Alors $a = (\lambda c)d$ et $b = \sigma'(c'[d' \cdot id])$, où $c \Rightarrow c'$ et $d \Rightarrow d'$.

$$\sigma'(((\lambda c)d)[s_{in}]) = \sigma'((\lambda c)[s_{in}])\sigma'(d[s_{in}]) \overset{R3.7}{=} (\lambda\sigma'(c[s_{i+1\,n}]))\sigma'(d[s_{in}]) \overset{HR}{\Rightarrow}$$

$$\Rightarrow \sigma'(\sigma'(c'[s_{i+1\,n}])\,[\sigma'(d'[s_{in}]) \cdot id]) = \sigma'(c'[s_{i+1\,n} \circ (d'[s_{in}] \cdot id)]) \overset{R3.7}{=}$$

$$= \sigma'(c'[d'[s_{in}] \cdot s_{in}]) = \sigma'(c'[(d' \cdot id) \circ s_{in}]) = \sigma'(\sigma'(c'[d' \cdot id])[s_{in}]).$$

(CLOS) : Alors $a = S(X[u])$, $b = S(X[v])$ et $u \Rightarrow v$.
Or, par H.R. $\sigma'(u \circ s_{in}) \Rightarrow \sigma'(v \circ s_{in})$.

$$\sigma'(X[u][s_{in}]) = S(X[\sigma'(u \circ s_{in})]) \overset{HR}{\Rightarrow} S(X[\sigma(v \circ s_{in})]) = \sigma'(X[v][s_{in}]).$$

(CONS) : Alors $s = S(c \cdot u)$ et $t = S(d \cdot v)$, où $c \Rightarrow d$ et $u \Rightarrow v$.

$$\sigma'((c \cdot u) \circ s_{in}) = S(\sigma'(c[s_{in}]) \cdot \sigma'(u \circ s_{in})) \overset{HR}{\Rightarrow}$$

$$\Rightarrow S(\sigma'(d[s_{in}]) \cdot \sigma'(v \circ s_{in})) = \sigma'((d \cdot v) \circ s_{in})$$

□

Lemme 3.11 *Soient* $a \in \Lambda X^t_{\sigma'}$ *et* $s, t, u \in \Lambda X^s_{\sigma'}$ *tels que* $s \Rightarrow t$.
Alors $\sigma'(a[s]) \Rightarrow \sigma'(a[t])$ *et* $\sigma'(u \circ s) \Rightarrow \sigma'(u \circ t)$.

Démonstration
On le montre par récurrence simultanée sur a et u.
Posons $s = a_1 \cdot \ldots \cdot a_k \cdot \uparrow^n$. D'après le lemme 3.9, il existe $p \geq 0$ et $b_1, \ldots, b_{k-p} \in \Lambda X^t_{\sigma'}$ tels que $t = b_1 \cdot \ldots \cdot b_{k-p} \cdot \uparrow^{n-p}$ avec $p \leq k$, $a_{k-p+j} \Rightarrow \mathtt{n-p+j}$ pour $1 \leq j \leq p$ et $a_i \Rightarrow b_i$ pour $1 \leq i \leq k-p$.

$a = bc$: L'H.R. et la règle *(APPL)* nous permettent de conclure.

$a = \lambda b$: Le lemme 3.10.2 garantit $\sigma'(s \circ \uparrow) = \sigma'(s \circ s_{02}) \Rightarrow \sigma'(t \circ s_{02}) = \sigma'(t \circ \uparrow)$, d'où $S(\mathtt{1} \cdot \sigma'(s \circ \uparrow)) \Rightarrow S(\mathtt{1} \cdot \sigma'(t \circ \uparrow))$.
On peut donc appliquer l'H.R. pour obtenir :

$$\sigma'((\lambda b)[s]) = \lambda(\sigma'(b[S(\mathtt{1} \cdot \sigma'(s \circ \uparrow))])) \Rightarrow \lambda(\sigma'(b[S(\mathtt{1} \cdot \sigma'(t \circ \uparrow))])) = \sigma'((\lambda b)[t]).$$

$a = \mathtt{m}$: Trois possibilités selon la valeur de $\mathtt{m}$:

- Si $m \leq k-p$, alors $\sigma'(\mathtt{m}[s]) = a_m \Rightarrow b_m = \sigma'(\mathtt{m}[t])$.
- Si $k-p < m \leq k$, alors $m = k-p+j$ avec $1 \leq j \leq p$. Donc, $\sigma'(\mathtt{m}[s]) = a_{k-p+j} \Rightarrow \mathtt{n-p+j} \overset{R3.3}{=} \sigma'(\mathtt{1}[\uparrow^{m-1} \circ t]) = \sigma'(\mathtt{m}[t])$.
- Si $m > k$, alors $\sigma'(\mathtt{m}[s]) \overset{R3.3}{=} \mathtt{m+n-k} \overset{R3.3}{=} \sigma'(\mathtt{m}[t])$.
  Et la règle *(REFL)* assure $\sigma'(\mathtt{m}[s]) \Rightarrow \sigma'(\mathtt{m}[t])$.

$a = X$: $\sigma'(X[s]) = S(X[s]) \Rightarrow S(X[t]) = \sigma'(X[t])$.

$a = X[v]$: $\sigma'(X[v][s]) = S(X[\sigma'(v \circ s)]) \overset{HR}{\Rightarrow} S(X[\sigma'(v \circ t)]) = \sigma'(X[v][t])$.

$u = \uparrow^q$: On considère $q \geq 0$, i.e. le cas $u = id$ est inclus ici. Nous avons encore trois possibilités à considérer selon la valeur de q.

- Si $q < k-p$, alors
  $\sigma'(\uparrow^q \circ s) = a_{q+1} \cdot \ldots \cdot a_{k-p} \cdot a_{k-p+1} \cdot \ldots \cdot a_k \cdot \uparrow^n \overset{(CONS)}{\Rightarrow}$
  $\Rightarrow S(b_{q+1} \cdot \ldots \cdot b_{k-p} \cdot \mathtt{n-p+1} \cdot \ldots \cdot \mathtt{n} \cdot \uparrow^n) \overset{R3.11}{=}$
  $= b_{q+1} \cdot \ldots \cdot b_{k-p} \cdot \uparrow^{n-p} \overset{R3.3}{=} \sigma'(\uparrow^q \circ t)$.
- Si $k-p \leq q < k$, alors $q = k-p+j$ avec $1 \leq j \leq p$. Donc,
  $\sigma'(\uparrow^q \circ s) = a_{q+1} \cdot \ldots \cdot a_k \cdot \uparrow^n \overset{(CONS)}{\Rightarrow}$
  $\Rightarrow S(\mathtt{n-p+j+1} \cdot \ldots \cdot \mathtt{n} \cdot \uparrow^n) \overset{R3.11}{=} \uparrow^{n-p+j} \overset{R3.3}{=} \sigma(\uparrow^q \circ t)$.
- Si $q \geq k$, alors $\sigma'(\uparrow^q \circ s) \overset{R3.3}{=} \uparrow^{q+n-k} \overset{R3.3}{=} \sigma'(\uparrow^q \circ t)$.

$u = b \cdot v$: $\sigma'((b \cdot v) \circ s) = S(\sigma'(b[s]) \cdot \sigma'(v \circ s)) \overset{HR}{\Rightarrow} S(\sigma'(b[t]) \cdot \sigma'(v \circ t)) = \sigma'((b \cdot v) \circ t)$.
□

Lemme 3.12 *Soient* $a, b \in \Lambda X^t_{\sigma'}$ *et* $s, t, u, v \in \Lambda X^s_{\sigma'}$ *tels que* $u \Rightarrow v$ *. Alors*

1. *Si* $a \Rightarrow b$ *, alors* $\sigma'(a[u]) \Rightarrow \sigma'(b[v])$ *.*
2. *Si* $s \Rightarrow t$ *, alors* $\sigma'(s \circ u) \Rightarrow \sigma'(t \circ v)$ *.*

Démonstration
On démontre les deux résultats par récurrence simultanée sur les longueurs des inférences de $a \Rightarrow b$ et $s \Rightarrow t$. On étudie donc la dernière règle utilisée.

(REFL) : C'est le lemme précédent.

(ABST) : Alors $a = \lambda c$, $b = \lambda d$ et $c \Rightarrow d$.
Le lemme 3.10.2 garantit $\sigma'(u \circ \uparrow) = \sigma'(u \circ s_{02}) \Rightarrow \sigma'(v \circ s_{02}) = \sigma'(v \circ \uparrow)$.
Donc, $S(\mathtt{1} \cdot \sigma'(u \circ \uparrow)) \Rightarrow S(\mathtt{1} \cdot \sigma'(v \circ \uparrow))$. Et on peut appliquer l'H.R. et la règle *(ABST)* pour obtenir :

$$\sigma'((\lambda c)[u]) = \lambda(\sigma'(c[S(\mathtt{1} \cdot \sigma'(u \circ \uparrow))])) \Rightarrow \lambda\sigma'(d[S(\mathtt{1} \cdot \sigma'(v \circ \uparrow))]) = \sigma'((\lambda d)[v]) .$$

(APPL) : Alors $a = c\,d$ et $b = c'\,d'$, où $c \Rightarrow c'$ et $d \Rightarrow d'$.

$$\sigma'((c\,d)[u]) = \sigma'(c[u])\sigma'(d[u]) \overset{HR}{\Rightarrow} \sigma'(c'[v])\,\sigma'(d'[v]) = \sigma'((c'\,d')[v]) .$$

(BETA) : Alors $a = (\lambda c)d$ et $b = \sigma'(c'[d' \cdot id])$, où $c \Rightarrow c'$ et $d \Rightarrow d'$. Dans l'étude de *(ABST)* on a déjà établi $S(\mathtt{1} \cdot \sigma'(u \circ \uparrow)) \Rightarrow S(\mathtt{1} \cdot \sigma'(v \circ \uparrow))$. Nous pouvons utiliser l'H.R. pour obtenir :

$$\sigma'(((\lambda c)d)[u]) = \sigma'((\lambda c)[u])\sigma'(d[u]) = (\lambda(\sigma'(c[S(\mathtt{1} \cdot \sigma'(u \circ \uparrow))])))\sigma'(d[u]) \overset{HR}{\Rightarrow}$$

$$\Rightarrow \sigma'(\sigma'(c'[S(\mathtt{1} \cdot \sigma'(v \circ \uparrow))])[\sigma'(d'[v]) \cdot id]) = \sigma'(c'[\mathtt{1} \cdot (v \circ \uparrow)][(d'[v]) \cdot id]) =$$

$$\sigma'(c'[(d'[v]) \cdot (v \circ id)]) = \sigma'(c'[(d'[v]) \cdot v]) = \sigma'(\sigma'(c'[d' \cdot id])[v]) .$$

(CLOS) : Alors $a = S(X[u'])$, $b = S(X[v'])$ et $u' \Rightarrow v'$.

$$\sigma'(X[u'][u]) = S(X[\sigma'(u' \circ u)]) \overset{HR}{\Rightarrow} S(X[\sigma'(v' \circ v)]) = \sigma'(X[v'][v]) .$$

(CONS) : Alors $s = S(c \cdot s')$ et $t = S(d \cdot t')$, où $c \Rightarrow d$ et $s' \Rightarrow t'$.

$$\sigma'((c \cdot s') \circ u) = S(\sigma'(c[u]) \cdot \sigma'(s' \circ u)) \overset{HR}{\Rightarrow} S(\sigma'(d[v]) \cdot \sigma'(t' \circ v)) = \sigma'((d \cdot t') \circ v) .$$

□

Corollaire 3.5 *Soient* $a, b, c, d \in \Lambda X^t_{\sigma'}$ *tels que* $a \Rightarrow c$ *et* $b \Rightarrow d$ *. Alors*

$$\sigma'(a[b \cdot id]) \Rightarrow \sigma'(c[d \cdot id]) .$$

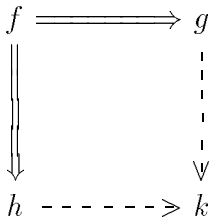
Démonstration
Comme $\sigma'(a[b \cdot id]) = \sigma'(a[S(b \cdot id)])$, d'après le lemme précédent, il suffit de montrer $S(b \cdot id) \Rightarrow S(d \cdot id)$. Or, ceci est assuré par la règle *(CONS)*.
□

3.6 Les résultats de confluence

Pour montrer la confluence forte de la parallélisation nous n'utilisons que le corollaire précédent. La confluence du λ'-calcul est donc assurée, et nous pouvons enfin utiliser la méthode d'interprétation pour obtenir la confluence du $\lambda\sigma'$-calcul sur les termes semi-clos.

Théorème 3.1 *La réduction $\Rightarrow$ est fortement confluente sur ΛX_σ .*

Démonstration
Nous devons montrer que pour tout $f, g, h \in \Lambda X_{\sigma'}$ tels que $f \Rightarrow g$ et $f \Rightarrow h$, il existe $k \in \Lambda X_{\sigma'}$ tel que $g \Rightarrow k$ et $h \Rightarrow k$.



Nous procédons par récurrence sur la longueur de l'inférence $f \Rightarrow g$. Analysons donc la dernière règle utilisée dans cette inférence.

(REFL) : Alors $g = f$ et le diagramme se ferme trivialement avec $k = h$.

(ABST) : Alors $f = \lambda a$, $g = \lambda b$ et $a \Rightarrow b$.
Remarquons que $h = \lambda c$ avec $a \Rightarrow c$. En effet, la dernière règle utilisée pour obtenir $\lambda a \Rightarrow h$ ne peut être que *(ABST)* ou *(REFL)*.
Alors par H.R. il existe $d \in \Lambda X^t_{\sigma'}$ tel que $b \Rightarrow d$ et $c \Rightarrow d$. En utilisant la règle *(ABST)* on conclut $\lambda b \Rightarrow \lambda d$ et $\lambda c \Rightarrow \lambda d$.

(APPL) : Alors $f = a\,b$ et $g = a'\,b'$, où $a \Rightarrow a'$ et $b \Rightarrow b'$.
Remarquons qu'il y a deux possibilités pour h, selon la dernière règle utilisée pour dériver $a\,b \Rightarrow h$.

(a) Si la règle est *(REFL)* ou *(APPL)*, $h = a''b''$, avec $a \Rightarrow a''$ et $b \Rightarrow b''$. Par H.R. il existe $c, d \in \Lambda X^t_{\sigma'}$ tels que $a' \Rightarrow c$, $a'' \Rightarrow c$, $b' \Rightarrow d$ et $b'' \Rightarrow d$. On conclut en appliquant *(APPL)*.

(b) Si la règle est *(BETA)*, $h = \sigma'(d''[b'' \cdot id])$, $a = \lambda d$, $d \Rightarrow d''$ et $b \Rightarrow b''$. D'après la remarque que nous avons fait dans l'étude de *(ABST)*, $a' = \lambda d'$, où $d \Rightarrow d'$.
L'H.R. assure l'existence de $c, e \in \Lambda X^t_{\sigma'}$ tels que $b' \Rightarrow c$, $b'' \Rightarrow c$. $d' \Rightarrow e$ et $d'' \Rightarrow e$.
En appliquant *(BETA)*, on obtient :

$$g = a'\,b' = (\lambda d')b' \Rightarrow \sigma'(e[c \cdot id])\,.$$

Et en utilisant le corollaire 3.5, on conclut :

$$h = \sigma'(d''[b'' \cdot id]) \Rightarrow \sigma'(e[c \cdot id])\,.$$

(BETA) : Alors $f = (\lambda a)b$ et $g = \sigma'(a'[b' \cdot id])$, où $a \Rightarrow a'$ et $b \Rightarrow b'$.
D'après un raisonnement analogue à celui fait précédemment, il y a deux possibilités pour h.

(a) $h = c\,b''$ où $\lambda a \Rightarrow c$ et $b \Rightarrow b''$.
Le raisonnement est symétrique au précédent.

(b) $h = \sigma'(a''[b'' \cdot id])$, avec $a \Rightarrow a''$ et $b \Rightarrow b''$.
Par H.R. on a $c,\, d \in \Lambda X^t_{\sigma'}$ tels que $a' \Rightarrow c$, $a'' \Rightarrow c$, $b' \Rightarrow d$ et $b'' \Rightarrow d$.
On utilise deux fois le corollaire 3.5 pour conclure $g \Rightarrow \sigma'(c[d \cdot id])$ et $h \Rightarrow \sigma'(c[d \cdot id])$.

(CLOS) : Alors $f = S(X[s])$ et $g = S(X[t])$, où $s \Rightarrow t$.
Remarquons que $S(X[s]) \Rightarrow h$ entraîne l'existence de $u \in \Lambda X^s_{\sigma'}$ telle que $h = S(X[u])$ et $s \Rightarrow u$. En effet,

1. Si $s = id$, alors $S(X[s]) = X$ et la règle ne peut être que *(REFL)* ou *(CLOS)*. Dans les deux cas, il suffit de prendre $u = id$. On rappelle que le seul $\Rightarrow$-réduit de id est id.
2. Si $s \neq id$, alors $S(X[s]) = X[s]$. Encore, la règle utilisée ne peut être que *(REFL)* ou *(CLOS)*. Si c'est *(REFL)*, prendre $u = s$ et si c'est *(CLOS)* prendre la substitution donnée par cette règle.

Par H.R. il existe $v \in \Lambda X^s_{\sigma'}$ telle que $t \Rightarrow v$ et $u \Rightarrow v$, et on conclut en appliquant *(CLOS)*

(CONS) : Alors $f = S(a \cdot s)$ et $g = S(b \cdot t)$, où $a \Rightarrow b$ et $s \Rightarrow t$.
On affirme $h = S(c \cdot u)$, où $a \Rightarrow c$ et $s \Rightarrow u$. En effet,

1. S'il existe n tel que $a = \mathtt{n}$ et $s = \uparrow^n$, alors $f = \uparrow^{n-1}$, et la règle pour dériver $f \Rightarrow h$ doit être *(REFL)* ou *(CONS)*. Dans les deux cas il suffit de prendre $c = a$ et $u = s$. On rappelle que $\mathtt{n}$ et $\uparrow^n$ ont un seul $\Rightarrow$-réduit : $\mathtt{n}$ et $\uparrow^n$, respectivement.
2. Si un tel n n'existe pas, $f = a \cdot s$. Les règles possibles pour dériver h sont toujours *(REFL)* ou *(CONS)*. Si elle est *(REFL)*, prendre $c = a$ et $u = s$. Si elle est *(CONS)*, la prémise fournit c et u.

Enfin, l'H.R. et l'application de *(CONS)* permettent de conclure.

□

Théorème 3.2 *Le λ'-calcul est confluent sur $\Lambda X_{\sigma'}$.*

Démonstration
La confluence forte de $\Rightarrow$ entraîne celle de $\Rightarrow^*$ (cf. [Bar84], lemme 3.2.2), donc $\twoheadrightarrow_{\beta'}$ est fortement confluent, i.e. $\rightarrow_{\beta'}$ est confluente.
□

Corollaire 3.6 *Le $\lambda\beta$-calcul, i.e. le λ-calcul à la de Bruijn, est confluent sur Λ .*

Démonstration
On constate aisément que les termes clos sont stables par λ'-dérivation. Ceci assure la confluence de $\rightarrow_{\beta'}$ sur Λ . Le corollaire 3.1.2 permet de conclure.
□

Théorème 3.3 *Le $\lambda\sigma'$-calcul est confluent sur ΛX .*

Démonstration
Nous utilisons la méthode d'interprétation (cf. lemme 1.4), en interprétant les termes et substitutions dans $\Lambda X_{\sigma'}$.

La simulation de β' dans le $\lambda\sigma'$-calcul (cf. remarque 3.6), l'interprétation de *(Beta)* par β' sur les σ'-formes normales (cf. proposition 3.3) et la confluence du λ'-calcul (cf. théorème précédent) permettent d'appliquer la méthode d'interprétation.
□

Chapitre 4

Confluence du $\lambda\sigma\eta$-calcul semi-clos

Dans ce chapitre nous introduisons la règle (*Eta*), qui est une règle conditionnelle, et montrons la confluence du calcul obtenu en ajoutant cette règle aux règles du $\lambda\sigma'$-calcul, que nous appelons $\lambda\sigma\eta$-calcul. Nous travaillons toujours sur l'ensemble des termes semi-clos introduit dans le chapitre précédent.

La méthode utilisée pour la preuve de confluence est encore la méthode d'interprétation. Le calcul dans lequel nous interprétons le $\lambda\sigma\eta$-calcul est appelé $\lambda\eta'$-calcul et il est obtenu en ajoutant une règle appelée η' au λ'-calcul que nous avons déjà utilisé pour interpréter le $\lambda\sigma'$-calcul. La règle η' est obtenue à partir d'une application de *(Eta)* suivie de σ'-normalisation du terme entier.

Notre preuve de confluence se ramène donc à la confluence du $\lambda\eta'$-calcul et celle-ci est montrée, comme dans le λ-calcul classique, grâce à la commutation des règles β' et η'.

Dans la section 4.1 nous présentons la règle (*Eta*) et montrons qu'elle est bien définie modulo σ'. Cette preuve se base sur des lemmes qui discutent l'existence de solutions de certaines équations et qui seront utilisés constamment dans le reste du chapitre.

La règle η' et le $\lambda\eta'$-calcul sont introduits dans la section 4.2, où nous donnons un algorithme pour vérifier la condition nécessaire pour appliquer *(Eta)*. Nous concluons que η' est une extension de η à $\Lambda X_{\sigma'}$. Les résultats obtenus dans cette section ne seront pas utilisés dans la suite.

Dans la section 4.3 nous prouvons que toute $\lambda\sigma\eta$-dérivation s'interprète, via σ'-normalisation, par une $\lambda\eta'$-dérivation. Cette preuve se ramène essentiellement à montrer :

$$\text{Si } f \to_{(Eta)} g \text{, alors } \sigma'(f) \twoheadrightarrow_{\eta'} \sigma'(g) .$$

Pour obtenir ce résultat il faut établir une série de lemmes analogues à ceux de la section 3.3 du chapitre précédent.

La section 4.4 est consacrée à la confluence de η'. En fait, nous montrons la confluence quasi-forte (cf. définition 1.8.3) de η'. Nous devons établir quelques lemmes techniques avant de montrer le théorème de confluence.

Enfin, dans la section 4.5, nous établissons la commutation de β' et η' pour obtenir la confluence du $\lambda\eta'$-calcul qui, grâce à la méthode d'interprétation, fournit celle du $\lambda\sigma\eta$-calcul.

4.1 Le $\lambda\sigma\eta$-calcul

La règle *(Eta)* a déjà été introduite dans le contexte du $\lambda\sigma$-calcul clos et du $\lambda\sigma_{\Uparrow}$-calcul sur les termes et substitutions ouverts (cf. [Har92]). Nous adaptons cette définition en vue d'ajouter cette règle au $\lambda\sigma'$-calcul sur les termes semi-clos.

Avant de définir *(Eta)* nous reproduisons la discussion informelle de T. Hardin pour introduire cette règle dans le $\lambda\sigma_{\Uparrow}$-calcul, qui est aussi valable pour notre calcul.

Signalons d'abord le lien entre les règles β et η dans le λ-calcul classique :

$$(\lambda x.\, a\, x)\, b \to_{\beta} a\, b \quad \text{et} \quad (\lambda x.\, a\, x)\, b \to_{\eta} a\, b\,.$$

Essayons de reproduire cette situation dans le $\lambda\sigma'$-calcul avec $a = \mathtt{2}\,\mathtt{1}$.

$$(\lambda(\mathtt{2}\,\mathtt{1}))\, b \to_{(Beta)} (\mathtt{2}\,\mathtt{1})[b \cdot id] \twoheadrightarrow_{\sigma'} \mathtt{1}\, b\,.$$

Remarquons que $\mathtt{2} = \mathtt{1}[\uparrow]$. De manière plus génerale :

$$(\lambda(c[\uparrow]\,\mathtt{1}))\, b \to_{(Beta)} (c[\uparrow]\,\mathtt{1})[b \cdot id] \twoheadrightarrow_{\sigma'} (c[\uparrow \circ (b \cdot id)])\, b \twoheadrightarrow_{\sigma'} c\, b\,.$$

On serait donc tenté d'essayer : $(\lambda(c[\uparrow]\,\mathtt{1})) \to_{(Eta)} c$.

Cette version de *(Eta)* correspond à la règle (SΛ) dans le cadre des combinateurs catégoriques (cf. [Har87]) qui, ajoutée au système $\mathcal{E}+(Beta)$ ($\lambda\sigma$ dans notre version), donne un système confluent sur un sous-ensemble de termes clos.

Mais cette formulation de *(Eta)* crée une infinité de paires critiques avec les autres σ'-règles, par exemple, avec *(Clos)* :

$$\lambda(c[s][\uparrow]\,\mathtt{1}) \to \lambda(c[s\circ \uparrow]\,\mathtt{1}) \text{ et } \lambda(c[s][\uparrow]\,\mathtt{1}) \to c[s]$$

$$\lambda(c[(s \circ t)\circ \uparrow]\,\mathtt{1}) \to \lambda(c[s \circ (t\circ \uparrow)]\,\mathtt{1}) \text{ et } \lambda(c[(s \circ t)\circ \uparrow]\,\mathtt{1}) \to c[s \circ t]\,, \text{ etc.}$$

On retrouve donc la même situation que dans le cas de $\lambda\sigma$: un système confluent sur un ensemble de termes clos, qui n'est même pas localement confluent sur tous les termes.

Etant données ces difficultés la règle de réécriture conditionnelle introduite par Hardin semble être la plus adéquate. Malheureusement, elle n'est pas une règle de réécriture au sens classique, car une condition existentielle doit être vérifiée pour appliquer cette règle. Dans la section suivante nous donnons une condition équivalente qui est plus maniable.

Définition 4.1 *Soient $a, b \in \Lambda X^t$. La règle (Eta) est donnée par :*

$$\lambda(a\mathtt{1}) \to_{(Eta)} b \quad \textit{si} \quad \sigma'(a) = \sigma'(b[\uparrow])\,.$$

*Le calcul obtenu par l'ajout de la règle (Eta) au $\lambda\sigma'$-calcul est appelé $\lambda\sigma\eta$-*calcul.

Le premier résultat que nous devons démontrer est que cette définition est bonne dans le sens qu'elle ne dépend pas (modulo σ') du terme b qui satisfait la condition $\sigma'(a) = \sigma'(b[\uparrow])$. Plus précisément, il faut montrer que si $\sigma'(b[\uparrow]) = \sigma'(c[\uparrow])$, alors $\sigma'(b) = \sigma'(c)$.

En vue d'obtenir ce résultat nous devons commencer par déterminer les solutions de certaines équations. Les lemmes de cette section seront utilisés à plusieurs reprises dans le reste du chapitre.

Lemme 4.1 *L'équation* $\sigma'(a[s_{n\,2}]) = \mathtt{m}$

1. *a une seule solution dans* $\Lambda X^t_{\sigma'}$: $a = \mathtt{m}$, *si* $m \leq n$;
2. *a une seule solution dans* $\Lambda X^t_{\sigma'}$: $a = \mathtt{m} - \mathtt{1}$, *si* $m > n + 1$;
3. *n'a pas de solution si* $m = n + 1$.

Démonstration
Remarquons d'abord que si $a \in \Lambda X^t_{\sigma'}$ est solution de $\sigma'(a[s_{n\,2}]) = \mathtt{m}$, d'après les remarques 3.4 et 3.5, a ne peut être qu'un numéro de de Bruijn. Soit donc $a = \mathtt{p}$. La remarque 3.3 donne :

$$\mathtt{p}[s_{n\,2}] \twoheadrightarrow_\sigma \begin{cases} \mathtt{p} & \text{si} \quad p - 1 < n \\ \mathtt{p+1} & \text{si} \quad p - 1 \geq n\,. \end{cases}$$

Dans le premier cas $\mathtt{p} = \mathtt{m}$ est la seule solution pourvu que $m - 1 < n$, i.e. $m \leq n$. Dans, le deuxième cas, $\mathtt{p} + \mathtt{1} = \mathtt{m}$, la seule solution est donc $\mathtt{p} = \mathtt{m} - \mathtt{1}$, si $m-2 \geq n$, i.e. si $m > n + 1$.
On constate donc que l'équation n'a pas de solution quand $m = n + 1$.
□

Lemme 4.2 *Soit* $m \geq 0$. *L'équation* $\sigma'(t \circ s_{n\,2}) = \uparrow^m$

1. *a une seule solution dans* $\Lambda X^s_{\sigma'}$: $t = \uparrow^{m-1}$, *si* $m > n$;
2. *n'a pas de solution si* $m \leq n$.

Démonstration
Récurrence sur la structure de t.
Montrons d'abord que si t est une solution de la forme $a \cdot s$, alors $t \notin \Lambda X^s_{\sigma'}$. En effet,

$$\sigma'((a \cdot s) \circ s_{n\,2}) = S(\sigma'(a[s_{n\,2}]) \cdot \sigma'(s \circ s_{n\,2}))\,.$$

Donc (cf. remarque 3.5.4), $\sigma'(a[s_{n\,2}]) = \mathtt{m} + \mathtt{1}$ et $\sigma'(s \circ s_{n\,2}) = \uparrow^{m+1}$. En utilisant le lemme précédent et l'H.R. on conclut $m + 1 > n$, $a = \mathtt{m}$ et $s = \uparrow^m$, d'où $t = a \cdot s$

n'est pas en σ'-forme normale.
Donc t est nécessairement de la forme $\uparrow^p$ où $p \geq 0$. La remarque 3.3 donne:

$$\uparrow^p \circ s_{n\,2} \twoheadrightarrow_\sigma \begin{cases} \mathtt{p+1} \cdot \ldots \cdot \mathtt{n} \cdot \uparrow^{n+1} & \text{si} \quad p < n \\ \uparrow^{p+1} & \text{si} \quad p \geq n\,. \end{cases}$$

On constate donc que $t = \uparrow^{m-1}$ est la seule solution pourvu que $m-1 \geq n$, i.e. $m > n$.
□

Nous voulons montrer maintenant que la seule solution de l'équation $\sigma'(t \circ s_{n\,2}) = s_{n\,2}$ est $t = id$. Si $t = a \cdot s$ nous sommes ramenés à considérer l'équation $\sigma'(s \circ s_{n\,2}) = \mathtt{2} \cdot \ldots \cdot \mathtt{n} \cdot \uparrow^{n+1}$. Ceci suggère d'introduire la définition suivante et essayer de montrer un résultat plus général (lemme 4.3).

Définition 4.2 *Soient $k \geq 1$, $n \geq 0$ et $1 \leq i \leq n+1$. On définit*

$$s_{i\,n\,k} = \mathtt{i} \cdot \mathtt{i+1} \cdot \ldots \cdot \mathtt{n} \cdot \uparrow^{n+k-1} \quad .$$

On convient pour $i = n+1$, $s_{i\,n\,k} = \uparrow^{n+k-1}$ et $s_{1\,0\,1} = id$.

Lemme 4.3 *Soient $n \geq 0$ et $1 \leq i \leq n+1$. L'équation $\sigma'(t \circ s_{n\,2}) = s_{i\,n\,2}$ a une seule solution dans $\Lambda X^s_{\sigma'}$, à savoir $t = \uparrow^{i-1}$.*

Démonstration
Par récurrence sur la complexité de $s_{i\,n\,2}$, i.e. sur l'entier $n-i+1$.

Si elle est nulle, i.e. $i = n+1$, l'équation devient $\sigma'(t \circ s_{n\,2}) = \uparrow^{n+1}$ et le lemme précédent donne la seule solution $t = \uparrow^n$.

Supposons donc que la complexité est positive, i.e. $i < n+1$.
Si t est de la forme $\uparrow^p$, on vérifie que t est solution ssi $p = i-1$ (cf. remarque 3.3).
Si t est de la forme $a \cdot u$, on a:

$$\sigma'((a \cdot u) \circ s_{n\,2}) = S(\sigma'(a[s_{n\,2}]) \cdot \sigma'(u \circ s_{n\,2}))\,.$$

Comme $s_{i\,n\,2} = \mathtt{i} \cdot s_{i+1\,n\,2}$, on conclut (cf. remarque 3.5.4) qu'il n'existe pas m tel que $\sigma'(a[s_{n\,2}]) = \mathtt{m}$ et $\sigma'(u \circ s_{n\,2})) = \uparrow^m$. Donc,

$$\sigma'((a \cdot u) \circ s_{n\,2}) = \sigma'(a[s_{n\,2}]) \cdot \sigma'(u \circ s_{n\,2}) = s_{i\,n\,2} = \mathtt{i} \cdot s_{i+1\,n\,2}\,.$$

On doit donc résoudre les deux équations:

1. $\sigma(a[s_{n\,2}]) = \mathtt{i}$.
2. $\sigma(u \circ s_{n\,2}) = s_{i+1\,n\,2}$.

Le lemme 4.1 donne une seule solution pour la première, à savoir $a = \mathtt{i}$.
Pour la deuxième on utilise l'H.R., la complexité de $s_{i+1\,n\,2}$ étant plus petite que celle de $s_{i\,n\,2}$. On a donc une seule solution : $u = \uparrow^i$. Ceci est absurde, car $t \in \Lambda X^s_{\sigma'}$. Donc, t n'est pas un cons et la seule solution est $t = \uparrow^{i-1}$.
□

Lemme 4.4 *Soit $t \in \Lambda X^s_\sigma$. Pour tout $n \geq 0$, on a $\sigma'(t \circ s_{n\,2}) \neq id$.*

Démonstration
Il suffit de calculer $\sigma'(t \circ s_{n\,2})$ selon la forme de t.
Si $t = \uparrow^m$ où $m \geq 0$, le résultat est un cons ou $\uparrow^{m+1}$, donc différent de id.
Si $t = a \cdot s$, on a :

$$\sigma'((a \cdot s) \circ s_{n\,2}) = S(\sigma'(a[s_{n\,2}]) \cdot \sigma'(s \circ s_{n\,2}))$$

Supposons qu'il existe $m \geq 1$ tel que $\sigma'(a[s_{n\,2}]) = \mathtt{m}$ et $\sigma'(s \circ s_{n\,2})) = \uparrow^m$. D'après les lemmes 4.1 et 4.2, $m > n + 1 \geq 1$. Donc (cf. remarque 3.5.4),

$$\sigma'((a \cdot s) \circ s_{n\,2}) = \uparrow^{m-1} \neq id, \text{ car } m - 1 > 0.$$

S'il n'existe pas un tel m, d'après la remarque 3.5.3, on a :

$$\sigma'((a \cdot s) \circ s_{n\,2}) = \sigma'(a[s_{n\,2}]) \cdot \sigma'(s \circ s_{n\,2}) \neq id.$$

□

Nous avons maintenant les éléments nécessaires pour montrer la bonne définition de (Eta) :

Proposition 4.1 *Soient $n \geq 0$, $b, c \in \Lambda X^t_{\sigma'}$ et $t, u \in \Lambda X^s_{\sigma'}$.*

1. *Si $\sigma'(b[s_{n\,2}]) = \sigma'(c[s_{n\,2}])$, alors $b = c$.*
2. *Si $\sigma'(t \circ s_{n\,2}) = \sigma'(u \circ s_{n\,2})$, alors $t = u$.*

Démonstration
On montre les deux résultats par récurrence simultanée sur la structure de b et t.

$b = b'b''$: $\sigma'(b[s_{n\,2}]) = \sigma'(b'[s_{n\,2}])\sigma'(b''[s_{n\,2}]) = \sigma'(c[s_{n\,2}])$.
Alors c ne peut être qu'une application $c'c''$ et l'H.R. permet de conclure.

$b = \lambda d$: $\sigma'(b[s_{n\,2}]) = \lambda\sigma'(d[s_{n+1\,2}]) = \sigma'(c[s_{n\,2}])$.
Alors c ne peut être qu'une abstraction λe et l'H.R. permet de conclure.

$b = \mathtt{m}$: D'après la remarque 3.3 on a :

$$\sigma'(b[s_{n\,2}]) \;=\; \begin{cases} \mathtt{m} & \text{si} \quad m \leq n \\ \mathtt{m+1} & \text{si} \quad m > n\,. \end{cases}$$

Dans le premier cas $\mathtt{m} = \sigma'(c[s_{n\,2}])$, et le lemme 4.1 donne la seule solution $c = \mathtt{m} = b$.
Dans le deuxième, $\mathtt{m+1} = \sigma'(c[s_{n\,2}])$, et encore le lemme 4.1 donne la seule solution $c = \mathtt{m} = b$.

$b = X$: $\sigma'(b[s_{n\,2}]) = X[s_{n\,2}] = \sigma'(c[s_{n\,2}])$.
En analysant la structure qui peut avoir c, on arrive à : soit $c = X$, soit $c = X[s]$.
Dans le premier cas on conclut immédiatement.
Soit donc $c = X[s]$, alors (le lemme précédent assure $\sigma'(s \circ s_{n\,2}) \neq id$)

$$X[s_{n\,2}] = \sigma'(X[s][s_{n\,2}]) = X[\sigma'(s \circ s_{n\,2})]\,.$$

Donc, $s_{n\,2} = \sigma'(s \circ s_{n\,2})$, et le lemme 4.3 avec $i = 1$ donne $s = id$. Donc, ce cas est impossible car $c \in \Lambda X^t_{\sigma'}$.

$b = X[s]$: Encore deux possibilités pour c :
Si $c = X$ on arrive, comme dans le cas précédent, à $s = id$, ce qui est absurde.
Si $c = X[s]$ on utilise l'H.R..

$t = \uparrow^m$: On considère $m \geq 0$, de sorte que le cas $t = id$ est inclu ici. La remarque 3.3 donne :

$$\sigma'(t \circ s_{n\,2}) \;=\; \begin{cases} s_{m+1\;n\,2} & \text{si} \quad m < n \\ \uparrow^{m+1} & \text{si} \quad m \geq n\,. \end{cases}$$

Dans le premier cas, le lemme 4.3 donne $u = \uparrow^m$. Dans le deuxième cas, le lemme 4.2 donne encore $u = \uparrow^m$.

$t = a \cdot s$: $\sigma'((a \cdot s) \circ s_{n\,2}) = S(\sigma'(a[s_{n\,2}]) \cdot \sigma'(s \circ s_{n\,2})) = \sigma'(u \circ s_{n\,2})$.
S'il existe $m \geq 1$ tel que $\sigma'(a[s_{n\,2}]) = \mathtt{m}$ et $\sigma'(s \circ s_{n\,2}) = \uparrow^m$, en raisonnant comme dans la démonstration du lemme 4.3, on arrive à $t \notin \Lambda X^s_{\sigma'}$.
Donc, $\sigma'(u \circ s_{n\,2}) = \sigma'(a[s_{n\,2}]) \cdot \sigma'(s \circ s_{n\,2})$. Si $u = d \cdot v$, on utilise l'H.R. et un raisonnement analogue à celui mentionné précédemment pour conclure.
Si $u = \uparrow^p$, alors $p < n$ car $\sigma'(u \circ s_{n\,2})$ est un cons, et dans ce cas $\sigma'(u \circ s_{n\,2}) = s_{p+1\;n\,2}$. Donc, $\sigma'(a[s_{n\,2}]) = \mathtt{p+1}$ et $\sigma'(s \circ s_{n\,2}) = s_{p+2\;n\,2}$.
Le lemme 4.1 assure $a = \mathtt{p+1}$ et le lemme 4.3 donne $s = \uparrow^{p+1}$. Et ceci est absurde car $t \in \Lambda X^s_{\sigma'}$.

□

Corollaire 4.1 *Soient* $b, c \in \Lambda X^t$. *Si* $\sigma'(b[\uparrow]) = \sigma'(c[\uparrow])$, *alors* $\sigma'(b) = \sigma'(c)$.

Démonstration
Puisque $\sigma'(\sigma'(b)[s_{02}]) = \sigma'(b[\uparrow]) = \sigma'(c[\uparrow]) = \sigma'(\sigma'(c)[s_{02}])$, la proposition précédente donne $\sigma'(b) = \sigma'(c)$.
□

4.2 Le $\lambda\eta'$-calcul

Nous définissons la règle η' à partir de *(Eta)* de manière analogue à la définition de β' à partir de *(Beta)*: la règle η' est donnée par une application de (Eta) suivie de σ'-normalisation totale. Nous appelons $\lambda\eta'$-calcul, le calcul dont les seules règles sont β' et η'.

Dans le chapitre précédent nous avons montré que la règle β' est une extension de la β-réduction clasique à $\Lambda X_{\sigma'}$. De même, η' est une extension de la η-réduction clasique à $\Lambda X_{\sigma'}$. Nous présentons ce résultat comme corollaire d'un lemme qui s'avère très intéressant lui-même, car il fournit un algorithme pour vérifier la condition de la règle *(Eta)*.

Définition 4.3 *La règle η' est définie sur $\Lambda X_{\sigma'}$ ainsi :*

$$f \to_{\eta'} g \quad ssi \quad il\ existe\ h \in \Lambda X\ tel\ que\ f \to_{(Eta)} h \quad et \quad g = \sigma'(h).$$

Le $\lambda\eta'$-calcul *est le calcul sur $\Lambda X_{\sigma'}$ dont les seules règles sont β' et η'.*

D'après cette définition la remarque suivante est immédiate.

Remarque 4.1 *Si $f \to_{\eta'} g$, alors $f \twoheadrightarrow_{\lambda\sigma\eta} g$.*

Nous avertissons le lecteur que les résultats montrés dans cette section ne seront utilisés dans la suite que pour montrer le corollaire 4.5. Nous le renvoyons à la section 1.5 pour les définitions de la règle η dans le λ-calcul à la de Bruijn, de la famille de condions $C\eta_n$ et des a_n. Nous commençons par étendre cette dernière définition à $\Lambda X_{\sigma'}$.

Définition 4.4 *Soient $f \in \Lambda X_{\sigma'}$ et $n \geq 1$. On définit partiellement f_n par récurrence sur la structure de f ainsi :*

$$(a\,b)_n = a_n\,b_n \qquad id_{\,n} = \text{indéfini}$$

$$(\lambda a)_n = \lambda a_{n+1} \qquad (a \cdot s)_n = a_n \cdot s_n$$

$$\mathtt{m}_{\,n} = \begin{cases} \mathtt{m}-\mathtt{1} & si\ \ m > n \\ \text{indéfini} & si\ \ m = n \\ \mathtt{m} & si\ \ m < n\,. \end{cases} \qquad (\uparrow^m)_n = \begin{cases} \uparrow^{m-1} & si\ \ m \geq n \\ \text{indéfini} & si\ \ m < n \end{cases}$$

$$X_{\,n} = \text{indéfini} \qquad (X[s])_n = S(X[s_n])$$

On sous-entend que chaque membre gauche est défini ssi tous les f_n apparaissant dans le membre droit correspondant sont définis.

Remarquons que nous avons été obligés de prendre la $SPID$-f.n. dans le cas $X[s]$ pour assurer $(X[s])_n \in \Lambda X^t_{\sigma'}$. En effet, la possibilité $s_n = id$ peut se présenter seulement quand $s = \uparrow$ et $n = 1$. Par contre, prendre la $SPID$-f.n. n'est pas nécessaire pour le cas $a \cdot s$, car $a_n = \mathtt{m}$ et $s_n = \uparrow^m$ entraînent $m+1 > n$, $a = \mathtt{m+1}$ et $s = \uparrow^{m+1}$, qui est imposible si $a \cdot s \in \Lambda X^s_{\sigma'}$.

Lemme 4.5 *Soient $a \in \Lambda X^t_{\sigma'}$, $s \in \Lambda X^s_{\sigma'}$ et $n \geq 1$.*

1. *Si a_n est défini, $a = \sigma'(a_n[s_{n-1\,2}])$.*
2. *Si s_n est définie, $s = \sigma'(s_n \circ s_{n-1\,2})$.*
3. *S'il existe $b \in \Lambda X^t_{\sigma'}$ tel que $a = \sigma'(b[s_{n-1\,2}])$, alors $b = a_n$.*
4. *S'il existe $t \in \Lambda X^s_{\sigma'}$ tel que $s = \sigma'(t \circ s_{n-1\,2})$, alors $t = s_n$.*

Démonstration
Les deux premiers résultats sont montrés par récurrence simultanée sur a et s.
Etudions par exemple le cas $a = \mathtt{m}$. Si a_n est défini alors $m \neq n$.
Si $m > n$, on a $\mathtt{m}_n = \mathtt{m-1}$ et $\mathtt{m} = \sigma'((\mathtt{m}-\mathtt{1})[s_{n-1\,2}])$ se vérifie.
Si $m < n$, on a $\mathtt{m}_n = \mathtt{m}$ et $\mathtt{m} = \sigma'(\mathtt{m}[s_{n-1\,2}])$ se vérifie aussi.
Pour les autres cas on utilise l'H.R., sauf pour le cas $s = \uparrow^m$ où un simple calcul permet de conclure.

Pour obtenir les deux derniers résultats, on montre sans difficulté par récurrence simultanée sur b et t :

$$(\sigma'(b[s_{n-1\,2}]))_n = b \quad \text{et} \quad (\sigma'(t \circ s_{n-1\,2}))_n = t\,.$$

□

Corollaire 4.2 *Soient $a \in \Lambda X^t_{\sigma'}$, $s \in \Lambda X^s_{\sigma'}$ et $n \geq 1$.*

1. *Il existe $b \in \Lambda X^t_{\sigma'}$ tel que $a = \sigma'(b[s_{n-1\,2}])$ ssi a_n est défini.*
2. *Il existe $t \in \Lambda X^s_{\sigma'}$ tel que $s = \sigma'(t \circ s_{n-1\,2})$ ssi s_n est définie.*

Ce corollaire donne donc un algorithme pour vérifier la condition nécessaire pour appliquer *(Eta)*. En effet, étant donné $a \in \Lambda X^t$, pour constater l'existence de $b \in \Lambda X^t$ tel que $\sigma'(a) = \sigma'(b[\uparrow])$, il suffit de vérifier si $(\sigma'(a))_1$ est défini. Et ceci peut se faire de manière récursive.

Un deuxième corollaire du lemme 4.5 est que la réduction η' est une extension de η à $\Lambda X_{\sigma'}$.

Corollaire 4.3 *Soient $a, b \in \Lambda$, alors $a \to_{\eta'} b$ ssi $a \to_\eta b$.*

Démonstration
Récurrence sur a. Le seul cas intéressant est quand la réduction a lieu à la racine. Supposons $a = \lambda(c\,\mathtt{1}) \to_{\eta'} b$. Donc, $b = \sigma'(d)$ et $c = \sigma'(d[\uparrow]) = \sigma'(b[\uparrow])$. D'après le lemme 4.5.3, $b = c_1$, et le lemme 1.6 assure que c satisfait $C\eta_1$ et que $b = c^{\downarrow}$. Alors (cf. définition 1.32), $a \to_{\eta} b$.
Réciproquement, supposons $a = \lambda(c\,\mathtt{1}) \to_{\eta} b$. Alors $b = c^{\downarrow}$ et c satisfait $C\eta_1$. Encore le lemme 1.6 garantit que c_1 est défini et $c_1 = b$. D'après le lemme 4.5.1, on déduit $c = \sigma'(b[\uparrow])$ et, comme $b \in \Lambda X^t_{\sigma'}$, on conclut $a \to_{\eta'} b$.
□

4.3 Interprétation de *(Eta)*

L'objectif de cette section est de montrer que toute $\lambda\sigma\eta$-dérivation s'interprète, via σ'-normalisation, par une $\lambda\eta'$-dérivation. Pour cela nous devons montrer une série de lemmes analogues à ceux de la section 3.3 du chapitre précédent. Nous nous contentons ici de signaler les différences essentielles par rapport à ces preuves-là.

La preuve de la remarque suivante est identique à celle de la remarque 3.9.

Remarque 4.2 *Soient* $a, b, c \in \Lambda X^t_{\sigma'}$ *et* $s, t, u \in \Lambda X^s_{\sigma'}$.

1. *Si* $a \to_{\eta'} b$, *alors* $a\,c \to_{\eta'} b\,c$.
2. *Si* $a \to_{\eta'} b$, *alors* $c\,a \to_{\eta'} c\,b$.
3. *Si* $a \to_{\eta'} b$, *alors* $\lambda a \to_{\eta'} \lambda b$.
4. *Si* $s \to_{\eta'} t$, *alors* $S(X[s]) \to_{\eta'} S(X[t])$.
5. *Si* $a \to_{\eta'} b$, *alors* $S(a \cdot s) \to_{\eta'} S(b \cdot s)$.
6. *Si* $s \to_{\eta'} t$, *alors* $S(a \cdot s) \to_{\eta'} S(a \cdot t)$.

Lemme 4.6 *Soient* $a \in \Lambda X^t_{\sigma'}$, $s, t \in \Lambda X^s_{\sigma'}$, $b \in \Lambda X^t$ *et* $u \in \Lambda X^s$.

1. *Si* $a \to_{(Eta)} b$, *alors* $\sigma'(a[s]) \to_{\eta'} \sigma'(b[s])$.
2. *Si* $t \to_{(Eta)} u$, *alors* $\sigma'(t \circ s) \to_{\eta'} \sigma'(u \circ s)$.

Démonstration
La preuve est analogue à celle du lemme 3.4, sauf le cas où il y a un radical à la racine qui est essentiellement différent :
Soient donc $a = \lambda(c\mathtt{1})$ et $b \in \Lambda X^t$ tel que $c = \sigma'(c) = \sigma'(b[\uparrow])$. On a

$$\sigma'((\lambda(c\mathtt{1}))[s]) = \lambda(\sigma'(c[\mathtt{1} \cdot (s \circ \uparrow)])\ \mathtt{1}[\mathtt{1} \cdot (s \circ \uparrow)]) = \lambda(\sigma'(c[\mathtt{1} \cdot (s \circ \uparrow)])\ \mathtt{1}).$$

Or, $\sigma'(c[1 \cdot (s \circ \uparrow)]) = \sigma'(b[\uparrow][1 \cdot (s \circ \uparrow)]) = \sigma'(b[s \circ \uparrow]) = \sigma'((b[s])[\uparrow])$.
Donc, $\sigma'((\lambda(c1))[s]) \rightarrow_{(Eta)} b[s]$, i.e. $\sigma'((\lambda(c1))[s]) \rightarrow_{\eta'} \sigma'(b[s])$.
□

Méta-remarque Le corollaire 3.2, les lemmes 3.5 et 3.6 et le corollaire 3.3 sont aussi valables, avec démonstrations identiques, en mettant (Eta) à la place de $(Beta)$, $\rightarrow_{\eta'}$ à la place de $\rightarrow_{\beta'}$ et $\twoheadrightarrow_{\eta'}$ à la place de $\twoheadrightarrow_{\beta'}$. Nous nous rapportons à ces nouvelles versions en ajoutant η à l'ancienne référence : lemmes 3.5η, 3.6η et corollaires 3.2η et 3.3η.

Proposition 4.2 *Soient $f, g \in \Lambda X$ tels que $f \rightarrow_{(Eta)} g$, alors $\sigma'(f) \twoheadrightarrow_{\eta'} \sigma'(g)$.*

Démonstration
Récurrence sur la structure de f. La preuve est analogue à celle de la proposition 3.3, sauf le cas où $f = \lambda a$ et le radical est à la racine, i.e. $a = b1$ et $g = c$ où $\sigma'(b) = \sigma'(c[\uparrow])$.

$$\sigma'(f) = \lambda(\sigma'(b)\ 1) = \lambda(\sigma'(c[\uparrow])\ 1) \rightarrow_{(Eta)} c\,.$$

Donc, $\sigma'(f) \rightarrow_{\eta'} \sigma'(g)$.
□

Corollaire 4.4 *Soient $f, g \in \Lambda X$ tels que $f \twoheadrightarrow_{\lambda\sigma\eta} g$, alors $\sigma'(f) \twoheadrightarrow_{\lambda\eta'} \sigma'(g)$.*

Démonstration
Par récurrence sur la longueur de la dérivation $f \twoheadrightarrow_{\lambda\sigma\eta} g$. Si la dernière règle est *(Beta)*, on utilise la proposition 3.3. Si elle est (Eta), on utilise la proposition précédente. Et si elle est une σ'-règle le résultat est évident.
□

4.4 Confluence quasi-forte de η'

Avant de montrer la confluence quasi-forte de η', nous devons montrer un lemme qui assure que les (Eta)-réduits des termes/substitutions de la forme $\sigma'(a[s_{n\,2}])$ / $\sigma'(s \circ s_{n\,2})$ sont aussi de cette forme-là et que la *(Eta)*-réduction a nécessairement lieu dans a ou dans s (cf. lemme 4.9).

Pour établir ce dernier résultat nous devons vérifier que la condition pour appliquer la règle *(Eta)* est satisfaite pour un terme $c' \in \Lambda X^t_{\sigma'}$ qui satisfait à son tour $\sigma'(c'[s_{n+1\,2}]) = \sigma'(b[\uparrow])$ pour un certain terme b. Nous devons donc montrer que, en présence de ces hypothèses, $c' = \sigma'(e[\uparrow])$ pour un certain terme e (cf. lemme 4.8).

Enfin, ce dernier résultat requiert encore un lemme qui assure la non-existence de solutions de certaines équations. Nous commençons donc par montrer ce résultat.

Lemme 4.7 *Soient $p \geq 0$, $q \geq 0$ et $1 \leq k \leq p+1$.*

1. *Si $p > q \geq k-1$, l'équation $s_{k\,p\,2} = \sigma'(t \circ s_{q\,2})$ n'a pas de solution dans ΛX^s.*

2. *Si* $p < q$, *l'équation* $s_{k\,p\,2} = \sigma'(t \circ s_{q\,2})$ *n'a pas de solution dans* ΛX^s.

3. *L'équation* $s_{p\,2} = \sigma'(t \circ s_{q\,2})$ *n'a pas de solution dans* ΛX^s, *si* $p \neq q$.

Démonstration

Rappelons $s_{k\,p\,2} = \mathtt{k} \cdot \ldots \cdot \mathtt{p} \cdot \uparrow^{p+1}$ (cf. définition 4.2).
Remarquons d'abord qu'il suffit de montrer qu'il n'y a pas de solution dans $\Lambda X^s_{\sigma'}$, car $\sigma'(t \circ s_{q\,2}) = \sigma'(\sigma'(t) \circ s_{q\,2})$.

1. Supposons qu'il y ait une solution $t \in \Lambda X^s_{\sigma'}$.

   (a) Si $t = \uparrow^m$, on étudie la valeur de m :

      i. Si $m < q$, alors $\sigma'(t \circ s_{q\,2}) = s_{m+1\,q\,2}$. Or, l'égalité $s_{k\,p\,2} = s_{m+1\,q\,2}$ est impossible au niveau structurel si $p \neq q$.

      ii. Si $m \geq q$, alors $\sigma'(t \circ s_{q\,2}) = \uparrow^{m+1}$. Donc, $s_{k\,p\,2} = \uparrow^{m+1}$, ce qui est impossible, puisque $s_{k\,p\,2}$ est un cons ($k < p+1$).

   (b) Si $t = a_1 \cdot \ldots \cdot a_j \cdot \uparrow^m$, avec $j \geq 1$, alors $a_j \neq \mathtt{m}$, car $t \in \Lambda X^s_{\sigma'}$. Remarquons que

   $$\sigma'(a_1[s_{q\,2}]) \cdot \ldots \cdot \sigma'(a_j[s_{q\,2}]) \cdot \sigma'(\uparrow^m \circ s_{q\,2}) \in \Lambda X^s_{\sigma'}.$$

   En effet, s'il existe n tel que $\sigma'(a_j[s_{q\,2}]) = \mathtt{n}$ et $\sigma'(\uparrow^m \circ s_{q\,2}) = \uparrow^n$, les lemmes 4.1 et 4.2 assurent $n > q+1$, $a_j = \mathtt{n-1}$ et $m = n-1$, ce qui est absurde, car $a_j \neq \mathtt{m}$. Donc,

   $$\sigma'(t \circ s_{q\,2}) = \sigma'(a_1[s_{q\,2}]) \cdot \ldots \cdot \sigma'(a_j[s_{q\,2}]) \cdot \sigma'(\uparrow^m \circ s_{q\,2}).$$

   Calculons $\sigma'(\uparrow^m \circ s_{q\,2})$ selon la valeur de m :

      i. Si $m < q$, alors $\sigma'(\uparrow^m \circ s_{q\,2}) = \mathtt{m+1} \cdot \ldots \cdot \mathtt{q} \cdot \uparrow^{q+1}$. Donc,

      $$s_{k\,p\,2} = \sigma'(a_1[s_{q\,2}]) \cdot \ldots \cdot \sigma'(a_j[s_{q\,2}]) \cdot \mathtt{m+1} \cdot \ldots \cdot \mathtt{q} \cdot \uparrow^{q+1},$$

      impossible si $p \neq q$.

      ii. Si $m \geq q$, alors $\sigma'(\uparrow^m \circ s_{q\,2}) = \uparrow^{m+1}$. Donc,

      $$s_{k\,p\,2} = \sigma'(a_1[s_{q\,2}]) \cdot \ldots \cdot \sigma'(a_j[s_{q\,2}]) \cdot \uparrow^{m+1}.$$

      On conclut $p = m$, $j = p-k+1$, et comme $k-1 \leq q < p$, on peut égaliser la (q+2-k)-ième composante des deux substitutions : $\mathtt{q+1} = \sigma'(a_{q+2-k}[s_{q\,2}])$. Ceci contredit le lemme 4.1.3.

2. On le montre par récurrence sur $p+1-k$.
   Si $p+1-k = 0$, alors $s_{k\,p\,2} = \uparrow^{p+1}$, et comme $p+1 \leq q$, le lemme 4.2 permet de conclure.
   Supposons donc $p+1-k > 0$ et supposons qu'il existe une solution $t \in \Lambda X^s_{\sigma'}$.

   (a) Si $t = \uparrow^m$, comme $s_{k\,p\,2}$ est un cons, nécessairement $m < q$. Donc,

   $$\mathtt{k} \cdot \ldots \cdot \mathtt{p} \cdot \uparrow^{p+1} = s_{k\,p\,2} = \sigma'(\uparrow^m \circ s_{q\,2}) = \mathtt{m+1} \cdot \ldots \cdot \mathtt{q} \cdot \uparrow^{q+1},$$

   ce qui est absurde, car $p < q$.

(b) Si $t = a \cdot s$, les lemmes 4.1 et 4.2, et $t \in \Lambda X^s_{\sigma'}$ donnent $\sigma'(t \circ s_{q\,2}) = \sigma'(a[s_{q\,2}]) \cdot \sigma'(s \circ s_{q\,2})$. Donc,

$$\mathtt{k} \cdot s_{k+1\,p\,2} = s_{k\,p\,2} = \sigma'(a[s_{q\,2}]) \cdot \sigma'(s \circ s_{q\,2}),$$

d'où $s_{k+1\,p\,2} = \sigma'(s \circ s_{q\,2})$, ce qui est absurde par H.R.

3. C'est une conséquence immédiate de deux résultats précédents avec $k = 1$.

□

Comme nous l'avons dit au commencement de cette section nous envisageons de montrer que pour tout terme $c \in \Lambda X^t_{\sigma'}$, l'existence d'un terme $b \in \Lambda X^t$ tel que $\sigma'(c[s_{n+1\,2}]) = \sigma'(b[\uparrow])$, entraîne l'existence d'un terme e tel que $c = \sigma'(e[\uparrow])$. Si nous essayons une preuve par récurrence sur la structure de c, nous constatons que le traitement du cas $c = \lambda a$ nécessite une hypothèse de récurrence plus forte. Nous montrons donc le lemme suivant plus général.

Lemme 4.8 *Soient $c \in \Lambda X^t_{\sigma'}$ et $s \in \Lambda X^s_{\sigma'}$, et soient i, j tels que $0 \leq j < i$.*

1. *S'il existe $b \in \Lambda X^t_{\sigma'}$ tel que $\sigma'(c[s_{i\,2}]) = \sigma'(b[s_{j\,2}])$, alors il existe $e \in \Lambda X^t_{\sigma'}$ tel que $c = \sigma'(e[s_{j\,2}])$.*

2. *S'il existe $t \in \Lambda X^s_{\sigma'}$ tel que $\sigma'(s \circ s_{i\,2}) = \sigma'(t \circ s_{j\,2})$, alors il existe $u \in \Lambda X^s_{\sigma'}$ tel que $s = \sigma'(u \circ s_{j\,2})$.*

Démonstration
Récurrence simultanée sur la structure de c et de s.

$c = a\,d$: $\sigma'(c[s_{i\,2}]) = \sigma'(a[s_{i\,2}])\,\sigma'(d[s_{i\,2}]) = \sigma'(b[s_{i\,2}])$. Alors b ne peut être qu'une application : $b = b'\,b''$, et on a par H.R. e', $e'' \in \Lambda X^t_{\sigma'}$ tels que $a = \sigma'(e'[s_{j\,2}])$ et $d = \sigma'(e''[s_{j\,2}])$. Il suffit donc de prendre $e = e'\,e''$.

$c = \lambda a$: $\sigma'((c[s_{i\,2}]) = \lambda\sigma'(a[s_{i+1\,2}]) = \sigma'(b[s_{i\,2}])$. Alors b ne peut être qu'une abstraction : $b = \lambda b'$. Donc, $\sigma'(a[s_{i+1\,2}]) = \sigma'(b'[s_{j+1\,2}])$, et comme $j+1 < i+1$, l'H.R. assure l'existemce de e' tel que $a = \sigma'(e'[s_{j+1\,2}])$. Prendre donc $e = \lambda e'$.

$c = \mathtt{m}$: Nous devons montrer que $\mathtt{m} = \sigma'(e[s_{j\,2}])$ a une solution. D'après le lemme 4.1, il suffit de vérifier $m \neq j+1$.
Supposons $m = j+1$. La remarque 3.3 donne $\sigma'((\mathtt{j}+\mathtt{1})[s_{i\,2}]) = \mathtt{j}+\mathtt{1}$. Alors, par hypothèse, $\mathtt{j}+\mathtt{1} = \sigma'(b[s_{j\,2}])$, ce qui contredit le lemme 4.1.

$c = X$: $\sigma'(c[s_{i\,2}]) = X[s_{i\,2}] = \sigma'(b[s_{j\,2}])$. Une simple analyse sur la structure de b, montre que b ne peut être que de la forme $X[v]$ ($b = X$ est exclu, car $j < i$). Comme $\sigma'(v \circ id) \neq id$ (cf. lemme 4.4), on déduit $X[s_{i\,2}] = X[\sigma'(v \circ s_{j\,2})]$. Ceci contredit le lemme 4.7.3. Donc, l'hypothèse n'étant pas satisfaite, il n'y a rien à démontrer.

$c = X[w]$: $\sigma'(c[s_{i\,2}]) \overset{L4.4}{=} X[\sigma'(w \circ s_{i\,2})] = \sigma'(b[s_{j\,2}])$. Encore une analyse sur la structure de b, montre que b ne peut être que de la forme $X[v]$ ($b = X$ est exclu maintenant à cause du lemme 4.7.3). L'H.R. donne u' telle que $w = \sigma'(u' \circ s_{j\,2})$, et il suffit de prendre $e = X[u']$.

$s = \uparrow^m$: On veut montrer que $\uparrow^m = \sigma'(u \circ s_{j\,2})$ a une solution. D'après le lemme 4.2, il suffit de constater $m > j$.
En effet, si $m \leq j$, alors $m < i$, donc $\sigma'(\uparrow^m \circ s_{i\,2}) = s_{m+1\,i\,2} = \sigma'(t \circ s_{j\,2})$, ce qui est impossible par le lemme 4.7.1.

$s = a \cdot w$: Comme $s \in \Lambda X^s_{\sigma'}$, $\sigma'(s \circ s_{i\,2}) = \sigma'(a[s_{i\,2}]) \cdot \sigma'(w \circ s_{i\,,2}) = \sigma'(t \circ s_{j\,2})$.
Si $t = \uparrow^m$, nécessairement $m < j$, car $\sigma'(t \circ s_{j\,2})$ est un cons. Donc,

$$\sigma'(t \circ s_{j\,2}) = s_{m+1\,j\,2} = \mathtt{m} + \mathtt{1} \cdot s_{m+2\,j\,2}\,.$$

Alors $\sigma'(w \circ s_{i\,2}) = s_{m+2\,j\,2}$, qui contredit le lemme 4.7.2.
Si $t = d \cdot v$, l'H.R. permet de conclure.

□

Lemme 4.9 *Soient* $a \in \Lambda X^t_{\sigma'}$, $b \in \Lambda X^t$, $s \in \Lambda X^s_{\sigma'}$ *et* $t \in \Lambda X^s$.

1. *Si* $\sigma'(a[s_{n\,2}]) \rightarrow_{(Eta)} b$, *alors il existe* $e \in \Lambda X^t$ *tel que* $\sigma'(b) = \sigma'(e[s_{n\,2}])$ *et* $a \rightarrow_{(Eta)} e$.
2. *Si* $\sigma'(s \circ s_{n\,2}) \rightarrow_{(Eta)} t$, *alors il existe* $u \in \Lambda X^t$ *tel que* $\sigma'(t) = \sigma'(u \circ s_{n\,2})$ *et* $s \rightarrow_{(Eta)} u$.

Démonstration
Les deux résultats sont montrés par récurrence simultanée sur la structure de a et s. Le seul cas intéressant est quand a est une abstraction.
Soit donc $a = \lambda c$. Alors $\sigma'(a[s_{n\,2}]) = \lambda\sigma'(c[s_{n+1\,2}])$.
Si la réduction est interne l'H.R. permet de conclure. Supposons donc qu'elle a lieu à la racine, i.e. $\sigma'(c[s_{n+1\,2}]) = d\,\mathtt{1}$. Un simple analyse sur la structure de c montre que $c = c'c''$. Alors $\sigma'(c'[s_{n+1\,2}]) = d$ et $\sigma'(c''[s_{n+1\,2}]) = \mathtt{1}$. D'après le lemme 4.1, $c'' = \mathtt{1}$. Donc,

$$\sigma'(a[s_{n\,2}]) = \lambda(\sigma'(c'[s_{n+1\,2}])\,\mathtt{1}) \rightarrow_{(Eta)} b\,.$$

Comme la réduction a lieu à la racine, $\sigma'(c'[s_{n+1\,2}]) = \sigma'(b[\uparrow]) = \sigma'(b[s_{0\,2}])$.
Le lemme précédent garantit donc l'existence de $e \in \Lambda X^t_{\sigma'}$ tel que $c' = \sigma'(e[s_{0\,2}]) = \sigma'(e[\uparrow])$. Alors

$$a = \lambda(c'\,\mathtt{1}) \rightarrow_{(Eta)} e\,.$$

Il suffit donc de montrer $\sigma'(b) = \sigma'(e[s_{n\,2}])$. Nous utiliserons le corollaire 4.1.
On vérifie aisément que $\sigma'(\uparrow \circ s_{n+1\,2}) = \sigma'(s_{n\,2} \circ \uparrow)$. Donc,

$$\sigma'(e[s_{n\,2}][\uparrow]) = \sigma'(e[s_{n\,2} \circ \uparrow]) = \sigma'(e[\uparrow \circ s_{n+1\,2}]) =$$

$$= \sigma'(e[\uparrow][s_{n+1}\,2]) = \sigma'(c'[s_{n+1}\,2]) = \sigma'(b[\uparrow])\,.$$

Le corollaire 4.1 permet donc de conclure.
□

Théorème 4.1 *La réduction η' est quasi-fortement confluente sur $\Lambda X_{\sigma'}$, et donc confluente.*

Démonstration
Nous allons montrer par récurrence sur la structure de $f \in \Lambda X_{\sigma'}$ que si $f \to_{(Eta)} g$ et $f \to_{(Eta)} h$, alors il existe $k \in \Lambda X_{\sigma'}$ tel que $\sigma'(g) \stackrel{\epsilon}{\to}_{\eta'} k$ et $\sigma'(h) \stackrel{\epsilon}{\to}_{\eta'} k$. Ceci est équivalent à la confluence quasi-forte de η'.
Le lemme 1.1 permettra d'établir la confluence.
On n'étudie que les cas où il y a des radicaux.

$f = a\,b$: Trois possibilités :

- Les deux réductions ont lieu dans a, i.e. $a \to_{(Eta)} a'$ et $a \to_{(Eta)} a''$. Alors $g = a'b$ et $h = a''b$. Par H.R. il existe $k' \in \Lambda X^t_{\sigma'}$ tel que $\sigma'(a') \stackrel{\epsilon}{\to}_{\eta'} k'$ et $\sigma'(a'') \stackrel{\epsilon}{\to}_{\eta'} k'$. D'après la remarque 4.2.1, il suffit de prendre $k = k'b$.
- Les deux réductions ont lieu dans b. Analogue au sous-cas précédent.
- Une réduction a lieu dans a et l'autre dans b, i.e.
  $a \to_{(Eta)} a'$, $b \to_{(Eta)} b'$, $g = a'b$ et $h = a\,b'$.
  Alors $b \to_{\eta'} \sigma'(b')$ et $a \to_{\eta'} \sigma'(a')$. Donc,

$$\sigma'(g) = \sigma'(a')\,b \stackrel{R4.2.2}{\to}_{\eta'} \sigma'(a')\,\sigma'(b') \quad \text{et}$$

$$\sigma'(h) = a\,\sigma'(b') \stackrel{R4.2.1}{\to}_{\eta'} \sigma'(a')\,\sigma'(b')\,.$$

$f = \lambda a$: Trois possibilités :

- Les deux réductions ont lieu dans a, alors l'H.R. et la remarque 4.2.3 permettent de conclure.
- Les deux réductions ont lieu à la racine. Evident.
- Une réduction est interne et l'autre à la racine. Alors $a = b\,\mathtt{1}$, $g = c$ où $c \in \Lambda X^t$ satisfait $\sigma'(c[\uparrow]) = b$, et $h = \lambda(d\,\mathtt{1})$ où $b \to_{(Eta)} d$.
  Or, $b = \sigma'(c[\uparrow]) = \sigma'(\sigma'(c)[\uparrow])$. Donc, le lemme précédent assure l'existence d'un terme $e \in \Lambda X^t$ tel que $\sigma'(d) = \sigma'(e[\uparrow])$ et $\sigma'(c) \to_{(Eta)} e$.
  Nous avons donc

$$\sigma'(h) = \lambda(\sigma'(d)\,\mathtt{1}) \to_{\eta'} \sigma'(e) \quad \text{et} \quad \sigma'(c) \to_{\eta'} \sigma'(e)\,.$$

$f = X[s]$: Les deux réductions sont nécessairement internes. On conclut grâce à l'H.R. et la remarque 4.2.4.

$f = a \cdot s$: Si les deux réductions ont lieu dans le même sous-terme principal, on conclut grâce à l'H.R. et les remarques 4.2.5 et 4.2.6.
Autrement, les remarques seules permettent de conclure.

□

4.5 Commutation et confluence

Dans la preuve de commutation nous avons besoin de la version correspondant à $(Beta)$ du lemme 4.9 :

Lemme 4.10 *Soient* $a \in \Lambda X^t_{\sigma'}$, $b \in \Lambda X^t$, $s \in \Lambda X^s_{\sigma'}$ *et* $t \in \Lambda X^s$.

1. *Si* $\sigma'(a[s_{n\,2}]) \rightarrow_{(Beta)} b$, *alors il existe* $e \in \Lambda X^t$ *tel que* $\sigma'(b) = \sigma'(e[s_{n\,2}])$ *et* $a \rightarrow_{(Beta)} e$.

2. *Si* $\sigma'(s \circ s_{n\,2}) \rightarrow_{(Beta)} t$, *alors il existe* $u \in \Lambda X^t$ *tel que* $\sigma'(t) = \sigma'(u \circ s_{n\,2})$ *et* $s \rightarrow_{(Beta)} u$.

Démonstration
Les deux résultats sont montrés par récurrence simultanée sur la structure de a et s . Le seul cas intéressant est quand a est une application.
Soit donc $a = c\,d$. Alors $\sigma'(a[s_{n\,2}]) = \sigma'(c[s_{n\,2}])\sigma'(d[s_{n\,2}])$.
Si la réduction est interne l'H.R. permet de conclure. Supposons donc qu'elle a lieu à la racine, i.e. $\sigma'(c[s_{n\,2}]) = \lambda c'$. En étudiant la forme de c , on arrive a la seule possibilité $c = \lambda c''$, d'où $c' = \sigma'(c''[s_{n+1\,2}])$. Alors

$$\sigma'(a[s_{n\,2}]) = (\lambda\sigma'(c''[s_{n+1\,2}]))\sigma'(d[s_{n\,2}]) \rightarrow_{(Beta)} (\sigma'(c''[s_{n+1\,2}])[\sigma'(d[s_{n\,2}]) \cdot id]) .$$

Or, $a = (\lambda c'')d \rightarrow_{(Beta)} c''[d \cdot id]$. Il suffit donc de prendre $e = c''[d \cdot id]$ et montrer

$$\sigma'(e[s_{n\,2}]) = \sigma'(\sigma'(c''[s_{n+1\,2}])[\sigma'(d[s_{n\,2}]) \cdot id]) .$$

En effet,

$$\sigma'(c''[d \cdot id][s_{n\,2}]) = \sigma'(c''[d[s_{n\,2}] \cdot s_{n\,2}]) \overset{R3.7}{=} \sigma'(c''[s_{n+1\,2} \circ (d[s_{n\,2}] \cdot id)]) =$$

$$= \sigma'(c''[s_{n+1\,2}][d[s_{n\,2}] \cdot id]) = \sigma'(\sigma'(c''[s_{n+1\,2}])[\sigma'(d[s_{n\,2}]) \cdot id]) .$$

□

Théorème 4.2 *Les réductions* β' *et* η' *commutent sur* $\Lambda X_{\sigma'}$.

Démonstration
Il suffit de vérifier les hypothèses du Lemme de Commutation (cf. lemme 1.2). On montre donc par récurrence sur la structure de $f \in \Lambda X_{\sigma'}$ que si $f \to_{(Eta)} g$ et $f \to_{(Beta)} h$, alors il existe $k \in \Lambda X_{\sigma'}$ tel que $\sigma'(g) \stackrel{\epsilon}{\to}_{\beta'} k$ et $\sigma'(h) \twoheadrightarrow_{\eta'} k$.
On n'étudie que les cas où il y a des radicaux.

$f = a\,b$: Quatre possibilités :

- Les deux réductions ont lieu dans a, i.e. $a \to_{(Eta)} a'$ et $a \to_{(Beta)} a''$. Alors $g = a'b$ et $h = a''b$. Par H.R. il existe $k' \in \Lambda X^t_{\sigma'}$ tel que $\sigma'(a') \stackrel{\epsilon}{\to}_{\beta'} k'$ et $\sigma'(a'') \twoheadrightarrow_{\eta'} k'$. D'après la remarque 3.9.1 et le lemme 3.5.1η il suffit de prendre $k = k'b$.
- Les deux réductions ont lieu dans b. Analogue au sous-cas précédent.
- Une réduction a lieu dans a et l'autre dans b, i.e.
  $a \to_{(Eta)} a'$, $b \to_{(Beta)} b'$, $g = a'b$ et $h = a\,b'$.
  Alors $b \to_{\beta'} \sigma'(b')$ et $a \to_{\eta'} \sigma'(a')$. Donc,

$$\sigma'(g) = \sigma'(a')\ b \stackrel{R3.9.2}{\to}_{\beta'} \sigma'(a')\,\sigma'(b') \quad \text{et}$$

$$\sigma'(h) = a\ \sigma'(b') \stackrel{R4.2.1}{\to}_{\eta'} \sigma'(a')\,\sigma'(b')\,.$$

- La $(Beta)$-réduction a lieu à la racine. Alors $a = \lambda c$. Trois sous-cas:
  - La (Eta)-réduction a lieu dans c. Alors $g = (\lambda c')b$ où $c \to_{(Eta)} c'$ et $h = c[b \cdot id]$. D'après le lemme 4.6 ($b \cdot id \in \Lambda X^s_{\sigma'}$), on a

$$\sigma'(c[b \cdot id]) \to_{\eta'} \sigma'(c'[b \cdot id])\,.$$

    Donc, $\sigma'(h) \to_{\eta'} \sigma'(c'[b \cdot id])$.
    D'autre part,

$$\sigma'(g) = (\lambda\sigma'(c'))\sigma'(b)) \to_{\beta'} \sigma'(\sigma'(c')[\sigma'(b) \cdot id]) = \sigma'(c'[b \cdot id])\,.$$

  - La (Eta)-réduction a lieu dans b. Alors $g = (\lambda c)b'$ où $b \to_{(Eta)} b'$ et toujours $h = c[b \cdot id]$.
    Comme dans le cas précédent,

$$\sigma'(g) = (\lambda\sigma'(c))\sigma'(b')) \to_{\beta'} \sigma'(c[b' \cdot id])\,.$$

    Il suffira donc de montrer que $\sigma'(c[b \cdot id]) \twoheadrightarrow_{\eta'} \sigma'(c[b' \cdot id])$.
    Or, $b \to_{\eta'} \sigma'(b')$. Alors (rappelons $b \in \Lambda X^t_{\sigma'}$),

$$b \cdot id = S(b \cdot id) \stackrel{R4.2.5}{\to}_{\eta'} S(\sigma'(b') \cdot id) = \sigma'(b' \cdot id)\,.$$

    Enfin, la version η du corollaire 3.3.1 permet de conclure :

$$\sigma'(c[b \cdot id]) \stackrel{C3.3.1\eta}{\twoheadrightarrow}_{\eta'} \sigma'(c[\sigma'(b' \cdot id)]) = \sigma'(c[b' \cdot id])\,.$$

– La (Eta)-réduction a lieu à la racine de a, i.e. $c = d\,\mathtt{1}$, $g = d'\,b$ où $d = \sigma'(d'[\uparrow])$ et $h = (d\,\mathtt{1})[b \cdot id]$. Donc, puisque $b \in \Lambda X^t_{\sigma'}$, on a :

$$\sigma'(h) = \sigma'(d[b \cdot id])\,b) = \sigma'(d'[\uparrow][b \cdot id])\,b =$$

$$= \sigma'(d'[id])\,b = \sigma'(d')\,b = \sigma'(g)\,.$$

$f = \lambda a$: Si les deux réductions ont lieu dans a, l'H.R., la remarque 3.9.3 et le lemme 3.5.3η permettent de conclure.
Supposons donc que la (Eta)-réduction a lieu à la racine, i.e. $a = b\,\mathtt{1}$. Alors $g = c$ où $b = \sigma'(c[\uparrow])$.
La $(Beta)$-réduction ne peut avoir lieu que dans b. Donc, $h = \lambda(b'\,\mathtt{1})$ où $b \rightarrow_{(Beta)} b'$.
Or, comme $b = \sigma'(c[\uparrow]) = \sigma'((\sigma'(c))[\uparrow]) \rightarrow_{(Beta)} b'$, d'après le lemme précédent, il existe $e \in \Lambda X^t_\sigma$ tel que $\sigma'(b') = \sigma'(e[\uparrow])$ et $\sigma'(c) \rightarrow_{(Beta)} e$. Donc, $\sigma'(c) \rightarrow_{\beta'} \sigma'(e)$.
D'autre part,

$$\sigma'(h) = \lambda(\sigma'(b')\,\mathtt{1}) \rightarrow_{\eta'} \sigma'(e)\,,$$

ce qui permet de conclure.

$f = X[s]$: Les deux réductions ne peuvent avoir lieu qu'à l'intérieur de s, alors l'H.R., la remarque 3.9.4 et le lemme 3.5.4η permettent de conclure.

$f = a \cdot s$: Il y a trois possibilités qui sont anologues aux trois premiers sous-cas du cas $f = a\,b$.

□

Théorème 4.3 *Le $\lambda\eta'$-calcul est confluent sur $\Lambda X_{\sigma'}$.*

Démonstration
C'est une conséquence du théorème de Hindley-Rosen (cf. théorème 1.1).
□

Corollaire 4.5 *Le $\lambda\beta\eta$-calcul, c'est-à-dire le λ-calcul à la de Bruijn extensionnel, est confluent sur Λ.*

Démonstration
On constate aisément que les termes clos sont stables par $\lambda\eta'$-dérivation. Ceci assure la confluence de $\lambda\eta'$ sur Λ. Le corollaire 4.3 permet de conclure.
□

Théorème 4.4 *Le $\lambda\sigma\eta$-calcul est confluent sur ΛX.*

Démonstration
Nous utilisons la méthode d'interprétation (cf. lemme 1.4), en interprétant les termes et substitutions dans $\Lambda X_{\sigma'}$.

Puisque β' peut être simulée dans le $\lambda\sigma'$-calcul (cf. remarque 3.6), elle est aussi simulée dans $\lambda\sigma\eta$. La remarque 4.1 garantit la simulations de η' dans $\lambda\sigma\eta$. D'autre part, l'interprétation de *(Beta)* par β' sur les σ'-formes normales (cf. proposition 3.3) fournit une interprétation de *(Beta)* dans le $\lambda\eta'$-calcul, et la proposition 4.2 donne l'interprétation de *(Eta)* dans $\lambda\eta'$.

Le $\lambda\eta'$-calcul étant confluent (cf. théorème précédent), on a les hypothèses du lemme d'interprétation.
□

Chapitre 5

Deux résultats de complétude

Les résultats de ce chapitre ont été présentés dans [CR91]. Nous donnons ici la version complète de ces résultats.

Dans la section 1.8 nous avons introduit le λ-calcul à la de Bruijn typé et le $\lambda\sigma$-calcul typé et nous avons appelé les théories correspondant à ces calculs **L1** et **S1**, respectivement.

Nous avons aussi énoncé le théorème de correction (cf. théorème 1.15) : si a est simplement typable dans le $\lambda\sigma$-calcul, alors sa σ-forme normale $\sigma(a)$ est simplement typable dans le λ-calcul.

La réciproque n'est pas vraie. Voici un contrexemple : prenons $\mathtt{1}[a : A \cdot s]$, où a est typable et en σ-forme normale, et s n'est pas typable (cela implique donc que $\mathtt{1}[a : A \cdot s]$ n'est pas typable non plus). On a $\sigma(\mathtt{1}[a : A \cdot s]) = a$, donc $\sigma(\mathtt{1}[a : A \cdot s])$ est typable.

Le problème est que la traduction σ perd de l'information (ici s). Dans ce chapitre, nous proposons une autre traduction du $\lambda\sigma$-calcul dans le λ-calcul, basée sur des règles de reécriture qui ne perdent pas d'information. Pour cette traduction nous pouvons prouver la complétude recherchée : un terme du $\lambda\sigma$-calcul est simplement typable si et seulement si sa traduction est simplement typable.

Nous commençons par formuler un ensemble $\mathcal{L}$ de règles dont nous prouvons qu'il est confluent et noethérien (section 5.1). Les formes normales pour ce système sont essentiellement des λ-termes : plus précisément, ce sont des λ-termes dans lesquels des opérations de recalage d'indice ($-[\uparrow]$) peuvent apparaître. Une seconde traduction permet d'obtenir des λ-expressions où ces recalages ont été effectués. Dans la section 5.2 nous présentons cette autre traduction et montrons qu'elle conserve les types. Le résultat principal de ce chapitre est le théorème de complétude 5.2 (section 5.3). Enfin, dans la section 5.4, nous adaptons ce résultat de complétude au $\lambda\sigma_{\Uparrow}$-calcul.

Notation 5.1 *Afin de simplifier la notation, nous utilisons dans ce chapitre la notation des ensembles de termes et substitutions non typés pour les ensembles typés correspondants. On note donc* Λ *, l'ensemble des termes du* λ*-calcul simplement typé*

en notation de de Bruijn; $\Lambda\sigma^t$ *, l'ensemble des termes du* $\lambda\sigma$*-calcul simplement typé;* $\Lambda\sigma^s$ *, l'ensemble des substitutions du* $\lambda\sigma$*-calcul simplement typé et* $\Lambda\sigma = \Lambda\sigma^t \cup \Lambda\sigma^s$ *.*

5.1 Le système $\mathcal{L}$

Définition 5.1 *Le système* $\mathcal{L}$ *est défini sur* $\Lambda\sigma$ *par les règles suivantes :*

$$\begin{array}{ll} (IdEnv) & a[id] \to a \\ (AntiClos) & a[s \circ t] \to a[s][t] \\ (AntiBeta) & a[b : A \cdot s] \to ((\lambda A.a)[s])b\,. \end{array}$$

Théorème 5.1 $\mathcal{L}$ *est noethérien.*

Démonstration
Soit $\mathbf{P}$ la fonction définie par récurrence simultanée sur les termes et substitutions par :

$$\begin{array}{ll} \mathbf{P}(1) = 1 & \mathbf{P}(id) = 1 \\ \mathbf{P}(ab) = \mathbf{P}(a) + \mathbf{P}(b) & \mathbf{P}(\uparrow) = 1 \\ \mathbf{P}(\lambda A.a) = \mathbf{P}(a) & \mathbf{P}(s \circ t) = \mathbf{P}(s) + \mathbf{P}(t) + 1 \\ \mathbf{P}(a[s]) = \mathbf{P}(a) + \mathbf{P}(s) & \mathbf{P}(a : A \cdot s) = \mathbf{P}(a) + \mathbf{P}(s) + 1 \end{array}$$

On vérifie aisément, par récurrence sur les contextes, que, pour tout contexte $C[\,]$, si $\mathbf{P}(x) > \mathbf{P}(y)$, alors $\mathbf{P}(C[x]) > \mathbf{P}(C[y])$. Montrons que $\mathbf{P}(a) > \mathbf{P}(b)$ pour chaque règle $a \to b$ de $\mathcal{L}$:

- $\mathbf{P}(a[id]) = \mathbf{P}(a) + 1 > \mathbf{P}(a)$.
- $\mathbf{P}(a[s \circ t]) = \mathbf{P}(a) + \mathbf{P}(s) + \mathbf{P}(t) + 1 > \mathbf{P}(a) + \mathbf{P}(s) + \mathbf{P}(t) = \mathbf{P}(a[s][t])$.
- $\mathbf{P}(a[b : A \cdot s]) = \mathbf{P}(a) + \mathbf{P}(b) + \mathbf{P}(s) + 1 > \mathbf{P}(a) + \mathbf{P}(b) + \mathbf{P}(s) = \mathbf{P}(((\lambda A.a)[s])b)$.

Donc, une réduction infinie produirait une suite décroissante infinie d'entiers naturels.
□

Corollaire 5.1 $\mathcal{L}$ *est confluent.*

Démonstration
$\mathcal{L}$ est localement confluent, car $\mathcal{L}$ n'a pas de paires critiques; donc (cf. lemme 1.3), noethérianité entraîne confluence.
□

Notation 5.2 $\mathcal{L}(a)$ *notera la* $\mathcal{L}$*-forme normale de* a *dont l'existence et unicité sont assurées par le théorème précedent et son corollaire, respectivement.*

La définition suivante formalise la notion de décalage retardé évoquée plus haut.

Définition 5.2 *L'ensemble de termes* $\Lambda\uparrow$ *est défini par:*

$$a ::= \mathtt{1} \mid ab \mid \lambda A.a \mid a[\uparrow] .$$

Définition 5.3 *On définit* $\mathtt{1}[\uparrow]^n$ *par récurrence sur* n, *ainsi:*

$$\begin{array}{lcl} \mathtt{1}[\uparrow]^1 & = & \mathtt{1}[\uparrow] \\ \mathtt{1}[\uparrow]^{n+1} & = & \mathtt{1}[\uparrow]^n[\uparrow] . \end{array}$$

On identifiera $\mathtt{n}+\mathtt{1}$ *avec* $\mathtt{1}[\uparrow]^n$. *Grâce à cette identification on a* $\Lambda \subset \Lambda\uparrow$.

Proposition 5.1 *Pour tout* $a \in \Lambda\sigma^t$, *si* a *est en* $\mathcal{L}$*-forme normale, alors* $a \in \Lambda\uparrow$. *Donc, l'ensemble de termes en* $\mathcal{L}$*-forme normale est* $\Lambda\uparrow$.

Démonstration
Par récurrence sur la structure du terme a. Le seul cas intéressant est $a = b[s]$, mais si a est en forme normale, alors b est en forme normale et, en plus, $s = \uparrow$ (autrement on aurait un $\mathcal{L}$-redex). Donc, on peut appliquer l' H.R. à b.
Remarquons que l'ensemble de termes en $\mathcal{L}$-forme normale est exactement $\Lambda\uparrow$, car ses éléments sont des formes normales.
□

5.2 La traduction T

Pour définir $T : \Lambda\uparrow \rightarrow \Lambda$, nous nous servirons des fonctions d'actualisation introduites dans le chapitre 1 (cf. défifnition 1.24). Afin d'éviter une surcharge d'indices, nous donnons une autre version de ces fonctions et conservons la même notation. Donc, dans ce chapitre, les fonctions U_i^n sont données par:

Définition 5.4 *On définit les fonctions* $U_i^n : \Lambda \rightarrow \Lambda$ *pour* $i \geq 0$ *et* $n \geq 1$ *par:*

$$\begin{array}{l} U_i^n(ab) = U_i^n(a)\, U_i^n(b) \\ U_i^n(\lambda A.a) = \lambda A.U_{i+1}^n(a) \\ U_i^n(\mathtt{m}) = \left\{ \begin{array}{ll} \mathtt{m}+\mathtt{n} & si\ m > i \\ \mathtt{m} & si\ m \leq i \end{array} \right. \end{array}$$

Définition 5.5 *On définit la fonction de traduction* $T : \Lambda\uparrow \rightarrow \Lambda$ *par:*

$$\begin{array}{l} T(\mathtt{1}) = \mathtt{1} \\ T(ab) = T(a)T(b) \\ T(\lambda A.a) = \lambda A.T(a) \\ T(a[\uparrow]) = U_0^1(T(a)) \end{array}$$

Remarque 5.1 *L'image de T est, en effet, contenue dans Λ. En fait, l'image de T est exactement Λ, car, pour $a \in \Lambda$, on a $T(a) = a$.*

Notation 5.3 *E étant un environnement, $\lg(E)$ notera sa longueur et E_m le m-ième type de E (en partant de la gauche).*

Lemme 5.1 *Pour tout environnement E, tout type A et tout entier $m \geq 1$,*

$$E \vdash_{\mathbf{L1}} \mathtt{m} : A \quad \textit{ssi} \quad \lg(E) \geq m \quad \textit{et} \quad E_m = A\,.$$

Démonstration
Par récurrence sur m. Le cas $\mathtt{m} = \mathtt{1}$ est précisément la règle **L1**-*var*.
Soit $E = E_1, E'$. On a les équivalences suivantes :

$E \vdash_{\mathbf{L1}} \mathtt{m}+\mathtt{1} : A$	ssi (**L1**-*var* n)
$E' \vdash_{\mathbf{L1}} \mathtt{m} : A$	ssi (H.R.)
$\lg(E') \geq m \quad$ et $\quad E'_m = A$	ssi
$\lg(E) \geq m+1 \quad$ et $\quad E_{m+1} = A$.	

□

Lemme 5.2 *Soient $a \in \Lambda$, $\vec{A} = A_1, \ldots, A_i$ et $\vec{B} = B_1, \ldots, B_n$. Alors*

$$\vec{A}, \vec{B}, E \vdash_{\mathbf{L1}} U_i^n(a) : C \quad \textit{ssi} \quad \vec{A}, E \vdash_{\mathbf{L1}} a : C\,.$$

Démonstration
Par récurrence sur la structure de a.

- $a = \mathtt{m}$ et $m > i$ (alors $U_i^n(a) = \mathtt{m}+\mathtt{n}$) :

$\vec{A}, \vec{B}, E \vdash_{\mathbf{L1}} \mathtt{m}+\mathtt{n} : C$	ssi (lemme 5.1)
$\lg(\vec{A}, \vec{B}, E) \geq m+n \quad$ et $\quad (\vec{A}, \vec{B}, E)_{m+n} = C$	ssi
$\lg(\vec{A}, E) \geq m \quad$ et $\quad (\vec{A}, E)_m = C$	ssi (lemme 5.1)
$\vec{A}, E \vdash_{\mathbf{L1}} \mathtt{m} : C$.	

- $a = \mathtt{m}$ et $m \leq i$ (alors $U_i^n(a) = \mathtt{m}$) :

$\vec{A}, \vec{B}, E \vdash_{\mathbf{L1}} \mathtt{m} : C$	ssi (lemme 5.1)
$A_m = C$	ssi (lemme 5.1)
$\vec{A}, E \vdash_{\mathbf{L1}} \mathtt{m} : C$.	

- $a = bc$ (alors $U_i^n(a) = U_i^n(b)\,U_i^n(c)$) :

$\vec{A}, \vec{B}, E \vdash_{\mathbf{L1}} U_i^n(b)\,U_i^n(c) : C$	ssi (**L1**-*app*)
$\vec{A}, \vec{B}, E \vdash_{\mathbf{L1}} U_i^n(b) : D \to C \quad$ et $\quad \vec{A}, \vec{B}, E \vdash_{\mathbf{L1}} U_i^n(c) : D$	ssi (H.R.)
$\vec{A}, E \vdash_{\mathbf{L1}} b : D \to C \quad$ et $\quad \vec{A}, E \vdash_{\mathbf{L1}} c : D$	ssi (**L1**-*app*)
$\vec{A}, E \vdash_{\mathbf{L1}} bc : C$.	

- $a = \lambda C_1.b$ (alors $U_i^n(a) = \lambda C_1.U_{i+1}^n(b)$) :
  $\vec{A}, \vec{B}, E \vdash_{\mathbf{L1}} \lambda C_1.U_{i+1}^n(b) : C$ ssi (**L1**-*lambda*)
  $C = C_1 \to C_2$ et $C_1, \vec{A}, \vec{B}, E \vdash_{\mathbf{L1}} U_{i+1}^n(b) : C_2$ ssi (H.R.)
  $C_1, \vec{A}, E \vdash_{\mathbf{L1}} b : C_2$ ssi (**L1**-*lambda*)
  $\vec{A}, E \vdash_{\mathbf{L1}} \lambda C_1.b : C_1 \to C_2$.

□

Lemme 5.3 *Pour tout $a \in \Lambda\uparrow$, tout environnement E et tout type A,*

$$E \vdash_{\mathbf{S1}} a : A \quad ssi \quad E \vdash_{\mathbf{L1}} T(a) : A .$$

Démonstration
Par récurrence sur la structure de a.

- $a = \mathtt{1}$ (alors $T(a) = \mathtt{1}$) :
  $E \vdash_{\mathbf{S1}} \mathtt{1} : A$ ssi (**S1**-*var*)
  $E_1 = A$ ssi (**L1**-*var*)
  $E \vdash_{\mathbf{L1}} \mathtt{1} : A$.

- $a = bc$ (alors $T(a) = T(b)\,T(c)$) :
  $E \vdash_{\mathbf{S1}} bc : A$ ssi (**S1**-*app*)
  $E \vdash_{\mathbf{S1}} b : B \to A$ et $E \vdash_{\mathbf{S1}} c : B$ ssi (H.R.)
  $E \vdash_{\mathbf{L1}} T(b) : B \to A$ et $E \vdash_{\mathbf{L1}} T(c) : B$ ssi (**L1**-*app*)
  $E \vdash_{\mathbf{L1}} T(b)\,T(c) : A$.

- $a = \lambda B.b$ (alors $T(a) = \lambda B.T(b)$) :
  $E \vdash_{\mathbf{S1}} \lambda B.b : A$ ssi (**S1**-*lambda*)
  $A = B \to C$ et $B, E \vdash_{\mathbf{S1}} b : C$ ssi (H.R.)
  $B, E \vdash_{\mathbf{L1}} T(b) : C$ ssi (**L1**-*lambda*)
  $E \vdash_{\mathbf{L1}} \lambda B.T(b) : B \to C$.

- $a = b[\uparrow]$ (alors $T(a) = U_0^1(T(b))$) :
  $E \vdash_{\mathbf{S1}} b[\uparrow] : A$ ssi (**S1**-*clos* et **S1**-*shift*)
  $E = E_1, E'$ et $E' \vdash_{\mathbf{S1}} b : A$ ssi (H.R.)
  $E' \vdash_{\mathbf{L1}} T(b) : A$ ssi (lemme 5.2 avec $i = 0$ et $n = 1$)
  $E \vdash_{\mathbf{L1}} U_0^1(T(b))$.

□

5.3 Complétude du $\lambda\sigma$-calcul

Nous montrons d'abord que les règles de réduction de $\mathcal{L}$ préservent le typage. Donc, les types seront préservés par passage à $\mathcal{L}$-forme normale. Grâce à ce résultat nous obtenons le théorème de complètude.

Lemme 5.4 *Pour toute règle $a \to b$ de $\mathcal{L}$, tout contexte $C[\,]$ de type terme et tout contexte $D[\,]$ de type substitution,*

1. $E \vdash_{\mathbf{S1}} C[a] : A \quad ssi \quad E \vdash_{\mathbf{S1}} C[b] : A$
2. $E \vdash_{\mathbf{S1}} D[a] \triangleright E' \quad ssi \quad E \vdash_{\mathbf{S1}} D[b] \triangleright E'$.

Démonstration
On montre les deux résultat en même temps par récurrence simultanée sur la structure des contextes.
Le seul cas intéressant est $C[\,] = [\,]$:

(IdEnv) : Alors $a = c[id]$ et $b = c$.
$E \vdash_{\mathbf{S1}} c[id] : A$ sii (**S1**-*clos* et **S1**-*id*) $E \vdash_{\mathbf{S1}} c : A$.

(AntiClos) : Alors $a = c[s \circ t]$ et $b = c[s][t]$.

$E \vdash_{\mathbf{S1}} c[s \circ t] : A$	ssi (**S1**-*clos*)
$E \vdash_{\mathbf{S1}} s \circ t \triangleright E' \quad$ et $\quad E' \vdash_{\mathbf{S1}} c : A$	ssi (**S1**-*comp*)
$E \vdash_{\mathbf{S1}} t \triangleright E'', \quad E'' \vdash_{\mathbf{S1}} s \triangleright E' \quad$ et $\quad E' \vdash_{\mathbf{S1}} c : A$	ssi (**S1**-*clos*)
$E \vdash_{\mathbf{S1}} t \triangleright E'' \quad$ et $\quad E'' \vdash_{\mathbf{S1}} c[s] : A$	ssi (**S1**-*clos*)
$E \vdash_{\mathbf{S1}} c[s][t] : A$.	

(AntiBeta) : Alors $a = c[d : B \cdot s]$ et $b = ((\lambda B.c)[s])d$.

$E \vdash_{\mathbf{S1}} c[d : B \cdot s] : A$	ssi (**S1**-*clos*)
$E \vdash_{\mathbf{S1}} (d : B \cdot s) : B, E' \quad$ et $\quad B, E' \vdash_{\mathbf{S1}} c : A$	ssi (**S1**-*cons*)
$E \vdash_{\mathbf{S1}} d : B \ , \quad E \vdash_{\mathbf{S1}} s \triangleright E' \quad$ et $\quad B, E' \vdash_{\mathbf{S1}} c : A$	ssi (**S1**-*lambda*)
$E \vdash_{\mathbf{S1}} d : B \ , \quad E \vdash_{\mathbf{S1}} s \triangleright E' \quad$ et $\quad E' \vdash_{\mathbf{S1}} \lambda B.c : B \to A$	ssi (**S1**-*clos*)
$E \vdash_{\mathbf{S1}} (\lambda B.c)[s] : B \to A \quad$ et $\quad E \vdash_{\mathbf{S1}} d : B$	ssi (**S1**-*app*)
$E \vdash_{\mathbf{S1}} ((\lambda B.c)[s])d : A$.	

□

Une conséquence immédiate du lemme précédent est la

Proposition 5.2 *Pour tout $a \in \Lambda\sigma^t$, on a l'équivalence :*

$$E \vdash_{\mathbf{S1}} a : A \quad ssi \quad E \vdash_{\mathbf{S1}} \mathcal{L}(a) : A .$$

Théorème 5.2 (Complétude) *Pour tout $a \in \Lambda\sigma^t$, tout environnement E et tout type A,*

$$E \vdash_{\mathbf{S1}} a : A \quad ssi \quad E \vdash_{\mathbf{L1}} T(\mathcal{L}(a)) : A .$$

Démonstration
$E \vdash_{\mathbf{S1}} a : A$ ssi (proposition 5.2) $E \vdash_{\mathbf{S1}} \mathcal{L}(a) : A$. Mais, d'après la proposition 5.1, $\mathcal{L}(a) \in \Lambda\uparrow$, donc le théorème découle du lemme 5.3.
□

5.4 Complétude du $\lambda\sigma_{\Uparrow}$-calcul

Nous finissons ce chapitre en adaptant le résultat de complétude de la section précédente au $\lambda\sigma_{\Uparrow}$-calcul (cf. section 1.8). L'intérêt de ce calcul, rappelons-le, est qu'il est confluent sur tous les termes/substitutions ouverts. Nous donnons la syntaxe pour la version typée et définissons les règles de typage ainsi :

Définition 5.6 *La syntaxe du $\lambda\sigma_{\Uparrow}$-calcul typé est obtenue à partir de celle du $\lambda\sigma$-calcul typé en y ajoutant un constructeur unaire de substitutions $\Uparrow_A$ pour chaque type A. Les règles de typage sont celles de la théorie* **S1** *à laquelle, pour chaque type A, la règle suivante est ajoutée :*

$$(\mathcal{S}\mathbf{1} - \mathit{lift}) \quad \frac{E \vdash s \triangleright E'}{A, E \vdash \Uparrow_A(s) \triangleright A, E'}$$

On nomme $\mathcal{S}\mathbf{1}$ cette théorie. On note $\Lambda\mathcal{S}^t$, $\Lambda\mathcal{S}^s$ l'ensemble des termes et l'ensemble des substitutions, respectivement; tandis que $\Lambda\mathcal{S} = \Lambda\mathcal{S}^t \cup \Lambda\mathcal{S}^s$.

Définition 5.7 *La fonction de traduction $J : \Lambda\mathcal{S} \to \Lambda\sigma$ est donnée par :*

$$\begin{array}{ll} J(\mathtt{1}) = \mathtt{1} & J(id) = id \\ J(ab) = J(a)J(b) & J(\uparrow) = \uparrow \\ J(\lambda A.a) = \lambda A.J(a) & J(a : A \cdot s) = J(a) : A \cdot J(s) \\ J(a[s]) = J(a)[J(s)] & J(s \circ t) = J(s) \circ J(t) \\ J(\Uparrow_A(s)) = \mathtt{1} : A \cdot (J(s) \circ \uparrow) & \end{array}$$

Proposition 5.3 *Soient $f \in \Lambda\mathcal{S}$, E un environnement et F un type, si f est un terme, ou un environnement, si f est une substitution. Alors*

$$E \vdash_{\mathcal{S}\mathbf{1}} f : F \quad \textit{ssi} \quad E \vdash_{\mathbf{S1}} J(f) : F \, .$$

Démonstration
Par récurrence simultanée sur la structure de f .
Etudions, par exemple le cas où $f = \Uparrow_A (s)$:

$$\begin{array}{ll} A, E \vdash_{\mathcal{S}\mathbf{1}} \Uparrow_A (s) \triangleright A, E' & \text{ssi (S}\mathbf{1}\text{-}\mathit{lift}\text{)} \\ E \vdash_{\mathcal{S}\mathbf{1}} s \triangleright E' & \text{ssi (H.R.)} \\ E \vdash_{\mathbf{S1}} J(s) \triangleright E' & \text{ssi (}\mathbf{S1}\text{-}\mathit{comp}\text{) et (}\mathbf{S1}\text{-}\mathit{shift}\text{)} \\ A, E \vdash_{\mathbf{S1}} J(s) \circ \uparrow \triangleright E' & \text{ssi (}\mathbf{S1}\text{-}\mathit{cons}\text{) et (}\mathbf{S1}\text{-}\mathit{var}\text{)} \\ A, E \vdash_{\mathbf{S1}} \mathtt{1} : A \cdot (J(s) \circ \uparrow) \triangleright A, E' \, . & \end{array}$$

□

Le théorème de complétude 5.2 et la proposition précédente nous assurent la complétude du $\lambda\sigma_{\Uparrow}$-calcul :

Théorème 5.3 *Soient $a \in \Lambda\mathcal{S}^t$, E un environnement et A un type. Alors*

$$E \vdash_{\mathcal{S}\mathbf{1}} a : A \quad \textit{ssi} \quad E \vdash_{\mathbf{L1}} T(\mathcal{L}(J(a))) : A \, .$$

Remarque 5.2 *On peut aussi donner une démonstration directe du théorème 5.3 en suivant les lignes de celle du théorème 5.2.*

En effet, on définit d'abord un système de réduction $\mathcal{L}'$ qui a les règles de $\mathcal{L}$ plus la règle

$$(AntiLift) \quad a[\Uparrow_A(s)] \to ((\lambda A.a)[s][\uparrow])\mathtt{1}$$

On démontre la noethérianité de $\mathcal{L}'$ en étendant la fonction de poids $\mathbf{P}$ par

$$\mathbf{P}(\Uparrow_A(s)) = \mathbf{P}(s) + 3\,.$$

On obtient, de même, la confluence de $\mathcal{L}'$ et une démonstration analogue à celle de la proposition 5.1 nous donne:

$$\textit{Pour tout } a \in \Lambda\mathcal{S}^t\,,\ \mathcal{L}'(a) \in \Lambda\uparrow\,.$$

Les lemmes 5.3 et 5.4 et la proposition 5.2 restent valides en substituant $\mathcal{L}$ par $\mathcal{L}'$ et $\mathbf{S1}$ par $\mathcal{S}\mathbf{1}$, et on arrive ainsi à un théorème analogue au théorème 5.2 :

$$\textit{Pour tout } a \in \Lambda\mathcal{S}^t\,,\ E \vdash_{\mathcal{S}\mathbf{1}} a : A \textit{ ssi } E \vdash_{\mathbf{L1}} T(\mathcal{L}'(a)) : A\ .$$

En fait, on démontre par récurrence sur $(\mathrm{pr}(a), \lg(a))$, où $\mathrm{pr}(a)$ est la profondeur de a (longueur de la réduction la plus longue de a à sa $\mathcal{L}'$-forme normale) et $\lg(a)$ sa taille, que

$$\mathcal{L}'(a) = \mathcal{L}(J(a))$$

ce qui fournit une autre démonstration du théorème 5.3.

Chapitre 6

Le $\lambda\nu$-calcul

Dans ce chapitre nous abandonnons la notation de de Bruijn et nous revenons à la notation utilisée habituellement pour dénoter les variables : les noms, i.e. les lettres x, y, z, etc.. Nous essayons donc d'intégrer cette formulation traditionnelle du calcul avec le nouvel ingrédient des substitutions explicites. Deux calculs de ce type ont été proposés dans [ACCL90], où l'on présente aussi les désavantages de ces calculs : absence de formes normales pour les substitutions, pas de bonnes propriétés de confluence. Ces deux calculs conservent la division des termes en deux sortes : termes proprement dits et substitutions. Nous essayons une autre formulation.

Nous voulons inclure dans le langage un opérateur ternaire $-[-/-]$ qui reflète la méta-substitution du λ-calcul le plus fidèlement possible. Nous souhaitons donc que le calcul contienne une règle :

$$(Beta) \quad (\lambda x.a)b \;\rightarrow\; a[b/x]\,.$$

Pour effectuer la substitution, nous devons la propager jusqu'aux variables. Calculer les substitutions sur les variables semble naturel : inclure une règle pour traiter le cas où la variable à être substituée est celle explicitée dans la substitution (cf. la clause 1 de la définition 1.18 [1]) :

$$(Var=) \quad x[b/x] \;\rightarrow\; b\,;$$

et une autre pour le cas où ces deux variables sont différentes (cf. la clause 2) :

$$(Var) \quad y[b/x] \;\rightarrow\; y\,.$$

Il faut introduire une règle qui permette le passage de la substitution sous les abstractions. Nous proposons d'introduire la règle suivante (cf. la clause 5) :

$$(Abs) \quad (\lambda y.a)[b/x] \rightarrow \lambda y.(a[b/x])\,.$$

[1] Les clauses mentionnées dans la suite sont celles de cette définition

Evidemment, cette règle, sans restrictions, introduira le problème de capture des variables. Nous la gardons telle qu'elle a été énoncée et, pour tenter d'éviter les captures, nous nous restreignons à un sous-ensemble de termes.

En fait, il y a deux situations à éviter :

- Celle de la clause 4, c'est-à-dire quand $x = y$. On peut arriver à cette situation ainsi :
$$(\lambda x.(\lambda x.a))b \to (\lambda x.a)[b/x]\,.$$
Il semble naturel d'exiger que toute variable d'un abstracteur soit différente de toutes les variables des autres abstracteurs.

- Celle de la clause 6, c'est-à-dire quand $y \in FV(b)$. On peut arriver à cette situation ainsi :
$$(\lambda x.(\lambda y.a))b \to (\lambda y.a)[b/x]\,.$$
Il semble suffissant d'exiger qu'aucune variable ne soit libre et liée à la fois.

Nous proposons donc d'imposer aux termes la condition, que nous appelons INV, qui consiste en les deux clauses suivantes :

I1. Des abstracteurs différents lient des variables différentes.

I2. Aucune variable n'est libre et liée à la fois.

Pour continuer la propagation de la substitution, nous devons ajouter une règle qui effectue le passage sous l'application. La règle correspondante du λ-calcul classique suggère la règle :

$$(Ap) \quad (a\,b)[c/x] \to (a[c/x])(b[c/x])\,.$$

Or, si l'on garde *(Ap)*, la propriété INV n'est pas préservée par dérivation. Voici un exemple où $I1$ n'est plus satisfait par le dérivé :

$$(\lambda z.zz)(\lambda x.x) \to_{(Beta)} (zz)[\lambda x.x/z] \to_{(Ap)}$$
$$\to_{(Ap)} (z[\lambda x.x/z])\,(z[\lambda x.x/z]) \twoheadrightarrow_{(Var=)} (\lambda x.x)(\lambda x.x)\,.$$

Pour éviter la perte de $I1$ dans cet exemple il suffit de renommer les variables liées de $\lambda x.x$ au moment de la distribution sous l'application zz :

$$(zz)[\lambda x.x/z] \twoheadrightarrow (z[\lambda x_1.x_1/z])(z[\lambda x_2.x_2/z]) \twoheadrightarrow (\lambda x_1.x_1)(\lambda x_2.x_2)\,.$$

Ce renommage permet aussi d'éviter la perte de $I2$ dans un sous-terme du réduit, ce qui pourrait aussi produire une capture. Voici un exemple de cette situation :

$$(\lambda z.zz)(\lambda xy.xy) \to_{(Beta)} ((zz)[\lambda xy.xy/z]) \to_{(Ap)}$$
$$\to_{(Ap)} (z[\lambda xy.xy/z])\,(z[\lambda xy.xy/z]) \twoheadrightarrow_{(Var=)} (\lambda xy.xy)(\lambda xy.xy) \to_{(Beta)}$$
$$\to_{(Beta)} (\lambda y.xy)[\lambda xy.xy/x] \twoheadrightarrow \lambda y.((\lambda xy.xy)y)\,.$$

En effet, le sous-terme $(\lambda xy.xy)y$ du réduit ne satisfait pas $I2$. En revanche, avec le renommage cet exemple devient :

$$(\lambda z.zz)(\lambda xy.xy) \rightarrow_{(Beta)} ((zz)[\lambda xy.xy/z]) \twoheadrightarrow$$
$$\twoheadrightarrow (z[\lambda x_1y_1.x_1y_1/z])\,(z[\lambda x_2y_2.x_2y_2/z]) \twoheadrightarrow_{(Var=)}$$
$$\twoheadrightarrow_{(Var=)} (\lambda x_1y_1.x_1y_1)(\lambda x_2y_2.x_2y_2) \rightarrow_{(Beta)}$$
$$\rightarrow_{(Beta)} (\lambda y_1.x_1y_1)[\lambda x_2y_2.x_2y_2/x_1] \twoheadrightarrow \lambda y_1.((\lambda x_2y_2.x_2y_2)y_1)\,.$$

Maintenant le sous-terme $(\lambda x_2y_2.x_2y_2)y_1$ satisfait bien $I2$.

Ceci suggère l'introduction de deux opérateurs de renommage, que nous dénotons de manière suffixe ainsi : $-\langle 1\rangle$ et $-\langle 2\rangle$. Ils sont activés au moment du passage de la substitution sous l'application par la règle :

$$(App)\quad (ab)[c/u] \;\rightarrow\; (a[c\langle 1\rangle/u])(b[c\langle 2\rangle/u])\,.$$

La fonction essentielle de ces opérateurs est le renommage des variables sous les abstracteurs, ce qui est décrit par la règle :

$$(Absk)\quad (\lambda u.a)\langle k\rangle \;\rightarrow\; \lambda u_k.(a\langle k\rangle[u_k/u])\,.$$

Dans la section 6.1 nous définissons le $\lambda\nu$-calcul en complétant l'ensemble de règles que nous venons d'introduire informellement; nous fixons les notations et nous montrons quelques résultats préliminaires. Dans la section 6.2, nous donnons quelques exemples qui illustrent l'introduction de la condition INV. Nous y constatons que toutes les expectatives ne sont pas satisfaites : même avec l'introduction des opérateurs de renommage et des règles qui les concernent la propriété INV n'est pas préservée par $\lambda\nu$-dérivation. Dans la section 6.3 nous montrons que le $\lambda\nu$-calcul est suffisamment puissant pour reproduire la β-réduction du λ-calcul classique. Dans la section 6.4, nous commençons l'étude du ν-calcul (le $\lambda\nu$-calcul sans la règle *(Beta)*), dont nous montrons la confluence. La section 6.5 est consacrée à la noethérianité du ν-calcul. Enfin, dans la section 6.6, nous définissons une stratégie de réduction qui préserve INV et nous montrons que, en utilisant cette stratégie, le $\lambda\nu$-calcul est correct par rapport au λ-calcul classique.

6.1 Présentation du calcul

Comme nous l'avons dit plus haut nous abandonnons la division en termes proprement dits et substitutions, mais nous retrouvons deux sortes à un autre niveau : une sorte *variable*, sous-sorte d'une sorte *terme*. La substitution est gérée en reflétant le traitement habituel du méta-langage sur la syntaxe. Ainsi, le langage possède un opérateur ternaire $-[-/-]$ de type $\mathit{terme} \times \mathit{terme} \times \mathit{variable} \Rightarrow \mathit{terme}$. L'interprétation du terme $a[b/x]$ est donc le terme a où la substitution de la variable x par le terme b doit être effectuée. Notre syntaxe devient donc très adéquate pour

exprimer l'idée intuitive de substitution et notre règle *(Beta)*, $(\lambda x.a)b \to a[b/x]$ reflète fidèlement la règle β du λ-calcul classique, maintenant exprimée à l'intérieur du langage.

Remarquons que si nous voulons définir un système de réécriture qui contienne les règles *(Var=)* et *(Var)*, en suivant la présentation que nous avons donnée dans la section 1.2, le système obtenu ne sera pas confluent. En effet, la raison est la suivante: *(Var=)* est une instance de *(Var)* et les réduits ne correspondent pas. Cependant, nous pouvons définir une notion de réduction non ambiguë en explicitant la précédence de *(Var=)* par rapport à *(Var)*. Autrement dit, on impose la condition : un *(Var)*-radical peut se réduire s'il n'est pas un *(Var=)*-radical.

Nous commençons par donner la syntaxe du $\lambda\nu$-calcul. Comme nous venons de le dire, les variables liées de certains termes pourront être renommées. Nous partons donc d'un ensemble dénombrable de variables de base, appelons-le V, à partir duquel nous construisons un ensemble de *variables étiquetées*, i.e. variables sur lesquelles quelques renommages ont été effectués. Ces variables seront les variables de notre calcul.

Définition 6.1 *La syntaxe du $\lambda\nu$-calcul est donnée par:*

Etiquettes	$k ::= 1 \mid 2$
Listes d'étiquettes	$\iota ::= nil \mid \iota@k$
Variables	$u ::= x_\iota \quad (x \in V)$
Termes	$a ::= u \mid ab \mid \lambda u.a \mid a[b/u] \mid a\langle k \rangle$

Nous dénotons V_e, l'ensemble des variables (étiquetées). La liste $\iota@k$, souvent notée ιk, est la liste obtenue en ajoutant l'étiquette k à la fin de la liste ι.
L'ensemble des termes du $\lambda\nu$-calcul est noté $\Lambda\nu$.

Notation 6.1 *Nous utilisons les lettres u, v, w pour dénoter des variables étiquetées, tandis que x, y, z dénotent des variables de base. Si $u = x_\iota$, la variable étiquetée u_k abrège $x_{\iota@k}$.*

La liste vide est dénotée ϵ, et convenons $x_\epsilon = x$. La longueur de la liste ι est notée $\lg(\iota)$.

L'ensemble des termes du λ-calcul avec des variable étiquetées est noté Λ_e au lieu de Λ_{V_e}, pour alléger la notation. Nous gardons Λ_V pour l'ensemble des termes du λ-calcul avec des variables sans étiquettes.

Remarque 6.1 *D'après l'identification $x_\epsilon = x$, on a les inclusions suivantes:*

$$\Lambda_V \subset \Lambda_e \subset \Lambda\nu .$$

(Beta)	$(\lambda u.a)b$	$\longrightarrow$	$a[b/u]$
(Var=)	$u[b/u]$	$\longrightarrow$	b
(Var)	$v[b/u]$	$\longrightarrow$	v
(Abs)	$(\lambda v.a)[b/u]$	$\longrightarrow$	$\lambda v.(a[b/u])$
(App)	$(ab)[c/u]$	$\longrightarrow$	$(a[c\langle 1\rangle/u])(b[c\langle 2\rangle/u])$
(Vark)	$u\langle k\rangle$	$\longrightarrow$	u
(Absk)	$(\lambda u.a)\langle k\rangle$	$\longrightarrow$	$\lambda u_k.(a\langle k\rangle[u_k/u])$
(Appk)	$(ab)\langle k\rangle$	$\longrightarrow$	$(a\langle k\rangle)(b\langle k\rangle)$

FIG. 6.1 - Le système de réécrirure $\lambda\nu$

Définition 6.2 *Le $\lambda\nu$-calcul est défini par l'ensemble de règles donné dans la figure 6.1. La règle (Var=) précède la règle (Var), i.e. pour réduire un radical de la forme $v[b/u]$ on compare d'abord les variables u et v : si elles coïncident, le réduit est b, tandis que si elles diffèrent, le réduit est v. Le ν-calcul est le calcul obtenu en privant le $\lambda\nu$-cacul de la règle (Beta).*

Evidemment les termes de Λ_e sont en ν-forme normale. Donc, le théorème suivant caractérise les ν-formes normales comme les termes de Λ_e.

Théorème 6.1 *Soit $a \in \Lambda\nu$ en ν-forme normale, alors $a \in \Lambda_e$.*

Démonstration
Supposons que la thèse n'est pas vraie, alors a contient au moins une clôture ou un renommage.
Si a contient au moins une clôture, prenons une clôture de longueur minimale contenue dans a ; elle sera de la forme $b[c/x]$, où $b = d\langle k\rangle$, car a est en forme normale. Or, a étant en forme normale et la clôture étant de longueur minimale, d est est encore de la forme $d'\langle k\rangle$, et ainsi de suite. Comme b est un terme fini, ce processus doit s'arrêter, i.e. $b = (\cdots(e\langle k_1\rangle)\cdots)\langle k_n\rangle$ où e n'est pas un renommage. Mais tout autre choix pour e est absurde.
Si a ne contient pas de clôtures, alors a contient au moins un renommage, et un raisonnement analogue au précédent nous permet de conclure par l'absurde.
□

Nous rappelons l'ordre préfixe sur les suites :

$$\iota \preceq \kappa \text{ ssi } \lg(\iota) \leq \lg(\kappa) \text{ et } \forall i \leq \lg(\iota)(\iota_i = \kappa_i).$$

La relation de compatibilité, qui s'avère adéquate pour définir la propriété INV dans le cadre plus général des termes avec des variables étiquetées, est donnée par :

$$\iota \prec\!\succ \kappa \text{ ssi } \iota \preceq \kappa \text{ ou } \kappa \preceq \iota.$$

Remarque 6.2 *Voici quelques propriétés évidentes de ces relations :*

1. $\preceq$ *est reflexive et transitive.*
2. *Si* $\iota \preceq \mu$ *et* $\kappa \preceq \mu$, *alors* $\iota \prec\!\!\succ \kappa$.
3. *Si* $\iota \prec\!\!\succ \kappa$, $\iota' \preceq \iota$ *et* $\kappa' \preceq \kappa$, *alors* $\iota' \prec\!\!\succ \kappa'$.

Notation 6.2 *On utilisera la notation* $\lambda u \in b$ *pour abréger* $\exists\gamma\exists a(b/\gamma = \lambda u.a)$ *et on notera* $\lambda u \in_\gamma b$ *quand on voudra mettre en évidence l'occurrence* γ.

La création des abstractions est bien controlée par la $\lambda\nu$-dérivation : les suites d'indices des variables des abstracteurs d'un terme dérivé ne peuvent être que des extensions des suites d'indices des variables des abstracteurs du terme de départ :

Lemme 6.1 *Soient* $z \in V$, $a, b \in \Lambda\nu$, $a \twoheadrightarrow_{\lambda\nu} b$ *et* $\lambda z_\iota \in b$, *alors il existe* $\kappa \preceq \iota$ *tel que* $\lambda z_\kappa \in a$.

Démonstration
On le montre par récurrence sur n, longueur de la dérivation.
Le cas $n = 0$ étant trivial, supposons $a \twoheadrightarrow c \to b$.
Si $\iota = \epsilon$, on constate que pour toute règle on a que si $\lambda z \in b$, alors $\lambda z \in c$, et par H.R., on déduit $\lambda z \in a$.
Si $\iota \neq \epsilon$, soit $\iota = \iota' k$. Si $\lambda z_\iota \in c$, on utilise l'hypothèse de récurrence et si $\lambda z_\iota \notin c$, alors la dernière règle appliquée est nécessairement *(Absk)* et $\lambda z_{\iota'} \in c$, l'H.R. garantit l'existence de $\kappa \preceq \iota'$ tel que $\lambda z_\kappa \in a$ et, $\preceq$ étant transitive, le lemme est démontré.
□

Nous étendons maintenant les définitions d'ensemble de variables et d'ensemble de variables libres aux termes du $\lambda\nu$-calcul et montrons que la $\lambda\nu$-dérivation ne crée pas de nouvelles variables libres.

Définition 6.3 *Pour* $a \in \Lambda\nu$, *on définit* $V_\nu(a)$, *l'* ensemble de variables *de* a, *et* $FV_\nu(a)$, *l'* ensemble des variables libres *de* a, *par récurrence sur la complexité de* a *:*

$V_\nu(u) = \{u\}$	$FV_\nu(u) = \{u\}$
$V_\nu(ab) = V_\nu(a) \cup V_\nu(b)$	$FV_\nu(ab) = FV_\nu(a) \cup FV_\nu(b)$
$V_\nu(\lambda u.a) = \{u\} \cup V_\nu(a)$	$FV_\nu(\lambda u.a) = FV_\nu(a) - \{u\}$
$V_\nu(a[b/u]) = \{u\} \cup V_\nu(a) \cup V_\nu(b)$	$FV_\nu(a[b/u]) = FV_\nu(b) \cup (FV_\nu(a) - \{u\})$
$V_\nu(a\langle k\rangle) = V_\nu(a)$	$FV_\nu(a\langle k\rangle) = FV_\nu(a)$

Remarque 6.3 *Si* $a \in \Lambda_e$, $V_\nu(a)$ *et* $FV_\nu(a)$ *coïncident avec les ensembles des variables et des variables libres définis habituellement.*

Lemme 6.2 *Soient* $a, b \in \Lambda\nu$. *Si* $a \twoheadrightarrow_{\lambda\nu} b$, *alors* $FV_\nu(b) \subset FV_\nu(a)$.

Démonstration
On le montre par récurrence sur n , longueur de la dérivation.
Le cas $n = 0$ est trivial. Si $a \twoheadrightarrow c \to b$, on constate d'abord que, pour chaque $\lambda\nu$-règle $r \to s$, $FV_\nu(s) \subset FV_\nu(r)$, et ensuite on vérifie par récurrence sur la complexité des contextes que, pour tout contexte C , si $FV_\nu(s) \subset FV_\nu(r)$, alors $FV_\nu(C[s]) \subset FV_\nu(C[r])$. On a donc $FV_\nu(b) \subset FV_\nu(c)$ et on conclut par H.R..
□

6.2 Le pseudo-invariant

On veut introduire maintenant une condition sur les termes du λ-calcul pour que les réductions dans le $\lambda\nu$-calcul soient correctes par rapport à la β-réduction. L'idée est la suivante : éviter que la règle *(Abs)* produise des captures des variables. Etudions quelques exemples pour mieux comprendre la situation.

Exemple 6.1 *Réduisons le terme* $(\lambda x x.x)y$ *dans les deux calculs :*

- $(\lambda x x.x)y \to_\beta (\lambda x.x)[\![y/x]\!] = \lambda x.x$,
- $(\lambda x x.x)y \to_{\lambda\nu} (\lambda x.x)[y/x] \to_{\lambda\nu} \lambda x.(x[y/x]) \to_{\lambda\nu} \lambda x.y$.

Cet exemple suggère une première condition à imposer au terme à réduire :

▷ Des abstracteurs différents lient des variables différentes.

Exemple 6.2 *Réduisons le terme* $(\lambda x y.x)y$ *dans les deux calculs :*

- $(\lambda x y.x)y \to_\beta (\lambda y.x)[\![y/x]\!] = \lambda z.y$,
- $(\lambda x y.x)y \to_{\lambda\nu} (\lambda y.x)[y/x] \to_{\lambda\nu} \lambda y.(x[y/x]) \to_{\lambda\nu} \lambda y.y$.

Cet exemple suggère une deuxième condition :

▷ Aucune variable n'est libre et liée à la fois.

Nous voulons généraliser ces idées pour donner une définition d'invariant pour les termes avec des variables étiquetées. Une notion préliminaire, celle de compatibilité, rendra plus claire cette généralisation.

Définition 6.4 *Deux variables étiquetées* u *et* v *sont* compatibles *ssi il existe une variable de base* x *telle que* $u = x_\iota$, $v = x_\kappa$ *et* $\iota \prec\!\!\succ \kappa$.

Définition 6.5 *On dira qu'un terme* $a \in \Lambda_e$ satisfait l' invariant, *et on le note* $a \models INV$, *ssi les deux conditions suivantes sont satisfaites :*

I1. *Des abstracteurs différents lient des variables non compatibles, i.e.*

$$si\ z \in V\ ,\ \lambda z_\iota \in_\gamma a\ ,\ \lambda z_\kappa \in_\delta a\ \ et\ \iota \prec\!\!\succ \kappa\ ,\ alors\ \iota = \kappa\ \ et\ \gamma = \delta\ .$$

I2. *Aucune variable libre n'est compatible avec une variable liée, i.e.*

$$\neg\exists z \in V\, \neg\exists \iota, \kappa\ \ telles\ que\ z_\iota \in FV(a)\ ,\ \lambda z_\kappa \in a\ \ et\ \iota \prec\!\!\succ \kappa\ .$$

Puisque sur les termes sans variables étiquetées les notions de compatibilité et d'égalité coïncident, les conditions de la définition que nous venons de donner sont en fait une généralisation des conditions suggérées plus haut :

Remarque 6.4 *Si* $a \in \Lambda_V$, $a \models INV$ *se réduit à :*

I1. Des abstracteurs différents lient des variables différentes.

I2. Aucune variable n'est libre et liée à la fois.

Remarque 6.5 *Tous les sous-termes d'un terme satisfaisant l'invariant le satisfont aussi.*

Démonstration
Récurrence sur la complexité du terme. Il suffit de montrer :

1. Si $ab \models INV$, alors $a \models INV$ et $b \models INV$.
2. Si $\lambda x_\iota.a \models INV$, alors $a \models INV$.

L'affirmation 1. est évidente. Quant à l'affirmation 2., le seul cas intéressant à considérer est celui où x_ι est libre dans a et $\lambda x_\kappa \in a$ avec $\kappa \prec\!\!\succ \iota$, mais le fait que $\lambda x_\iota.a \models INV$ exclut cette possibilité.
□

La question qui surgit naturellement est de savoir si l'invariant ainsi défini est vraiment invariant par rapport à la $\lambda\nu$-réduction, i.e. si $a \models INV$ et $a \twoheadrightarrow_{\lambda\nu} b$, alors $b \models INV$. La réponse est negative. En effet :

Exemple 6.3 *Perte de* INV *par* $\lambda\nu$*-réduction.*

Démonstration
Soient $c = (\lambda x.xx)(\lambda y.y)$ et $a = (\lambda z.zz)c$.
Evidemment, $a \models INV$. On construira $b \in \Lambda_e$ tel que $a \twoheadrightarrow_{\lambda\nu} b$ et $b \not\models INV$. On commence la réduction :

$$a \to (zz)[c/z] \to z[c\langle 1\rangle/z]\, z[c\langle 2\rangle/z] \twoheadrightarrow c\langle 1\rangle c\langle 2\rangle$$

On continue la réduction séparément dans $c\langle 1\rangle$ et $c\langle 2\rangle$ en suivant deux stratégies différentes :

$$c\langle 1\rangle \twoheadrightarrow (\lambda x_1.x_1x_1)(\lambda y_1.y_1) \twoheadrightarrow (\lambda y_{11}.y_{11})(\lambda y_{12}.y_{12})$$

$$c\langle 2\rangle \twoheadrightarrow ((\lambda y_1.y_1)(\lambda y_2.y_2))\langle 2\rangle \twoheadrightarrow (\lambda y_{12}.y_{12})(\lambda y_{22}.y_{22})$$

On a donc la $\lambda\nu$-réduction suivante :

$$a \twoheadrightarrow c\langle 1\rangle c\langle 2\rangle \twoheadrightarrow ((\lambda y_{11}.y_{11})(\lambda y_{12}.y_{12}))((\lambda y_{12}.y_{12})(\lambda y_{22}.y_{22})) = b$$

On constate que $b \not\models INV$ ($b \not\models$ **I1**), car on trouve dans b, par exemple, deux λy_{12}'s différents.
□

Faut-il changer le pseudo-invariant INV? D'après le contrexemple précédent on aurait envie d'affaiblir la première condition à :

- ▷ Si deux λ's lient des variables égales, alors aucun d'eux ne se trouve sous la portée de l'autre.

Mais l'exemple suivant montre que celle-ci n'est pas une bonne solution pour la conservation de INV par $\lambda\nu$-réduction.

Exemple 6.4 $(\lambda yx.y)(\lambda x.x) \twoheadrightarrow_{\lambda\nu} \lambda x.(\lambda x.x)$

Ce qui s'est passé est que la règle *(Beta)* a fait entrer le deuxième λx sous la portée du premier λx. Donc, trouver la bonne notion d'invariant semble plus compliqué.

Nous gardons quand-même la définition 6.5 et par commodité continuerons à l'appeler "invariant" même s'il ne l'est pas. De toutes façons nous $\lambda\nu$-réduirons en utilisant une stratégie qui conserve l'invariant et montrerons la correction du calcul (cf. théorème 6.5) pour cette stratégie.

6.3 Simulation du λ-calcul dans le $\lambda\nu$-calcul

Le but de cette section est de montrer que toute β-dérivation $a \twoheadrightarrow_\beta b$ peut être simulée par une $\lambda\nu$-dérivation $a' \twoheadrightarrow_{\lambda\nu} b'$ telle que $a \equiv_\alpha a'$, $b \equiv_\alpha b'$, $a' \models INV$ et $b' \models INV$ (cf. théorème 6.2). Ceci montre que le $\lambda\nu$-calcul est suffisamment puissant pour simuler le λ-calcul classique.

Nous montrons d'abord que, sous certaines conditions, la ν-dérivation effectue la méta-substitution du λ-calcul classique quand le terme à substituer est une variable :

Lemme 6.3 *Soient* $b \in \Lambda_e$, $v \notin V(b)$ *et* $\lambda u \notin b$, *alors* $b[v/u] \twoheadrightarrow_\nu b[\![v/u]\!]$.

Démonstration
On le montre par récurrence sur la complexité de b.

- Si $b \in V_e$, le lemme est immédiat.
- Si $b = c\,d$ (alors $v \notin V(c)$, $v \notin V(d)$, $\lambda u \notin c$ et $\lambda u \notin d$) on a :

$$\begin{aligned}(c\,d)[v/u] \to_\nu c[v\langle 1\rangle/u]\,d[v\langle 2\rangle/u] \twoheadrightarrow_\nu c[v/u]\,d[v/u] &\overset{HR}{\twoheadrightarrow}_\nu \\ \twoheadrightarrow_\nu c[\![v/u]\!]\,d[\![v/u]\!] = (c\,d)[\![v/u]\!]&.\end{aligned}$$

- Si $b = \lambda w.c$ (alors $w \neq u$, $w \neq v$, $v \notin V(c)$ et $\lambda u \notin c$) on a :

$$(\lambda w.c)[v/u] \to_\nu \lambda w.(c[v/u]) \overset{HR}{\twoheadrightarrow}_\nu \lambda w.c[\![v/u]\!] = (\lambda w.c)[\![v/u]\!]\,.$$

  La dernière égalité étant justifiée par la clause 5. de la définition 1.18, puisque $w \neq v$.

□

Nous présentons maintenant une stratégie pour réduire les renommages de sorte que l'invariant soit préservé. En plus, les dérivés donnés par cette stratégie satisfont plusieurs propriétés qui seront utiles dans la suite. Plus précisément :

Proposition 6.1 [2] *Si* $b \in \Lambda_e$ *et* $b \models INV$, *alors il existe* $b_k \in \Lambda_e$, $k \in \{1,2\}$ *tels que*

i) $b\langle k\rangle \twoheadrightarrow_\nu b_k$.

ii) $b \equiv_\alpha b_k$.

iii) $FV(b) = FV(b_k)$.

[2] Dans le cas où b est une variable il y a abus de notation. En effet, si $b = u = x_\iota$, la variable u_k dénote (cf. notation 6.1) $x_{\iota k}$, tandis que le b_k fourni par cette proposition sera x_ι. Cependant, le contexte dans lequel nous utiliserons cette notation ambigüe sera toujours suffisamment clair pour faire le bon choix.

iv) $b_k \models INV$.

v) $\neg\exists z \in V,\ \neg\exists \iota, \kappa\ (\iota \prec\!\succ \kappa\ \wedge\ \lambda z_\iota \in b_1\ \wedge\ \lambda z_\kappa \in b_2)$.

vi) $\lambda z_\iota \in b_k\ \Rightarrow\ \exists \iota'\ (\iota = \iota' k\ \wedge\ \lambda z_{\iota'} \in b)$.

Démonstration Par récurrence sur la structure de b .

I) Si $b \in V_e$, alors $b\langle k\rangle \to b$ et il suffit de prendre $b_k = b$.

II) Si $b = cd$, alors $c, d \models INV$ et par H.R. il existe c_k, d_k tels que i)-vi) sont satisfaites. Nous montrons que $b_k = c_k d_k$ satisfait aussi i)-vi).

i) $cd\langle k\rangle \to c\langle k\rangle d\langle k\rangle \twoheadrightarrow c_k d_k$.

ii) $c_k \equiv_\alpha c$ et $d_k \equiv_\alpha d$ entraînent $c_k d_k \equiv_\alpha cd$.

iii) $FV(c_k d_k) = FV(c_k) \cup FV(d_k) = FV(c) \cup FV(d) = FV(cd)$.

iv) Démontrons que $c_k d_k \models INV$.

I1) Soient $\lambda z_\iota \in_\gamma c_k d_k$, $\lambda z_\kappa \in_\delta c_k d_k$, $\iota \prec\!\succ \kappa$.

- Si $\gamma = 1\gamma'$ et $\delta = 1\delta'$, alors $\lambda z_\iota \in_{\gamma'} c_k$ et $\lambda z_\kappa \in_{\delta'} c_k$. Comme $c_k \models INV$, on a $\iota = \kappa$ et $\gamma' = \delta'$. Donc, $\gamma = \delta$.
- Si $\gamma = 2\gamma'$ et $\delta = 2\delta'$, on raisonne comme dans le cas précédent.
- Si $\gamma = 1\gamma'$ et $\delta = 2\delta'$, alors $\lambda z_\iota \in_{\gamma'} c_k$ et $\lambda z_\kappa \in_{\delta'} d_k$. Par H.R. vi), on a $\iota = \iota' k$, $\kappa = \kappa' k$, $\lambda z_{\iota'} \in c$ et $\lambda z_{\kappa'} \in d$, ce qui est absurde car $b \models INV$ et $\iota' \prec\!\succ \kappa'$. Donc, ce cas-ci n'es pas possible.
- Si $\gamma = 2\gamma'$ et $\delta = 1\delta'$, on raisonne comme dans le cas précédent.

I2) Supposons que **I2** n'est pas satisfaite. Alors il existe $z \in V$ et des suites ι, κ telles que $z_\iota \in FV(c_k d_k)$, $\lambda z_\kappa \in c_k d_k$, $\iota \prec\!\succ \kappa$.

- Si $z_\iota \in FV(c_k)$ et $\lambda z_\kappa \in c_k$, absurde, car $c_k \models INV$.
- Si $z_\iota \in FV(d_k)$ et $\lambda z_\kappa \in d_k$, absurde, car $d_k \models INV$.
- Si $z_\iota \in FV(c_k)$ et $\lambda z_\kappa \in d_k$, alors par H.R. iii) $z_\iota \in FV(c)$ et par H.R. vi) $\kappa = \kappa' k$ et $\lambda z_{\kappa'} \in d$. Mais alors $z_\iota \in FV(cd)$ et $\lambda z_{\kappa'} \in cd$, avec $\iota \prec\!\succ \kappa'$, ce qui est absurde car $b \models INV$.
- Si $z_\iota \in FV(d_k)$ et $\lambda z_\kappa \in c_k$, analogue au cas précédent.

v) Supposons qu'il existe $z \in V$, et des suites ι, κ telles que $\iota \prec\!\succ \kappa$, $\lambda z_\iota \in c_1 d_1$ et $\lambda z_\kappa \in c_2 d_2$.

- Si $\lambda z_\iota \in c_1$ et $\lambda z_\kappa \in c_2$, c'est absurde par H.R. v).
- Si $\lambda z_\iota \in d_1$ et $\lambda z_\kappa \in d_2$, c'est analogue au cas précédent.
- Si $\lambda z_\iota \in c_1$ et $\lambda z_\kappa \in d_2$, alors par H.R. vi) on a $\iota = \iota' 1$, $\kappa = \kappa' 2$, $\lambda z_{\iota'} \in c$ et $\lambda z_{\kappa'} \in d$. Mais alors $\iota' \prec\!\succ \kappa'$, ce qui est absurde car $b \models INV$.

- Si $\lambda z_\iota \in d_1$ et $\lambda z_\kappa \in c_2$, c'est analogue au cas précédent.

vi) Soit $\lambda z_\iota \in c_k d_k$. Alors $\lambda z_\iota \in c_k$ ou $\lambda z_\iota \in d_k$, et les deux cas entraînent par H.R. vi) que $\iota = \iota' k$ et que $\lambda z_{\iota'} \in cd$.

III) Si $b = \lambda x_\iota.c$, alors $c \models INV$ et par H.R. il existe c_k tels que i)-vi).

i) $(\lambda x_\iota.c)\langle k\rangle \to \lambda x_{\iota k}.(c\langle k\rangle[x_{\iota k}/x_\iota]) \overset{HR}{\twoheadrightarrow}_\nu \lambda x_{\iota k}.(c_k[x_{\iota k}/x_\iota])$.
Comme $b \models INV$ on sait que pour tout κ tel que $\kappa \prec\!\!\succ \iota$, $\lambda x_\kappa \notin c$. En plus, $\lambda x_\iota \notin c_k$, autrement H.R. vi) permet de conclure $\iota = \iota' k$ et $\lambda x_{\iota'} \in c$, ce qui est absurde. Et encore, $x_{\iota k} \notin V(c_k)$. En effet, si $\lambda x_{\iota k} \in c_k$, par H.R. vi) on aurait $\lambda x_\iota \in c$, absurde car $b \models INV$. Donc, si $x_{\iota k} \in V(c_k)$ nécéssairement $x_{\iota k} \in FV(c_k) = FV(c)$ et alors $x_{\iota k} \in FV(b)$ ce qui contredit $b \models INV$.
On a donc les hypothèses du lemme 6.3 pour conclure

$$c_k[x_{\iota k}/x_\iota] \twoheadrightarrow_\nu c_k[\![x_{\iota k}/x_\iota]\!]$$

Et finalement on a $(\lambda x_\iota.c)\langle k\rangle \twoheadrightarrow_\nu \lambda x_{\iota k}.c_k[\![x_{\iota k}/x_\iota]\!]$.
Donc, on prendra $(\lambda x_\iota.c)_k = \lambda x_{\iota k}.c_k[\![x_{\iota k}/x_\iota]\!]$.

ii) $\lambda x_{\iota k}.c_k[\![x_{\iota k}/x_\iota]\!] \equiv_\alpha \lambda x_\iota.c_k \equiv_\alpha \lambda x_\iota.c$. La première α-équivalence est justifiée par $x_{\iota k} \notin FV(c_k)$ et la deuxième par H.R.

iii) $FV(\lambda x_{\iota k}.c_k[\![x_{\iota k}/x_\iota]\!]) = FV(c_k[\![x_{\iota k}/x_\iota]\!]) - \{x_{\iota k}\} =$
$= FV(c_k) - \{x_\iota, x_{\iota k}\} = FV(c_k) - \{x_\iota\} \overset{HRiii)}{=} FV(c) - \{x_\iota\} = FV(\lambda x_\iota.c)$.

iv) Démontrons maintenant que $\lambda x_{\iota k}.c_k[\![x_{\iota k}/x_\iota]\!] \models INV$

I1) Soient $\lambda z_\mu \in_\gamma \lambda x_{\iota k}.c_k[\![x_{\iota k}/x_\iota]\!]$, $\lambda z_\kappa \in_\delta \lambda x_{\iota k}.c_k[\![x_{\iota k}/x_\iota]\!]$ et $\mu \prec\!\!\succ \kappa$. D'après le lemme A.1, le seul cas intéressant à considérer est $\mu = \iota k$, $\gamma = \epsilon$ et $\lambda z_\kappa \in c_k[\![x_{\iota k}/x_\iota]\!]$, i.e. $\lambda z_\kappa \in c_k$ (lemme A.1). Alors par H.R.vi) on a $\kappa = \kappa' k$ et $\lambda z_{\kappa'} \in c$. Mais, d'après la remarque 6.2.2 on a $\iota \prec\!\!\succ \kappa'$, ce qui est absurde car $b \models INV$. Donc, ce cas-ci ne se produit jamais.

I2) Supposons que **I2** n'est pas satisfaite. Alors il existe $z \in V$, et des suites μ, κ tels que $\mu \prec\!\!\succ \kappa$, $z_\mu \in FV(\lambda x_{\iota k}.c_k[\![x_{\iota k}/x_\iota]\!])$ et $\lambda z_\kappa \in_\gamma \lambda x_{\iota k}.c_k[\![x_{\iota k}/x_\iota]\!]$.
Alors, par le lemme A.2 $z_\mu \in FV(c_k) \overset{HR}{=} FV(c)$. Il y a deux possibilités:

- Si $\gamma = \epsilon$, alors $\kappa = \iota k$ et $z = x$. Alors $\mu \prec\!\!\succ \iota$ et $z_\mu \neq x_\iota$, car $x_\iota \notin FV(c_k[\![x_{\iota k}/x_\iota]\!])$. Donc, $z_\mu \in FV(b)$, ce qui est absurde car $b \models INV$.
- Si $\gamma \neq \epsilon$, alors $\lambda z_\kappa \in c_k[\![x_{\iota k}/x_\iota]\!]$, et par lemme A.1 on a $\lambda z_\kappa \in c_k$, d'où, par H.R. vi) $\kappa = \kappa' k$ et $\lambda z_{\kappa'} \in c$. Mais $\kappa' \prec\!\!\succ \mu$, ce qui est absurde car $c \models INV$.

v) Supposons qu'il existe $z \in V$, et des suites μ, κ tels que $\mu \prec\!\succ \kappa$, $\lambda z_\mu \in_\gamma \lambda x_{\iota 1}.c_1[\![x_{\iota 1}/x_\iota]\!]$ et $\lambda z_\kappa \in_\delta \lambda x_{\iota 2}.c_2[\![x_{\iota 2}/x_\iota]\!]$.

- Si $\gamma = \delta = \epsilon$, on a une contradiction, car $\iota 1 \not\prec\!\succ \iota 2$.
- Si $\gamma \neq \epsilon$ et $\delta \neq \epsilon$, par lemme A.1, on a $\lambda z_\mu \in c_1$ et $\lambda z_\kappa \in c_2$, ce qui est absurde par H.R.
- Si $\gamma = \epsilon$ et $\delta \neq \epsilon$, alors $z_\mu = x_{\iota 1}$ et par lemme A.1 $\lambda z_\kappa \in c_2$. Alors, par H.R., $\kappa = \kappa' 2$ et $\lambda z_{\kappa'} \in c$, ce qui est absurde car $\kappa' \prec\!\succ \iota$ et $b \models INV$.
- Si $\delta = \epsilon$ et $\gamma \neq \epsilon$, c'est analogue au cas précédent.

vi) Supposons $\lambda z_\kappa \in_\gamma \lambda x_{\iota k}.c_k[\![x_{\iota k}/x_\iota]\!]$.

- Si $\gamma = \epsilon$, alors $z = x$ et $\kappa = \iota k$. Puisque $\lambda x_\iota \in \lambda x_\iota.c$, ce cas est évident.
- Si $\gamma \neq \epsilon$, par lemme A.1 on a que $\lambda z_\kappa \in c_k$, et par H.R. on sait que $\kappa = \kappa' k$ et $\lambda z_{\kappa'} \in c$, d'où $\lambda z_{\kappa'} \in \lambda z_\iota.c$.

□

La proposition suivante est un cas particulier du théorème 6.2 que nous avons annoncé au commencement de cette section. Plus précisément, le cas où une seule β-contraction a lieu à la racine.

Proposition 6.2 *Soient $x \in V$ et $a, b \in \Lambda_e$. Si $(\lambda x_\iota.a)b \models INV$, alors il existe $t \in \Lambda_e$ tel que $a[b/x_\iota] \twoheadrightarrow_\nu t$, $t \equiv_\alpha a[\![b/x_\iota]\!]$ et $t \models INV$.*

Démonstration
On procède par récurrence sur la structure de a.

I) Si $a \in V_e$, c'est immédiat.

II) Si $a = c\,d$, alors

$$(c\,d)[b/x_\iota] \to c[b\langle 1\rangle/x_\iota]\,d[b\langle 2\rangle/x_\iota] \twoheadrightarrow_\nu c[b_1/x_\iota]\,d[b_2/x_\iota]$$

où les b_k sont ceux donnés par la proposition précédente.
Pour appliquer l'H.R. il faut $(\lambda x_\iota.c)b_1 \models INV$ et $(\lambda x_\iota.d)b_2 \models INV$. Démontrons la première condition (la deuxième est analogue).

I1) Si **I1** n'est pas satisfaite, alors il y a deux cas intéressants :

- Si $\lambda x_\kappa \in b_1$ et $\kappa \prec\!\succ \iota$, alors $\kappa = \kappa' 1$ et $\lambda x_{\kappa'} \in b$, ce qui est absurde, car $(\lambda x_\iota.a)b \models INV$ et $\kappa' \prec\!\succ \iota$.
- Si $\lambda z_\kappa \in b_1$, $\lambda z_\mu \in c$ et $\kappa \prec\!\succ \mu$, c'est analogue au cas précédent.

I2) Si **I2** n'est pas satisfaite, alors il y a trois cas intéressants :

- Si $x_\kappa \in FV(b_1)$ et $\kappa \prec\!\succ \iota$, alors $x_\kappa \in FV(b)$, ce qui est absurde car $(\lambda x_\iota.a)b \models INV$.

- Si $z_\kappa \in FV(b_1)$, $\lambda z_\mu \in c$ et $\kappa \prec\!\!\diamond \mu$, c'est analogue au cas précédent.
- Si $z_\kappa \in FV(c) - \{x_\iota\}$, $\lambda z_\mu \in b_1$ et $\kappa \prec\!\!\diamond \mu$, alors $z_\kappa \in FV(\lambda x_\iota.a)$, $\mu = \mu'1$ et $\lambda z_{\mu'} \in b$, ce qui est encore absurde.

Alors on peut conclure, en utilisant H.R., que $cd[b/x_\iota] \twoheadrightarrow c'd'$, où $c' \equiv_\alpha c[\![b_1/x_\iota]\!]$ et $d' \equiv_\alpha d[\![b_2/x_\iota]\!]$. Et encore $c'd' \equiv_\alpha c[\![b/x_\iota]\!]d[\![b/x_\iota]\!] = cd[\![b/x_\iota]\!]$, car $b_k \equiv_\alpha b$.
Vérifions maintenant que $c'd' \models INV$. Par H.R. on sait que $c' \models INV$ et que $d' \models INV$.

I1) Soient $z \in V$ et μ, κ des suites tels que $\lambda z_\mu \in_\gamma c'd'$, $\lambda z_\kappa \in_\delta c'd'$ et $\kappa \prec\!\!\diamond \mu$. Il n'y a que deux cas intéressants et analogues :

1) $\lambda z_\mu \in c'$ et $\lambda z_\kappa \in d'$.
2) $\lambda z_\mu \in d'$ et $\lambda z_\kappa \in c'$.

Analysons le premier :
D'après le lemme 6.1 on sait qu'il existe $\mu' \preceq \mu$ et $\kappa' \preceq \kappa$, et donc $\mu' \prec\!\!\diamond \kappa'$, telles que $\lambda z_{\mu'} \in c[b_1/x_\iota]$ et $\lambda z_{\kappa'} \in d[b_2/x_\iota]$.
Il y a quatre possibilités :

- Si $\lambda z_{\mu'} \in c$ et $\lambda z_{\kappa'} \in d$, alors on a une contradiction car $a \models INV$.
- Si $\lambda z_{\mu'} \in c$ et $\lambda z_{\kappa'} \in b_2$, alors $\kappa' = \kappa''2$ et $\lambda z_{\kappa''} \in b$, ce qui est absurde car $(\lambda x_\iota.a)b \models INV$ et $\kappa'' \prec\!\!\diamond \mu'$.
- Si $\lambda z_{\mu'} \in b_1$ et $\lambda z_{\kappa'} \in d$, analogue au cas précédent.
- Si $\lambda z_{\mu'} \in b_1$ et $\lambda z_{\kappa'} \in b_2$, c'est absurde d'après la proposition 6.1 v).

Donc, ce cas ne peut pas se produire.

I2) Supposons qu'il existe des suites μ, κ et $z_\mu \in FV(c'd')$ telles que $\lambda z_\kappa \in c'd'$ et $\mu \prec\!\!\diamond \kappa$. Il n'y a que deux cas intéressants qui sont d'ailleurs analogues :

1) $z_\mu \in FV(c')$ et $\lambda z_\kappa \in d'$.
2) $z_\mu \in FV(d')$ et $\lambda z_\kappa \in c'$.

Analysons le premier :
D'après le lemme 6.2, $z_\mu \in FV(c[b_1/x_\iota]) = (FV(c) - \{x_\iota\}) \cup FV(b_1)$, et le lemme 6.1 garantit l'existence d'une suite $\kappa' \preceq \kappa$ telle que $\lambda z_{\kappa'} \in d[b_2/x_\iota]$.
Il y a quatre possibilités :

- Si $z_\mu \in FV(c) - \{x_\iota\}$ et $\lambda z_{\kappa'} \in d$, absurde car $cd \models INV$.
- Si $z_\mu \in FV(b_1) = FV(b)$ et $\lambda z_{\kappa'} \in d$, absurde car $(\lambda x.a)b \models INV$.
- Si $z_\mu \in FV(c) - \{x_\iota\}$ et $\lambda z_{\kappa'} \in b_2$, alors il existe $\kappa'' \preceq \kappa'$ telle que $\lambda z_{\kappa''} \in b$ ce qui est aussi absurde car $(\lambda x.a)b \models INV$.
- Si $z_\mu \in FV(b_1) = FV(b)$ et $\lambda z_{\kappa'} \in b_2$, absurde car $b \models INV$.

III) Si $a = \lambda y_\kappa.c$, alors $(\lambda y_\kappa.c)[b/x_\iota] \to \lambda y_\kappa.c[b/x_\iota]$. Pour appliquer H.R. il faut montrer que $(\lambda x_\iota.c)b \models INV$ avec l'hypothèse $(\lambda x_\iota.(\lambda y_\kappa.c))b \models INV$

I1) $(\lambda x_\iota.c)b \models \mathbf{I1}$ est évident en tenant compte de l'hypothèse.

I2) Supposons qu'il existe des suites μ, ξ et $z \in V$ telles que

$$z_\mu \in FV((\lambda x_\iota.c)b) \ , \quad \lambda z_\xi \in (\lambda x_\iota.c)b \quad \text{et} \quad \mu \nsim \xi \, .$$

Il y a deux cas intéressants :

- Si $z_\mu \in FV(b)$ et $\lambda z_\xi \in c$, on a une contradiction à cause de l'hypothèse.
- Si $z_\mu \in FV(c) - \{x_\iota\}$ et $\lambda z_\xi \in b$, il y a deux cas à considrer :
  - ⋆ Si $z_\mu \neq y_\kappa$, alors $z_\mu \in FV(\lambda x_\iota.(\lambda y_\kappa.c))$, impossible par l' hypothèse.
  - ⋆ Si $z_\mu = y_\kappa$, alors $z = y$ et $\mu = \kappa$ d'où $\lambda y_\xi \in b$ et $\xi \nsim \kappa$ ce qui est aussi impossible d'après l'hypothèse.

On peut donc appliquer l'H.R. pour obtenir

$$\lambda y_\kappa.c[b/x_\iota] \twoheadrightarrow_\nu \lambda y_\kappa.c' \quad \text{avec} \quad c[b/x_\iota] \twoheadrightarrow c' \quad \text{et} \quad c' \equiv_\alpha c[\![b/x_\iota]\!]$$

Alors $\lambda y_\kappa.c' \equiv_\alpha \lambda y_\kappa.c[\![b/x_\iota]\!] = (\lambda y_\kappa.c)[\![b/x_\iota]\!]$ car $y_\kappa \notin FV(b)$.
Il nous reste à démontrer que $\lambda y_\kappa.c' \models INV$ en sachant $c' \models INV$.

I1) Supposons $\lambda y_\mu \in c'$ avec $\mu \nsim \kappa$. D'après le lemme 6.1 il existe $\mu' \preceq \mu$ telle que $\lambda y_{\mu'} \in c[b/x_\iota]$. Les deux options $\lambda y_{\mu'} \in c$ et $\lambda y_{\mu'} \in b$ sont impossibles à cause de l'hypothèse.

I2) Supposons $y_\mu \in FV(c')$ avec $\mu \nsim \kappa$ et $\mu \neq \kappa$. Le lemme 6.2 nous assure

$$y_\mu \in FV(c[b/x_\iota]) = FV(b) \cup (FV(c) - \{x_\iota\}) \, .$$

Et les deux options $y_\mu \in FV(b)$ et $y_\mu \in FV(c) - \{x_\iota\}$ sont aussi impossibles à cause de l'hypothèse.

□

Nous avons donc démontré la partie cruciale du théorème de simulation. Nous généralisons la proposition précédente au cas où la β-contraction à lieu à l'intérieur du terme.

Proposition 6.3 *Soient $a, b \in \Lambda_e$ tels que $a \models INV$ et $a \to_\beta b$, alors il existe $t \in \Lambda_e$ tel que $a \twoheadrightarrow_{\lambda\nu} t$, $t \equiv_\alpha b$ et $t \models INV$.*

Démonstration
Par récurrence sur la structure de a.

I) Si $a \in V_e$, il n'y a rien à démontrer.

II) Si $a = c\,d$, il y a trois cas selon l'occurrence du radical :

1. Si $c \to_\beta c'$, par H.R. il existe t' tel que $c \twoheadrightarrow_{\lambda\nu} t'$, $t' \equiv_\alpha c'$ et $t' \models INV$. Il suffit de prendre $t = t'd$. En suivant la méthode de la proposition précédente on vérifie sans difficulté $t \models INV$.
2. Si $d \to_\beta d'$, c'est analogue au cas précédent.
3. Si $c = \lambda x_\iota.e$ et $b = e[\![d/x_\iota]\!]$, on a

$$a = (\lambda x_\iota.e)d \to_{\lambda\nu} e[d/x_\iota] \overset{P6.2}{\twoheadrightarrow}_{\lambda\nu} t\,,$$

où $t \equiv_\alpha e[\![d/x_\iota]\!]$ et $t \models INV$.

III) Si $a = \lambda x_\iota.c$, il existe d tel que $c \to_\beta d$ et $b = \lambda x_\iota.d$. Par H.R. il existe t' tel que $c \twoheadrightarrow_{\lambda\nu} t'$, $t' \equiv_\alpha b$ et $t' \models INV$. Il suffit de prendre $t = \lambda x_\iota.t'$. La méthode de la proposition précédente permet encore de vérifier $t \models INV$.

□

Remarque 6.6 *Pour tout terme $a \in \Lambda_e$, il existe $b \in \Lambda_e$ tel que $a \equiv_\alpha b$ et $b \models INV$.*

Démonstration
Par récurrence sur a. Le cas délicat est $a = c\,d$. Par H.R. il existe $c', d' \in \Lambda_e$ tels que c', $d' \models INV$. Chaque $\lambda x_\iota \in c'$ tel que $\lambda x_\kappa \in d'$ ou $x_\kappa \in FV(d')$ avec $\iota \prec\!\!\succ \kappa$ doit être α-converti. On choisit pour cette α-conversion une variable fraîche non étiquetée qui ne soit ni libre ni liée dans $c'd'$. On est sûr donc que la clause 6 de la définition de méta-substitution (cf. définition 1.18) ne sera jamais utilisée, de sorte qu'il n'y aura pas d'apparition de nouvelles variables pendant ce procedé.
□

Un raisonnement par récurrence sur la quantité de β-contractions dans la β-dérivation fournit la preuve du théorème de simulation.

Théorème 6.2 *Soient $a, b \in \Lambda_e$ tels que $a \twoheadrightarrow_\beta b$, alors il existe a', $b' \in \Lambda_e$ tels que $a' \twoheadrightarrow_{\lambda\nu} b'$, $a' \equiv_\alpha a$, $b' \equiv_\alpha b$, $a' \models INV$ et $b' \models INV$.*
Si $a \models INV$, on peut choisir $a' = a$.

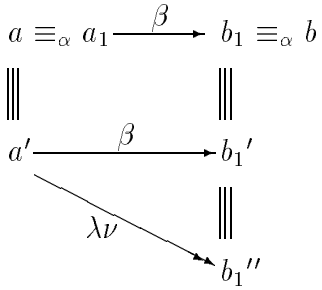
Démonstration
Rappelons d'abord que $\twoheadrightarrow_\beta = (\beta \cup \equiv_\alpha)^*$ (cf. définition 1.21). Donc, puisque la α-congruence contient l'identité syntaxique, la dérivation $a \twoheadrightarrow_\beta b$ peut s'ecrire ainsi :

$$a \equiv_\alpha a_1 \to_\beta b_1 \equiv_\alpha a_2 \to_\beta \ldots \to_\beta b_n \equiv_\alpha b\,.$$

Nous montrons le théorème par récurrence sur n.
Si $n = 1$, alors $a \equiv_\alpha a_1 \to_\beta b_1 \equiv_\alpha b$. D'après la remarque précédente il existe a' tel que $a' \models INV$ et $a' \equiv_\alpha a$. Le lemme A.7 fournit ${b_1}' \in \Lambda_e$ tel que $a' \to_\beta {b_1}'$ et

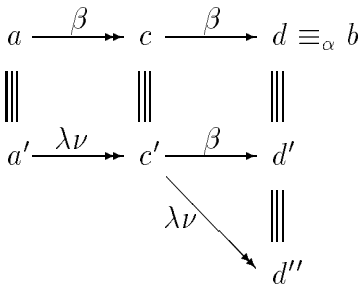
$b_1' \equiv_\alpha b_1$.
La proposition précédente donne b_1'' tel que $a' \twoheadrightarrow_{\lambda\nu} b_1''$, $b_1'' \models INV$ et $b_1'' \equiv_\alpha b_1'$. Donc, il suffit de prendre $b' = b_1''$.

Le diagramme suivant illustre cette situation :



Il nous reste donc à montrer l'étape inductive.

Supposons que $a \overset{n}{\twoheadrightarrow}_\beta c \rightarrow_\beta d \equiv_\alpha b$. L'hypothèse de récurrence assure l'existence de $a', c' \in \Lambda_e$ tel que $a' \twoheadrightarrow_{\lambda\nu} c'$, $a' \equiv_\alpha a$, $c' \equiv_\alpha c$ et $a', c' \models INV$. Le lemme A.7 nous fournit $d' \in \Lambda_e$ tel que $c' \rightarrow_\beta d'$ et $d' \equiv_\alpha d$.
Le théorème étant déjà démontré pour $n = 1$, on a $d'' \in \Lambda_e$ tel que $c' \twoheadrightarrow_{\lambda\nu} d''$, $d'' \equiv_\alpha d'$ et $d'' \models INV$. Donc, $a' \twoheadrightarrow_{\lambda\nu} d''$, $d'' \equiv_\alpha b$, et il suffit de prendre $b' = d''$. Le diagramme suivant illustre cette situation :



□

6.4 Confluence du ν-calcul

La stratégie de réduction que nous présentons dans la section 6.6 nécessite l'existence et unicité des ν-formes normales. Afin d'obtenir l'unicité nous montrons la confluence du ν-calcul. Pour cela, nous divisons le calcul en deux sous-calculs : le ν_1-calcul qui gère les renommages, et le ν_2-calcul qui gère les substitutions. Nous montrons d'abord que ces deux réductions commutent. Pour pouvoir appliquer le théorème de Hindley-Rossen (cf. théorème 1.1), il suffit de montrer la confluence des deux calculs séparément. Notre procédé est le même pour les deux calculs : nous montrons la confluence locale par analyse par cas, et la noethérianité, en exhibant des fonctions de poids adéquates, et nous concluons grâce au lemme de Newman (cf. lemme 1.3).

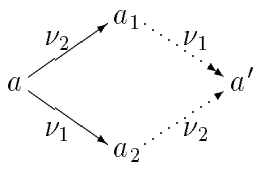
Puisque dans la section suivante nous montrons la noethérianité du ν-calcul, il suffit en fait de montrer la confluence locale, ce qui en principe semble plus économique. Mais la seule économie est au niveau des preuves de normalisation de ν_1 et ν_2, qui sont très simples. En effet, une preuve de confluence locale requiert l'analyse de tous les paires critiques de ν_1 (testés pour la confluence locale de ν_1), de ν_2 (testés pour la confluence locale de ν_2) et de ceux engendrés par l'interaction des règles de ν_1 avec des règles de ν_2 (testés pour la commutation). On gagne donc, en divisant une preuve très longue en trois, un résultat supplémentaire : la commutation de deux sous-systèmes.

Définition 6.6 *On définit deux sous-calculs du ν-calcul :*

1. *Le ν_1-calcul, dont les règles sont (App1), (Abs1), (Var1), (App2), (Abs2) et (Var2).*
2. *Le ν_2-calcul, dont les règles sont (App), (Abs), (Var=), (Var).*

Proposition 6.4 *Les réductions ν_1 et ν_2 commutent.*

Démonstration
D'après le lemme 1.2, il suffit de montrer

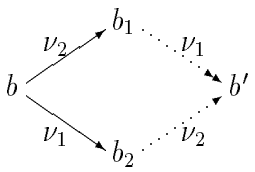


On le fera par récurrence sur la complexité de a.

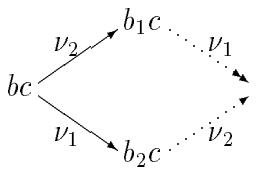
I) Si $a \in V_e$, alors a est en ν_1-forme normale et il n'y a rien à démontrer.

II) Si $a = bc$, il y a quatre possibilités :

i) Les deux réductions ont lieu dans b. Par H.R. on obtient b' tel que



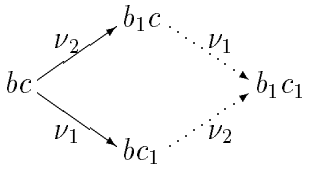
. Donc,



.

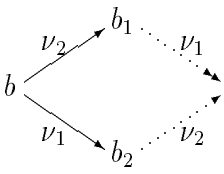
ii) Les deux réductions ont lieu dans c. On raisonne comme dans le cas précédent.

iii) La ν_2-réduction a lieu dans b et la ν_1-réduction a lieu dans c. Dans ce cas on complète le diagramme de la façon suivante :

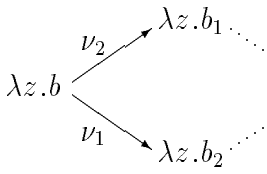


iv) La ν_2-réduction a lieu dans c et la ν_1-réduction a lieu dans b. Analogue au cas précédent.

III) Si $a = \lambda z.b$, les deux réductions auront lieu dans b. Par H.R. on obtient b' tel que



. Donc,

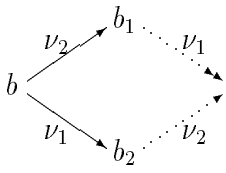


.

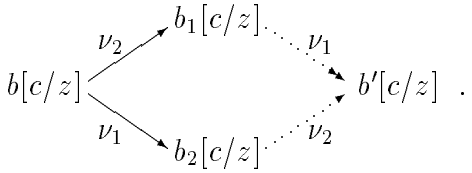
IV) Si $a = b[c/z]$, on considère les deux possibilités suivantes :

i) Le ν_2-radical n'est pas à la racine. Alors il y a quatre sous-cas à considérer :

i.1) Si les deux réductions ont lieu dans b, alors par H.R. on obtient b' tel que



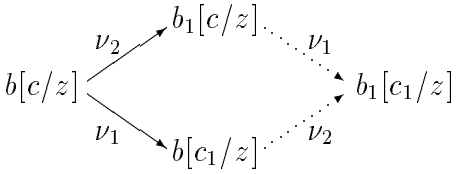
. Donc,



.

i.2) Si les deux réductions ont lieu dans c, c'est analogue au cas précédent.

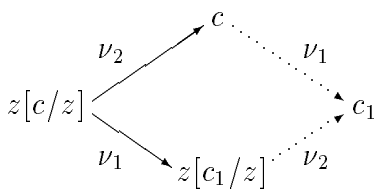
i.3) Si la ν_2-réduction a lieu dans b et la ν_1-réduction a lieu dans c, alors on complète le diagramme de la façon suivante :



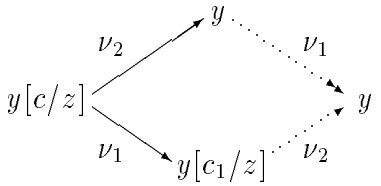
i.4) Si la ν_2-réduction a lieu dans c et la ν_1-réduction a lieu dans b, c'est analogue au cas précédent.

ii) Le ν_2-radical est à la racine. Alors il y a quatre sous-cas à considérer selon la règle du ν_2-calcul qu'on utilise :

ii.1) Si la règle est *(Var=)*, le diagramme se complète comme suit :

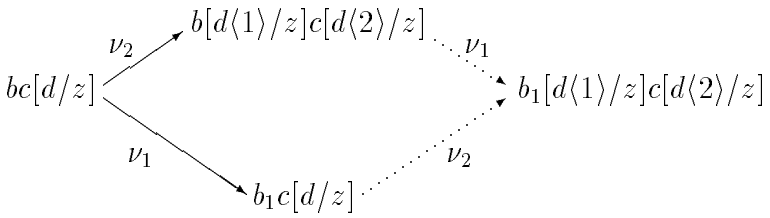


ii.2) Si la règle est *(Var)*, le diagramme se complète comme suit :



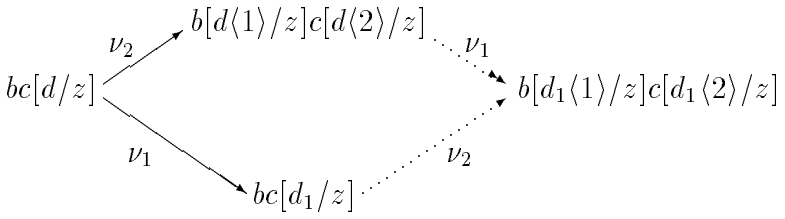
ii.3) Si la règle est *(App)*, il y a trois possibilités selon le sous-terme où la ν_1-réduction aura lieu :

ii.3.1) Si la ν_1-réduction a lieu dans la première composante de l'application, alors on complète le diagramme :



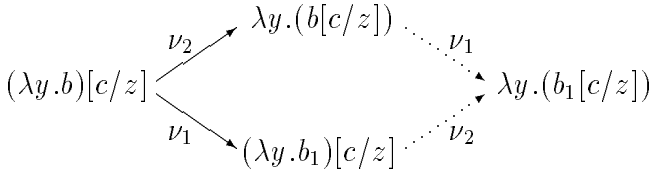
ii.3.2) Si la ν_1-réduction a lieu dans la deuxième composante de l'application, on procède de manière analogue au cas précédent.

ii.3.3) Si la ν_1-réduction a lieu sous la substitution, on complète le diagramme de la façon suivante :

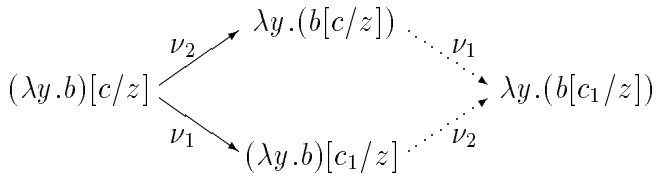


ii.4) Si la règle est *(Abs)*, il y a deux possibilités selon le sous-terme où la ν_1-réduction aura lieu :

ii.4.1) Si la ν_1-réduction a lieu sous l'abstraction, on ferme le diagramme :

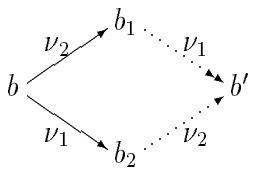


ii.4.2) Si la ν_1-réduction a lieu sous la substitution, on complète le diagramme :

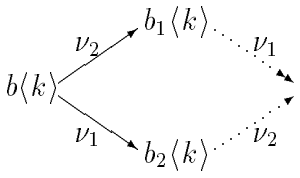


V) Si $a = b\langle k\rangle$, on considère les deux possibilités suivantes :

i) Le ν_1-radical n'est pas à la racine. Et comme le ν_2-radical doit se trouver dans b, par H.R. on obtient b' tel que



. Donc,

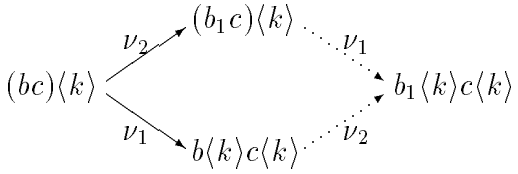


.

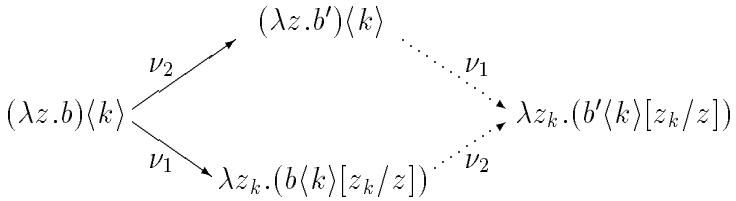
ii) Le ν_1-radical est à la racine. Alors il y a trois sous-cas à considérer selon la règle du ν_1-calcul qu'on utilise :

ii.1) Le cas $b \in V_e$ est impossible car dans ce cas $b\langle k\rangle$ est en ν_2-forme normale.

ii.2) Si la règle utilisée est *(Appk)*, il y a deux sous-cas analogues selon la composante de l'application où se trouve le ν_2-radical. Complétons le diagramme pour l'un de ces cas :



ii.3) Si la règle utilisée est *(Absk)*, la ν_2-réduction aura lieu sous l'abstraction, plus précisément si $b \to_{\nu_2} b'$, on complète le diagramme de la façon suivante :



□

Nous donnons maintenant une preuve très simple de la noethérianité de ν_1 et de ν_2. Dans la section suivante nous montrons indépendemment la noethérianité de ν, qui entraîne évidemment celle de ν_1 et ν_2. Nous voulons quand même inclure ces deux preuves car elles renseignent sur le poids polynomial correspondant à chacun des sous-calculs.

Proposition 6.5 *La réduction ν_1 est noethérienne.*

Démonstration
On définit une fonction de poids $P : \Lambda_e \to \mathbb{N}$ par

$$\begin{aligned} P(u) &= 1 \quad (x \in V_e) \\ P(ab) &= P(a) + P(b) + 1 \\ P(\lambda u.a) &= P(a) + 1 \\ P(a\langle k\rangle) &= 3 \cdot P(a) \\ P(a[b/u]) &= P(a) + P(b) \end{aligned}$$

1. Pour chaque règle $a \to b$ de ν_1 on vérifie sans difficulté que $P(a) > P(b)$.
2. On montre par récurrence sur la complexité du contexte que pour chaque contexte C et pour chaque règle $a \to b$ de ν_1 on a $P(C(a)) > P(C(b))$.

□

Proposition 6.6 *La réduction ν_2 est noethérienne.*

Démonstration
On définit une fonction de poids $Q : \Lambda_e \to \mathbb{N}$ par

$$\begin{aligned} Q(u) &= 1 \quad (x \in V_e) \\ Q(ab) &= Q(a) + Q(b) + 1 \\ Q(\lambda u.a) &= Q(a) + 1 \\ Q(a\langle k\rangle) &= Q(a) \\ Q(a[b/u]) &= 2 \cdot Q(a) \cdot Q(b) \end{aligned}$$

et le reste de la démonstration est analogue à la démonstration de la proposition précédente.
□

Proposition 6.7 *La réduction ν_1 est localement confluente.*

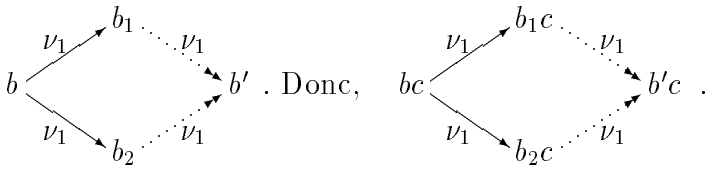
Démonstration
L'algorithme de superposition de Knuth-Bendix pourrait, en principe, nous donner une démonstration immédiate; or, il n'est pas certain qu'on puisse appliquer ce résultat à notre calcul, étant donné la précédence soulignée dans la définition 6.2.

On montre donc la proposition par récurrence sur la complexité de a.

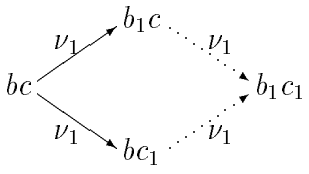
I) Si a est une variable, a est en ν_1-forme normale et il n'y a rien à démontrer.

II) Si $a = bc$, il y a deux possibilités essentiellement différentes :

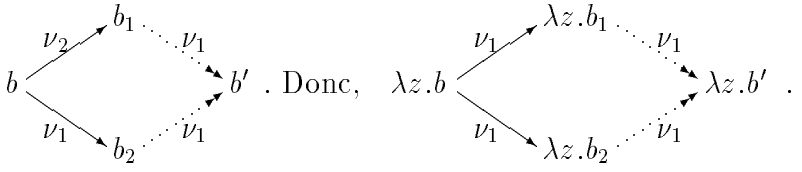
i) Les deux réductions ont lieu simultanément dans b ou elles ont lieu simultanément dans c. Examinons, par exemple, le premier cas. Par H.R. il existe b' tel que



ii) Une réduction a lieu dans b et l'autre dans c. Alors on complète le diagramme :

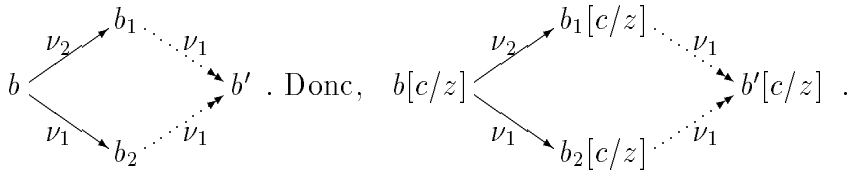


III) Si $a = \lambda z.b$, les deux réductions ont lieu dans b et par H.R. on obtient b' tel que

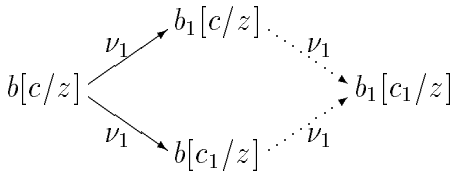


IV) Si $a = b[c/z]$, il y a deux possibilités essentiellement différentes :

i) Les deux réductions ont lieu simultanément dans b ou elles ont lieu simultanément dans c. Examinons, par exemple, le premier cas. Par H.R. il existe b' tel que

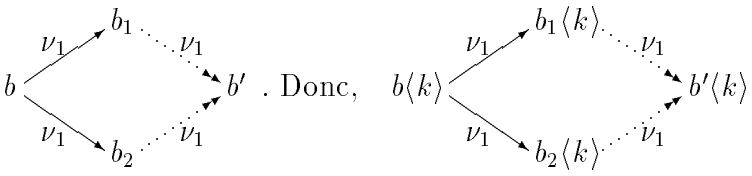


ii) Une réduction a lieu dans b et l'autre dans c. Alors on complète le diagramme :



V) Si $a = b\langle k\rangle$ on considère les deux possibilités suivantes :

i) Aucun des ν_1-radicaux ne se trouve à la racine. Alors les deux réductions ont lieu dans b et par H.R. il existe b' tel que

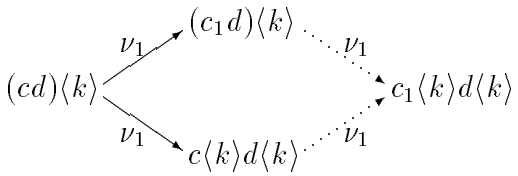


ii) Au moins l'un des ν_1-radicaux est à la racine. Supposons que l'autre réduction n'a pas lieu à la racine, autrement le résultat est évident. On considère trois sous-cas selon la règle utilisée :

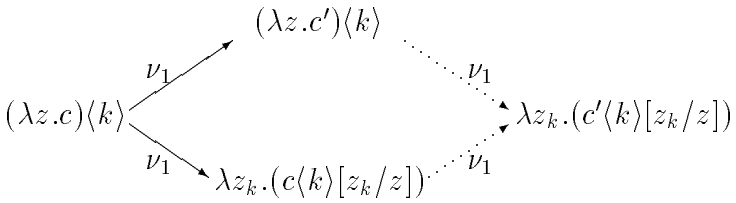
ii.1) Si la règle est *(Vark)*, l'autre réduction a nécessairement lieu à la racine, ce qui contredit notre supposition.

ii.2) Si la règle est *(Appk)*, il y a deux sous-cas analogues selon que la réduction qui n'a pas lieu à la racine a lieu dans c ou dans d. Sup-

posons qu'elle a lieu dans c et complétons le diagramme :



ii.3) Si la règle est *(Absk)*, la réduction qui n'a pas lieu à la racine a lieu dans c. Supposons donc que $c \to_{\nu_1} c'$, et complétons le diagramme :

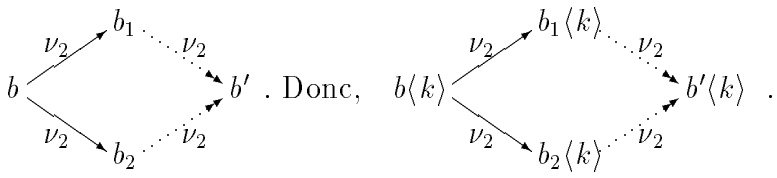


□

Proposition 6.8 *La réduction ν_2 est localement confluente.*

Démonstration
La démonstration suit les lignes de la démonstration de la proposition précédente. On démontre la confluence locale par récurrence sur la complexité du terme a sur lequel s'effectuent les ν_2-réductions.
Les cas $a \in V_e$, $a = bc$ et $a = \lambda z.b$ étant exactement analogues à ceux de la démonstration précédente, on se contente d'étudier les deux autres cas.

I) Si $a = b\langle k\rangle$, les ν_2-réductions ont lieu dans b. Par H.R. on sait qu'il existe b' tel que



. Donc,

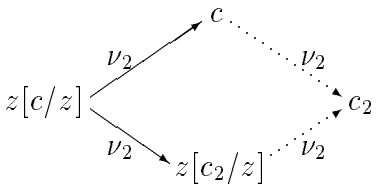
.

II) Si $a = b[c/z]$, il y a deux possibilités :

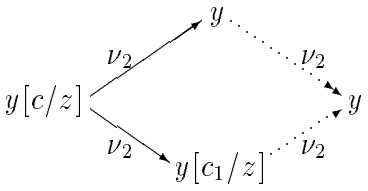
i) Aucun des ν_2-radicaux ne se trouve à la racine. L'étude de ce sous-cas est exactement analogue au cas IV) de la démonstration précédente.

ii) Un radical se trouve à la racine et l'autre à l'intérieur. Alors il y a quatre sous-cas selon la règle qu'on utilise pour réduire le radical à la racine.

ii.1) Si la règle est *(Var=)*, on complète le diagramme :

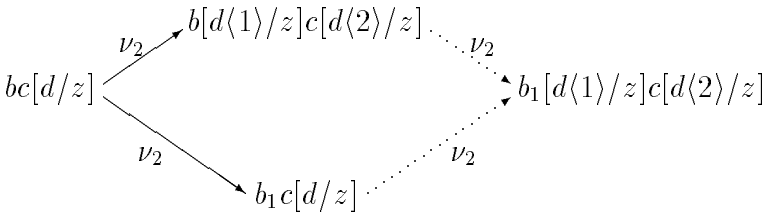


ii.2) Si la règle est *(Var)*, le diagramme se complète comme suit :



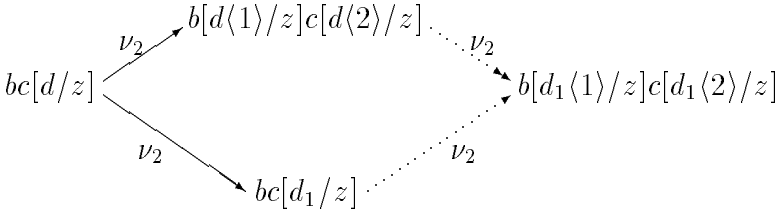
ii.3) Si la règle est *(App)*, il y a trois possibilités selon le sous-terme où la réduction intérieure a lieu :

ii.3.1) Si la réduction intérieure a lieu dans la première composante de l'application, on complète le diagramme :



ii.3.2) Si la réduction intérieure a lieu dans la deuxième composante de l'application, on raisonne comme dans le cas précédent.

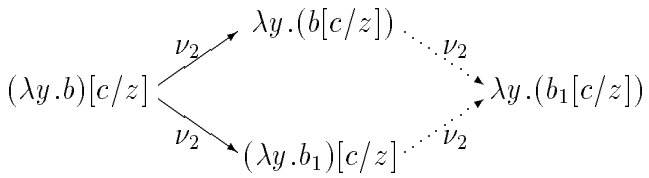
ii.3.3) Si la réduction intérieure a lieu sous la substitution, on complète le diagramme de la façon suivante :



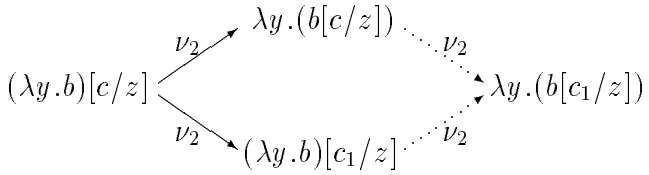
ii.4) Si la règle est *(Abs)*, il y a deux possibilités selon le sous-terme où la réduction intérieure a lieu :

ii.4.1) Si la réduction intérieure a lieu sous l'abstraction, on complète

le diagramme :



ii.4.2) Si la réduction intérieure a lieu sous la substitution, on complète le diagramme :



□

Théorème 6.3 *La ν-réduction est confluente.*

Démonstration
Puisque noethérienité et confluence locale entraînent confluence (cf. lemme 1.3), les propositions 6.5, 6.6, 6.7 et 6.8 nous garantissent la confluence de ν_1 et ν_2.
Or, $\nu = \nu_1 \cup \nu_2$ et ν_1 et ν_2 commutent par la proposition 6.4. On a, donc, les hypothèses du Théorème de Hindley-Rosen (théorème 1.1) qui nous fournit la confluence de ν.
□

6.5 Noethérianité du ν-calcul

La confluence du ν-calcul nous garantit l'unicité des formes normales, tandis que l'existence est assurée par la noethérianité du calcul. Dans cette section nous montrons la noethérianité par récurrence sur la complexité des termes. La remarque 6.7 nous permet de traiter les cas des applications et des abstractions. Le cas des renommages est reglé grâce au lemme 6.5. Ce lemme nécessite un sous-lemme (lemme 6.4) : certaines substitutions sur des termes fortement normalisables sont fortement normalisables. Enfin le lemme 6.6 nous permet de règler le cas des substitutions en général.

Notation 6.3 *Dans cette section SN notera le sous-ensemble de $\Lambda\nu$ des termes fortement normalisables, i.e. ceux à partir desquels il n'y a pas de réductions infinies. Si $a \in SN$,* $\mathrm{prof}(a)$ *dénotera la* profondeur *de a, i.e. la longueur de la réduction la plus longue qui est d'ailleurs bien définie grâce au lemme de König. D'autre part,* $\lg(a)$ *notera la longueur ou complexité de a définie de façon évidente.*

Remarque 6.7 *Pour tous* $a, b \in \Lambda\nu$, $u \in V_e$ *on a :*

1. $ab \in SN$ *ssi* $a \in SN$ *et* $b \in SN$.

2. $\lambda u.a \in SN$ *ssi* $a \in SN$.

Démonstration
Les règles du ν-calcul ne réduisent ni les applications ni les abstractions.
□

Lemme 6.4 *Soient* $a \in SN$, $n \geq 0$, $\langle k_1, \cdots, k_n \rangle \in \{1,2\}^*$, *et* $u, v \in V_e$. *Alors* $a[v\langle k_1\rangle \cdots \langle k_n\rangle / u] \in SN$.

Démonstration
On le montre par récurrence sur $(\mathrm{prof}(a), \mathrm{lg}(a), n)$.
Le cas $(\mathrm{prof}(a), \mathrm{lg}(a), n) = (0,1,0)$ se réduit à démontrer que pour toute variable $w \in V_e$, $w[v/u] \in SN$, ce qui est immédiat. Soit donc $a[v\langle k_1\rangle \cdots \langle k_n\rangle / u]$ tel que $(\mathrm{prof}(a), \mathrm{lg}(a), n) > (0,1,0)$ et démontrons qu'il est fortement normalisable en vérifiant que tous les réduits possibles sont fortement normalisables.

i) La réduction a lieu dans a . On a donc

$$a[v\langle k_1\rangle \cdots \langle k_n\rangle / u] \to_\nu a'[v\langle k_1\rangle \cdots \langle k_n\rangle / u] ,$$

et on peut appliquer l'H.R. à ce dernier terme car $\mathrm{prof}(a') < \mathrm{prof}(a)$.

ii) La réduction a lieu dans $v\langle k_1\rangle \cdots \langle k_n\rangle$. Dans ce cas il n'y a qu'une réduction possible :

$$a[v\langle k_1\rangle \cdots \langle k_n\rangle / u] \to_\nu a[v\langle k'_1\rangle \cdots \langle k'_{n-1}\rangle / u] , \text{ où } k'_i = k_{i+1} ,$$

et on peut appliquer l'H.R. à ce dernier terme car n a diminué de 1, la profondeur et longueur n'ayant pas changé.

iii) La réduction a lieu à la racine. Il y a quatre sous-cas selon la règle utilisée :

iii.1) La règle est *(Var)*, alors le réduit est une variable qui est fortement normalisable.

iii.2) La règle est *(Var=)*, alors le réduit est $v\langle k_1\rangle \cdots \langle k_n\rangle$ qui est aussi fortement normalisable (on le vérifie par récurrence sur n).

iii.3) La règle est *(App)*. On a donc

$$(bc)[v\langle k_1\rangle \cdots \langle k_n\rangle / u] \to_\nu (b[v\langle k_1\rangle \cdots \langle k_n\rangle\langle 1\rangle / u])(c[v\langle k_1\rangle \cdots \langle k_n\rangle\langle 2\rangle / u]) .$$

Puisque $\mathrm{prof}(bc) \geq \mathrm{prof}(b)$, $\mathrm{prof}(bc) \geq \mathrm{prof}(c)$, $\mathrm{lg}(bc) > \mathrm{lg}(b)$ et $\mathrm{lg}(bc) > \mathrm{lg}(c)$, on peut utiliser l'H.R. pour conclure que

$$b[v\langle k_1\rangle \cdots \langle k_n\rangle\langle 1\rangle / u] \in SN \quad \text{et} \quad c[v\langle k_1\rangle \cdots \langle k_n\rangle\langle 2\rangle / u] \in SN .$$

La remarque précédente nous garantit que le réduit est fortement normalisable.

iii.4) La règle est *(Abs)*. On a donc

$$(\lambda w.b)[v\langle k_1\rangle \cdots \langle k_n\rangle/u] \to_\nu \lambda w.(b[v\langle k_1\rangle \cdots \langle k_n\rangle/u])$$

Mais, $\mathrm{prof}(\lambda w.b) = \mathrm{prof}(b)$ et $\lg(\lambda w.b) > \lg(b)$; donc, on applique l'H.R. pour obtenir $(b[v\langle k_1\rangle \cdots \langle k_n\rangle/u]) \in SN$, et la remarque précédente nous permet encore une fois de conclure.

□

Lemme 6.5 *Soient $a \in \Lambda\nu$ et $k \in \{1,2\}$. Si $a \in SN$, alors $a\langle k\rangle \in SN$.*

Démonstration
On le montre par récurrence sur $(\mathrm{prof}(a), \lg(a))$.
Le cas $(\mathrm{prof}(a), \lg(a)) = (0,1)$ se réduit à montrer que pour $u \in V_e$, $u\langle k\rangle \in SN$, ce qui est évident. Soit donc $a\langle k\rangle$ tel que $(\mathrm{prof}(a), \lg(a)) > (0,1)$, et montrons qu'il est fortement normalisable en vérifiant que tous les réduits possibles sont fortement normalisables.

i) La réduction a lieu dans a. On a donc

$$a\langle k\rangle \to_\nu a'\langle k\rangle \text{ , où } a \to_\nu a'$$

et on peut appliquer l'H.R. à ce dernier terme car $\mathrm{prof}(a') < \mathrm{prof}(a)$.

ii) La réduction a lieu à la racine. Il y a trois sous-cas selon la règle utilisée :

ii.1) La règle est *(Vark)*, alors le réduit est une variable qui est fortement normalisable.

ii.2) La règle est *(Appk)*. On a donc

$$(bc)\langle k\rangle \to_\nu b\langle k\rangle c\langle k\rangle$$

Puisque $\mathrm{prof}(bc) \geq \mathrm{prof}(b)$, $\mathrm{prof}(bc) \geq \mathrm{prof}(c)$, $\lg(bc) > \lg(b)$ et $\lg(bc) > \lg(c)$; on peut utiliser l'H.R. pour conclure que $b\langle k\rangle \in SN$ et $c\langle k\rangle \in SN$. La remarque précédente nous garantit que le réduit est SN.

ii.3) La règle est *(Absk)*. On a donc

$$(\lambda u.b)\langle k\rangle \to_\nu \lambda u_k.(b\langle k\rangle[u_k/u])$$

Mais, $\mathrm{prof}(\lambda u.b) = \mathrm{prof}(b)$ et $\lg(\lambda u.b) > \lg(b)$; donc, on applique l'H.R. por obtenir $b\langle k\rangle \in SN$, le lemme précédent (cas $n = 0$) et la remarque 6.7 nous permettent de conclure.

□

Lemme 6.6 *Soient $a, b \in \Lambda\nu$ et $u \in V_e$. Si $a \in SN$ et $b \in SN$, alors $a[b/u] \in SN$.*

Démonstration
Récurrence sur $(\text{prof}(a), \lg(a), \text{prof}(b))$.
Le cas $(\text{prof}(a), \lg(a), \text{prof}(b)) = (0, 1, 0)$ se réduit à montrer que pour toute variable $v \in V_e$ et pour tout b en ν-forme normale, $v[b/u] \in SN$, ce qui est immédiat. Soit donc $a[b/u]$ tel que $(\text{prof}(a), \lg(a), \text{prof}(b)) > (0, 1, 0)$ et montrons qu'il est fortement normalisable en vérifiant que tous les réduits possibles sont SN.

i) La réduction a lieu dans a. On a donc

$$a[b/u] \to_\nu a'[b/u]$$

et on peut appliquer l'H.R. à ce dernier terme car $\text{prof}(a') < \text{prof}(a)$.

ii) La réduction a lieu dans b. On a

$$a[b/u] \to_\nu a[b'/u]$$

et on peut appliquer l'H.R. à ce dernier terme car $\text{prof}(b') < \text{prof}(b)$, la profondeur et longueur de a n'ayant pas changé.

iii) La réduction a lieu à la racine. Il y a quatre sous-cas selon la règle utilisée :

iii.1) La règle est *(Var)*, alors le réduit est une variable qui est fortement normalisable.

iii.2) La règle est *(Var=)*, alors le réduit est b qui est aussi fortement normalisable par hypothèse.

iii.3) La règle est *(App)*. On a donc

$$(cd)[b/u] \to_\nu (c[b\langle 1\rangle/u])(d[b\langle 2\rangle/u])$$

Puisque $\text{prof}(cd) \geq \text{prof}(c)$, $\text{prof}(cd) \geq \text{prof}(d)$, $\lg(cd) > \lg(c)$ et $\lg(cd) > \lg(d)$, et comme $b\langle k\rangle \in SN$ par le lemme précédent on peut utiliser l'H.R. pour conclure que $c[b\langle 1\rangle/u] \in SN$ et $c[b\langle 2\rangle/u] \in SN$. La remarque 6.7 garantit que le réduit est SN.

iii.4) La règle est *(Abs)*. On a donc

$$(\lambda v.c)[b/u] \to_\nu \lambda v.(c[b/u])$$

Mais, $\text{prof}(\lambda v.c) = \text{prof}(c)$ et $\lg(\lambda v.c) > \lg(c)$; donc, on applique l'H.R. pour obtenir $(c[b/u]) \in SN$, et la remarque 6.7 nous permet de conclure.

□

Théorème 6.4 *Le ν-calcul est noethérien.*

Démonstration
On montrera que tout $a \in \Lambda\nu$ est fortement normalisable par récurrence sur la complexité de a.

- Si $a \in V_e$, évident.
- Si $a = bc$, remarque 6.7.
- Si $a = \lambda z.b$, idem.
- Si $a = b\langle k \rangle$, lemme 6.5.
- Si $a = b[c/y]$, lemme 6.6.

□

Corollaire 6.1 *Tout terme $a \in \Lambda\nu$ a une, et une seule, ν-forme normale $\nu(a)$.*

Démonstration
Le théorème précédent garantit l'existence de la forme normale, tandis que le théorème 6.3 assure l'unicité.
□

La remarque suivante, dont la démonstration est immédiate, sera utilisée sans mention explicite dans la suite.

Remarque 6.8 *Soient $a, b \in \Lambda\nu$ et $u \in V_e$. On a les égalités suivantes :*

1. $\nu(ab) = \nu(a)\nu(b)$
2. $\nu(\lambda u.a) = \lambda u.\nu(a)$
3. $\nu(a\langle k \rangle) = \nu(\nu(a)\langle k \rangle)$
4. $\nu(a[b/u]) = \nu(\nu(a)[\nu(b)/u])$

6.6 Correction et confluence du $\lambda\nu$-calcul

Dans cette section nous montrons que le $\lambda\nu$-calcul est correct par rapport au λ-calcul quand on utilise une certaine stratégie de réduction, que nous appelons $B\nu$.

Nous avons déjà montré que toute β-réduction correspond, à α-congruence près, à une application de *(Beta)* suivie de ν-normalisation (cf. démonstration de la proposition 6.3). Nous montrons maintenant la réciproque : une application de *(Beta)* suivie de ν-normalisation correspond, à α-congruence près, à un pas de β-réduction.

Proposition 6.9 *Soit $a \in \Lambda_e$ tel que $a \models INV$ et $a \xrightarrow{Beta} a'$, alors il existe $t \in \Lambda_e$ tel que $a \rightarrow_\beta t$ et $t \equiv_\alpha \nu(a')$.*

Démonstration
On le montrera par récurrence sur la complexité de a.

I) Si $a \in V_e$, alors a ne contient pas de *(Beta)*-radicaux et la proposition est triviale.

II) Si $a = bc$, il y a trois sous-cas à étudier selon l'occurrence du *(Beta)*-radical :

i) Si $b \xrightarrow{Beta} b'$, alors $a' = b'c$, et par H.R. on sait qu'il existe t' tel que $b \rightarrow_\beta t'$ et $t' \equiv_\alpha \nu(b')$. Mais alors $bc \rightarrow_\beta t'c$ et $t'c \equiv_\alpha \nu(b')c = \nu(b'c)$.

ii) Si $c \xrightarrow{Beta} c'$, analogue au cas précédent.

iii) Si $b = \lambda u.d$, alors $a' = d[c/u]$. D'après la proposition 6.2 on obtient $t' \in \Lambda_e$ tel que $d[c/u] \twoheadrightarrow_\nu t'$ et $t' \equiv_\alpha d[\![c/u]\!]$. D'où $\nu(d[c/u]) = t'$ et, évidemment, $(\lambda u.d)c \rightarrow_\beta d[\![c/u]\!]$.

III) Si $a = \lambda u.b$, alors $b \xrightarrow{Beta} b'$ et $a' = \lambda u.b'$. Par H.R. on sait qu'il existe t' tel que $b \rightarrow_\beta t'$ et $t' \equiv_\alpha \nu(b')$. Mais alors $\lambda u.b \rightarrow_\beta \lambda u.t'$ et $\lambda u.t' \equiv_\alpha \lambda u.\nu(b') = \nu(\lambda u.b')$.

□

Nous montrons maintenant que la *(Beta)*-réduction suivie de ν-normalisation préserve l'invariant.

Proposition 6.10 *Soit $a \in \Lambda_e$ tel que $a \models INV$ et $a \xrightarrow{Beta} a'$ alors $\nu(a') \models INV$.*

Démonstration
On le montrera par récurrence sur la complexité de a.

I) Si $a \in V_e$, alors a ne contient pas de *(Beta)*-radicaux et la proposition est triviale.

II) Si $a = bc$, il y a trois sous-cas à étudier selon l'occurrence du *Beta*-radical :

i) Si $b \xrightarrow{Beta} b'$, alors $a' = b'c$, et par H.R., $\nu(b') \models INV$. Mais $\nu(a') = \nu(b')c$. On veut donc démontrer que $\nu(b')c \models INV$.

I1) Le seul cas intéressant est $\lambda z_\iota \in \nu(b')$, $\lambda z_\kappa \in c$ et $\iota \prec\!\!\!\cdot\!\!\!\succ \kappa$. Mais alors le lemme 6.1 nous donne $\iota' \preceq \iota$ telle que $\lambda z_{\iota'} \in b$, ce qui est absurde car $\iota' \prec\!\!\!\cdot\!\!\!\succ \kappa$ et $bc \models INV$. Donc ce cas ne se produit pas.

I2) Il y a deux cas intéressants à considérer :

- Si $z_\iota \in FV(\nu(b'))$, $\lambda z_\kappa \in c$ et $\iota \prec\!\!\!\cdot\!\!\!\succ \kappa$, alors le lemme 6.2 nous assure $z_\iota \in FV(b)$, absurde.
- Si $z_\iota \in FV(c)$, $\lambda z_\kappa \in \nu(b')$ et $\iota \prec\!\!\!\cdot\!\!\!\succ \kappa$, mais alors le lemme 6.1 nous fournit $\kappa' \preceq \kappa$ telle que $\lambda z_{\kappa'} \in b$, ce qui est encore absurde.

Donc, les cas gênants ne se produisent pas.

ii) Si $c \xrightarrow{Beta} c'$, c'est analogue au cas précédent.

iii) Si $b = \lambda u.d$ et la réduction a lieu à la racine, alors $a' = d[c/u]$ .La proposition 6.2 assure l'existence de $t \in \Lambda_e$ tel que $a' \twoheadrightarrow_\nu t$ et $t \models INV$. Mais alors $\nu(a') = t$, et donc $\nu(a') \models INV$.

III) Si $a = \lambda z_\iota.b$, nécessairement $b \xrightarrow{Beta} b'$ et $a' = \lambda z_\iota.b'$. Par H.R. $\nu(b') \models INV$. On veut montrer $\lambda z_\iota.\nu(b') \models INV$, car $\nu(a') = \lambda z_\iota.\nu(b')$.

I1) Le seul cas intéressant est $\lambda z_\kappa \in \nu(b')$ avec $\kappa \prec\!\!\succ \iota$. Le lemme 6.1 nous donne $\kappa' \preceq \kappa$ telle que $\lambda z_{\kappa'} \in b$, absurde. Donc ce cas est impossible.

I2) Le seul cas intéressnat est $z_\kappa \in FV(\nu(b'))$ avec $\kappa \prec\!\!\succ \iota$ et $\kappa \neq \iota$. Le lemme 6.2 assure $z_\kappa \in FV(b)$, absurde. Donc ce cas est aussi impossible.

□

La proposition précédente suggère la stratégie de réduction à suivre : après chaque application de *(Beta)*, ν-réduire à forme normale, de sorte que toutes les *(Beta)*-réductions s'effectuent sur des termes dans Λ_e. Plus précisément :

Définition 6.7 *Une $\lambda\nu$-dérivation est $B\nu$ ssi les trois conditions suivantes sont satisfaites :*

1. *La règle (Beta) n'est appliquée qu'à des termes dans Λ_e.*
2. *Le premier terme de la réduction satisfait l'invariant.*
3. *Le dernier terme de la réduction appartient à Λ_e.*

Quand $a \twoheadrightarrow_{\lambda\nu} b$ et la dérivation est $B\nu$ on utilesera la notation $a \overset{B\nu}{\twoheadrightarrow} b$.

Enfin nous montrons que, en suivant cette stratégie de réduction, le $\lambda\nu$-calcul est correct par rapport au λ-calcul, i.e que toute dérivation $B\nu$ correspond à une β-dérivation du λ-calcul. Nous montrons aussi que cette stratégie préserve l'invariant.

Théorème 6.5 *Soient $a, b \in \Lambda_e$ tels que $a \overset{B\nu}{\twoheadrightarrow} b$ alors $b \models INV$ et $a \twoheadrightarrow_\beta b$.*

Démonstration
Soit n le nombre des fois que la règle *(Beta)* est appliquée pendant la réduction. On montrera le théorème par récurrence sur n.
Si $n = 1$, la réduction est de la forme $a \xrightarrow{Beta} c \twoheadrightarrow_\nu \nu(c) = b$ et le théorème est exactement la proposition 6.9, car, rappelons-le, la α-congruence est implicite dans $\twoheadrightarrow_\beta$.
Supposons maintenant que $a \overset{B\nu}{\twoheadrightarrow} c \xrightarrow{Beta} d \twoheadrightarrow_\nu \nu(d) = b$. Par H.R. $c \models INV$ et $a \twoheadrightarrow_\beta c$. Puisque $c \models INV$, la proposition 6.10 garantit $b \models INV$, et la

proposition 6.9 assure $c \rightarrow_\beta b$. Donc, $a \twoheadrightarrow_\beta b$.
□

Nous avons donc une sorte de double simulation : la simulation (à α-congruence près) du λ-calcul dans le $\lambda\nu$-calcul (cf. théorème 6.2) et la simulation (restreinte aux $B\nu$-dérivations) du $\lambda\nu$-calcul dans le λ-calcul. Nous pouvons donc utiliser la confluence de ce dernier pour obtenir un résultat de confluence pour le $\lambda\nu$ calcul.

Théorème 6.6 *La $B\nu$-dérivation est confluente à α-congruence près.*
Plus précisément, si $a, b, c \in \Lambda_e$ vérifient $a \overset{B\nu}{\twoheadrightarrow} b$ et $a \overset{B\nu}{\twoheadrightarrow} c$, il existe $b', c' \in \Lambda_e$ tels que $b \overset{B\nu}{\twoheadrightarrow} b'$, $c \overset{B\nu}{\twoheadrightarrow} c'$ et $b' \equiv_\alpha c'$.

Démonstration
D'après le théorème précédent, $b, c \models INV$, $a \twoheadrightarrow_\beta b$ et $a \twoheadrightarrow_\beta c$. La confluence du λ-calcul assure l'existence de $d \in \Lambda_e$ tel que $b \twoheadrightarrow_\beta d$ et $c \twoheadrightarrow_\beta d$.

Remarquons maintenant que les $\lambda\nu$-dérivations fournies par la proposition 6.3 et le théorème 6.2 sont, en fait, des $B\nu$-dérivations. Donc, selon ce dernier théorème, il existe $b', c' \in \Lambda_e$ tels que $b \overset{B\nu}{\twoheadrightarrow} b'$, $c \overset{B\nu}{\twoheadrightarrow} c'$ et $b' \equiv_\alpha d \equiv_\alpha c'$.
□

Ce dernier théorème est le seul résultat de confluence que nous avons obtenu pour le $\lambda\nu$-calcul. Cependant, si nous revenons à l'exemple de perte de l'invariant (cf. exemple 6.3) et continuons la dérivation du terme a, nous arrivons à la forme normale $\lambda y_{22}.y_{22}$. Il est facile de constater que, dans cet exemple, toute dérivation avec stratégie $B\nu$ conduit au même résultat. Nous posons donc la question suivante :

Question 6.1 *La $B\nu$-dérivation est-elle confluente même en enlevant la restriction "à α-congruence près"?*

On peut trouver facilement un terme et deux dérivations qui montrent que le $\lambda\nu$-calcul n'est pas confluent :

Exemple 6.5 *Le $\lambda\nu$-calcul n'est pas confluent, même en partant d'un terme qui satisfait INV.*

Voici deux formes normales différentes du terme $(\lambda y.(\lambda x.x)y)(\lambda z.z)$.

$$(\lambda y.(\lambda x.x)y)(\lambda z.z) \twoheadrightarrow_{\lambda\nu} (\lambda y.y)(\lambda z.z) \twoheadrightarrow_{\lambda\nu} \lambda z.z$$

$$(\lambda y.(\lambda x.x)y)(\lambda z.z) \twoheadrightarrow_{\lambda\nu} (\lambda x.x)[(\lambda z.z)\langle 1\rangle/y]\, y[(\lambda z.z)\langle 2\rangle/y] \twoheadrightarrow_{\lambda\nu}$$
$$\twoheadrightarrow_{\lambda\nu} (\lambda x.x)(\lambda z_2.z_2) \twoheadrightarrow_{\lambda\nu} \lambda z_2.z_2$$

□

Cependant, nous constatons que les formes normales sont α-congruentes. Nous posons donc une deuxième question :

Question 6.2 *Le $\lambda\nu$-calcul est-il confluent, modulo α-congruence, quand on part d'un terme satisfaisant l'invariant?*

Chapitre 7

Le $\lambda\tau$-calcul

Dans ce chapitre nous présentons le $\lambda\tau$-calcul et montrons la normalisation faible de ce calcul. C'est le seul résultat obtenu sur ce calcul jusqu'à présent. Nous considérons donc que le plus gros du travail reste encore à faire.

Le $\lambda\tau$-calcul est un système de réécriture obtenu par orientation de la théorie équationnelle présentée par T. Ehrhard (voir [Ehr88], annexe). Cette théorie provient d'une axiomatisation d'un certain type de modèles des constructions. Le calcul de constructions considéré est une version modifiée par T. Streicher [Str88] du Calcul des Constructions de T. Coquand et G. Huet. Le calcul des constructions est une théorie de types qui contient la logique intuitioniste d'ordre supérieur (système **F** de Girard) et des types dépendants du premier ordre comme dans le système de Martin-Löf (voir [ML85]).

Le $\lambda\tau$-calcul est proche du $\lambda\sigma_{\Uparrow}$-calcul. La différence essentielle entre $\lambda\tau$ et les $\lambda\sigma$-calculs que nous avons présentés dans cette thèse se trouve au niveau de l'opérateur *cons*. En effet, puisque la règle *(Beta)*, n'utilise cet opérateur que dans le cas particulier où la substituion est id, l'idée de remplacer *cons* par un opérateur unaire surgit naturellement.

7.1 Présentation du calcul

Dans cette section nous introduisons le calcul, caractérisons les τ-formes normales et constatons la confluence locale.

Définition 7.1 *La syntaxe du $\lambda\tau$-calcul est donnée par:*

$$\begin{array}{lll} \textbf{Termes} & a ::= & \mathtt{1} \mid ab \mid \lambda a \mid a[s] \\ \textbf{Substitutions} & s ::= & id \mid \uparrow \mid a/ \mid s \circ t \mid \Uparrow(s) \end{array}$$

*Les règles de réduction du $\lambda\tau$-*calcul *sont données dans la figure 7.1. Le τ-calcul est le sous-système du $\lambda\tau$-calcul obtenu en enlevant la règle (Beta).*

(Beta)	$(\lambda a)b$	$\longrightarrow$	$a[b/]$
(App)	$(ab)[s]$	$\longrightarrow$	$a[s]b[s]$
(Lambda)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a[\Uparrow(s)])$
(Clos)	$(a[s])[t]$	$\longrightarrow$	$a[s \circ t]$
(Fvar)	$\mathtt{1}[a/]$	$\longrightarrow$	a
(Fvlift1)	$\mathtt{1}[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{1}$
(Fvlift2)	$\mathtt{1}[\Uparrow(s) \circ t]$	$\longrightarrow$	$\mathtt{1}[t]$
(Ass)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$
(MapEnv)	$(a/) \circ s$	$\longrightarrow$	$\Uparrow(s) \circ (a[s]/)$
(Shift)	$\uparrow \circ (a/)$	$\longrightarrow$	id
(ShiftLift1)	$\uparrow \circ \Uparrow(s)$	$\longrightarrow$	$s \circ \uparrow$
(ShiftLift2)	$\uparrow \circ (\Uparrow(s) \circ t)$	$\longrightarrow$	$s \circ (\uparrow \circ t)$
(Lift1)	$\Uparrow(s) \circ \Uparrow(t)$	$\longrightarrow$	$\Uparrow(s \circ t)$
(Lift2)	$\Uparrow(s) \circ (\Uparrow(t) \circ u)$	$\longrightarrow$	$\Uparrow(s \circ t) \circ u$
(IdL)	$id \circ s$	$\longrightarrow$	s
(IdR)	$s \circ id$	$\longrightarrow$	s
(LiftId)	$\Uparrow(id)$	$\longrightarrow$	id
(IdEnv)	$a[id]$	$\longrightarrow$	a

FIG. 7.1 - Le système de réécrirure $\lambda\tau$

Notation 7.1 *On note $\Lambda\tau^t$ l'ensemble des termes et $\Lambda\tau^s$ l'ensemble des substitutions de ce calcul, tandis que $\Lambda\tau = \Lambda\tau^t \cup \Lambda\tau^s$.*

Théorème 7.1 *Les τ-formes normales sont données par :*

$$\begin{array}{lll} \textbf{Termes} & a ::= & \mathtt{1} \mid ab \mid \lambda a \mid \mathtt{1}[\uparrow^n] \\ \textbf{Substitutions} & s ::= & id \mid \uparrow^n \mid a/ \mid \underbrace{\Uparrow(s) \mid \Uparrow(s) \circ \uparrow^n \mid \Uparrow(s) \circ a/}_{s \neq id} \end{array} .$$

Démonstration

D'abord on remarque que les termes et substitutions ainsi définis sont en fait des formes normales. On montrera qu'il n'y en a pas d'autres.

Supposons que $a[s]$ est en forme normale, alors a est nécessairement le terme $\mathtt{1}$, autrement on aurait un *(App)*, un *(Lambda)* ou un *(Clos)*-radical. Analysons maintenant les possibilités pour s. On voit que s ne peut être qu'une composition ou $\uparrow$, autrement on aurait un *(IdEnv)*, un *(Fvar)* ou un *(Fvlift1)*-radical. Si $s = t \circ u$,

alors $t = \uparrow$ nécessairement, autrement on aurait un *(IdL)*, un *(MapEnv)*, un *(Ass)* ou un *(Fvlift2)*-radical. Démontrons l'affirmation suivante :

Affirmation Si $\uparrow \circ u$ est en τ-forme normale, $u = \uparrow^n$.

On le montre par récurrence sur la complexité de u. Si u a complexité 1, alors $u = \uparrow$ ($u = id$ est impossible par *(IdR)*). Supposons donc $\uparrow \circ u$ en f.n. et la complexité de u non triviale, u ne peut être qu'une composition, autrement on aurait un *(Shift)* ou un *(ShiftLift1)*-radical. Alors $u = u_1 \circ u_2$, mais nécessairement $u_1 = \uparrow$, autrement on aurait un *(IdL)*, un *(MapEnv)*, un *(Ass)* ou un *(ShiftLift2)*-radical. On conclut par H.R. que $u_2 = \uparrow^m$, et donc $u = \uparrow^{m+1}$. L'affirmation est donc démontrée. □

Alors $s = \uparrow^n$ et $a[s] = \mathtt{1}[\uparrow^n]$. Remarquons que si ab est en f.n., alors a et b sont en f.n., et que si λa est en f.n., alors a est en f.n.. On a donc démontré que les termes en forme normale sont exactement ceux décrits dans l'enoncé du théorème.

Etudions maintenant les substitutions en forme normale. Remarquons d'abord que si $a/$ est en f.n., alors a est en f.n., et que si $\Uparrow(s)$ est en f.n., alors s est en f.n..

Il sufit d'étudier les compositions car $a/$ et $\Uparrow(s)$, avec $s \neq id$ peuvent avoir des radicaux seulement à l'intérieur. Soit donc $s \circ t$ en f.n. et analysons les possibilités pour s et t. On constate que s ne peut être que $\uparrow$ ou de la forme $\Uparrow(u)$ avec $u \neq id$, autrement on aurait un *(IdL)*, un *(MapEnv)* ou un *(Ass)*-radical. Si $s = \uparrow$, l'affirmation précédente nous assure que $t = \uparrow^n$.

Soit donc $\Uparrow(u) \circ t$, où $u \neq id$ et analysons les possibilités pour t.

- $t \neq id$, autrement on aurait un *(IdR)*-radical.
- $t \neq \Uparrow(t')$, autrement on aurait un *(Lift1)*-radical.
- $t = a/$ est une possibilité valable si a est en f.n..
- $t = \uparrow$ est aussi une possibilité valable.
- $t = t_1 \circ t_2$, on continue l'analyse. D'après ce qu'on vient de voir pour le cas de la composition t_1 ne peut être que $\uparrow$ ou $\Uparrow(t'_1)$, mais ce dernier cas est exclu par *(lift2)*. Donc, $t_1 = \uparrow$, et l'affirmation nous permet de conclure $t_2 = \uparrow^n$.

□

Notation 7.2 *On note Θ l'ensemble des substitutions en τ-forme normale. En identifiant comme d'habitude $\mathtt{1}[\uparrow^n]$ avec l'entier de de Bruijn $\mathtt{n}$, l'ensemble des termes en forme normale s'identifie avec l'ensemble des termes du λ-calcul en notation de de Bruijn. Donc, on conserve la notation Λ pour l'ensemble des termes en forme normale du τ-calcul.*

Théorème 7.2 *Le $\lambda\tau$-calcul est localement confluent.*

Démonstration
D'abord on remarque que les règles préservent les sortes. Après on montre que le système non typé est localement confluent en utilisant le logiciel KB.
□

7.2 Noethérianité faible du τ-calul

Nous montrons dans cette section, par récurrence structurelle, que tous les termes et substitutions du τ-calcul sont faiblement normalisables, i.e. qu'il existe une τ-dérivation qui aboutit à une forme normale (théorème 7.3). Nous décrivons aussi la stratégie à suivre: leftmost-innermost, i.e. celle où à chaque étape de la dérivation on réduit le radical le plus à gauche de l'ensemble des radicaux les plus internes. Deux cas de cette récurrence, les clôtures et compositions, demandent un travail préalable.

Nous procédons ainsi: pour montrer que $a[s]$ et $t \circ s$ sont faiblement normalisables (w.n.), nous normalisons d'abord $a \twoheadrightarrow a'$, $s \twoheadrightarrow s'$ et $t \twoheadrightarrow t'$, ce qui est possible par H.R.. Il suffit donc de montrer que $a'[s']$ et $t' \circ s'$ sont w.n. avec les hypothèses aditionnelles $a' \in \Lambda$ et $t', s' \in \Theta$ (lemmes 7.5 et 7.6).

Nous continuons l'analyse seulement pour $a'[s']$. Les mêmes lemmes intermédiaires que nous devrons montrer pour arriver à la normalisation faible de $a'[s']$, serviront aussi pour établir que $t' \circ s'$ est w.n..

On essaie donc de montrer que $a'[s']$ est w.n. par récurrence sur a'. Le cas $a' = 1[\uparrow^n]$ est le seul qui n'est pas immédiat. Pour l'établir nous montrons que $\uparrow^n \circ s'$ est w.n. par récurrence sur n (lemme 7.4). Le cas $n = 1$ est le lemme 7.3. Quand on etudie $s' = \Uparrow (u) \circ \uparrow^p$, on est ramené à montrer que $u \circ \uparrow^{p+1}$ est w.n. (lemme 7.2). Enfin, le cas $u = b/$ nous conduit à montrer que $b[\uparrow^{p+1}]$ est w.n.. Or, quand on essaie une récurrence sur b, le cas $b = \lambda b'$ oblige à montrer un résultat plus fort: $b[\Uparrow^m (\uparrow^{p+1})]$ est w.n. (lemme 7.1), où $\Uparrow^m$ dénote m applications consécutives de l'opérateur $\Uparrow$. Plus précidément:

Définition 7.2 *Soient $s \in \Lambda\tau^s$ et $m \geq 0$. On définit $\Uparrow^m (s)$ par récurrence sur m de la façon suivante:*

$$\Uparrow^0 (s) = s$$
$$\Uparrow^{m+1} = \Uparrow (\Uparrow^m (s))$$

Nous aurons besoin d'une notation pour abréger $\uparrow \circ (\uparrow \circ (\ldots \circ (\uparrow \circ s)))$:

Définition 7.3 *Soient $s, t \in \Lambda\tau^s$ et $p \geq 0$. On définit $s \overset{p}{\circ} t$ par récurrence sur p ainsi:*

$$s \overset{0}{\circ} t = t$$
$$s \overset{p+1}{\circ} t = s \circ (s \overset{p}{\circ} t)$$

Notation 7.3 *Dans le reste de ce chapitre $f \twoheadrightarrow g$ dénotera la τ-dérivation leftmost-innermost et $f \to g$ un pas de τ-réduction en suivant cette stratégie.*

Nous montrons d'abord une remarque générale qui sera utile pour les lemmes suivants.

Remarque 7.1 *Soient* $s \in \Theta$ *et* $p, m, n \geq 0$ *. Alors :*

i) $\uparrow \overset{p}{\circ} \uparrow^q = \uparrow^{p+q}$.

ii) $\uparrow \overset{p+1}{\circ} s = \uparrow \overset{p}{\circ} (\uparrow \circ s)$.

iii) $\uparrow^{p+1} \circ s \twoheadrightarrow \uparrow \overset{p+1}{\circ} s$.

iv) Si $m \leq p$ *, alors* $\uparrow^p \circ \Uparrow^m (\uparrow^n) \twoheadrightarrow \uparrow^{n+p}$.

v) Si $m > p$ *, alors* $\uparrow^p \circ \Uparrow^m (\uparrow^n) \twoheadrightarrow \Uparrow^{m-p} (\uparrow^n) \circ \uparrow^p$.

Démonstration
Les deux premiers énoncés se démontrent par récurrence sur p .

Quant au troisième, on constate que la dérivation $\uparrow^{p+1} \circ s \twoheadrightarrow \uparrow \overset{p+1}{\circ} s$ est en fait la seule dérivation possible : il n'y a qu'un radical (de type *(Ass)*) à chaque étape. C'est donc après avoir complété l'association à droite que le $\uparrow$ le plus à droite peut interagir avec s .

Montrons le quatrième et cinquième simultanément. D'après l'item précédent $\uparrow^p \circ \Uparrow^m (\uparrow^n) \twoheadrightarrow \uparrow \overset{p}{\circ} \Uparrow^m (\uparrow^n)$. Dans le réduit il n'y a qu'un radical : $\uparrow \circ \Uparrow^m (\uparrow^n)$. La règle *(ShiftLift1)* permet alors que le $\uparrow$ à gauche du $\Uparrow$, traverse le $\Uparrow^m$. Après application de cette règle on a : $\uparrow \overset{p-1}{\circ} (\Uparrow^{m-1} (\uparrow^n) \circ \uparrow)$. A partir de maintenant on ne trouve à chaque étape qu'un seul radical (de type *(ShiftLift2)*). Cette règle permet alors que le $\uparrow$ à gauche du lift continue à traverser le $\Uparrow^{m-1}$. Chaque passage de $\uparrow$ décrémente l'exposant du $\Uparrow$ de 1 . Il y a donc deux possibilités :

1. Tous les $\Uparrow$ sont consommés ($m \leq p$), c'est le cas *iv)* et le réduit est $\uparrow^{n+p}$.
2. Les $\uparrow$ sont consommés avant les $\Uparrow$ ($m > p$), c'est le cas *v)* et le réduit est $\Uparrow^{m-p} (\uparrow^n) \circ \uparrow^p$.

□

Nous pouvons déjà commencer à montrer les lemmes annoncés au début de cette section.

Lemme 7.1 *Soient* $a \in \Lambda$, $m \geq 0$ *et* $n \geq 1$ *. Alors il existe* $a' \in \Lambda$ *tel que*

$$a[\Uparrow^m (\uparrow^n)] \twoheadrightarrow_\tau a' .$$

Démonstration
On le montre par récurrence sur a :

– Si $a = \mathtt{1}$, on considère deux possibilités :

⋆ Si $m = 0$, alors $\mathtt{1}[\Uparrow^m (\uparrow^n)] = \mathtt{1}[\uparrow^n]$.

$\star$ Si $m > 0$, alors $\mathtt{1}[\Uparrow^m(\uparrow^n)] \to \mathtt{1}$, par *(Fvlift1)*.

– Si $a = bc$, alors $(bc)[\Uparrow^m(\uparrow^n)] \to b[\Uparrow^m(\uparrow^n)]c[\Uparrow^m(\uparrow^n)]$, et par H.R. il existe $b', c' \in \Lambda$ tels que $b[\Uparrow^m(\uparrow^n)]\, c[\Uparrow^m(\uparrow^n)] \twoheadrightarrow b'c'$

– Si $a = \lambda b$, alors $(\lambda b)[\Uparrow^m(\uparrow^n)] \to \lambda(b[\Uparrow^{m+1}(\uparrow^n)]$, et encore par H.R. on obtient b' tel que $b[\Uparrow^{m+1}(\uparrow^n)] \twoheadrightarrow b'$, d'où $(\lambda b)[\Uparrow^m(\uparrow^n)] \twoheadrightarrow \lambda b'$.

– Si $a = \mathtt{1}[\uparrow^p]$, alors on considère :

$\star$ Si $m \leq p$, alors $\mathtt{1}[\uparrow^p][\Uparrow^m(\uparrow^n)] \to \mathtt{1}[\uparrow^p \circ \Uparrow^m(\uparrow^n)] \twoheadrightarrow \mathtt{1}[\uparrow^{n+p}]$ (par *(Clos)* et remarque 7.1.iv)).

$\star$ Si $m > p$, alors

$$\mathtt{1}[\uparrow^p][\Uparrow^m(\uparrow^n)] \to \mathtt{1}[\uparrow^p \circ \Uparrow^m(\uparrow^n)] \twoheadrightarrow \mathtt{1}[\Uparrow^{m-p}(\uparrow^n) \circ \uparrow^p] \to \mathtt{1}[\uparrow^p]$$

(par *(Clos)*, remarque 7.1.v) et *(Fvlift2)*).

□

Lemme 7.2 *Si $s \in \Theta$ et $p \geq 1$, alors il existe $t \in \Theta$ telle que $t \neq id$ et $s \circ \uparrow^p \twoheadrightarrow t$.*

Démonstration
On le montrera par récurrence sur $s \in \Theta$.

– Si $s = id$, alors $id \circ \uparrow^p \to \uparrow^p$ (par *(IdL)*).

– Si $s = \uparrow^n$, alors $\uparrow^n \circ \uparrow^p \twoheadrightarrow \uparrow^{n+p}$ (par *(Ass)*).

– Si $s = a/$, alors $a/ \circ \uparrow^p \to \Uparrow(\uparrow^p) \circ (a[\uparrow^p]/) \twoheadrightarrow \Uparrow(\uparrow^p) \circ a'/$ (par *(MapEnv)* et lemme précédent avec $m = 0$).

– Si $s = \Uparrow(u)$, alors $\Uparrow(u) \circ \uparrow^p$ est en f.n..

– Si $s = \Uparrow(u) \circ \uparrow^n$, alors $(\Uparrow(u) \circ \uparrow^n) \circ \uparrow^p \twoheadrightarrow \Uparrow(u) \circ \uparrow^{p+n}$ (par *(Ass)*).

– Si $s = \Uparrow(u) \circ a/$, alors

$$(\Uparrow(u) \circ a/) \circ \uparrow^p \to \Uparrow(u) \circ (a/ \circ \uparrow^p) \to \Uparrow(u) \circ (\Uparrow(\uparrow^p) \circ (a[\uparrow^p]/)) \twoheadrightarrow$$
$$\twoheadrightarrow \Uparrow(u) \circ (\Uparrow(\uparrow^p) \circ a'/) \to \Uparrow(u \circ \uparrow^p) \circ a'/ \twoheadrightarrow \Uparrow(t) \circ a'/$$

(par *(Ass)*, *(MapEnv)*, lemme précédent avec $m = 0$, *(Lift2)* et H.R.).
Remarquons que le réduit obtenu est bien une f.n., car $t \neq id$, et que la stratégie suivie est innermost (on réduit d'abord $a[\uparrow^p]$, avant d'appliquer *(Lift2)*).

□

Lemme 7.3 *Si $s \in \Theta$, alors il existe $t \in \Theta$ telle que $\uparrow \circ s \twoheadrightarrow t$.*

Démonstration
On le montrera par récurrence sur $s \in \Theta$.

- Si $s = id$, alors $\uparrow \circ id \to \uparrow$ (par *(IdR)*).
- Si $s = \uparrow^n$, alors $\uparrow \circ \uparrow^n = \uparrow^{n+1}$, qui est en f.n..
- Si $s = a/$, alors $\uparrow \circ a/ \to id$ (par *(Shift)*).
- Si $s = \Uparrow(u)$, alors $\uparrow \circ \Uparrow(u) \to u \circ \uparrow$ (par *(ShiftLift1)*), et on utilise le lemme précédent.
- Si $s = \Uparrow(u) \circ \uparrow^n$, alors $\uparrow \circ (\Uparrow(u) \circ \uparrow^n) \to u \circ \uparrow^{n+1}$ (par *(ShiftLift2)*), et on utilise le lemme précédent.
- Si $s = \Uparrow(u) \circ a/$, alors $\uparrow \circ (\Uparrow(u) \circ a/) \to u \circ (\uparrow \circ a/) \to u \circ id \to u$ (par *(ShiftLift2)*, *(Shift)* et *(IdR)*).

□

Lemme 7.4 *Si $s \in \Theta$ et $n \geq 0$, alors il existe $t \in \Theta$ telle que $\uparrow \overset{n}{\circ} s \twoheadrightarrow t$.*

Démonstration
On le montre par récurrence sur n. Le cas $n = 0$ est immédiat.
Considérons donc $\uparrow \overset{n+1}{\circ} s$:

$$\uparrow \overset{n+1}{\circ} s = \uparrow \circ (\uparrow \overset{n}{\circ} s) \overset{HR}{\twoheadrightarrow} \uparrow \circ t'$$

où $t' \in \Theta$, et par le lemme précédent, il existe $t \in \Theta$ telle que $\uparrow \circ t' \twoheadrightarrow t$.
Remarquons que la stratégie est toujours innermost : on réduit la composition à la racine après avoir réduit $\uparrow \overset{n}{\circ} s$.
□

Lemme 7.5 *Soient $a \in \Lambda$ et $s \in \Theta$, alors il existe $a' \in \Lambda$ tel que $a[s] \twoheadrightarrow a'$.*

Démonstration
On le montre par récurrence sur a. Remarquons que si $s = id$, il suffit de prendre $a' = a$, grâce à la règle *(IdEnv)*. Supposons donc $s \neq id$.

- Si $a = \mathtt{1}$, on le montre en étudiant $s \in \Theta$.
  - ⋆ Si $s = \uparrow^n$, alors $\mathtt{1}[\uparrow^n] \in \Lambda$.
  - ⋆ Si $s = a/$, alors $\mathtt{1}[a/] \to a$ (par *(Fvar)*), et $a \in \Lambda$ car $s \in \Theta$.
  - ⋆ Si $s = \Uparrow(u)$, alors $\mathtt{1}[\Uparrow(u)] \to \mathtt{1}$ (par *(Fvlift1)*).
  - ⋆ Si $s = \Uparrow(u) \circ \uparrow^n$, alors $\mathtt{1}[\Uparrow(u) \circ \uparrow^n] \to \mathtt{1}[\uparrow^n]$ (par *(Fvlift2)*).

$\star$ Si $s = \Uparrow(u) \circ a/$, alors $\mathtt{1}[\Uparrow(u) \circ a/] \to \mathtt{1}[a/] \to a$ (par *(Fvlift2)* et *(Fvar)*), et $a \in \Lambda$ car $s \in \Theta$.

– Si $a = bc$, alors $(bc)[s] \to b[s]\,c[s] \twoheadrightarrow b'c'$ (par *(App)* et H.R.).

– Si $a = \lambda b$, alors $(\lambda b)[s] \to \lambda(b[\Uparrow(s)]) \twoheadrightarrow \lambda b'$ (par *(Lambda)* et H.R. car $s \in \Theta$ et $s \neq id$ entraîne $\Uparrow(s) \in \Theta$).

– Si $a = \mathtt{1}[\uparrow^n]$, alors $\mathtt{1}[\uparrow^n][s] \to \mathtt{1}[\uparrow^n \circ s] \twoheadrightarrow \mathtt{1}[\uparrow \overset{n}{\circ} s] \twoheadrightarrow \mathtt{1}[t] \twoheadrightarrow b \in \Lambda$ (par *(Clos)*, remarque 7.1.iii), lemme précédent et le cas $a = \mathtt{1}$ du lemme présent).

□

Lemme 7.6 *Soient $s, t \in \Theta$, alors il existe $u \in \Theta$ telle que $s \circ t \twoheadrightarrow u$.*

Démonstration

On le montrera par récurrence sur $s \in \Theta$.Remarquons que si $t = id$, il suffit de prendre $u = s$, grâce à la règle *(IdR)*. Supposons donc $t \neq id$.

– Si $s = id$, alors $id \circ t \to t$ (par *(IdL)*).

– Si $s = \uparrow^n$, alors $\uparrow^n \circ t \twoheadrightarrow \uparrow \overset{n}{\circ} t \twoheadrightarrow u \in \Theta$ (par la remarque 7.1.iii) et le lemme 7.4).

– Si $s = a/$, alors $a/ \circ t \to \Uparrow(t) \circ (a[t]/) \twoheadrightarrow \Uparrow(t) \circ a'/$ (par *(MapEnv)* et lemme précédent, car $s \in \Theta$ entraîne $a \in \Lambda$).

– Si $s = \Uparrow(s')$, alors on le montre par analyse par cas, selon la structure de $t \in \Theta$.

$\star$ Si $t = \uparrow^n$, alors $\Uparrow(s') \circ \uparrow^n \in \Theta$.

$\star$ Si $t = a/$, alors $\Uparrow(s') \circ a/ \in \Theta$.

$\star$ Si $t = \Uparrow(t')$, alors $\Uparrow(s') \circ \Uparrow(t') \to \Uparrow(s' \circ t') \twoheadrightarrow \Uparrow(u')$ (par *(Lift1)* et H.R.). Si $u' \neq id$, alors $\Uparrow(u') \in \Theta$ et on prend $u = \Uparrow(u')$. Si $u' = id$, on prend $u = id$.

$\star$ Si $t = \Uparrow(t') \circ \uparrow^n$, alors on a la réduction suivante :

$$\Uparrow(s') \circ (\Uparrow(t') \circ \uparrow^n) \to \Uparrow(s' \circ t') \circ \uparrow^n \twoheadrightarrow \Uparrow(u') \circ \uparrow^n$$

(par *(Lift2)*, H.R. et lemme 7.2). Si $u' \neq id$, on prend $u = \Uparrow(u') \circ \uparrow^n \in \Theta$. Si $u' = id$, on prend $u = \uparrow^n$.

$\star$ Si $t = \Uparrow(t') \circ a/$, alors on a la réduction :

$$\Uparrow(s') \circ (\Uparrow(t') \circ a/) \to \Uparrow(s' \circ t') \circ a/ \twoheadrightarrow \Uparrow(u') \circ a/$$

(par *(Lift2)* et H.R.). Si $u' \neq id$, on prend $u = \Uparrow(u') \circ a/ \in \Theta$. Si $u' = id$, on prend $u = a/$.

– Si $s = \Uparrow(s') \circ \uparrow^n$, alors on a la réduction suivante :

$$(\Uparrow(s') \circ \uparrow^n) \circ t \to \Uparrow(s') \circ (\uparrow^n \circ t) \twoheadrightarrow \Uparrow(s') \circ (\overset{n}{\uparrow} \circ t) \twoheadrightarrow \Uparrow(s') \circ u' \twoheadrightarrow u$$

(par *(Ass)*, remarque 7.1.iii), lemme 7.4 et H.R.).

– Si $s = \Uparrow(s') \circ a/$, alors on a la réduction :

$$(\Uparrow(s') \circ a/) \circ t \to \Uparrow(s') \circ (a/ \circ t) \to \Uparrow(s') \circ (\Uparrow(t) \circ (a[t])/) \twoheadrightarrow$$
$$\twoheadrightarrow \Uparrow(s') \circ (\Uparrow(t) \circ a'/) \twoheadrightarrow u$$

(par *(Ass)*, *(MapEnv)*, lemme 7.5 et H.R., car $\Uparrow(t) \circ a'/ \in \Theta$).

□

Théorème 7.3 *Le τ-calcul est faiblement noethérien (WN).*

Démonstration
On le montre par récurrence simultanée sur la structure des termes et des substitutions. Le résultat à démontrer est le suivant:
Pour tout $a \in \Lambda\tau^t$ et $s \in \Lambda\tau^s$ il existe $a' \in \Lambda$ et $s' \in \Theta$ tels que $a \twoheadrightarrow_\tau a'$ et $s \twoheadrightarrow_\tau s'$.
Les cas $a = \mathtt{1}$, $s = id$ et $s = \uparrow$ sont immédiats.

– Si $a = bc$, par H.R. il existe $b', c' \in \Lambda$ tels que $b \twoheadrightarrow b'$ et $c \twoheadrightarrow c'$. Alors $b'c' \in \Lambda$ et $a \twoheadrightarrow b'c'$.

– Si $a = \lambda b$, par H.R. il existe $b' \in \Lambda$ tel que $b \twoheadrightarrow b'$. Alors $\lambda b' \in \Lambda$ et $a \twoheadrightarrow \lambda b'$.

– Si $a = b[s]$, par H.R. il existe $b' \in \Lambda$ et $s' \in \Theta$ tels que $b \twoheadrightarrow b'$ et $s \twoheadrightarrow s'$. Alors $a \twoheadrightarrow b'[s']$ et par le lemme 7.5 il existe $b'' \in \Lambda$ tel que $b'[s'] \twoheadrightarrow b''$, donc $a \twoheadrightarrow b''$.

– Si $s = b/$, par H.R. il existe $b' \in \Lambda$ tel que $b \twoheadrightarrow b'$. Alors $b'/ \in \Lambda$ et $s \twoheadrightarrow b'/$.

– Si $s = t \circ u$, par H.R. il existe $t', u' \in \Theta$ telles que $t \twoheadrightarrow t'$ et $u \twoheadrightarrow u'$. Alors $s \twoheadrightarrow t' \circ u'$ et par le lemme 7.6 il existe $s' \in \Theta$ telle que $t' \circ u' \twoheadrightarrow s'$, donc $s \twoheadrightarrow s'$.

– Si $s = \Uparrow(t)$, par H.R. il existe $t' \in \Theta$ telle que $t \twoheadrightarrow t'$.
Si $t' = id$, on prend $s' = id$, d'après la règle *(LiftId)*. Si $t' \neq id$, alors $\Uparrow(t') \in \Theta$ et $s \twoheadrightarrow \Uparrow(t')$.

□

Nous concluons ce chapitre en formulant deux conjectures.

Conjecture 7.1 *Le τ-calcul est noethérien, et donc confluent.*

Conjecture 7.2 *Le $\lambda\tau$-calcul est confluent.*

Nous considérons que la confluence du $\lambda\tau$-calcul pourrait être montrée par la méthode d'interprétation, après avoir établi la noethérianité de τ.

Chapitre 8

Conclusion et problèmes ouverts

Les $\lambda\sigma$-calculs sont un essai fructueux pour démasquer "l'éminence grise" du λ-calcul : la substitution. Les travaux de Abadi, Cardelli, Curien, Hardin et Lévy le confirment. Notre apport personnel à l'étude de ces calculs peut être synthétisé en :

- Une technique nouvelle pour montrer la noethérianité de σ.
- La confluence du $\lambda\sigma'$-calcul et du $\lambda\sigma\eta$-calcul sur les termes semi-clos.
- Une interprétation du $\lambda\sigma$-calcul typé du premier ordre dans le λ-calcul correspondant pour laquelle on montre la correction et la complétude.

Il reste encore à établir la noethérianite des $\lambda\sigma$-calculs typés. Nous consacrons la section 8.2 à la présentation de ce problème. Signalons ici qu'une preuve de terminaison des calculs typés devrait fournir une preuve de terminaison des σ-calculs associés, au moins sur les termes typés. C'est la raison pour laquelle nous considérons qu'aucune technique permettant d'établir la noethérianité des σ-calculs n'est à écarter.

Les résultats de confluence obtenus pour le $\lambda\sigma'$-calcul et pour sa version extensionnelle mettent en valeur l'ensemble des termes semi-clos. Ces termes sont intéressants car, premièrement, ils contiennent les termes du λ-calcul, et deuxièmement, ils permettent de formaliser une notion de modularité. On s'autorise ainsi à travailler avec des "boîtes noires" (les variables de sorte `terme`) grâce auxquelles les termes peuvent être considérés comme des contextes.

Ces résultats de confluence sont montrés par la méthode d'interprétation. On constate encore une fois que cette méthode s'avère l'outil adéquat pour cette sorte de calculs.

Nous avons rappelé que la σ-normalisation, en tant qu'interprétation du $\lambda\sigma$-calcul typé du premier ordre dans le λ-calcul, ne fournit pas la complétude. La technique proposée dans le chapitre 5 pour résoudre ce problème est d'utiliser un système de réécriture auxiliaire dont les règles ne perdent pas l'information. Peut-être cette technique pourrait-elle s'adapter aux problèmes présentés dans les sections suivantes.

Nous avons constaté que la notation de de Bruijn s'est avérée fructueuse dans le cadre des substitutions explicites, tandis que notre essai d'intégrer la notation usuelle n'est pas tout à fait satisfaisant. Dans le $\lambda\nu$-calcul, nous sommes obligés d'imposer une condition au terme de départ qui devrait être préservée par les dérivations pour le bon fonctionnement du calcul. Or, la condition trouvée ne satisfait pas ces expectatives. Malgré ces obstacles nous avons réussi à montrer la confluence d'une stratégie de réduction pour le $\lambda\nu$-calcul. Mais cette confluence est valide modulo α-conversion. Nous ne sommes donc pas arrivés à nous libérer complètement du problème des renommages. Les questions que nous formulons à la fin du chapitre 6 montrent qu'il y a encore des problèmes intéressants à résoudre pour ce calcul.

Seulement la noethérianité faible du τ-calcul a été montrée, en vérifiant que la stratégie leftmost innermost est normalisante. La noethérianité reste encore un problème ouvert. Nous considérons donc que ce calcul est un bon exemple pour tester les nouvelles méthodes de terminaison.

Pour conclure nous énonçons les problèmes ouverts les plus importants concernant les $\lambda\sigma$-calculs : développements finis et normalisation forte des versions typés, et discutons brièvement leur difficulté.

8.1 Développements Finis

Le théorème des développements finis est un théorème central dans le λ-calcul classique, car parmi ses conséquences les plus importantes se trouvent les deux grands théorèmes du λ-calcul : le théorème de confluence et le théorème de standardisation.

Nous présentons ici la formulation la plus simple donnée dans [Bar84], section 11.2. L'autre version, plus compliquée du point de vue formel, mais très claire par contre au niveau intuitif, dit que pour tout terme, ses développements sont toujours finis. Les développements d'un terme $a \in \Lambda_V$, sont des dérivations partant de a où l'on ne s'autorise à réduire que les radicaux présents dans a ou dans les résidus de ces radicaux, tandis que les réductions de radicaux creés sont interdites.

La version plus simple dit qu'une certaine réduction dans une extension de Λ_V est fortement normalisable. Nous définissons d'abord cette extension.

Définition 8.1 *L'ensemble de termes Λ'_V est défini récursivement par :*

$$a ::= x \mid ab \mid \lambda x.a \mid (\lambda_0 x.a)\, b \qquad (x \in V)$$

Nous étendons maintenant la définition de la méta-substitution du λ-calcul classique (cf. définition 1.18) à Λ'_V :

Définition 8.2 *Soient $a, b, c \in \Lambda_V$ et $x \in V$, on définit la méta-substitution en ajoutant aux clauses de la définition 1.18 la clause suivante :*

$$7.\ \ ((\lambda_0 y.a)\, b)[\![c/x]\!] = ((\lambda_0 y.a[\![c/x]\!]))\, b[\![c/x]\!]\,.$$

Définition 8.3 *Le λ_0-calcul est le calcul engendré par la règle β_0 définie ainsi:*

$$(\lambda_0 x.a)\, b \to_{\beta_0} a[\![b/x]\!]\,.$$

Théorème 8.1 (Développements Finis) *Le λ_0-calcul est noethérien.*

Démonstration
Voir [Bar84], proposition 11.2.20.
□

Pour mieux comprendre le fonctionnement de β_0, voyons comment elle réduit le terme non-normalisable classique $(\lambda x.xx)(\lambda x.xx)$ dans sa λ_0-version :

$$(\lambda_0 x.xx)(\lambda x.xx) \to_{\beta_0} (\lambda x.xx)(\lambda x.xx)\,,$$

et $(\lambda x.xx)(\lambda x.xx)$ est une β_0-forme normale.
Remarquons que le sous-terme $\lambda x.xx$ de $(\lambda_0 x.xx)(\lambda x.xx)$ ne peut être indexé, car il n'est pas la première composante d'un radical.

La version du théorème des développements finis pour les $\lambda\sigma$-calculs est encore un problème ouvert. Nous donnons la formulation précise pour le $\lambda\sigma$-calcul.

Définition 8.4 *L'ensemble de termes et substitutions du $\lambda_0\sigma$-calcul est donné récursivement par :*

Termes	$a ::= 1 \mid ab \mid \lambda a \mid (\lambda_0 a)b \mid a[s]$
Substitutions	$s ::= id \mid \uparrow \mid a \cdot s \mid s \circ t\,.$

L'ensemble des règles du $\lambda_0\sigma$-calcul est obtenu en ajoutant à σ la règle suivante :

$$(Beta_0) \qquad (\lambda_0 a)\, b \;\longrightarrow\; a\,[b \cdot id]$$

Conjecture 8.1 *Le $\lambda_0\sigma$-calcul est noethérien.*

La difficulté majeure, si nous voulons nous servir de la noethérianité du λ_0-calcul et de la σ-normalisation, est engendrée par les règles de σ qui "effacent". Nous revenons sur ce problème dans la section suivante.

8.2 Normalisation des λσ-calculs typés

Une propriété essentielle que possède le λ-calcul typé, et qui n'est pas présente dans le λ-calcul non-typé, est la noethérianité. Il existe plusieurs preuves de ce résultat pour le λ-calcul classique, parmi lesquelles celle de Tait [Tai67] a été facilement étendue à des nouveaux systèmes.

Des variations de la preuve de Tait sont celles données dans [HS86] et [GLT90]. Dans ce dernier ouvrage on l'adapte au λ-calcul avec couples et elle est ensuite étendue aux systèmes **F** de Girard et **T** de Gödel.

La preuve de Tait se base sur l'introduction d'une notion abstraite: celle de *réductibilité* dont nous donnons la définition.

Définition 8.5 *L'ensemble de* termes réductibles de type A *est défini par récurrence sur le type* A *ainsi :*

1. *Si* a *est de type atomique* A, a *est réductible ssi il est fortement normalisable.*
2. *Si* a *est de type* $B \to C$, a *est réductible ssi, pour tout terme réductible* b *de type* B, *le terme* $a\,b$ *est réductible de type* C.

La preuve de noethérianité de Tait consiste en deux étapes :

- Tout terme réductible est fortement normalisable.
- Tous les termes sont réductibles.

Malgré nos efforts, la preuve de Tait n'a pas pu être adaptée pour les $\lambda\sigma$-calculs. Cependant, nous formulons la conjecture suivante :

Conjecture 8.2 *Le* $\lambda\sigma$*-calcul typé (cf. section 1.9), le* $\lambda\sigma'$*-calcul typé et le* $\lambda\sigma_{\Uparrow}$*-calcul typé (cf. définition 5.6) sont noethériens.*

Le problème est le suivant : si la conjecture 8.2 était démontrable par un argument à la Tait, une notion adéquate de réductibilité pour le $\lambda\sigma$-calcul devrait être formulée. En fait, la notion introduite dans la définition 8.5 fonctionne bien pour le λ-calcul dont la seule règle est β, tandis que la définition cherchée devrait être capable de gérer toutes les règles de $\lambda\sigma$.

On pourrait aussi être tenté d'utiliser la noethérianité du λ-calcul typé pour montrer celle du $\lambda\sigma$-calcul. Malheureusement, si la manière de relier ces deux calculs est la version typée de l'interprétation introduite dans le chapitre 3, qui d'ailleurs semble être la plus naturelle, on se rend vite compte de l'echec. En effet, étant donné que le $\lambda\sigma$-calcul possède des règles qui "effacent" (*(VarCons)* et *(ShiftCons)*), une dérivation de longueur infinie peut devenir une dérivation interprétée de longueur nulle. Par exemple, supposons une dérivation infinie $a_1 \to_{\lambda\sigma} a_2 \to_{\lambda\sigma} \cdots$. On peut donc construire la dérivation infinie suivante :

$$\uparrow \circ (a_1 : A \cdot s) \to_{\lambda\sigma} \uparrow \circ (a_2 : A \cdot s) \to_{\lambda\sigma} \cdots .$$

Or, toutes les substitutions de cette dérivation, après mise en σ-formes normales deviennent s.

Une formulation adéquate de la technique du chapitre 5 serait peut-être utile pour éclaircir ces problèmes.

Annexe A

Résultats du λ-calcul utilisés

Dans cet annexe nous énonçons les résultats du λ-calcul que nous avons utilisés dans le chapitre 6, et donnons la méthode à suivre pour leurs preuves. Nous renvoyons aux définitions 1.18 et 1.19 de méta-substitution et α-congruence, respectivement.

Lemme A.1 *Soit* $c \in \Lambda_V$. *Si* $x \notin V(c)$, *alors* $\lambda z \in c$ *ssi* $\lambda z \in c[\![x/y]\!]$.

Démonstration
Récurrence sur c .
□

Lemme A.2 *Soit* $c \in \Lambda_V$. *Si* $x \notin V(c)$, *alors* $FV(\lambda x.c[\![x/y]\!]) \subseteq FV(c)$.

Démonstration
Soit $W = FV(\lambda x.c[\![x/y]\!]) = FV(c[\![x/y]\!]) - \{x\}$.
Si $y \notin FV(c)$ alors $c[\![x/y]\!] = c$, et alors $W \subseteq FV(c)$.
Si $y \in FV(c)$ alors $FV(c[\![x/y]\!]) = \{x\} \cup (FV(c) - \{y\})$, et comme $x \notin FV(c)$ on a $W = FV(c) - \{y\} \subseteq FV(c)$.
□

Les lemmes suivants ont pour objectif d'aboutir au lemme A.7. Le reste des lemmes n'est pas utilisé dans le chapitre 6. Leurs preuves sont routinières.

Lemme A.3 *Soient* $a, b \in \Lambda_V$. *Si* $x \in FV(a)$, *alors*

$$a[\![b[\![y/x]\!]/y]\!] \equiv_\alpha a[\![b/y]\!][\![y/x]\!] .$$

Démonstration
Récurrence sur a .
□

Lemme A.4 *Soient* $a, b \in \Lambda_V$. *Si* $x \in FV(a)$ *et* $z \notin FV(ya)$, *alors*

$$a[\![z/y]\!][\![y/x]\!][\![b[\![y/x]\!]/z]\!] \equiv_\alpha a[\![b/y]\!][\![y/x]\!] .$$

Démonstration
Propriétés des substitutions (voir [HS86]) et lemme précédent.
□

Lemme A.5 *Soient* $a, b \in \Lambda_V$. *Si* $x \notin FV(a)$, *alors*

$$a[\![b[\![z/x]\!]/z]\!] \equiv_\alpha a[\![b/z]\!][\![z/x]\!] .$$

Démonstration
Récurrence sur a .
□

Pour la définition de $\rightarrow_\beta$ voir la définition 1.20. Nous utilisons ici la notation $\stackrel{1}{\rightarrow}_\beta$ pour souligner qu'il s'agit d'un pas de β-réduction qui nous intéresse.

Lemme A.6 *Soient* $a, b, c \in \Lambda_V$. *Si* $a \stackrel{1}{\rightarrow}_\beta b$ *et* $y \notin FV(a)$, *alors il existe* c *tel que* $c \equiv_\alpha b[\![y/x]\!]$ *et* $a[\![y/x]\!] \stackrel{1}{\rightarrow}_\beta c$.

Démonstration
Récurrence sur a et utiliser les lemmes A.4 et A.5.
□

Lemme A.7 *Si* $a \stackrel{1}{\rightarrow}_\beta b$ *et* $a \equiv_\alpha a'$, *alors il existe* b' *tel que* $a' \stackrel{1}{\rightarrow}_\beta b'$ *et* $b \equiv_\alpha b'$.

Démonstration
Récurrence sur n = quantité de α-conversions dans $a \equiv_\alpha a'$. Le pas inductif est trivial. Pour démontrer le cas $n = 1$ utiliser le lemme précédent.
□

Annexe B

Confluence locale automatique

Cette annexe contient les démonstrations de la confluence locale du σ-calcul, du σ'-calcul, du $\lambda\sigma'$-calcul et du $SPID$-calcul sur les termes ouverts, obtenues avec le logiciel KB développé à l' I.N.R.I.A..

L'application $a\,b$ est notée app(a,b) ; l'abstraction λa est notée lb(a) ; la fermeture $a[s]$ est notée clos(a,s) ; la substitution $\uparrow$ est notée sh et le cons $a \cdot s$ est noté cons(a,s) .

B.1 Confluence locale du σ-calcul

```
KB Version experimentale Mars 84 [Unix]
List of constructors = ()
List of AC operators = ()
List of infix operators = (o)
List of data-files = ( /kb/subs)
[load /users/formel/rios/kb/subs.l]
Mode (free/auto) = free
KB: complete subs
do you want a trace? y
[fasl /udd/formel/KB/V8/kb-trace.o]
----------------------------------------------------------------
Given set of equations:

clos(1,id) = 1
clos(1,cons(a,x)) = a
clos(app(a,b),x) = app(clos(a,x),clos(b,x))
clos(lb(a),x) = lb(clos(a,cons(1,xosh)))
clos(clos(a,x),y) = clos(a,xoy)
idox = x
shoid = sh
shocons(a,x) = x
cons(a,x)oy = cons(clos(a,y),xoy)
```

```
(xoy)oz = xo(yoz)
------------------------------------------------------------------
R1 : clos(1,id) ---> 1
R2 : clos(1,cons(x,y)) ---> x
R3 : clos(app(x,y),z) ---> app(clos(x,z),clos(y,z))
R4 : clos(lb(x),y) ---> lb(clos(x,cons(1,yosh)))
R5 : clos(clos(x,y),z) ---> clos(x,yoz)
R6 : idox ---> x
R7 : shoid ---> sh
R8 : shocons(x,y) ---> y
R9 : cons(x,y)oz ---> cons(clos(x,z),yoz)
R10 : (xoy)oz ---> xo(yoz)
------------------------------------------------------------------

superposition of rules : RR1 and RR5

normalization of : clos(clos(1,id),x)
R1 : clos(1,id) ---> 1
occurrence : [ 1 ]
substitution :

normal form : clos(1,x)

normalization of : clos(clos(1,id),x)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ ]
substitution :
x' = 1
y' = id

clos(1,idox)
R6 : idox' ---> x'
occurrence : [ 2 ]
substitution :
x' = x

normal form : clos(1,x)

------------------------------------------------------------------

superposition of rules : RR6 and RR10

normalization of : (idox)oy
R6 : idox' ---> x'
occurrence : [ 1 ]
substitution :
x' = x

normal form : xoy

normalization of : (idox)oy
```

```
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ ]
substitution :
x' = id
y' = x

ido(xoy)
R6 : idox' ---> x'
occurrence : [ ]
substitution :
x' = xoy

normal form : xoy

-----------------------------------------------------------------

superposition of rules : RR7 and RR10

normalization of : (shoid)ox
R7 : shoid ---> sh
occurrence : [ 1 ]
substitution :

normal form : shox

normalization of : (shoid)ox
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ ]
substitution :
x' = sh
y' = id

sho(idox)
R6 : idox' ---> x'
occurrence : [ 2 ]
substitution :
x' = x

normal form : shox

-----------------------------------------------------------------

superposition of rules : RR2 and RR5

normalization of : clos(clos(1,cons(x,y)),z)
R2 : clos(1,cons(x',y')) ---> x'
occurrence : [ 1 ]
substitution :
x' = x
y' = y

normal form : clos(x,z)
```

```
normalization of : clos(clos(1,cons(x,y)),z)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ ]
substitution :
x' = 1
y' = cons(x,y)

clos(1,cons(x,y)oz)
R9 : cons(x',y')oz' ---> cons(clos(x',z'),y'oz')
occurrence : [ 2 ]
substitution :
x' = x
y' = y
z' = z

clos(1,cons(clos(x,z),yoz))
R2 : clos(1,cons(x',y')) ---> x'
occurrence : [ ]
substitution :
x' = clos(x,z)
y' = yoz

normal form : clos(x,z)

----------------------------------------------------------------------

superposition of rules : RR8 and RR10

normalization of : (shocons(x,y))oz
R8 : shocons(x',y') ---> y'
occurrence : [ 1 ]
substitution :
x' = x
y' = y

normal form : yoz

normalization of : (shocons(x,y))oz
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ ]
substitution :
x' = sh
y' = cons(x,y)

sho(cons(x,y)oz)
R9 : cons(x',y')oz' ---> cons(clos(x',z'),y'oz')
occurrence : [ 2 ]
substitution :
x' = x
y' = y
```

```
z' = z

shocons(clos(x,z),yoz)
R8 : shocons(x',y') ---> y'
occurrence : [ ]
substitution :
x' = clos(x,z)
y' = yoz

normal form : yoz

----------------------------------------------------------------

superposition of rules : RR5 and RR5

normalization of : clos(clos(clos(x,y),z),u)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ 1 ]
substitution :
x' = x
y' = y
z' = z

clos(clos(x,yoz),u)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ ]
substitution :
x' = x
y' = yoz
z' = u

clos(x,(yoz)ou)
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ 2 ]
substitution :
x' = y
y' = z
z' = u

normal form : clos(x,yo(zou))

normalization of : clos(clos(clos(x,y),z),u)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ ]
substitution :
x' = clos(x,y)
y' = z

clos(clos(x,y),zou)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
```

```
occurrence : [ ]
substitution :
x' = x
y' = y
z' = zou

normal form : clos(x,yo(zou))

----------------------------------------------------------------

superposition of rules : RR10 and RR10

normalization of : ((xoy)oz)ou
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ 1 ]
substitution :
x' = x
y' = y
z' = z

(xo(yoz))ou
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ ]
substitution :
x' = x
y' = yoz
z' = u

xo((yoz)ou)
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ 2 ]
substitution :
x' = y
y' = z
z' = u

normal form : xo(yo(zou))

normalization of : ((xoy)oz)ou
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ ]
substitution :
x' = xoy
y' = z

(xoy)o(zou)
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ ]
substitution :
x' = x
```

```
y' = y
z' = zou

normal form : xo(yo(zou))

----------------------------------------------------------------

superposition of rules : RR3 and RR5

normalization of : clos(clos(app(x,y),z),u)
R3 : clos(app(x',y'),z') ---> app(clos(x',z'),clos(y',z'))
occurrence : [ 1 ]
substitution :
x' = x
y' = y
z' = z

clos(app(clos(x,z),clos(y,z)),u)
R3 : clos(app(x',y'),z') ---> app(clos(x',z'),clos(y',z'))
occurrence : [ ]
substitution :
x' = clos(x,z)
y' = clos(y,z)
z' = u

app(clos(clos(x,z),u),clos(clos(y,z),u))
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ 1 ]
substitution :
x' = x
y' = z
z' = u

app(clos(x,zou),clos(clos(y,z),u))
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ 2 ]
substitution :
x' = y
y' = z
z' = u

normal form : app(clos(x,zou),clos(y,zou))

normalization of : clos(clos(app(x,y),z),u)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ ]
substitution :
x' = app(x,y)
y' = z
```

```
clos(app(x,y),zou)
R3 : clos(app(x',y'),z') ---> app(clos(x',z'),clos(y',z'))
occurrence : [ ]
substitution :
x' = x
y' = y
z' = zou

normal form : app(clos(x,zou),clos(y,zou))

------------------------------------------------------------------

superposition of rules : RR4 and RR5

normalization of : clos(clos(lb(x),y),z)
R4 : clos(lb(x'),y') ---> lb(clos(x',cons(1,y'osh)))
occurrence : [ 1 ]
substitution :
x' = x
y' = y

clos(lb(clos(x,cons(1,yosh))),z)
R4 : clos(lb(x'),y') ---> lb(clos(x',cons(1,y'osh)))
occurrence : [ ]
substitution :
x' = clos(x,cons(1,yosh))
y' = z

lb(clos(clos(x,cons(1,yosh)),cons(1,zosh)))
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ 1 ]
substitution :
x' = x
y' = cons(1,yosh)
z' = cons(1,zosh)

lb(clos(x,cons(1,yosh)ocons(1,zosh)))
R9 : cons(x',y')oz' ---> cons(clos(x',z'),y'oz')
occurrence : [ 1 2 ]
substitution :
x' = 1
y' = yosh
z' = cons(1,zosh)

lb(clos(x,cons(clos(1,cons(1,zosh)),(yosh)ocons(1,zosh))))
R2 : clos(1,cons(x',y')) ---> x'
occurrence : [ 1 2 1 ]
substitution :
x' = 1
```

```
y' = zosh

lb(clos(x,cons(1,(yosh)ocons(1,zosh))))
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ 1 2 2 ]
substitution :
x' = y
y' = sh
z' = cons(1,zosh)

lb(clos(x,cons(1,yo(shocons(1,zosh)))))
R8 : shocons(x',y') ---> y'
occurrence : [ 1 2 2 2 ]
substitution :
x' = 1
y' = zosh

normal form : lb(clos(x,cons(1,yo(zosh))))

normalization of : clos(clos(lb(x),y),z)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ ]
substitution :
x' = lb(x)
y' = y

clos(lb(x),yoz)
R4 : clos(lb(x'),y') ---> lb(clos(x',cons(1,y'osh)))
occurrence : [ ]
substitution :
x' = x
y' = yoz

lb(clos(x,cons(1,(yoz)osh)))
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ 1 2 2 ]
substitution :
x' = y
y' = z
z' = sh

normal form : lb(clos(x,cons(1,yo(zosh))))

------------------------------------------------------------------

superposition of rules : RR9 and RR10

normalization of : (cons(x,y)oz)ou
R9 : cons(x',y')oz' ---> cons(clos(x',z'),y'oz')
occurrence : [ 1 ]
```

```
substitution :
x' = x
y' = y
z' = z

cons(clos(x,z),yoz)ou
R9 : cons(x',y')oz' ---> cons(clos(x',z'),y'oz')
occurrence : [ ]
substitution :
x' = clos(x,z)
y' = yoz
z' = u

cons(clos(clos(x,z),u),(yoz)ou)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ 1 ]
substitution :
x' = x
y' = z
z' = u

cons(clos(x,zou),(yoz)ou)
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ 2 ]
substitution :
x' = y
y' = z
z' = u

normal form : cons(clos(x,zou),yo(zou))

normalization of : (cons(x,y)oz)ou
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ ]
substitution :
x' = cons(x,y)
y' = z

cons(x,y)o(zou)
R9 : cons(x',y')oz' ---> cons(clos(x',z'),y'oz')
occurrence : [ ]
substitution :
x' = x
y' = y
z' = zou

normal form : cons(clos(x,zou),yo(zou))
Time: 9 s. [GC: 0 s. ]
Complete set: *subs
```

B.2 Confluence locale de σ' et de $\lambda\sigma'$

Nous avons conservé de la transcription de la session correspondant au $\lambda\sigma'$-calcul seulement les superpositions de paires critiques nécessitant l'introduction de nouvelles règles (R1, R7, R11, R12, R13). On constate ainsi la confluence locale du $\lambda\sigma'$-calcul.

Quant à celle du σ'-calcul, il suffit de vérifier que la règle *(Beta)* (R13) n'est jamais utilisée pour fermer une paire critique engendrée par les autres règles.

Voici les règles fournies au logiciel KB :

```
-----------------------------------------------------------------

R1 : clos(x,id) ---> x

R2 : clos(1,cons(x,y)) ---> x

R3 : clos(app(x,y),z) ---> app(clos(x,z),clos(y,z))

R4 : clos(lb(x),y) ---> lb(clos(x,cons(1,yosh)))

R5 : clos(clos(x,y),z) ---> clos(x,yoz)

R6 : idox ---> x

R7 : xoid ---> x

R8 : shocons(x,y) ---> y

R9 : cons(x,y)oz ---> cons(clos(x,z),yoz)

R10 : (xoy)oz ---> xo(yoz)

R11 : cons(1,sh) ---> id

R12 : cons(clos(1,x),shox) ---> x

R13 : app(lb(x),y) ---> clos(x,cons(y,id))

-----------------------------------------------------------------

superposition of rules : RR6 and RR7

normalization of : idoid
R6 : idox' ---> x'
occurrence : [ ]
substitution :
x' = id

normal form : id

normalization of : idoid
R7 : x'oid ---> x'
occurrence : [ ]
substitution :
```

```
x' = id

normal form : id

-----------------------------------------------------------------

superposition of rules : RR11 and RR2

normalization of : clos(1,cons(1,sh))
R11 : cons(1,sh) ---> id
occurrence : [ 2 ]
substitution :

clos(1,id)
R1 : clos(x',id) ---> x'
occurrence : [ ]
substitution :
x' = 1

normal form : 1

normalization of : clos(1,cons(1,sh))
R2 : clos(1,cons(x',y')) ---> x'
occurrence : [ ]
substitution :
x' = 1
y' = sh

normal form : 1

-----------------------------------------------------------------

superposition of rules : RR11 and RR8

normalization of : shocons(1,sh)
R11 : cons(1,sh) ---> id
occurrence : [ 2 ]
substitution :

shoid
R7 : x'oid ---> x'
occurrence : [ ]
substitution :
x' = sh

normal form : sh

normalization of : shocons(1,sh)
R8 : shocons(x',y') ---> y'
occurrence : [ ]
substitution :
x' = 1
```

```
y' = sh

normal form : sh

-----------------------------------------------------------------

superposition of rules : RR1 and RR12

normalization of : cons(clos(1,id),shoid)
R1 : clos(x',id) ---> x'
occurrence : [ 1 ]
substitution :
x' = 1

cons(1,shoid)
R7 : x'oid ---> x'
occurrence : [ 2 ]
substitution :
x' = sh

cons(1,sh)
R11 : cons(1,sh) ---> id
occurrence : [ ]
substitution :

normal form : id

normalization of : cons(clos(1,id),shoid)
R12 : cons(clos(1,x'),shox') ---> x'
occurrence : [ ]
substitution :
x' = id

normal form : id

-----------------------------------------------------------------

superposition of rules : RR7 and RR12

normalization of : cons(clos(1,id),shoid)
R7 : x'oid ---> x'
occurrence : [ 2 ]
substitution :
x' = sh

cons(clos(1,id),sh)
R1 : clos(x',id) ---> x'
occurrence : [ 1 ]
substitution :
x' = 1

cons(1,sh)
```

```
R11 : cons(1,sh) ---> id
occurrence : [ ]
substitution :

normal form : id

normalization of : cons(clos(1,id),shoid)
R12 : cons(clos(1,x'),shox') ---> x'
occurrence : [ ]
substitution :
x' = id

normal form : id

----------------------------------------------------------------

superposition of rules : RR1 and RR5

normalization of : clos(clos(x,id),y)
R1 : clos(x',id) ---> x'
occurrence : [ 1 ]
substitution :
x' = x

normal form : clos(x,y)

normalization of : clos(clos(x,id),y)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ ]
substitution :
x' = x
y' = id

clos(x,idoy)
R6 : idox' ---> x'
occurrence : [ 2 ]
substitution :
x' = y

normal form : clos(x,y)

----------------------------------------------------------------

superposition of rules : RR1 and RR5

normalization of : clos(clos(x,y),id)
R1 : clos(x',id) ---> x'
occurrence : [ ]
substitution :
x' = clos(x,y)

normal form : clos(x,y)
```

```
normalization of : clos(clos(x,y),id)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ ]
substitution :
x' = x
y' = y
z' = id

clos(x,yoid)
R7 : x'oid ---> x'
occurrence : [ 2 ]
substitution :
x' = y

normal form : clos(x,y)

----------------------------------------------------------------

superposition of rules : RR7 and RR10

normalization of : (xoid)oy
R7 : x'oid ---> x'
occurrence : [ 1 ]
substitution :
x' = x

normal form : xoy

normalization of : (xoid)oy
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ ]
substitution :
x' = x
y' = id

xo(idoy)
R6 : idox' ---> x'
occurrence : [ 2 ]
substitution :
x' = y

normal form : xoy

----------------------------------------------------------------

superposition of rules : RR7 and RR10

normalization of : (xoy)oid
R7 : x'oid ---> x'
occurrence : [ ]
substitution :
```

```
x' = xoy

normal form : xoy

normalization of : (xoy)oid
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ ]
substitution :
x' = x
y' = y
z' = id

xo(yoid)
R7 : x'oid ---> x'
occurrence : [ 2 ]
substitution :
x' = y

normal form : xoy

----------------------------------------------------------------

superposition of rules : RR12 and RR2

normalization of : clos(1,cons(clos(1,x),shox))
R12 : cons(clos(1,x'),shox') ---> x'
occurrence : [ 2 ]
substitution :
x' = x

normal form : clos(1,x)

normalization of : clos(1,cons(clos(1,x),shox))
R2 : clos(1,cons(x',y')) ---> x'
occurrence : [ ]
substitution :
x' = clos(1,x)
y' = shox

normal form : clos(1,x)

----------------------------------------------------------------

superposition of rules : RR2 and RR12

normalization of : cons(clos(1,cons(x,y)),shocons(x,y))
R2 : clos(1,cons(x',y')) ---> x'
occurrence : [ 1 ]
substitution :
x' = x
y' = y
```

```
cons(x,shocons(x,y))
R8 : shocons(x',y') ---> y'
occurrence : [ 2 ]
substitution :
x' = x
y' = y

normal form : cons(x,y)

normalization of : cons(clos(1,cons(x,y)),shocons(x,y))
R12 : cons(clos(1,x'),shox') ---> x'
occurrence : [ ]
substitution :
x' = cons(x,y)

normal form : cons(x,y)

------------------------------------------------------------------

superposition of rules : RR12 and RR8

normalization of : shocons(clos(1,x),shox)
R12 : cons(clos(1,x'),shox') ---> x'
occurrence : [ 2 ]
substitution :
x' = x

normal form : shox

normalization of : shocons(clos(1,x),shox)
R8 : shocons(x',y') ---> y'
occurrence : [ ]
substitution :
x' = clos(1,x)
y' = shox

normal form : shox

------------------------------------------------------------------

superposition of rules : RR8 and RR12

normalization of : cons(clos(1,cons(x,y)),shocons(x,y))
R8 : shocons(x',y') ---> y'
occurrence : [ 2 ]
substitution :
x' = x
y' = y

cons(clos(1,cons(x,y)),y)
R2 : clos(1,cons(x',y')) ---> x'
occurrence : [ 1 ]
```

```
substitution :
x' = x
y' = y

normal form : cons(x,y)

normalization of : cons(clos(1,cons(x,y)),shocons(x,y))
R12 : cons(clos(1,x'),shox') ---> x'
occurrence : [ ]
substitution :
x' = cons(x,y)

normal form : cons(x,y)

------------------------------------------------------------------

superposition of rules : RR1 and RR3

normalization of : clos(app(x,y),id)
R1 : clos(x',id) ---> x'
occurrence : [ ]
substitution :
x' = app(x,y)

normal form : app(x,y)

normalization of : clos(app(x,y),id)
R3 : clos(app(x',y'),z') ---> app(clos(x',z'),clos(y',z'))
occurrence : [ ]
substitution :
x' = x
y' = y
z' = id

app(clos(x,id),clos(y,id))
R1 : clos(x',id) ---> x'
occurrence : [ 1 ]
substitution :
x' = x

app(x,clos(y,id))
R1 : clos(x',id) ---> x'
occurrence : [ 2 ]
substitution :
x' = y

normal form : app(x,y)

------------------------------------------------------------------

superposition of rules : RR1 and RR4
```

```
normalization of : clos(lb(x),id)
R1 : clos(x',id) ---> x'
occurrence : [ ]
substitution :
x' = lb(x)

normal form : lb(x)

normalization of : clos(lb(x),id)
R4 : clos(lb(x'),y') ---> lb(clos(x',cons(1,y'osh)))
occurrence : [ ]
substitution :
x' = x
y' = id

lb(clos(x,cons(1,idosh)))
R6 : idox' ---> x'
occurrence : [ 1 2 2 ]
substitution :
x' = sh

lb(clos(x,cons(1,sh)))
R11 : cons(1,sh) ---> id
occurrence : [ 1 2 ]
substitution :

lb(clos(x,id))
R1 : clos(x',id) ---> x'
occurrence : [ 1 ]
substitution :
x' = x

normal form : lb(x)

------------------------------------------------------------------

superposition of rules : RR7 and RR9

normalization of : cons(x,y)oid
R7 : x'oid ---> x'
occurrence : [ ]
substitution :
x' = cons(x,y)

normal form : cons(x,y)

normalization of : cons(x,y)oid
R9 : cons(x',y')oz' ---> cons(clos(x',z'),y'oz')
occurrence : [ ]
substitution :
x' = x
```

```
y' = y
z' = id

cons(clos(x,id),yoid)
R1 : clos(x',id) ---> x'
occurrence : [ 1 ]
substitution :
x' = x

cons(x,yoid)
R7 : x'oid ---> x'
occurrence : [ 2 ]
substitution :
x' = y

normal form : cons(x,y)

-----------------------------------------------------------------

superposition of rules : RR11 and RR9

normalization of : cons(1,sh)ox
R11 : cons(1,sh) ---> id
occurrence : [ 1 ]
substitution :

idox
R6 : idox' ---> x'
occurrence : [ ]
substitution :
x' = x

normal form : x

normalization of : cons(1,sh)ox
R9 : cons(x',y')oz' ---> cons(clos(x',z'),y'oz')
occurrence : [ ]
substitution :
x' = 1
y' = sh

cons(clos(1,x),shox)
R12 : cons(clos(1,x'),shox') ---> x'
occurrence : [ ]
substitution :
x' = x

normal form : x

-----------------------------------------------------------------

superposition of rules : RR12 and RR9
```

```
normalization of : cons(clos(1,x),shox)oy
R12 : cons(clos(1,x'),shox') ---> x'
occurrence : [ 1 ]
substitution :
x' = x

normal form : xoy

normalization of : cons(clos(1,x),shox)oy
R9 : cons(x',y')oz' ---> cons(clos(x',z'),y'oz')
occurrence : [ ]
substitution :
x' = clos(1,x)
y' = shox

cons(clos(clos(1,x),y),(shox)oy)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ 1 ]
substitution :
x' = 1
y' = x
z' = y

cons(clos(1,xoy),(shox)oy)
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ 2 ]
substitution :
x' = sh
y' = x
z' = y

cons(clos(1,xoy),sho(xoy))
R12 : cons(clos(1,x'),shox') ---> x'
occurrence : [ ]
substitution :
x' = xoy

normal form : xoy

------------------------------------------------------------------

superposition of rules : RR13 and RR3

normalization of : clos(app(lb(x),y),z)
R13 : app(lb(x'),y') ---> clos(x',cons(y',id))
occurrence : [ 1 ]
substitution :
x' = x
y' = y
```

```
clos(clos(x,cons(y,id)),z)
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ ]
substitution :
x' = x
y' = cons(y,id)
z' = z

clos(x,cons(y,id)oz)
R9 : cons(x',y')oz' ---> cons(clos(x',z'),y'oz')
occurrence : [ 2 ]
substitution :
x' = y
y' = id
z' = z

clos(x,cons(clos(y,z),idoz))
R6 : idox' ---> x'
occurrence : [ 2 2 ]
substitution :
x' = z

normal form : clos(x,cons(clos(y,z),z))

normalization of : clos(app(lb(x),y),z)
R3 : clos(app(x',y'),z') ---> app(clos(x',z'),clos(y',z'))
occurrence : [ ]
substitution :
x' = lb(x)
y' = y

app(clos(lb(x),z),clos(y,z))
R4 : clos(lb(x'),y') ---> lb(clos(x',cons(1,y'osh)))
occurrence : [ 1 ]
substitution :
x' = x
y' = z

app(lb(clos(x,cons(1,zosh))),clos(y,z))
R13 : app(lb(x'),y') ---> clos(x',cons(y',id))
occurrence : [ ]
substitution :
x' = clos(x,cons(1,zosh))
y' = clos(y,z)

clos(clos(x,cons(1,zosh)),cons(clos(y,z),id))
R5 : clos(clos(x',y'),z') ---> clos(x',y'oz')
occurrence : [ ]
substitution :
```

```
x' = x
y' = cons(1,zosh)
z' = cons(clos(y,z),id)

clos(x,cons(1,zosh)ocons(clos(y,z),id))
R9 : cons(x',y')oz' ---> cons(clos(x',z'),y'oz')
occurrence : [ 2 ]
substitution :
x' = 1
y' = zosh
z' = cons(clos(y,z),id)

clos(x,cons(clos(1,cons(clos(y,z),id)),(zosh)ocons(clos(y,z),id)))
R2 : clos(1,cons(x',y')) ---> x'
occurrence : [ 2 1 ]
substitution :
x' = clos(y,z)
y' = id

clos(x,cons(clos(y,z),(zosh)ocons(clos(y,z),id)))
R10 : (x'oy')oz' ---> x'o(y'oz')
occurrence : [ 2 2 ]
substitution :
x' = z
y' = sh
z' = cons(clos(y,z),id)

clos(x,cons(clos(y,z),zo(shocons(clos(y,z),id))))
R8 : shocons(x',y') ---> y'
occurrence : [ 2 2 2 ]
substitution :
x' = clos(y,z)
y' = id

clos(x,cons(clos(y,z),zoid))
R7 : x'oid ---> x'
occurrence : [ 2 2 ]
substitution :
x' = z

normal form : clos(x,cons(clos(y,z),z))

Time: 15 s. [GC: 0 s. ]

----------------------------------------------------------------

Complete set: *lsubs
```

B.3 Confluence locale du $SPID$-calcul

Pour vérifier la confluence locale du $SPID$-calcul il suffit en fait de constater que dans la session transcrite dans la section précédente chaque surperposition des $SPID$-règles (R1, R7, R11, R12) est gérée seulement avec des $SPID$-règles. Puisque'il n'y a que deux superpositions nous reproduisons quand même la session obtenue en fournissant au logiciel KB seulement les $SPID$-règles :

```
-----------------------------------------------------------------

R1 : clos(x,id) ---> x
R2 : xoid ---> x
R3 : cons(1,sh) ---> id
R4 : cons(clos(1,x),shox) ---> x
-----------------------------------------------------------------

superposition of rules : RR1 and RR4

normalization of : cons(clos(1,id),shoid)
R1 : clos(x',id) ---> x'
occurrence : [ 1 ]
substitution :
x' = 1

cons(1,shoid)
R2 : x'oid ---> x'
occurrence : [ 2 ]
substitution :
x' = sh

cons(1,sh)
R3 : cons(1,sh) ---> id
occurrence : [ ]
substitution :

normal form : id

normalization of : cons(clos(1,id),shoid)
R4 : cons(clos(1,x'),shox') ---> x'
occurrence : [ ]
substitution :
x' = id

normal form : id

-----------------------------------------------------------------

superposition of rules : RR2 and RR4

normalization of : cons(clos(1,id),shoid)
R2 : x'oid ---> x'
occurrence : [ 2 ]
```

```
substitution :
x' = sh

cons(clos(1,id),sh)
R1 : clos(x',id) ---> x'
occurrence : [ 1 ]
substitution :
x' = 1

cons(1,sh)
R3 : cons(1,sh) ---> id
occurrence : [ ]
substitution :

normal form : id

normalization of : cons(clos(1,id),shoid)
R4 : cons(clos(1,x'),shox') ---> x'
occurrence : [ ]
substitution :
x' = id

normal form : id

Time: 1 s. [GC: 0 s. ]

----------------------------------------------------------------------

Complete set: *spid
```

Annexe C

Sous-systèmes de CCL

Les termes clos de CCL sont définis par :

$$x ::= Id \mid Fst \mid Snd \mid App \mid \Lambda x \mid x \circ y \mid < x , y >$$

Le système $\mathcal{E}$ est donné par l'ensemble de règles suivantes :

$$\begin{array}{ll}
(Ass) & (x \circ y) \circ z \rightarrow x \circ (y \circ z) \\
(IdL) & Id \circ x \rightarrow x \\
(Fst) & Fst \circ < x , y > \rightarrow x \\
(Snd) & Snd \circ < x , y > \rightarrow y \\
(Dpair) & < x , y > \circ z \rightarrow < x \circ z , y \circ z > \\
(D\Lambda) & (\Lambda x) \circ y \rightarrow \Lambda(x \circ < y \circ Fst , Snd >) \\
(FId) & Fst \circ Id \rightarrow Fst \\
(SId) & Snd \circ Id \rightarrow Snd
\end{array}$$

Le système SUBST est obtenu en ajoutant aux règles de $\mathcal{E}$ les règles :

$$\begin{array}{ll}
(IdR) & x \circ Id \rightarrow x \\
(FSI) & < Fst , Snd > \rightarrow Id \\
(SP) & < Fst \circ x , Snd \circ x > \rightarrow x
\end{array}$$

et en enlevant les règles *(FId)* et *(SId)* qui sont des instances de *(IdR)*.

La première preuve de terminaison de SUBST donnée dans [HL86] consiste à montrer que le nombre d'applications de la règle *(DΛ)*, dans toute dérivation d'un terme donné, est borné par une valeur ne dépendant que du terme. Il faut pour cela estimer le nombre de symboles Λ et de symboles $<, >$ qui peuvent être présents dans un même réduit. Ces estimations sont données par des fonctions qui vérifient la propriété de sous-terme ($f(C[s]) \geq f(s)$) mais ne sont pas compatibles avec la structure (si $f(s) \geq f(t)$, alors $f(C[s]) \geq f(C[t])$). La fonction estimant le nombre de paires est très compliquée et nécessite la construction d'une liste auxiliaire servant à représenter un terme. Cette dernière construction ne vérifie pas la propriété de sous-terme mais est compatible avec la structure.

Dans [Har89], T. Hardin a montré que le système $\mathcal{E}$ + *(Beta)* est confluent sur un sous-ensemble $\mathcal{D}$ de termes, qui contient les termes du λ-calcul. Ce système implémente donc la β-réduction avec gestion explicite des substitutions. Par contre, le calcul SUBST + *(Beta)* n'est pas confluent. Ces résultats sont obtenus en utilisant la méthode d'interprétation (cf. lemme 1.4). L'interprétation est faite par $\mathcal{E}$-normalisation. La non-confluence est montrée en construisant un contre-exemple pour le λ-calcul avec couples, assez différent de celui de Klop, et en le traduisant dans CCL.

Bibliographie

[ACCL90] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. Technical Report 1176, INRIA, 1990. A paraître dans Journal of Functional Programming.

[Bar84] H. Barendregt. *The Lambda Calculus : Its Syntax and Semantics (revised edition)*. North Holland, 1984.

[BS81] S. Burris and H. P. Sankappanavar. *A Course in Universal Algebra*. Springer-Verlag, 1981.

[CCM87] G. Cousineau, P.-L. Curien, and M. Mauny. The Categorical Abstract Machine. *Science of Computer Programming*, 8:173–202, 1987.

[CF58] H. B. Curry and R. Feys. *Combinatory Logic*, volume 1. North Holland, 1958.

[CH88] T. Coquand and G. Huet. The calculus of constructions. *Information and computation*, 76(2/3):95–120, February/March 1988.

[CHL91] P.-L. Curien, T. Hardin, and J.-J. Lévy. Confluence properties of weak and strong calculi of explicit substitutions. Technical report, Centre d'Etudes et de Recherche en Informatique, CNAM, 1991.

[CHR91] P.-L. Curien, T. Hardin, and A. Ríos. Normalisation Forte du Calcul des Substitutions. Technical Report 91-16, LIENS - Ecole Normale Supérieure, 1991.

[CHR92] P.-L. Curien, T. Hardin, and A. Ríos. Strong normalization of substitutions. In I.M. Havel and V. Koubek, editors, *MFCS*, pages 209–217, Prague, 1992. LNCS 629, Springer-Verlag.

[CHS72] H. B. Curry, J. R. Hindley, and J. P. Seldin. *Combinatory Logic*, volume 2. North Holland, 1972.

[Chu41] A. Church. *The Calculi of Lambda-Conversion*. Princeton University Press, Princeton, 1941.

[CR91] P.-L. Curien and A. Ríos. Un résultat de Complétude pour les substitutions explicites. *Comptes Rendus de l'Académie des Sciences*, 312, I:471–476, 1991.

[Cur86] P.-L. Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming.* Pitman, 1986. Revised edition : Birkhäuser (1993).

[Cur88] P.-L. Curien. The $\lambda\rho$-calculi: an Abstract Framework for Closures. Technical report, LIENS - Ecole Normale Supérieure, 1988.

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

[dB78] N. de Bruijn. Lambda-Calculus notation with namefree formulas involving symbols that represent reference transforming mappings. *Indag. Mat.*, 40:348–356, 1978.

[Ehr88] T. Ehrhard. *Une Sémantique Catégorique des Types Dépendants. Application au Calcul des Constructions.* PhD thesis, Université Paris VII, Paris, France, 1988.

[GLT90] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types.* Cambridge University Press, 1990.

[Gra79] G. Gratzer. *Universal Algebra.* Springer-Verlag, 1979.

[Har87] T. Hardin. *Résultats de Confluence pour les Règles Fortes de la Logique Combinatoire Catégorique et Liens avec les Lambda-Calculs.* PhD thesis, Université Paris VII, Paris, France, 1987.

[Har89] T. Hardin. Confluence Results for the Pure Strong Categorical Logic CCL : λ-calculi as Subsystems of CCL. *Theoretical Computer Science*, 65(2):291–342, 1989.

[Har92] T. Hardin. Eta-conversion for the languages of explicit substitutions. Technical report, Centre d'Etudes et de Recherche en Informatique, CNAM, 1992.

[HL86] T. Hardin and A. Laville. Proof of Termination of the Rewriting System SUBST on CCL. *Theoretical Computer Science*, 46:305–312, 1986.

[HL89] T. Hardin and J.-J. Lévy. A Confluent Calculus of Substitutions. *France-Japan Artificial Intelligence and Computer Science Symposium*, December 1989.

[HS86] J. R. Hindley and J. P. Seldin. *Introduction to combinators and lambda-calculus.* London Mathematical Society, 1986.

[Hue80] G. Huet. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. *Journal of the Association for Computing Machinery*, 27:797–821, October 1980.

[KB70] D. Knuth and P. Bendix. Simple Word Problems in Universal Algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[Klo80] J. W. Klop. Combinatory Reduction Systems. *Mathematical Center Tracts*, 27, 1980.

[Klo90] J. W. Klop. Term Rewriting Systems. Technical report, Centrum voor Wiskunde en Informatica, 1990.

[Mau85] M. Mauny. *Compilation des langages fonctionnels dans les combinateurs catégoriques. Application au langage ML.* PhD thesis, Université Paris VII, Paris, France, 1985.

[ML85] P. Martin-Löf. *Intuitionistic type theory.* Bibliopolis, 1985.

[MM82] A. Martelli and U. Montanari. An Efficient Unification Algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, 1982.

[MS86] M. Mauny and A. Suárez. Implementing functional languages in the Categorical Abstract Machine. *Proceedings of the Symposium on LISP and Functional Programming*, 1986.

[New42] M. H. A. Newman. On theories with a combinatorial definition of "equivalence". *Ann. of Math.*, 43(2):223–243, 1942.

[Ros73] B. K. Rosen. Tree-Manipulating Systems and Church-Rosser Theorems. *Journal of the Association for Computing Machinery*, 20:160–187, January 1973.

[Sle85] M. Sleep. Issues for implementing lambda languages. *Notes for Ustica Workshop*, September 1985.

[Sta78] J. Staples. A Graph-like Lambda-Calculus for which leftmost-outer-most reduction is optimal. In Claus, editor, *Graph Grammars and their Applications to Computer Science and Biology.* Springer-Verlag, 1978.

[Str88] T. Streicher. *Correctness and completeness of a semantic of the calculus of constructions with respect to interpretation in doctrines of constructions.* PhD thesis, Passau, 1988.

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

[Zan92] H. Zantema. Termination of Term Rewriting by Interpretation. Technical Report RUU-CS-92-14, Department of Computer Science, Utrecht University, 1992.

Index