Les ressources explicites vues par la théorie de la réécriture.

Fabien Renaud

Université Paris Diderot - Paris 7

École doctorale
Sciences Mathématiques de Paris Centre

THÈSE

pour l'obtention du diplôme de

DOCTEUR DE L'UNIVERSITÉ PARIS DIDEROT

Spécialité Informatique

LES RESSOURCES EXPLICITES VUES PAR LA THÉORIE DE LA RÉÉCRITURE

Présentée et soutenue publiquement par

Fabien RENAUD

le 7 Décembre 2011, devant le jury composé de

Silvia Ghilezan	Rapporteur
Stefano Guerrini	Examinateur
Delia Kesner	Directeur
Pierre Lescanne	Examinateur
Ralph Matthes	Examinateur
Simona Ronchi Della Rocca	Rapporteur

Remerciements

Cette thèse fût longue et de nombreuses personnes m'ont soutenu, scientifiquement ou personnellement, pour arriver au manuscrit que vous lisez.

J'aimerais tout d'abord remercier Delia Kesner, directrice de stage de master puis de thèse. Tout au long de ce périple scientifique, elle m'a guidé scientifiquement, soutenu dans les moments difficiles, et m'a poussé pour aller de l'avant. Au fil des discussions et collaborations, j'ai appris la rigueur scientifique [1] et syntaxique [2]. Il va sans dire que cette thèse n'aurait pas vu le jour sans elle.

Merci également à Silvia Ghilezan et Simona Ronchi Della Rocca qui m'ont fait l'honneur de bien vouloir être rapporteurs [3]. Leurs lectures attentives et points de vue ont permis d'améliorer la thèse et de prendre du recul. Merci également à Stefano Guerrini, Piere Lescanne, et Ralph Matthes d'avoir bien voulu faire partie de mon jury. Leurs présences m'honorent.

Je vais maintenant essayer de procéder à des remerciements plus ou moins chronologiques, de l'ensemble des personnes ayant été impliqué de près ou de loin dans cette thèse. Tout commence évidemment par la famille qui m'a toujours soutenu même lorsqu'elle ne comprenait pas vraiment le pourquoi du comment, voire rien du tout. Passons directement au DUT GTR que j'ai suivi à Paris 13 (et oui j'ai soudé des fibres optiques dans une vie antérieure). Les cours de cryptographie de Frédéric Toumazet avec les yeux qui brillent m'ont montré la voie de la recherche, et m'ont décidé à suivre une licence. Par la suite, Christophe Fouqueré m'a poussé à suivre les cours du MPRI. Une fois le master commencé les choses s'accélèrent. Merci à la bande avec qui j'ai pu travailler et notamment Mehdi, Philippe, et Till. Arrivé en thèse, grâce aux bureaux surpeuplés de PPS, la liste des remerciements s'agrandit. D'abord le 6C10 que j'ai connu avec Barbara, Nicolas, Mauricio, Pierre, Samuel. Et "ceux d'en face" : Christine, Gim, Grégoire, Séverine, Mehdi, Stéphane G. Et puis certains s'en vont et d'autres arrivent : Antoine, Alexis, Beniamino, Bob S, Boris, Carlos, Flavio, Flavien, Jaap, Jakob, Jonas, Gabriel, Glycojenga, Marie-Aude, Mathias P&K, Siddarth, Stéphane Z, Thibaut, Whathemovie, Zhiwu. S'en est suivi un exil forcé en 6A51 avec Alberto, Giulio, Guillaume, Sarah, Shahin. Merci également à Renaud pour les cantines compte-rendu du week-end et le jaja. Je n'oublie évidemment pas les secrétaires, sans qui les chercheurs ne sont pas grand chose : Audrey, Michèle, Régine, Véronique, et bien sûr Odile.

J'en profite pour faire un petit apparté pour remercier toutes les personnes qui m'ont si gentiment accueilli lors de mon séjour à Buenos Aires : Alejandro Ríos, et tout particulièrement Eduardo Bonelli. Les jeunes et moins jeunes ne sont pas oubliés : Beta, Carlos (pour la conduite), Grabriela, Pablo (pour ne pas avoir passé un anniversaire tout seul).

Je continue maintenant avec les remerciements plus spécifiques, des personnes avec qui j'ai pu interagir. Le premier, à qui j'ai pu poser toutes les questions que

[1] notamment avec des brouillons plus rouges que noirs

[2] en anglais, et même en français

[3] j'ai laissé la forme masculine car rapporteuse est trop connoté, et rapportrice trop bizarre

je voulais (réécriture, LaTeX,...) est sûrement Thibault (juste pour t'embêter !). J'ai pu discuter également avec Antoine de complexité implicite, avec Flavio, Mehdi, et Stéphane G de Coq (or ?), avec Jonas d'extensions de Kahn, avec Stéphane Z de faisage de Coq (alors les substitutions on les représente comment ?), avec Pierre de bornes de réductions. Avec Beniamino j'ai pu discuter de tellement de choses que je ne suis pas sûr de me rappeler : complexité, KAM, LL, substitutions explicites, métaconfluence. Merci à Guillaume pour mathord, sans qui ma thèse aurait été un peu plus moche. Merci également à la Kajsija d'Aleks qui m'a donné la force de rédiger.

Merci également au cercle qui ne m'a pas vraiment apporté de réponses à mes questions scientifiques mais qui à défaut m'a permis de décompresser : Adrien, Alex, Arnaud, Bapt, Ben, Bidoche, Boy, GG, Paulo, Yannou ; ainsi qu'aux copines plus ou moins respectives. Merci à Chapitre 20 qui a rhytmé les jeudis de ma rédaction par des dégustations ma foi sympathiques. Merci à la forêt de Fontainebleau, pour tous ces moments de détente, ainsi qu'à tous les varappeurs que j'y ai croisé : Chien-loup, Igor, Ismaël, Jérôme, Julien, Philippe, Manu, Renaud,...

Un très grand merci à Anna, pour ses relectures de thèse, mais également pour le support affectif, logistique, et culinaire lors de toutes ces années. Puss och kram ! (är det för mycket ?)

Mes dernières pensées lors de la rédaction de ces remerciements vont à ma mère. Ce manuscrit lui est dédié.

Table des matières

Chapitre 1

Introduction

Le λ-calcul

De la fondation des mathématiques... En 1933, Alonzo Church propose un système [Chu33] connu en tant que λ-*calcul*, ayant pour but de donner une base logique à la notion de fonction. La notoriété de ce calcul grandit dans les années qui suivirent. Kleene et Church [Kle36, Chu36] montrèrent que la classe des fonctions définissables à l'aide du λ-calcul est équivalente aux fonctions récursives de Herbrand-Gödel, elles-mêmes équivalentes aux fonctions calculables par une machine de Turing [Tur37]. Avec l'invention de l'ordinateur, il est maintenant plus facile de comprendre l'important de ce résultat : si on ne peut pas écrire une fonction avec un λ-terme, alors un ordinateur ne pourra pas la calculer.

...Au premier langage de programmation fonctionnelle. Malgré sa mise au point avant le premier ordinateur, le λ-calcul reste le langage de programmation fonctionnelle théorique universel, et se trouve être à la base des langages réels, tel que ceux de la famille ML (OCaml [Oca], SML/NJ [SML], ...), Haskell [Has], ou encore LISP (CommonLisp [LIS], Scheme [Big],...). Son succès peut être expliqué par sa simplicité, et par le fait qu'il y seulement deux briques de base : les *termes* du λ-calcul, et la *réduction* les reliant. Un terme est, comme en mathématiques, soit une *variable*, notée x, y, z, soit une *fonction* notée $\lambda x.t$ (c'est à dire une fonction qui prend x en argument et retourne t), soit l'*application* d'une fonction t à son argument u noté $t\ u$. D'un point de vue programmation, $\lambda x.t$ est un programme qui prend x en entrée et retourne t, et $t\ u$ l'exécution du programme t avec u passé en argument. Une grande partie de la force du λ-calcul réside dans le fait qu'un argument peut lui même être à son tour une fonction. On dira que les fonctions sont des citoyens de première classe. Une autre propriété essentielle est le fait que lorsque l'on ajoute une information supplémentaire, à savoir le *typage* des λ-termes, les « programmes bien typés ne plantent jamais » comme l'a montré Robin Milner [Mil78]. On peut même s'assurer à l'aide du typage que l'on écrit uniquement des programmes qui terminent. Le fait de considérer des termes typés ne nuit pas à la clarté du programme car il existe des mécanismes automatiques qui peuvent retrouver les informations de typage d'un programme non typé, et vérifier qu'elles sont correctes, c'est à dire que le programme qui en résulte est bien typé. Ces informations de type deviennent donc facultatives pour le programmeur. Tout comme les termes, la machinerie sous-jacente qui va exécuter (ou *réduire* selon la terminologie du λ-calcul) un λ-terme est également très simple. Elle consiste en une unique règle de réduction, appelée

la *β-réduction*, qui transforme un terme $(\lambda x.t)\ u$ en $t\{x/u\}$. Cette réduction est notée $(\lambda x.t)\ u \to_\beta t\{x/u\}$. La notation $t\{x/u\}$ représente l'opération qui substitue chaque occurrence de la variable x par le terme u.

Les substitutions explicites

Le λ-calcul en tant que langage pour exprimer les fonctions calculables est pleinement satisfaisant. En revanche, si l'on veut s'en servir pour faire un langage de programmation, différents problèmes apparaissent. Le principal, autour duquel va s'articuler toute cette thèse, est le fait que l'opération de *substitution* $t\{x/u\}$ n'a pas été défini dans le λ-calcul lui-même. Contrairement à la théorie, une implémentation du λ-calcul ne peut pas laisser certains détails de coté, c'est à dire qu'elle doit spécifier cette opération de substitution de manière exacte et précise. Cette définition n'étant pas canonique, cela peut donner lieu à des implémentations très différentes. De ce fait, il est plus difficile d'étudier de façon théorique la base commune aux différentes implémentations.

Même si la définition suivante de substitution est souvent admise :

$$\begin{array}{lcl}
(\lambda y.t)\{x/u\} & := & \lambda y.t\{x/u\} \\
(t\ v)\{x/u\} & := & t\{x/u\}\ v\{x/u\} \\
x\{x/u\} & := & u \\
y\{x/u\} & := & y
\end{array}$$

elle ne permet pas de parler du comportement qu'ont les environnements dans les langages de programmations, c'est à dire des listes de substitutions qui restent en attente d'évaluation.

C'est pour toutes ces raisons qu'il est intéressant de définir la substitution comme étant un citoyen de première classe, d'en spécifier les règles qui définissent son comportement, et d'en étudier les propriétés. On parlera alors de *substitution explicite*, contrairement à la *substitution implicite*, ou métasubstitution, que nous avons utilisé jusqu'alors.

La première incursion dans le domaine est l'article de De Bruijn [dB78] [1]. Même si la présentation est différente de celle connue actuellement, et que le mot "explicite" n'est pas présent, la base d'un calcul qui réduit de façon élémentaire est jetée. Cet article fût important pour le projet Automath [Bro], permettant d'exprimer de façon formelle des propriétés mathématiques afin de les prouver automatiquement. Il faudra attendre dix ans pour que l'on s'intéresse non seulement aux côtés pratiques, mais également aux aspects théoriques de ces calculs. Les articles [HL89] de Hardin et Lévy, et [Cur88] de Curien seront les précurseurs de l'article « Explicit Substitutions » de Abadi, Cardelli, Curien, et Lévy [ACCL90], considéré comme la référence du premier calcul avec substitutions explicites dans la littérature. Contrairement à ses ancêtres, il n'est plus question ici de s'intéresser uniquement à des versions simplifiées suffisant à décrire les langages de programmation, mais de prendre

[1] Une version avec des notations plus modernes est disponible [Les94a]

en compte le système de réécriture dans toute sa généralité. C'est cette façon de traiter le problème de la substitution qui va créer une dynamique "substitutions explicites" et donner lieu à des dizaines de calculs, à la quête de celui qui aura les meilleurs propriétés. La communauté sera d'autant plus motivée que quelques années plus tard, il apparaitra que le calcul proposé dans [ACCL90] possède un défaut majeur [Mel95] (voir propriété PSN, page 5).

Pour pouvoir illustrer l'ensemble des propriétés que l'on s'attend à trouver dans un calcul avec substitutions explicites, nous utilisons le langage $\lambda\mathtt{x}^-$ [BR95, Blo97] où la substitution explicite est représentée par le constructeur $[\_/\_]$:

$$\begin{array}{lll}
(\lambda x.t)u & \to_{\mathtt{B}} & t[x/u] \\
(\lambda y.t)[x/u] & \to_{\mathtt{x}^-} & \lambda y.t[x/u] \\
(t\ v)[x/u] & \to_{\mathtt{x}^-} & t[x/u]\ v[x/u] \\
x[x/u] & \to_{\mathtt{x}^-} & u \\
y[x/u] & \to_{\mathtt{x}^-} & y
\end{array}$$

Parmi les règles présentées on peut distinguer $\to_{\mathtt{B}}$ qui remplace la règle β du λ-calcul par une autre créant une substitution explicite identifiée par un nouvel opérateur du langage. Celui-ci n'est plus "méta", mais se situe au même niveau que l'application ou l'abstraction. Cette nouvelle sorte de substitution n'est plus effectuée de suite mais propagée par les règles $\to_{\mathtt{x}^-}$, qui forment le sous-calcul de propagation. Prises à part, les règles $\to_{\mathtt{x}^-}$ doivent terminer car elles ne créent pas de nouvelles substitutions à propager. On dira que ce sous-calcul a la propriété de normalisation forte. Ce n'est évidemment pas le cas de l'ensemble formé par $\lambda\mathtt{x}^-$ car comme il existe des programmes qui ne terminent pas, il existe également des termes du λ-calcul qui ne terminent pas, et donc des termes de $\lambda\mathtt{x}^-$. On désire en effet que le calcul avec substitutions explicites se comporte de la même façon que le λ-calcul, y compris lorsqu'il y a des boucles infinies.

On peut lister les différentes propriétés théoriques auxquelles on peut s'attendre pour un tel langage :

- **Correction (ou simulation)** : Si un terme du λ-calcul se réduit en un autre, alors on veut pouvoir simuler cette β-réduction dans le calcul avec substitutions explicites.

  Par exemple le terme $(\lambda x.xx)\ \lambda y.y$ se β-réduit en $\lambda y.y$ par les étapes $(\lambda x.xx)\ \lambda y.y \to_\beta (\lambda y.y)\ (\lambda y.y) \to_\beta \lambda y.y$. Dans le langage $\lambda\mathtt{x}^-$, en partant du même terme initial, on arrive bien au même résultat, en faisant évidemment plus d'étapes :

$$\begin{array}{ll}
(\lambda x.xx)\ \lambda y.y & \to_{\mathtt{B}} \\
(xx)[x/\lambda y.y] & \to_{\mathtt{x}^-} \\
x[x/\lambda y.y]\ x[x/\lambda y.y] & \to_{\mathtt{x}^-} \\
(\lambda y.y)\ x[x/\lambda y.y] & \to_{\mathtt{B}} \\
y[y/x[x/\lambda y.y]] & \to_{\mathtt{x}^-} \\
x[x/\lambda y.y] & \to_{\mathtt{x}^-} \\
\lambda y.y &
\end{array}$$

- **Confluence** : si au cours de l'évaluation du programme on poursuit dans deux directions différentes, alors les deux réductions doivent mener au même résultat in fine. Par exemple si dans le programme $(1+1)+(2+3)$ on évalue $2+(2+3)$ ou $(1+1)+(5)$, on est sûr d'arriver par la suite à un résultat commun, à savoir $2+5$. De la même façon, on avait plusieurs choix lors de la réduction que l'on a effectuée pour illustrer la β-simulation. On aurait pu faire le choix suivant :

$$\begin{array}{ll}
(\lambda x.x\ x)\ \lambda y.y & \to_{\mathtt{B}} \\
(x\ x)[x/\lambda y.y] & \to_{\mathtt{x}^-} \\
x[x/\lambda y.y]\ x[x/\lambda y.y] & \to_{\mathtt{x}^-} \\
(\lambda y.y)\ x[x/\lambda y.y] & \to_{\mathtt{x}^-} \\
(\lambda y.y)\ \lambda y.y & \to_{\mathtt{B}} \\
y[y/\lambda y.y] & \to_{\mathtt{x}^-} \\
\lambda y.y &
\end{array}$$

  Nous sommes bien arrivés au même résultat.

  Tous les langages que nous étudions dans cette thèse sont confluents, comme l'est d'ailleurs le λ-calcul.

  Les techniques pour prouver la confluence reposent principalement sur des jeux de simulation entre le calcul avec substitutions explicites et le λ-calcul, ainsi que sur la propriété de confluence du λ-calcul lui même [Chu41].

- **Confluence sur des termes ouverts** : également appelée métaconfluence, cette propriété s'intéresse à la confluence lorsque l'on a des termes dont on ne connait pas toute la structure, appelés *termes ouverts* ou *métatermes*. Par opposition, on parlera de *termes clos* pour les termes dont on connait la structure entière. Les termes ouverts peuvent être utilisés pour effectuer de l'unification d'ordre supérieur [DHKP96, Hue76] ou implémenter des métalangages [Nad02].

  Le concept de terme méta est le même que celui utilisé pour spécifier les règles de réduction où l'on écrit par exemple $(t\ v)[x/u] \to_{\mathtt{x}^-} t[x/u]\ v[x/u]$. Ici t, v et u sont des termes "variables" dans le sens où l'on ne connait pas leurs structures sous-jacente. De même que pour la confluence, pour qu'un calcul soit métaconfluent, il faut qu'il soit possible à partir d'une divergence dans la réduction de revenir à un terme commun. Le calcul $\lambda\mathtt{x}^-$ n'est pas métaconfluent comme le montre l'exemple suivant :

$$\begin{array}{ccccc}
t[x/u][y/v] & {}_{\mathtt{B}}\leftarrow & ((\lambda x.t)\ u)[y/v] & \to_{\mathtt{x}^-} & (\lambda x.t)[y/v]\ u[y/v] \\
? & & & & \to_{\mathtt{x}^-} \\
t[y/v][x/u[y/v]] & & {}_{\mathtt{B}}\leftarrow & & \lambda x.t[y/v]\ u[y/v]
\end{array}$$

  En effet entre les termes $t[x/u][y/v]$ et $t[y/v][x/u[y/v]]$ il n'existe pas de réduction possible si on ne connait pas la structure de t.

La métaconfluence implique non seulement la confluence sur les termes clos, mais indique également que les diagrammes peuvent être fermés de façon locale.

Pour montrer la métaconfluence, il n'est plus possible de se reposer sur la confluence du λ-calcul tel qu'on peut le faire pour montrer la confluence sur les termes clos. A la place, on va utiliser les techniques basées sur la réduction parallèle [Bar84].

- **Préservation de la normalisation forte (PSN)** : si un programme défini par un λ-terme termine, alors cela doit être également le cas lorsqu'on l'évalue avec les règles du calcul avec substitutions explicites. C'était le cas pour le terme $(\lambda x.xx)\ \lambda y.y$ lorsqu'on l'a évalué pour illustrer la β-simulation. Il faut noter qu'un langage avec substitutions explicites correct, c'est à dire qui peut simuler le λ-calcul, ne préserve pas forcément la normalisation forte. Si le λ-terme t se réduit en t' alors il existe un chemin pour simuler cette réduction dans le calcul avec substitutions explicites, mais il peut également exister un autre chemin qui mène sur une boucle infinie.

  Malgré son apparente simplicité, la PSN est l'une des propriétés les plus difficiles à garantir et à démontrer. De plus, contrairement à toute attente, elle n'est pas vraie pour $\lambda\sigma$, le langage pionnier, comme l'a montré Melliès avec son contre-exemple [Mel95].

  C'est finalement plusieurs années plus tard qu'apparaitront des calculs comme $\lambda\mathtt{x}$ [BR95, Blo97] ou $\lambda\upsilon$ [BBLRD96] qui sacrifieront de la métaconfluence pour obtenir PSN.

  Les techniques pour montrer PSN sont très variées. Elles peuvent être faciles à mettre en oeuvre pour des calculs simples en se contentant d'utiliser des RPO ($\lambda\mathtt{x}$, ...). On peut noter également la technique qui consiste à remonter l'historique de créations des substitutions en utilisant un argument de minimalité [BBLRD96]. Une façon également simple de montrer PSN est d'utiliser la propriété dite IE, c'est à dire que si un terme $t\{x/u\}$ termine, alors sa version explicite $t[x/u]$ termine également. Il reste ensuite à prouver IE, ce qui peut être fait simplement par induction dans certains cas [AK10] ou bien via un système complexe de calculs auxiliaires avec étiquettes [DG01, ABR00, Kes07]. Une autre façon de montrer la PSN est l'utilisation des techniques de simulation. Elles consistent à simuler un calcul pour lequel on veut montrer PSN dans un autre pour lequel on sait que PSN est vérifiée [Kes07].

- **Normalisation forte** : cette propriété stipule que tout programme typé termine. Pour considérer des termes avec substitutions explicites typés, il va falloir étendre le système de typage du λ-calcul pour gérer le cas de la substitution. Même si cette extension est souvent naturelle, les preuves se compliquent nettement. Ces dernières reposent (comme pour le λ-calcul) sur des techniques de réducibilité [Bar84], ou utilisent plus simplement la PSN.

Il existe également une variante, la normalisation faible. Elle indique qu'il existe, parmi toutes les réductions possibles à partir d'un terme, un chemin qui termine. Elle est suffisante pour les implémentations. En effet, toute implémentation est déterministe, c'est à dire qu'elle va choisir un unique chemin à suivre qui sera indiqué par une stratégie. Il suffit donc d'imposer une stratégie normalisante, qui choisit donc toujours le bon chemin, pour ne pas tomber dans une impasse.

On demande également, comme mentionné ci-dessus, à ce que les règles qui ne font que propager les substitutions explicites terminent.

- **Composition totale** : c'est une généralisation de la notion de simulation. Tout terme $t[x/u]$ du calcul avec substitutions explicites doit pouvoir se réduire sur $t\{x/u\}$. C'est une propriété théorique très forte, qui en implique facilement d'autres (simulation, confluence sur des termes ouverts, ...). Le calcul $\lambda\mathtt{x}^-$ ne possède pas cette propriété de composition totale car il n'a aucune règle permettant de composer deux compositions. Par exemple $x[x/y][y/z]$ ne se réduit pas sur $x[x/z]$. Il doit en effet se réduire d'abord sur $y[y/z]$ puis sur z, ce qui n'est pas le résultat escompté.

Réussir à obtenir l'ensemble des propriétés mentionnées ci-dessus pour un calcul avec substitutions explicites est un savant jeu d'équilibres. En effet, pour obtenir la composition totale ou la métaconfluence, il semble nécessaire d'ajouter une règle telle $t[x/u][y/v] \to t[y/v][x/u[y/v]]$ qui permet d'obtenir la composition de deux substitutions pour continuer ensuite la propagation. Malheureusement, une version naïve de composition telle que celle-ci va irrémédiablement conduire au contre-exemple de Melliès, qui repose sur le fait que le calcul permet de composer des substitutions lorsque la variable y n'apparaît dans u. De nombreuses restrictions permettent de rattraper PSN, comme interdire à une substitution de traverser une abstraction [LM99], ou spécifier une stratégie de réduction [GL98], mais presque toutes se font au détriment d'une ou plusieurs autres propriétés.

Restreindre la règle de composition $t[x/u][y/v] \to t[y/v][x/u[y/v]]$ à la condition que « y apparaisse dans u » permet déjà d'éviter le contre-exemple de Melliès. Il faut ajouter une équation qui permute les substitutions indépendantes pour pouvoir avoir réellement toutes les propriétés attendues (notamment la métaconfluence ou la composition totale) :

$$t[x/u][y/v] \equiv t[y/v][x/u]$$

Indépendant signifie que y n'apparaît pas dans u et que x n'apparaît pas dans v.

Mis à part la catégorisation des calculs ayant telle ou telle propriété, il existe les distinctions suivantes qui sont orthogonales aux précédentes propriétés (excepté la présence d'équations, qui est une question ouverte) :

- **Noms / Indices** : La présentation de variables avec des noms est celle que nous avons exclusivement utilisée jusqu'à présent, en se contentant de lettres. Elle présente cependant un gros défaut que l'on peut illustrer par l'exemple

suivant : si on applique naïvement la β-réduction, le terme $(\lambda y.\lambda x.y)\ x$ se réduit en $\lambda x.x$. Or le terme $(\lambda y.\lambda z.y)\ x$ qui devrait être équivalent à $(\lambda y.\lambda x.y)\ x$ se réduit en $\lambda z.x$, désignant un tout autre terme. Cela illustre le phénomène de capture de variables, qui peut être évité si l'on renomme toutes les variables liées (celles qui apparaissent sous la forme λx) d'un terme de telle façon à avoir des noms différents des variables libres (celles qui n'apparaissent pas sous la forme λx).

Utiliser cette solution dans la pratique, pour un langage de programmation, est complètement irréaliste. Afin d'avoir un langage un minimum efficace, il est nécessaire d'utiliser un système tout à fait différent, qui utilise des entiers, au lieu de lettres, comme les *indices de De Bruijn* [dB72] ou les moins connus niveaux de De Bruijn [dB72]. De cette façon on évite le renommage à chaque pas de réduction. La contrepartie, mais bien moins couteuse que les renommages, est la mise à jour de certains indices lorsqu'ils traversent des symboles lieurs. La manipulation de ces indices devient également plus fastidieuse.

Même si le calcul $\lambda\sigma$ a été présenté à la fois avec des noms et des indices, c'est en utilisant des indices que le principal des propriétés a été développé. On peut en effet remarquer que tout comme les substitutions explicites, l'usage des indices permet de se rapprocher de la réalité. La plupart des successeurs de $\lambda\sigma$, tels que $\lambda\sigma_w$ [CHL92], $\lambda\upsilon$ [BBLRD96, Les94b] ont également utilisé des indices.

C'est seulement par la suite que des calculs avec noms firent leur apparition. Le plus simple est sans doute $\lambda\mathtt{x}$ [BR95, Blo97, Ros92, Ros96] décrit par les règles suivantes :

$$
\begin{array}{lcll}
(\lambda x.t)\ u & \to & t[x/u] & \\
x[x/u] & \to & u & \\
t[x/u] & \to & t & x \text{ n'apparaît pas dans } t \\
(\lambda y.t)[x/u] & \to & \lambda y.t[x/u] & \\
(v\ w)[x/u] & \to & v[x/u]\ w[x/u] &
\end{array}
$$

Il existe des calculs ayant des indices [MRZ10] et d'autres ayant des noms [Kes09] avec toutes les bonnes propriétés listées ci-dessus. Les deux calculs cités en exemple sont d'ailleurs isomorphes [MRZ10], ce qui montre que le choix entre indices et noms n'est pas important en ce qui concerne les propriétés théoriques du calcul.

- **Réduction forte / faible** : La réduction forte est celle qui autorise les réductions peu importe l'endroit où elles prennent place dans le terme. La réduction faible, au contraire, interdit les réductions se trouvant en dessous d'un symbole λ. Cette restriction est suffisante pour implémenter des langages fonctionnels et permet des simplifications, notamment sur la mise à jour des indices. Elle est donc suffisante pour les langages de programmation, et de ce fait ces derniers utilisent systématiquement la réduction faible.

Dans [CHL92] une version faible de $\lambda\sigma$ a été présentée avec l'avantage d'être confluente sur les termes ouverts, contrairement à sa version forte. Il faut noter que la majorité des études théoriques sur les calculs avec substitutions explicites considèrent la réduction forte. D'un point de vue plus pratique, [FMS05] présente différents calculs avec des réductions faibles dans une optique de performance.

- **Présence d'équations** : Les équations permettent d'identifier des termes moralement égaux. L'équation la plus simple est sûrement celle qui permute des substitutions indépendantes : $t[x/u][y/v] \equiv t[y/v][x/u]$ (si x n'est pas dans v et y pas dans u). Elle a été introduite par [Kes07], se basant sur des idées de [KL05], se basant à son tour sur les réseaux de preuves de la logique linéaire [Gir87].

  Il a été conjecturé que λ`es` [Kes07] possède toutes les propriétés listées ci-dessus (toutes sont montrées sauf la métaconfluence). Ce fût finalement une simplification de ce calcul, λ`ex` [Kes09], également avec une équation, qui atteint le but de vérifier l'ensemble de ces propriétés.

  Le calcul λ`ex` comporte les règles et l'équation suivante :

$$
\begin{array}{lcll}
(\lambda x.t)\ u & \to & t[x/u] & \\
x[x/u] & \to & u & \\
t[x/u] & \to & t & \text{si } x \text{ n'est pas dans } t \\
(v\ w)[x/u] & \to & v[x/u]\ w[x/u] & \\
(\lambda y.t)[x/u] & \to & \lambda y.t[x/u] & \\
t[x/u][y/v] & \to & t[y/v][x/u[y/v]] & \text{si } y \text{ est dans } u \\
 & & & \\
t[x/u][y/v] & \equiv & t[y/v][x/v] & \text{si } y \text{ n'est pas dans } u\text{, et } x \text{ pas dans } v
\end{array}
$$

  Il est important de noter que malgré la différence avec λ`x` qui peut paraître minime (règle de composition et équation en plus), les techniques nécessaires à mettre en oeuvre pour prouver PSN pour λ`ex` sont sans commune mesure.

  Pour les calculs avec indices, le seul ayant toutes les propriétés possède également une équation [MRZ10].

  C'est toujours un challenge de comprendre s'il existe un calcul sans équations ayant toutes les propriétés listées auparavant.

- **Structure / Distance** : Pour la plupart des calculs dans la littérature, la propagation de la substitution explicite $t[x/u]$ dépend de la structure de t. Cependant, une autre sorte de système de propagation existe, basé sur la notion de *multiplicité* et d'*action à distance* [Mil07, dB87, Ned92, SP94, KLN05, Ó Conchúir06]. On parle de calculs structurels et de calculs à distance.

Par exemple, le calcul $\lambda\mathtt{j}$ [AK10] [2], basé sur le formalisme graphique de [AG09] définit son mécanisme de propagation de la façon suivante :

$$\begin{array}{lll} t[x/u] & \to t & \text{si } x \text{ n'est pas dans } t \\ t[x/u] & \to t\{x/u\} & \text{si } x \text{ est une fois dans } t \\ t[x/u] & \to t_{x\rightsquigarrow y}[x/u][y/u] & \text{si } x \text{ est au moins deux fois dans } t \\ t[x/u][y/v] & \equiv\ t[y/v][x/u] & \text{si les substitutions sont indépendantes} \end{array}$$

où $t_{x\rightsquigarrow y}$ est le *remplacement non-déterministe* d'au moins une occurrence de x par y. Malgré la présence de la substitution implicite, on peut tout de même parler de calcul avec substitutions explicites car si on implémente $\lambda\mathtt{j}$ avec des graphes, l'opération $t\{x/u\}$ est aussi atomique que pouvait l'être la règle $x[x/u] \to u$ dans d'autres calculs.

Le calcul $\lambda\mathtt{j}$ [AK10] est également intéressant pour sa règle de création des substitutions explicites :

$$(\lambda x.t)\mathtt{L}\ u \to_{\mathtt{B}} t[x/u]\mathtt{L}$$

où $\mathtt{L}$ est une liste de substitution. La réduction suivante est possible avec $\lambda\mathtt{j}$:

$$\begin{array}{ll} (\lambda x.\lambda y.x\ y)\ a\ b & \to \\ (\lambda y.x\ y)[x/a]\ b & \to \\ (x\ y)[y/b][x/a] & \to \\ (a\ y)[y/b] & \to \\ (a\ b) & \to \end{array}$$

Dans le chapitre 4, nous proposons d'abord une étude de la confluence pour un calcul générique avec substitutions explicites qui satisfait un ensemble d'axiomes. Parmi ces axiomes, la simulation du λ-calcul permet de tirer partie de la confluence du λ-calcul. Grâce à cette méthode, la confluence d'un calcul se résume à montrer la confluence et la normalisation du calcul de propagation des substitutions, ainsi qu'à la vérification de différents axiomes naturels pour un tel calcul.

Dans la suite du chapitre, nous montrons que le $\lambda\mathtt{j}$-calcul a la propriété de métaconfluence. Pour y parvenir, nous introduisons une caractérisation des termes auxquels on a appliqué autant que possible la règle qui duplique une substitution. Cela permet de pouvoir indiquer comment clore une divergence après un renommage non-déterministe. En effet, alors que pour le $\lambda\mathtt{j}$-calcul avec des termes clos il suffisait d'effectuer la composition totale pour revenir à un terme commun, il faut maintenant être plus subtil car la composition totale n'est plus applicable. Cette caractérisation d'un terme auquel on a appliqué autant que possible la règle de contraction est indépendante de toute notion de confluence, et serait donc réutilisable dans des travaux futurs. Après avoir isolé la difficulté du non-déterminisme, on mettra en oeuvre la technique de réduction parallèle.

[2] Le calcul a été introduit en tant que « The structural λ-calculus » mais nous le catégorisons en tant que calcul à distance. En effet, le sens de structure dans le titre est celui des règles structurelles de la logique, et non de structure d'un terme.

Les ressources explicites

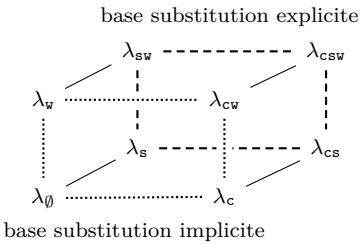
Il est possible d'aller plus loin dans l'explicitation des mécanismes du λ-calcul. On peut en effet voir, à la lumière de la logique linéaire [Gir87], la substitution explicite comme une substitution linéaire (c'est à dire qui lie une unique occurrence de variable) combinée à un effacement de variable (si la variable n'est pas là) et à une duplication (qui permet de dupliquer une variable si elle apparaît plusieurs fois). En se basant sur cette idée, le calcul λ`lxr` [KL07] possède non seulement substitution explicite, mais également *contraction* et *affaiblissement* (qui sont les constructions syntaxiques indiquant qu'il y aura une duplication et un effacement au moment de l'exécution). La contraction qui s'écrit $\mathcal{C}_x^{y|z}(t)$ représente le terme t où les variables y et z sont identifiées en la seule et même variable x. Il n'est donc pas possible d'écrire $x\ x$ dans ce calcul. De la même façon, il n'est plus possible d'écrire une fonction $\lambda x.t$ avec une variable x qui n'est pas utilisée dans t. A la place, on écrira $\lambda x.\mathcal{W}_x(t)$ où x n'apparaît donc pas dans t. Avec la présence de ces deux nouveaux opérateurs, il est nécessaire d'ajouter les règles qui vont les propager, et les faire interagir entre eux-mêmes et avec la substitution explicite. La sémantique associée à ce calcul fera descendre les contractions au plus bas dans les termes, et monter les affaiblissements au plus haut. Les interactions avec la substitution sont plus compliquées, notamment lorsqu'une substitution explicite rencontre une variable contractée ou affaiblie comme dans $\mathcal{C}_x^{y|z}(t)[x/u]$ ou $\mathcal{W}_x(t)[x/u]$.

Une fois ce formalisme défini, on peut se poser la question de savoir ce que serait un calcul avec seulement une partie de ses ressources explicites. Il existe des calculs avec affaiblissement et substitution comme [DG01] ou bien avec uniquement contraction et affaiblissement comme [vO01]. Mais de manière plus générale on peut dire que chacune de ces ressources pourrait être traité implicitement ou explicitement. Ceci donne les huit possibilités suivantes, où `c`, `s`, et `w` désignent respectivement `c`ontraction, `s`ubstitution, et `w`eakening (affaiblissement) :

Ressource `c`	Ressource `s`	Ressource `w`
Im	Im	Im
Ex	Im	Im
Im	Ex	Im
Im	Im	Ex
Ex	Ex	Im
Ex	Im	Ex
Im	Ex	Ex
Ex	Ex	Ex

Pour illustrer ce concept autrement, voici un prisme composé de deux *bases*, une où la substitution est implicite, et l'autre où elle est explicite. Par exemple λ`c` (qui correspond à la deuxième ligne du précédent tableau) est le langage où seulement la contraction est explicite, et λ`ws` (qui correspond à l'avant dernière ligne du précédent tableau) le langage où seulement l'affaiblissement et la substitution

sont explicites.



Cette présentation, avec les arrêtes dessinées, insiste sur le fait que l'on peut factoriser certaines propriétés et espérer pouvoir transférer certaines caractéristiques automatiquement d'un calcul à un autre par des jeux de simulations.

Dans le chapitre 5, nous considérons l'étude d'un prisme des ressources. Il faut noter qu'il existe une multitude de prismes, au moins un pour chaque calcul avec substitutions explicites existant. L'ajout des opérateurs de contraction et d'affaiblissement ainsi que la modularisation du prisme pour obtenir les bonnes propriétés listées précédemment est encore une fois un jeu d'équilibre extrêmement délicat. Certains calculs se prêtent plus ou moins bien à ce jeu de simulations dans un prisme. Dans l'étude que l'on va faire, le calcul avec le plus de ressources explicites est une variante de $\lambda\mathtt{lxr}$ avec une règle de duplication non-déterministe comme dans $\lambda\mathtt{j}$. Le calcul $\lambda_{\mathtt{s}}$ qui gère uniquement de manière explicite la substitution présentera lui aussi une duplication non-déterministe. La PSN de ce calcul sera étudiée séparément au chapitre 3.

Pour décrire le prisme lui-même, nous introduisons les termes et les réductions qui sont définies pour l'ensemble du prisme, et non pour chaque calcul. Chacun des huit calculs sera paramétré par un ensemble de *sortes* $\{\mathtt{c}, \mathtt{s}, \mathtt{w}\}$, représentant respectivement la contraction, la substitution, et l'affaiblissement. Si une sorte appartient à un calcul donné alors le traitement des opérations correspondantes sont rendues explicites, comme présenté dans le tableau introduit précédemment. La notion de base sera également importante, car certains théorèmes sont vraies dans une base mais pas dans l'autre. Par exemple, la composition totale aura uniquement du sens dans la base de la substitution explicite. Ensuite par un jeu de simulations permettant de transporter des propriétés d'un calcul à un autre, la confluence sera montrée grâce à celle du λ-calcul, ainsi que la PSN grâce à celle de $\lambda_{\mathtt{s}}$ montrée au chapitre 3. Après avoir introduit des types simples, nous montrons que les termes typés de tous ces langages sont fortement normalisant.

Pour montrer que $\lambda_{\mathtt{s}}$ bénéficie de PSN, nous utilisons une méthode adaptée de [Kes07]. Celle-ci nécessite de montrer que la normalisation d'un terme avec substitution implicite implique la normalisation du même terme avec substitution explicite. Pour y arriver, il va falloir utiliser un système de substitutions avec des *étiquettes* afin d'identifier les termes normalisant utilisés dans le corps des substitutions. Pour propager cette nouvelle sorte de substitution, un système de propagation sera ajouté, qui pourra se diviser en deux sous-systèmes. L'un s'occupera de la partie

interne des substitutions étiquetées, l'autre du reste. On pourra ainsi montrer que si $t\{x/u\}$ et u terminent dans le système original, alors la version explicite étiquetée termine également. Il faudra pour conclure s'occuper de l'autre sens, et montrer que si un terme avec substitution explicite étiquetée termine, alors sa version non étiquetée termine aussi.

Longueur des réductions

Comme nous l'avons vu auparavant, les longueurs de réductions pour aller d'un terme à un autre peuvent être différentes selon que l'on utilise tel ou tel calcul avec substitutions explicites.

Si on fixe un calcul possédant la normalisation forte, par exemple en se restreignant aux programmes typés, il peut être intéressant de connaître les longueurs maximales de réductions. On peut ainsi obtenir une garantie sur le temps d'exécution d'un programme, en supposant que chaque pas de réduction prenne un temps constant. Malheureusement, dans le cas général d'un calcul qui permet d'exprimer au moins les λ-termes typés, cette borne maximale est tellement grande qu'elle n'est pas utile dans la pratique. En effet, pour le λ-calcul, Beckmann a montré que les bornes maximales correspondent à des tours d'exponentielles dépendant de la taille du terme initial, mais surtout du type du programme [Bec01]. Pire encore, il a montré que ces bornes sont exactes, c'est à dire que l'on ne peut pas vraiment espérer mieux pour le cas général.

Nous allons étudier les bornes maximales pour les calculs avec substitutions explicites. Ces problèmes jamais étudiés auparavant permettraient de mieux connaitre l'écart pratique séparant le λ-calcul des calculs avec substitutions explicites. Cela pourrait également renseigner sur l'écart entre les implémentations et les calculs avec substitutions explicites. Ainsi, on pourrait mieux se rendre compte du niveau d'abstraction des calculs avec substitutions explicites.

Dans le chapitre 6, nous faisons une étude de la complexité pour un le calcul $\lambda\mathtt{x}^{-}$, et conjecturons des éléments de réponse à la question de la quantification de l'écart de complexité séparant le λ-calcul des λ-calculs avec substitution explicite. Pour y parvenir, nous effectuons des expériences pratiques sur la forme que prennent certains réductions. Grâce à un programme qui analyse la taille d'un terme au cours de sa réduction, nous remarquons des schémas qui se répètent au cours du temps, ce qui permet d'obtenir des informations sur des directions futures de recherche.

Certification

La vérification des programmes prend une place de plus en plus importante au fur et à mesure que les programmes eux-mêmes se diffusent partout dans la vie de tous les jours. Dans les domaines où des programmes défaillants pourraient mettre des vies en jeu, la simple chasse au bug ne suffit plus. Il faut s'assurer qu'il n'y a plus aucun bug non pas en les cherchant, mais en ayant la preuve mathématique que le

programme n'en contient aucun, c'est à dire qu'il est correct.

Ces preuves peuvent parler des programmes eux-mêmes, ou bien des propriétés qu'on en attend. Plutôt que de s'intéresser aux programmes eux-mêmes, on peut également considérer la certification des compilateurs, comme développé dans le projet CompCert [Com]. De la même manière, l'étude de la sémantique des langages de programmation et de leurs corrections apparaît comme un prérequis avant de s'intéresser aux programmes que l'on pourra écrire avec.

C'est à cet endroit que les substitutions explicites entrent en jeu car elles ont une place importante pour nombre de langages fonctionnels. Cela présente un grand intérêt car avec le grand nombre de règles que présentent certains calculs, il est relativement facile de faire des erreurs. De plus, cela offre l'avantage de se poser des questions sur certains détails qui eux, étaient restés implicites. Par exemple le remplacement non-déterministe n'a jamais été spécifié formellement pour $\lambda\mathtt{j}$ [AK10].

Pour parvenir à vérifier un programme ou une preuve, différents outils existent. Certains prouveurs fonctionnent de manière automatique, d'autres de manière interactive. Il est évident que l'on pourra prouver plus des propriétés plus complexes si on indique à l'ordinateur comment procéder. Les prouveurs automatiques comme Alt Ergo [CCK06] ou CVC3 [BT07] sont en général articulés autour de solveurs plus ou moins complexe du problème SAT. Les prouveurs interactifs possèdent des théories bien plus complexes que les prouveurs automatiques. On peut citer Coq [Coq] qui est basé sur le Calcul des Constructions inductives.

Il faut tout de même rappeler que avant de pouvoir prouver certaines propriétés, que cela soit automatiquement ou interactivement, il faut d'abord *formaliser* le langage que l'on désirer étudier. Cette formalisation va être plus ou moins simple selon que l'on étudie l'arithmétique, pour laquelle il n'y a souvent pas de formalisation à proprement parler car elle est la plupart du temps intégrée dans le prouveur, ou bien un langage avec des lieurs. En effet, un désavantage que possède nombre de prouveurs est l'inexistence de mécanisme permettant de s'occuper de la gestion des variables liées.

Si on veut formaliser un langage avec indices tel que $\lambda\sigma$ il n'y aura pas de problème car le mécanisme de gestion des lieurs est déjà complètement formalisé, mais si on s'intéresse à $\lambda\mathtt{x}$ alors il va falloir spécifier ses règles de réduction avec des indices, ou bien utiliser un mécanisme compliqué de renommage des variables pour éviter le phénomène de capture. De plus, il faudrait aussi pouvoir identifier des termes "jumeaux" tels que $\lambda x.x$ et $\lambda y.y$. Pour gérer ce problème, d'autres prouveurs comme Abella [Gac08] utilisent la notion de syntaxe d'ordre supérieure où ni noms, ni entiers ne sont utilisés. Une dernière possibilité est l'utilisation de la logique nominale [GP02] comme le propose Isabelle [Isa].

Dans le Chapitre 6, nous formalisons le calcul $\lambda\mathtt{j}$ en Coq en choisissant d'utiliser un mélange d'indices pour représenter les variables liées, ainsi que de noms pour les variables libres. Nous expliquons les mécanismes à mettre en œuvre, ainsi que l'intérêt de formaliser un tel calcul. Nous donnons une définition formelle du remplacement non-déterministe, et, après avoir expliquer les difficultés techniques liées à l'utilisation d'indices, nous donnons une preuve formelle de la propriété de

composition totale pour $\lambda\mathtt{j}$.

Résumé de la thèse et de ses contributions

- Chapitre 2 : nous définissons toutes les notions présentées informellement dans cette introduction qui servent tout au long de la thèse. Ainsi, nous revenons plus en détail sur les notions de type et de stratégie.
- Chapitre 3 : nous montrons que le langage $\lambda\mathtt{s}$, au cœur du chapitre 5, bénéficie de la propriété PSN. Ce résultat permettra au chapitre 5 de montrer PSN pour l'ensemble des calculs du prisme.
- Chapitre 4 : la question de la confluence est traitée de différentes manières. D'abord d'un point de vue axiomatique pour l'ensemble des calculs utilisant des noms décrits dans cette thèse. Dans un deuxième temps, nous nous intéressons à la métaconfluence du calcul $\lambda\mathtt{j}$ [Ren].
- Chapitre 5 : c'est là où est défini le prisme des ressources, la contribution la plus importante de la thèse. Les résultats de ce chapitre ont été publiés dans [KR09] puis révisés et étendus dans [KR11].
- Chapitre 6 : pour cette dernière partie, nous proposons une ouverture plus pratique. Dans un premier temps nous étudions la longueur des réductions de $\lambda\mathtt{x}^{-}$ et conjecturons des éléments de réponse. Dans un second temps nous formalisons $\lambda\mathtt{j}$ en Coq et montrons qu'il possède la propriété de composition totale.
- Chapitre 7 : nous finissons par une conclusion et une présentation des différentes lignes de travaux futurs.

Chapter 2

Preliminaries

Contents

In this Chapter we present some known technical material needed for the comprehension of the thesis. The prerequisites only consist of mathematical basis. In Section 2.1 we present Abstract Reduction Systems, which are used to express general notions related to reduction. In Section 2.2 we formally introduce the λ-calculus, and all its specific notions. In Section 2.3 we introduce explicit substitutions which are at the heart of the thesis. We present some notions concerning explicit substitutions in general, and some others only concerning calculi with explicit substitutions used later in the thesis. Finally, in Section 2.4, we show the relation between explicit substitutions and logic, through the Curry-Howard isomorphism.

2.1 Abstract Reduction Systems

We first state some general properties independent of the reduction system we are considering. All those notions will be applicable both for the λ-calculus and any λ-calculus with explicit substitutions. For more details about this section, consult the reference book for rewriting, Terese [Ter03].

Definition 2.1.1 (Abstraction reduction system (ARS))**.** *An abstract reduction system (ARS) is a set A (the objects) equipped with a binary relation $\mapsto_R$ (or $\mapsto$ if there is no ambiguity) over A. We write $t \mapsto u$ if $(t, u) \in \mapsto$. We say that t reduces*

to u. The relation $\mapsto_R^+$ is the transitive closure of $\mapsto_R$ and $\mapsto_R^*$ is the reflexive and transitive closure of $\mapsto_R$. Given two ARSs $(A, \mapsto_{R_1})$ and $(A, \mapsto_{R_2})$, the ARS $(A, \mapsto_{R_1 \cup R_2})$ is the ARS whose reduction relation is given by $\mapsto_{R_1} \cup \mapsto_{R_2}$.

Definition 2.1.2 (Sub-ARS)**.** *Let $(A, \mapsto_R)$ and $(B, \mapsto_{R'})$ be two ARSs. Then $(A, \mapsto_R)$ is a sub-ARS of $(B, \mapsto_{R'})$ if the following conditions are satisfied:*

- $A \subseteq B$;
- $\forall a, a' \in A,\ a \mapsto_R a' \iff a \mapsto_{R'} a'$;
- $\forall a \in A,\ \forall b \in B,\ a \mapsto_{R'} b \implies b \in A$.

Definition 2.1.3 (Reduction modulo)**.** *Let $(A, \mapsto_R)$ be an ARS with an associated equivalence relation $\equiv$ over A. Let t, t', u, u' be objects in A. In this thesis we use the following notations:*

- *$t \Rightarrow_R t'$ if $t \mapsto_R t'$ (this is an alternative notation for $t \mapsto_R t'$).*
- *$t \to_R t'$ if $t \equiv u \mapsto_R u' \equiv t'$*
- *$t \to_R^* t'$ is the transitive and reflexive closure of $\to_R$. As we consider equivalence classes, the reflexive case is $t \equiv t'$, and not $t = t'$.*

Notice that $(A, \to_R)$ is also an ARS.

Latter in the thesis, the "normal" notion of reduction for a calculus will be written $\to_R$ i.e. we almost always want to consider equivalences in the reduction if there are ones. Even for a calculus without equivalences, we will also use the notation $\to_R$ (this is the case where $\equiv$ is simply the syntactical identity). The notation $t \Rightarrow_R t'$ will thus only be used to denote that in a calculus with an equivalence we are not considering the equivalence.

Definition 2.1.4 (Normal form)**.** *Given an ARS $(A, \mapsto_R)$, we say that an object $t \in A$ is in $\to_R$-normal form (resp. $\Rightarrow_R$-normal form), written $t \in \mathcal{NF}_{\to_R}$ (resp. $t \in \mathcal{NF}_{\Rightarrow_R}$) or $t \in \mathcal{NF}_R$ if no ambiguity arises, if there is no t' such that $t \to_R t'$ (resp. $t \Rightarrow_R t'$). If t reduces to v and this latter one is in normal form, then v is said to be a normal form of t.*

Definition 2.1.5 (Strongly normalizing (SN))**.** *Given an ARS $(A, \mapsto_R)$, if there is no infinite $\to_R$-reduction (resp. $\Rightarrow_R$) starting at $t \in A$, then we say that t is $\to_R$-strongly normalising (resp. $\Rightarrow_R$-strongly normalising), written $t \in \mathcal{SN}_{\to_R}$ (resp. $t \in \mathcal{SN}_{\Rightarrow_R}$), or simply $t \in \mathcal{SN}_R$ if no ambiguity arises. An ARS is $\to_R$-strongly normalising (resp. $\Rightarrow_R$-strongly normalising) iff all its objects are.*

Theorem 2.1.6. *Let $(A, \mapsto_{R_1})$ and $(A, \mapsto_{R_2})$ be two ARSs. Let $(B, \mapsto_R)$ be another ARS and g a relation s.t. $g \subseteq A \times B$. Suppose that for all a, a', b, b'*

(P1) *$a\ g\ b$ & $a \to_{R_1} a'$ imply $\exists b'$ s.t. $a'\ g\ b'$ & $b \to_R^* b'$.*

(P2) *$a\ g\ b$ & $a \to_{R_2} a'$ imply $\exists b'$ s.t. $b\ g\ b'$ & $b \to_R^+ b'$.*

(P3) *The relation* $\rightarrow_{R_1}$ *is SN.*

then, $a\ g\ b$ & $b \in \mathcal{S}N_R$ *imply* $a \in \mathcal{S}N_{R_1 \cup R_2}$.

Proof. A proof by contradiction can be easily done as follows. Suppose $a \notin \mathcal{S}N_{(R_1 \cup R_2)}$. Then, there is an infinite $(R_1 \cup R_2)$-reduction sequence starting at a, and since R_1 is a well-founded relation by P3, this reduction sequence has necessarily the form

$$a \rightarrow^*_{R_1} a_1 \rightarrow^+_{R_2} a_2 \rightarrow^*_{R_1} a_3 \rightarrow^+_{R_2} \ldots \infty$$

and can be projected by P1 and P2 into an infinite $\rightarrow_R$-reduction sequence as follows:

$$\begin{array}{cccccccc} a & \rightarrow^*_{R_1} & a_1 & \rightarrow^+_{R_2} & a_2 & \rightarrow^*_{R_1} & \ldots & \infty \\ b & \rightarrow^*_A & b_1 & \rightarrow^+_R & b_2 & \rightarrow^*_R & \ldots & \infty \end{array}$$

We thus get a contradiction with the fact the $b \in \mathcal{S}N_R$. □

Definition 2.1.7 (Confluence)**.** *Given an ARS* $(A, \mapsto R)$ *with an equivalence relation* $\equiv$ *over objects. We say that:*

1. $\rightarrow_R$ *is confluent if* ${}^*_R\leftarrow \cdot \rightarrow^*_R \subseteq \rightarrow^*_R \cdot {}^*_R\leftarrow$
2. $\Rightarrow_R$ *is confluent modulo* $\equiv$ *if* ${}^*_R\Leftarrow \cdot \equiv \cdot \Rightarrow_R \subseteq \Rightarrow^*_R \cdot \equiv \cdot {}^*_R\Leftarrow$
3. $\Rightarrow_R$ *is Church-Rosser (CR) modulo* $\equiv$ *if* $\leftrightarrow^*_R \subseteq \Rightarrow^*_R \cdot \equiv \cdot {}^*_R\Leftarrow$ *where* $\leftrightarrow^*_R := (\Rightarrow_R \cup {}_R\Leftarrow \cup \equiv)^*$

Lemma 2.1.8. *CR modulo* $\equiv$ *is the strongest property i.e.* $3 \Rightarrow 2$ *and* $3 \Rightarrow 1$ *but* $1 \not\Rightarrow 3$ *and* $2 \not\Rightarrow 3$.

Proof. Cf. [Ter03]. □

Definition 2.1.9 (Convergence)**.** *An ARS* $(A, \mapsto R)$ *is* $\rightsquigarrow$*-convergent (*$\rightsquigarrow$ *being* $\rightarrow_R$ *or* $\Rightarrow_R$*) if it is* $\rightsquigarrow$*-confluent and* $\rightsquigarrow$*-strongly normalizing. Thus any object has a unique* $\rightsquigarrow$*-normal form in a* $\rightsquigarrow$*-convergent ARS, written* $\downarrow_R (t)$ *(if* $\rightsquigarrow = \rightarrow_R$*) or* $\Downarrow_R (t)$ *(if* $\rightsquigarrow = \Rightarrow_R$*) or simply* $\downarrow(t)$ *if no ambiguity arises.*

Definition 2.1.10 (Length of a reduction)**.** *The length of a reduction starting from an object* a *to an object* a' *is its number of steps. The equivalence steps are not considered. In the case of a convergent ARS, the longest length from an object* a *to its unique normal form is denoted* $\eta_{\rightarrow_R}(a)$ *or* $\eta_{\Rightarrow_R}(a)$*. The notation* $\eta(a)$ *will be used if no ambiguity arises.*

Definition 2.1.11 (Strategy)**.** *A strategy for an ARS* $(A, \mapsto_R)$ *is a sub-ARS having the same objects and the same normal forms. Furthermore, for every object* t*, there is only one* u *s.t.* t *reduces to* u*. The strategy thus defines a function over objects.*

Definition 2.1.12 (One-step strategy)**.** *A strategy* $(A, \mapsto_S)$ *of an ARS* $(A, \mapsto_R)$ *is one-step if* T *reduces to* T' *in one step by* $\rightarrow_S$ *(resp.* $\Rightarrow_S$*) then* T *also reduces in one step to* T' *by* $\rightarrow_R$ *(resp.* $\Rightarrow_R$*).*

Definition 2.1.13 (Many-step strategy)**.** *A strategy* $(A, \mapsto_S)$ *of an ARS* $(A, \mapsto R)$ *is many-step if* T *reduces to* T' *in one step by* $\to_S$ *(resp.* $\Rightarrow_S$*) then* T *reduces in several steps to* T' *by* $\to_R$ *(resp.* $\Rightarrow_R$*).*

Definition 2.1.14 (Effective stragtegy)**.** *A strategy is effective if it is computable.*

Definition 2.1.15 (Perpetual strategy)**.** *A strategy* $\rightsquigarrow$ *of an ARS* $(A, \mapsto_R)$ *is perpetual if* $\forall a \notin \mathcal{S}N_{\to_R}$ *(resp.* $\forall a \notin \mathcal{S}N_{\Rightarrow_R}$*),* $a \rightsquigarrow b \implies b \notin \mathcal{S}N_{\to_R}$ *(resp.* $a \rightsquigarrow b \implies b \notin \mathcal{S}N_{\Rightarrow_R}$*).*

Definition 2.1.16 (Maximal strategy)**.** *A strategy* $(A, \to_{max})$ *of an ARS* $(A, \mapsto_R)$ *is* $\to_R$*-maximal (resp.* $\Rightarrow_R$*-maximal) if* $\forall a \notin \mathcal{NF}_{\to_R}$ *(resp.* $\forall a \notin \mathcal{NF}_{\Rightarrow}$*),* $a \to_{max} b \implies \eta_{\to_R}(b) < \eta_{\to_R}(a)$ *(resp.* $a \to_{max} b \implies \eta_{\Rightarrow_R}(b) < \eta_{\Rightarrow_R}(a)$*).*

Lemma 2.1.17. *If a strategy is maximal, then it is perpetual. The converse is not true*

Proof. Straightforward. □

2.2 Lambda Calculus

2.2.1 Basis

The λ-calculus can be seen as an ARS (Δ, β) where Δ is given by Definition 2.2.1 and β by Definition 2.2.12.

Definition 2.2.1 (λ-term)**.** *The set* Δ *of* λ*-terms is defined by the following grammar:*

$$t, u ::= x \mid \lambda x.t \mid t\ u$$

where x *belongs to an infinite set of variables. The term* $t\ u$ *is called an application and* $\lambda x.t$ *an abstraction. When it is clear from the context, we will write term instead of* λ*-term.*

Definition 2.2.2 (Tree representation, depth)**.** *The inductive definition of* λ*-terms allow us to see them as trees. The application* $t\ u$ *is a node "application" with the tree representation of* t *as left son, and the tree representation of* u *as a right son. The abstraction* $\lambda x.t$ *is a node "abstraction" with a unique son consisting of the tree representation of* t*. A variable is a unique node.*

A subterm u *of* t *is at depth* n *in the tree representation of* t *if one has to cross* n *edges from the root of the tree to the tree representation of* u*.*

Definition 2.2.3 (Height)**.** *The height* $h(t)$ *of a term* t *is defined by* $h(x) := 0$*,* $h(\lambda x.t) := 1 + h(t)$*,* $h(u\ v) := 1 + max(h(u), h(v))$*.*

Notation 2.2.4. *The abstraction takes precedence over application that is with* $\lambda x.t\ u$ *we mean* $\lambda x.(t\ u)$*. The term* $((...((uv_1)v_2)...)v_n)$ *will be abbreviated as* $uv_1...v_n$*.*

Definition 2.2.5 (Free variables, Bound variables, Closed term)**.** *The set* $\mathtt{fv}(t)$ *of the free variables of a term* t *is defined by induction as:*

$$\begin{array}{lcl} \mathtt{fv}(x) & := & x \\ \mathtt{fv}(\lambda x.u) & := & \mathtt{fv}(u) \setminus \{x\} \\ \mathtt{fv}(u\ v) & := & \mathtt{fv}(u) \cup \mathtt{fv}(v) \end{array}$$

The set $\mathtt{bv}(t)$ *of bound variables of a term* t *is defined by induction as:*

$$\begin{array}{lcl} \mathtt{bv}(x) & := & \emptyset \\ \mathtt{bv}(\lambda x.u) & := & \mathtt{bv}(u) \cup \{x\} \\ \mathtt{bv}(u\ v) & := & \mathtt{bv}(u) \cup \mathtt{bv}(v) \end{array}$$

A term t *is closed if* $\mathtt{fv}(t) = \emptyset$*.*

Definition 2.2.6 (Number of occurences)**.** *The number of free occurrences of the variable* x *in a term* t*, written* $|t|_x$*, is defined by induction on* t*:*

$$\begin{array}{lcl} |x|_x & := & 1 \\ |y|_x & := & 0 \\ |\lambda y.t|_x & := & |t|_x \\ |v\ w|_x & := & |v|_x + |w|_x \end{array}$$

Definition 2.2.7 (Position)**.** *A position in a term* t *describes a subterm in* t *by means of a finite sequence (eventually empty, written* ε*) of* 0 *and* 1*. The subterm of* t *at position* p *is given by* $pos(p, t)$ *which is defined by induction on terms:*

$$\begin{array}{lcl} pos(\varepsilon, t) & := & t \\ pos(0 \cdot p, u\ v) & := & pos(p, u) \\ pos(1 \cdot p, u\ v) & := & pos(p, v) \\ pos(0 \cdot p, \lambda x.u) & := & pos(p, u) \end{array}$$

For instance, the subterm of $y\ (\lambda x.xz)$ *at position* 101 *is* z*.*

Definition 2.2.8 (Barendregt's convention, α-conversion)**.** *In all the thesis, we will intensively use the Barendregt's convention[Bar84], that is, free and bound variables will have different names. Furthermore, different bound variables will also have different names. The congruence on terms generated by the renaming of bound variables is the* α*-conversion or* α*-equivalence. We write* $t =_\alpha u$ *if* u *can be obtained from* t *by a renaming of bound variables.*

Example 2.2.9. *The term* $\lambda x.x\ y$ *is* α*-equivalent to* $\lambda z.z\ y$ *but not to* $\lambda x.x\ z$*.*

In all the thesis, the terms will always be considered modulo α-conversion

Definition 2.2.10 (Implicit substitution)**.** *The implicit substitution of x by u in t, written $t\{x/u\}$, is defined by induction on t, following the Barendregt's convention:*

$$\begin{array}{lcll} x\{x/u\} & := & u & \\ y\{x/u\} & := & u & \text{with } x \neq y \\ (\lambda y.v)\{x/u\} & := & \lambda y.v\{x/u\} & \text{with } x \neq y \\ (v\ w)\{x/u\} & := & v\{x/u\}\ w\{x/u\} & \end{array}$$

Example 2.2.11. *For instance $(\lambda y.x\ y)\{x/y\}$ is equal to $\lambda z.y\ z$ and not $\lambda z.z\ z$ which would result in the capture of a free variable.*

Definition 2.2.12 (β-redex)**.** *In a term t any subterm of the form $(\lambda x.u)\ v$ is called a β-redex (or redex if there is no ambiguity). Notice that redexes can be nested.*

Definition 2.2.13 (β-reduction)**.** *The unique rule of reduction of the λ-calculus is the β-rule, which is the contextual closure of the root rule contracting β-redexes:*

$$(\lambda x.t)\ u \mapsto_\beta t\{x/u\}$$

Example 2.2.14. *The following β-reductions are possible:*

- $(\lambda x.xz)(\lambda y.y) \to_\beta (\lambda y.y)z \to_\beta z$
- $(\lambda x.xx)(\lambda y.y) \to_\beta (\lambda y.y)\ (\lambda z.z) \to_\beta \lambda z.z$
- $(\lambda x.x\ x)\ (\lambda y.y\ y) \to_\beta (\lambda y.y\ y)(\lambda z.z\ z) \to_\beta (\lambda z.z\ z)(\lambda z'.z'\ z') \to_\beta ...$

Notice that we have used the α-equivalence in the second and third example. Remark also that the last reduction sequence is non-terminating.

2.2.2 Lambda Calculus with De Bruijn Indexes

To avoid the capture of free variables, the implicit substitution operation has to be performed using α-conversion. Substitution is an operation which has a huge cost since for every β-step, a renaming has to be done, implying to go trough all the term. Needless to say that there is no way that this approach can be used in practice.

The solution [dB72], proposed by De Bruijn, is to drop names, and instead use indexes. Those indexes will represent the number of symbols λ between a variable and the binder. We will however keep names for free variables even if they can themselves be turned into indexes pointing outside the term.

Example 2.2.15. *The λ-term $\lambda z.(\lambda y.y\ (\lambda x.x))\ (\lambda x.z\ x)$ is equivalent to $\lambda(\lambda 1(\lambda 1))\ (\lambda 21)$.*

The β-reduction for terms with indexes is more complex since we have to perform the replacement and maintain the coherence of the indexes after a substitution.

Let $t\langle n/u\rangle$ be the replacement function of indexes by De Bruijn terms. The β-rule is now $(\lambda.u)\ v \to_{\beta_{db}} u\langle 1/v\rangle$. The index 1 has to be replaced in u by v (if

we do not consider abstractions). Under λ, it is not 1 anymore which has to be replaced but 1 plus the number of abstractions traversed that is: $(u1u2)\langle n/v\rangle = u1\langle n/v\rangle\ u2\langle n/v\rangle$, $(\lambda.u)\langle n/v\rangle = \lambda.(u\langle n+1/v\rangle)$. For the case $m\langle n/u\rangle$, it is more complex. If $m < n$, then we simply have $m\langle n/u\rangle = m$. However, if $m > n$, that means that the index m referred to an abstraction above the head one in $(\lambda.u)\ v$. During the β-reduction, as a symbol λ is removed, we thus have $m\langle n/v\rangle = m-1$. We can thus summarise the replacement function:

$$\begin{array}{lcl}
x\langle n/v\rangle & := & x \\
m\langle n/v\rangle & := & \begin{cases} m & \text{if } m < n \\ \mathtt{U}_0^n(v) & \text{if } m = n \\ m-1 & \text{if } m > n \end{cases} \\
(u1u2)\langle n/v\rangle & := & u1\langle n/v\rangle\ u2\langle n/v\rangle \\
(\lambda.u)\langle n/v\rangle & := & \lambda.(u\langle n+1/v\rangle)
\end{array}$$

where $\mathtt{U}_0^n(v)$ is the update of the bound variables in v. The idea (in a first time we do not consider abstractions in v) is to increase indexes in v by $n-1$ (the n abstractions above the index, minus 1 since there is an abstraction which disappear during the reduction). Of course, it is not sufficient since we have to take into account abstractions in v, which requires to generalize $\mathtt{U}_0^n(v)$ into $\mathtt{U}_k^n(v)$, where k is the number of abstractions traversed in v. The update function is thus defined as:

$$\begin{array}{lcl}
\mathtt{U}_k^n(x) & := & x \\
\mathtt{U}_k^n(u1\ u2) & := & \mathtt{U}_k^n(u1)\ \mathtt{U}_k^n(u2) \\
\mathtt{U}_k^n(\lambda.u) & := & \lambda.\mathtt{U}_{k+1}^n(u) \\
\mathtt{U}_k^n(p) & := & \begin{cases} p+n-1 & \text{if } p > k \\ p & \text{otherwise} \end{cases}
\end{array}$$

As it is quite cumbersome to reason with indexes, we will focus on the λ-calculus with names, keeping in mind that there exists a translation from each class of α-equivalent terms to a unique representative with indexes. The main theoretical difficulties of the λ-calculus properties do not come from the use of indexes, so we are not avoiding pitfalls.

2.2.3 Types

As we have seen in Example 2.2.14, a reduction from a term can be infinite. The first characterisation of a subset of strongly normalizing terms uses types [Chu40].

Definition 2.2.16 (Simple Types)**.** *Simple types $(T, U, ...)$ are built over a countable set of atomic symbols $(\iota, \gamma, ...)$ and the type constructor $\rightarrow$. A type with only an atomic symbol is a ground type.*

For instance, $\iota \rightarrow (\gamma \rightarrow \iota)$ is a type.

Definition 2.2.17 (Level)**.** *A ground type ι has level $lv(\iota) := 0$ and $lv(T \rightarrow U) := max(lv(T)+1, lv(U))$. The level of a term is the level of its type.*

Definition 2.2.18 (Degree)**.** *The degree of a term t, $g(t)$, is the maximum of the levels of subterms of t.*

Definition 2.2.19 (Environment)**.** *An environment is a finite set of pairs of the form $x : T$. If $\Gamma = \{x_1 : T_1, ..., x_n : T_n\}$ is an environment then the domain of Γ is $\mathtt{dom}(\Gamma) = \{x_1, ..., x_n\}$. The renaming of an environment is the renaming of its domain.*

Definition 2.2.20 (Compatible and Disjoint environments)**.** *Two environments Γ and Δ are said to be compatible if $x : T \in \Gamma$ and $x : U \in \Delta$ imply $T = U$. Two environments Γ and Δ are said to be disjoint if there is no common variable in their environments.*

Definition 2.2.21 (Compatible and Disjoint union)**.** *Compatible union (resp. disjoint union) is defined to be the union of compatible (resp. disjoint) environments. the domain of Γ is $\mathtt{dom}(\Gamma) = \{x_1, ..., x_n\}$.*

Definition 2.2.22 (Typing judgements)**.** *Typing judgements have the form $\Gamma \vdash t : T$ for t a term, T a type and Γ an environment.*

Definition 2.2.23 (Typing rules, Well-typed term)**.** *A term t is well-typed (or simply typed) if there is a type T and an environment Γ s.t. the typing judgement $\Gamma \vdash t : T$ is derivable with the following rules:*

$$\frac{}{\Gamma, x : T \vdash x : T}\ var$$

$$\frac{\Gamma, x : T \vdash t : U}{\Gamma \vdash \lambda x.t : T \to U}\ abs \qquad \frac{\Gamma \vdash t : T \to U \quad \Gamma \vdash u : T}{\Gamma \vdash t\ u : U}\ app$$

Theorem 2.2.24 (Strong normalisation)**.** *Every well-typed term is strongly normalizing.*

Proof. A lot of proofs exist in the literature. A classical one can be found in [Bar92] while a short one can be found in [Dav]. □

Notice that the converse is not true. Indeed, the term $\lambda x.x\ x$ is β-strongly normalisable (and even in normal form) but it is not typable by means of the system of Definition 2.2.23. However, *intersection types* are able to completely capture β-strongly normalising terms, i.e. a λ-term t is typable in the intersection typing system of [CDCV81] iff t is β-strongly normalising.

2.2.4 Strategies

For a given λ-term, there can be several redexes in it, and thus, several ways to reduce it. Thus, β-reduction is non-deterministic. A specific path to follow can be given through a strategy which will select the redex to be contracted. This

can be useful for both theoretical and practical reasons. Perpetual and maximal strategies, aka the bad and the worst strategies, can be of great help to prove different theoretical properties. We will give one of each for the λ-calculus. A survey of both strategies can be found in [vRSSX99] and an elegant method to prove that a strategy is maximal can be found in [vO07]. From the practical point of view, it is a necessity to specify a strategy when implementing the λ-calculus because we do not want to choose among redexes. Even if the λ-calculus is confluent, we want to know the path followed during the reduction, making the debugging easier.

The importance of strategies can be summarised in the following example:

Example 2.2.25. *The term $(\lambda x.y)\ ((\lambda z.z\ z)\ (\lambda w.w\ w))$ has two redexes. If we choose to reduce the leftmost one, then the term reduces to y. If we always choose to reduce the rightmost, then it keeps reducing forever.*

Maximal strategies They can be very useful for theoretical purposes. Indeed they can help to have upper bounds on λ-reductions [Bec01, dM03].

Definition 2.2.26 ([Bar84])**.** *The strategy $F^{\beta}_{\infty}(\_)$ on a λ-term t s.t. $t \notin \mathcal{NF}_{\lambda}$ is defined as follows where $C[\![\_]\!]$ is the context where appears the leftmost redex:*

$$F^{\beta}_{\infty}(C[\![(\lambda x.t)\ u]\!]) \quad := \quad \begin{cases} C[\![(\lambda x.t)\ F^{\beta}_{\infty}(u)]\!] & \text{if } x \notin \mathtt{fv}(t) \text{ and } u \notin \mathcal{NF} \\ C[\![t\{x/u\}]\!] & \text{otherwise} \end{cases}$$

Lemma 2.2.27 ([dV87])**.** *The strategy $F^{\beta}_{\infty}(\_)$ is effective and maximal.*

Example 2.2.28. *The term $t = (\lambda x.x\ x)\ ((\lambda y.y)\ z)$ reduces using $F^{\beta}_{\infty}(\_)$ in the following way: $(\lambda x.x\ x)\ ((\lambda y.y)\ z) \to_{\beta} ((\lambda y.y)\ z)\ ((\lambda y.y)\ z) \to_{\beta} z\ ((\lambda y.y)\ z) \to_{\beta} z\ z$. If we had reduced the rightmost redex at the first step this would have given $(\lambda x.x\ x)\ ((\lambda y.y)\ z) \to_{\beta} (\lambda x.x\ x)\ z \to_{\beta} z\ z$ and thus lead to a shorter reduction.*

The intuition behind the maximal strategy is to let redexes be duplicated. As the leftmost redex cannot be duplicated or erased (as it not an argument of another redex), it is the good choice to start with.

Perpetual strategies They can be useful to show different properties such as for example finiteness of developments [dV85]. By Lemma 2.1.17, the strategy $F^{\beta}_{\infty}(\_)$ is also perpetual. However, as we have seen in the case of ARSs, there are perpetual strategies which are not maximal. We illustrate this fact with the following definition:

Definition 2.2.29 ([BK82])**.** *The strategy $F_1(\_)$ on a λ-term t s.t. $t \notin \mathcal{NF}_{\lambda}$ is defined as follows where $C[\![\_]\!]$ is the context where appears the leftmost redex:*

$$F_1(C[\![(\lambda x.t)\ u]\!]) \quad := \quad \begin{cases} C[\![(\lambda x.t)\ F_1(u)]\!] & \text{if } u \notin \mathcal{SN}_{\lambda} \\ C[\![t\{x/u\}]\!] & \text{otherwise} \end{cases}$$

Notice that the strategy is not effective, since the predicate $\in \mathcal{SN}_{\lambda}$ is not computable.

The strategy $F_1(\_)$ is perpetual but not maximal. Indeed, consider the term t defined in Example 2.2.28, and notice that the reduction followed is the second one, with a step less than the maximal reduction.

2.3 λ-calculi with Explicit Substitutions

Almost all the calculi with explicit substitutions defined in this thesis will be built over the same grammar:

Definition 2.3.1. *The set of terms with explicit substitutions is built by extending the grammar of λ-terms with an explicit substitution operator:*

$$t, u ::= x \mid \lambda x.t \mid t\ u \mid t[x/u]$$

Notation 2.3.2. *The term $t[x_1/v_1]\dots[x_n/v_n]$ is sometimes abbreviated as $t[\overline{x}/\overline{v}]$ when n is clear from the context, or simply $t\mathtt{L}$.*

All the typing systems of calculi with explicit substitutions will be constructed in the same way:

Definition 2.3.3 (Simple explicit typing system)**.** *The typing system of the λ-calculus is extended with the following rule:*

$$\frac{\Gamma, x : B \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash t[x/u] : A}$$

Definition 2.3.4. *All the notions defined for the λ-calculus are easily extended to handle the case of the new explicit substitution constructor:*

- *Height:* $h(t[x/u]) := 1 + max(h(t), h(u))$
- *Free variables:* $\mathtt{fv}(t[x/u] := (\mathtt{fv}(t) \setminus \{x\}) \cup \mathtt{fv}(u)$
- *Bound variables:* $\mathtt{bv}(t[x/u] := \mathtt{bv}(t) \cup \mathtt{bv}(u) \cup \{x\}$
- *Number of occurrences:* $|t[y/u]|_x := |t|_x + |u|_x$
- *Position:* $pos(0 \cdot p, t[x/u]) := pos(p, t)$, $pos(1 \cdot p, t[x/u]) := pos(p, u)$
- *Implicit substitution:* $t[y/v]\{x/u\} := t\{x/u\}[y/v\{x/u\}]$

To ease the reading, we will use the following convention for notations:

Convention 2.3.5. *Let $\lambda\mathtt{p}$ be a calculus with explicit substitutions. Then its rule creating expliciting substitutions will be called $\to_B$, the set of rules propagating explicit substitutions $\to_{\mathtt{p}}$, and the whole reduction system $\to_{\lambda\mathtt{p}}$. The terms of the calculus $\lambda\mathtt{p}$ will be called $\mathtt{p}$-terms.*

If the calculus has an equation $\equiv$, then any reduction relation $\to_{\mathtt{z}}$ ($z \in \{B, \mathtt{p}, \lambda\mathtt{p}\}$) will be considered modulo $\equiv$. To consider the reduction alone (not modulo), we will use the notation $\Rightarrow_{\mathtt{z}}$.

We present different calculi which we will work with, or evoke further in the thesis. Most of them have been already informally described in the introduction.

2.3.1 The Calculi $\lambda\mathtt{x}$ and $\lambda\mathtt{x}^-$

The $\lambda\mathtt{x}^-$ [BR95, Blo97] will be used in Chapter 6 for a complexity study. The $\lambda\mathtt{x}$-calculus is often evoked all along the thesis.

Definition 2.3.6 ($\lambda\mathtt{x}$ and $\lambda\mathtt{x}^-$)**.** *The* $\mathtt{x}$*-terms are those of Definition 2.3.1 and both calculi share the rules:*

$$\begin{array}{lll} (\lambda x.t)u & \to_B & t[x/u] \\ (\lambda y.t)[x/u] & \to_{SLam} & \lambda y.t[x/u] \\ (t\ v)[x/u] & \to_{SApp} & t[x/u]\ v[x/u] \\ x[x/u] & \to_{SVar} & u \end{array}$$

The only difference comes from the rule of garbage collection. The $\lambda\mathtt{x}^-$*-calculus has the simple rule:*

$$x[y/u] \to_{SGc^-} x$$

whereas $\lambda\mathtt{x}$*-calculus has a more general version:*

$$t[y/u] \to_{SGc} t \text{ if } y \notin \mathtt{fv}(t)$$

The reduction relation $\to_{\mathtt{x}}$ *(resp.* $\to_{\mathtt{x}^-}$*) is generated by the rules* $\to_{SLam}, \to_{SApp}, \to_{SVar}$*, and* $\to_{SGc}$ *(resp.* $\to_{SGc^-}$*). The reduction relation* $\to_{\lambda\mathtt{x}}$ *(resp.* $\to_{\lambda\mathtt{x}^-}$*) is generated by* $\to_{\mathtt{x}}$ *(resp.* $\to_{\mathtt{x}^-}$*) and* $\to_B$*.*

The following example is valid for both calculi (i.e. the general version of the garbage collection rule is not used):

Example 2.3.7. *Let* $I = \lambda x.x$ *and* $t = (\lambda x.x\ y)\ I$*. The term* t *can reduce in the following way:* $t \to_B (x\ y)[x/I] \to_{SApp} x[x/I]\ y[x/I] \to_{SVar} I\ y[x/I] \to_{SGc} I\ y \to_B x[x/y] \to_{SVar} y$*.*

They both share the same properties:

- Simulation of the λ-calculus
- Confluence
- Strong normalisation on typed terms (using the typing system described in Definition 2.3.3.
- PSN

They however do not have neither Full Composition nor metaconfluence as shown in the introduction.

2.3.2 The Calculus $\lambda\mathtt{j}$

The $\lambda\mathtt{j}$-calculus [AK10] will be considered in Chapter 4 when we study its metaconfluence, and in Chapter 6 when we formalize it, and prove that it enjoys Full Composition.

Definition 2.3.8 ($\lambda\mathtt{j}$)**.** *The* $\mathtt{j}$*-terms are those of Definition 2.3.1 and the reduction rules are:*

$$\begin{array}{lll}
(\lambda x.t)\mathtt{L}\ u & \to_{\mathtt{dB}} & t[x/u]\mathtt{L} \\
t[x/u] & \to_{\mathtt{w}} & t \qquad\qquad |t|_x = 0 \\
t[x/u] & \to_{\mathtt{d}} & t\{x/u\} \qquad\quad |t|_x = 1 \\
t[x/u] & \to_{\mathtt{c}} & t_{x\rightsquigarrow y}[x/u][y/u] \quad |t|_x > 1 \ \&\ y \textit{ fresh} \\
t[x/u][y/v] & \equiv & t[y/v][x/u] \qquad x \notin \mathtt{fv}(v)\ \&\ y \notin \mathtt{fv}(u)
\end{array}$$

where $\mathtt{L}$ *is a list of substitutions (possibly empty) and* $t_{x\rightsquigarrow y}$ *is the non-deterministic renaming of* m *(*$1 \leq m < |t|_x$*) occurrences of* x *in* t *by* y*. The rule creating an explicit substitution is called* $\mathtt{dB}$ *because it is a generalisation of the rule* B *introduced before (Definition 2.3.6) but acts at a* **d***istance. The reduction relation* $\to_{\mathtt{j}}$ *(resp.* $\to_{\lambda\mathtt{j}}$*) is generated by the rules* $\to_{\mathtt{w}}, \to_{\mathtt{d}}, \to_{\mathtt{c}}$ *(resp.* $\to_{\mathtt{w}}, \to_{\mathtt{d}}, \to_{\mathtt{c}}$ *and* $\mathtt{dB}$*)) modulo* $\equiv$*.*

Example 2.3.9. *The reduction of the term* $(x\ x)[x/y]$ *gives either* $(x\ x)[x/y] \to_{\mathtt{c}} (x_1\ x_2)[x_1/y][x_2/y] \to_{\mathtt{d}}^* y\ y$ *or* $(x\ x)[x/y] \to_{\mathtt{c}} (x_2\ x_1)[x_1/y][x_2/y]$*.*

There is a deep correspondence between the $\mathtt{j}$-terms and the $\mathtt{j}$-dags [AG09], a graphical formalism in between proof-nets and graph rewriting. The explicit substitution $t[x/u]$ corresponds in this framework to a special link called **jump** in the graphical framework. The three rules of $\to_{\mathtt{j}}$ propagating explicit substitutions have their counterpart in the logical rules of the $\mathtt{j}$-dags. Indeed, coming back to Definition 2.3.8, the rule $\to_{\mathtt{w}}$ stands for weakening, the rule $\to_{\mathtt{d}}$ stands for dereliction, and the rule $\to_{\mathtt{c}}$ stands for contraction. In particular the origin of the contraction rule with a non-determinisic renaming is very natural in this graphical formalism.

The $\lambda\mathtt{j}$-calculus enjoys all the properties expected from a calculus with explicit substitutions listed in the introduction [AK10]. Particularly metaconfluence is shown in Chapter 4 of this thesis. The two following measures on $\mathtt{j}$-terms will be used in Chapter 4 to show termination of $\to_{\mathtt{j}}$ on metaterms, and also in Chapter 3 to show termination of $\to_{\mathtt{s}}$ on $\mathtt{s}$-terms (cf. infra):

Definition 2.3.10 (Potential multiplicity)**.** *The potential multiplicity of the variable* x *in a term* t *is defined as* $M_x(t) := 0$ *if* $x \notin \mathtt{fv}(t)$*. Otherwise,*

$$\begin{array}{lll}
M_x(x) & := & 1 \\
M_x(\lambda y.t) & := & M_x(t) \\
M_x(tu) & := & M_x(t) + M_x(u) \\
M_x(t[y/u]) & := & M_x(t) + M_y(t) \cdot max(1, M_x(u))
\end{array}$$

Definition 2.3.11 (Whole multiplicity)**.** *The whole multiplicity of a term t is defined as:*

$$\begin{array}{lcl} \mathtt{jm}(x) & := & [\,] \\ \mathtt{jm}(\lambda y.t) & := & \mathtt{jm}(t) \\ \mathtt{jm}(tu) & := & \mathtt{jm}(t) \sqcup \mathtt{jm}(u) \\ \mathtt{jm}(t[y/u]) & := & [M_y(t)] \sqcup \mathtt{jm}(t) \sqcup max(1, M_y(t)) \cdot \mathtt{jm}(u) \end{array}$$

where:

- $\sqcup$ *is the multiset union,*
- $n \cdot [a_1, ..., a_k] := [n \cdot a_1, ..., n \cdot a_k]$,
- *and* $[\,]$ *is the empty multiset.*

This last measure, based on the previous one, decreases for each rule of $\mathtt{j}$.

2.3.3 The Calculus $\lambda_{\mathtt{s}}$

The $\lambda_{\mathtt{s}}$-calculus is the heart of the prismoid of resources presented in Chapter 5. The PSN property for $\lambda\mathtt{s}$ will be proved separately in Chapter 3.

Definition 2.3.12 ($\lambda_{\mathtt{s}}$)**.** *The* $\mathtt{s}$*-terms are those of Definition 2.3.1 and the reduction rules are:*

$$\begin{array}{llll} (\lambda x.t)\ u & \to_{\mathtt{B}} & t[x/u] & \\ t[x/u] & \to_{\mathtt{SGc}} & t & x \notin \mathtt{fv}(t) \\ x[x/u] & \to_{\mathtt{V}} & u & \\ (\lambda y.t)[x/u] & \to_{\mathtt{SL}} & \lambda y.t[x/u] & \\ (t\ v)[x/u] & \to_{\mathtt{SA_L}} & t[x/u]\ v & x \notin \mathtt{fv}(v) \\ (t\ v)[x/u] & \to_{\mathtt{SA_R}} & t\ v[x/u] & x \notin \mathtt{fv}(t) \\ t[y/v][x/u] & \to_{\mathtt{SS}} & t[y/v[x/u]] & x \in \mathtt{fv}(v) \setminus \mathtt{fv}(t) \\ t[x/u] & \to_{\mathtt{SDup}} & t_{x \rightsquigarrow y}[x/u][y/u] & |t|_x > 1 \ \&\ y \textit{ fresh} \\ t[x/u][y/v] & \equiv_{\mathtt{SS}_{\mathcal{C}}} & t[y/v][x/u] & y \notin \mathtt{fv}(u) \ \&\ x \notin \mathtt{fv}(v) \end{array}$$

The reduction relation $\Rightarrow_{\mathtt{s}}$ (resp. $\to_{\mathtt{s}}$) is generated by all the previous rules except $\mathtt{B}$ *(resp. modulo $\equiv_{\mathtt{SS}_{\mathcal{C}}}$).*

The reduction relation $\Rightarrow_{\lambda_{\mathtt{s}}}$ (resp. $\to_{\lambda_{\mathtt{s}}}$) is generated by all the previous rules (resp. modulo $\equiv_{\mathtt{SS}_{\mathcal{C}}}$).

The $\lambda\mathtt{s}$-calculus enjoys all the good properties, except metaconfluence for which it is an open question. The PSN property is shown in Chapter 3, and the other properties in Chapter 5. A version very similar (the side condition of the rule $\mathtt{SS}$ is simply $x \notin \mathtt{fv}(t)$) has been rediscovered [Gue11] in the context of proof search in the calculus of structures.

2.4 Logic

The Curry-Howard correspondence establishes a bijection between the minimal intuitionistic logic in natural deduction style and the λ-calculus. The notions of the two systems are related at the following levels:

- Proofs $\leftrightarrow$ programs
- Logic formulae $\leftrightarrow$ types
- Cut-elimination $\leftrightarrow$ β-reduction

This strong connection can be extended to calculi with explicit substitutions. By removing informations about terms, the set of typing rules

$$\frac{}{\Gamma, x : T \vdash x : T} \qquad \frac{\Gamma, x : U \vdash t : T \qquad \Gamma \vdash u : U}{\Gamma \vdash t[x/u] \ : T}$$

$$\frac{\Gamma, x : T \vdash t : U}{\Gamma \vdash \lambda x.t : T \to U} \qquad \frac{\Gamma \vdash t : T \to U \qquad \Gamma \vdash u : T}{\Gamma \vdash t\ u : U}$$

gives the following logic rules:

$$\frac{}{\Gamma, T \vdash T}\ \texttt{axiom} \qquad \frac{\Gamma, U \vdash T \qquad \Gamma \vdash U}{\Gamma \vdash T}\ \texttt{cut}$$

$$\frac{\Gamma, T \vdash U}{\Gamma \vdash T \to U} \to \texttt{intro} \qquad \frac{\Gamma \vdash T \to U \qquad \Gamma \vdash T}{\Gamma \vdash U} \to \texttt{elim}$$

The rules `axiom`, $\to$`intro`, and $\to$`elim` define the minimal intuitionistic logic in natural deduction style. The rule `cut` belongs however to the rules of sequent calculus. We thus have a mix between natural deduction and sequent calculus. It is possible to define a calculus with explicit substitutions such that there is a perfect correspondence between the calculus and intuitionistic sequent calculus [Her94].

In our logical system, all proofs can be written without the `cut`-rule as normal forms of calculi with explicit substitutions do not contain the explicit substitution constructor. The elimination of the rule `cut` thus corresponds to normalizing terms. For instance if we take the rule $x[x/u] \to u$ of $\lambda\mathtt{x}$, we have the following correspondence in the logical system:

$$\frac{\dfrac{}{\Gamma, x : T \vdash x : T}\ \texttt{axiom} \qquad \dfrac{\vdots}{\Gamma \vdash u : T}}{\Gamma \vdash x[x/u] \ : T}\ \texttt{cut} \qquad \rightsquigarrow_{(cut-elim)} \qquad \frac{\vdots}{u : T}$$

It is possible to go further in the decomposition of intuitionistic logic, as suggested by Linear Logic [Gir87], which provides a mechanism to explicitly control the

use of resources in logic. Indeed, intuitionistic logic can be encoded by a fragment called multiplicative exponential linear logic (MELL), which notably allows a fine control of erasure (weakening) and duplication (contraction).

MELL proofs can be specified by means of sequents, as usually done in logic, but Proof-Nets [Gir87] gives a better graphical formalism that eliminates bureaucratic syntactical details. Different notions of calculi with explicit substitutions were explained in terms of MELL Proof-Nets. A first example is [DCKP00]. Another relevant case is given by the $\lambda\mathtt{lxr}$-calculus [KL07], closed to the calculus $\lambda_{\mathtt{csw}}$ defined in Chapter 5, that can be understood as an algebraic specification of intuitionistic MELL Proof-Nets. Last but not least, as we have already stated in Section 2.3.2, there is a graphical formalism for λ-terms inspired by intuitionistic MELL known as $\mathtt{j}$-dags [AG09], that gives a bureaucratic free interpretation of the $\lambda\mathtt{j}$-terms.

Chapter 3

Preservation of Strong Normalisation of $\lambda_{\mathtt{s}}$

Contents

The goal of this chapter is to show Preservation of Strong Normalisation for the $\lambda_{\mathtt{s}}$-calculus, a particular sub-calculus of the prismoid of resources which will be introduced and studied in Chapter 5.

For that, we use a modular technique introduced in [Kes09], stating that PSN is just a consequence of the **IE**-property, which states that normalisation of a term affected by an **I**mplicit substitution implies normalisation of the same term affected by the corresponding **E**xplicit substitution. Formally,

$$\textbf{IE}\ u \in SN_{\lambda_{\mathtt{s}}}\ \&\ t\{x/u\}\overline{v_n} \in SN_{\lambda_{\mathtt{s}}} \text{ imply } t[x/u]\overline{v_n} \in SN_{\lambda_{\mathtt{s}}}$$

The **IE**-property is not easy to prove, we follow here the same ideas in [Kes09], which consists in the following steps:

- We enrich the $\lambda_{\mathtt{s}}$-grammar with labelled substitutions, which are used to mark strongly normalising $\mathtt{s}$-terms used as parameters of substitutions which remain strongly normalising all along the reduction sequences. For instance $t[\![x/u]\!]$ indicates that u is an $\mathtt{s}$-term (without labels) s.t $u \in SN_{\lambda_{\mathtt{s}}}$.

- We enrich the reduction system $\to_{\lambda_{\mathtt{s}}}$ with another system $\to_{\underline{\mathtt{s}}}$ (thus obtaining a new system $\to_{\lambda_{\underline{\mathtt{s}}}}$). The new system is only used to propagate labelled substitutions, and does not change the normalisable nature of labelled substitutions.

- We show that $u \in SN_{\lambda_{\mathtt{s}}}$ & $t\{x/u\}\overline{v_n} \in SN_{\lambda_{\mathtt{s}}}$ imply $t[\![x/u]\!]\overline{v_n} \in \mathcal{S}N_{\lambda_{\underline{\mathtt{s}}}}$. Doing so, we turn the substitution $\{x/u\}$ into a labelled substitution $[\![x/u]\!]$, particularly because $u \in SN_{\lambda_{\mathtt{s}}}$.
- We finally show that $t[\![x/u]\!]\overline{v_n} \in \mathcal{S}N_{\lambda_{\underline{\mathtt{s}}}}$ implies $t[x/u]\overline{v_n} \in \mathcal{S}N_{\lambda_{\mathtt{s}}}$. This is done by performing an unlabelling on labelled substitutions.

The two first points are developed in Section 3.1 and the two last points in Section 3.2. The implication $IE \implies PSN$ is developed in Section 3.3.

The proof technique we use in this section was introduced in [Kes09] to show PSN for the $\lambda\mathtt{ex}$-calculus, a simple calculus with explicit substitutions, containing in particular a composition rule and an equation (as in $\lambda_{\mathtt{s}}$). The main part of the proof in [Kes09] is not specific to the calculus $\lambda\mathtt{ex}$ since it is abstract enough to be applied to other calculi with explicit substitutions sharing the explicit grammar introduced in Definition 2.3.3 and enjoying Full Composition. The major difference in the PSN proof for $\lambda_{\mathtt{s}}$ that we present in this chapter is the termination proof for the labelled subcalculus. However, to give a self-contained presentation, we reproduce here several parts of the original proof in [Kes09].

3.1 The Labelling Technique

In this section we introduce the set of labelled terms and their associated reduction systems. The key idea is that bodies of labelled substitutions are strongly normalising terms, this invariant being kept during the reduction. The main rewriting system $\to_{\lambda_{\underline{\mathtt{s}}}}$ associated to labelled terms is split in two relations $\to_{\lambda_{\underline{\mathtt{s}}}{}^i}$ and $\to_{\lambda_{\underline{\mathtt{s}}}{}^e}$. The idea is that $\to_{\lambda_{\underline{\mathtt{s}}}{}^i}$ steps will be weakly projected (eventually empty steps) into $\lambda_{\mathtt{s}}$ whereas $\to_{\lambda_{\underline{\mathtt{s}}}{}^e}$ will be strongly projected (at least one step) into $\lambda_{\mathtt{s}}$. Formally if $\mathtt{xc}(\_)$ is a function mapping labelled terms to $\mathtt{s}$-terms, and t, t' are labelled terms then $t \to_{\lambda_{\underline{\mathtt{s}}}{}^i} t'$ implies $\mathtt{xc}(t) \to^*_{\lambda_{\mathtt{s}}} \mathtt{xc}(t')$ and $t \to_{\lambda_{\underline{\mathtt{s}}}{}^e} t'$ implies $\mathtt{xc}(t) \to^+_{\lambda_{\mathtt{s}}} \mathtt{xc}(t')$

The key lemma of this section states that $\to_{\lambda_{\underline{\mathtt{s}}}{}^i}$ is terminating.

3.1.1 The Labelled Terms

Given a set of variables $\mathbb{S}$, the $\mathbb{S}$-labelled terms or $\mathbb{S}$-terms (or simply labelled terms if $\mathbb{S}$ is clear from the context), are given by:

$$
\begin{array}{rcl}
T_{\mathbb{S}} & ::= & x \\
 & | & T_{\mathbb{S}}\ T_{\mathbb{S}} \\
 & | & \lambda x.T_{\mathbb{S}} \quad (x \notin \mathbb{S}) \\
 & | & T_{\mathbb{S}}[x/T_{\mathbb{S}}] \quad (x \notin \mathbb{S}) \\
 & | & T_{\mathbb{S}}[\![x/v]\!] \quad (v \in T_{\mathbb{S}} \cap SN_{\lambda_{\mathtt{s}}} \ \&\ \mathtt{fv}(v) \subseteq \mathbb{S} \ \&\ x \notin \mathbb{S})
\end{array}
$$

Labelled substitutions can only contain $\lambda_{\mathtt{s}}$-terms so in particular they cannot contain other labelled substitutions inside them. The reason to add $x \notin \mathbb{S}$ as a side-condition is to have a grammar stable by α-conversion. Remark that without

this condition, if $\mathbb{S} = \{y\}$ $x[\![x/y]\!][y/u]$ is α-equivalent to $x[\![x/z]\!][z/u]$, leading to a term not in the grammar anymore. We will thus only consider $\mathbb{S}$-terms where $x \notin \mathbb{S}$ in the terms of the form $u[x/v]$, $u[\![x/v]\!]$, and $\lambda x.v$. As we want to label normalising substitutions only at the root of a term this will not cause any problem.

Bodies of labelled substitutions are normalising by definition and do not loose this property thanks to the semantics of the rules propagating labelled substitutions. Indeed, these rules guarantee that labelled substitutions can traverse/commute normal substitutions but not the converse. The $\mathbb{S}$-terms will thus be trivially stable by the reduction defined by the following set of equations and rules:

Equations :

$t[x/u][\![y/v]\!]$	$\equiv_{\underline{\mathtt{SSc}}}$	$t[\![y/v]\!][x/u]$	$y \notin \mathtt{fv}(u)$ & $x \notin \mathtt{fv}(v)$
$t[\![x/u]\!][\![y/v]\!]$	$\equiv_{\underline{\mathtt{SSc}}}$	$t[\![y/v]\!][\![x/u]\!]$	$y \notin \mathtt{fv}(u)$ & $x \notin \mathtt{fv}(v)$

Rules :

$t[\![x/u]\!]$	$\to_{\underline{\mathtt{SGc}}}$	t	$x \notin \mathtt{fv}(t)$
$x[\![x/u]\!]$	$\to_{\underline{\mathtt{V}}}$	u	
$(\lambda y.t)[\![x/u]\!]$	$\to_{\underline{\mathtt{SL}}}$	$\lambda y.t[\![x/u]\!]$	
$(t\ v)[\![x/u]\!]$	$\to_{\underline{\mathtt{SA_L}}}$	$t[\![x/u]\!]\ v$	$x \notin \mathtt{fv}(v)$
$(t\ v)[\![x/u]\!]$	$\to_{\underline{\mathtt{SA_R}}}$	$t\ v[\![x/u]\!]$	$x \notin \mathtt{fv}(t)$
$t[y/v][\![x/u]\!]$	$\to_{\underline{\mathtt{SS}}}$	$t[y/v[\![x/u]\!]]$	$x \in \mathtt{fv}(v) \setminus \mathtt{fv}(t)$
$t[\![x/u]\!]$	$\to_{\underline{\mathtt{SDup}}}$	$t_{x \rightsquigarrow y}[\![x/u]\!][\![y/u]\!]$	$\lvert t\rvert_x > 1$ & y fresh

The $\Rightarrow_{\underline{\mathtt{s}}}$ (resp. $\to_{\underline{\mathtt{s}}}$) reduction relation is generated by all the previous reduction rules (resp. modulo $\equiv_{\underline{\mathtt{SSc}}}$) conversion. These relations can be simulated by their corresponding unlabelled reduction relations on $\mathtt{s}$-terms. We also consider the relation $\to_{\lambda_{\underline{\mathtt{s}}}} := \to_{\lambda_{\mathtt{s}}} \cup \to_{\underline{\mathtt{s}}}$ (resp. $\Rightarrow_{\lambda_{\mathtt{s}}} := \Rightarrow_{\lambda_{\mathtt{s}}} \cup \Rightarrow_{\underline{\mathtt{s}}}$) on labelled terms where $\to_{\lambda_{\mathtt{s}}}$ and $\Rightarrow_{\lambda_{\mathtt{s}}}$ are the reduction relations given in Definition 2.3.12.

Example 3.1.1. *The $\mathbb{S}$-term $(z\ z)[z/y][\![y/x]\!]$ can be reduced like this:*

$$
\begin{array}{ll}
(z\ z)[z/y][\![y/x]\!] & \to_{\lambda_{\underline{\mathtt{s}}}} \\
(z[z/y]\ z[z/y])[\![y/x]\!] & \to_{\lambda_{\underline{\mathtt{s}}}} \\
(z[z/y]\ z[z/y'])[\![y/x]\!][\![y'/x]\!] & \to^*_{\lambda_{\underline{\mathtt{s}}}} \\
(y\ y')[\![y/x]\!][\![y'/x]\!] & \to^*_{\lambda_{\underline{\mathtt{s}}}} \\
(x\ x) &
\end{array}
$$

3.1.2 Internal and External Reductions

We now split $\to_{\lambda_{\underline{\mathtt{s}}}}$ in two disjoint relations $\to_{\lambda\underline{\mathtt{s}}^i}$ and $\to_{\lambda\underline{\mathtt{s}}^e}$ which will be projected into $\lambda_{\mathtt{s}}$-reduction sequences differently.

Definition 3.1.2. *The internal reduction relation $\to_{\lambda\underline{\mathtt{s}}^i}$ is taken as the following reduction relation $\Rightarrow_{\lambda\underline{\mathtt{s}}^i}$ on $\equiv_{\mathtt{SSc},\underline{\mathtt{SSc}}}$-equivalence classes:*

- *If $u \Rightarrow_{\lambda_{\mathtt{s}}} u'$, then $t[\![x/u]\!] \Rightarrow_{\lambda\underline{\mathtt{s}}^i} t[\![x/u']\!]$.*

- *If* $t \Rightarrow_{\underline{\mathtt{s}}} t'$*, then* $t \Rightarrow_{\lambda\underline{\mathtt{s}}^i} t'$*.*
- *If* $t \Rightarrow_{\lambda\underline{\mathtt{s}}^i} t'$*, then* $t\ u \Rightarrow_{\lambda\underline{\mathtt{s}}^i} t'\ u, u\ t \Rightarrow_{\lambda\underline{\mathtt{s}}^i} u\ t'$*,* $\lambda x.t \Rightarrow_{\lambda\underline{\mathtt{s}}^i} \lambda x.t', t[x/u] \Rightarrow_{\lambda\underline{\mathtt{s}}^i} t'[x/u]$*,* $u[x/t] \Rightarrow_{\lambda\underline{\mathtt{s}}^i} u[x/t']$*,* $t[\![x/u]\!] \Rightarrow_{\lambda\underline{\mathtt{s}}^i} t'[\![x/u]\!]$*.*

The external reduction relation $\rightarrow_{\lambda\underline{\mathtt{s}}^e}$ *is taken as the following reduction relation* $\Rightarrow_{\lambda\underline{\mathtt{s}}^e}$ *on* $\equiv_{\mathtt{SS_C},\underline{\mathtt{SS_C}}}$*-equivalence classes:*

- *If the redex contracted in the reduction* $t \Rightarrow_{\lambda_{\mathtt{s}}} t'$ *occurs outside a labelled substitution, then* $t \Rightarrow_{\lambda\underline{\mathtt{s}}^e} t'$*.*
- *If* $t \Rightarrow_{\lambda\underline{\mathtt{s}}^e} t'$*, then* $tu \Rightarrow_{\lambda\underline{\mathtt{s}}^e} t'u, ut \Rightarrow_{\lambda\underline{\mathtt{s}}^e} ut', \lambda x.t \Rightarrow_{\lambda\underline{\mathtt{s}}^e} \lambda x.t', t[x/u] \Rightarrow_{\lambda\underline{\mathtt{s}}^e} t'[x/u], u[x/t] \Rightarrow_{\lambda\underline{\mathtt{s}}^e} u[x/t']$*,* $t[\![x/u]\!] \Rightarrow_{\lambda\underline{\mathtt{s}}^e} t'[\![x/u]\!]$*.*

Example 3.1.3. *The following reduction sequence illustrates the internal and the external notions of reduction:*

$$\begin{array}{ll} \lambda z.z[\![x/(\lambda y.y)\ y']\!] & \rightarrow_{\lambda\underline{\mathtt{s}}^i} \\ \lambda z.z[\![x/y[y/y']]\!] & \rightarrow_{\lambda\underline{\mathtt{s}}^i} \\ \lambda z.z[\![x/y']\!] & \rightarrow_{\lambda\underline{\mathtt{s}}^e} \\ \lambda z.z & \end{array}$$

Lemma 3.1.4. $\rightarrow_{\lambda_{\underline{\mathtt{s}}}} = \rightarrow_{\lambda\underline{\mathtt{s}}^i} \cup \rightarrow_{\lambda\underline{\mathtt{s}}^e}$

Proof. Straightforward. □

3.1.3 Termination of $\rightarrow_{\lambda\underline{\mathtt{s}}^i}$

As $\rightarrow_{\lambda\underline{\mathtt{s}}^i}$ will only be weakly projected into $\lambda_{\mathtt{s}}$, we need to guarantee that there are no infinite $\rightarrow_{\lambda\underline{\mathtt{s}}^i}$ reductions starting from a labelled term. This will be useful in Section 3.2 to relate termination of $\lambda_{\mathtt{s}}$ to that of $\rightarrow_{\lambda_{\underline{\mathtt{s}}}}$.

We show termination of $\rightarrow_{\lambda\underline{\mathtt{s}}^i}$ using several measures that will be combined using a lexicographic order.

The first one counts the number of free occurrences of variables, giving to them more weight if they appear in the body of labelled substitutions.

Definition 3.1.5.

$$\begin{array}{lcl@{\qquad}lcl} \mathtt{af}_x(z) & := & 0 & \mathtt{af}_x(\lambda y.t) & := & \mathtt{af}_x(t) \\ \mathtt{af}_x(x) & := & 1 & \mathtt{af}_x(t[\![y/u]\!]) & := & \mathtt{af}_x(t) + \mathtt{af}_y(t).\mathtt{af}_x(u) \\ \mathtt{af}_x(tu) & := & \mathtt{af}_x(t) + \mathtt{af}_x(u) & \mathtt{af}_x(t[y/u]) & := & \mathtt{af}_x(t) + \mathtt{af}_x(u) \end{array}$$

Remark that $\mathtt{af}_x(t) = 0$ if $x \notin \mathtt{fv}(t)$ and thus $\mathtt{af}_x(t[\![y/u]\!]) = \mathtt{af}_x(t)$ if $x \notin \mathtt{fv}(u)$. We also have $\mathtt{af}_x(t) = \mathtt{af}_y(t\{x/y\})$ for any y fresh.

Lemma 3.1.6. $\mathtt{af}_x(u[\![x]\!]) = \mathtt{af}_x(u[\![y]\!]) + \mathtt{af}_y(u[\![y]\!])$ *with* y *fresh.*

Proof. By induction on the number of free occurrences of x in u. □

We now define another function which counts the number of variables and give more weight to those appearing inside bodies of labelled substitution.

Definition 3.1.7. *Let* $\exp(k) = 2^{2^k}$.

$$\begin{array}{lcl@{\qquad}lcl} \mathtt{dep}(x) & := & 1 & \mathtt{dep}(t[x/u]) & := & \mathtt{dep}(t) + \mathtt{dep}(u) \\ \mathtt{dep}(\lambda y.t) & := & \mathtt{dep}(t) & \mathtt{dep}(tu) & := & \mathtt{dep}(t) + \mathtt{dep}(u) \\ \mathtt{dep}(t[\![x/u]\!]) & := & \mathtt{dep}(t) + \exp(\mathtt{af}_x(t)).\mathtt{dep}(u) & & & \end{array}$$

Let $\phi(t) := 1 + \eta_{\lambda_\mathtt{s}}(t) + \mathtt{maxk}_{\lambda_\mathtt{s}}(t)$ where $\mathtt{maxk}_{\lambda_\mathtt{s}}(t) := \mathtt{max}\{\mathtt{k}(t') | t \to^*_{\lambda_\mathtt{s}} t'\}$ with the following definition for the function k:

$$\begin{array}{lcl@{\qquad}lcl} \mathtt{k}(x) & := & 1 & \mathtt{k}(t[x/u]) & := & \mathtt{k}(t).(\mathtt{k}(u) + 1) \\ \mathtt{k}(\lambda x.t) & := & \mathtt{k}(t) + 1 & \mathtt{k}(t[\![x/u]\!]) & := & \mathtt{k}(t).\phi(u) \\ \mathtt{k}(tu) & := & \mathtt{k}(t) + \mathtt{k}(u) + 1 & & & \end{array}$$

Remark that $\mathtt{k}$ and ϕ are not mutually recursive because in the case of the labelled substitution of $\mathtt{k}$, there are no labelled substitutions in the subterm u so when $\phi(u)$ calls one more time $\mathtt{k}(\_)$, the case of the labelled substitution cannot be reached.

We have the following properties on the previous functions:

- $\phi(v) \geq 2$
- $v \to_{\lambda_\mathtt{s}} v'$ implies $\eta_{\lambda_\mathtt{s}}(v) > \eta_{\lambda_\mathtt{s}}(v')$ and $\mathtt{maxk}_{\lambda_\mathtt{s}}(v) \geq \mathtt{maxk}_{\lambda_\mathtt{s}}(v')$ so that $\phi(v) > \phi(v')$.

Furthermore, we need extra properties concerning the non-deterministic replacement:

Lemma 3.1.8. *Let x, z be distinct variables, and y a fresh variable s.t. $y \notin \mathbb{S}$.*

1. $\mathtt{dep}(t) = \mathtt{dep}(t_{x \rightsquigarrow y})$
2. $\mathtt{af}_z(t_{x \rightsquigarrow y}) = \mathtt{af}_z(t)$
3. $\mathtt{af}_x(t) = \mathtt{af}_x(t_{x \rightsquigarrow y}) + \mathtt{af}_y(t_{x \rightsquigarrow y})$

Proof. The first point is true since $\mathtt{dep}(\_)$ does not directly take into account variables, the only way there could be a difference would be a call to a call to $\mathtt{af}_x(\_)$ or $\mathtt{af}_y(\_)$ which is impossible by α-conversion.

The other points are straightforward. □

We show that $\mathtt{af}_x(t)$ is stable under reduction, and that $\mathtt{dep}(\_)$ and $\mathtt{k}(\_)$ decrease in such a way that we can prove that $\to_{\lambda\underline{\mathtt{s}}^i}$ terminates:

Lemma 3.1.9. *Let t, u be $\mathbb{S}$-terms and let $z \notin \mathbb{S}$.*

1. $t \equiv_{\mathtt{P}_{\underline{\mathtt{C}}}} t'$ *implies* $\mathtt{af}_z(t) = \mathtt{af}_z(t')$, $\mathtt{dep}(t) = \mathtt{dep}(t')$, *and* $\mathtt{k}(t) = \mathtt{k}(t')$.

2. $t \to_{\underline{\mathtt{SL}},\underline{\mathtt{SA_L}},\underline{\mathtt{SA_R}},\underline{\mathtt{SS}}} t'$ *implies* $\mathtt{af}_z(t) = \mathtt{af}_z(t')$, $\mathtt{dep}(t) = \mathtt{dep}(t')$, *and* $\mathtt{k}(t) > \mathtt{k}(t')$.

3. $t \to_{\underline{\mathtt{V}},\underline{\mathtt{SGc}},\underline{\mathtt{SDup}}} t'$ *implies* $\mathtt{af}_z(t) = \mathtt{af}_z(t')$ *and* $\mathtt{dep}(t) > \mathtt{dep}(t')$.

Proof. We only show the interesting cases, as the other ones are straightforward:

- If $t = t_1[\![x/u]\!][\![y/v]\!] \equiv_{\underline{\mathtt{SSc}}} t_1[\![y/v]\!][\![x/u]\!] = t'$, with $y \notin \mathtt{fv}(u)$ & $x \notin \mathtt{fv}(v)$, then

  –
  $$\begin{array}{ll}
  \mathtt{af}_z(t_1[\![x/u]\!][\![y/v]\!]) & = \\
  \mathtt{af}_z(t_1[\![x/u]\!]) + \mathtt{af}_y(t_1[\![x/u]\!]).\mathtt{af}_y(v) & = \\
  \mathtt{af}_z(t_1) + \mathtt{af}_x(t_1).\mathtt{af}_z(u) + \mathtt{af}_y(t_1[\![x/u]\!]).\mathtt{af}_z(v) & = \\
  \mathtt{af}_z(t_1) + \mathtt{af}_x(t_1).\mathtt{af}_z(u) + (\mathtt{af}_y(t_1) + \mathtt{af}_x(t_1).\mathtt{af}_y(u)).\mathtt{af}_z(v) & = \\
  \mathtt{af}_z(t_1) + \mathtt{af}_x(t_1).\mathtt{af}_z(u) + \mathtt{af}_y(t_1).\mathtt{af}_z(v) & = \\
  \mathtt{af}_z(t_1) + \mathtt{af}_y(t_1).\mathtt{af}_z(v) + (\mathtt{af}_x(t_1) + \mathtt{af}_y(t_1).\mathtt{af}_x(v)).\mathtt{af}_z(u) & = \\
  \mathtt{af}_z(t_1[\![y/v]\!]) + \mathtt{af}_x(t_1[\![y/v]\!]).\mathtt{af}_z(u) & = \\
  \mathtt{af}_z(t_1[\![y/v]\!][\![x/u]\!]) &
  \end{array}$$

  –
  $$\begin{array}{ll}
  \mathtt{dep}(t_1[\![x/u]\!][\![y/v]\!]) & = \\
  \mathtt{dep}(t_1[\![x/u]\!]) + \exp(\mathtt{af}_y(t_1[\![x/u]\!])).\mathtt{dep}(v)) & = \\
  \mathtt{dep}(t_1) + \exp(\mathtt{af}_x(t_1)).\mathtt{dep}(u) + \exp(\mathtt{af}_y(t_1[\![x/u]\!])).\mathtt{dep}(v) & = \\
  \mathtt{dep}(t_1) + \exp(\mathtt{af}_x(t_1)).\mathtt{dep}(u) + \exp(\mathtt{af}_y(t_1)).\mathtt{dep}(v) & = \\
  \mathtt{dep}(t_1[\![y/v]\!]) + \exp(\mathtt{af}_x(t_1[\![y/v]\!])).\mathtt{dep}(u) & = \\
  \mathtt{dep}(t_1[\![y/v]\!][\![x/u]\!]) &
  \end{array}$$

  – $\mathtt{k}(t) = \mathtt{k}(t_1).\phi(u).\phi(v) = \mathtt{k}(t')$

- If $t_1[y/t_2][\![x/v]\!] \to_{\underline{\mathtt{SS}}} t_1[y/t_2[\![x/v]\!]] = t'$ with $x \notin \mathtt{fv}(t_1)$ & $x \in \mathtt{fv}(t_2)$

  –
  $$\begin{array}{ll}
  \mathtt{af}_z(t_1[y/t_2][\![x/v]\!]) & = \\
  \mathtt{af}_z(t_1[y/t_2]) + \mathtt{af}_x(t_1[y/t_2]).\mathtt{af}_z(v) & = \\
  \mathtt{af}_z(t_1[y/t_2]) + \mathtt{af}_x(t_2).\mathtt{af}_z(v) & = \\
  \mathtt{af}_z(t_1) + \mathtt{af}_z(t_2) + \mathtt{af}_x(t_2).\mathtt{af}_z(v) & = \mathtt{af}_z(t')
  \end{array}$$

  –
  $$\mathtt{dep}(t) = \mathtt{dep}(t_1) + \mathtt{dep}(t_2) + \exp(\mathtt{af}_x(t_2)).\mathtt{dep}(v) = \mathtt{dep}(t')$$

  –
  $$\begin{array}{ll}
  \mathtt{k}(t_1[y/t_2][\![x/v]\!]) & = \\
  \mathtt{k}(t_1[y/t_2]).\phi(v) & = \\
  \mathtt{k}(t_1).(\mathtt{k}(t_2) + 1).\phi(v) & = \\
  \mathtt{k}(t_1).(\mathtt{k}(t_2).\phi(v) + \phi(v)) & > \\
  \mathtt{k}(t_1).(\mathtt{k}(t_2[\![x/v]\!]) + 1) & = \mathtt{k}(t')
  \end{array}$$

- If $t = x[\![x/v]\!] \to_{\underline{\mathtt{V}}} v = t'$, then

–
$$\begin{array}{ll} \mathtt{af}_z(x[\![x/v]\!]) & = \\ \mathtt{af}_z(x) + \mathtt{af}_x(x).\mathtt{af}_z(v) & = \\ \mathtt{af}_z(v) & \end{array}$$

–
$$\begin{array}{ll} \mathtt{dep}(x[\![x/v]\!]) & = \\ \mathtt{dep}(x) + \exp(\mathtt{af}_x(x)).\mathtt{dep}(v) & = \\ 1 + 4.\mathtt{dep}(v) & > \\ \mathtt{dep}(v) & \end{array}$$

- $t[\![x/u]\!] \to_{\underline{\mathtt{SDup}}} t_{x\rightsquigarrow y}[\![x/u]\!][\![y/u]\!]$ with $|t|_x > 1$

–
$$\begin{array}{ll} \mathtt{af}_z(t[\![x/u]\!]) & = \\ \mathtt{af}_z(t) + \mathtt{af}_x(t).\mathtt{af}_z(u) & =_{L.\ 3.1.8} \\ \mathtt{af}_z(t) + (\mathtt{af}_x(t_{x\rightsquigarrow y}) + \mathtt{af}_y(t_{x\rightsquigarrow y})).\mathtt{af}_z(u) & =_{L.\ 3.1.8} \\ \mathtt{af}_z(t_{x\rightsquigarrow y}) + \mathtt{af}_x(t_{x\rightsquigarrow y}).\mathtt{af}_z(u) + \mathtt{af}_y(t_{x\rightsquigarrow y}).\mathtt{af}_z(u) & = \\ \mathtt{af}_z(t_{x\rightsquigarrow y}[\![x/u]\!]) + \mathtt{af}_y(t_{x\rightsquigarrow y}).\mathtt{af}_z(u) & =_{(y\notin \mathtt{fv}(u))} \\ \mathtt{af}_z(t_{x\rightsquigarrow y}[\![x/u]\!]) + \mathtt{af}_y(t_{x\rightsquigarrow y}[\![x/u]\!]).\mathtt{af}_z(u) & = \mathtt{af}_z(t') \end{array}$$

–
$$\begin{array}{ll} \mathtt{dep}(t[\![x/u]\!]) & = \\ \mathtt{dep}(t) + \exp(\mathtt{af}_x(t)).\mathtt{dep}(u) & =_{L.\ 3.1.8} \\ \mathtt{dep}(t) + \exp(\mathtt{af}_x(t_{x\rightsquigarrow y}) + \mathtt{af}_y(t_{x\rightsquigarrow y})).\mathtt{dep}(u) & =_{L.\ 3.1.8} \\ \mathtt{dep}(t_{x\rightsquigarrow y}) + \exp(\mathtt{af}_x(t_{x\rightsquigarrow y}) + \mathtt{af}_y(t_{x\rightsquigarrow y})).\mathtt{dep}(u) & > \\ \mathtt{dep}(t_{x\rightsquigarrow y}) + \exp(\mathtt{af}_x(t_{x\rightsquigarrow y})).\mathtt{dep}(u) + \exp(\mathtt{af}_y(t_{x\rightsquigarrow y})).\mathtt{dep}(u) & = \\ \mathtt{dep}(t_{x\rightsquigarrow y}[\![x/u]\!]) + \exp(\mathtt{af}_y(t_{x\rightsquigarrow y})).\mathtt{dep}(u) & =_{(y\notin \mathtt{fv}(u))} \\ \mathtt{dep}(t_{x\rightsquigarrow y}[\![x/u]\!]) + \exp(\mathtt{af}_y(t_{x\rightsquigarrow y}[\![x/u]\!])).\mathtt{dep}(u) & = \mathtt{dep}(t') \end{array}$$

- All inductive cases easily hold because the measures take into account all the subterms.

□

Lemma 3.1.10. *The reduction relation $\to_{\underline{\mathtt{s}}}$ is terminating.*

Proof. Since $t \to_{\underline{\mathtt{s}}} t'$ implies $\langle \mathtt{dep}(t), \mathtt{k}(t)\rangle >_{lex} \langle \mathtt{dep}(t'), \mathtt{k}(t')\rangle$ by Lemma 3.1.9 and $>_{lex}$ is a well-founded relation, then $\to_{\underline{\mathtt{s}}}$ terminates. □

We can now conclude this section:

Lemma 3.1.11. *The reduction relation $\to_{\lambda\underline{\mathtt{s}}^i}$ is terminating.*

Proof. [Kes09] By Lemma 3.1.9(1), it is enough to consider $\Rightarrow_{\lambda\underline{\mathtt{s}}^i}$. We thus show that $t \Rightarrow_{\lambda\underline{\mathtt{s}}^i} t'$ implies $\mathtt{af}_z(t) \geq \mathtt{af}_z(t')$ for $z \notin \mathbb{S}$ and $\langle \mathtt{dep}(t), \mathtt{k}(t)\rangle >_{lex} \langle \mathtt{dep}(t'), \mathtt{k}(t')\rangle$. We proceed by induction on $\Rightarrow_{\lambda\underline{\mathtt{s}}^i}$.

- If $t \Rightarrow_{\lambda\underline{\mathtt{s}}^i} t'$ comes from $t \Rightarrow_{\underline{\mathtt{s}}} t'$ then we conclude with Lemma 3.1.9.

- If $u[\![x/v]\!] \Rightarrow_{\lambda\underline{\mathtt{s}}^i} u'[\![x/v]\!]$, then we get $\mathtt{af}_x(u) = \mathtt{af}_x(u')$ by i.h. If $\mathtt{dep}(u) > \mathtt{dep}(u')$, then $\mathtt{dep}(u[\![x/v]\!]) > \mathtt{dep}(u'[\![x/v]\!])$. If $\mathtt{dep}(u) = \mathtt{dep}(u')$, then $\mathtt{k}(u) > \mathtt{k}(u')$ so that $\mathtt{dep}(u[\![x/v]\!]) = \mathtt{dep}(u'[\![x/v]\!])$ and $\mathtt{k}(u[\![x/v]\!]) > \mathtt{k}(u'[\![x/v]\!])$.

- If $u[\![x/v]\!] \Rightarrow_{\lambda\underline{\mathtt{s}}^i} u[\![x/v']\!]$, then we have $\mathtt{af}_x(u) = \mathtt{af}_x(u')$ by i.h. If $\mathtt{dep}(v) > \mathtt{dep}(v')$, then $\mathtt{dep}(u[\![x/v]\!]) > \mathtt{dep}(u[\![x/v']\!])$. If $\mathtt{dep}(v) = \mathtt{dep}(v')$, then $\mathtt{dep}(u[\![x/v]\!]) = \mathtt{dep}(u[\![x/v']\!])$ and and $\phi(v) > \phi(v')$ implies $\mathtt{k}(u[\![x/v]\!]) = \mathtt{k}(u[\![x/v']\!])$.

□

3.2 The IE Property

We can now prove the first elementary step of the (IE) property, connecting the normalisation of an implicit substitution in the reduction system $\lambda_{\mathtt{s}}$ to the normalisation of a labelled substitution in $\lambda_{\underline{\mathtt{s}}}$:

$$u \in SN_{\lambda_{\mathtt{s}}} \ \&\ t\{x/u\}\overline{v_n} \in SN_{\lambda_{\mathtt{s}}} \text{ imply } t[\![x/u]\!]\overline{v_n} \in \mathcal{S}N_{\lambda_{\underline{\mathtt{s}}}}$$

This requires, in addition to the termination of the reduction system $\rightarrow_{\lambda\underline{\mathtt{s}}^i}$, to project the steps of the labelled reductions in $\rightarrow_{\lambda_{\mathtt{s}}}$-steps.

3.2.1 Projection

We define a function $\mathtt{xc}$ which maps labelled terms to $\mathtt{s}$-terms.

$$\begin{array}{lcl@{\qquad}lcl}
\mathtt{xc}(x) & := & x & \mathtt{xc}(t[x/u]) & := & \mathtt{xc}(t)[x/\mathtt{xc}(u)] \\
\mathtt{xc}(tu) & := & \mathtt{xc}(t)\ \mathtt{xc}(u) & \mathtt{xc}(t[\![x/v]\!]) & := & \mathtt{xc}(t)\{x/v\} \\
\mathtt{xc}(\lambda y.t) & := & \lambda y.\mathtt{xc}(t) & & &
\end{array}$$

Lemma 3.2.1. *Let t be a labelled term. If $t \rightarrow_{\underline{\mathtt{s}}} t'$, then $\mathtt{xc}(t) = \mathtt{xc}(t')$*

Proof. The interesting case is $t = t_1[y/t_2][\![x/v]\!] \rightarrow_{\mathtt{SS}} t_1[y/t_2[\![x/v]\!]] = t'$ with $x \in \mathtt{fv}(t_2) \setminus \mathtt{fv}(t_1)$:

$$\begin{array}{ll}
\mathtt{xc}(t) & = \\
\mathtt{xc}(t_1)[y/\mathtt{xc}(t_2)]\{x/v\} & = \\
\mathtt{xc}(t_1)\{x/v\}[y/\mathtt{xc}(t_2)\{x/v\}] & = \\
\mathtt{xc}(t_1)[y/\mathtt{xc}(t_2)\{x/v\}] & = \mathtt{xc}(t')
\end{array}$$

□

Lemma 3.2.2 (Projecting $\rightarrow_{\lambda_{\mathtt{s}}}$)**.** *Let t, t' be labelled terms. Then,*

1. *$t \equiv_{\mathtt{SS_C}} t'$ or $t \equiv_{\underline{\mathtt{SS_C}}} t'$ implies $\mathtt{xc}(t) = \mathtt{xc}(t')$*

2. *$t \Rightarrow_{\lambda\underline{\mathtt{s}}^i} t'$ implies $\mathtt{xc}(t) \rightarrow^*_{\lambda_{\mathtt{s}}} \mathtt{xc}(t')$*

3. *$t \Rightarrow_{\lambda\underline{\mathtt{s}}^e} t'$ implies $\mathtt{xc}(t) \rightarrow^+_{\lambda_{\mathtt{s}}} \mathtt{xc}(t')$*

Proof. [Kes09]

1. By induction on the conversion relation.

2. In the case of an internal reduction, the interesting cases are:

   - If $u[\![x/v]\!] \Rightarrow_{\lambda_{\underline{\mathtt{s}}}^i} u[\![x/v']\!]$ comes from $v \Rightarrow_{\lambda_\mathtt{s}} v'$ then $\mathtt{xc}(u[\![x/v]\!]) = \mathtt{xc}(u)\{x/v\} \Rightarrow_{\lambda_\mathtt{s}} \mathtt{xc}(u)\{x/v'\} = \mathtt{xc}(u[\![x/v']\!])$
   - If $t \Rightarrow_{\lambda_{\underline{\mathtt{s}}}^i} t'$ comes from $t \Rightarrow_{\underline{\mathtt{s}}} t'$ (so that also $t \to_{\underline{\mathtt{s}}} t'$), then Lemma 3.2.1 gives $\mathtt{xc}(t) = \mathtt{xc}(t')$.

3. In the case of an external reduction, the interesting cases are:

   - If $t \Rightarrow_{\lambda_{\underline{\mathtt{s}}}^e} t'$ comes from a reduction $t \Rightarrow_{\lambda_\mathtt{s}} t'$ which occurs outside a labelled substitution, then $\mathtt{xc}(t) \to^*_{\lambda_\mathtt{s}} \mathtt{xc}(t')$ can be easily shown by induction on $t \Rightarrow_{\lambda_\mathtt{s}} t'$.
   - If $u[\![x/v]\!] \Rightarrow_{\lambda_{\underline{\mathtt{s}}}^i} u'[\![x/v]\!]$ comes from $u \Rightarrow_{\lambda_{\underline{\mathtt{s}}}^i} u'$, then $\mathtt{xc}(u[\![x/v]\!]) = \mathtt{xc}(u)\{x/v\} \to^*_{\lambda_\mathtt{s}(i.h.)} \mathtt{xc}(u')\{x/v\} = \mathtt{xc}(u'[\![x/v]\!])$.

□

Lemma 3.2.3. *Let t be a labelled term. If $\mathtt{xc}(t) \in \mathcal{S}N_{\lambda_\mathtt{s}}$ then $t \in \mathcal{S}N_{\lambda_{\underline{\mathtt{s}}}}$*

Proof. [Kes09] We apply the Theorem 2.1.6 by taking $R_1 = \to_{\lambda_{\underline{\mathtt{s}}}^i}$, $R_2 = \to_{\lambda_{\underline{\mathtt{s}}}^e}$, $R = \lambda_\mathtt{s}$ and $a\ g\ b$ iff $\mathtt{xc}(a) = b$. Lemma 3.2.2 guarantees Properties P1 and P2, and Lemma 3.1.11 guarantees Property P3. We then get that $\mathtt{xc}(t) \in SN_{\lambda_\mathtt{s}}$ implies $t \in \mathcal{S}N_{\to_{\lambda_{\underline{\mathtt{s}}}^i} \cup \to_{\lambda_{\underline{\mathtt{s}}}^e}}$ which is exactly $\mathcal{S}N_{\to_{\lambda_{\underline{\mathtt{s}}}}}$ by Lemma 3.1.4. We thus conclude. □

Corollary 3.2.4. *Let $t, u, \overline{v_n}$ be $\mathtt{s}$-terms. If $u \in \mathcal{S}N_{\lambda_\mathtt{s}}$ and $t\{x/u\}\overline{v_n} \in \mathcal{S}N_{\lambda_\mathtt{s}}$ then $t[\![x/u]\!]\overline{v_n} \in \mathcal{S}N_{\lambda_{\underline{\mathtt{s}}}}$*

Proof. Take $\mathbb{S} = \mathtt{fv}(u)$. The hypothesis $u \in \mathcal{S}N_{\lambda_\mathtt{s}}$ allows us to construct the $\mathbb{S}$-labelled term $t[\![x/u]\!]\overline{v_n}$. Moreover $\mathtt{xc}(t) = t$ so that $\mathtt{xc}(t[\![x/u]\!]\overline{v_n}) = t\{x/u\}\overline{v_n}$ and we thus conclude by Lemma 3.2.3. □

We can now prove the last elementary step, connecting the normalisation of a labelled substitution in the reduction system $\lambda_{\underline{\mathtt{s}}}$ to the normalisation of an explicit substitution in $\lambda_\mathtt{s}$:

$$t[\![x/u]\!]\overline{v_n} \in \mathcal{S}N_{\lambda_{\underline{\mathtt{s}}}} \text{ implies } t[x/u]\overline{v_n} \in \mathcal{S}N_{\lambda_\mathtt{s}}$$

This requires to perform an unlabelling, transforming labelled substitutions into regular explicit substitutions.

3.2.2 Unlabelling

We define a function $\mathtt{U}$ which maps $\mathbb{S}$-terms to $\mathtt{ex}$-terms.

Definition 3.2.5.

$$\begin{array}{lcl} \mathtt{U}(x) & := & x \\ \mathtt{U}(tu) & := & \mathtt{U}(t)\mathtt{U}(u) \\ \mathtt{U}(\lambda x.t) & := & \lambda x.\mathtt{U}(t) \\ \mathtt{U}(t[x/u]) & := & \mathtt{U}(t)[x/\mathtt{U}(u)] \\ \mathtt{U}(t[\![x/u]\!]) & := & \mathtt{U}(t)[x/\mathtt{U}(u)] \end{array}$$

Remark that $\mathtt{fv}(\mathtt{U}(t)) = \mathtt{fv}(t)$.

Lemma 3.2.6. *Let t_1 be labelled term s.t. $\mathtt{U}(t_1) \to_{\lambda_{\mathtt{s}}} u_2$. Then there exists a labelled term u_1 s.t. $t_1 \to_{\lambda_{\underline{\mathtt{s}}}} u_1$ and $\mathtt{U}(u_1) = u_2$.*

Proof. By induction on $\to_{\lambda_{\mathtt{s}}}$. The interesting cases are the following ones, the others being straightforward:

- $t_1 = t[\![x/u]\!]$ and:

$$\mathtt{U}(t_1) = \mathtt{U}(t)[x/\mathtt{U}(u)] \to_{\mathtt{SDup}} \mathtt{U}(t)_{x\rightsquigarrow y}[x/\mathtt{U}(u)][y/\mathtt{U}(u)] = u_2$$

  Notice that we can apply $\to_{\mathtt{SDup}}$ thanks to the preservation of free variables of $\mathtt{U}()$. We can conclude with $t_1 \to_{\underline{\mathtt{SDup}}} t_{x\rightsquigarrow y}[\![x/u]\!][\![y/u]\!] = u_1$.

- The case where $\mathtt{U}(t_1) = t[x/u][y/v] \to t[x/u[y/v]]$ with $y \in \mathtt{fv}(u) \setminus \mathtt{fv}(t)$ is the case which justifies the need of the set $\mathbb{S}$. Indeed there are several ways to label the $\mathtt{s}$-term $\mathtt{U}(t_1)$. For instance the labelling $t[\![x/u]\!][\![y/v]\!]$ (with $y \in \mathtt{fv}(u)$) cannot reduce on $t[\![x/u[\![y/v]\!]]\!]$. However thanks to the fact that $y \notin \mathbb{S}$ we have a contradiction.

□

Lemma 3.2.7. *Let $t \in T_{\mathbb{S}}$. If $t \in \mathcal{S}N_{\lambda_{\underline{\mathtt{s}}}}$, then $\mathtt{U}(t) \in SN_{\lambda_{\mathtt{s}}}$*

Proof. To show that $\mathtt{U}(t) \in SN_{\lambda_{\mathtt{s}}}$, we have to show that all reducts are in $SN_{\lambda_{\mathtt{s}}}$ i.e. $\forall t' \mathtt{U}(t) \equiv t_1 \Rightarrow_{\lambda_{\mathtt{s}}} t_2 \equiv t' \Rightarrow t' \in SN_{\lambda_{\mathtt{s}}}$. This is done by induction on $\eta_{\lambda_{\underline{\mathtt{s}}}}(t)$ using Lemma 3.2.6.

□

Taking $\mathbb{S} = \mathtt{fv}(u)$ and transforming the $\mathtt{s}$-term $s[x/u]\overline{u_n}$ into the $\mathbb{S}$-term $s[\![x/u]\!]\overline{u_n}$ we have the following special case:

Corollary 3.2.8. *If $s[\![x/u]\!]\overline{u_n} \in \mathcal{S}N_{\lambda_{\underline{\mathtt{s}}}}$, then we get $s[x/u]\overline{u_n} \in SN_{\lambda_{\mathtt{s}}}$.*

We can finally conclude this section:

Lemma 3.2.9 ((IE) Property)**.** *If $u \in SN_{\lambda_{\mathtt{s}}}$ and $s\{x/u\}\overline{u_n} \in SN_{\lambda_{\mathtt{s}}}$ then $s[x/u]\overline{u_n} \in SN_{\lambda_{\mathtt{s}}}$.*

Proof. By Corollaries 3.2.4 and 3.2.8. □

3.3 The PSN Proof

In this last section we deduce that the $\lambda_{\mathtt{s}}$-calculus enjoys indeed Preservation of Strong Normalisation. As we can use the same perpetual strategy defined in [Kes09] (thanks to the fact that it is a many-step strategy based on the Full Composition property), we simply have to follow the proof of [Kes09], performing only straightforward modifications.

The modular approach allows to deduce from the (IE) property that the following inductive definition gives exactly the set of strongly normalising $\mathtt{s}$-terms:

Definition 3.3.1. *The inductive set $\mathcal{I}SN$ is defined as follows:*

$$\frac{t_1, ..., t_n \in \mathcal{I}SN \quad n \geq 0}{xt_1..t_n \in \mathcal{I}SN}\ (\mathtt{var}) \qquad \frac{u[x/v]t_1...t_n \in \mathcal{I}SN \quad n \geq 0}{(\lambda x.u)vt_1...t_n \in \mathcal{I}SN}\ (\mathtt{app})$$

$$\frac{u\{x/v\}t_1...t_n \in \mathcal{I}SN \quad v \in \mathcal{I}SN \quad n \geq 0}{u[x/v]t_1...t_n \in \mathcal{I}SN}\ (\mathtt{subs}) \qquad \frac{u \in \mathcal{I}SN}{\lambda x.u \in \mathcal{I}SN}\ (\mathtt{abs})$$

Before being able to show that $\mathcal{I}SN$ is exactly the set of strongly normalising terms of $\lambda_{\mathtt{s}}$, we introduce the perpetual strategy of [Kes09]. It gives a reduct for any $t \notin NF_{\lambda_{\mathtt{s}}}$. There are four cases: if $t = x\ t_1...t_n$, rewrite the left-most t_i which is reducible ; if $t = \lambda x.u$, rewrite u; if $t = (\lambda x.s)u\overline{v_n}$, rewrite the head redex $(\lambda x.s)u$; if $t = s[x/u]\overline{v_n}$ and $u \notin SN_{\lambda_{\mathtt{s}}}$, rewrite u and if $u \in SN_{\lambda_{\mathtt{s}}}$, rewrite the head redex $s[x/u]$ using the full composition property.

Definition 3.3.2 (Perpetual Strategy)**.** *The many-step strategy $\rightsquigarrow$ is defined by induction:*

$$\frac{}{(\lambda x.t)u\overline{u_n} \rightsquigarrow t[x/u]\overline{u_n}}\ (p-\mathtt{B})$$

$$\frac{\overline{u_n} \in NF_{\lambda_{\mathtt{s}}} \quad t \rightsquigarrow t'}{x\overline{u_n}t\overline{v_m} \rightsquigarrow x\overline{u_n}t'\overline{v_m}}\ (\mathtt{p-var}) \qquad \frac{t \rightsquigarrow t'}{\lambda x.t \rightsquigarrow \lambda x.t'}\ (\mathtt{p-abs})$$

$$\frac{u \in SN_{\lambda_{\mathtt{s}}}}{t[x/u]\overline{u_n} \rightsquigarrow t\{x/u\}\overline{u_n}}\ (\mathtt{p-subs1}) \qquad \frac{u \notin SN_{\lambda_{\mathtt{s}}} \quad u \rightsquigarrow u'}{t[x/u]\overline{u_n} \rightsquigarrow t[x/u']\overline{u_n}}\ (\mathtt{p-subs2})$$

The strategy is deterministic so that $t \rightsquigarrow u$ and $t \rightsquigarrow v$ implies $u = v$. The strategy is also perpetual which is the key to show that $\mathcal{I}SN$ is equal to the set $SN_{\lambda_{\mathtt{s}}}$. Notice that this is a many-step strategy because we use Full Composition in the rule `p-subs1` to transform the explicit substitution $\_[x/u]$ into $\_\{x/u\}$. This allows to be independent of the rules specific to a calculus and thus have the same definition for all the calculi having Full Composition.

We can show that $\rightsquigarrow$ only uses $\rightarrow_{\lambda_{\mathtt{s}}}$ reduction steps, and nothing more:

Lemma 3.3.3 ($\rightsquigarrow$ is a strategy for $\rightarrow_{\lambda_{\mathtt{s}}}$)**.** *If $t \rightsquigarrow t'$, then $t \rightarrow^+_{\lambda_{\mathtt{s}}} t'$.*

Proof. By induction on $\rightsquigarrow$, using Full Composition. □

Theorem 3.3.4 (Perpetuality Theorem)**.** *Let $t \rightsquigarrow t'$. If $t' \in SN_{\lambda_{\mathsf{s}}}$, then $t \in SN_{\lambda_{\mathsf{s}}}$.*

Proof. [Kes09]. By induction on the strategy $\rightsquigarrow$.

- $t = x\overline{u_n}t\overline{v_m} \rightsquigarrow x\overline{u_n}t'\overline{v_m}$ by (p-var). If $x\overline{u_n}t'\overline{v_m} \in SN_{\lambda_{\mathsf{s}}}$ then $\overline{u_n}, t', \overline{v_m} \in SN_{\lambda_{\mathsf{s}}}$. By i.h. $t \in SN_{\lambda_{\mathsf{s}}}$ so that $x\overline{u_n}t\overline{v_m} \in SN_{\lambda_{\mathsf{s}}}$.
- $t = \lambda x.t \rightsquigarrow \lambda x.t'$ by (p-abs) is straightforward by induction hypothesis.
- $t = (\lambda x.s)u\overline{u_n} \rightsquigarrow s[x/u]\overline{u_n} = t'$ by (p-B). If $s[x/u]\overline{u_n} \in SN_{\lambda_{\mathsf{s}}}$ then $s, u, \overline{u_n} \in SN_{\lambda_{\mathsf{s}}}$. One shows by induction on $\eta_{\lambda\mathsf{s}}(s)+\eta_{\lambda\mathsf{s}}(u)+\eta_{\lambda\mathsf{s}}(\overline{u_n})$ that every reduct from $(\lambda x.s)u\overline{u_n}$ is in $SN_{\lambda_{\mathsf{s}}}$. Thus $(\lambda x.s)u\overline{u_n} \in SN_{\lambda_{\mathsf{s}}}$.
- $t = s[x/u]\overline{u_n} \rightsquigarrow s[x/u']\overline{u_n}$ by (p-subs2) such that $u \notin SN_{\lambda_{\mathsf{s}}}$ and $u \rightsquigarrow u'$. If $s[x/u']\overline{u_n} \in SN_{\lambda_{\mathsf{s}}}$ then in particular $u' \in SN_{\lambda_{\mathsf{s}}}$ and $u \in SN_{\lambda_{\mathsf{s}}}$ by i.h. So we can conclude by contradiction.
- $t = s[x/u]\overline{u_n} \rightsquigarrow s\{x/u\}\overline{u_n} = t'$ by (p-subs1) so that $u \in SN_{\lambda_{\mathsf{s}}}$. Then the **(IE)** property (Lemma 3.2.9) allows to conclude.

□

Lemma 3.3.5. $\mathcal{I}SN = \mathcal{S}N_{\lambda_{\mathsf{s}}}$

Proof. [Kes09].

- Given $t \in SN_{\lambda_{\mathsf{s}}}$, we prove that $t \in \mathcal{I}SN$ by induction on $\langle \eta_{\lambda_{\mathsf{s}}}(t), t\rangle$.
  - If $t = xt_1...t_n$, then in particular $t_1...t_n \in SN_{\lambda_{\mathsf{s}}}$ so that by i.h. $t_1...t_n \in \mathcal{I}SN$ and thus $t \in \mathcal{I}SN$ by case (`var`).
  - If $t = \lambda x.u$, then in particular $u \in SN_{\lambda_{\mathsf{s}}}$ so that by i.h. $u \in \mathcal{I}SN$ and thus $t \in \mathcal{I}SN$ by case (`abs`).
  - If $t = u[x/v]t_1...t_n$, then in particular $v \in SN_{\lambda_{\mathsf{s}}}$ so that by i.h. $v \in \mathcal{I}SN$. Moreover, $u[x/v]t_1...t_n \to^+_{\lambda_{\mathsf{s}}} u\{x/v\}t_1...t_n$ so that $t \in SN_{\lambda_{\mathsf{s}}}$ implies $\eta_{\lambda_{\mathsf{s}}}(u\{x/v\}t_1t_n) < \eta_{\lambda_{\mathsf{s}}}(t)$ and thus by i.h. $u\{x/v\}t_1...t_n \in \mathcal{I}SN$. We conclude that $t \in \mathcal{I}SN$ by case (`subs`).
  - If $t = (\lambda x.u)vt_1...t_n$, then $t \to_{\lambda_{\mathsf{s}}} u[x/v]t_1...t_n$ implies in particular $\eta_{\lambda_{\mathsf{s}}}(u[x/v]t_1...t_n) < \eta_{\lambda_{\mathsf{s}}}(t)$ so that by i.h. $u[x/v]t_1...t_n \in \mathcal{I}SN$. We conclude that $t \in \mathcal{I}SN$ by case (`app`).
- Given $t \in \mathcal{I}SN$, we prove that $t \in SN_{\lambda_{\mathsf{s}}}$ by induction on $t \in \mathcal{I}SN$ using Theorem 3.3.4.
  - If $t = xt_1...t_n \in \mathcal{I}SN$ comes from $t_1...t_n \in \mathcal{I}SN$, then $t_1...t_n \in SN_{\lambda_{\mathsf{s}}}$ holds by the i.h. and thus $t \in SN_{\lambda_{\mathsf{s}}}$.
  - If $t = \lambda x.u \in \mathcal{I}SN$ comes from $u \in \mathcal{I}SN$, then $u \in SN_{\lambda_{\mathsf{s}}}$ holds by the i.h. and thus $t \in SN_{\lambda_{\mathsf{s}}}$.

- If $t = u[x/v]t_1...t_n \in \mathcal{I}SN$ comes from $u\{x/v\}t_1...t_n \in \mathcal{I}SN$ and $v \in \mathcal{I}SN$, then by the i.h. $u\{x/v\}t_1...t_n \in SN_{\lambda_s}$ and $v \in SN_{\lambda_s}$. Thus in particular $t \rightsquigarrow u\{x/v\}t_1...t_n$ and by perpetuality we have $t \in SN_{\lambda_s}$.
- If $t = (\lambda x.u)vt_1...t_n \in \mathcal{I}SN$ comes from $u[x/v]t_1...t_n \in \mathcal{I}SN$, then $u[x/v]t_1...t_n \in SN_{\lambda_s}$ by the i.h. Since $t \rightsquigarrow u[x/v]t_1...t_n$, then we conclude $t \in SN_{\lambda_s}$ by perpetuality.

□

We can also use the inductive definition for strongly normalising λ-terms:

Definition 3.3.6 (Inductive definition of $\mathcal{S}N_\beta$[vR96])**.**

$$\frac{u \in \mathcal{S}N_\beta}{\lambda x.u \in \mathcal{S}N_\beta}\ (\mathtt{abs}_\beta) \qquad \frac{t_1, ..., t_n \in \mathcal{S}N_\beta \quad n \geq 0}{xt_1..t_n \in \mathcal{S}N_\beta}\ (\mathtt{var}_\beta)$$

$$\frac{u\{x/v\}t_1...t_n \in \mathcal{S}N_\beta \quad n \geq 0 \quad v \in \mathcal{S}N_\beta}{(\lambda x.u)vt_1...t_n \in \mathcal{S}N_\beta}\ (\mathtt{app}_\beta)$$

Theorem 3.3.7 (PSN for λ-terms)**.** *If $t \in \mathcal{S}N_\beta$, then $t \in SN_{\lambda_s}$.*

Proof. By induction on $\mathcal{S}N_\beta$, using Definition 3.3.1 thanks to Lemma 3.3.5.

- If $t = xt_1...t_n$ with $t_i \in \mathcal{S}N_\beta$, then $t_i \in SN_{\lambda_s}$ by the i.h. so that the (`var`) rule allows to conclude.
- The case $t = \lambda x.u$ is similar.
- If $t = (\lambda x.u)vt_1...t_n$ with $u\{x/v\}t_1...t_n \in \mathcal{S}N_\beta$ and $v \in \mathcal{S}N_\beta$, then both terms are in SN_{λ_s} by the i.h. so that the (`subs`) gives $u[x/v]t_1...t_n \in SN_{\lambda_s}$ and the (`app`) rule gives $(\lambda x.u)vt_1...t_n \in SN_{\lambda_s}$.

□

Chapter 4

Confluence Results

Contents

Confluence is one of the essential properties which are desirable for functional calculi. While other properties such as PSN or Full Composition do not hold for some of the calculi presented in this thesis, all of them enjoy confluence on terms. It is then interesting to understand which are the common properties of all these calculi that are needed to guarantee confluence on terms.

In the first part of this Chapter we show confluence on terms in an axiomatic way for calculi with names, as done in [Kes96] for calculi with De Bruijn indices. This means that we develop an abstract proof of confluence for calculi with explicit substitution which is just based on a set of axioms. More precisely, we define a general scheme for calculi with explicit substitutions which describes (as axioms) the abstract operational properties that are sufficient to guarantee confluence on terms. In order to apply our abstract proof to a particular and concrete calculus, one just needs to verify that all the axioms hold for it.

Independently, in the second part of the Chapter, we show that λj enjoys confluence on metaterms. For that, we first extend the λj-calculus in two ways: we add metaterms to the original grammar of λj and we add an equation to its reduction system. We then characterise normal forms of metaterms in a concise and inductive way; this is particularly useful to carry out the proofs developed in this chapter. This characterization makes possible to forget the difficult aspects of the non-deterministic replacement operation of λj. We can finally apply the Tait-Martin Löf's technique to show confluence on metaterms; this is done by defining a subtle simultaneous reduction on metaterms with strongly uses the characterization of normal forms mentioned above.

4.1 Axiomatic Confluence

4.1.1 Generic Proof

In this section, we show that confluence on terms holds for an abstract calculus $\lambda\mathtt{prop}$ satisfying some axiomatic properties. The intended meaning of this abstract calculus is to extract from all the λ-calculi with explicit substitutions existing in the litterature a uniform description of their terms and their behaviour, together with a set of common features needed to guarantee confluence on terms.

As there exist different kinds of substitutions in the litterature (simple, parrallel, with lists, ...), we define an abstract data type to define explicit substitutions in a generic way, so that we can capture all of them. This abstract data type will be then implemented in different ways by the various concrete calculi in the litterature. For example, we are able to encompass *simple* substitutions $t[x/u]$ as we have seen in Chapter 3 for $\lambda\mathtt{s}$, or *simple* substitutions as defined in the $\lambda\mathtt{x} \parallel \mathtt{c}$-calculus [Blo97].

The specification of the operational semantics of the abstract calculus is very liberal since we want to capture at the same time structural calculi (such as $\lambda\mathtt{x}$) as well as distance calculi (such as $\lambda\mathtt{j}$), which do not share in general common reduction rules. This is an important difference with [Kes96] that is not able to capture distance calculi. Moreover, we want to capture calculi specified by means of reduction rules and *equations*, another major difference with [Kes96] that is not able to handle equational systems. As a consequence, we do not precise the form of the reduction rules and the equations of the abstract calculus; the behaviour of the abstract calculi is just specified by means of a set of axioms to be enjoyed by the normal forms of terms (w.r.t. the substitution calculus).

Definition 4.1.1 (The abstract $\lambda\mathtt{prop}$-calculus)**.** *The terms of the abstract calculus* $\lambda\mathtt{prop}$*, and their abstract lists are defined by the following grammar:*

$$\mathcal{T},\mathcal{U} ::= x \mid \lambda x.\mathcal{T} \mid \mathcal{T}\ \mathcal{U} \mid \mathcal{T}\mathbb{S}$$

$$\mathbb{S} ::= \mathtt{nil} \mid \mathtt{add}(\mathbb{S}, x \mapsto \mathcal{T})$$

We use capital letters to emphasize that we are using terms containing macros for substitutions. The set of variables in the domain of $\mathbb{S}$ *is written* $\mathtt{dom}(\mathbb{S})$ *and defined as* $\mathtt{dom}(\mathtt{add}(\mathbb{S}, \mathtt{x} \mapsto \mathcal{U})) = \mathtt{dom}(\mathbb{S}) \cup \{\mathtt{x}\}$, $\mathtt{dom}(\mathtt{nil}) = \emptyset$.

We write $(x \mapsto \mathcal{U}) \in \mathbb{S}$ *if* $(x \mapsto \mathcal{U})$ *belongs to the list* $\mathbb{S}$.

We associate to the abstract calculus $\lambda\mathtt{prop}$ *a reduction relation* $\to_{\mathtt{prop}}$ *propagating explicit substitutions, and a rule* $\to_{\mathtt{B}}$ *creating explicit substitutions. The reduction relation* $\to_{\lambda\mathtt{prop}}$ *is generated by the reduction relations* $\to_{\mathtt{B}}$ *and* $\to_{\mathtt{prop}}$.

As we have done for all the calculi we have introduced, we consider abstract lists modulo α-conversion. For instance, $(\lambda y.\mathcal{T})\mathbb{S}$ means that $y \notin \mathtt{dom}(\mathbb{S})$. Furthermore, if $(x \mapsto \mathcal{U}) \in \mathbb{S}$ and $(y \mapsto \mathcal{V}) \in \mathbb{S}$ then $x \neq y$.

To enjoy confluence, it is enough for a calculus to enjoy the following properties:

1. Be able to implement the macros of the abstract calculus $\lambda\texttt{prop}$ i.e. there exists a translation $\int\cdot\wr$ from the abstract grammar of $\lambda\texttt{prop}$ to its own grammar. All calculi share the same cases for the variable, the abstraction, and the application: $\int x\wr := x$, $\int\lambda x.\mathcal{T}\wr := \lambda x.\int\mathcal{T}\wr$, $\int\mathcal{V}\ \mathcal{W}\wr := \int\mathcal{V}\wr\ \int\mathcal{W}\wr$. In the following, to have a lighter notation, we will simply write $\mathcal{T}$ instead of $\int\mathcal{T}\wr$.

2. Has a rule $\texttt{B}$ of the shape $(\lambda x.\mathcal{T})\mathbb{S}\ \ \mathcal{U} \to_{\texttt{B}} \mathcal{T}\,\texttt{add}(\mathbb{S}, x \mapsto \mathcal{U})$. where $\mathbb{S}$ can be empty. Remark that by α-conversion, $x \notin \texttt{dom}(\mathbb{S})$.

3. The subset of rules which propagates substitutions $\to_{\texttt{prop}}$ is confluent and strongly normalizing. It thus defines a function which will be written $\downarrow(\_)$. Furthermore, we define $\to_{\lambda\texttt{prop}}$ as the union of $\to_{\texttt{B}}$ and $\to_{\texttt{prop}}$

4. Every term $\downarrow(\mathcal{T})$ is a λ-term i.e. it does not contain any explicit substitution

5. The normal forms of the abstract reduction relation $\to_{\texttt{prop}}$ must verify the following equations:

   (a)
   - $\downarrow(x\,\texttt{nil}) = x$
   - $\downarrow(x\,\texttt{add}(\mathbb{S}, x \mapsto \mathcal{U})) = \downarrow(\mathcal{U}\mathbb{S})$
   - $\downarrow(x\,\texttt{add}(\mathbb{S}, y \mapsto \mathcal{V})) = \downarrow(x\mathbb{S})\{y/\downarrow(\mathcal{V})\}$ if $x \in \texttt{dom}(\mathbb{S})$
   - $\downarrow(x\mathbb{S}) = x$ if $x \notin \texttt{dom}(\mathbb{S})$

   (b) $\downarrow((\mathcal{V}\ \mathcal{W})\mathbb{S}) = \downarrow(\mathcal{V}\mathbb{S})\ \downarrow(\mathcal{W}\mathbb{S})$

   (c) $\downarrow((\lambda y.\mathcal{T})\mathbb{S}) = \lambda y.\downarrow(\mathcal{T}\mathbb{S})$

To illustrate the first and the second property, we can take for instance $\lambda\texttt{x}$. It is able to implement the macros of $\lambda\texttt{prop}$ since we can extend the translation $\int\cdot\wr$ to handle the case of the list with: $\int\mathcal{T}\,\texttt{nil}\wr := \int\mathcal{T}\wr$, and $\int\mathcal{T}\,\texttt{add}(\mathbb{S}, x \mapsto \mathcal{U})\wr := \int\mathcal{T}\mathbb{S}\wr[x/\int\mathcal{U}\wr]$. The calculus $\lambda\texttt{x}$ has a rule $\texttt{B}$ which is of the shape $(\lambda x.\mathcal{T})\mathbb{S}\ \mathcal{U} \to_{\texttt{B}} \mathcal{T}\,\texttt{add}(\mathbb{S}, x \mapsto \mathcal{U})$ where $\mathbb{S}$ empty.

We also allow the abstract calculus to be equipped with an equivalence relation $\equiv$ if the following property is satisfied:

6. If $\mathcal{T} \equiv \mathcal{T}'$ then $\downarrow(\mathcal{T}) = \downarrow(\mathcal{T}')$.

Following our previous convention, the notations $\to_{\texttt{B}}, \to_{\texttt{prop}}, \to_{\lambda\texttt{prop}}$ denote now reduction modulo $\equiv$.

From Property 5, we can deduce the following property about garbage collection:

Lemma 4.1.2. *Let $\mathcal{T}$,$\mathcal{U}$ be terms, x, y variables, and $\mathbb{S}$ an abstract list.*

If $\texttt{fv}(\mathcal{T}) \cap \texttt{dom}(\mathbb{S}) = \emptyset$ then $\downarrow(\mathcal{T}\mathbb{S}) = \downarrow(\mathcal{T})$.

Proof. By induction on t. □

We can also characterize the normal form of an abstract list:

Lemma 4.1.3. *Let $\mathcal{T}$,$\mathcal{U}$ be terms and $\mathbb{S}$ an abstract list Then, $\downarrow(\mathcal{T}\,\texttt{add}(\mathbb{S}, x \mapsto \mathcal{U})) = \downarrow(\mathcal{T}\mathbb{S})\{x/\downarrow(\mathcal{U})\}$*

Proof. First notice that by α-conversion, $\mathtt{dom}(\mathbb{S}) \cap \mathtt{fv}(\mathcal{U}) = \emptyset$. We proceed by induction on $\langle \mathtt{size}(\mathtt{dom}(\mathbb{S})), |\mathcal{T}| \rangle$ We then proceed by case analysis on $\mathcal{T}$:

- $\mathcal{T} = x$, $\downarrow(x\,\mathtt{add}(\mathbb{S}, x \mapsto \mathcal{U})) =_{P.\ 5a} \downarrow(\mathcal{U}\mathbb{S}) =_{L.\ 4.1.2} \downarrow(U) = x\{x/\downarrow(\mathcal{U})\} =_{L.\ 4.1.2} \downarrow(x\mathbb{S})\{x/\downarrow(\mathcal{U})\}$.
- $\mathcal{T} = y$ with $y \notin \mathtt{dom}(\mathbb{S})$, $\downarrow(y\,\mathtt{add}(\mathbb{S}, x \mapsto \mathcal{U})) =_{L.\ 4.1.2}= y =_{L.\ 4.1.2} \downarrow(y\mathbb{S})$
- $\mathcal{T} = y$ with $(y \mapsto \mathcal{V}) \in \mathbb{S}$. $\downarrow(y\,\mathtt{add}(\mathbb{S}, x \mapsto \mathcal{U})) =_{P.\ 5a} \downarrow(y\mathbb{S})\{x/\downarrow(\mathcal{U})\}$
- The cases $\mathcal{T} = \lambda z.\mathcal{V}$ and $\mathcal{T} = \mathcal{V}\ \mathcal{W}$ are straightforward by induction using Property 5.
- $\mathcal{T} = \mathcal{V}\mathbb{S}_1$,

$$\begin{array}{ll} \downarrow(\mathcal{V}\mathbb{S}_1\,\mathtt{add}(\mathbb{S}, x \mapsto \mathcal{U})) & =_{P.\ 3} \\ \downarrow(\downarrow(\mathcal{V}\mathbb{S}_1)\,\mathtt{add}(\mathbb{S}, x \mapsto \mathcal{U})) & =_{i.h.} \\ \downarrow(\downarrow(\mathcal{V}\mathbb{S}_1)\mathbb{S})\{x/\downarrow(\mathcal{U})\} & =_{P.\ 3} \\ \downarrow(\mathcal{V}\mathbb{S}_1\mathbb{S})\{x/\downarrow(\mathcal{U})\} & \end{array}$$

  We can apply the induction hypothesis since $\downarrow(V\mathbb{S}_1)$ contains no explicit substitutions thanks to Property 4. Notice that it will not be possible with metaterms since a metaterm normal form can contain explicit substitutions.

□

Every β-reduction can be simulated in $\lambda\mathtt{prop}$:

Lemma 4.1.4. *If $t \to_\beta t'$ then $t \to^*_{\lambda\mathtt{prop}} t'$*

Proof. We only consider the root case, as inductive ones are straightforward. The reduction is of the shape $t = (\lambda x.v)w \to_\beta v\{x/w\} = t'$. By the rule $\mathtt{B}$, $(\lambda x.v)w \to_B v\,\mathtt{add}(nil, x \mapsto w)$ By Lemma 4.1.3, with the special case when $\mathbb{S}$ is empty, we have $\downarrow(v\,\mathtt{add}(\mathtt{nil}, x \mapsto w)) = \downarrow(v)\{x/\downarrow(u)\} = v\{x/u\}$ since v and u are already in normal form as they are λ-terms without explicit substitutions. □

Lemma 4.1.5. *If $\mathcal{T} \to_{\mathtt{B}} \mathcal{T}'$ then $\downarrow(\mathcal{T}) \to^*_\beta \downarrow(\mathcal{T}')$.*

Proof. We only consider the root case, as the inductive ones are straightforward. We are thus in the situation where $\mathcal{T} = (\lambda z.\mathcal{V})\mathbb{S}\,\mathcal{U} \to_{\mathtt{B}} \mathcal{V}\,\mathtt{add}(\mathbb{S}, z \mapsto \mathcal{U}) = \mathcal{T}'$. Then, $\downarrow(\mathcal{T}) =_{P.\ 5} (\lambda z.\downarrow(\mathcal{V}\mathbb{S}))\ \downarrow(\mathcal{U}) \to_\beta \downarrow(\mathcal{V}\mathbb{S})\{z/\downarrow(\mathcal{U})\} =_{L.4.1.3} \downarrow(\mathcal{V}\,\mathtt{add}(\mathbb{S}, z \mapsto \mathcal{U}))$. □

We now introduce the key method, defined in [HL89] which let us benefit from the fact that $\mathtt{prop}$ is confluent and terminating:

Lemma 4.1.6 (Interpretation method modulo)**.** *Let $R = R_1 \cup R_2/\approx$ where R_1 is terminating, R_2 an arbitrary reduction, and $\approx$ an equivalence relation. If there exists a reduction $\Rrightarrow$ on the set of R_1 normal forms satisfying:*

1. $\Rrightarrow \subseteq R$

2. $t \to_{R_1/\approx} t' \;\Rightarrow \downarrow_{R_1}(t) \approx \downarrow_{R_1}(t')$

3. $t \to_{R_2/\approx} t' \;\Rightarrow \downarrow_{R_1}(t) \Rrightarrow^* \downarrow_{R_1}(t')$

then confluence of $\Rrightarrow$ *implies confluence of* R.

Proof. Suppose that $t \to_R t_1$ and $t \to_R t_2$. By definition, $t_1 \to_{R_1} \downarrow_{R_1}(t_1)$, $t_2 \to_{R_1} \downarrow_{R_1}(t_2)$, $t \to_{R_1} \downarrow_{R_1}(t)$. It is easy to show by induction on the length of the derivation that if $U \to_R U'$, then $\downarrow_{R_1}(U) \Rrightarrow^* \downarrow_{R_1}(U')$. Hence $\downarrow_{R_1}(t) \Rrightarrow^* \downarrow_{R_1}(t_1)$ and $\downarrow_{R_1}(t) \Rrightarrow^* \downarrow_{R_1}(t_2)$. By confluence of $\Rrightarrow$, it exists t_3 such that $\downarrow_{R_1}(t_1) \Rrightarrow^* t_3{}^* \Lleftarrow \downarrow_{R_1}(t_2)$. Finally the first hypothesis ensures that $t_1 \to_R t_3$ and $t_2 \to_R t_3$. □

Theorem 4.1.7. *The abstract calculus* $\lambda\texttt{prop}$ *is confluent.*

Proof. Taking $R_1 = \texttt{prop}$, $R_2 = \texttt{B}$, $\approx = \equiv$ and $\Rrightarrow = \to_\beta^*$, we respectively satisfy Points 1,2,3 of Lemma 4.1.6 using Lemma 4.1.4, Property 6, and Lemma 4.1.5. □

4.1.2 Applications

The generic proof we gave in the previous subsection can be applied to all calculi with names described in this thesis. Notice that since we have defined the behaviour of the propagation calculus through normal forms, we can deal both with structural calculi and calculi at distance. Almost all the properties are straightforward except the third one, requiring strong normalisation and confluence of the propagation subcalculus.

We illustrate different applications of our abstract result (Theorem 4.1.7).

- $\lambda\texttt{x}$: this calculus is quite simple since $\mathbb{S}$ is empty in the $\texttt{B}$ rule of the abstract calculus $\lambda\texttt{prop}$. We recall the implementation: $\int\!\mathcal{T}\,\texttt{nil}\wr := \int\!\mathcal{T}\wr$, and $\int\!\mathcal{T}\,\texttt{add}(\mathbb{S}, x \mapsto \mathcal{U})\wr := \int\!\mathcal{T}\mathbb{S}\wr[x/\int\!\mathcal{U}\wr]$.

  As $\texttt{x}$ is convergent [Blo97], $\lambda\texttt{x}$ turns out to be confluent.

- $\lambda\texttt{j}$: the implementation is also defined as $\int\!\mathcal{T}\,\texttt{nil}\wr := \int\!\mathcal{T}\wr$, and $\int\!\mathcal{T}\,\texttt{add}(\mathbb{S}, x \mapsto \mathcal{U})\wr := \int\!\mathcal{T}\mathbb{S}\wr[x/\int\!\mathcal{U}\wr]$; the only difference being that $\mathbb{S}$ is not empty in the rule $\texttt{B}$.

  As $\texttt{j}$ is convergent [AK10], $\lambda\texttt{j}$ turns out to be confluent.

- $\lambda\texttt{x}||\texttt{c}$: this calculus is defined by the grammar:

  $$t, u ::= x \mid \lambda x.t \mid t\ u \mid t[x_1, ..., x_n/u_1, ..., u_m]$$

  The explicit substitution construction is shortened as $t[\overline{x}/\overline{u}]$. The rules of the system are:

  $$\begin{array}{lcll}
(\lambda x.t)\ u & \to_{\texttt{B}} & t[x/u] & \\
x_i[\overline{x}/\overline{u}] & \to & u & \\
y[\overline{x}/\overline{u}] & \to & u & \text{with } y \neq x_i \forall i \\
(\lambda y.v)[\overline{x}/\overline{u}] & \to & \lambda y.v[\overline{x}/\overline{u}] & \\
(v\ w)[\overline{x}/\overline{u}] & \to & v[\overline{x}/\overline{u}]\ w[\overline{x}/\overline{u}] & \\
t[\overline{x}/\overline{u}][\overline{y}/\overline{v}] & \to & t[\overline{x}, \overline{y}/u_1[\overline{y}/\overline{v}], ..., u_n[\overline{y}/\overline{v}]] &
\end{array}$$

It is enough to define $\int\mathcal{T}\,\mathtt{nil}\wr := \int\mathcal{T}\wr$ and $\int\mathcal{T}\,\mathtt{add}(\mathbb{S}, x \mapsto \mathcal{U})\wr := \int\mathcal{T}\mathbb{S}\wr[x/\int\mathcal{U}\wr]$ without handling the case of parallel substitutions. Indeed, a simple substitution is enough for the rule B and more complex parallel substitutions are only created by the propagating rules.

The system is terminating [Blo97] and the critical pairs are easily joinable. Thus, $\lambda\mathtt{x}||\mathtt{c}$ is confluent.

- $\lambda\mathtt{s}$: to prove termination for $\lambda\mathtt{s}$, we can use the measures defined for $\lambda\mathtt{j}$ in Chapter 2.3.2.

  Once again, the critical pairs are easily joinable and we can thus conclude that $\lambda\mathtt{s}$ turns out to be confluent.

4.2 Metaconfluence of $\lambda\mathtt{j}$

This section focuses on the $\lambda\mathtt{j}$-calculus extended with metavariables. We prove metaconfluence of the calculus equipped with an equivalence relation $\equiv$. This confers to $\lambda\mathtt{j}$ all the good properties that one can expect.

To prove metaconfluence we combine the interpretation method [Har89] with ideas from Tait-Martin Löf's technique [Bar84]. However, before being able to apply those standard methods, we need to specify some subtle properties of the non-deterministic replacement used in the specification of the operational semantics of $\lambda\mathtt{j}$. As the equivalence (sub)relation $\equiv$ is a strong bisimulation, we will be able to extend metaconfluence to the stronger Church-Rosser property.

While the standard method to show confluence for λ-calculus and associated calculi is the one from Tait-Martin Löf, it is the interpretation method, which seems to be the simplest in this field. Indeed, one can benefit from the fact that almost all reduction relations of explicit substitutions calculi can be split into a rule creating substitutions and others propagating them. Furthermore, it is often easy to show that this last set of rules is confluent and terminating. When dealing with terms, one can show confluence of calculi with explicit substitutions thanks to the confluence of the λ-calculus [ACCL90, KR97], while with metavariables, it is necessary to combine the notion of simultaneous reduction introduced by Tait and Martin Löf to obtain the diamond property of the interpretation calculus [KR95, Kes07].

The result will immediatly extend to the case where other equivalences defining strong bisimulations on $\lambda\mathtt{j}$ metaterms are added, such as the σ-equivalence of Regnier [Reg91]. Furthermore, our result implies metaconfluence on $\lambda\mathtt{j}$-dags, the graph formalism from which $\lambda\mathtt{j}$ is inspired, since there is a strong bisimulation between those two systems.

Related works: Other techniques exist to show confluence of calculi with explicit substitutions such as the Yokouchi-Hikita [YH90] method, used for example for $\lambda\sigma_{\Uparrow}$ [CHL96] and $\lambda\mathtt{x}$ [RBL09]; the Z-technique [vO08b], used for example for $\lambda\mathtt{ex}$ [Kes09]; the technique of decreasing diagrams [vO08a] used for $\lambda\mathtt{x}||\mathtt{c}$ [Blo97].

Section 4.2.1 introduces the meta $\lambda\mathtt{j}$-calculus and its properties; Section 4.2.2 defines notations and basic lemmas needed to handle the non-deterministic notion of replacement ; Section 4.2.3 presents the proof of metaconfluence strictly speaking; Section 4.2.4 generalizes metaconfluence to the Church-Rosser property.

4.2.1 The Meta $\lambda\mathtt{j}$-calculus and some Basic Properties

We extend the grammar of $\lambda\mathtt{j}$ with metavariables $\mathbb{X}_\Delta$ in order to denote incomplete programs. To soundly instantiate metavariables with terms, the set of variables Δ indexing or decorating $\mathbb{X}$ denote the set of variables which are free in it. Thus $\mathbb{X}_\Delta$ is a metaterm whose free variables are Δ.

Definition 4.2.1 (Metaterms)**.** *Terms of the calculus are defined by the following grammar, as usual modulo α-conversion:* $t, u ::= x \mid \lambda x.t \mid t\ u \mid t[x/u] \mid \mathbb{X}_\Delta$

The metaterm $y[x/v]\ \mathbb{X}_{y,z}$ can be instantiated for example by the term $y[x/v]\ y\ z$ or $y[x/v]\ ((\lambda x.xz)[w/y])$.

Definition 4.2.2 (Variables and Occurences)**.** *The notion of free variables is extended to metavariables by* $\mathtt{fv}(\mathbb{X}_\Delta) := \Delta$. *We write* $\mathtt{fmvar}(t)$ *for the set of all the free variables of all the metavariables of* t.

The number of free occurences of x in t is extended to metavariables, counting one in the case of $\mathbb{X}_\Delta$ with $x \in \Delta$. The notation $\|t\|_x$ represents the number of occurrences of x in metavariables of t.

The list of substitutions $[x_1/u]...[x_n/u]$ will be abbreviated as $[x_1, ..., x_n/u]$.

Definition 4.2.3 (Non-deterministic replacement on metaterms)**.** *The non-deterministic replacement given in Definition 2.3.8 is simply extended to metaterms.*

For instance, the term $(x[z/\mathbb{X}_x])_{x\rightsquigarrow y}$ denotes either $y[z/\mathbb{X}_x]$ or $x[z/\mathbb{X}_y]$ and $x\ \mathbb{X}_x[z/\mathbb{Z}_x]_{x\rightsquigarrow y_1,y_2}$ denotes either $x\ \mathbb{X}_{y_1}[z/\mathbb{Z}_{y_2}]$ or $x\ \mathbb{X}_{y_2}[z/\mathbb{Z}_{y_1}]$.

Definition 4.2.4 (Implicit substitution)**.** *The implicit substitution is defined on* t only *when* $x \notin \mathtt{fmvar}(t)$ [1].

$$\begin{array}{lcll}
x\{x/u\} & := & u & \\
y\{x/u\} & := & y & \\
(\lambda y.v)\{x/u\} & := & \lambda y.v\{x/u\} & x \neq y \\
(v\ w)\{x/u\} & := & v\{x/u\}\ w\{x/u\} & \\
v[y/w]\{x/u\} & := & v\{x/u\}[y/w\{x/u\}] & x \neq y
\end{array}$$

Considering implicit substitution in the context of metavariables, other approaches exist, consisting of changing back $\mathbb{X}_\Delta\{x/u\}$ into $\mathbb{X}_\Delta[x/u]$ even when x

[1] It makes no sense to define the implicit substitution when x appears in a metavariable because by definition, we do not know what would be the result since the structure of the metavariable is unknown

appears in a metavariable [Kes07]. This makes an important difference because doing so, one can move the implicit substitution downward the term and then finally transform it in a jump. The implicit substitution thus switches from a static to a dynamic status which is not that what one expects.

Implicit substitution enjoys the following well-known property:

Lemma 4.2.5. *Let* $x, y \notin \mathtt{fmvar}(t) \cup \mathtt{fmvar}(u)$. *Then,* $t\{x/u\}\{y/v\} = t\{y/v\}\{x/u\{y/v\}\}$.

Proof. By induction on t. □

We can now redefine rules of $\lambda\mathtt{j}$ in order to deal with metaterms as follows:

Definition 4.2.6 ($\lambda\mathtt{j}$ reduction rules)**.**

$$
\begin{array}{llll}
(\lambda x.t)\mathtt{L}\ u & \to_{\mathtt{dB}} & t[x/u]\mathtt{L} & \\
t[x/u] & \to_{\mathtt{w}} & t & \textit{if } |t|_x = 0 \\
t[x/u] & \to_{\mathtt{d}} & t\{x/u\} & \textit{if } |t|_x = 1 \textit{ and } x \notin \mathtt{fmvar}(t) \\
t[x/u] & \to_{\mathtt{c}} & t_{x \rightsquigarrow y}[x/u][y/u] & \textit{if } |t|_x > 1 \\
t[x/u][y/v] & \equiv & t[y/v][x/u] & \textit{if } x \notin \mathtt{fv}(v) \textit{ and } y \notin \mathtt{fv}(u)
\end{array}
$$

For instance a reduction of the metaterm $((\lambda z.x\ \ z)\ \ \mathbb{X}_x)[x/y]$ is $((\lambda z.x\ \ z)\ \ \mathbb{X}_x)[x/y] \to_{\mathtt{B}} (x\ \ z)[z/\mathbb{X}_x][x/y] \to_{\mathtt{d}} (x\ \ \mathbb{X}_x)[x/y] \to_{\mathtt{c}} (x_1\ \ \mathbb{X}_{x_2})[x_1/y][x_2/y] \to_{\mathtt{d}} (y\ \ \mathbb{X}_{x_2})[x_2/y]$ with the last metaterm in normal form.

Notice that rule $\mathtt{d}$ is stated in the original version of $\lambda\mathtt{j}$ (defined only on terms) by "$t[x/u] \to_{\mathtt{d}} t\{x/u\}$ if $|t|_x = 1$". In order to be consistent with Definition 4.2.4 of implicit substitution we adopt here a slight variation. As stated before, the propagation rules are defined by case analysis on the number of occurrences of the bound variable. The letters in the rules stand for weakening, derelicition, and contraction. $\mathtt{dB}$ is the rule $\mathtt{B}$ which is the usual name for the creation of jumps at a distance. The rule $\mathtt{c}$ should be considered as rules schema. It does not correspond to a single rule but to as many possible partitions between x and y exist. In particular, one can choose a different partition for each possible $\mathtt{c}$-reduction. Of course, when implementing the calculus, one should specify a particular strategy to partition variables. However, since we work with the calculus and not with a particular implementation of it, we will never specify it in the sequel.

The $\lambda\mathtt{j}$-calculus was first presented in [AK10] without $\equiv$ since the four rules are sufficient to enjoy almost all properties including confluence on terms. The equivalence relation is however needed to close the following critical pair on metaterms:

$$
\begin{array}{ccccc}
(\mathbb{X}_x\ \mathbb{Z}_y)[x/u][y/u] & \leftarrow_{\mathtt{c}} & (\mathbb{X}_x\ \mathbb{Z}_x)[x/u] & \to_{\mathtt{c}} & (\mathbb{X}_y\ \mathbb{Z}_x)[x/u][y/u] \\
\equiv & & & & =_\alpha \\
 & & (\mathbb{X}_x\ \mathbb{Z}_y)[y/u][x/u] & &
\end{array}
$$

Lemma 4.2.7 (Full composition)**.** *The* $\lambda\mathtt{j}$*-calculus enjoys full composition on terms [AK10] and metaterms i.e. if* $x \notin \mathtt{fmvar}(t)$*, then* $t[x/u] \to^*_{\lambda\mathtt{j}} t\{x/u\}$.

Proof. By induction on the number of occurrences of x in t. □

Lemma 4.2.8. *The reduction $\to_{\mathtt{j}}$ is terminating and confluent on metaterms. Thus for any term t, there is one unique normal form (up to $\equiv$) written $\downarrow(t)$.*

Proof. The confluence property is straightforward. For the termination property, the proof of [AK10] is modified by extending the notion of potential multiplicity given in Definition 2.3.10 to metavariables with $M_x(\mathbb{X}_\Delta) := 1$ if $x \in \Delta$, 0 otherwise. □

In general, all the proofs for $\lambda\mathtt{j}$ are the same when extending to metaterms since the kind (meta or not) of an occurrence is stable by reduction and substitution.

Lemma 4.2.9. *Metaterms in $\mathcal{NF}_{\mathtt{j}}$ are of the following shape with $u, v \in \mathcal{NF}_{\mathtt{j}}$:*

$$x \mid \mathbb{X}_\Delta \mid \lambda x.u \mid u\ v \mid u[x/v] \text{ with } \|u\|_x = |u|_x = 1$$

For instance $(\mathbb{X}_x\ y)[x/u]$ is in $\mathtt{j}$-normal form but $(\mathbb{X}_x\ x)[x/u]$ is not.

Lemma 4.2.10. *If $t \to^*_{\lambda\mathtt{j}} t'$ then $\mathtt{fmvar}(t') \subseteq \mathtt{fmvar}(t)$.*

Proof. By induction of the rules of $\lambda\mathtt{j}$. □

Lemma 4.2.10 will be used implicitly for cases where we have $t[y/u] \to^*_{\lambda\mathtt{j}} t'[y/u]$ with $x \notin \mathtt{fmvar}(t)$ and want to perform the rule $t'[y/u] \to_{\mathtt{d}} t'\{y/u\}$. We need to have $x \notin \mathtt{fmvar}(t')$, which is true by the lemma.

Lemma 4.2.11. *Let t be a metaterm and x a variable such that $x \notin \mathtt{fmvar}(t)$ Then, several properties from $\lambda\mathtt{j}$ on simple terms hold:*

1. *If $t \to^*_{\lambda\mathtt{j}} t'$, $u \to^*_{\lambda\mathtt{j}} u'$ then $t\{x/u\} \to^*_{\lambda\mathtt{j}} t'\{x/u'\}$. In particular, if $t \equiv t'$ and $u \equiv u'$ then $t\{x/u\} \equiv t'\{x/u'\}$.*

2. *Let t, u be metaterms in $\mathtt{j}$-normal form. Then $t\{x/u\}$ is also in $\mathtt{j}$-normal form.*

Lemma 4.2.12. *If $\downarrow((\lambda x.v)\mathtt{L}\ u) = (\lambda x.v')\mathtt{L}'\ u'$ then $\downarrow(v[x/u]\mathtt{L}) = \downarrow(v'[x/u']\mathtt{L}')$*

Proof. We first notice that u' is the normal form of u. Then we can notice that v' in $(\lambda x.v')\mathtt{L}'u'$ is the normal form of v with substitutions and renamings only determined by $\mathtt{L}$. In the same way, $\mathtt{L}'$ is determined by $\mathtt{L}$ and the number of occurrences of variables it binds in v. To conclude, we remark that the normals forms of the subterms of $\downarrow(v[x/u]\mathtt{L})$ are determined in the same way. □

4.2.2 Non-deterministic Replacement

We state several properties dealing with the most subtle aspect of $\lambda\mathtt{j}$, the non-deterministic replacement. The aim is to fully expand substitutions with respect to the rules of $\mathtt{j}$, thus obtaining a term which is renaming-independent. In this section we will thus characterise $\downarrow(t[x/u])$ (with t, u in $\mathtt{j}$-normal form) as $t_{x\rightsquigarrow y_1,...,y_n}[y_1/u]...[y_n/u]\{x/u\}$ where this last term is t where all free occurences of x in a metavariable have been renamed into different fresh variables $y_1, ..., y_n$ with an explicit substitution introduced for every y_i, anf finally an implicit substitution to deal with occurences of x not in metavariables. The main goal in this section will be to show that if $t \to^*_{\lambda\mathtt{j}} t'$ and $u \to^*_{\lambda\mathtt{j}} u'$ then $t_{x\rightsquigarrow y_1,...,y_n}[y_1/u]...[y_n/u] \to^*_{\lambda\mathtt{j}} t'_{x\rightsquigarrow z_1,...,z_m}[y_1/u']...[y_n/u']$. This is necessary to show that the simultaneous reduction is included in $\to_{\lambda\mathtt{j}}$ (point 1 of Lemma 4.1.6).

All lemmas are completely independent from the confluence section and could be used for other studies of $\lambda\mathtt{j}$.

Definition 4.2.13 (Total non-deterministic renamings on metavariables)**.** *Let t be a metaterm containing exactly n free occurrences of the variable x in the decorations of its metavariables. Let ρ be an* injective *function from the n positions of these metavariables in t to a set of fresh variables $\{y_1, ..., y_n\}$. Then $t_{\rho:x\rightsquigarrow y_1,...,y_n}$ denotes the meta-term t where each occurrence of x in a metavariable of t at position p is replaced by the variable $\rho(p)$. All the other occurrences of x which do not appear in metavariables of t are not modified. The set $\{y_1, ..., y_n\}$ is the image of ρ, written $\mathtt{img}(\rho)$. Notice that the operation ρ is deterministic.*

We also need a variant, which still renames all occurrences of x in metavariables, but which is not injective: it is denoted by $t_{\rho:x\rightarrowtail y_1,...,y_n}$.

A renaming $\rho : x$ is valid w.r.t. a term t if for every position defined by the mapping there is a variable x in a metavariable in t.

In the sequel, we will only consider valid renamings w.r.t. the terms in which they are applied. We will simply write $t_{\rho:x}$ instead of $t_{\rho:x\rightsquigarrow y_1,...,y_n}$ or $t_{\rho:x\rightarrowtail y_1,...,y_n}$ when names of fresh variables are irrelevant or clear from the context. The metaterm $t_{\rho:x\rightsquigarrow y_1,...,y_n}[y_1/u]...[y_n/u]$ is abbreviated as $t_{\rho:x,u}$. If $\|t\|_x = 0$ then $t_{\rho:x,u}$ should be read as t.

For instance, the metaterm $\mathbb{Z}_x[y/\mathbb{X}_x]_{\rho:x\rightsquigarrow y_1,y_2}$ where ρ maps the occurrence of x at the position 0 to y_1 and the occurrence of x at the position 1 to y_2 is exactly $\mathbb{Z}_{y_1}[y/\mathbb{X}_{y_2}]$. Taking the same ρ, the metaterm $\mathbb{Z}_x[z/\mathbb{X}_x]_{\rho:x,u}$ is $\mathbb{Z}_{y_1}[z/\mathbb{X}_{y_2}][y_1/u][y_2/u]$.

The metaterm $t = (x\ \mathbb{X}_x)\ \mathbb{X}_x$ affected by the non injective renaming ρ associating all occurrences of x in metavariables of t to y is: $((x\ \mathbb{X}_x)\ \mathbb{X}_x)_{\rho:x\rightarrowtail y} = (x\ \mathbb{X}_y)\ \mathbb{X}_y$.

The notation $t_{\rho:x,u}$ reflects our methodology: rename all occurrences of x in metavariables to benefit from $\equiv$ which makes the renaming ρ irrelevant. Non injective renamings will be used when talking about metaterms in which renamed variables are not bound by independent, and thus permutable, substitutions.

Lemma 4.2.14. *The following assertions are true regardless the injectivity of the renamings:*

- $\forall\rho, \exists!\rho_v, \rho_w$ *s.t.* $(v\ w)_{\rho:x} = v_{\rho_v:x}\ v_{\rho_w:x}$
- $\forall\rho_v, \rho_w, \exists!\rho$ *s.t.* $(v\ w)_{\rho:x} = v_{\rho_v:x}\ w_{\rho_w:x}$.
- $\forall\rho, \exists!\rho_v$ *s.t.* $(\lambda y.v)_{\rho:x} = \lambda y.v_{\rho_v:x}$
- $\forall\rho_v, \exists!\rho$ *s.t.* $(\lambda y.v)_{\rho:x} = \lambda y.v_{\rho_v:x}$
- $\forall\rho, \exists!\rho_v, \rho_w$ *s.t.* $v[y/w]_{\rho:x} = v_{\rho_v:x}[y/w_{\rho_w:x}]$
- $\forall\rho_v, \rho_w, \exists!\rho$ *s.t.* $v[y/w]_{\rho:x} = v_{\rho_v:x}[y/w_{\rho_w:x}]$
- $\forall\rho, \exists!\rho_v, \rho_w$ *s.t.* $v\{y/w\}_{\rho:x} = v_{\rho_v:x}\{y/w_{\rho_w:x}\}$ *if* $|v|_y = 1$
- $\forall\rho_v, \rho_w, \exists!\rho$ *s.t.* $v\{y/w\}_{\rho:x} = v_{\rho_v:x}\{y/w_{\rho_w:x}\}$ *if* $|v|_y = 1$

Furthermore we have $\mathtt{img}(\rho) = \mathtt{img}(\rho_v) \cup \mathtt{img}(\rho_w)$ *for all cases except the abstraction case where* $\mathtt{img}(\rho) = \mathtt{img}(\rho_v)$.

Proof. We only show the first case as the others are similar. For each mapping like $\rho(0p) = y_i$ (resp. $\rho(1p) = y_i$) with p a position, we construct ρ_v (resp. ρ_w) like $\rho_v(p) = y_i$ (resp. $\rho_w(p) = y_i$). □

For instance, consider the metaterm $\mathbb{Z}_x[y/\mathbb{X}_x]_{\rho:x\rightsquigarrow y_1,y_2}$ with a renaming ρ associating 0 to y_1 and 1 to y_2, then there exists two renamings ρ_1 and ρ_2 (which respectively associate 0 to y_1 and 0 to y_2) such that $\mathbb{Z}_x[y/\mathbb{X}_x]_{\rho:x\rightsquigarrow y_1,y_2} = \mathbb{Z}_{y_1}[y/\mathbb{X}_{y_2}] = \mathbb{Z}_{x_{\rho_1:x\rightsquigarrow y_1}}[y/\mathbb{X}_{x_{\rho_2:x\rightsquigarrow y_2}}]$.

Lemma 4.2.15. *For any metaterm t and renaming ρ, there exist unique $\rho', z_1, ..., z_m$ such that* $\downarrow(t_{\rho:x\rightarrowtail y_1,...,y_n}) = \downarrow(t)_{\rho':x\rightarrowtail z_1,...z_m}$ *with* $\mathtt{img}(\rho') \subseteq \mathtt{img}(\rho)$.

Proof. By induction on t using Lemma 4.2.14. □

Example 4.2.16. *Consider the metaterm $(z\ z)[z/\mathbb{X}_x]$ with the particular case where the renaming ρ is injective such that* $(z\ z)[z/\mathbb{X}_x]_{\rho:x\rightarrowtail y} = (z\ z)[z/\mathbb{X}_y]$. *Thus* $\downarrow((z\ z)[z/\mathbb{X}_y]) = (z_1\ z_2)[z/\mathbb{X}_y][z_2/\mathbb{X}_y] = (z_1\ z_2)[z/\mathbb{X}_x][z_2/\mathbb{X}_x]_{\rho':x\rightarrowtail y} = \downarrow((z\ z)[z/\mathbb{X}_x])_{\rho':x\rightarrowtail y}$ *with ρ' being the renaming associating to each occurrence of x the variable y.*

We can now state a technical property which let us deal with the non-deterministic replacement in a "deterministic" way.

Property 4.2.17. *Let t, u be metaterms in* $\mathtt{j}$ *normal form. Then, for any ρ we have* $\downarrow(t[x/u]) \equiv t_{\rho:x,u}\{x/u\}$ *and thus for any ρ, ρ',* $t_{\rho:x,u} \equiv t_{\rho':x,u}$.

Proof. If $|t|_x = 0$ or $|t|_x = 1$ then this is straightforward. Otherwise, $t[x/u]$ reduces in many $\mathtt{c}$-steps to $t_{\rho:x\rightsquigarrow y_1...y_n}[x/u][y_1/u]...[y_n/u]$ for any ρ. We can apply Full Composition and obtain $t_{\rho:x,u}\{x/u\}$ which is in $\mathtt{j}$-normal form since for every y_i $\|t\|_{y_i} = 1$ and because implicit substitution preserves $\mathtt{j}$-normal form by Lemma 4.2.11:2. □

This property implies that the non-determinism is irrelevant thanks to substitutions permutation. We can easily generalize to the case where subterms are not in j-normal form:

Corollary 4.2.18. *Let t, u be metaterms. Then, for any ρ we have $\downarrow(t[x/u]) \equiv \downarrow(t)_{\rho:x,\downarrow(u)}\{x/\downarrow(u)\}$ and thus for any ρ, ρ', $\downarrow(t)_{\rho:x,\downarrow(u)} \equiv \downarrow(t)_{\rho':x,\downarrow(u)}$.*

Lemma 4.2.19. *If $t \equiv t'$ then for any ρ, there exists a unique ρ' s.t. $t_{\rho:x} \equiv t'_{\rho':x}$*

Proof. We first show that if $t \equiv t'$ then for any ρ, there exists a unique ρ' such that $t_{\rho:x} \equiv t'_{\rho':x}$. If $t = v[y/w][z/u] \equiv v[z/u][y/w] = t'$, we have unique ρ_v, ρ_u and ρ_w s.t $t_{\rho:x} =_{L.\ 4.2.14} v_{\rho_v:x}[y/w_{\rho_w:x}][z/u_{\rho_u:x}] \equiv v_{\rho_v:x}[z/u_{\rho_u:x}][y/w_{\rho_w:x}] =_{L.\ 4.2.14} t'_{\rho':x}$. The inductive cases are straightforward. The general case when $t \equiv t'$ is done by induction. □

Notice that the Lemma 4.2.19 is not true for any ρ'. For instance, take the reflexive case where $t = t' = \mathbb{X}_x\ \mathbb{X}_x$. Two different assignments for t are $\mathbb{X}_{x_1}\ \mathbb{X}_{x_2}$ and $\mathbb{X}_{x_2}\ \mathbb{X}_{x_1}$, but they are not equivalent modulo $\equiv$.

Lemma 4.2.20. *If $u \to^*_{\lambda\mathtt{j}} u'$ then $t_{\rho:x,u} \to^*_{\lambda\mathtt{j}} t_{\rho:x,u'}$. In particular, if $u \equiv u'$ then $t_{\rho:x,u} \equiv t_{\rho:x,u'}$.*

Proof. Straightforward. □

Lemma 4.2.21. *If $t \equiv t'$, $u \equiv u'$ then for any ρ, ρ', $t_{\rho:x,u} \equiv t'_{\rho':x,u'}$. Furthermore, $t_{\rho:x,u}\{x/u\} \equiv t'_{\rho':x,u'}\{x/u'\}$.*

Proof. The second part is obvious from the first by Lemma 4.2.11:1. For the first, by Lemma 4.2.19, there exists a unique ρ'' s.t. $t_{\rho:x} \equiv t'_{\rho'':x}$. As $\equiv$ is closed by context, we have $t_{\rho:x,u} \equiv t'_{\rho'':x,u}$ and $t'_{\rho'':x,u} \equiv t'_{\rho'':x,u'}$ by Lemma 4.2.20. Property 4.2.17 concludes with $t'_{\rho'':x,u'} \equiv t'_{\rho':x,u'}$. □

We illustrate the first case by simply considering $u = u'$. Take for instance $t = v[z_1/\mathbb{X}_x][z_2/\mathbb{Z}_x]$ and consider the renamed metaterm $t_{\rho:x} = v[z_1/\mathbb{X}_{x_1}][z_2/\mathbb{Z}_{x_2}]$. Then for $t' = v[z_2/\mathbb{Z}_x][z_1/\mathbb{X}_x]$ we have:

- if $t_{\rho':x} = v[z_2/\mathbb{Z}_{x_1}][z_1/\mathbb{X}_{x_2}]$ then

$$\begin{array}{ll} t_{\rho:x,u} & = \\ v[z_1/\mathbb{X}_{x_1}][z_2/\mathbb{Z}_{x_2}][x_1/u][x_2/u] & \equiv \\ v[z_2/\mathbb{Z}_{x_2}][z_1/\mathbb{X}_{x_1}][x_1/u][x_2/u] & \equiv \\ v[z_2/\mathbb{Z}_{x_2}][z_1/\mathbb{X}_{x_1}][x_2/u][x_1/u] & =_\alpha \\ v[z_2/\mathbb{Z}_{x_1}][z_1/\mathbb{X}_{x_2}][x_1/u][x_2/u] & = t'_{\rho':x,u} \end{array}$$

- if $t'_{\rho'':x} = v[z_2/\mathbb{Z}_{x_2}][z_1/\mathbb{X}_{x_1}]$ then

$$\begin{array}{ll} t_{\rho:x,u} & = \\ v[z_1/\mathbb{X}_{x_1}][z_2/\mathbb{Z}_{x_2}][x_1/u][x_2/u] & \equiv \\ v[z_2/\mathbb{Z}_{x_2}][z_1/\mathbb{X}_{x_1}][x_1/u][x_2/u] & = t'_{\rho'':x}u \end{array}$$

Lemma 4.2.22. *If $x \notin \mathtt{fmvar}(t)$ then $\downarrow(t\{x/u\}) \equiv \downarrow(t)\{x/\downarrow(u)\}$*

Proof. By induction on t. All cases are straightforward except the substitution case. Let $t = v[y/w]$. By α-conversion $y \notin \mathtt{fv}(u)$.

$$
\begin{array}{ll}
\downarrow(v[y/w]\{x/u\}) & = \\
\downarrow(v\{x/u\}[y/w\{x/u\}]) & \equiv_{C.\ 4.2.18} \\
\downarrow(v\{x/u\})_{\rho:y,\downarrow(w\{x/u\})}\{y/\downarrow(w\{x/u\})\} & \equiv_{i.h.} \\
(\downarrow(v)\{x/\downarrow(u)\})_{\rho:y,\downarrow(w)\{x/\downarrow(u)\}}\{y/\downarrow(w)\{x/\downarrow(u)\}\} & =_{(\alpha)} \\
(\downarrow(v))_{\rho:y,\downarrow(w)}\{x/\downarrow(u)\}\{y/\downarrow(w)\{x/\downarrow(u)\}\} & =_{L.4.2.5} \\
\downarrow(v)_{\rho':y,\downarrow(w)}\{y/\downarrow(w)\}\{x/\downarrow(u)\} & \equiv_{C.\ 4.2.18} \\
\downarrow(v[y/w])\{x/\downarrow(u)\} &
\end{array}
$$

□

The next technical lemma will be useful in the next section (Lemma 4.2.35).

Lemma 4.2.23. *Let t be a metaterm and x a variable. If $t \to^*_{\lambda\mathtt{j}} t'$ and $u \to^*_{\lambda\mathtt{j}} u'$ then for any ρ, ρ', $t_{\rho:x,u} \to^*_{\lambda\mathtt{j}} t'_{\rho':x,u'}$.*

Proof. We reason by induction on the length of the reduction from t to t'.

If the length is zero then $t \equiv t'$. By Lemma 4.2.21, taking $u = u'$, we have $t_{\rho:x,u} \equiv t'_{\rho':x,u}$. By Lemma 4.2.20 $t'_{\rho':x,u} \to^*_{\lambda\mathtt{j}} t'_{\rho':x,u'}$ which concludes this subcase.

Otherwise, $t \to^*_{\lambda\mathtt{j}} t_1 \to_{\lambda\mathtt{j}} t'$. By induction hypothesis, for any ρ, there exists a ρ_1 s.t. $t_{\rho:x,u} \to^*_{\lambda\mathtt{j}} t_{1\rho_1:x,u'}$. For the last step, we have to show that for any ρ_1, if $t_1 \to_{\lambda\mathtt{j}} t'$ there exists a ρ' s.t. $t_{1\rho_1:x,u} \to^*_{\lambda\mathtt{j}} t'_{\rho':x,u}$. It is actually sufficient to show that for any ρ_1, ρ', if $t_1 \to_{\lambda\mathtt{j}} t'$ then $t_{1\rho_1:x,u} \to^*_{\lambda\mathtt{j}} t'_{\rho':x,u}$. We reason by cases on $t_1 \to_{\lambda\mathtt{j}} t'$:

- $t_1 = (\lambda y.v)\mathtt{L}\ w \to_{\mathtt{dB}} v[y/w]\mathtt{L} = t'$

$$
\begin{array}{ll}
t_{1\rho:x,u} & = \\
((\lambda y.v)\mathtt{L}\ w)_{\rho_1:x\rightsquigarrow\overline{z}}\overline{[z/u]} & =_{L.\ 4.2.14} \\
((\lambda y.v_{\rho_v:x})\mathtt{L}_{\rho_L:x}\ w_{\rho_w:x})\overline{[z/u]} & \to_{\mathtt{dB}} \\
v_{\rho_v:x}[y/w_{\rho_w:x}]\mathtt{L}_{\rho_L:x}\overline{[z/u]} & =_{L.\ 4.2.14} \\
(v[y/w]\mathtt{L})_{\rho'':x\rightsquigarrow\overline{z}}\overline{[z/u]} & \equiv_{C.4.2.18} t'_{\rho':x,u}
\end{array}
$$

- $t_1 = v[y/w] \to_{\mathtt{d}} v\{y/w\} = t'$ As y appears exactly once in v, we have the same number of z_i in t_1 and in t'. Notice that we can apply the rule $\to_{\mathtt{d}}$ since the multiplicity of y cannot be changed by a renaming of x.

$$
\begin{array}{ll}
t_{1\rho:x,u} & = \\
(v[y/w])_{\rho_1:x\rightsquigarrow\overline{z}}\overline{[z/u]} & =_{L.\ 4.2.14} \\
v_{\rho_v:x}[y/w_{\rho_w:x}]\overline{[z/u]} & \to_{\mathtt{d}} \\
v_{\rho_v:x}\{y/w_{\rho_w:x}\}\overline{[z/u]} & =_{L.\ 4.2.14} \\
(v\{y/w\})_{\rho'':x}\overline{[z/u]} & \equiv_{C.4.2.18} t'_{\rho':x,u}
\end{array}
$$

- $t_1 = v[y/u] \to_{\mathtt{w}} v = t'$ with $y \notin \mathtt{fv}(v)$

$$
\begin{array}{ll}
{t_1}_{\rho:x,u} & = \\
(v[y/u])_{\rho_1:x\rightsquigarrow\overline{z}}[z_1/u]...[z_n/u] & =_{L.\ 4.2.14} \\
v_{\rho_v:x\rightsquigarrow z'_1...z'_m}[y/u_{\rho_u:x}][z_1/u]...[z_n/u] & \to_{\mathtt{w}} \\
\text{ with } \{z'_1, ..., z'_m\} \text{ a subset of } \{z_1, ..., z_n\} & \\
v_{\rho_v:x\rightsquigarrow z'_1...z'_m}[z_1/u]...[z_n/u] & \to^*_{\mathtt{w}} \\
v_{\rho_v:x\rightsquigarrow z'_1,...,z'_m}[z'_1/u]...[z'_m/u] & \equiv_{C.4.2.18} {t'}_{\rho_v:x,u}
\end{array}
$$

- $t_1 = v[y/w] \to_{\mathtt{c}} v_{y\rightsquigarrow z}[y/w][z/w] = t'$ with $n = |v|_y > 1$

$$
\begin{array}{ll}
{t_1}_{\rho_1:x,u} & = \\
(v[y/w])_{\rho_1:x\rightsquigarrow z_1,...,z_n}[z_1/u]...[z_n/u] & =_{L.\ 4.2.14} \\
v_{\rho_v:x}[y/w_{\rho_w:x}][z_1/u]...[z_n/u] & \to_{\mathtt{c}} \\
(v_{y\rightsquigarrow z})_{\rho_v:x}[y/w_{\rho_w:x}][z/w_{\rho_w:x}][z_1/u]...[z_n/u] & = \\
(v_{y\rightsquigarrow z})_{\rho_v:x}\ [y/w_{\rho_w:x\rightsquigarrow z'_1...z'_m}] & \\
\qquad [z/w_{\rho_w:x\rightsquigarrow z'_1...z'_m}] & \equiv \\
\qquad [z_1/u]...[z_n/u] & \\
\text{(with } \{z'_1...z'_m\} \text{ a subset of } \{z_1...z_n\} \text{ s.t.} & \\
\{z_1, ..., z_n\} = \{z'_1, ..., z'_m\} \cup \{z'_{m+1}, ..., z'_n\}) & \\
(v_{y\rightsquigarrow z})_{\rho_v:x}[y/w_{\rho_w:x\rightsquigarrow z'_1...z'_m}]\ [z/w_{\rho_w:x\rightsquigarrow z'_1...z'_m}] & \\
\qquad [z'_1/u]...[z'_m/u] & \to^*_{\mathtt{c}} \\
\qquad [z'_{m+1}/u]...[z'_n/u] & \\
(v_{y\rightsquigarrow z})_{\rho_v:x}[y/w_{\rho_w:x\rightsquigarrow z'_1...z'_m}]\ [z/w_{\rho_w:x\rightsquigarrow z''_1...z''_m}] & \\
\qquad [z'_1/u]...[z'_m/u] & \\
\qquad [z''_1/u]...[z''_m/u] & =_{L.\ 4.2.14} \\
\qquad [z'_{m+1}/u]...[z'_n/u] & \\
(v_{y\rightsquigarrow z}[y/w][z/w])_{\rho':x}\ [z'_1/u]...[z'_m/u] & \\
\qquad [z''_1/u]...[z''_m/u] & \equiv_{C.4.2.18} {t'}_{\rho':x,u} \\
\qquad [z'_{m+1}/u]...[z'_n/u] &
\end{array}
$$

□

4.2.3 Confluence on Metaterms

We combine the interpretation method with the idea of simultaneous reduction defined in Tait-Martin Löf's technique.

We first isolate the difficulties thanks to the interpretation method and reduce the confluence of $\lambda\mathtt{j}$ to the confluence of a system defined on $\mathtt{j}$-normal forms, called the interpretation calculus. There are several possibilities to define this latter one leading to different proofs. We follow ideas from Tait-Martin Löf and choose a calculus reducing simultaneously metaterms.

The second part of the proof is to finally show that the interpretation calculus enjoys the diamond property, which implies that it is confluent. This finally implies that $\lambda\mathtt{j}$ is confluent.

Following the interpretation method given in Lemma 4.1.6 by taking $R = \lambda\mathtt{j}$, $R_1 = \mathtt{j}$, $R_2 = \mathtt{B}$, and $\approx = \equiv$ we thus have to find a relation $\Rrightarrow$, prove Points 1 and 3 since Point 2 was proved by Lemma 4.2.8, and of course, show that $\Rrightarrow$ is confluent.

If we had wanted to show confluence on terms and not on metaterms we would be in the situation where normal forms are actually lambda terms. We could thus easily conclude by using the fact that the λ-calculus is confluent.

In the rest of this section we will this:

1. Following Tait-Martin Löf, define $\Rrightarrow$ as a simultaneous reduction wich mimics β reductions by taking the $\mathtt{j}$-normal form after each $\mathtt{B}$ step.

2. Prove Points 1 and 3 of Lemma 4.1.6 [2]

3. Prove confluence of $\Rrightarrow$ by showing that it enjoys the diamond property, that is any critical pair can be closed in one step.

There are several ways to define the simultaneous reduction relation. We choose this high-level definition, which let us use all the properties we have proved in Section 4.2.2.

Definition 4.2.24 (Simultaneous reduction)**.** *We define the simultaneous reduction $\Rrightarrow$ on $\mathtt{j}$-normal forms as follows:*

- $x \Rrightarrow x$
- $\mathbb{X} \Rrightarrow \mathbb{X}$
- *If* $u \Rrightarrow u'$ *then* $\lambda x.u \Rrightarrow \lambda x.u'$
- *If* $u \Rrightarrow u'$ *and* $v \Rrightarrow v'$ *then* $u\ v \Rrightarrow u'\ v'$
- *If* $t \Rrightarrow t'$, $u \Rrightarrow u'$ *for any injective* ρ, ρ' , $t_{\rho:x,u} \Rrightarrow t'_{\rho':x,u'}$.
- *If* $v \Rrightarrow v'$, $u \Rrightarrow u'$, $L \Rrightarrow L'$ $(\lambda x.v)\mathtt{L}\ u \Rrightarrow \downarrow(v'[x/u']\mathtt{L}')$

The fifth and most subtle rule is valid since ρ and ρ' are injective total renamings.

Example 4.2.25. *For instance,* $(\lambda x.(\mathbb{X}_x\ x)(\mathbb{X}_x\ y))[y/z]\ u$ *simultaneously reduces to* $((\mathbb{X}_{x1}\ u)\ (\mathbb{X}_{x2}\ z))[x_1/u][x_2/u]$ *or to* $((\mathbb{X}_{x2}\ u)\ (\mathbb{X}_{x1}\ z))[x_1/u][x_2/u]$. *The two terms are equivalent modulo* $\equiv$.

The following example illustrates the use of the fifth rule. Let $t = (\lambda y.y\ y)\ \mathbb{X}_x \Rrightarrow \mathbb{X}_x\ \mathbb{X}_x = t'$. *There is only one valid renaming* ρ *for* t *which associates the variable* z *to the unique occurrence of* x. *Then* $t_{\rho:x,u} = ((\lambda y.y\ y)\ \mathbb{X}_z)[z/u]$ *simultaneously reduces to* $(\mathbb{X}_{z_1}\ \mathbb{X}_{z_2})[z_1/u][z_2/u]$ *or to* $(\mathbb{X}_{z_2}\ \mathbb{X}_{z_1})[z_1/u][z_2/u]$. *Notice that those two metaterms are equivalent up to* $\equiv$.

[2] instead of showing $\rightarrow \subseteq \Rrightarrow \subseteq \rightarrow^*$ as done in the Tait-Martin Löf's technique since it is not true in our case

The difficulty of the proof relies on the fact that $\lambda\mathtt{j}$ combines non-deterministic replacement and action at distance. Indeed, in all calculi defined by a case analysis on the constructor at the left of the substitution, it is possible to have a rule $\mathbb{X}_\Delta[x_1/u_1]...[x_n/u_n] \Rrightarrow \mathbb{X}_\Delta[x_1/u'_1]...[x_n/u'_n]$ with $x_i \notin \mathtt{fv}(u_j)$ and $u_i \Rrightarrow u'_i$, instead of the complex $t_{\rho:x,u} \Rrightarrow {t'}_{\rho':x,u'}$.

As a normal form may contain jumps we need to handle equations in the simultaneous reductions. To do so, we introduce a new reduction:

Definition 4.2.26 (Simultaneous reduction modulo)**.** *The simultaneous reduction modulo $t \Rrightarrow_\equiv t'$ occurs if there exists t_1, t_2 s.t. $t \equiv t_1 \Rrightarrow t_2 \equiv t'$.*

Lemma 4.2.27. *If $t \Rrightarrow t'$, $u \Rrightarrow u'$, $x \notin \mathtt{fmvar}(t)$ then $t\{x/u\} \Rrightarrow_\equiv t'\{x/u'\}$.*

Proof. By induction on $t \Rrightarrow t'$. We only consider the interesting cases:

- ${v_{\rho:y,w}} \Rrightarrow {v'}_{\rho':y,w'}$ coming from $v \Rrightarrow v'$ and $w \Rrightarrow w'$. By α-conversion $x \neq y$ and $y \notin \mathtt{fv}(u)$. By induction hypothesis, $v\{x/u\} \Rrightarrow_\equiv v'\{x/u'\}$ and $w\{x/u\} \Rrightarrow_\equiv w'\{x/u'\}$. By definition of the implicit substitution, $t\{x/u\} = v\{x/u\}_{\rho:y,w\{x/u\}}$ and by definition of $\Rrightarrow$, $v\{x/u\}_{\rho:y,w\{x/u\}} \Rrightarrow_\equiv v'\{x/u'\}_{\rho:y,w'\{x/u'\}}$ which is equal to $t'\{x/u'\}$.
- $(\lambda y.v)L\ w \Rrightarrow \downarrow(v'[y/w']L')$ coming from $v \Rrightarrow v'$, $\mathtt{L} \Rrightarrow \mathtt{L}'$, and $w \Rrightarrow w'$. By α-conversion $x \neq y$. By induction hypothesis, $v\{x/u\} \Rrightarrow_\equiv v'\{x/u'\}$, $w\{x/u\} \Rrightarrow_\equiv w'\{x/u'\}$ and $\mathtt{L}\{x/u\} \Rrightarrow_\equiv \mathtt{L}'\{x/u'\}$.

$$
\begin{array}{ll}
t\{x/u\} & = \\
(\lambda y.v\{x/u\})\mathtt{L}\{x/u\}\ w\{x/u\} & \Rrightarrow_\equiv \\
\downarrow(v'\{x/u'\}[y/w'\{x/u'\}](\mathtt{L}'\{x/u'\})) & = \\
\downarrow((v'[y/w']\mathtt{L})\{x/u'\}) & \equiv_{L.\ 4.2.22} \\
\downarrow(v'[y/w']\mathtt{L})\{x/\downarrow(u')\} & =_{(u' \in \mathcal{NF}_\mathtt{j})} \\
\downarrow(v'[y/w']L')\{x/u'\} &
\end{array}
$$

□

Lemma 4.2.28. *If $t \Rrightarrow t'$ and $u \Rrightarrow u'$ then $\downarrow(t[x/u]) \Rrightarrow_\equiv \downarrow(t'[x/u'])$.*

Proof. By hypothesis, both t and u are in $\mathtt{j}$-normal form. Then,

$$
\begin{array}{ll}
\downarrow(t[x/u]) & \equiv_{P.4.2.17} \\
t_{\rho:x,u}\{x/u\} & \Rrightarrow_\equiv (\text{hyp, def and L. } 4.2.27) \\
{t'}_{\rho':x,u'}\{x/u'\} & \equiv_{P.4.2.17} \\
\downarrow(t'[x/u']) &
\end{array}
$$

□

It is thanks to our definition of simultaneous reduction that this proof is short. Otherwise, a proof by induction on $\Rrightarrow$ is needed, and it can take several pages as in [Kes07]. We can generalize the previous lemma to the case where the terms t and t' are affected by a list of substitutions:

Lemma 4.2.29. *If $t \Rrightarrow t'$, $\mathtt{L} \Rrightarrow \mathtt{L}'$ and $u \Rrightarrow u'$ then $\downarrow(t[x/u]\mathtt{L}) \Rrightarrow_{\equiv} \downarrow(t'[x/u']\mathtt{L}')$.*

Proof. By hypothesis, both t, u and $\mathtt{L}$ are in $\mathtt{j}$-normal form. We proceed by induction on the length of $\mathtt{L}$. If the length is equal to 0 we then fall in Lemma 4.2.28. Otherwise,

$$
\begin{array}{ll}
\downarrow(t[x/u]\mathtt{L}) & = \\
\downarrow(t[x/u][y_1/w_1]...[y_n/w_n]) & \equiv_{P.4.2.17} \\
\downarrow(t[x/u][y_1/w_1]...[y_{n-1}/w_{n-1}])_{\rho:y_n,w_n}\{y_n/w_n\} & \Rrightarrow_{\equiv \text{i.h., def and L. 4.2.27}} \\
\downarrow(t'[x/u'][y_1/w'_1]...[y_{n-1}/w'_{n-1}])_{\rho':y_n,w'_n}\{y_n/w'_n\} & \equiv_{P.4.2.17} \\
\downarrow(t'[x/u'][y_1/w'_1]...[y_n/w'_n]) & = \\
\downarrow(t'[x/u']\mathtt{L}') &
\end{array}
$$

□

This technical lemma will be useful to show an admissible rule necessary to show the diamond property. It is the only one allowing to go from an injective renaming to a non injective one and vice-versa.

Lemma 4.2.30. *If $t \Rrightarrow t'$ then for any injective ρ there exists a unique (possibly non injective) ρ' and variables $z_1, ..., z_m$ such that:*

1. $t_{\rho:x\rightsquigarrow y_1...y_n} \Rrightarrow {t'}_{\rho':x\rightarrowtail z_1...z_m}$ *with* $\mathtt{img}(\rho') \subseteq \mathtt{img}(\rho)$.
2. $\downarrow({t'}_{\rho':x\rightarrowtail z_1...z_m}[y_1/u_1]...[y_n/u_n]) = {t'}_{\rho_2:x\rightsquigarrow z'_1...z'_k}[z'_1/u_1]...[z'_k/u_n]$ *for any injective ρ_2 and $u_i \in \mathcal{NF}_{\mathtt{j}}$.*

Proof. 1. By induction on $t \Rrightarrow t'$.

- The case $t = x \Rrightarrow x = t'$ is straightforward.
- The case $t = \mathbb{X}_\Delta \Rrightarrow \mathbb{X}_\Delta = t'$ is only interesting when $x \in \mathtt{fv}(\Delta)$. Then there is only one renaming possible since x appears only once and thus taking $z_1 = y$ we have $\mathbb{X}_{\Delta x\rightsquigarrow y} \Rrightarrow \mathbb{X}_{\Delta x\rightarrowtail y}$. Furthermore, $\downarrow(\mathbb{X}_{\Delta x\rightsquigarrow y}[y/u]) = \mathbb{X}_{\Delta x\rightsquigarrow y}[y/u]$, since by hypothesis y is a fresh variable and thus cannot appear in Δ which is equal to ${t'}_{\rho_2:x,u}$ with ρ_2 being the unique injective renaming possible in t'.
- $t = v\ w \Rrightarrow v'\ w' = t'$.

  $(vw)_{\rho:x\rightsquigarrow y_1...y_n} =_{L.\ 4.2.14} v_{\rho_v:x\rightsquigarrow y'_1...y'_k}\ w_{\rho_w:x\rightsquigarrow y''_1...y''_p}$

  By induction hypothesis,

  $v_{\rho:x\rightsquigarrow y'_1...y'_n} \Rrightarrow v'_{\rho'_v:x\rightsquigarrow z'_1...z'_m}$ and $w_{\rho:x\rightsquigarrow y''_1...y''_p} \Rrightarrow w'_{\rho'_w:x\rightsquigarrow z''_1...z''_q}$ with $\mathtt{img}(\rho'_v) \subseteq \mathtt{img}(\rho_v)$ and $\mathtt{img}(\rho'_w) \subseteq \mathtt{img}(\rho_w)$

  Thus, $v_{\rho:x\rightsquigarrow y'_1...y'_k}\ w_{\rho:x\rightsquigarrow y''_1...y''_p} \Rrightarrow {v'}_{\rho'_v:x\rightarrowtail z'_1...z'_m}\ {w'}_{\rho'_w:x\rightarrowtail z''_1...z''_q}$ which is equal by Lemma 4.2.14 to $(v'\ w')_{\rho':x\rightarrowtail z'_1...z'_m,z''_1...z''_q}$. Furthermore, $\mathtt{img}(\rho') = \mathtt{img}(\rho'_v) \cup \mathtt{img}(\rho'_w)$ which are a subset of $\mathtt{img}(\rho_v) \cup \mathtt{img}(\rho_w)$ by hypothesis and $\mathtt{img}(\rho_v) \cup \mathtt{img}(\rho_w) = \mathtt{img}(\rho)$ by Lemma 4.2.14.

- Cases $\lambda z.v \Rrightarrow \lambda z.v'$ and $v_{\rho:x,w} \Rrightarrow v'_{\rho':x,w'}$ are similar (i.h. and Lemma 4.2.14).
- $(\lambda z.v)\mathtt{L}\ w \Rrightarrow \downarrow(v'[z/w']\mathtt{L}')$.

$$
\begin{array}{ll}
((\lambda z.v)\mathtt{L}\ w) & =_{L.\ 4.2.14} \\
(\lambda z.v_{\rho_v:x\rightsquigarrow y'_1\ldots y'_\alpha})\mathtt{L}_{\rho_L:x\rightsquigarrow y'''_1\ldots y'''_\gamma}\ w_{\rho_w:x\rightsquigarrow y''_1\ldots y''_\beta} & \Rrightarrow \\
\downarrow(v'_{\rho'_v:x\rightsquigarrow z'_1\ldots z'_\delta}[z/w'_{\rho'_w:x\rightsquigarrow z''_1\ldots z''_\varepsilon}]\mathtt{L}'_{\rho'_\mathtt{L}:x\rightsquigarrow z'''_1\ldots z'''_\zeta}) & =_{L.\ 4.2.14} \\
\downarrow((v'[z/w']\mathtt{L}')_{\rho'':x\rightsquigarrow z'_1\ldots z'_\delta,z''_1\ldots z''_\varepsilon,z'''_1\ldots z'''_\zeta}) & =_{L.\ 4.2.15} \\
\downarrow(v'[z/w']\mathtt{L}')_{\rho':x\rightsquigarrow z_1\ldots z_m} & = \\
\downarrow(t')_{\rho':x\rightsquigarrow z_1\ldots z_m} &
\end{array}
$$

The fact that $\mathtt{img}(\rho') \subseteq \mathtt{img}(\rho)$ is easy using Lemmas 4.2.14 and 4.2.15.

2. By applying rules $\mathtt{w}$ and $\mathtt{c}$.

□

Example 4.2.31. *Take $t = (\lambda y.y\ y)\ \mathbb{X}_x \Rrightarrow \mathbb{X}_x\ \mathbb{X}_x = t'$ together with the unique valid renaming for t which associates the variable z to the unique occurrence of x. Then $t_{\rho:x\rightsquigarrow z} = (\lambda y.y\ y)\ \mathbb{X}_z \Rrightarrow \mathbb{X}_z\ \mathbb{X}_z = t'_{\rho':x\rightarrowtail z}$ with ρ' the renaming associating to all the occurrences of x in t' the variable z.*

To illustrate the second point of Lemma 4.2.30, take any $u \in \mathcal{NF}_\mathtt{j}$ and notice that $\downarrow((\mathbb{X}_y\ \mathbb{X}_y)[y/u])$ can reduce either to $(\mathbb{X}_{y_1}\ \mathbb{X}_{y_2})[y_1/u][y_2/u]$ or to $(\mathbb{X}_{y_2}\ \mathbb{X}_{y_1})[y_1/u][y_2/u]$ which correspond to $t'_{\rho_2:x\rightsquigarrow y_1,y_2}[y_1/u][y_2/u]$ or $t'_{\rho'_2:x\rightsquigarrow y_1,y_2}[y_1/u][y_2/u]$, ρ_2 and ρ'_2 being the two valid renamings for t'.

Lemma 4.2.32. *We have the following admissible rules:*

1. *If $t \Rrightarrow t'$, $u \Rrightarrow u'$, $\|t\|_x = 1$ then $t[x/u] \Rrightarrow \downarrow(t'[x/u'])$.*
2. *If $t \Rrightarrow t'$, $u_i \Rrightarrow u'_i$ then $t_{\rho:x\rightsquigarrow y_1,\ldots,y_n}\overline{[y_i/u_i]} \Rrightarrow t'_{\rho:x\rightsquigarrow z_1,\ldots,z_m}\overline{[z_i/u'_i]}$*

Proof. 1. Reading the fourth rule when there is only one occurrence of x gives $t[x/u] \Rrightarrow t'_{\rho':x,u'}$ which is equal to $t'_{\rho':x,u'}\{x/u\}$ since x does not appear in $t'_{\rho':x,u'}$ and it thus equal to $\downarrow(t'[x/u'])$.

2. By Lemma 4.2.30.1, $t_{\rho:x\rightsquigarrow y_1,\ldots,y_n} \Rrightarrow t'_{\rho':x\rightarrowtail z_1\ldots z_m}$. By the first point we have, $t_{\rho:x\rightsquigarrow y_1,\ldots,y_n}\overline{[y_i/u_i]} \Rrightarrow \downarrow(t'_{\rho':x\rightarrowtail z_1\ldots z_m}\overline{[y_i/u_i]})$ and by Lemma 4.2.30:2, $\downarrow(t'_{\rho':x\rightarrowtail z_1\ldots z_m}\overline{[y_i/u_i]}) = t'_{\rho:x\rightsquigarrow z_1,\ldots,z_m}\overline{[z_i/u'_i]}$

□

Lemma 4.2.33. *If $t \rightarrow_\mathtt{B} t'$ then $\downarrow(t) \Rrightarrow_\equiv \downarrow(t)'$.*

Proof. By induction on the relation $t \rightarrow_\mathtt{B} t'$. If the reduction occurs at the root of t i.e. if $t = (\lambda x.v)\mathtt{L}\ u \rightarrow_\mathtt{B} v[x/u]\mathtt{L} = t'$, notice that $\downarrow(t) = (\lambda x.v')\mathtt{L}'\ u'$. Then by Lemma 4.2.12, $\downarrow(t') = \downarrow(v'[x/u']\mathtt{L}')$ and we conclude by definition of $\Rrightarrow$.

Otherwise, for the subterm u where the $\mathtt{B}$ redex is performed we have $u \Rrightarrow u'$. We easily conclude by induction hypothesis except for the two following cases:

- $t = v[x/u] \to_{\mathtt{B}} v[x/u'] = t'$. $\downarrow(t) \equiv \downarrow(v)_{\rho:x,\downarrow(u)}\{x/\downarrow(u)\}$ with $\downarrow(u) \Rrightarrow \downarrow(u')$ by induction hypothesis. By Lemma 4.2.27 and definition of $\Rrightarrow$ we get that $\downarrow(v)_{\rho:x,\downarrow(u)}\{x/\downarrow(u)\} \Rrightarrow \downarrow(v)_{\rho:x,\downarrow(u')}\{x/\downarrow(u')\}$

- The case $t = u[x/v] \to_{\mathtt{B}} u'[x/v] = t'$ is similar to the previous one.

□

We can now deduce Point 3 of Lemma 4.1.6:

Corollary 4.2.34. *If $t \to_{\mathtt{B}} t'$ then $\downarrow(t) \Rrightarrow_{\equiv} \downarrow(t')$.*

And Point 1 of Lemma 4.1.6:

Lemma 4.2.35. *If $t \Rrightarrow_{\equiv} t'$ then $t \to^*_{\lambda\mathtt{j}} t'$.*

Proof. The case $t \equiv t'$ is straightforward. Otherwise, it is sufficient to show that $t \Rrightarrow t_1$ implies $t \to^*_{\lambda\mathtt{j}/\equiv} t_1$. Indeed if $t \equiv t_0 \Rrightarrow t'_0 \equiv t_1$ then by hypothesis, $t_0 \to^*_{\lambda\mathtt{j}} t'_0$ and thus $t \to^*_{\lambda\mathtt{j}} t_1$.

The interesting case is the following one:

- $t = v_{\rho:x,u} \Rrightarrow v'_{\rho':x,u'} = t_1$ coming from $v \Rrightarrow v'$, $u \Rrightarrow u'$. By induction hypothesis, $v \to^*_{\lambda\mathtt{j}} v'$ and $u \to^*_{\lambda\mathtt{j}} u'$. We can then conclude with Lemma 4.2.23.

□

Lemma 4.2.36. *If $t \Rrightarrow_{\equiv} t'$, $u \Rrightarrow_{\equiv} u'$ then for any ρ, ρ' $t_{\rho:x,u} \Rrightarrow_{\equiv} t'_{\rho':x,u'}$*

Proof. By hypothesis, $t \equiv t_1 \Rrightarrow t_2 \equiv t'$, $u \equiv u_1 \Rrightarrow u_2 \equiv u'$. By Lemma 4.2.21, $t_{\rho:x,u} \equiv t_{1\rho:x,u_1}$. By hypothesis and definition of $\Rrightarrow$, for any ρ, $t_{1\rho:x,u_1} \Rrightarrow t_{2\rho:x,u_2}$. Again, by Lemma 4.2.21, $t_{2\rho:x,u_2} \equiv t'_{\rho:x,u'}$ which concludes. □

Lemma 4.2.37 ($\Rrightarrow_{\equiv}$ has the diamond property)**.** *If $t_1 \Lleftarrow_{\equiv} t \Rrightarrow_{\equiv} t_2$, then there exists t_3 such that $t_1 \Rrightarrow_{\equiv} t_3 \Lleftarrow_{\equiv} t_2$.*

1. We first prove that if $t' \overset{i}{\Lleftarrow} t \equiv^1 t_1$ then it exists t'_1 s.t $t' \equiv t'_1 \overset{i}{\Lleftarrow} t_1$. We proceed by induction on $\equiv^1$. The base cases are the following ones:

   - $(u'_{\rho_x:x,v'})_{\rho_y:y,w'} \Lleftarrow u[x/v][y/w] \equiv^1 u[y/w][x/v]$ with:
     - $x \notin \mathtt{fv}(w)$ and $y \notin \mathtt{fv}(v)$
     - $\|u\|_x = \|u\|_y = 1$
     - $u \Rrightarrow u'$, $v \Rrightarrow v'$, $w \Rrightarrow w'$

Then,

$$
\begin{array}{ll}
u[y/w][x/v] & \Rrightarrow \\
({u''}_{\rho'_y:y,w''})_{\rho'_x:x,v''} & = \\
({u''}_{\rho'_y:y\rightsquigarrow b_1,\ldots,b_m}\overline{[b/w'']})_{\rho'_x:x\rightsquigarrow a_1,\ldots,a_n}\overline{[a/v'']} & =_{L.\ 4.2.14} \\
({u''}_{\rho'_y:y\rightsquigarrow b_1,\ldots,b_m})_{\rho''_x:x\rightsquigarrow a_1,\ldots,a_n}\overline{[b/w'']}\overline{[a/v'']} & \equiv \\
({u''}_{\rho'_y:y\rightsquigarrow b_1,\ldots,b_m})_{\rho''_x:x\rightsquigarrow a_1,\ldots,a_n}\overline{[a/v'']}\overline{[b/w'']} & = \\
({u''}_{\rho''_x:x\rightsquigarrow a_1,\ldots,a_n})_{\rho'_y:y\rightsquigarrow b_1,\ldots,b_m}\overline{[a/v'']}\overline{[b/w'']} & = \\
({u''}_{\rho''_x:x\rightsquigarrow a_1,\ldots,a_n})_{\rho'_y:y\rightsquigarrow b_1,\ldots,b_m}\overline{[a/v'']}\overline{[b/w'']} & =_{L.\ 4.2.14} \\
({u''}_{\rho'_x:x\rightsquigarrow a_1,\ldots,a_n}\overline{[a/v'']})_{\rho''_y:y\rightsquigarrow b_1,\ldots,b_m}\overline{[b/w'']} & \\
({u''}_{\rho''_x:x,v''})_{\rho''_y:y,w''} &
\end{array}
$$

By hypothesis, $u' \Lleftarrow u \equiv^1 u$, $v' \Lleftarrow v \equiv^1 v$, and $w' \Lleftarrow w \equiv^1 w$. Thus, by induction hypothesis, $u' \equiv u'' \Lleftarrow u$, $v' \equiv v'' \Lleftarrow v$, and $w' \equiv w'' \Lleftarrow w$. We thus have:

$$
\begin{array}{ll}
({u''}_{\rho''_x:x,v''})_{\rho''_y:y,w''} & \equiv \\
({u'}_{\rho''_x:x,v'})_{\rho''_y:y,w'} & \equiv_{P.\ 4.2.17} \\
({u'}_{\rho_x:x,v'})_{\rho_y:y,w'} &
\end{array}
$$

- ${u'}_{\rho':x,v'} \Lleftarrow u_{\rho:x,v} \equiv^1 {u_1}_{\rho_1:x,v}$. By induction hypothesis, $u' \equiv u'_1 \Lleftarrow u_1$, $v' \equiv v'_1 \Lleftarrow v$ and thus ${u'}_{\rho':x,v'} \equiv_{L.\ 4.2.21} {u'_1}_{\rho'_1:x,v'} \equiv {u'_1}_{\rho'_1:x,v'_1} \Lleftarrow {u_1}_{\rho_1:x,v}$
- $$
  \begin{array}{lcc}
  {u'}_{\rho':x,v'} & \Lleftarrow & u_{\rho:x\rightsquigarrow y_1,\ldots,y_n}\overline{[y/v]} \\
  & & \equiv^1 \\
  & & u_{\rho_1:x\rightsquigarrow y_1,\ldots,y_n}[y_1/v]\ldots[y_i/v_1]\ldots[y_n/v]
  \end{array}
  $$

  By induction hypothesis, $v' \equiv v'_1 \Lleftarrow v$ and $v' \equiv v'_1 \Lleftarrow v_1$.
  By Lemma 4.2.32, $u_{\rho_1:x\rightsquigarrow y_1,\ldots,y_n}[y_1/v]\ldots[y_i/v_1]\ldots[y_n/v] \Rrightarrow u_{\rho'_1:x,v'_1}$.
  Finally $u_{\rho'_1:x,v'_1} \equiv_{L.\ 4.2.20} u_{\rho'_1:x,v'} \equiv_{P.\ 4.2.17} {u'}_{\rho':x,v'}$.
- $\downarrow(v'[x/w']\mathtt{L}') \Lleftarrow (\lambda x.v)\mathtt{L}\ w \equiv (\lambda x.v)\mathtt{L}_1\ w$ with $\mathtt{L} \equiv^1 \mathtt{L}_1$. By induction hypothesis, $w' \equiv w'_1 \Lleftarrow w$, $v' \equiv v'_1 \Lleftarrow v$, $\mathtt{L}' \equiv \mathtt{L}'_1 \Lleftarrow \mathtt{L}_1$. We thus have to show that $\downarrow(v'[x/w']\mathtt{L}') \equiv \downarrow(v'_1[x/w'_1]\mathtt{L}'_1)$ which is true by definition of $\downarrow(\ )$.
- The cases $\downarrow(v'[x/w']\mathtt{L}') \Lleftarrow (\lambda x.v)\mathtt{L}\ w \equiv (\lambda x.v_1)\mathtt{L}\ w$ and $\downarrow(v'[x/w']\mathtt{L}') \Lleftarrow (\lambda x.v)\mathtt{L}\ w \equiv (\lambda x.v)\mathtt{L}\ w_1$ are similar to the previous case.

2. We prove $t' \Lleftarrow t \equiv t_1$ implies $t' \equiv t'_1 \Lleftarrow t_1$.

Proof. By induction on the number k of $\equiv$ steps between t and t_1. If $k = 0$, then this is straightforward. If $k = k' + 1$, we conclude with the following

diagram:

$$\begin{array}{ccccccc} t & \equiv & t_0 & \equiv & \dots & t_1 \\ & \text{Point 1} & & & i.h. & \\ \Downarrow & & \Downarrow & & & \Downarrow \\ t' & \equiv & t'_0 & \equiv & \dots & t'_1 \end{array}$$

□

3. We prove $t' \Leftarrow_{\equiv} t \equiv t_1$ implies $t' \equiv t'_1 \Leftarrow t_1$.

   Proof. If $t' \Leftarrow_{\equiv} t \equiv t_1$ then we have t_0 and t'_0 s.t. $t' \equiv t'_0 \Leftarrow t_0 \equiv t_1$. By the previous point, $t_0 \Rrightarrow t'_0$. We can conclude by applying one more time the previous point. □

4. We prove $t_1 \Leftarrow t \Rrightarrow t_2$ implies there exists t_3 such that $t_1 \Rrightarrow_{\equiv} t_3 \Leftarrow_{\equiv} t_2$.

   Proof. By induction on $\Rrightarrow$. The interesting cases are the following ones:

   - $(\lambda x.v_1)\mathtt{L}_1\ u_1 \Leftarrow (\lambda x.v)\mathtt{L}\ u \Rrightarrow (\lambda x.v_2)\mathtt{L}_2\ u_2$ with $u \Rrightarrow u_1$, $u \Rrightarrow u_2$, $v \Rrightarrow v_1$, $v \Rrightarrow v_2$. We can conclude by induction hypothesis.
   - $(\lambda x.v_1)\mathtt{L}_1\ u_1 \Leftarrow (\lambda x.v)\mathtt{L}\ u \Rrightarrow \downarrow(v_2[x/u_2]\mathtt{L}_2)$ with $u \Rrightarrow u_1$, $u \Rrightarrow u_2$, $v \Rrightarrow v_1$, $v \Rrightarrow v_2$. By induction hypothesis, $u_1 \Rrightarrow u_3 \Leftarrow u_2$ and $v_1 \Rrightarrow v_3 \Leftarrow v_2$. By hypothesis, $(\lambda x.v_1)\mathtt{L}_1\ u_1 \Rrightarrow \downarrow(v_3[x/u_3]\mathtt{L}_3)$ and by Lemma 4.2.29 $\downarrow(v_2[x/u_2]\mathtt{L}_2) \Rrightarrow \downarrow(v_3[x/u_3]\mathtt{L}_3)$ which concludes.
   - The case $\downarrow(v_1[x/u_1]\mathtt{L}_1) \Leftarrow (\lambda x.v)\ u \Rrightarrow \downarrow(v_2[x/u_2]\mathtt{L}_2)$ is similar to the previous one.
   - $t_{1_{\rho_1:x,u_1}} \Leftarrow t_{\rho:x,u} \Rrightarrow t_{2_{\rho_2:x,u_2}}$. By i.h., $t_1 \Rrightarrow_{\equiv} t_3 \Leftarrow_{\equiv} t_2$ and $u_1 \Rrightarrow_{\equiv} u_3 \Leftarrow_{\equiv} u_2$. By Lemma 4.2.36, $t_{1_{\rho_1:x,u_1}} \Rrightarrow_{\equiv} t_{3_{\rho_3:x,u_3}}$ and $t_{2_{\rho_2:x,u_2}} \Rrightarrow_{\equiv} t_{3_{\rho'_3:x,u_3}}$. We can conclude with Property 4.2.17 since $t_{3_{\rho_3:x,u_3}} \equiv t_{3_{\rho'_3:x,u_3}}$.

   □

5. We finally prove that $t_1 \Leftarrow_{\equiv} t \Rrightarrow_{\equiv} t_2$ implies there exists t_3 such that $t_1 \Rrightarrow_{\equiv} t_3 \Leftarrow_{\equiv} t_2$.

   Proof. Let $t_1 \Leftarrow_{\equiv} t \equiv u \Rrightarrow u' \equiv t_2$. By the third point, there is u_1 such that $t_1 \equiv u_1 \Leftarrow u$ and by the fourth point there is t_3 such that $u_1 \Rrightarrow_{\equiv} t_3 \Leftarrow_{\equiv} u'$. We conclude that $t_1 \Rrightarrow_{\equiv} t_3 \Leftarrow_{\equiv} t_2$. □

We finally conclude since the diamond property obviously implies confluence:

Corollary 4.2.38. *The simultaneous reduction $\Rrightarrow_{\equiv}$ is confluent.*

Theorem 4.2.39 (Metaconfluence of $\rightarrow_{\lambda\mathtt{j}}$)**.** *The reduction relation $\rightarrow_{\lambda\mathtt{j}}$ is confluent on metaterms.*

Proof. Thanks to Lemma 4.1.6, it is implied by Lemmas 4.2.35 and 4.2.8, and Corollaries 4.2.34 and 4.2.38. □

4.2.4 Extensions

Several strong extensions can be easily implied by metaconfluence thanks to the fact that the equivalence $\equiv$ can always be postponed with respect to $\lambda\mathtt{j}$ because it is a strong bisimulation.

Lemma 4.2.40. *The reduction relation $\equiv$ is an internal strong bisimulation i.e. if $t \equiv u$ then $t \Rightarrow_{\lambda\mathtt{j}} t'$ implies $u \Rightarrow_{\lambda\mathtt{j}} u' \equiv t'$.*

Proof. Assume $t \equiv t'$ holds in n steps, which is written as $t \equiv^n t'$, and that $t' \Rightarrow_{\lambda\mathtt{j}} s'$. We show that it exists s s.t. $t \Rightarrow_{\lambda\mathtt{j}} s$ and $s \equiv s'$. We proceed by induction on n. The proof is almost the same as the one for $\lambda\mathtt{j}$ and is straightforward.

□

Lemma 4.2.41 ($\equiv$ postponement)**.** *If $t \to^*_{\lambda\mathtt{j}} t'$ then $t \Rightarrow^*_{\lambda\mathtt{j}} \cdot \equiv t'$. Furthermore, the number of $\lambda\mathtt{j}$ steps is preserved during the reduction.*

Proof. By definition of internal strong bisimulation. □

Confluence modulo and Church-Rosser modulo: As shown in Chapter 2, there exists stronger properties than metaconfluence for a given reduction relation modulo.

Theorem 4.2.42. *The reduction relation $\Rightarrow_{\lambda\mathtt{j}}$ is confluent on metaterms modulo $\equiv$.*

Proof. If $t_1 \ {}^*_{\lambda\mathtt{j}}\!\Leftarrow t \equiv t' \Rightarrow^*_{\lambda\mathtt{j}} t_2$ then by Theorem 4.2.39, $t_1 \to^*_{\lambda\mathtt{j}} t_3 \ {}^*_{\lambda\mathtt{j}}\!\leftarrow t_2$. By Lemma 4.2.41, we can postpone $\equiv$ and get $t_1 \Rightarrow^*_{\lambda\mathtt{j}}\!\equiv t_3 \equiv \ {}^*_{\lambda\mathtt{j}}\!\Leftarrow t_2$ which concludes.

□

Theorem 4.2.43. *The reduction relation $\Rightarrow_{\lambda\mathtt{j}}$ is Church-Rosser modulo $\equiv$ on metaterms.*

Proof. Let $t \leftrightarrow^*_{\equiv} t'$ in n steps. We proceed by induction on n. The case $n = 1$ is straightforward. Otherwise, we analyse the first step.

- If $t \Rightarrow_{\lambda\mathtt{j}} t_1 \leftrightarrow^*_{\equiv} t'$ then we have $t_1 \Rightarrow^*_{\lambda\mathtt{j}}\!\equiv \ {}^*_{\lambda\mathtt{j}}\!\Leftarrow t'$ by induction hypothesis and can immediately conclude.

- If $t \equiv t_1 \leftrightarrow^*_{\equiv} t'$ then we have $t_1 \Rightarrow^*_{\lambda\mathtt{j}} s_1 \equiv s_2 \ {}^*_{\lambda\mathtt{j}}\!\Leftarrow t'$ by induction hypothesis. By Theorem 4.2.42 $t \Rightarrow_{\lambda\mathtt{j}} s_0 \equiv s_1$ and we can conclude by transitivity of $\equiv$.

- If $t \ {}_{\lambda\mathtt{j}}\!\Leftarrow t_1 \leftrightarrow^*_{\equiv} t'$ then we have $t_1 \Rightarrow^*_{\lambda\mathtt{j}} s_1 \equiv s_2 \ {}^*_{\lambda\mathtt{j}}\!\Leftarrow t'$. By Theorem 4.2.42 $t \Rightarrow_{\lambda\mathtt{j}} t_3 \equiv s_0 \ {}^*_{\lambda\mathtt{j}}\!\Leftarrow s_1$. To conclude, we transform the reduction $t' \Rightarrow^*_{\lambda\mathtt{j}} s_2 \equiv s_1 \Rightarrow_{\lambda\mathtt{j}}\!\equiv s_0 \equiv t_3$ in $t' \Rightarrow^*_{\lambda\mathtt{j}} s_2 \Rightarrow_{\lambda\mathtt{j}}\!\equiv s_0 \equiv t_3$ with Lemma 4.2.41.

□

Sigma equivalence: Other equivalence relations are defined in [AK10] including σ-equivalence on λ-terms [Reg91]. It is specified by means of the following equations:

$$\begin{array}{llll} (\lambda y.t)[x/u] & \equiv_{\sigma_1} & \lambda y.t[x/u] & y \notin \mathtt{fv}(u) \\ (t\ v)[x/u] & \equiv_{\sigma_2} & t[x/u]\ v & x \notin \mathtt{fv}(v) \end{array}$$

The equivalence relation $\equiv \cup \equiv_{\sigma_1} \cup \equiv_{\sigma_2}$ also defines a strong bisimulation on terms, and metaterms. We can thus extend the Church-Rosser property as we have done for $\equiv$, thus obtaining that the reduction relation $\lambda\mathtt{j}$ is Church-Rosser modulo $\equiv \cup \equiv_{\sigma_1} \cup \equiv_{\sigma_2}$ on metaterms.

$\lambda\mathtt{j}$-dags: An other easy consequence of our work is that $\lambda\mathtt{j}$-dags [AG09], the graph formalism of $\lambda\mathtt{j}$, extended with metaterms is confluent. Indeed, there is a strong bisimulation between the two formalisms. All the notions of equivalence on terms do not apply in this graphical representation since they become simply equalities.

Chapter 5

The Prismoid of Resources

Contents

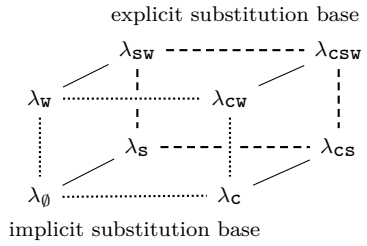
In this chapter we develop an homogeneous framework, called the *prismoid of resources*, which provides eight languages – the *vertexes* of the prismoid – dedicated to the control of resources for the λ-calculus, together with different transformation functions – the *arrows* of the prismoid – between these languages. Using those transformations, we show confluence, PSN, and strong normalization. The results of this chapter have been developped in collaboration with Delia Kesner and presented in [KR11], which is an extended and revised version of [KR09].

Each vertex of the prismoid is a specialised λ-calculus defined by a set of *well-formed terms* and a set of axioms and reduction rules as well. Each calculus is parametrised by a set of *sorts* wich are of two kinds: resources $\mathtt{w}$ (**w**eakening) and $\mathtt{c}$ (**c**ontraction), and cut-elimination operation $\mathtt{s}$ (**s**ubstitution). If a sort in the set $\{\mathtt{c}, \mathtt{s}, \mathtt{w}\}$ belongs to a given calculus, then the treatment of the corresponding operations to deal with this sort is completely explicit in this calculus, i.e. is given by syntax and rules belonging to the language itself. The eight calculi of the prismoid correspond to 2^3 different ways to combine the sorts $\{\mathtt{c}, \mathtt{s}, \mathtt{w}\}$ by means of explicit or implicit (meta-level) operations.

Thus for example, the $\lambda_{\mathtt{cs}}$-calculus has only explicit control of **c**ontraction and **s**ubstitution, the λ-calculus (called here $\lambda_{\emptyset}$-calculus), has no explicit control at all, and the $\lambda_{\mathtt{csw}}$-calculus – a slight variation of $\lambda\mathtt{lxr}$ [KL07] – has explicit control of everything.

For every subset of sorts $\mathcal{B} \subseteq \{\mathtt{c}, \mathtt{s}, \mathtt{w}\}$, the corresponding $\mathcal{B}$-calculus of the prismoid implements λ-calculus in the sense that β-reduction can be simulated

by $\mathcal{B}$-reduction. It is also possible to take off some explicit information from a given $\mathcal{B}$-calculus in order to project $\mathcal{B}$-reduction into a less refined relation. More precisely, for every $\mathcal{A} \subseteq \{\mathtt{c}, \mathtt{w}\}$, $\mathcal{A}$-reduction (resp. $\mathcal{A} \cup \mathtt{s}$-reduction) is projected into β-reduction (resp. $\mathtt{s}$-reduction). This asymmetry between languages with and without sort $\mathtt{s}$ are reflected in the prismoid by means of two conceptually different *bases*. The base $\mathfrak{B}_I$ contains all the calculi without explicit substitutions, namely $\{\lambda_\emptyset, \lambda_\mathtt{c}, \lambda_\mathtt{w}, \lambda_\mathtt{cw}\}$, and the base $\mathfrak{B}_E$ only contains those with explicit substitutions, i.e. $\{\lambda_\mathtt{s}, \lambda_\mathtt{cs}, \lambda_\mathtt{sw}, \lambda_\mathtt{csw}\}$.



For all the calculi of the prismoid we study a set of properties which guarantee that they are well-behaved, namely, simulation of β-reduction, confluence, preservation of β-strong normalisation (PSN) and strong normalisation (SN) for simply typed terms. Thus in particular, none of the calculi suffers from Mellies' counter-example [Mel95]. Full composition is also shown for all calculi with sort $\mathtt{s}$, i.e. those included in the explicit substitution base. Each property is stated and proved by making the set of sorts a *parameter*, so that the properties for each vertex of the prismoid turn out to be a particular case of some general abstract proof, which may hold for the whole prismoid or just for only one base.

Road Map: Section 5.1 introduces syntax and operational semantics of the prismoid. Section 5.2 explores how to enrich the λ-calculus by adding more explicit control of resources, while Section 5.3 deals with the dual operation which forgets information given by explicit weakening and contraction. Section 5.4 is devoted to PSN and confluence on untyped terms. Finally, typed terms are introduced in Section 5.5 together with a SN proof for them.

5.1 Terms and Rules of the Prismoid

5.1.1 Terms

We assume a denumerable set of variable symbols $x, y, z, \ldots$. Lists and sets of variables are denoted by capital Greek letters $\Gamma, \Delta, \Pi, \ldots$. We write $\Gamma; y$ for $\Gamma \cup \{y\}$ when $y \notin \Gamma$.

Definition 5.1.1 (Obligation set difference)**.** *We use $\Gamma \setminus\!\!\setminus \Delta$ for obligation set difference which is equal to set difference when $\Delta \subseteq \Gamma$ but undefined otherwise.*

Definition 5.1.2 (Terms)**.** *Terms are given by the grammar:*

$$t, u ::= x \mid \lambda x.t \mid tu \mid t[x/u] \mid \mathcal{W}_x(t) \mid \mathcal{C}_x^{y|z}(t)$$

Definition 5.1.3. *Free and bound variables of t, respectively written* $\mathtt{fv}(t)$ *and* $\mathtt{bv}(t)$*, are extended w.r.t. to the λ-calculus with the contraction and weakening constructors: $\mathcal{C}_x^{y|z}(u)$ binds y and z in u, x is free in $\mathcal{C}_x^{y|z}(u)$ and in $\mathcal{W}_x(t)$.*

Given three lists of *distinct* variables $\Gamma = x_1, \ldots, x_n$, $\Delta = y_1, \ldots, y_n$ and $\Pi = z_1, \ldots, z_n$ of the same length, the notations $\mathcal{W}_\Gamma(t)$ and $\mathcal{C}_\Gamma^{\Delta|\Pi}(t)$ mean, respectively, $\mathcal{W}_{x_1}(\ldots \mathcal{W}_{x_n}(t))$ and $\mathcal{C}_{x_1}^{y_1|z_1}(\ldots \mathcal{C}_{x_n}^{y_n|z_n}(t))$. These notations will extend naturally to sets of variables of same size thanks to the equivalence relation in Figure 5.2. The particular cases $\mathcal{C}_\emptyset^{\emptyset|\emptyset}(t)$ and $\mathcal{W}_\emptyset(t)$ mean simply t.

Definition 5.1.4 (Renaming)**.** *Given lists $\Gamma = x_1, \ldots, x_n$ and $\Delta = y_1, \ldots, y_n$ of* distinct *variables, the renaming of Γ by Δ in t, written $R_\Delta^\Gamma(t)$, is the capture-avoiding simultaneous substitution of y_i for every free occurrence of x_i in t.*

For example $R_{y_1 y_2}^{x_1 x_2}(\mathcal{C}_{x_1}^{y|z}(x_2 y z)) = \mathcal{C}_{y_1}^{y|z}(y_2 y z)$.

Definition 5.1.5 (α-conversion)**.** *The α-conversion is the (standard) congruence generated by renaming of bound variables. For example, $\lambda x_1.x_1 \mathcal{C}_x^{y_1|z_1}(y_1 z_1) \equiv_\alpha \lambda x_2.x_2 \mathcal{C}_x^{y_2|z_2}(y_2 z_2)$.*

All the operations defined along the chapter are considered modulo alpha-conversion so that in particular capture of variables is not possible.

Definition 5.1.6 (Positive free variables)**.** *The set of positive free variables in a term t, written $\mathtt{fv}^+(t)$ is defined by induction on t as:*

$$
\begin{array}{lcll}
\mathtt{fv}^+(y) & := & \{y\} & \\
\mathtt{fv}^+(\lambda y.u) & := & \mathtt{fv}^+(u) \setminus \{y\} & \\
\mathtt{fv}^+(u\ v) & := & \mathtt{fv}^+(u) \cup \mathtt{fv}^+(v) & \\
\mathtt{fv}^+(\mathcal{W}_y(u)) & := & \mathtt{fv}^+(u) & \\
\mathtt{fv}^+(u[y/v]) & := & (\mathtt{fv}^+(u) \setminus \{y\}) \cup \mathtt{fv}^+(v) & \\
\mathtt{fv}^+(\mathcal{C}_y^{z|w}(u)) & := & (\mathtt{fv}^+(u) \setminus \{z, w\}) \cup \{y\} & \textit{if } z \in \mathtt{fv}^+(u) \textit{ or } w \in \mathtt{fv}^+(u) \\
\mathtt{fv}^+(\mathcal{C}_y^{z|w}(u)) & := & \mathtt{fv}^+(u) & \textit{otherwise}
\end{array}
$$

The intuition is that $\mathtt{fv}^+(t)$ denotes the free variables of t which represent a term variable at the end of some (possibly empty) "contraction chain". For the reader familiar with proof-nets, each positive free variable can be intuitively associated to a leaf in a tree made of contraction nodes.

For instance, x is a positive free variable in $\mathcal{C}_x^{x_1|x_2}(\mathcal{W}_{x_1}(y)\ x_2)$ because there is a chain from the contraction $\mathcal{C}_x^{x_1|x_2}(\_)$ to the term variable x_2. Moreover, x is also positive in $\mathcal{C}_x^{x_1|x_2}(\mathcal{C}_{x_1}^{y|z}(z))$ because there is a chain from x to the term variable z. However x is not positive in $\mathcal{C}_x^{x_1|x_2}(\mathcal{C}_{x_1}^{x_3|x_4}(y))$ because there is no chain starting at x and ending on a term variable.

Definition 5.1.7 (Number of occurences of positive free variables)**.** *The number of occurrences of the positive free variable x in the term t is written $|t|_x^+$. We extend this definition to sets by $|t|_\Gamma^+ = \Sigma_{x \in \Gamma} |t|_x^+$.*

$$\frac{}{x \Vdash_{\mathcal{B}} x} \qquad \frac{\Gamma \Vdash_{\mathcal{B}} u \quad \Delta \Vdash_{\mathcal{B}} v}{\Gamma \uplus_{\mathcal{B}} \Delta \Vdash_{\mathcal{B}} uv} \qquad \frac{\Gamma \Vdash_{\mathcal{B}} u}{\Gamma \setminus\!\!\setminus_{\mathcal{B}} x \Vdash_{\mathcal{B}} \lambda x.u} \qquad \frac{\Gamma \Vdash_{\mathcal{B}} u}{\Gamma; x \Vdash_{\mathcal{B}} \mathcal{W}_x(u)}\,(\mathtt{w} \in \mathcal{B})$$

$$\frac{\Gamma \Vdash_{\mathcal{B}} v \quad \Delta \Vdash_{\mathcal{B}} u}{\Gamma \uplus_{\mathcal{B}} (\Delta \setminus\!\!\setminus_{\mathcal{B}} x) \Vdash_{\mathcal{B}} u[x/v]}\,(\mathtt{s} \in \mathcal{B}) \qquad \frac{\Gamma \Vdash_{\mathcal{B}} u}{x; (\Gamma \setminus\!\!\setminus_{\mathcal{B}} \{y, z\}) \Vdash_{\mathcal{B}} \mathcal{C}_x^{y|z}(u)}\,(\mathtt{c} \in \mathcal{B})$$

Figure 5.1: Well-formed terms of the prismoid

Thus for example, given $t = \mathcal{W}_{x_1}(xx)\ \mathcal{W}_x(y)\ \mathcal{C}_z^{z_1|z_2}(z_2)$, we have $x, y, z \in \mathtt{fv}^+(t)$ with $|t|_x^+ = 2$, $|t|_y^+ = |t|_z^+ = 1$ but $x_1 \notin \mathtt{fv}^+(t)$.

We now introduce the non-deterministic replacement which, unlike for other calculi where it was used such as $\lambda\mathtt{j}$, only deals with positive free variables.

Definition 5.1.8 (Positive non-deterministic replacement)**.** *Given a list of* distinct *variables $x_1 \dots x_n$, which are all fresh in t, we write $t_{\overset{+}{x \rightsquigarrow x_1 \dots x_n}}$, for the capture-avoiding non-deterministic replacement of $n \geq 1$* positive *occurrences of x in t by the variables $x_1 \dots x_n$.*

Thus for example, $(\mathcal{W}_x(t)\ x\ x)_{\overset{+}{x \rightsquigarrow y_1 y_2}}$ denotes $\mathcal{W}_x(t)\ y_1\ y_2$ or $\mathcal{W}_x(t)\ y_2\ y_1$. In the same way, $(\mathcal{W}_x(t)\ x\ x)_{\overset{+}{x \rightsquigarrow y}}$ denotes either $\mathcal{W}_x(t)\ y\ x$ or $\mathcal{W}_x(t)\ x\ y$, but neither $\mathcal{W}_y(t)\ x\ x$ nor $\mathcal{W}_x(t)\ y\ y$.

Definition 5.1.9. *Now, let us consider a set of resources $\mathcal{R} = \{\mathtt{c}, \mathtt{w}\}$ and a set of sorts $\mathcal{S} = \mathcal{R} \cup \{\mathtt{s}\}$. For every subset $\mathcal{B} \subseteq \mathcal{S}$, we define a calculus $\lambda_{\mathcal{B}}$ in the prismoid of resources which is equipped with a set of well-formed terms, denoted $\mathcal{T}_{\mathcal{B}}$ and defined in Section 5.1.2, together with a reduction relation, denoted $\to_{\mathcal{B}}$ and defined in Section 5.1.3.*

Each calculus $\lambda_{\mathcal{B}}$ belongs to a base : the explicit substitution base $\mathfrak{B}_E$ which contains all the calculi having at least sort $\mathtt{s}$ and the implicit substitution base $\mathfrak{B}_I$ containing all the other calculi.

5.1.2 Well-Formed terms

Definition 5.1.10 (Well-formed)**.** *A term t belongs to the set of well-formed terms $\mathcal{T}_{\mathcal{B}}$ iff there exists a set Γ s.t. $\Gamma \Vdash_{\mathcal{B}} t$ is derivable in the system given by the rules appearing in Figure 5.1. A term $t \in \mathcal{T}_{\mathcal{B}}$ is also called a $\mathcal{B}$-term.*

From now on we only consider well-formed terms.

In the previous rules, the symbol ; is used to denote disjoint union. Also, $\uplus_{\mathcal{B}}$ means standard union if $\mathtt{c} \notin \mathcal{B}$ and disjoint union if $\mathtt{c} \in \mathcal{B}$. Similarly, $\Gamma \setminus\!\!\setminus_{\mathcal{B}} \Delta$ is used for $\Gamma \setminus \Delta$ if $\mathtt{w} \notin \mathcal{B}$ and for $\Gamma \setminus\!\!\setminus \Delta$ if $\mathtt{w} \in \mathcal{B}$.

Notice that variables, applications and abstractions belong to all calculi of the prismoid while weakening, contraction and substitutions only appear in calculi having the corresponding sort. If t is a $\mathcal{B}$-term, then $\mathtt{w} \in \mathcal{B}$ implies that bound variables of t cannot be useless, and $\mathtt{c} \in \mathcal{B}$ implies that no free variable of t has more than

one free occurrence. Thus for example the term $\lambda z.x\ y$ belongs to the calculus $\lambda_{\mathcal{B}}$ only if $\mathtt{w} \notin \mathcal{B}$ (thus it belongs to $\lambda_\emptyset$, $\lambda_{\mathtt{c}}$, $\lambda_{\mathtt{s}}$, $\lambda_{\mathtt{cs}}$), and $(xz)[z/yx]$ belongs to $\lambda_{\mathcal{B}}$ only if $\mathtt{s} \in \mathcal{B}$ and $\mathtt{c} \notin \mathcal{B}$ (thus it belongs to $\lambda_{\mathtt{s}}$ and $\lambda_{\mathtt{sw}}$). A useful property is that $\Gamma \Vdash_{\mathcal{B}} t$ implies $\Gamma = \mathtt{fv}(t)$.

We introduce the following measure $\mathtt{o}_x(t)$ which counts free occurrences of x in t by taking care of duplications if the variable is contracted.

Definition 5.1.11 (Number of contracted occurences)**.** *The number of contracted occurrences of the free variable x in the well-formed term t, written $\mathtt{o}_x(t)$, is defined modulo α-conversion so that bound variables of t are assumed to be disjoint from x. Formally,*

$$
\begin{array}{lcll}
\mathtt{o}_x(x) & := & 1 & \\
\mathtt{o}_x(y) & := & 0 & \\
\mathtt{o}_x(\lambda y.t) & := & \mathtt{o}_x(t) & \\
\mathtt{o}_x(tu) & := & \mathtt{o}_x(t) + \mathtt{o}_x(u) & \\
\mathtt{o}_x(t[y/u]) & := & \mathtt{o}_x(t) + \mathtt{o}_x(u) & \\
\mathtt{o}_x(\mathcal{W}_y(t)) & := & \begin{cases} 1 & \textit{if } x = y \\ \mathtt{o}_x(t) & \textit{if } x \neq y \end{cases} & \\
\mathtt{o}_x(\mathcal{C}_y^{y_1|y_2}(t)) & := & \begin{cases} 1 + \mathtt{o}_{y_1}(t) + \mathtt{o}_{y_2}(t) & \textit{if } x = y \\ \mathtt{o}_x(t) & \textit{if } x \neq y \end{cases} &
\end{array}
$$

We extend this definition to sets by $\mathtt{o}_\Gamma(t) := \sum_{x\in\Gamma} \mathtt{o}_x(t)$.

Before introducing the notion of substitution, we need an extra function which cleans-up useless resources.

Definition 5.1.12 (Deletion)**.** *Given a $\mathcal{B}$-term t and a set of variables Γ, the deletion function $\mathtt{del}_\Gamma(t)$ removes from t all the occurrences of variables in Γ that are useless,* i.e. *that are free but not positive in t. This operation is defined modulo alpha-conversion so that bound variables of t are always assumed to be disjoint from Γ.*

$$
\begin{array}{lcll}
\mathtt{del}_\Gamma(y) & := & y & \\
\mathtt{del}_\Gamma(u\ v) & := & \mathtt{del}_\Gamma(u)\ \mathtt{del}_\Gamma(v) & \\
\mathtt{del}_\Gamma(\lambda y.u) & := & \lambda y.\mathtt{del}_\Gamma(u) & \\
\mathtt{del}_\Gamma(u[y/v]) & := & \mathtt{del}_\Gamma(u)[y/\mathtt{del}_\Gamma(v)] & \\
\mathtt{del}_\Gamma(\mathcal{W}_x(u)) & := & \begin{cases} u & \textit{if } x \in \Gamma \\ \mathcal{W}_x(\mathtt{del}_\Gamma(u)) & \textit{if } x \notin \Gamma \end{cases} & \\
\mathtt{del}_\Gamma(\mathcal{C}_x^{y|z}(u)) & := & \begin{cases} \mathtt{del}_{\Gamma\setminus x\cup\{y,z\}}(u) & \textit{if } x \in \Gamma \ \&\ x \notin \mathtt{fv}^+(\mathcal{C}_x^{y|z}(u)) \\ \mathcal{C}_x^{y|z}(\mathtt{del}_\Gamma(u)) & \textit{otherwise} \end{cases} &
\end{array}
$$

For example, $\mathtt{del}_x(\mathcal{W}_x(a)\ x) = a\ x$ and $\mathtt{del}_x(\mathcal{C}_x^{x_1|x_2}(y)\ x) = y\ x$. This operation does not increase the size of terms. Moreover, if $x \in \mathtt{fv}(t) \setminus \mathtt{fv}^+(t)$, then $\mathtt{size}(\mathtt{del}_x(t)) < \mathtt{size}(t)$. Also, $\mathtt{del}_\Gamma(t) = t$ if $\mathtt{fv}(t) \cap \Gamma = \emptyset$.

Lemma 5.1.13 (Preservation of Well-Formed Terms by Deletion)**.** *If* $\Gamma \Vdash_{\mathcal{B}} t$ *and* $\Delta \subseteq \Gamma$ *then* $(\Gamma \setminus\!\!\setminus_{\mathcal{B}} (\Delta \setminus \mathtt{fv}(\mathtt{del}_\Delta(t)))) \Vdash_{\mathcal{B}} \mathtt{del}_\Delta(t)$*, which simplifies to* $\Gamma \setminus\!\!\setminus_{\mathcal{B}} \Delta \Vdash_{\mathcal{B}} \mathtt{del}_\Delta(t)$ *if* $|t|^+_\Delta = 0$ *.*

Proof. By induction on $\mathtt{size}(t)$. □

For instance, cleaning-up useless x in the term $x\ \mathcal{W}_x(y)$ gives $\{x, y\} \setminus\!\!\setminus_{\mathtt{w}} (x \setminus \{x, y\}) \Vdash_{\mathtt{w}} \mathtt{del}_x(x\ \mathcal{W}_x(y))$ that is $x, y \Vdash_{\mathtt{w}} x\ y$.

To introduce the reduction rules of the prismoid we need a meta-level notion of substitution, defined on alpha-equivalence classes, which is at the same time the one implemented by the explicit control of resources.

Definition 5.1.14 (Well-formed substitution)**.** *A well-formed substitution is a pair of the form* $\{x/u\}$*, where the term* u*, called the body of the substitution, is a well-formed term. More precisely, if* $u \in \mathcal{T}_{\mathcal{B}}$*, the substitution is also called a* $\mathcal{B}$*-substitution.*

The application of a $\mathcal{B}$*-substitution* $\{x/u\}$ *to a* $\mathcal{B}$*-term* t *(called the target of the substitution), written* $t\{x/u\}$*, is defined as follows:*

- *If* $|t|^+_x = 0$*, then*
  - *If* $|t|_x = 0$ *or* $\mathtt{w} \notin \mathcal{B}$ *then* $t\{x/u\} := \mathtt{del}_x(t)$*.*
  - *Otherwise,* $t\{x/u\} := \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(\mathtt{del}_x(t))$*.*
- *If* $|t|^+_x \geq 2$*, then* $t\{x/u\} := t_{x \overset{+}{\rightsquigarrow} y}\{y/u\}\{x/u\}$*.*
- *If* $|t|^+_x = 1$*,* $t\{x/u\} := \mathtt{del}_x(t)\{\!\{x/u\}\!\}$ *where* $t\{\!\{x/u\}\!\}$ *is defined by induction on* t *as follows:*

$$
\begin{array}{lll}
x\{\!\{x/u\}\!\} & := u & \\
y\{\!\{x/u\}\!\} & := y & x \neq y \\
(s\ v)\{\!\{x/u\}\!\} & := s\{\!\{x/u\}\!\}\ v\{\!\{x/u\}\!\} & \\
(\lambda y.v)\{\!\{x/u\}\!\} & := \lambda y.v\{\!\{x/u\}\!\} & x \neq y\ \&\ y \notin \mathtt{fv}(u) \\
s[y/v]\{\!\{x/u\}\!\} & := s\{\!\{x/u\}\!\}[y/v\{\!\{x/u\}\!\}] & x \neq y\ \&\ y \notin \mathtt{fv}(u) \\
\mathcal{W}_y(v)\{\!\{x/u\}\!\} & := \mathcal{W}_{y\setminus\mathtt{fv}(u)}(v\{\!\{x/u\}\!\}) & x \neq y \\
\mathcal{C}_y^{y_1|y_2}(v)\{\!\{x/u\}\!\} & := \begin{cases} \mathcal{C}_y^{y_1|y_2}(v\{\!\{x/u\}\!\}) & \begin{cases} x \neq y \\ y_1, y_2, y \notin \mathtt{fv}(u) \end{cases} \\ \mathcal{C}_\Gamma^{\Delta|\Pi}(v\{x_1/R^\Gamma_\Delta(u)\}\{x_2/R^\Gamma_\Pi(u)\}) & \begin{cases} x = y \\ \Gamma := \mathtt{fv}(u) \\ \Delta, \Pi \textit{ are fresh} \end{cases} \end{cases} &
\end{array}
$$

For instance, $(\mathcal{W}_x(a)\ \mathcal{W}_x(b))\{x/y\} = \mathcal{W}_y(a\ b)$ and $(\mathcal{C}_x^{x_1|x_2}(a)\ x)\{x/b\} = a\ b$.

This definition looks complex, this is because it is covering all the calculi of the prismoid by a unique homogeneous specification. The restriction of this operation to particular subsets of resources results in simplified notions of substitutions. As

a typical example, the previous definition can be shown to be equivalent to the well-known notion of higher-order substitution on $\emptyset$-terms [Bar84] given by:

$$\begin{array}{lcll} x\{x/u\} & := & u & \\ y\{x/u\} & := & y & x \neq y \\ (\lambda y.v)\{x/u\} & := & \lambda y.v\{x/u\} & x \neq y \ \& \ y \notin \mathtt{fv}(u) \\ (s\ v)\{x/u\} & := & s\{x/u\}\ v\{x/u\} & \end{array}$$

Substitution definition also simplifies to the following one for c-terms:

$$\begin{array}{lcll} x\{x/u\} & := & u & \\ y\{x/u\} & := & y & x \neq y \\ (\lambda y.v)\{x/u\} & := & \lambda y.v\{x/u\} & x \neq y \ \& \ y \notin \mathtt{fv}(u) \\ (s\ v)\{x/u\} & := & s\{x/u\}\ v\{x/u\} & \\ \mathcal{C}_y^{y_1|y_2}(t)\{x/u\} & := & \mathcal{C}_y^{y_1|y_2}(t\{x/u\}) & \begin{cases} x \neq y \\ y, y_1, y_2 \notin fv(u) \end{cases} \\ \mathcal{C}_x^{y_1|y_2}(t)\{x/u\} & := & \mathcal{C}_\Gamma^{\Delta|\Pi}(t\{y_1/R_\Delta^\Gamma(u)\}\{y_2/R_\Pi^\Gamma(u)\}) & \begin{cases} x \in \mathtt{fv}^+(\mathcal{C}_x^{y_1|y_2}(t)) \\ \Gamma := \mathtt{fv}(u) \\ \Delta, \Pi \text{ are fresh} \end{cases} \\ \mathcal{C}_x^{y_1|y_2}(t)\{x/u\} & := & \mathtt{del}_x(\mathcal{C}_x^{y_1|y_2}(t)) & x \notin \mathtt{fv}^+\mathcal{C}_x^{y_1|y_2}(t) \end{array}$$

Lemma 5.1.15. *Definitions of $t\{x/u\}$ and $t\{\!\{x/u\}\!\}$ are well-founded.*

Proof. By induction on $\langle \mathtt{o}_x(t), \mathtt{size}(t)\rangle$. □

Lemma 5.1.16. *Let $t \in \mathcal{T}_\mathcal{B}$ s.t. $|t|_x^+ \geq 1$. Then substitution verifies the following equalities:*

$$\begin{array}{lcll} x\{x/u\} & = & u & \\ y\{x/u\} & = & y & x \neq y \\ (\lambda y.v)\{x/u\} & = & \lambda y.v\{x/u\} & x \neq y \\ (s\ v)\{x/u\} & = & s\{x/u\}\ v\{x/u\} & \\ s[y/v]\{x/u\} & = & s\{x/u\}[y/v\{x/u\}] & x \neq y \\ \mathcal{W}_y(t)\{x/u\} & = & \mathcal{W}_y(t\{x/u\}) & x \neq y \ \& \ y \notin \mathtt{fv}(u) \\ \mathcal{W}_y(t)\{x/u\} & = & t\{x/u\} & x \neq y \ \& \ y \in \mathtt{fv}(u) \\ \mathcal{C}_y^{y_1|y_2}(t)\{x/u\} & = & \mathcal{C}_y^{y_1|y_2}(t\{x/u\}) & x \neq y \ \& \ y \notin \mathtt{fv}(u) \\ \mathcal{C}_x^{x_1|x_2}(t)\{x/u\} & = & \mathcal{C}_\Gamma^{\Delta|\Pi}(t\{x_1/R_\Delta^\Gamma(u)\}\{x_2/R_\Pi^\Gamma(u)\}) & \begin{cases} \Gamma = \mathtt{fv}(u) \\ \Delta, \Pi \textit{ are fresh} \end{cases} \end{array}$$

Proof. By substitution definition.

□

Lemma 5.1.17. *Let $t \in \mathcal{T}_\mathcal{B}$. The function* $\mathtt{del}()$ *enjoys the following properties :*

1. $x \notin \mathtt{fv}(\mathtt{del}_x(t))$ *if* $x \notin \mathtt{fv}^+(t)$.
2. $\mathtt{del}_x(\mathtt{del}_y(t)) = \mathtt{del}_y(\mathtt{del}_x(t))$.

3. $\mathtt{del}_x(t\{y/v\}) = \mathtt{del}_x(t)\{y/v\}$ *if* $x \notin \mathtt{fv}(v)$.

4. $\mathtt{del}_x(t\{\!\{y/v\}\!\}) = \mathtt{del}_x(t)\{\!\{y/v\}\!\}$ *if* $x \notin \mathtt{fv}(v)$.

5. $\mathtt{del}_x(t)\{\!\{x/v\}\!\} = \mathtt{del}_x(t)$ *if* $x \notin \mathtt{fv}^+(t)$.

6. $\mathtt{del}_x(t) = t$ *if* $|t|_x = |t|_x^+$.

7. $t\{x/u\}\{y/u\} = \mathtt{del}_{x,y}(t)\{\!\{x/u\}\!\}\{\!\{y/u\}\!\}$ *if* $|t|_x^+ \geq 1$ *or* $|t|_y^+ \geq 1$.

Proof. By induction on $\mathtt{size}(t)$. □

For instance, $\mathtt{del}_x(\mathcal{W}_y(\mathcal{W}_x(z))\{y/w\}) = \mathtt{del}_x(\mathcal{W}_w(\mathcal{W}_x(z))) = \mathcal{W}_w(z) = \mathcal{W}_y(z)\{y/w\} = \mathtt{del}_x(\mathcal{W}_y(\mathcal{W}_x(z)))\{y/w\}$ illustrates the third case.

5.1.3 Rewriting rules and equations

We now introduce the reduction system of the prismoid. In the last column of Figure 5.2 we use the notation $\mathcal{A}^+$ (resp. $\mathcal{A}^-$) to specify that the equation/rule belongs to the calculus $\lambda_{\mathcal{B}}$ iff $\mathcal{A} \subseteq \mathcal{B}$ (resp. $\mathcal{A} \cap \mathcal{B} = \emptyset$). Thus, each calculus $\lambda_{\mathcal{B}}$ contains only a strict subset of the reduction rules and equations in Figure 5.2.

All the equations and rules can be understood by means of Proof-Nets reduction (see for example [KL07]). The reduction rules can be split into four groups: the first one fires implicit/explicit substitution, the second one implements substitution by decrementing multiplicity of variables and/or performing propagation, the third one pulls weakening operators as close to the top as possible and the fourth one pushes contractions as deep as possible. Alpha-conversion guarantees that no capture of variables occurs during reduction. The use of positive conditions (conditions involving positive free variables) in some of the rules will become clear when discussing projection at the end of Section 5.3.

The notations $\Rightarrow_{\mathcal{R}}$, $\equiv_{\mathcal{E}}$ and $\rightarrow_{\mathcal{R}\cup\mathcal{E}}$, mean, respectively, the rewriting (resp. equivalence and rewriting modulo) relation generated by the rules $\mathcal{R}$ (resp. equations $\mathcal{E}$ and rules $\mathcal{R}$ modulo equations $\mathcal{E}$). Similarly, $\Rightarrow_{\mathcal{B}}$, $\equiv_{\mathcal{B}}$ and $\rightarrow_{\mathcal{B}}$ mean, respectively, the rewriting (resp. equivalence and rewriting modulo) relation generated by the rules (resp. the equations and rules modulo equations) of the calculus $\lambda_{\mathcal{B}}$. Thus for example the reduction relation $\rightarrow_{\emptyset}$ is only generated by the β-rule exactly as in λ-calculus. Another example is $\rightarrow_{\mathtt{C}}$ which can be written $\rightarrow_{\{\beta,\mathtt{CL},\mathtt{CA_L},\mathtt{CA_R},\mathtt{CGc}\}\cup\{\mathtt{CC}_{\mathcal{A}},\mathtt{C}_{\mathcal{C}},\mathtt{CC}_{\mathcal{C}}\}}$. Sometimes we mix both notations to denote particular subrelations, thus for example $\rightarrow_{\mathtt{C}\setminus\beta}$ means $\rightarrow_{\{\mathtt{CL},\mathtt{CA_L},\mathtt{CA_R},\mathtt{CGc}\}\cup\{\mathtt{CC}_{\mathcal{A}},\mathtt{C}_{\mathcal{C}},\mathtt{CC}_{\mathcal{C}}\}}$. We give in the appendix an independent specification for each calculus of the prismoid.

Among the eight calculi of the prismoid we can distinguish the $\lambda_{\emptyset}$-calculus, known as λ-calculus, which is defined by means of the $\rightarrow_{\emptyset}$-reduction relation on $\emptyset$-terms. Another language of the prismoid is the $\lambda_{\mathtt{CSW}}$-calculus, a variation of $\lambda\mathtt{lxr}$ [KL07], defined by means of the $\rightarrow_{\{\mathtt{C},\mathtt{S},\mathtt{W}\}}$-reduction relation on $\{\mathtt{c},\mathtt{s},\mathtt{w}\}$-terms. A last example is the $\lambda_{\mathtt{W}}$-calculus given by means of $\rightarrow_{\mathtt{W}}$-reduction, that is, $\rightarrow_{\{\beta,\mathtt{LW},\mathtt{AW_l},\mathtt{AW_r}\}\cup\{\mathtt{WW}_{\mathcal{C}}\}}$.

Equations :

$(\mathtt{CC}_{\mathcal{A}})$	$\mathcal{C}_w^{x\|z}(\mathcal{C}_x^{y\|p}(t))$	$\equiv\ \mathcal{C}_w^{x\|y}(\mathcal{C}_x^{z\|p}(t))$		$\mathtt{c}^+$
$(\mathtt{C}_{\mathcal{C}})$	$\mathcal{C}_x^{y\|z}(t)$	$\equiv\ \mathcal{C}_x^{z\|y}(t)$		$\mathtt{c}^+$
$(\mathtt{CC}_{\mathcal{C}})$	$\mathcal{C}_a^{b\|c}(\mathcal{C}_x^{y\|z}(t))$	$\equiv\ \mathcal{C}_x^{y\|z}(\mathcal{C}_a^{b\|c}(t))$	$x \neq b, c\ \&\ a \neq y, z$	$\mathtt{c}^+$
$(\mathtt{WW}_{\mathcal{C}})$	$\mathcal{W}_x(\mathcal{W}_y(t))$	$\equiv\ \mathcal{W}_y(\mathcal{W}_x(t))$		$\mathtt{w}^+$
$(\mathtt{SS}_{\mathcal{C}})$	$t[x/u][y/v]$	$\equiv\ t[y/v][x/u]$	$y \notin \mathtt{fv}(u)\ \&\ x \notin \mathtt{fv}(v)$	$\mathtt{s}^+$

Rules :

(β)	$(\lambda x.t)\ u$	$\to\ t\{x/u\}$		$\mathtt{s}^-$
$(\mathtt{B})$	$(\lambda x.t)\ u$	$\to\ t[x/u]$		$\mathtt{s}^+$
$(\mathtt{V})$	$x[x/u]$	$\to\ u$		$\mathtt{s}^+$
$(\mathtt{SGc})$	$t[x/u]$	$\to\ t$	$x \notin \mathtt{fv}(t)$	$\mathtt{s}^+\ \&\ \mathtt{w}^-$
$(\mathtt{SDup})$	$t[x/u]$	$\to\ t_{\underset{x \rightsquigarrow y}{+}}[x/u][y/u]$	$\|t\|_x^+ > 1\ \&\ y$ fresh	$\mathtt{s}^+\ \&\ \mathtt{c}^-$
$(\mathtt{SL})$	$(\lambda y.t)[x/u]$	$\to\ \lambda y.t[x/u]$		$\mathtt{s}^+$
$(\mathtt{SA_L})$	$(t\ v)[x/u]$	$\to\ t[x/u]\ v$	$x \notin \mathtt{fv}(v)$	$\mathtt{s}^+$
$(\mathtt{SA_R})$	$(t\ v)[x/u]$	$\to\ t\ v[x/u]$	$x \notin \mathtt{fv}(t)$	$\mathtt{s}^+$
$(\mathtt{SS})$	$t[x/u][y/v]$	$\to\ t[x/u[y/v]]$	$y \in \mathtt{fv}^+(u) \setminus \mathtt{fv}(t)$	$\mathtt{s}^+$
$(\mathtt{SW_1})$	$\mathcal{W}_x(t)[x/u]$	$\to\ \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(t)$		$(\mathtt{sw})^+$
$(\mathtt{SW_2})$	$\mathcal{W}_y(t)[x/u]$	$\to\ \mathcal{W}_{y\setminus\mathtt{fv}(u)}(t[x/u])$	$x \neq y$	$(\mathtt{sw})^+$
$(\mathtt{LW})$	$\lambda x.\mathcal{W}_y(t)$	$\to\ \mathcal{W}_y(\lambda x.t)$	$x \neq y$	$\mathtt{w}^+$
$(\mathtt{AW_l})$	$\mathcal{W}_y(u)\ v$	$\to\ \mathcal{W}_{y\setminus\mathtt{fv}(v)}(u\ v)$		$\mathtt{w}^+$
$(\mathtt{AW_r})$	$u\ \mathcal{W}_y(v)$	$\to\ \mathcal{W}_{y\setminus\mathtt{fv}(u)}(u\ v)$		$\mathtt{w}^+$
$(\mathtt{SW})$	$t[x/\mathcal{W}_y(u)]$	$\to\ \mathcal{W}_{y\setminus\mathtt{fv}(t)}(t[x/u])$		$(\mathtt{sw})^+$
$(\mathtt{SCa})$	$\mathcal{C}_x^{y\|z}(t)[x/u]$	$\to\ \mathcal{C}_\Gamma^{\Delta\|\Pi}(t[y/R_\Delta^\Gamma(u)][z/R_\Pi^\Gamma(u)])$	$\begin{cases} y, z \in \mathtt{fv}^+(t) \\ \Gamma := \mathtt{fv}(u) \\ \Delta, \Pi \text{ are fresh} \end{cases}$	$(\mathtt{cs})^+$
$(\mathtt{CL})$	$\mathcal{C}_w^{y\|z}(\lambda x.t)$	$\to\ \lambda x.\mathcal{C}_w^{y\|z}(t)$		$\mathtt{c}^+$
$(\mathtt{CA_L})$	$\mathcal{C}_w^{y\|z}(t\ u)$	$\to\ \mathcal{C}_w^{y\|z}(t)\ u$	$y, z \notin \mathtt{fv}(u)$	$\mathtt{c}^+$
$(\mathtt{CA_R})$	$\mathcal{C}_w^{y\|z}(t\ u)$	$\to\ t\ \mathcal{C}_w^{y\|z}(u)$	$y, z \notin \mathtt{fv}(t)$	$\mathtt{c}^+$
$(\mathtt{CS})$	$\mathcal{C}_w^{y\|z}(t[x/u])$	$\to\ t[x/\mathcal{C}_w^{y\|z}(u)]$	$y, z \in \mathtt{fv}^+(u)$	$(\mathtt{cs})^+$
$(\mathtt{SCb})$	$\mathcal{C}_w^{y\|z}(t)[x/u]$	$\to\ \mathcal{C}_w^{y\|z}(t[x/u])$	$x \neq w\ \&\ y, z \notin \mathtt{fv}(u)$	$(\mathtt{cs})^+$
$(\mathtt{CW_1})$	$\mathcal{C}_w^{y\|z}(\mathcal{W}_y(t))$	$\to\ R_w^z(t)$		$(\mathtt{cw})^+$
$(\mathtt{CW_2})$	$\mathcal{C}_w^{y\|z}(\mathcal{W}_x(t))$	$\to\ \mathcal{W}_x(\mathcal{C}_w^{y\|z}(t))$	$x \neq y, z$	$(\mathtt{cw})^+$
$(\mathtt{CGc})$	$\mathcal{C}_w^{y\|z}(t)$	$\to\ R_w^z(t)$	$y \notin \mathtt{fv}(t)$	$\mathtt{c}^+\&\ \mathtt{w}^-$

Figure 5.2: The reduction rules and equations of the prismoid

In order to show that well-formed terms are stable by reduction we first need the following property.

Lemma 5.1.18 (Preservation of Well-Formed Terms by Substitution)**.** *Let* $\Gamma \Vdash_{\mathcal{B}} t$ *and* $\Delta \Vdash_{\mathcal{B}} u$ *and* $x \notin \Delta$*. If* ($x \in \mathtt{fv}^+(t)$ *or* $\mathtt{w} \in \mathcal{B}$) *and* $(\Gamma \setminus\!\!\setminus_{\mathcal{B}} x) \uplus_{\mathcal{B}} \Delta$ *is defined, then* $(\Gamma \setminus\!\!\setminus_{\mathcal{B}} x) \uplus_{\mathcal{B}} \Delta \Vdash_{\mathcal{B}} t\{x/u\}$*. Otherwise,* $\Gamma \setminus\!\!\setminus_{\mathcal{B}} x \Vdash_{\mathcal{B}} t\{x/u\}$*.*

Proof. By induction on $\langle \mathtt{o}_x(t), \mathtt{size}(t)\rangle$.

- If $|t|_x^+ = 0$ and ($|t|_x = 0$ or $\mathtt{w} \notin \mathcal{B}$) then we are done by Lemma 5.1.13.

- If $|t|_x^+ = 0$ and $|t|_x \neq 0$ and $\mathtt{w} \in \mathcal{B}$ then $t\{x/u\} = \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(\mathtt{del}_x(t))$. By hypothesis $\Gamma \Vdash_{\mathcal{B}} t$ and by Lemma 5.1.13, $\Gamma \setminus\!\!\setminus_{\mathcal{B}} x \Vdash_{\mathcal{B}} \mathtt{del}_x(t)$. By definition, $\Gamma \setminus\!\!\setminus_{\mathcal{B}} x; (\Delta \setminus \Gamma) \Vdash_{\mathcal{B}} \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(\mathtt{del}_x(t))$. If $\mathtt{c} \in \mathcal{B}$, then $\Gamma \cap \Delta = \emptyset$ so that the left part of the last statement is exactly $\Gamma \setminus\!\!\setminus_{\mathcal{B}} x \uplus_{\mathcal{B}} \Delta$ and thus we are done. Otherwise $\mathtt{c} \notin \mathcal{B}$, then we trivially conclude since $\Gamma \setminus\!\!\setminus_{\mathcal{B}} x; (\Delta \setminus \Gamma) = \Gamma \setminus\!\!\setminus_{\mathcal{B}} x \uplus_{\mathcal{B}} \Delta$.

- If $|t|_x^+ = n+1$ with $n \geq 1$ then we have $\uplus_{\mathcal{B}} = \cup$ and :

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}[hyp]\\ \vdots \\ \Gamma \Vdash_{\mathcal{B}} t\end{array}}{\Gamma, x_1, \ldots, x_n \Vdash_{\mathcal{B}} t_{\overset{+}{x \rightsquigarrow x_1 \ldots x_n}}}\, x_1 \ldots x_n \text{ fresh}}{\Gamma, x_2, \ldots, x_n \cup \Delta \Vdash_{\mathcal{B}} t_{\overset{+}{x \rightsquigarrow x_1 \ldots x_n}}\{x_1/u\}}\, i.h.}{\vdots}\, i.h.}{\Gamma \cup \Delta \cup \ldots \cup \Delta \Vdash_{\mathcal{B}} t_{\overset{+}{x \rightsquigarrow x_1 \ldots x_n}}\{x_1/u\}\ldots\{x_n/u\}}\, i.h.}{(\Gamma \cup \Delta \cup \ldots \cup \Delta) \setminus\!\!\setminus_{\mathcal{B}} x \cup \Delta \Vdash_{\mathcal{B}} t_{\overset{+}{x \rightsquigarrow x_1 \ldots x_n}}\{x_1/u\}\ldots\{x_n/u\}\{x/u\}}\, i.h.
$$

  We conclude since the last set of variables is equal to $(\Gamma \setminus\!\!\setminus_{\mathcal{B}} x) \cup \Delta$ with $\Gamma \setminus\!\!\setminus_{\mathcal{B}} x$ well defined since $|t|_x^+ = n+1$. We can use the i.h. in the first three cases since $\mathtt{o}_{x_i}(t_{\overset{+}{x \rightsquigarrow x_1 \ldots x_n}}\{x_1/u\}\ldots\{x_{i-1}/u\}) < \mathtt{o}_x(t)$ and in the last case because $\mathtt{o}_x(t_{\overset{+}{x \rightsquigarrow x_1 \ldots x_n}}\{x_1/u\}\ldots\{x_n/u\}) < \mathtt{o}_x(t)$.

- Now we analyse all interesting cases where $|t|_x^+ = 1$:

  - $t = x$, then $\Gamma = x$ and $t\{x/u\} = u$ so that $\Delta \Vdash_{\mathcal{B}} t\{x/u\}$ by hypothesis.
  - $t = \lambda y.t'$, so that $y \neq x$ by α-conversion. We have $\Gamma = \Gamma' \setminus\!\!\setminus_{\mathcal{B}} y$ (so that $\Gamma' \Vdash_{\mathcal{B}} t'$), thus $(\lambda y.t')\{x/u\} = \lambda y.\mathtt{del}_x(t')\{\!\{x/u\}\!\} = \lambda y.t'\{x/u\}$ and

    We conclude since $(\Gamma' \setminus\!\!\setminus_{\mathcal{B}} x \uplus_{\mathcal{B}} \Delta) \setminus\!\!\setminus_{\mathcal{B}} y = \Gamma \setminus\!\!\setminus_{\mathcal{B}} x \uplus_{\mathcal{B}} \Delta$ as desired.
  - $t = v\ w$. We have $\Gamma = \Gamma_v \uplus_{\mathcal{B}} \Gamma_w$, $\Gamma_v \Vdash_{\mathcal{B}} v$ and $\Gamma_w \Vdash_{\mathcal{B}} w$. Suppose $|v|_x^+ = 1$ (the case where $|w|_x^+ = 1$ is symmetric). Thus $(v\ w)\{x/u\} = \mathtt{del}_x(v)\{\!\{x/u\}\!\}\ \mathtt{del}_x(w)\{\!\{x/u\}\!\} = v\{x/u\}\ w$ and :

$$\dfrac{\dfrac{\begin{array}{c}[hyp]\\ \vdots\\ \Gamma_v \Vdash_{\mathcal{B}} v\end{array}}{\Gamma_v \setminus\!\!\setminus_{\mathcal{B}} x \uplus \Delta \Vdash_{\mathcal{B}} v\{x/u\}}\,i.h. \qquad \begin{array}{c}[hyp]\\ \vdots\\ \Gamma_w \Vdash_{\mathcal{B}} w\end{array}}{(\Gamma_v \setminus\!\!\setminus_{\mathcal{B}} x \uplus_{\mathcal{B}} \Gamma_w) \uplus_{\mathcal{B}} \Delta \Vdash_{\mathcal{B}} v\{x/u\}\ w}$$

We can conclude since $\Gamma_v \setminus x \uplus_{\mathcal{B}} \Gamma_w = \Gamma \setminus x$

– $t = \mathcal{C}_x^{y|z}(t')$. By hypothesis we have $x;\Gamma' \setminus\!\!\setminus_{\mathcal{B}} \{y,z\} \Vdash_{\mathcal{B}} \mathcal{C}_x^{y|z}(t')$ (so that $\Gamma' \Vdash_{\mathcal{B}} t'$) with $\Gamma = x;\Gamma' \setminus\!\!\setminus_{\mathcal{B}} \{y,z\}$. Definition of substitution gives $t\{x/u\} = \mathtt{del}_x(\mathcal{C}_x^{y|z}(t'))\{\!\{x/u\}\!\} = \mathcal{C}_x^{y|z}(t')\{\!\{x/u\}\!\} = \mathcal{C}_\Delta^{\Delta'|\Delta''}(t'\{y/u'\}\{z/u''\})$, where $\Delta = \mathtt{fv}(u)$,
If $\mathtt{o}_y(t') > 0$ and $\mathtt{o}_z(t') > 0$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}[hyp]\\ \vdots\\ \Gamma' \Vdash_{\mathcal{B}} t'\end{array}}{\Gamma' \setminus\!\!\setminus_{\mathcal{B}} y \uplus_{\mathcal{B}} \Delta' \Vdash_{\mathcal{B}} t'\{y/u'\}}\,i.h.}{\Gamma' \setminus\!\!\setminus_{\mathcal{B}} \{y,z\} \uplus_{\mathcal{B}} \Delta' \uplus_{\mathcal{B}} \Delta'' \Vdash_{\mathcal{B}} t'\{y/u'\}\{z/u''\}}\,i.h.}{\Gamma' \setminus\!\!\setminus_{\mathcal{B}} \{y,z\} \uplus_{\mathcal{B}} \Delta \Vdash_{\mathcal{B}} \mathcal{C}_\Delta^{\Delta'|\Delta''}(t'\{y/u'\}\{z/u''\})}$$

The first (resp. second) application of the i.h. is valid since $\mathtt{o}_y(t') < \mathtt{o}_x(t)$ (resp. $\mathtt{o}_z(t'\{y/u'\}) < \mathtt{o}_x(t)$). We can conclude since $\Gamma \setminus\!\!\setminus_{\mathcal{B}} x = \Gamma' \setminus\!\!\setminus_{\mathcal{B}} \{y,z\}$.
Finally, suppose $\mathtt{o}_y(t') = 0$ and $\mathtt{w} \notin \mathcal{B}$ (otherwise, the proof is similar to another detailed case). Then,

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}[hyp]\\ \vdots\\ \Gamma' \Vdash_{\mathcal{B}} t'\end{array}}{\Gamma' \setminus\!\!\setminus_{\mathcal{B}} y \Vdash_{\mathcal{B}} t'\{y/u'\}}\,i.h.}{\Gamma' \setminus\!\!\setminus_{\mathcal{B}} \{y,z\} \uplus_{\mathcal{B}} \Delta'' \Vdash_{\mathcal{B}} t'\{y/u'\}\{z/u''\}}\,i.h.}{(\Gamma' \setminus\!\!\setminus_{\mathcal{B}} \{y,z\} \uplus_{\mathcal{B}} \Delta') \setminus\!\!\setminus_{\mathcal{B}} \Delta' \setminus\!\!\setminus_{\mathcal{B}} \Delta'';\Delta \Vdash_{\mathcal{B}} \mathcal{C}_\Delta^{\Delta'|\Delta''}(t'\{y/u'\}\{z/u''\})}$$

We can conclude since $(\Gamma' \setminus\!\!\setminus_{\mathcal{B}} \{y,z\} \uplus_{\mathcal{B}} \Delta') \setminus\!\!\setminus_{\mathcal{B}} \Delta' \setminus\!\!\setminus_{\mathcal{B}} \Delta'';\Delta$ is exactly $\Gamma' \setminus\!\!\setminus_{\mathcal{B}} \{y,z\} \uplus_{\mathcal{B}} \Delta$ ($\setminus\!\!\setminus_{\mathcal{B}} = \setminus$ since $\mathtt{w} \notin \mathcal{B}$ and $; = \uplus_{\mathcal{B}}$ since $\mathtt{c} \in \mathcal{B}$.

– The case $t = w[y/v]$ is similar to lambda and application together.

□

For instance, suppose $x \Vdash_{\mathtt{C}} \mathcal{C}_x^{x_1|x_2}(x_1\ x_2)$ and $y \Vdash_{\mathtt{C}} y$. In this case, we have $\setminus\!\!\setminus_{\mathtt{C}} = \setminus$ and $\uplus_{\mathcal{B}}$ is the disjoint union. $(x \setminus\!\!\setminus x) \uplus_{\mathtt{C}} y = y$ is defined and $\mathcal{C}_x^{x_1|x_2}(x_1\ x_2)\{x/y\} = \mathcal{C}_y^{y_1|y_2}(y_1\ y_2)$ so that $y \Vdash_{\mathtt{C}} \mathcal{C}_y^{y_1|y_2}(y_1\ y_2)$.

As expected, substitution enjoys the following property.

Lemma 5.1.19 (Substitution Permutation)**.** *Let $t, u, v \in \mathcal{T}_{\mathcal{B}}$ s.t. $x \notin \mathtt{fv}(v)$ and $y \notin \mathtt{fv}(u)$. Then:*

1. $t\{x/u\}\{y/v\} \equiv_{\mathcal{B}} t\{y/v\}\{x/u\}$
2. $t\{\!\{x/u\}\!\}\{\!\{y/v\}\!\} \equiv_{\mathcal{B}} t\{\!\{y/v\}\!\}\{\!\{x/u\}\!\}$

Proof. We prove both statements simultaneously by induction on the tuple $\langle \mathtt{o}_{\{x,y\}}(t), \mathtt{size}(t)\rangle$.

1.
   - First, we treat cases where $|\mathtt{fv}^+(t)|_x \geq 2$ or $|\mathtt{fv}^+(t)|_y \geq 2$. Let us suppose $|\mathtt{fv}^+(t)|_x \geq 2$ and $|\mathtt{fv}^+(t)|_y \geq 2$, the other cases being similar. Then $|\mathtt{fv}^+(t)|_x = n+1$ and $|\mathtt{fv}^+(t)|_y = m+1$ so that:

$$\begin{array}{ll} & t\{x/u\}\{y/v\} \\ = & t_{x \overset{+}{\rightsquigarrow} x_1 \ldots x_n} \overline{\{x_n/u\}}\{x/u\}_{y \overset{+}{\rightsquigarrow} y_1 \ldots y_m} \{y_1/v\}\ldots\{y_n/v\}\{y/v\} \\ & \qquad \text{where } \overline{\{x_n/u\}} = \{x_1/u\}\ldots\{x_n/u\} \\ = & t_{x \overset{+}{\rightsquigarrow} x_1 \ldots x_n\, y \overset{+}{\rightsquigarrow} y_1 \ldots y_m} \overline{\{x_n/u\}}\{x/u\}\{y_1/v\}\ldots\{y_n/v\}\{y/v\} \\ \equiv_{\mathcal{B}} (i.h.) & t_{x \overset{+}{\rightsquigarrow} x_1 \ldots x_n\, y \overset{+}{\rightsquigarrow} y_1 \ldots y_m} \{y_1/v\}\ldots\{y_n/v\}\{y/v\}\overline{\{x_n/u\}}\{x/u\} \\ = & t\{y/v\}\{x/u\} \end{array}$$

   - If $|t|_x^+ = 0$ and ($|\mathtt{fv}(t)|_x = 0$ or $\mathtt{w} \notin \mathcal{B}$) then

$$t\{x/u\}\{y/v\} = \mathtt{del}_x(t)\{y/v\} =_{L.\ 5.1.17:3} \mathtt{del}_x(t\{y/v\}) = t\{y/v\}\{x/u\}$$

   - If $|t|_x^+ = 0$ and $|\mathtt{fv}(t)|_x \neq 0$ and $\mathtt{w} \in \mathcal{B}$ then

$$t\{x/u\}\{y/v\} = \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(\mathtt{del}_x(t))\{y/v\}$$

   There are two interesting cases :
   - $|t|_y^+ = 0$ and $|\mathtt{fv}(t)|_y > 0$

$$\begin{array}{ll} & t\{x/u\}\{y/v\} = \\ = & \mathcal{W}_{\mathtt{fv}(v)\setminus\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(\mathtt{del}_y(\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(\mathtt{del}_x(t)))) \\ = & \mathcal{W}_{\mathtt{fv}(v)\setminus\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(\mathtt{del}_y(\mathtt{del}_x(t)))) \\ =_{L.\ 5.1.17:2} & \mathcal{W}_{\mathtt{fv}(v)\setminus\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(\mathtt{del}_x(\mathtt{del}_y(t)))) \\ = & \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(v)\setminus\mathtt{fv}(t)}(\mathtt{del}_x(\mathcal{W}_{\mathtt{fv}(v)\setminus\mathtt{fv}(t)}(\mathtt{del}_y(t)))) \\ = & \mathcal{W}_{\mathtt{fv}(v)\setminus\mathtt{fv}(t)}(\mathtt{del}_y(t))\{x/u\} \\ = & t\{y/v\}\{x/u\} \end{array}$$

   - $|t|_y^+ = 1$

$$\begin{array}{ll} & t\{x/u\}\{y/v\} \\ = & \mathtt{del}_y(\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(\mathtt{del}_x(t)))\{\!\{y/v\}\!\} \\ = & \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)\setminus\mathtt{fv}(v)}(\mathtt{del}_y(\mathtt{del}_x(t))\{\!\{y/v\}\!\}) \\ =_{L.\ 5.1.17:2} & \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(v)\setminus\mathtt{fv}(t)}(\mathtt{del}_x(\mathtt{del}_y(t)\{\!\{y/v\}\!\})) \\ = & \mathtt{del}_y(t)\{\!\{y/v\}\!\}\{x/u\} \\ = & t\{y/v\}\{x/u\} \end{array}$$

- We now consider the case where $|t|_x^+ = |t|_y^+ = 1$. We proceed by case analysis on t.

  - The case $t = z$ is impossible by hypothesis.
  - $t = \lambda w.t'$.

$$\begin{array}{ll} & (\lambda w.t')\{x/u\}\{y/v\} \\ = & \mathtt{del}_x(\lambda w.t')\{\!\{x/u\}\!\}\{y/v\} \\ = & (\lambda w.\mathtt{del}_x(t')\{\!\{x/u\}\!\})\{y/v\} \\ =_{L.\ 5.1.17:4} & \lambda w.\mathtt{del}_y(\mathtt{del}_x(t'))\{\!\{x/u\}\!\}\{\!\{y/v\}\!\} \\ \equiv_{\mathcal{B}} (i.h.) & \lambda w.\mathtt{del}_y(\mathtt{del}_x(t'))\{\!\{y/v\}\!\}\{\!\{x/u\}\!\} \\ =_{L.\ 5.1.17:2} & \lambda w.\mathtt{del}_x(\mathtt{del}_y(t'))\{\!\{y/v\}\!\}\{\!\{x/u\}\!\} \\ =_{L.\ 5.1.17:4} & \lambda w.\mathtt{del}_x(\mathtt{del}_y(t')\{\!\{y/v\}\!\})\{\!\{x/u\}\!\} \\ = & (\lambda w.t')\{y/v\}\{x/u\} \end{array}$$

  - $t = w\ w'$.

$$\begin{array}{ll} & t\{x/u\}\{y/v\} \\ = & w\{x/u\}\{y/v\}\ w'\{x/u\}\{y/v\} \\ \equiv_{\mathcal{B}} (i.h.) & w\{y/v\}\{x/u\}\ w'\{y/v\}\{x/u\} \\ = & t\{y/v\}\{x/u\} \end{array}$$

  - The case $t = s[z/w]$ is similar to the application case.
  - The case $t = \mathcal{W}_x(t')$ is impossible by hypothesis.
  - The case $t = \mathcal{W}_z(t')$ with $z \neq x, y$ is straightforward by induction.
  - $t = \mathcal{C}_a^{b|c}(t')$. We only consider the case where $a = x$

$$\begin{array}{ll} & t\{x/u\}\{y/v\} \\ = & \mathcal{C}_\Gamma^{\Delta|\Pi}(t'\{b/R_\Delta^\Gamma(u)\}\{c/R_\Pi^\Gamma(u)\})\{y/v\} \\ = & \mathcal{C}_\Gamma^{\Delta|\Pi}(t'\{b/R_\Delta^\Gamma(u)\}\{c/R_\Pi^\Gamma(u)\}\{y/v\}) \\ \equiv_{\mathcal{B}} (i.h.) & \mathcal{C}_\Gamma^{\Delta|\Pi}(t'\{y/v\}\{b/R_\Delta^\Gamma(u)\}\{c/R_\Pi^\Gamma(u)\}) \\ = & \mathcal{C}_a^{b|c}(t'\{y/v\})\{x/u\} \\ = & \mathcal{C}_a^{b|c}(\mathtt{del}_y(t')\{\!\{y/v\}\!\})\{x/u\} \\ = & t\{y/v\}\{x/u\} \end{array}$$

2. This statement can be proved in a similar way.

□

Lemma 5.1.20 (Preservation of Well-Formed Terms by Reduction)**.**
If $\Gamma \Vdash_{\mathcal{B}} t$ and $t \to_{\mathcal{B}} u$, then there exists a set Δ s.t. $\Delta \subseteq \Gamma$ and $\Delta \Vdash_{\mathcal{B}} u$. Moreover if $\mathtt{w} \in \mathcal{B}$, $\Delta = \Gamma$.

Proof. By induction on $\mathtt{size}(t)$ using Lemma 5.1.18. □

Lemma 5.1.21. *Let $t \in \mathcal{T}_{\mathcal{B}}$ and $\Gamma \subseteq \mathtt{fv}(t)$ s.t. $|t|^+_\Gamma = 0$. Then $t \to^*_{\mathcal{B}} \mathtt{del}_\Gamma(t)$ if $\mathtt{w} \notin \mathcal{B}$, and $t \to^*_{\mathcal{B}} \mathcal{W}_\Gamma(\mathtt{del}_\Gamma(t))$, if $\mathtt{w} \in \mathcal{B}$.*

Proof. By induction on $\mathtt{size}(t)$. □

For instance $\mathcal{C}^{y|z}_x(w) \to_{\mathtt{CGc}} w = \mathtt{del}_x(\mathcal{C}^{y|z}_x(w))$ and $\mathcal{W}_y(z)\ \mathcal{W}_z(a) \to_{\mathtt{AW_l}} \to_{\mathtt{AW_r}} \mathcal{W}_y(z\ \mathcal{W}_z(a)) = \mathcal{W}_y(\mathtt{del}_y(\mathcal{W}_y(z)\ \mathcal{W}_z(a)))$.

Lemma 5.1.22 (Full Composition)**.** *Let $t[\overline{y}/\overline{v}] \in \mathcal{T}_{\mathcal{B}}$ be a term having independent substitutions $[\overline{y}/\overline{v}]$. Then $t[\overline{y}/\overline{v}] \to^*_{\mathcal{B}} t\{\overline{y}/\overline{v}\}$.*

Proof. By induction on $\langle \mathtt{o}_{\overline{y}}(t), \mathtt{size}(t)\rangle$, where $\mathtt{o}_{\overline{y}}(t) = \Sigma_{i\in\{1\ldots n\}}\mathtt{o}_{y_i}(t)$. Let $[\overline{y}/\overline{v}] = [x/u][\overline{x}/\overline{u}]$. We first show $t[x/u] \to^*_{\mathcal{B}} t\{x/u\}$, so that $t\{x/u\}[\overline{x}/\overline{u}] \to^*_{\mathcal{B}} t\{x/u\}\{\overline{x}/\overline{u}\} = t\{\overline{y}/\overline{v}\}$ by the i.h. since independence of $[\overline{y}/\overline{v}]$ imply $\mathtt{o}_{\overline{x}}(t\{x/u\}) < \mathtt{o}_{\overline{y}}(t)$.

- If $x \notin \mathtt{fv}(t)$, then $t[x/u] \to_{\mathtt{SGc}} t = t\{x/u\}$.
- If $|t|^+_x = n+1 \geq 2$, then we can apply n times the rule $\mathtt{SDup}$ in such a way that each reduction step only replaces one occurrence of the truly free variable x of t. This gives the following, where we can apply the i.h. since the substitutions are independent:

$$\begin{array}{ll}
t[x/u] & \to_{\mathtt{SDup}} \\
t_{x \overset{+}{\rightsquigarrow} z_n}[x/u][z_n/u] & \to_{\mathtt{SDup}} \\
\vdots & \\
t_{x \overset{+}{\rightsquigarrow} z_1 \ldots z_n}[x/u][z_1/u]\ldots[z_n/u] & \equiv_{\mathtt{SS}_{\mathcal{C}}} \\
t_{x \overset{+}{\rightsquigarrow} z_1 \ldots z_n}[z_1/u]\ldots[z_n/u][x/u] & \to^*_{\mathcal{B}} (i.h.) \\
t_{x \overset{+}{\rightsquigarrow} z_1 \ldots z_n}\{z_1/u\}\ldots\{z_n/u\}\{x/u\} & = t\{x/u\}
\end{array}$$

- If $|t|^+_x = 0$ and $|\mathtt{fv}(t)|_x > 0$, we consider the case where $\mathtt{w} \in \mathcal{B}$, as the one where $\mathtt{w} \notin \mathcal{B}$ is similar to the case where $x \notin \mathtt{fv}(t)$:

$$\begin{array}{ll}
t[x/u] & \to^*_{L.\ 5.1.21} \\
\mathcal{W}_x(\mathtt{del}_x(t))[x/u] & \to_{\mathtt{SW_1}} \\
\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(\mathtt{del}_x(t))}(\mathtt{del}_x(t)) & = \\
\mathcal{W}_{\mathtt{fv}(u)\setminus(\mathtt{fv}(t)\setminus\{x\})}(\mathtt{del}_x(t)) & = (x \notin \mathtt{fv}(u)) \\
\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(\mathtt{del}_x(t)) & = t\{x/u\}
\end{array}$$

- Now, consider the case where $|t|^+_x = 1$. We proceed by case analysis on t:
  - $t = x$. Then $x[x/u] \to_{\mathtt{V}} u = t\{x/u\}$.
  - $t = \lambda y.t'$. Then $t[x/u] \to_{\mathtt{SL}} \lambda y.t'[x/u] \to^*_{\mathcal{B}} (i.h.)\lambda y.t'\{x/u\} = t\{x/u\}$.

– $t = v\ w$.

If $x \in \mathtt{fv}^+(v)$ (so that $x \notin \mathtt{fv}^+(w)$) and $x \in \mathtt{fv}(w)$:

$$\begin{array}{ll}
(v\ w)[x/u] & \to^*_{\mathcal{B}}\ \ (L.\ 5.1.21) \\
(v\ \mathcal{W}_x(\mathtt{del}_x(w)))[x/u] & \to_{\mathtt{AW_r}} \\
(v\ \mathtt{del}_x(w))[x/u] & \to_{\mathtt{SA_L}} \\
(v[x/u]\ \mathtt{del}_x(w)) & \to^*_{\mathcal{B}}\ \ (i.h.) \\
(v\{x/u\}\ \mathtt{del}_x(w)) & = \\
(\mathtt{del}_x(v)\{\!\{x/u\}\!\}\ \mathtt{del}_x(w)) & =_{L.\ 5.1.17:5} \\
\mathtt{del}_x(v)\{\!\{x/u\}\!\}\ \mathtt{del}_x(w)\{\!\{x/u\}\!\} & = (v\ w)\{x/u\}
\end{array}$$

If $x \in \mathtt{fv}^+(v)$ (so that $x \notin \mathtt{fv}^+(w)$) and $x \notin \mathtt{fv}(w)$:

$$\begin{array}{ll}
(v\ w)[x/u] & \to_{\mathtt{SA_L}} \\
v[x/u]\ w & \to^*_{\mathcal{B}}\ \ (i.h.) \\
v\{x/u\}\ w & = \\
\mathtt{del}_x(v)\{\!\{x/u\}\!\}\ w & = (v\ w)\{x/u\}
\end{array}$$

If $x \in \mathtt{fv}^+(w)$, then the proof is similar but uses rules $\mathtt{AW_l}$ and $\mathtt{SA_R}$.

– $t = v[y/w]$. Similar to the previous case using $\mathtt{SW}$ and $\mathtt{SS}_{\mathcal{C}}$ in the first case; $\mathtt{SW_2}$ and $\mathtt{SS}$ in the second case.

– $t = \mathcal{W}_y(v)$.

The case $y = x$ is impossible by hypothesis so that $y \neq x$ and we have:

$$\begin{array}{ll}
\mathcal{W}_y(v)[x/u] & \to_{\mathtt{SW_2}} \\
\mathcal{W}_{y\setminus\mathtt{fv}(u)}(v[x/u]) & \to^*_{\mathcal{B}}\ \ (i.h.) \\
\mathcal{W}_{y\setminus\mathtt{fv}(u)}(v\{x/u\}) & = \\
\mathcal{W}_y(\mathtt{del}_x(v)\{\!\{x/u\}\!\}) & = \\
\mathcal{W}_y(\mathtt{del}_x(v))\{\!\{x/u\}\!\} & = \mathcal{W}_y(v)\{x/u\}
\end{array}$$

– $t = \mathcal{C}_y^{y_1|y_2}(v)$. We consider the case where $y = x$, the other one is straightforward. Let $\Gamma = \mathtt{fv}(u)$. Then,

$$\begin{array}{ll}
\mathcal{C}_x^{y_1|y_2}(v)[x/u] & \to_{\mathtt{SCa}} \\
\mathcal{C}_\Gamma^{\Delta|\Pi}(v[y_1/R^\Gamma_\Delta(u)][y_2/R^\Gamma_\Pi(u)]) & \to^*_{\mathcal{B}}\ \ (i.h.) \\
\mathcal{C}_\Gamma^{\Delta|\Pi}(v\{y_1/R^\Gamma_\Delta(u)\}\{y_2/R^\Gamma_\Pi(u)\}) & = \\
\mathcal{C}_x^{y_1|y_2}(v)\{x/u\} &
\end{array}$$

□

For instance, if $\Gamma = \mathtt{fv}(u)$, Π, Δ are fresh, $u_1 = R^{\Gamma}_{\Delta}(u)$ and $u_2 = R^{\Gamma}_{\Pi}(u)$, then

$$\begin{array}{ll}
\mathcal{C}^{y_1|y_2}_{y}(\mathcal{W}_{y_1}(\mathcal{W}_x(y_2)))[x/v][y/u] & \equiv_{\mathtt{SS}_{\mathcal{C}}} \\
\mathcal{C}^{y_1|y_2}_{y}(\mathcal{W}_{y_1}(\mathcal{W}_x(y_2)))[y/v][x/v] & \to_{\mathtt{SCa}} \\
\mathcal{C}^{\Delta|\Pi}_{\Gamma}(\mathcal{W}_{y_1}(\mathcal{W}_x(y_2))[y_1/u_1][y_2/u_2])[x/v] & \to_{\mathtt{SW}_1} \\
\mathcal{C}^{\Delta|\Pi}_{\Gamma}(\mathcal{W}_{\mathtt{fv}(u_1)}(\mathcal{W}_x(y_2))[y_2/u_2])[x/v] & \to_{\mathtt{SW}_2} \\
\mathcal{C}^{\Delta|\Pi}_{\Gamma}(\mathcal{W}_{\mathtt{fv}(u_1)}(\mathcal{W}_x(y_2)[y_2/u_2]))[x/v] & \to_{\mathtt{SW}_2} \\
\mathcal{C}^{\Delta|\Pi}_{\Gamma}(\mathcal{W}_{\mathtt{fv}(u_1)}(\mathcal{W}_x(y_2[y_2/u_2])))[x/v] & \to_{\mathtt{V}} \\
\mathcal{C}^{\Delta|\Pi}_{\Gamma}(\mathcal{W}_{\mathtt{fv}(u_1)}(\mathcal{W}_x(u_2)))[x/v] &
\end{array}$$

This is correct since:

$$\begin{array}{ll}
\mathcal{C}^{y_1|y_2}_{y}(\mathcal{W}_{y_1}(\mathcal{W}_x(y_2)))[x/v]\{y/u\} & = \\
\mathcal{C}^{y_1|y_2}_{y}(\mathcal{W}_{y_1}(\mathcal{W}_x(y_2)))[x/v]\{\!\{y/u\}\!\} & = \\
\mathcal{C}^{\Delta|\Pi}_{\Gamma}(\mathcal{W}_{y_1}(\mathcal{W}_x(y_2))\{y_1/u_1\}\{y_2/u_2\})[x/v] & = \\
\mathcal{C}^{\Delta|\Pi}_{\Gamma}(\mathcal{W}_{\mathtt{fv}(u_1)}(\mathcal{W}_x(u_2)))[x/v] &
\end{array}$$

5.2 Adding Resources

This section is devoted to the simulation of the $\lambda_{\emptyset}$-calculus into richer calculi having more resources. We consider the function $\mathtt{AR}_{\mathcal{A}}(\_) : \mathcal{T}_{\emptyset} \mapsto \mathcal{T}_{\mathcal{A}}$ for $\mathcal{A} \subseteq \mathcal{R}$ which enriches a $\lambda_{\emptyset}$-term in order to fulfill the constraints needed to be an $\mathcal{A}$-term. Adding is done not only on a static level (the terms) but also on a dynamic level (the reduction).

Definition 5.2.1 (Adding function)**.**

$$\begin{array}{lll}
\mathtt{AR}_{\mathcal{A}}(x) & := x & \\
\mathtt{AR}_{\mathcal{A}}(\lambda x.t) & := \begin{cases} \lambda x.\mathcal{W}_x(\mathtt{AR}_{\mathcal{A}}(t)) & \mathtt{w} \in \mathcal{A} \ \&\ x \notin \mathtt{fv}(t) \\ \lambda x.\mathtt{AR}_{\mathcal{A}}(t) & \textit{otherwise} \end{cases} & \\
\mathtt{AR}_{\mathcal{A}}(t\ u) & := \begin{cases} \mathcal{C}^{\Delta|\Pi}_{\Gamma}(R^{\Gamma}_{\Delta}(\mathtt{AR}_{\mathcal{A}}(t))R^{\Gamma}_{\Pi}(\mathtt{AR}_{\mathcal{A}}(u))) & \begin{cases} \mathtt{c} \in \mathcal{A} \ \&\ \Gamma := \mathtt{fv}(t) \cap \mathtt{fv}(u) \\ \Delta, \Pi \ \textit{are fresh} \end{cases} \\ \mathtt{AR}_{\mathcal{A}}(t)\ \mathtt{AR}_{\mathcal{A}}(u) & \textit{otherwise} \end{cases} &
\end{array}$$

For example, adding resource $\mathtt{c}$ (resp. $\mathtt{w}$) to $t = \lambda x.yy$ gives $\lambda x.\mathcal{C}^{y_1|y_2}_{y}(y_1y_2)$ (resp. $\lambda x.\mathcal{W}_x(yy)$), while adding both of them gives $\lambda x.\mathcal{W}_x(\mathcal{C}^{y_1|y_2}_{y}(y_1y_2))$.

Lemma 5.2.2. *Let $t \in \mathcal{T}_{\emptyset}$, then we have*

1. $\mathtt{fv}(t) = \mathtt{fv}(\mathtt{AR}_{\mathcal{A}}(t)) = \mathtt{fv}^{+}(\mathtt{AR}_{\mathcal{A}}(t))$.

2. $\mathtt{del}_{\Gamma}(\mathtt{AR}_{\mathcal{A}}(t)) = \mathtt{AR}_{\mathcal{A}}(t)$.

Proof. By induction on $\mathtt{size}(t)$. □

Point 1 says that $\mathtt{AR}_{\mathcal{A}}()$ only adds useful (i.e. positive) variables; thus deleting any non positive free variable in $\mathtt{AR}_{\mathcal{A}}(t)$ will leave the term unchanged as stated by Point 2.

We now establish the relation between $\mathtt{AR}_{\mathcal{A}}()$ and well-formed substitution; this is a technical key lemma of the chapter.

Lemma 5.2.3. *Let $t, u \in \mathcal{T}_{\emptyset}$ and $\mathcal{A} \subseteq \mathcal{R}$. Then*

- *If $\mathtt{c} \notin \mathcal{A}$ then $\mathtt{AR}_{\mathcal{A}}(t)\{x/\mathtt{AR}_{\mathcal{A}}(u)\} = \mathtt{AR}_{\mathcal{A}}(t\{x/u\})$.*
- *If $\mathtt{c} \in \mathcal{A}$ then $\mathcal{C}_{\Gamma}^{\Delta|\Pi}(R_{\Delta}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(t))\{x/R_{\Pi}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(u))\}) \to_{\mathcal{A}}^{*} \mathtt{AR}_{\mathcal{A}}(t\{x/u\})$ where $\Gamma = (\mathtt{fv}(t) \setminus x) \cap \mathtt{fv}(u)$ and Δ, Π are fresh sets of variables.*

Proof. By induction on $\mathtt{size}(t)$, using the simplified definition of substitution for $\emptyset$-terms in Section 5.1.2. By Lemma 5.2.2:1, x cannot be a free variable of t which is not positive so that we can use the simplification notion of substitution given by Lemma 5.1.16. The case $\mathtt{c} \notin \mathcal{A}$ can be easily done by i.h. so we only consider $\mathtt{c} \in \mathcal{A}$.

First suppose $x \notin \mathtt{fv}(t)$. Then,

$$\begin{array}{ll}
\mathcal{C}_{\Gamma}^{\Delta|\Pi}(R_{\Delta}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(t))\{x/R_{\Pi}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(u))\}) & = \\
\mathcal{C}_{\Gamma}^{\Delta|\Pi}(R_{\Delta}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(t))) & \to_{\mathtt{CGc}} \\
R_{\Gamma}^{\Delta}(R_{\Delta}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(t))) & = \\
\mathtt{AR}_{\mathcal{A}}(t) & = \\
\mathtt{AR}_{\mathcal{A}}(t\{x/u\}) &
\end{array}$$

Otherwise, $x \in \mathtt{fv}(t)$ (and in particular, $x \in \mathtt{fv}^{+}(t)$ by Lemma 5.2.2:1). We consider different cases.

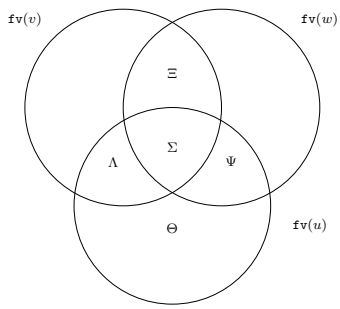
- The case $t = x$ is similar to the case where $\mathtt{c} \notin \mathcal{A}$.
- $t = \lambda y.t'$.
  - $y \notin \mathtt{fv}(t')$ and $\mathtt{w} \in \mathcal{A}$.

$$\begin{array}{ll}
\mathcal{C}_{\Gamma}^{\Delta|\Pi}((R_{\Delta}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(\lambda y.t')))\{x/R_{\Pi}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(u))\}) & = \\
\mathcal{C}_{\Gamma}^{\Delta|\Pi}((\lambda y.\mathcal{W}_y(R_{\Delta}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(t'))))\{x/R_{\Pi}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(u))\}) & = \\
\mathcal{C}_{\Gamma}^{\Delta|\Pi}(\lambda y.\mathcal{W}_y(R_{\Delta}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(t'))\{x/R_{\Pi}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(u))\})) & \to_{\mathtt{CL}} \\
\lambda y.\mathcal{C}_{\Gamma}^{\Delta|\Pi}(\mathcal{W}_y(R_{\Delta}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(t'))\{x/R_{\Pi}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(u))\})) & \to_{\mathtt{CW}_2} \\
\lambda y.\mathcal{W}_y(\mathcal{C}_{\Gamma}^{\Delta|\Pi}(R_{\Delta}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(t'))\{x/R_{\Pi}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(u))\})) & \to_{\mathcal{A}}^{*} (i.h.) \\
\lambda y.\mathcal{W}_y(\mathtt{AR}_{\mathcal{A}}(t'\{x/u\})) & = \\
\mathtt{AR}_{\mathcal{A}}(\lambda y.t'\{x/u\}) & = \\
\mathtt{AR}_{\mathcal{A}}((\lambda y.t')\{x/u\}) &
\end{array}$$

– Otherwise

$$\begin{array}{ll}
\mathcal{C}_{\Gamma}^{\Delta|\Pi}((R_{\Delta}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(\lambda y.t')))\{x/R_{\Pi}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(u))\}) & = \\
\mathcal{C}_{\Gamma}^{\Delta|\Pi}(\lambda y.(R_{\Delta}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(t')))\{x/R_{\Pi}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(u))\}) & \rightarrow_{\mathtt{CL}} \\
\lambda y.\mathcal{C}_{\Gamma}^{\Delta|\Pi}(R_{\Delta}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(t'))\{x/R_{\Pi}^{\Gamma}(\mathtt{AR}_{\mathcal{A}}(u))\}) & \rightarrow_{\mathcal{A}}^{*} (i.h.) \\
\lambda y.\mathtt{AR}_{\mathcal{A}}(t\{x/u\}) & = \\
\mathtt{AR}_{\mathcal{A}}((\lambda y.t')\{x/u\})
\end{array}$$

- $t = v\ w$. Then by α-equivalence we can suppose $x \notin \mathtt{fv}(u)$. Let us consider the following names for the sets of free variables of the terms under consideration.



Note that $\Phi = \mathtt{fv}(t) \cap \mathtt{fv}(u)$ is a permutation of Σ, Λ, Ψ.
Also note that $\mathtt{fv}(v) \cap \mathtt{fv}(w)$ is a permutation of Σ, Ξ and hence

$$\mathtt{AR}_{\mathcal{A}}(t) \equiv \mathcal{C}_{\Sigma,\Xi}^{\Sigma_3,\Xi_3|\Sigma_4,\Xi_4}(R_{\Sigma_3,\Xi_3}^{\Sigma,\Xi}(\mathtt{AR}_{\mathcal{A}}(v))R_{\Sigma_4,\Xi_4}^{\Sigma,\Xi}(\mathtt{AR}_{\mathcal{A}}(w)))$$

We then have:

$$\begin{array}{ll}
 & \mathcal{C}_{\Sigma,\Lambda,\Psi}^{\Sigma_1,\Lambda_1,\Psi_1|\Sigma_2,\Lambda_2,\Psi_2}(R_{\Sigma_1,\Lambda_1,\Psi_1}^{\Sigma,\Lambda,\Psi}(\mathtt{AR}_{\mathcal{A}}(t))\{x/R_{\Sigma_2,\Lambda_2,\Psi_2}^{\Sigma,\Lambda,\Psi}(\mathtt{AR}_{\mathcal{A}}(u))\}) \\
= & \mathcal{C}_{\Sigma,\Lambda,\Psi}^{\Sigma_1,\Lambda_1,\Psi_1|\Sigma_2,\Lambda_2,\Psi_2}(\mathcal{C}_{\Sigma_1,\Xi}^{\Sigma_3,\Xi_3|\Sigma_4,\Xi_4}(v'w')\{x/R_{\Sigma_2,\Lambda_2,\Psi_2}^{\Sigma,\Lambda,\Psi}(\mathtt{AR}_{\mathcal{A}}(u))\}) \\
= & H
\end{array}$$

where $v' = R_{\Lambda_1,\Sigma_3,\Xi_3}^{\Lambda,\Sigma,\Xi}(\mathtt{AR}_{\mathcal{A}}(v))$ and $w' = R_{\Psi_1,\Sigma_4,\Xi_4}^{\Psi,\Sigma,\Xi}(\mathtt{AR}_{\mathcal{A}}(w))$.

– If $x \in \mathtt{fv}(v) \cap \mathtt{fv}(w)$, then x is in Ξ (since $x \notin \mathtt{fv}(u)$), so Ξ is a permutation of $\Xi'; x$ for some list Ξ'. Hence $\mathcal{C}_{\Sigma_1,\Xi}^{\Sigma_3,\Xi_3|\Sigma_4,\Xi_4}()$ is equivalent by $\mathtt{CC}_{\mathcal{C}}$ to $\mathcal{C}_{\Sigma_1,\Xi'}^{\Sigma_3,\Xi_3'|\Sigma_4,\Xi_4'}(\mathcal{C}_{x}^{x_3|x_4}())$, where $\Xi_3'; x_3$ and $\Xi_4'; x_4$ are the corresponding permutations of Ξ_3 and Ξ_4, respectively. Noticing that $\mathtt{fv}(u)$ is a permutation of $\Theta, \Sigma, \Lambda, \Psi$, so that

$$H \equiv_{\mathtt{CC}_{\mathcal{C}}} \mathcal{C}_{\Sigma,\Lambda,\Psi}^{\Sigma_1,\Lambda_1,\Psi_1|\Sigma_2,\Lambda_2,\Psi_2}(\mathcal{C}_{\Sigma_1,\Xi'}^{\Sigma_3,\Xi_3'|\Sigma_4,\Xi_4'}(\mathcal{C}_{x}^{x_3|x_4}(v'w'))\{S\})$$

where

$$S = x/R_{\Sigma_2,\Lambda_2,\Psi_2}^{\Sigma,\Lambda,\Psi}(\mathtt{AR}_{\mathcal{A}}(u))$$

Performing substitution S gives :

$$\mathcal{C}_{\Sigma,\Lambda,\Psi}^{\Sigma_1,\Lambda_1,\Psi_1|\Sigma_2,\Lambda_2,\Psi_2}(\mathcal{C}_{\Sigma_1,\Xi'}^{\Sigma_3,\Xi_3'|\Sigma_4,\Xi_4'}(\mathcal{C}_{\Theta,\Sigma_2,\Lambda_2,\Psi_2}^{\Theta_5,\Sigma_5,\Lambda_5,\Psi_5|\Theta_6,\Sigma_6,\Lambda_6,\Psi_6}(H_1)))$$

where H_1 is equal to:

$$(v'w')\{x_3/R^{\Theta,\Sigma,\Lambda,\Psi}_{\Theta_5,\Sigma_5,\Lambda_5,\Psi_5}(\mathtt{AR}_{\mathcal{A}}(u))\}\{x_4/R^{\Theta,\Sigma,\Lambda,\Psi}_{\Theta_6,\Sigma_6,\Lambda_6,\Psi_6}(\mathtt{AR}_{\mathcal{A}}(u))\}$$

$$= v'\{x_3/R^{\Theta,\Sigma,\Lambda,\Psi}_{\Theta_5,\Sigma_5,\Lambda_5,\Psi_5}(\mathtt{AR}_{\mathcal{A}}(u))\}\ w'\{x_4/R^{\Theta,\Sigma,\Lambda,\Psi}_{\Theta_6,\Sigma_6,\Lambda_6,\Psi_6}(\mathtt{AR}_{\mathcal{A}}(u))\}$$

Now we rearrange the contractions:

$$\begin{array}{ll}
& \mathcal{C}^{\Sigma_1,\Lambda_1,\Psi_1|\Sigma_2,\Lambda_2,\Psi_2}_{\Sigma,\Lambda,\Psi}(\mathcal{C}^{\Sigma_3,\Xi'_3|\Sigma_4,\Xi'_4}_{\Sigma_1,\Xi'}(H_2)) \\
& \qquad \text{where } H_2 := \mathcal{C}^{\Theta_5,\Sigma_5,\Lambda_5,\Psi_5|\Theta_6,\Sigma_6,\Lambda_6,\Psi_6}_{\Theta,\Sigma_2,\Lambda_2,\Psi_2}(H_1) \\
\equiv_{\mathtt{CC}_{\mathcal{C}}} & \mathcal{C}^{\Theta_5|\Theta_6}_{\Theta}(\mathcal{C}^{\Xi'_3|\Xi'_4}_{\Xi'}(\mathcal{C}^{\Lambda_1|\Lambda_2}_{\Lambda}(\mathcal{C}^{\Lambda_5|\Lambda_6}_{\Lambda_2}(\mathcal{C}^{\Psi_1|\Psi_2}_{\Psi}(\mathcal{C}^{\Psi_5|\Psi_6}_{\Psi_2}(H_3)))))) \\
& \qquad \text{where } H_3 := \mathcal{C}^{\Sigma_1|\Sigma_2}_{\Sigma}(\mathcal{C}^{\Sigma_3|\Sigma_4}_{\Sigma_1}(\mathcal{C}^{\Sigma_5|\Sigma_6}_{\Sigma_2}(H_1))) \\
\equiv_{\mathtt{CC}_{\mathcal{A}}} & \mathcal{C}^{\Theta_5|\Theta_6}_{\Theta}(\mathcal{C}^{\Xi'_3|\Xi'_4}_{\Xi'}(\mathcal{C}^{\Lambda_2|\Lambda_6}_{\Lambda}(\mathcal{C}^{\Lambda_1|\Lambda_5}_{\Lambda_2}(\mathcal{C}^{\Psi_5|\Psi_2}_{\Psi}(\mathcal{C}^{\Psi_1|\Psi_6}_{\Psi_2}(H_4)))))) \\
& \qquad \text{where } H_4 := \mathcal{C}^{\Sigma_1|\Sigma_2}_{\Sigma}(\mathcal{C}^{\Sigma_3|\Sigma_5}_{\Sigma_1}(\mathcal{C}^{\Sigma_4|\Sigma_6}_{\Sigma_2}(H_1))) \\
\equiv_{\mathtt{CC}_{\mathcal{C}}} & \mathcal{C}^{\Theta_5,\Xi'_3,\Lambda_2,\Psi_5,\Sigma_1|\Theta_6,\Xi'_4,\Lambda_6,\Psi_2,\Sigma_2}_{\Theta,\Xi',\Lambda,\Psi,\Sigma}(\mathcal{C}^{\Lambda_1,\Sigma_3|\Lambda_5,\Sigma_5}_{\Lambda_2,\Sigma_1}(H_5)) \\
& \qquad \text{where } H_5 := \mathcal{C}^{\Psi_1,\Sigma_4|\Psi_6,\Sigma_6}_{\Psi_2,\Sigma_2}(H_1)
\end{array}$$

This term can be reduced by $\mathtt{CA}_{\mathtt{L}}$ and then by $\mathtt{CA}_{\mathtt{R}}$ to

$$H' := \mathcal{C}^{\Theta_5,\Xi'_3,\Lambda_2,\Psi_5,\Sigma_1|\Theta_6,\Xi'_4,\Lambda_6,\Psi_2,\Sigma_2}_{\Theta,\Xi',\Lambda,\Psi,\Sigma}(PQ)$$

$$\begin{array}{ll}
P & := \mathcal{C}^{\Lambda_1,\Sigma_3|\Lambda_5,\Sigma_5}_{\Lambda_2,\Sigma_1}(v'\{x_3/R^{\Theta,\Sigma,\Lambda,\Psi}_{\Theta_5,\Sigma_5,\Lambda_5,\Psi_5}(\mathtt{AR}_{\mathcal{A}}(u))\}) \\
& = R^{\Theta,\Xi',\Lambda,\Psi,\Sigma}_{\Theta_5,\Xi'_3,\Lambda_2,\Psi_5,\Sigma_1}(\mathcal{C}^{\Lambda_1,\Sigma_3|\Lambda_5,\Sigma_5}_{\Lambda,\Sigma}(R^{\Lambda,\Sigma}_{\Lambda_1,\Sigma_3}(R^{x}_{x_3}(\mathtt{AR}_{\mathcal{A}}(v))))\{S_P\})
\end{array}$$

and where

$$S_P = x_3/R^{\Sigma,\Lambda}_{\Sigma_5,\Lambda_5}(\mathtt{AR}_{\mathcal{A}}(u))$$

$$\begin{array}{ll}
Q & := \mathcal{C}^{\Psi_1,\Sigma_4|\Psi_6,\Sigma_6}_{\Psi_2,\Sigma_2}(w'\{x_4/R^{\Theta,\Sigma,\Lambda,\Psi}_{\Theta_6,\Sigma_6,\Lambda_6,\Psi_6}(\mathtt{AR}_{\mathcal{A}}(u))\}) \\
& = R^{\Theta,\Xi',\Lambda,\Psi,\Sigma}_{\Theta_6,\Xi'_4,\Lambda_6,\Psi_2,\Sigma_2}(\mathcal{C}^{\Psi_1,\Sigma_4|\Psi_6,\Sigma_6}_{\Psi,\Sigma}(R^{\Psi,\Sigma}_{\Psi_1,\Sigma_4}(R^{x}_{x_4}(\mathtt{AR}_{\mathcal{A}}(w))))\{S_Q\})
\end{array}$$

and where

$$S_Q = x_4/R^{\Sigma,\Psi}_{\Sigma_6,\Psi_6}(\mathtt{AR}_{\mathcal{A}}(u))$$

We can now apply the i.h. to both subterms and we get:

$$\begin{array}{ll}
P & \to^*_{\mathcal{A}} P' = R^{\Theta,\Xi',\Lambda,\Psi,\Sigma}_{\Theta_5,\Xi'_3,\Lambda_2,\Psi_5,\Sigma_1}(\mathtt{AR}_{\mathcal{A}}(v\{x/u\})) \\
Q & \to^*_{\mathcal{A}} Q' := R^{\Theta,\Xi',\Lambda,\Psi,\Sigma}_{\Theta_6,\Xi'_4,\Lambda_6,\Psi_2,\Sigma_2}(\mathtt{AR}_{\mathcal{A}}(w\{x/u\}))
\end{array}$$

So H' reduces to

$$\mathcal{C}^{\Theta_5,\Xi'_3,\Lambda_2,\Psi_5,\Sigma_1|\Theta_6,\Xi'_4,\Lambda_6,\Psi_2,\Sigma_2}_{\Theta,\Xi',\Lambda,\Psi,\Sigma}(P'Q')$$

which is $\mathtt{AR}_{\mathcal{A}}(v\{x/u\}w\{x/u\}) = \mathtt{AR}_{\mathcal{A}}((v\ w)\{x/u\})$.

– If $x \in \mathtt{fv}(v)$ et $x \notin \mathtt{fv}(w)$, the term H can be transformed to:

$$\begin{array}{ll}
 & \mathcal{C}^{\Sigma_1,\Lambda_1,\Psi_1|\Sigma_2,\Lambda_2,\Psi_2}_{\Sigma,\Lambda,\Psi}(\mathcal{C}^{\Sigma_3,\Xi_3|\Sigma_4,\Xi_4}_{\Sigma_1,\Xi}((v'\ w')\{x/S_x\})) \\
 & \qquad \text{with } S_x = R^{\Sigma,\Lambda,\Psi}_{\Sigma_2,\Lambda_2,\Psi_2}(\mathtt{AR}_{\mathcal{A}}(u)) \\
= & \mathcal{C}^{\Sigma_1,\Lambda_1,\Psi_1|\Sigma_2,\Lambda_2,\Psi_2}_{\Sigma,\Lambda,\Psi}(\mathcal{C}^{\Sigma_3,\Xi_3|\Sigma_4,\Xi_4}_{\Sigma_1,\Xi}(v'\{x/S_x\}\ w')) \\
\equiv_{\mathtt{CC}_{\mathcal{A}},\mathtt{CC}_{\mathcal{C}}} & \mathcal{C}^{\Sigma_1,\Psi_2,\Xi_3|\Sigma_4,\Psi_1,\Xi_4}_{\Sigma,\Psi,\Xi}(\mathcal{C}^{\Sigma_3,\Lambda_1|\Sigma_2,\Lambda_2}_{\Sigma_1,\Lambda}(v'\{x/S_x\}\ w')) \\
\to_{\mathtt{CA}_{\mathtt{L}}} & \mathcal{C}^{\Sigma_1,\Psi_2,\Xi_3|\Sigma_4,\Psi_1,\Xi_4}_{\Sigma,\Psi,\Xi}(\mathcal{C}^{\Sigma_3,\Lambda_1|\Sigma_2,\Lambda_2}_{\Sigma_1,\Lambda}(v'\{x/S_x\})\ w') \\
= & \mathcal{C}^{\Sigma_1,\Psi_2,\Xi_3|\Sigma_4,\Psi_1,\Xi_4}_{\Sigma,\Psi,\Xi}(R^{\Sigma,\Psi,\Xi}_{\Sigma_1,\Psi_2,\Xi_3}(V)\ R^{\Sigma,\Psi,\Xi}_{\Sigma_4,\Psi_1,\Xi_4}(\mathtt{AR}_{\mathcal{A}}(w))) \\
= & H'
\end{array}$$

where

$$V := \mathcal{C}^{\Sigma_3,\Lambda_1|\Sigma_2,\Lambda_2}_{\Sigma,\Lambda}(R^{\Lambda,\Sigma}_{\Lambda_1,\Sigma_3}(\mathtt{AR}_{\mathcal{A}}(v))\{x/R^{\Sigma,\Lambda}_{\Sigma_2,\Lambda_2}(\mathtt{AR}_{\mathcal{A}}(u))\})$$

which reduces by the i.h. to $\mathtt{AR}_{\mathcal{A}}(v\{x/u\})$. Hence,

$$H' \to^*_{\mathcal{A}} \mathcal{C}^{\Sigma_1,\Psi_2,\Xi_3|\Sigma_4,\Psi_1,\Xi_4}_{\Sigma,\Psi,\Xi}(R^{\Sigma,\Psi,\Xi}_{\Sigma_1,\Psi_2,\Xi_3}(\mathtt{AR}_{\mathcal{A}}(v\{x/u\}))R^{\Sigma,\Psi,\Xi}_{\Sigma_4,\Psi_1,\Xi_4}(\mathtt{AR}_{\mathcal{A}}(w)))$$

which is exactly $\mathtt{AR}_{\mathcal{A}}(v\{x/u\}w) = \mathtt{AR}_{\mathcal{A}}((vw)\{x/u\})$.

– If $x \in \mathtt{fv}(v)$ et $x \notin \mathtt{fv}(w)$ the proof is symmetric.

– The case $x \notin \mathtt{fv}(v)$ and $x \notin \mathtt{fv}(w)$ cannot happen since we assumed $x \in \mathtt{fv}(t)$.

□

For instance if $\mathtt{c} \in \mathcal{A}$, $t = (z\ x)\ z$ and $u = z$, then:

$$\begin{array}{ll}
\mathcal{C}^{z_3|z_4}_{z}(R^{z}_{z_3}(\mathtt{AR}_{\mathcal{A}}((z\ x)\ z))\{x/z_4\}) & = \\
\mathcal{C}^{z_3|z_4}_{z}(\mathcal{C}^{z_1|z_2}_{z_3}((z_1\ x)\ z_2)\{x/z_4\}) & = \\
\mathcal{C}^{z_3|z_4}_{z}(\mathcal{C}^{z_1|z_2}_{z_3}((z_1\ z_4)\ z_2)) & \equiv \\
\mathcal{C}^{z_3|z_2}_{z}(\mathcal{C}^{z_1|z_4}_{z_3}((z_1\ z_4)\ z_2)) & \to_{\mathtt{CA}_{\mathtt{L}}} \\
\mathcal{C}^{z_3|z_2}_{z}(\mathcal{C}^{z_1|z_4}_{z_3}(z_1\ z_4)\ z_2) & = \\
\mathcal{C}^{z_3|z_2}_{z}(R^{z}_{z_3}(\mathtt{AR}_{\mathcal{A}}(z\ z))\ z_2) & = \\
\mathtt{AR}_{\mathcal{A}}((z\ z)\ z)) &
\end{array}$$

Theorem 5.2.4 (Simulation (i))**.** *Let $t \in \mathcal{T}_{\emptyset}$ such that $t \to_{\emptyset} t'$. Let $\mathcal{A} \subseteq \mathcal{R}$.*

- *If $\mathtt{w} \in \mathcal{A}$, then $\mathtt{AR}_{\mathcal{A}}(t) \to^+_{\mathcal{A}} \mathcal{W}_{\mathtt{fv}(t)\setminus\mathtt{fv}(t')}(\mathtt{AR}_{\mathcal{A}}(t'))$.*
- *If $\mathtt{w} \notin \mathcal{A}$, then $\mathtt{AR}_{\mathcal{A}}(t) \to^+_{\mathcal{A}} \mathtt{AR}_{\mathcal{A}}(t')$.*

Proof. By induction on the reduction relation $\to_{\beta}$ using Lemma 5.2.3.

- The root case $t = (\lambda x.t_1)\ u \to_{\beta} t_1\{x/u\} = t'$ is done using Lemmas 5.2.2 and 5.2.3:

- $\mathcal{A} = \{\mathtt{w}\}$
  - $x \in \mathtt{fv}(t_1)$

$$\begin{array}{ll} \mathtt{AR}_{\mathcal{A}}(t) & = \\ \lambda x.\mathtt{AR}_{\mathcal{A}}(t_1)\ \mathtt{AR}_{\mathcal{A}}(u) & \to_\beta \\ \mathtt{AR}_{\mathcal{A}}(t_1)\{x/\mathtt{AR}_{\mathcal{A}}(u)\} & =_{L.\ 5.2.3}\ \mathtt{AR}_{\mathcal{A}}(t_1\{x/u\}) \end{array}$$

  - $x \notin \mathtt{fv}(t_1)$

$$\begin{array}{ll} \mathtt{AR}_{\mathcal{A}}(t) & = \\ \lambda x.\mathcal{W}_x(\mathtt{AR}_{\mathcal{A}}(t_1))\ \mathtt{AR}_{\mathcal{A}}(u) & \to_\beta \\ \mathcal{W}_x(\mathtt{AR}_{\mathcal{A}}(t_1))\{x/\mathtt{AR}_{\mathcal{A}}(u)\} & = \\ \mathcal{W}_{\mathtt{fv}(\mathtt{AR}_{\mathcal{A}}(u))\setminus\mathtt{fv}(\mathtt{AR}_{\mathcal{A}}(t_1))}(\mathtt{AR}_{\mathcal{A}}(t_1)) & =_{(L.\ 5.2.2)} \\ \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t_1)}(\mathtt{AR}_{\mathcal{A}}(t_1)) & = \end{array}$$

    The statement is then true since

$$\begin{array}{ll} \mathtt{fv}((\lambda x.t_1)\ u) \setminus \mathtt{fv}(t_1\{x/u\}) & = \\ (\mathtt{fv}(t_1) \setminus x \cup \mathtt{fv}(u)) \setminus \mathtt{fv}(t_1) & = \\ \mathtt{fv}(u) \setminus \mathtt{fv}(t_1) & = \end{array}$$

- $\mathcal{A} = \{\mathtt{c}\}$.
  Let $\Gamma = \mathtt{fv}(\lambda x.t_1) \cap \mathtt{fv}(u)$.

$$\begin{array}{ll} \mathtt{AR}_{\mathcal{A}}(t) & = \\ \mathcal{C}_\Gamma^{\Delta|\Pi}(R_\Delta^\Gamma(\mathtt{AR}_{\mathcal{A}}(\lambda x.t_1))\ R_\Pi^\Gamma(\mathtt{AR}_{\mathcal{A}}(u))) & = \\ \mathcal{C}_\Gamma^{\Delta|\Pi}(R_\Delta^\Gamma(\lambda x.\mathtt{AR}_{\mathcal{A}}(t_1))\ R_\Pi^\Gamma(\mathtt{AR}_{\mathcal{A}}(u))) & = \\ \mathcal{C}_\Gamma^{\Delta|\Pi}(\lambda x.R_\Delta^\Gamma(\mathtt{AR}_{\mathcal{A}}(t_1))\ R_\Pi^\Gamma(\mathtt{AR}_{\mathcal{A}}(u))) & \to_\beta \\ \mathcal{C}_\Gamma^{\Delta|\Pi}(R_\Delta^\Gamma(\mathtt{AR}_{\mathcal{A}}(t_1))\{x/R_\Pi^\Gamma(\mathtt{AR}_{\mathcal{A}}(u))\}) & \to_{\mathcal{A}}^* \ (L.\ 5.2.3) \\ \mathtt{AR}_{\mathcal{A}}(t_1\{x/u\}) & \end{array}$$

- $\mathcal{A} = \{\mathtt{c}, \mathtt{w}\}$
  - The case $x \in \mathtt{fv}(t_1)$ is the same as $\mathcal{A} = \mathtt{c}$.
  - $x \notin \mathtt{fv}(t_1)$

$$\begin{array}{ll} \mathtt{AR}_{\mathcal{A}}(t) & = \\ \mathcal{C}_\Gamma^{\Delta|\Pi}(R_\Delta^\Gamma(\mathtt{AR}_{\mathcal{A}}(\lambda x.\mathcal{W}_x(t_1)))\ R_\Pi^\Gamma(\mathtt{AR}_{\mathcal{A}}(u))) & = \\ \mathcal{C}_\Gamma^{\Delta|\Pi}(R_\Delta^\Gamma(\lambda x.\mathcal{W}_x(\mathtt{AR}_{\mathcal{A}}(t_1)))\ R_\Pi^\Gamma(\mathtt{AR}_{\mathcal{A}}(u))) & = \\ \mathcal{C}_\Gamma^{\Delta|\Pi}(\lambda x.\mathcal{W}_x(R_\Delta^\Gamma(\mathtt{AR}_{\mathcal{A}}(t_1)))\ R_\Pi^\Gamma(\mathtt{AR}_{\mathcal{A}}(u))) & \to_\beta \\ \mathcal{C}_\Gamma^{\Delta|\Pi}(\mathcal{W}_x(R_\Delta^\Gamma(\mathtt{AR}_{\mathcal{A}}(t_1)))\{x/R_\Pi^\Gamma(\mathtt{AR}_{\mathcal{A}}(u))\}) & = (x \notin \Gamma) \\ \mathcal{C}_\Gamma^{\Delta|\Pi}(R_\Delta^\Gamma(\mathtt{AR}_{\mathcal{A}}(\mathcal{W}_x(t_1)))\{x/R_\Pi^\Gamma(\mathtt{AR}_{\mathcal{A}}(u))\}) & \to^* \ (L.\ 5.2.3) \\ \mathtt{AR}_{\mathcal{A}}(\mathcal{W}_x(t_1)\{x/u\}) & = \\ \mathtt{AR}_{\mathcal{A}}(\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t_1)}(t_1)) & = \\ \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t_1)}(\mathtt{AR}_{\mathcal{A}}(t_1)) & = \end{array}$$

The statement is then true since

$$\begin{array}{ll} \mathtt{fv}((\lambda x.t_1)\ u) \setminus \mathtt{fv}(t_1\{x/u\}) & = \\ (\mathtt{fv}(t_1) \setminus x \cup \mathtt{fv}(u)) \setminus \mathtt{fv}(t_1) & = \\ \mathtt{fv}(u) \setminus \mathtt{fv}(t_1) & \end{array}$$

- If $\lambda x.u \Rightarrow_\beta \lambda x.u'$ with $u \Rightarrow_\beta u'$, then we only consider the case $\mathtt{w} \in \mathcal{A}$ as the other ones are straightforward.

  - If $x \notin \mathtt{fv}(u)$, then

$$\begin{array}{lll} \mathtt{AR}_{\mathcal{A}}(\lambda x.u) & = & \lambda x.\mathcal{W}_x(\mathtt{AR}_{\mathcal{A}}(u)) \\ & \to^+_{\mathcal{A}\ (i.h.)} & \lambda x.\mathcal{W}_x(\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(u')}(\mathtt{AR}_{\mathcal{A}}(u'))) \\ & = & \lambda x.\mathcal{W}_x(\mathcal{W}_{\mathtt{fv}(\lambda x.u)\setminus\mathtt{fv}(\lambda x.u')}(\mathtt{AR}_{\mathcal{A}}(u'))) \\ & \equiv_{\mathtt{WW}_{\mathcal{C}}} & \lambda x.\mathcal{W}_{\mathtt{fv}(\lambda x.u)\setminus\mathtt{fv}(\lambda x.u')}(\mathcal{W}_x(\mathtt{AR}_{\mathcal{A}}(u'))) \\ & \to^*_{\mathtt{LW}} & \mathcal{W}_{\mathtt{fv}(\lambda x.u)\setminus\mathtt{fv}(\lambda x.u')}(\lambda x.\mathcal{W}_x(\mathtt{AR}_{\mathcal{A}}(u'))) \end{array}$$

  - If $x \in \mathtt{fv}(u)$, then

$$\begin{array}{lll} \mathtt{AR}_{\mathcal{A}}(\lambda x.u) & = & \lambda x.\mathtt{AR}_{\mathcal{A}}(u) \\ & \to^+_{\mathcal{A}\ (i.h.)} & \lambda x.\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(u')}(\mathtt{AR}_{\mathcal{A}}(u')) \\ & = & \lambda x.\mathcal{W}_{\mathtt{fv}(\lambda x.u)\setminus\mathtt{fv}(u')}(\mathcal{W}_{x\setminus\mathtt{fv}(u')}(\mathtt{AR}_{\mathcal{A}}(u'))) \\ & = & \lambda x.\mathcal{W}_{\mathtt{fv}(\lambda x.u)\setminus\mathtt{fv}(\lambda x.u')}(\mathcal{W}_{x\setminus\mathtt{fv}(u')}(\mathtt{AR}_{\mathcal{A}}(u'))) \\ & \to^*_{\mathtt{LW}} & \mathcal{W}_{\mathtt{fv}(\lambda x.u)\setminus\mathtt{fv}(\lambda x.u')}(\lambda x.\mathcal{W}_{x\setminus\mathtt{fv}(u')}(\mathtt{AR}_{\mathcal{A}}(u'))) \end{array}$$

- If $uv \Rightarrow_\beta u'v$ with $u \Rightarrow_\beta u'$, we only consider the case where $\mathtt{c} \in \mathcal{A}$ as the other is straightforward.

  Let consider the following names:

$$\begin{array}{lll} \Sigma & = & \mathtt{fv}(u') \cap \mathtt{fv}(v) \\ \Lambda & = & \mathtt{fv}(u') \setminus (\mathtt{fv}(u') \cap \mathtt{fv}(v)) \\ \Psi & = & (\mathtt{fv}(u) \cap \mathtt{fv}(v)) \setminus \mathtt{fv}(u') \\ \Xi & = & (\mathtt{fv}(u) \setminus \mathtt{fv}(u')) \setminus \mathtt{fv}(v) \end{array}$$

  Note in particular that $\mathtt{fv}(u) \cap \mathtt{fv}(v)$ is a permutation of Σ, Ψ. Correspondingly, let Σ_l, Ψ_l and Σ_r, Ψ_r be fresh variables.

  We have:

$$
\begin{array}{ll}
 & \mathtt{AR}_{\mathcal{A}}(u\ v) \\
\equiv & \mathcal{C}^{\Sigma_l,\Psi_l|\Sigma_r,\Psi_r}_{\Sigma,\Psi}(R^{\Sigma,\Psi}_{\Sigma_l,\Psi_l}(\mathtt{AR}_{\mathcal{A}}(u))\ R^{\Sigma,\Psi}_{\Sigma_r,\Psi_r}(\mathtt{AR}_{\mathcal{A}}(v))) \\
\to^{+}_{\mathcal{A}\ (i.h.)} & \mathcal{C}^{\Sigma_l,\Psi_l|\Sigma_r,\Psi_r}_{\Sigma,\Psi}(R^{\Sigma,\Psi}_{\Sigma_l,\Psi_l}(\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(u')}(\mathtt{AR}_{\mathcal{A}}(u')))\ R^{\Sigma,\Psi}_{\Sigma_r,\Psi_r}(\mathtt{AR}_{\mathcal{A}}(v))) \\
\equiv_{\mathtt{WW}_{\mathcal{C}}} & \mathcal{C}^{\Sigma_l,\Psi_l|\Sigma_r,\Psi_r}_{\Sigma,\Psi}(R^{\Sigma,\Psi}_{\Sigma_l,\Psi_l}(\mathcal{W}_{\Xi,\Psi}(\mathtt{AR}_{\mathcal{A}}(u')))R^{\Sigma,\Psi}_{\Sigma_r,\Psi_r}(\mathtt{AR}_{\mathcal{A}}(v))) \\
= & \mathcal{C}^{\Sigma_l,\Psi_l|\Sigma_r,\Psi_r}_{\Sigma,\Psi}(\mathcal{W}_{\Xi}(\mathcal{W}_{\Psi_l}(R^{\Sigma}_{\Sigma_l}(\mathtt{AR}_{\mathcal{A}}(u'))))R^{\Sigma,\Psi}_{\Sigma_r,\Psi_r}(\mathtt{AR}_{\mathcal{A}}(v))) \\
\to^{*}_{\mathtt{AW}_1} & \mathcal{C}^{\Sigma_l,\Psi_l|\Sigma_r,\Psi_r}_{\Sigma,\Psi}(\mathcal{W}_{\Xi\setminus R^{\Sigma,\Psi}_{\Sigma_r,\Psi_r}(\mathtt{fv}(v))}(t')) \\
 & \text{where } t' = \mathcal{W}_{\Psi_l\setminus R^{\Sigma,\Psi}_{\Sigma_r,\Psi_r}(\mathtt{fv}(v))}(R^{\Sigma}_{\Sigma_l}(\mathtt{AR}_{\mathcal{A}}(u'))R^{\Sigma,\Psi}_{\Sigma_r,\Psi_r}(\mathtt{AR}_{\mathcal{A}}(v))) \\
= & \mathcal{C}^{\Sigma_l,\Psi_l|\Sigma_r,\Psi_r}_{\Sigma,\Psi}(\mathcal{W}_{\Xi}(\mathcal{W}_{\Psi_l}(R^{\Sigma}_{\Sigma_l}(\mathtt{AR}_{\mathcal{A}}(u'))R^{\Sigma,\Psi}_{\Sigma_r,\Psi_r}(\mathtt{AR}_{\mathcal{A}}(v))))) \\
\to^{*}_{\mathtt{CW}_2} & \mathcal{W}_{\Xi}(\mathcal{C}^{\Sigma_l,\Psi_l|\Sigma_r,\Psi_r}_{\Sigma,\Psi}(\mathcal{W}_{\Psi_l}(R^{\Sigma}_{\Sigma_l}(\mathtt{AR}_{\mathcal{A}}(u'))R^{\Sigma,\Psi}_{\Sigma_r,\Psi_r}(\mathtt{AR}_{\mathcal{A}}(v))))) \\
\to^{*}_{\mathtt{CW}_1} & \mathcal{W}_{\Xi}(\mathcal{C}^{\Sigma_l|\Sigma_r}_{\Sigma}(R^{\Psi_r}_{\Psi}(R^{\Sigma}_{\Sigma_l}(\mathtt{AR}_{\mathcal{A}}(u'))R^{\Sigma,\Psi}_{\Sigma_r,\Psi_r}(\mathtt{AR}_{\mathcal{A}}(v))))) \\
= & \mathcal{W}_{\Xi}(\mathcal{C}^{\Sigma_l|\Sigma_r}_{\Sigma}(R^{\Sigma}_{\Sigma_l}(\mathtt{AR}_{\mathcal{A}}(u'))R^{\Sigma}_{\Sigma_r}(\mathtt{AR}_{\mathcal{A}}(v))))
\end{array}
$$

Then it suffices to notice that $\Xi = \mathtt{fv}(uv) \setminus \mathtt{fv}(u'v)$.

- The case $uv \Rightarrow_{\beta} uv'$ is similar to the previous one.

□

For instance, if $t = (\lambda z.y)\ w \to_{\beta} y = t'$ then $\mathtt{AR}_{\mathcal{A}}(t) = (\lambda z.\mathcal{W}_z(y))\ w \to_{\beta} \mathcal{W}_w(y) = \mathcal{W}_{\mathtt{fv}(t)\setminus\mathtt{fv}(t')}(\mathtt{AR}_{\mathcal{A}}(t'))$.

Since meta-level substitution can also be simulated by the explicit one by Lemma 5.1.22, then we obtain a more general simulation result.

Corollary 5.2.5 (Simulation (ii))**.** *Let $t \in \mathcal{T}_{\emptyset}$ such that $t \to_{\emptyset} t'$. Let $\mathcal{B} = \mathcal{A} \cup \{\mathtt{s}\}$, where $\mathcal{A} \subseteq \mathcal{R}$.*

- *If $\mathtt{w} \in \mathcal{A}$, then $\mathtt{AR}_{\mathcal{A}}(t) \to^{+}_{\mathcal{B}} \mathcal{W}_{\mathtt{fv}(t)\setminus\mathtt{fv}(t')}(\mathtt{AR}_{\mathcal{A}}(t'))$.*
- *If $\mathtt{w} \notin \mathcal{A}$, then $\mathtt{AR}_{\mathcal{A}}(t) \to^{+}_{\mathcal{B}} \mathtt{AR}_{\mathcal{A}}(t')$.*

For instance, if $t = (\lambda z.y)\ w \to_{\beta} y = t'$ then $\mathtt{AR}_{\mathtt{w}}(t) = (\lambda z.\mathcal{W}_z(y))\ w \to_{\mathtt{sw}} \mathcal{W}_w(y) = \mathcal{W}_{\mathtt{fv}(t)\setminus\mathtt{fv}(t')}(\mathtt{AR}_{\mathtt{w}}(t'))$.

While Corollary 5.2.5 states that adding resources to the $\lambda_{\emptyset}$-calculus is well behaved, this does not necessarily hold for *any* arbitrary calculus of the prismoid. Thus for example, what happens when the $\lambda_{\mathtt{s}}$-calculus is enriched with resource $\mathtt{w}$? Is it possible to simulate each $\mathtt{s}$-reduction step by a sequence of $\mathtt{sw}$-reduction steps? Unfortunately the answer is no: suppose the function $\mathtt{AR}_{\mathcal{A}}(\_)$ is extended to $\mathtt{s}$-terms in a natural way; then we have $t_1 = (x\ y)[z/v] \to_{\mathtt{s}} x\ y[z/v] = t_2$ but $\mathtt{AR}_{\mathtt{w}}(t_1) = \mathcal{W}_z(x\ y)[z/v] \not\to_{\mathtt{sw}} x\ \mathcal{W}_z(y)[z/v] = \mathtt{AR}_{\mathtt{w}}(t_2)$.

5.3 Removing Resources

In this section we give a mechanism to remove resources, that is, to change the status of weakening and/or contraction from explicit to implicit. This is dual to

the operation adding resources to terms presented in Section 5.2. Whereas adding is only defined within the implicit base, removing is defined in both bases. As adding, removing is not only done on a static level, but also on a dynamic one. Thus for example, removing translates any $\mathtt{csw}$-reduction sequence into a $\mathcal{B}$-reduction sequence, for any $\mathcal{B} \in \{\mathtt{s}, \mathtt{cs}, \mathtt{sw}\}$.

Definition 5.3.1 (Collapsing function)**.** *We first define the collapsing function* $\mathtt{S}^{\Gamma}_{z}(\_)$ *of a well-formed term* t *without contractions s.t.* $z \notin \mathtt{fv}(t)$ *as follows:*

$$
\begin{array}{lcl}
\mathtt{S}^{\Gamma}_{z}(w) & := & \begin{cases} w & \text{if } w \notin \Gamma \\ z & \text{if } w \in \Gamma \end{cases} \\
\mathtt{S}^{\Gamma}_{z}(uv) & := & \mathtt{S}^{\Gamma}_{z}(u)\mathtt{S}^{\Gamma}_{z}(v) \\
\mathtt{S}^{\Gamma}_{z}(\lambda w.u) & := & \lambda w.\mathtt{S}^{\Gamma}_{z}(u), \ \text{if } w \notin \Gamma \\
\mathtt{S}^{\Gamma}_{z}(u[w/v]) & := & \mathtt{S}^{\Gamma}_{z}(u)[w/\mathtt{S}^{\Gamma}_{z}(v)], \ \text{if } w \notin \Gamma \\
\mathtt{S}^{\Gamma}_{z}(\mathcal{W}_w(v)) & := & \begin{cases} \mathtt{S}^{\Gamma}_{z}(v) & \mathtt{S}^{\Gamma}_{z}(w) \in \mathtt{fv}(\mathtt{S}^{\Gamma}_{z}(v)) \\ \mathcal{W}_{\mathtt{S}^{\Gamma}_{z}(w)}(\mathtt{S}^{\Gamma}_{z}(v)) & \text{otherwise} \end{cases}
\end{array}
$$

The collapsing function renames the variables of a term by removing also the weakened ones that do not respect well-formedness. Indeed, if $\mathcal{W}_x(u)$ appears in the image term, then $x \notin \mathtt{fv}(u)$. Thus for example $\mathtt{S}^{y,z}_{x}(\mathcal{W}_y(\mathcal{W}_z(x))) = x$.

Lemma 5.3.2. *Let* $\mathtt{c} \notin \mathcal{B}$ *and* $t \in \mathcal{T}_{\mathcal{B}}$*. Then,*

1. $\mathtt{S}^{\Gamma}_{z}(t) = t$ *if* $\Gamma \cap \mathtt{fv}(t) = \emptyset$*.*
2. $\mathtt{S}^{x,y}_{z}(t) = R^{x}_{z}(t)$ *if* $y \notin \mathtt{fv}(t)$*.*
3. $\mathtt{del}_x(\mathtt{S}^{x_1,x_2}_{x}(t)) = \mathtt{S}^{x_1,x_2}_{x}(\mathtt{del}_{x_1,x_2}(t))$*.*
4. $\mathtt{S}^{x,x_3}_{z}(\mathtt{S}^{x_1,x_2}_{x}(t)) = \mathtt{S}^{x_1,x_2,x_3}_{z}(t)$*.*
5. $\mathtt{S}^{x_3,x_4}_{z}(\mathtt{S}^{x_1,x_2}_{x}(t)) = \mathtt{S}^{x_1,x_2}_{x}(\mathtt{S}^{x_3,x_4}_{z}(t))$ *if* $x \neq x_3, x_4$*.*
6. $\mathtt{S}^{\Gamma}_{z}(t)_{x \overset{+}{\rightsquigarrow} y} = \mathtt{S}^{\Gamma}_{z}(t_{x \overset{+}{\rightsquigarrow} y})$ *if* $x, y \notin \Gamma, z$*.*

Proof. All the statements are straightforward by induction on $\mathtt{size}(t)$. □

Definition 5.3.3. *A well-formed term* t *is said to be well-signed iff for every variable* $x \in \mathtt{fv}(t)$*,* $x \in \mathtt{fv}^{+}(t)$ *implies* $|t|_x = |t|^{+}_{x}$*.*

Thus for example, $\mathcal{W}_x(y)\mathcal{W}_x(z)$ and $x(yx)$ are well-signed while $\mathcal{W}_x(y)x$ does not.

Lemma 5.3.4. *Let* $\mathtt{c} \notin \mathcal{B}$*. Suppose* $t, u \in \mathcal{T}_{\mathcal{B}}$ *are well-signed. Then,*

1. $\mathtt{del}_{\Gamma}(t) = t$ *if* $x \in \Gamma$ *implies* $x \in \mathtt{fv}^{+}(t)$*.*
2. $\mathtt{S}^{\Gamma,y}_{z}(t) = R^{y}_{z}(\mathtt{S}^{\Gamma}_{z}(t))$ *if* $y \in \mathtt{fv}^{+}(t)$*.*
3. $\mathtt{del}_x(\mathtt{S}^{\Gamma}_{z}(t)) = \mathtt{S}^{\Gamma}_{z}(\mathtt{del}_x(t))$ *with* $x \notin \Gamma$ *and* $x \neq z$*.*

4. $\mathtt{S}_x^{y,z}(t)\{x/u\} = t\{\!\{y/u\}\!\}\{\!\{z/u\}\!\}$ *if* $(|t|_y^+ \geq 1$ *or* $|t|_z^+ \geq 1)$ *and* $\mathtt{fv}(t) \cap \mathtt{fv}(u) = \emptyset$.

5. $\mathtt{S}_z^{x,y}(t\{w/u\}) = \mathtt{S}_z^{x,y}(t)\{w/\mathtt{S}_z^{x,y}(u)\}$ *if* $\mathtt{fv}(t) \cap \mathtt{fv}(u) = \emptyset$ *and* x, y *cannot be both in* t *or in* u.

6. *If* $t \to_{\mathcal{B}} t'$, *then* $\mathtt{S}_z^{\Gamma}(t) \to_{\mathcal{B}} \mathtt{S}_z^{\Gamma}(t')$.

Proof. All the properties can be shown by induction on $\mathtt{size}(t)$, except the last one which can be shown by induction on the reduction relation.

□

The function $\mathtt{RR}_{\mathcal{A}}(\_) : \mathcal{T}_{\mathcal{B}} \mapsto \mathcal{T}_{\mathcal{B}\setminus\mathcal{A}}$ removes $\mathcal{A} \subseteq \mathcal{R}$ from a $\mathcal{B}$-term.

$$
\begin{array}{lcll}
\mathtt{RR}_{\mathcal{A}}(x) & := & x & \\
\mathtt{RR}_{\mathcal{A}}(\lambda x.t) & := & \lambda x.\mathtt{RR}_{\mathcal{A}}(t) & \\
\mathtt{RR}_{\mathcal{A}}(t\ u) & := & \mathtt{RR}_{\mathcal{A}}(t)\ \mathtt{RR}_{\mathcal{A}}(u) & \\
\mathtt{RR}_{\mathcal{A}}(t[x/u]) & := & \mathtt{RR}_{\mathcal{A}}(t)[x/\mathtt{RR}_{\mathcal{A}}(u)] & \\
\mathtt{RR}_{\mathcal{A}}(\mathcal{W}_x(t)) & := & \begin{cases} \mathcal{W}_x(\mathtt{RR}_{\mathcal{A}}(t)) & \text{if } \mathtt{w} \notin \mathcal{A} \\ \mathtt{RR}_{\mathcal{A}}(t) & \text{if } \mathtt{w} \in \mathcal{A} \end{cases} & \\
\mathtt{RR}_{\mathcal{A}}(\mathcal{C}_x^{y|z}(t)) & := & \begin{cases} \mathcal{C}_x^{y|z}(\mathtt{RR}_{\mathcal{A}}(t)) & \text{if } \mathtt{c} \notin \mathcal{A} \\ \mathtt{S}_x^{y,z}(\mathtt{del}_{y,z}(\mathtt{RR}_{\mathcal{A}}(t))) & \text{if } \mathtt{c} \in \mathcal{A}\ \&\ x \in \mathtt{fv}^+(\mathcal{C}_x^{y|z}(t)) \\ \mathtt{S}_x^{y,z}(\mathtt{RR}_{\mathcal{A}}(t)) & \text{if } \mathtt{c} \in \mathcal{A}\ \&\ x \notin \mathtt{fv}^+(\mathcal{C}_x^{y|z}(t)) \end{cases} &
\end{array}
$$

It is worth noticing that $\mathtt{RR}_{\mathcal{A}}(t)$ is always a well-signed term when $\mathtt{c} \in \mathcal{A}$.

For example, $\mathtt{RR}_{\mathtt{C}}(\mathcal{C}_x^{x_1|x_2}(\mathcal{C}_y^{y_1|y_2}(\mathcal{W}_{y_1}(\mathcal{W}_{y_2}(x_1)))[y/x_2)) = \mathcal{W}_y(x)[y/x]$ and $\mathtt{RR}_{\mathtt{W}}(\mathcal{W}_x(z_1)\ \mathcal{W}_y(z_2)) = z_1\ z_2$. More interestingly, $\mathtt{RR}_{\mathtt{C}}(\mathcal{C}_{y_2}^{x_1|x_3}(\mathcal{W}_{x_1}(y_1)x_3))$ is y_1y_2 and not $\mathcal{W}_{y_2}(y_1)y_2$. This is because when projecting contractions, we do not want to leave negative variables whose positive occurrences come from the image of the projection. This is particularly useful when projecting a $\mathtt{SCa}$-reduction step. Indeed, let us suppose

$$
\underbrace{\mathcal{C}_x^{y_1|y_2}(\mathcal{C}_{y_2}^{x_1|x_3}(\mathcal{W}_{x_1}(y_1)x_3))[x/z]}_{=\,t_0} \to_{\mathtt{SCa}} \underbrace{\mathcal{C}_z^{z_1|z_2}(\mathcal{C}_{y_2}^{x_1|x_3}(\mathcal{W}_{x_1}(y_1)x_3)[y_1/z_1][y_2/z_2])}_{=\,t_1}
$$

Then, projecting contractions gives

$$
\mathtt{RR}_{\mathtt{C}}(t_0) = (xx)[x/z] \to_{\mathtt{SDup}} (y_1y_2)[y_1/z][y_2/z] = \mathtt{RR}_{\mathtt{C}}(t_1)
$$

Remark that the removing function $\mathtt{RR}_{\mathcal{A}}(\_)$ is the identity if the resources $\mathcal{A}$ to be removed are not in the term, i.e. $\mathtt{RR}_{\mathcal{A}}(t) = t$ if $t \in \mathcal{T}_{\mathcal{B}\setminus\mathcal{A}}$.

The operation $\mathtt{RR}_{\mathcal{A}}(\_)$ enjoys the following properties:

Lemma 5.3.5. *Let* $t \in \mathcal{T}_{\mathcal{B}}$. *Then, for all* $\mathcal{A} \subseteq \mathcal{R}$

1. $R^{\Gamma}_{\Delta}(\mathtt{RR}_{\mathcal{A}}(t)) = \mathtt{RR}_{\mathcal{A}}(R^{\Gamma}_{\Delta}(t))$.
2. $\mathtt{fv}^{+}(\mathtt{RR}_{\mathcal{A}}(t)) = \mathtt{fv}^{+}(t)$.
3. $\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t)) = \mathtt{fv}(t)$ *if* $\mathtt{w} \in \mathcal{B} \setminus \mathcal{A}$, $\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t)) \subseteq \mathtt{fv}(t)$ *otherwise.*
4. $\mathtt{RR}_{\mathcal{A}}(t)_{\overset{+}{x \rightsquigarrow y_1 \ldots y_n}} = \mathtt{RR}_{\mathcal{A}}(t_{\overset{+}{x \rightsquigarrow y_1 \ldots y_n}})$ *if* $\mathtt{c} \notin \mathcal{B}$.
5. $\mathtt{del}_{\Gamma}(\mathtt{RR}_{\mathcal{A}}(t)) = \mathtt{RR}_{\mathcal{A}}(\mathtt{del}_{\Gamma}(t))$.

Proof. By induction on $\mathtt{size}(t)$. □

Lemma 5.3.6. *Let* $t, u \in \mathcal{T}_{\mathcal{B}}$ *and* $\mathcal{A} \subseteq \mathcal{R}$. *If* $t\{x/u\} \in \mathcal{T}_{\mathcal{B}}$, *then* $\mathtt{RR}_{\mathcal{A}}(t\{x/u\}) = \mathtt{RR}_{\mathcal{A}}(t)\{x/\mathtt{RR}_{\mathcal{A}}(u)\}$.

Proof. If $x \notin \mathtt{fv}(t)$ then the property is straightforward so that suppose $x \in \mathtt{fv}(t)$. We first prove $\mathtt{RR}_{\mathcal{A}}(t\{x/u\}) = \mathtt{RR}_{\mathcal{A}}(t)\{x/\mathtt{RR}_{\mathcal{A}}(u)\}$ when $|t|^{+}_{x} \leq 1$. Now, to prove in the general case that $\mathtt{RR}_{\mathcal{A}}(t\{x/u\}) = \mathtt{RR}_{\mathcal{A}}(t)\{x/\mathtt{RR}_{\mathcal{A}}(u)\}$ we proceed by induction on $|t|^{+}_{x}$.

- If $|t|^{+}_{x} = n + 1 \geq 2$, then $\mathtt{c} \notin \mathcal{B}$. We have

$$\begin{array}{ll}
\mathtt{RR}_{\mathcal{A}}(t\{x/u\}) & = \\
\mathtt{RR}_{\mathcal{A}}(t_{\overset{+}{x \rightsquigarrow x_1 \ldots x_n}}\{x_1/u\} \ldots \{x_n/u\}\{x/u\}) & =_{i.h.} \\
\mathtt{RR}_{\mathcal{A}}(t_{\overset{+}{x \rightsquigarrow x_1 \ldots x_n}})\{x_1/\mathtt{RR}_{\mathcal{A}}(u)\} \ldots \{x_n/\mathtt{RR}_{\mathcal{A}}(u)\}\{x/\mathtt{RR}_{\mathcal{A}}(u)\} & =_{L.\ 5.3.5:4} \\
\mathtt{RR}_{\mathcal{A}}(t)_{\overset{+}{x \rightsquigarrow x_1 \ldots x_n}}\{x_1/\mathtt{RR}_{\mathcal{A}}(u)\} \ldots \{x_n/\mathtt{RR}_{\mathcal{A}}(u)\}\{x/\mathtt{RR}_{\mathcal{A}}(u)\} & = \\
\mathtt{RR}_{\mathcal{A}}(t)\{x/\mathtt{RR}_{\mathcal{A}}(u)\} &
\end{array}$$

  We now show $\mathtt{RR}_{\mathcal{A}}(t\{x/u\}) = \mathtt{RR}_{\mathcal{A}}(t)\{x/\mathtt{RR}_{\mathcal{A}}(u)\}$ when $|t|^{+}_{x} \leq 1$. We proceed by induction on $\langle \mathtt{o}_x(t), \mathtt{size}(t) \rangle$.

- If $|t|^{+}_{x} = 0$ we have three cases.
  - If $|\mathtt{fv}(t)|_x = 0$ or $\mathtt{w} \notin \mathcal{B}$ then: $\mathtt{RR}_{\mathcal{A}}(t\{x/u\}) = \mathtt{RR}_{\mathcal{A}}(\mathtt{del}_x(t)) =_{L.\ 5.3.5:5} \mathtt{del}_x(\mathtt{RR}_{\mathcal{A}}(t)) = \mathtt{RR}_{\mathcal{A}}(t)\{x/\mathtt{RR}_{\mathcal{A}}(u)\}$.
  - If $|\mathtt{fv}(t)|_x > 0$ and $\mathtt{w} \in \mathcal{B}$ and $\mathtt{w} \notin \mathcal{A}$ then:

$$\begin{array}{ll}
\mathtt{RR}_{\mathcal{A}}(t\{x/u\}) & = \\
\mathtt{RR}_{\mathcal{A}}(\mathcal{W}_{\mathtt{fv}(u) \setminus \mathtt{fv}(t)}(\mathtt{del}_x(t))) & = \\
\mathcal{W}_{\mathtt{fv}(u) \setminus \mathtt{fv}(t)}(\mathtt{RR}_{\mathcal{A}}(\mathtt{del}_x(t))) & =_{L.\ 5.3.5:5} \\
\mathcal{W}_{\mathtt{fv}(u) \setminus \mathtt{fv}(t)}(\mathtt{del}_x(\mathtt{RR}_{\mathcal{A}}(t))) & =_{L.\ 5.3.5:3} \\
\mathcal{W}_{\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u)) \setminus \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t))}(\mathtt{del}_x(\mathtt{RR}_{\mathcal{A}}(t))) & = \mathtt{RR}_{\mathcal{A}}(t)\{x/\mathtt{RR}_{\mathcal{A}}(u)\}
\end{array}$$

  - If $|\mathtt{fv}(t)|_x > 0$ and $\mathtt{w} \in \mathcal{B}$ and $\mathtt{w} \in \mathcal{A}$ then:

$$\begin{array}{ll}
\mathtt{RR}_{\mathcal{A}}(t\{x/u\}) & = \\
\mathtt{RR}_{\mathcal{A}}(\mathcal{W}_{\mathtt{fv}(u) \setminus \mathtt{fv}(t)}(\mathtt{del}_x(t))) & = \\
\mathtt{RR}_{\mathcal{A}}(\mathtt{del}_x(t)) & =_{L.\ 5.3.5:5} \\
\mathtt{del}_x(\mathtt{RR}_{\mathcal{A}}(t)) & = \mathtt{RR}_{\mathcal{A}}(t)\{x/\mathtt{RR}_{\mathcal{A}}(u)\}
\end{array}$$

- We now consider the case where $|t|_x^+ = 1$
  - If $t = x$ then $\mathtt{RR}_{\mathcal{A}}(x)\{x/\mathtt{RR}_{\mathcal{A}}(u)\} = \mathtt{RR}_{\mathcal{A}}(u) = \mathtt{RR}_{\mathcal{A}}(x\{x/u\})$.
  - The case $t = \lambda y.v$ is straightforward by induction.
  - $t = v\ w$.

    If $x \in \mathtt{fv}^+(v)$ (so that $x \in \mathtt{fv}^+(\mathtt{RR}_{\mathcal{A}}(v))$), then

$$\begin{array}{ll}
\mathtt{RR}_{\mathcal{A}}(v\ w)\{x/\mathtt{RR}_{\mathcal{A}}(u)\} & = \\
(\mathtt{RR}_{\mathcal{A}}(v)\ \mathtt{RR}_{\mathcal{A}}(w))\{x/\mathtt{RR}_{\mathcal{A}}(u)\} & = \\
\mathtt{RR}_{\mathcal{A}}(v)\{x/\mathtt{RR}_{\mathcal{A}}(u)\}\ \mathtt{del}_x(\mathtt{RR}_{\mathcal{A}}(w)) & =_{i.h.} \\
\mathtt{RR}_{\mathcal{A}}(v\{x/u\})\ \mathtt{del}_x(\mathtt{RR}_{\mathcal{A}}(w)) & =_{L.\ 5.3.5} \\
\mathtt{RR}_{\mathcal{A}}(v\{x/u\})\ \mathtt{RR}_{\mathcal{A}}(\mathtt{del}_x(w)) & = \\
\mathtt{RR}_{\mathcal{A}}(v\{x/u\}\ \mathtt{del}_x(w)) & = \\
\mathtt{RR}_{\mathcal{A}}((v\ w)\{x/u\}) &
\end{array}$$

    The case $x \in \mathtt{fv}^+(w)$ is symmetric.
  - $t = v[y/w]$. This case is similar to the previous one.
  - $t = \mathcal{W}_y(v)$.

    If $y \neq x$ & $\mathtt{b} \neq \mathtt{w}$, then

$$\begin{array}{ll}
\mathtt{RR}_{\mathcal{A}}(\mathcal{W}_y(v))\{x/\mathtt{RR}_{\mathcal{A}}(u)\} & = \\
\mathcal{W}_y(\mathtt{RR}_{\mathcal{A}}(v))\{x/\mathtt{RR}_{\mathcal{A}}(u)\} & = \\
\mathcal{W}_y(\mathtt{del}_x(\mathtt{RR}_{\mathcal{A}}(v)))\{\!\{x/\mathtt{RR}_{\mathcal{A}}(u)\}\!\} & = \\
\mathcal{W}_{y\setminus\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u))}(\mathtt{del}_x(\mathtt{RR}_{\mathcal{A}}(v))\{\!\{x/\mathtt{RR}_{\mathcal{A}}(u)\}\!\}) & =_{L.\ 5.3.5} \\
\mathcal{W}_{y\setminus\mathtt{fv}(u)}(\mathtt{del}_x(\mathtt{RR}_{\mathcal{A}}(v))\{\!\{x/\mathtt{RR}_{\mathcal{A}}(u)\}\!\}) & =_{L.\ 5.3.5} \\
\mathcal{W}_{y\setminus\mathtt{fv}(u)}(\mathtt{RR}_{\mathcal{A}}(\mathtt{del}_x(v))\{\!\{x/\mathtt{RR}_{\mathcal{A}}(u)\}\!\}) & =_{i.h.} \\
\mathcal{W}_{y\setminus\mathtt{fv}(u)}(\mathtt{RR}_{\mathcal{A}}(\mathtt{del}_x(v)\{\!\{x/u\}\!\})) & = \\
\mathtt{RR}_{\mathcal{A}}(\mathcal{W}_{y\setminus\mathtt{fv}(u)}(\mathtt{del}_x(v)\{\!\{x/u\}\!\})) & = \\
\mathtt{RR}_{\mathcal{A}}(\mathcal{W}_y(v)\{x/u\}) &
\end{array}$$

    If $y \neq x$ & $\mathtt{b} = \mathtt{w}$, then

$$\begin{array}{ll}
\mathtt{RR}_{\mathcal{A}}(\mathcal{W}_y(v))\{x/\mathtt{RR}_{\mathcal{A}}(u)\} & = \\
\mathtt{RR}_{\mathcal{A}}(v)\{x/\mathtt{RR}_{\mathcal{A}}(u)\} & =_{i.h.} \\
\mathtt{RR}_{\mathcal{A}}(v\{x/u\}) & = \\
\mathtt{RR}_{\mathcal{A}}(\mathtt{del}_x(v)\{\!\{x/u\}\!\}) & = \\
\mathtt{RR}_{\mathcal{A}}(\mathcal{W}_{y\setminus\mathtt{fv}(u)}(\mathtt{del}_x(v)\{\!\{x/u\}\!\})) & \\
\mathtt{RR}_{\mathcal{A}}(\mathcal{W}_y(v)\{x/u\}) &
\end{array}$$

  - $t = \mathcal{C}_y^{y_1|y_2}(v)$.

    Most of the cases are done using the i.h. and Lemma 5.3.5 except the one where $y = x$ & $\mathtt{c} \in \mathcal{A}$. We use the following notations: $\Gamma = \mathtt{fv}(u)$,

Δ, Π are sets of fresh variables, $\Gamma_1 = \{x \in \Gamma | |\Gamma|_x^+ \geq 1\}$, $\Gamma_0 = \Gamma \setminus \Gamma_1$, $\Delta_1, \Pi_1, \Delta_0, \Pi_0$ are similarly defined.

$$
\begin{array}{ll}
\mathtt{RR_C}(\mathcal{C}_x^{y_1|y_2}(v))\{x/\mathtt{RR_C}(u)\} & = \\
\mathtt{S}_x^{y_1,y_2}(\mathtt{del}_{y_1,y_2}(\mathtt{RR_C}(v)))\{x/\mathtt{RR_C}(u)\} & =_{L.\ 5.3.4:4} \\
\mathtt{del}_{y_1,y_2}(\mathtt{RR_C}(v))\{\!\{y_1/\mathtt{RR_C}(u)\}\!\}\{\!\{y_2/\mathtt{RR_C}(u)\}\!\} & =_{L.\ 5.1.17:7} \\
\mathtt{RR_C}(v)\{y_1/\mathtt{RR_C}(u)\}\{y_2/\mathtt{RR_C}(u)\} & = \\
\mathtt{RR_C}(v)\{y_1/R_\Gamma^\Delta(R_\Delta^\Gamma(\mathtt{RR_C}(u)))\}\{y_2/R_\Gamma^\Pi(R_\Pi^\Gamma(\mathtt{RR_C}(u)))\} & =_{L.\ 5.3.5:1} \\
\mathtt{RR_C}(v)\{y_1/R_\Gamma^\Delta(\mathtt{RR_C}(R_\Delta^\Gamma(u)))\}\{y_2/R_\Gamma^\Pi(\mathtt{RR_C}(R_\Pi^\Gamma(u)))\} & =_{L.\ 5.3.2:2} \\
\mathtt{RR_C}(v)\{y_1/\mathtt{S}_\Gamma^{\Delta,\Pi}(\mathtt{RR_C}(R_\Delta^\Gamma(u)))\}\{y_2/\mathtt{S}_\Gamma^{\Delta,\Pi}(\mathtt{RR_C}(R_\Pi^\Gamma(u)))\} & =_{L.\ 5.3.4:5\,\&\,L.\ 5.3.2:1} \\
\mathtt{S}_\Gamma^{\Delta,\Pi}(\mathtt{RR_C}(v)\{y_1/\mathtt{RR_C}(R_\Delta^\Gamma(u))\}\{y_2/\mathtt{RR_C}(R_\Pi^\Gamma(u))\}) & =_{i.h.} \\
\mathtt{S}_{\Gamma_0}^{\Delta_0,\Pi_0}(\mathtt{S}_{\Gamma_1}^{\Delta_1,\Pi_1}(\mathtt{RR_C}(v\{y_1/R_\Delta^\Gamma(u)\}\{y_2/R_\Pi^\Gamma(u)\}))) & =_{L.\ 5.3.4:1} \\
\mathtt{S}_{\Gamma_0}^{\Delta_0,\Pi_0}(\mathtt{S}_{\Gamma_1}^{\Delta_1,\Pi_1}(\mathtt{del}_{\Delta_1,\Pi_1}(\mathtt{RR_C}(v\{y_1/R_\Delta^\Gamma(u)\}\{y_2/R_\Pi^\Gamma(u)\})))) & = \\
\mathtt{RR_C}(\mathcal{C}_\Gamma^{\Delta|\Pi}(v\{y_1/R_\Delta^\Gamma(u)\}\{y_2/R_\Pi^\Gamma(u)\})) & = \\
\mathtt{RR_C}(\mathcal{C}_x^{y_1|y_2}(v)\{x/u\}) &
\end{array}
$$

□

To illustrate Lemma 5.3.6, let us consider the terms $t = \mathcal{C}_x^{y|z}(\mathcal{W}_y(z))$ and $u = \mathcal{W}_a(\lambda w.w)$. Then $t\{x/u\} = \mathcal{C}_a^{a_1|a_2}(\mathcal{W}_{a_1}(\mathcal{W}_{a_2}(\lambda w.w)))$. We thus have:

$$\mathtt{RR_C}(t\{x/u\}) = \mathtt{S}_a^{a_1,a_2}(\mathcal{W}_{a_1}(\mathcal{W}_{a_2}(\lambda w.w))) = \mathcal{W}_a(\mathtt{S}_a^{a_1,a_2}(\lambda w.w)) = \mathcal{W}_a(\lambda w.w)$$

and

$$\mathtt{RR_C}(t)\{x/\mathtt{RR_C}(u)\} = x\{x/\mathcal{W}_a(\lambda w.w)\} = \mathcal{W}_a(\lambda w.w)$$

Calculi of the prismoid include rules/equations to handle substitution but also other rules/equations to handle resources $\{\mathtt{c}, \mathtt{w}\}$. Moreover, implicit (resp. explicit) substitution is managed by the β-rule (resp. the whole system $\mathtt{s}$). We can then split the reduction relation $\to_\mathcal{B}$ in two different parts: one for (implicit or explicit) substitution, which can be *s*trictly projected into itself, and another one for weakening and contraction, which can be projected into a more subtle way given by the following statement.

Theorem 5.3.7 (Projection)**.** *Let $\mathcal{A} \subseteq \mathcal{R}$ such that $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{S}$ and let $t \in \mathcal{T}_\mathcal{B}$. If $t \equiv_\mathcal{B} u$, then $\mathtt{RR}_\mathcal{A}(t) \equiv_{\mathcal{B}\setminus\mathcal{A}} \mathtt{RR}_\mathcal{A}(u)$. Otherwise:*

- *If $\mathtt{s} \notin \mathcal{B}$:*
  - *If $t \Rightarrow_\beta u$, then $\mathtt{RR}_\mathcal{A}(t) \to_\beta^+ \mathtt{RR}_\mathcal{A}(u)$.*
  - *If $t \Rightarrow_{\mathcal{B}\setminus\beta} u$, then $\mathtt{RR}_\mathcal{A}(t) \to_{\mathcal{B}\setminus\beta\setminus\mathcal{A}}^* \mathtt{RR}_\mathcal{A}(u)$ and $\mathtt{RR}_\mathcal{B}(t) = \mathtt{RR}_\mathcal{B}(u)$.*
- *Otherwise,*
  - *If $t \Rightarrow_\mathtt{s} u$, then $\mathtt{RR}_\mathcal{A}(t) \to_\mathtt{s}^+ \mathtt{RR}_\mathcal{A}(u)$.*
  - *If $t \Rightarrow_{\mathcal{B}\setminus\mathtt{s}} u$, then $\mathtt{RR}_\mathcal{A}(t) \to_{\mathcal{B}\setminus\mathtt{s}\setminus\mathcal{A}}^* \mathtt{RR}_\mathcal{A}(u)$.*

Proof. By induction on the reduction relation. For the points involving $\mathtt{RR}_{\mathcal{A}}(\_)$, one can first consider the case where $\mathcal{A}$ is a singleton. Then the general result follows from two successive applications of the simpler property.

We only show here the following interesting cases where $\mathtt{c} \in \mathcal{A}$.

- Let $t = \mathcal{C}_x^{y|z}(t_1)[x/u] \to_{\mathtt{SCa}} \mathcal{C}_\Gamma^{\Delta|\Pi}(t_1[y/R_\Delta^\Gamma(u)][z/R_\Pi^\Gamma(u)]) = t'$, with $y, z \in \mathtt{fv}^+(t_1)$, $\Gamma = \mathtt{fv}(u)$ and Π, Δ fresh. Then,

$$
\begin{array}{ll}
\mathtt{RR}_{\mathcal{A}}(t) & = \\
\mathtt{S}_x^{y,z}(\mathtt{del}_{y,z}(\mathtt{RR}_{\mathcal{A}}(t_1)))[x/\mathtt{RR}_{\mathcal{A}}(u)] & =_{L.\ 5.3.4:1} \\
\mathtt{S}_x^{y,z}(\mathtt{RR}_{\mathcal{A}}(t_1))[x/\mathtt{RR}_{\mathcal{A}}(u)] & =_{L.\ 5.3.4:2} \\
R_x^z(R_x^y(\mathtt{RR}_{\mathcal{A}}(t_1)))[x/\mathtt{RR}_{\mathcal{A}}(u)] & \to_{\mathtt{SDup}} \\
\mathtt{RR}_{\mathcal{A}}(t_1)[y/\mathtt{RR}_{\mathcal{A}}(u)][z/\mathtt{RR}_{\mathcal{A}}(u)] & = \\
\mathtt{RR}_{\mathcal{A}}(t_1)[y/R_\Gamma^\Delta(R_\Delta^\Gamma(\mathtt{RR}_{\mathcal{A}}(u)))][z/R_\Pi^\Gamma(R_\Pi^\Gamma(\mathtt{RR}_{\mathcal{A}}(u)))] & =_{L.\ 5.3.2:2} \\
\mathtt{RR}_{\mathcal{A}}(t_1)[y/\mathtt{S}_{\Gamma_0}^{\Delta_0}(\mathtt{S}_{\Gamma_1}^{\Delta_1}(\mathtt{RR}_{\mathcal{A}}(R_\Delta^\Gamma(u))))][z/\mathtt{S}_{\Gamma_0}^{\Pi_0}(\mathtt{S}_{\Gamma_1}^{\Pi_1}(\mathtt{RR}_{\mathcal{A}}(R_\Pi^\Gamma(u))))] & =_{L.\ 5.3.2:1} \\
\mathtt{S}_{\Gamma_0}^{\Delta_0,\Pi_0}(\mathtt{S}_{\Gamma_1}^{\Delta_1,\Pi_1}(\mathtt{RR}_{\mathcal{A}}(t_1)[y/\mathtt{RR}_{\mathcal{A}}(R_\Delta^\Gamma(u))][z/\mathtt{RR}_{\mathcal{A}}(R_\Pi^\Gamma(u))])) & =_{L.\ 5.3.4:1} \\
\mathtt{S}_{\Gamma_0}^{\Delta_0,\Pi_0}(\mathtt{S}_{\Gamma_1}^{\Delta_1,\Pi_1}(\mathtt{del}_{\Delta_1,\Pi_1}(\mathtt{RR}_{\mathcal{A}}(t_1)[y/\mathtt{RR}_{\mathcal{A}}(R_\Delta^\Gamma(u))][z/\mathtt{RR}_{\mathcal{A}}(R_\Pi^\Gamma(u))]))) & = \mathtt{RR}_{\mathcal{A}}(t')
\end{array}
$$

- $t = \mathcal{C}_x^{y|z}(t_1) \to_{\mathtt{CGc}} R_x^y(t_1) = t'$ with $z \notin \mathtt{fv}(t_1)$. We first notice that $z \notin \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t_1))$ by Lemma 5.3.5:3. Also, $\mathtt{w} \notin \mathcal{B}$, otherwise z would be in $\mathtt{fv}(t_1)$ so that $\mathtt{RR}_{\mathcal{A}}(t_1)$ does not contain weakening nor contractions. There are two cases to consider:

  - y is positive

$$
\begin{array}{ll}
\mathtt{RR}_{\mathcal{A}}(t) & = \\
\mathtt{S}_x^{y,z}(\mathtt{del}_{y,z}(\mathtt{RR}_{\mathcal{A}}(t_1))) & = \\
\mathtt{S}_x^{y,z}(\mathtt{RR}_{\mathcal{A}}(t_1)) & =_{L.\ 5.3.4:1} \\
\mathtt{S}_x^{y,z}(\mathtt{del}_y(\mathtt{RR}_{\mathcal{A}}(t_1))) & =_{L.\ 5.3.2:2} \\
R_x^y(\mathtt{RR}_{\mathcal{A}}(t_1)) & =_{L.\ 5.3.5:1} \\
\mathtt{RR}_{\mathcal{A}}(R_x^y(t_1)) & = \mathtt{RR}_{\mathcal{A}}(t')
\end{array}
$$

  - y is not positive

$$
\begin{array}{ll}
\mathtt{RR}_{\mathcal{A}}(t) & = \\
\mathtt{S}_x^{y,z}(\mathtt{RR}_{\mathcal{A}}(t_1)) & =_{L.\ 5.3.2:2} \\
R_x^y(\mathtt{RR}_{\mathcal{A}}(t_1)) & =_{L.\ 5.3.5:1} \\
\mathtt{RR}_{\mathcal{A}}(R_x^y(t_1)) & = \mathtt{RR}_{\mathcal{A}}(t')
\end{array}
$$

- $t = v[x/u] \to_{\mathtt{SDup}} v_{[y]_x}[x/u][y/u] = t'$, if $|v|_x^+ > 1$ & y fresh .
  By Lemma 5.3.5:2 $|\mathtt{fv}^+(\mathtt{RR}_{\mathcal{A}}(v))|_x^+ > 1$. Then,

$$
\begin{array}{ll}
\mathtt{RR}_{\mathcal{A}}(t) & = \\
\mathtt{RR}_{\mathcal{A}}(v)[x/\mathtt{RR}_{\mathcal{A}}(u)] & \to_{\mathtt{SDup}} \\
\mathtt{RR}_{\mathcal{A}}(v)_{[y]_x}[x/\mathtt{RR}_{\mathcal{A}}(u)][y/\mathtt{RR}_{\mathcal{A}}(u)] & =_{L.\ 5.3.5:4} \\
\mathtt{RR}_{\mathcal{A}}(v_{[y]_x})[x/\mathtt{RR}_{\mathcal{A}}(u)][y/\mathtt{RR}_{\mathcal{A}}(u)] & = \\
\mathtt{RR}_{\mathcal{A}}(t') &
\end{array}
$$

- $t = \mathcal{C}_w^{x|z}(\mathcal{C}_x^{y|p}(t_1)) \equiv_{\mathtt{CC}_\mathcal{A}} \mathcal{C}_w^{x|y}(\mathcal{C}_x^{z|p}(t_1)) = t'$ with $p \in \mathtt{fv}^+(t_1)$. Then,

$$\begin{array}{ll}
\mathtt{RR}_\mathcal{A}(t) & = \\
\mathtt{RR}_\mathcal{A}(\mathcal{C}_w^{x|z}(\mathcal{C}_x^{y|p}(t_1))) & = \\
\mathtt{S}_w^{x,z}(\mathtt{del}_{x,z}(\mathtt{RR}_\mathcal{A}(\mathcal{C}_x^{y|p}(t_1)))) & = \\
\mathtt{S}_w^{x,z}(\mathtt{del}_{x,z}(\mathtt{S}_x^{y,p}(\mathtt{del}_{y,p}(\mathtt{RR}_\mathcal{A}(t_1))))) & = \\
\mathtt{S}_w^{x,z}(\mathtt{del}_x(\mathtt{del}_z(\mathtt{S}_x^{y,p}(\mathtt{del}_{y,p}(\mathtt{RR}_\mathcal{A}(t_1)))))) & =_{L.\ 5.3.4:3} \\
\mathtt{S}_w^{x,z}(\mathtt{del}_x(\mathtt{S}_x^{y,p}(\mathtt{del}_z(\mathtt{del}_{y,p}(\mathtt{RR}_\mathtt{b}(t_1)))))) & = \\
\mathtt{S}_w^{x,z}(\mathtt{del}_x(\mathtt{S}_x^{y,p}(\mathtt{del}_{z,y,p}(\mathtt{RR}_\mathtt{b}(t_1))))) & =_{L.\ 5.3.2:3} \\
\mathtt{S}_w^{x,z}(\mathtt{S}_x^{y,p}(\mathtt{del}_{y,p}(\mathtt{del}_{z,y,p}(\mathtt{RR}_\mathtt{b}(t_1))))) & = \\
\mathtt{S}_w^{x,z}(\mathtt{S}_x^{y,p}(\mathtt{del}_{z,y,p}(\mathtt{RR}_\mathtt{b}(t_1)))) & =_{L.\ 5.3.2:4} \\
\mathtt{S}_w^{y,p,z}(\mathtt{del}_{z,y,p}(\mathtt{RR}_\mathtt{b}(t_1))) & = \\
\mathtt{S}_w^{x,y}(\mathtt{del}_{x,y}(\mathtt{S}_x^{z,p}(\mathtt{del}_{z,p}(\mathtt{RR}_\mathtt{b}(t_1))))) & = \mathtt{RR}_\mathtt{b}(t')
\end{array}$$

- $t = \mathcal{C}_{x'}^{y'|z'}(\mathcal{C}_x^{y|z}(t_1)) \equiv_{\mathtt{CC}_\mathcal{C}} \mathcal{C}_x^{y|z}(\mathcal{C}_{x'}^{y'|z'}(t_1)) = t'$. Then,

  - Both contracted variables are positive. Then,

$$\begin{array}{ll}
\mathtt{RR}_\mathcal{A}(t) & = \\
\mathtt{RR}_\mathcal{A}(\mathcal{C}_{x'}^{y'|z'}(\mathcal{C}_x^{y|z}(t_1))) & = \\
\mathtt{S}_{x'}^{y',z'}(\mathtt{del}_{y',z'}(\mathtt{S}_x^{y,z}(\mathtt{del}_{y,z}(\mathtt{RR}_\mathcal{A}(t_1))))) & =_{L.\ 5.3.4:3} \\
\mathtt{S}_{x'}^{y',z'}(\mathtt{S}_x^{y,z}(\mathtt{del}_{y',z',y,z}(\mathtt{RR}_\mathcal{A}(t_1)))) & =_{L.\ 5.3.2:5} \\
\mathtt{S}_x^{y,z}(\mathtt{S}_{x'}^{y',z'}(\mathtt{del}_{y',z',y,z}(\mathtt{RR}_\mathcal{A}(t_1)))) & =_{L.\ 5.3.4:3} \\
\mathtt{S}_x^{y,z}(\mathtt{del}_{y,z}(\mathtt{S}_{x'}^{y',z'}(\mathtt{del}_{y',z'}(\mathtt{RR}_\mathcal{A}(t_1))))) & = \mathtt{RR}_\mathcal{A}(t')
\end{array}$$

  - Both contracted variables are negative. Then,

$$\begin{array}{ll}
\mathtt{RR}_\mathcal{A}(t) & = \\
\mathtt{RR}_\mathcal{A}(\mathcal{C}_{x'}^{y'|z'}(\mathcal{C}_x^{y|z}(t_1))) & = \\
\mathtt{S}_{x'}^{y',z'}(\mathtt{S}_x^{y,z}(\mathtt{RR}_\mathcal{A}(t_1))) & =_{L.\ 5.3.2:5} \\
\mathtt{S}_x^{y,z}(\mathtt{S}_{x'}^{y',z'}(\mathtt{RR}_\mathcal{A}(t_1))) & = \mathtt{RR}_\mathcal{A}(t')
\end{array}$$

  - One contracted variable is positive, the other one is negative. Then,

$$\begin{array}{ll}
\mathtt{RR}_\mathcal{A}(t) & = \\
\mathtt{RR}_\mathcal{A}(\mathcal{C}_{x'}^{y'|z'}(\mathcal{C}_x^{y|z}(t_1))) & = \\
\mathtt{S}_{x'}^{y',z'}(\mathtt{S}_x^{y,z}(\mathtt{del}_{y,z}(\mathtt{RR}_\mathcal{A}(t_1)))) & =_{L.\ 5.3.2:5} \\
\mathtt{S}_x^{y,z}(\mathtt{S}_{x'}^{y',z'}(\mathtt{del}_{y,z}(\mathtt{RR}_\mathcal{A}(t_1)))) & =_{L.\ 5.3.4:3} \\
\mathtt{S}_x^{y,z}(\mathtt{del}_{y,z}(\mathtt{S}_{x'}^{y',z'}(\mathtt{RR}_\mathcal{A}(t_1)))) & = \mathtt{RR}_\mathcal{A}(t')
\end{array}$$

- The other root cases use Lemmas 5.3.2, 5.3.4, 5.3.5, and 5.3.6.

- Cases of internal reductions use Lemma 5.3.4:6. In particular, the case $t = \mathcal{C}_x^{y|z}(t')$ also uses Lemma 5.3.2.

□

For instance, the reduction $t = \mathcal{C}_x^{y|z}(y\ z)[x/a] \to_{\mathtt{SCa}} \mathcal{C}_a^{a_1|a_2}((y\ z)[y/a_1][z/a_2]) = t'$ is projected into $\mathtt{RR_C}(t) = (x\ x)[x/a] \to_{\mathtt{SDup}} (x\ y)[x/a][y/a] =_\alpha (y\ z)[y/a][z/a] = \mathtt{RR_C}(t')$.

It is now time to discuss the need of positive conditions (conditions involving positive free variables) in the specification of the reduction rules of the prismoid. For that, let us consider a relaxed form of the $\mathtt{SS}_1$-rule: $t[x/u][y/v] \to t[x/u[y/v]]$ if $y \in \mathtt{fv}(u) \setminus \mathtt{fv}(t)$ (instead of $y \in \mathtt{fv}^+(u) \setminus \mathtt{fv}(t)$).

The need for the condition $y \in \mathtt{fv}(u)$ is well-known [Blo97], otherwise PSN does not hold. The need for the condition $y \notin \mathtt{fv}(t)$ is also natural if one wants to preserve well-formed terms. Now, the reduction step $t_1 = x[x/\mathcal{W}_y(z)][y/y'] \to_{\mathtt{SS}_1} x[x/\mathcal{W}_y(z)[y/y']] = t_2$ in the calculus with sorts $\{\mathtt{s}, \mathtt{w}\}$ cannot be projected into $\mathtt{RR_w}(t_1) = x[x/z][y/y'] \to_{\mathtt{SS}_1} x[x/z[y/y']] = \mathtt{RR_w}(t_2)$ since $y \notin \mathtt{fv}(z)$. Similar examples can be given to justify positive conditions in rules $\mathtt{SDup}, \mathtt{SCa}$ and $\mathtt{CS}$.

Lemma 5.3.8. *Let $t \in \mathcal{T}_\emptyset$ and let $\mathcal{A} \subseteq \mathcal{R}$. Then $\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(t)) = t$.*

Proof. By induction on $\mathtt{size}(t)$.

- $t = x$

$$\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(x)) = x$$

- $t = \lambda x.t'$
  - $\mathtt{w} \in \mathcal{A}$ and $\mathtt{x} \notin \mathtt{fv}(t')$

$$\mathtt{RR}_\mathcal{A}(\lambda x.\mathcal{W}_x(\mathtt{AR}_\mathcal{A}(t'))) = \lambda x.\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(t')) =_{i.h.} \lambda x.t'$$

  - Otherwise, we are easily done by i.h.

- $t = v\ w$
  - $\mathtt{c} \in \mathcal{A}$

$$\begin{array}{ll}
\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(t)) & = \\
\mathtt{RR}_\mathcal{A}(\mathcal{C}_\Gamma^{\Delta|\Pi}(R_\Delta^\Gamma(\mathtt{AR}_\mathcal{A}(v))\ R_\Pi^\Gamma(\mathtt{AR}_\mathcal{A}(w)))) & = \\
\mathtt{S}_{\Gamma,\Gamma}^{\Delta,\Pi}(\mathtt{RR}_\mathcal{A}(R_\Delta^\Gamma(\mathtt{AR}_\mathcal{A}(v))\ R_\Pi^\Gamma(\mathtt{AR}_\mathcal{A}(w)))) & = \\
\mathtt{S}_{\Gamma,\Gamma}^{\Delta,\Pi}(\mathtt{RR}_\mathcal{A}(R_\Delta^\Gamma(\mathtt{AR}_\mathcal{A}(v)))\ \mathtt{RR}_\mathcal{A}(R_\Pi^\Gamma(\mathtt{AR}_\mathcal{A}(w)))) & =_{L.\ 5.3.2} \\
\mathtt{S}_{\Gamma,\Gamma}^{\Delta,\Pi}(R_\Delta^\Gamma(\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(v)))\ R_\Pi^\Gamma(\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(w)))) & = \\
\mathtt{S}_{\Gamma,\Gamma}^{\Delta,\Pi}(R_\Delta^\Gamma(\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(v))))\ \mathtt{S}_{\Gamma,\Gamma}^{\Delta,\Pi}(R_\Pi^\Gamma(\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(w)))) & =_{L.\ 5.3.2} \\
\mathtt{S}_\Gamma^\Delta(R_\Delta^\Gamma(\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(v))))\ \mathtt{S}_\Gamma^\Pi(R_\Pi^\Gamma(\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(w)))) & =_{L.\ 5.3.2} \\
R_\Gamma^\Delta(R_\Delta^\Gamma(\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(v))))\ R_\Gamma^\Pi(R_\Pi^\Gamma(\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(w)))) & = \\
\mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(v))\ \mathtt{RR}_\mathcal{A}(\mathtt{AR}_\mathcal{A}(w)) & =_{i.h.} t
\end{array}$$

  - Otherwise, we are easily done by i.h.

□

The following property states that administration of weakening and/or contraction is terminating in any calculus.

Lemma 5.3.9. *If* $\mathtt{s} \notin \mathcal{B}$*, then the reduction relation* $\to_{\mathcal{B}\setminus\beta}$ *is terminating. If* $\mathtt{s} \in \mathcal{B}$*, then the reduction relation* $\to_{\mathcal{B}\setminus\mathtt{s}}$ *is terminating.*

Proof. The reduction relation $\to_{\mathcal{B}\setminus\beta}$ is contained in $\to_{\mathcal{B}\setminus\mathtt{s}}$ so it is sufficient to show termination of the biggest relation. We show that $w \to_{\mathcal{B}\setminus\mathtt{s}} w'$ implies $\langle \mathtt{S}(w'), \mathtt{I}(w'), \mathtt{L}(w')\rangle <_{\mathtt{lex}} \langle \mathtt{S}(w), \mathtt{I}(w), \mathtt{L}(w)\rangle$ where $\mathtt{S}(t)$, $\mathtt{I}(t)$ and $\mathtt{L}(t)$ are defined by induction as follows :

$$
\begin{array}{lcl@{\qquad}lcl}
\mathtt{S}(x) & := & 1 & \mathtt{L}(x) & := & 1 \\
\mathtt{S}(\lambda x.t) & := & \mathtt{S}(t) & \mathtt{L}(\lambda x.t) & := & \mathtt{L}(t) \\
\mathtt{S}(v\ w) & := & \mathtt{S}(v) + \mathtt{S}(w) & \mathtt{L}(t\ u) & := & \mathtt{L}(t) + \mathtt{L}(u) \\
\mathtt{S}(\mathcal{W}_x(t)) & := & \mathtt{S}(t) & \mathtt{L}(\mathcal{W}_x(t)) & := & \mathtt{L}(t) \\
\mathtt{S}(\mathcal{C}_x^{y|z}(t)) & := & \mathtt{S}(t) & \mathtt{L}(\mathcal{C}_x^{y|z}(t)) & := & \mathtt{L}(t) + 1 \\
\mathtt{S}(t[x/u]) & := & \mathtt{S}(t) + \mathtt{M}_x(t).\mathtt{S}(u) & \mathtt{L}(t[x/u]) & := & \mathtt{L}(t).(\mathtt{L}(u) + 1)
\end{array}
$$

$$
\begin{array}{lcl}
\mathtt{I}(x) & := & 2 \\
\mathtt{I}(\lambda x.t) & := & 2.\mathtt{I}(t) + 2 \\
\mathtt{I}(t\ u) & := & 2.(\mathtt{I}(t) + \mathtt{I}(u)) + 2 \\
\mathtt{I}(\mathcal{W}_x(t)) & := & \mathtt{I}(t) + 1 \\
\mathtt{I}(\mathcal{C}_x^{y|z}(t)) & := & 2.\mathtt{I}(t) \\
\mathtt{I}(t[x/u]) & := & \mathtt{I}(t).(\mathtt{I}(u) + 1)
\end{array}
$$

with $\mathtt{M}_x(t)$ defined as follows :
If $x \notin \mathtt{fv}(t)$ then $\mathtt{M}_x(t) := 1$, otherwise :

$$
\begin{array}{lcl}
\mathtt{M}_x(x) & := & 1 \\
\mathtt{M}_x(\lambda y.t) & := & \mathtt{M}_x(t) \\
\mathtt{M}_x(t\ u) & := & \begin{cases} \mathtt{M}_x(t) & \text{if } x \in \mathtt{fv}(t) \setminus \mathtt{fv}(u) \\ \mathtt{M}_x(u) & \text{if } x \in \mathtt{fv}(u) \setminus \mathtt{fv}(t) \\ \mathtt{M}_x(t) + \mathtt{M}_x(u) & \text{if } x \in \mathtt{fv}(t) \cap \mathtt{fv}(u) \end{cases} \\
\mathtt{M}_x(\mathcal{W}_y(t)) & := & \begin{cases} 1 & \text{if } x = y \\ \mathtt{M}_x(t) & \text{if } x \neq y \end{cases} \\
\mathtt{M}_x(\mathcal{C}_y^{y_1|y_2}(t)) & := & \begin{cases} 1 + \mathtt{M}_{y_1}(t) + \mathtt{M}_{y_2}(t) & \text{if } x = y \\ \mathtt{M}_x(t) & \text{if } x \neq y \end{cases} \\
\mathtt{M}_x(t[y/u]) & := & \begin{cases} \mathtt{M}_x(t) + \mathtt{M}_y(t).(\mathtt{M}_x(u) + 1) & \text{if } x \in \mathtt{fv}(u) \cap \mathtt{fv}(t) \\ \mathtt{M}_y(t).(\mathtt{M}_x(u) + 1) & \text{if } x \in \mathtt{fv}(u) \setminus \mathtt{fv}(t) \\ \mathtt{M}_x(t) & \text{otherwise} \end{cases}
\end{array}
$$

We also have to show that $w \to_{\mathcal{B}\setminus\mathtt{s}} w'$ implies $\mathtt{M}_z(w) = \mathtt{M}_z(w')$. If $z \notin \mathtt{fv}(w)$, we have $\mathtt{M}_z(w) = \mathtt{M}_z(w')$. So suppose $z \in \mathtt{fv}(w)$.

We show properties for the equation $\mathtt{SS}_{\mathcal{C}}$ (other are straightforward), and some interesting cases for the rules:

- $t[x/u][y/v] \equiv_{\mathtt{SS}_C} t[y/v][x/u]$ with $x \notin \mathtt{fv}(v)$ and $y \notin \mathtt{fv}(u)$.
  - Multiplicity (suppose $y, x \in \mathtt{fv}(t)$ (other cases are similar))
    - $z \in \mathtt{fv}(t) \cap \mathtt{fv}(u) \cap \mathtt{fv}(v)$

$$\begin{array}{ll} \mathtt{M}_z(w) & = \\ \mathtt{M}_z(t[x/u]) + \mathtt{M}_y(t).(\mathtt{M}_z(v)+1) & = \\ \mathtt{M}_z(t) + \mathtt{M}_x(t).(\mathtt{M}_z(u)+1) + \mathtt{M}_y(t).(\mathtt{M}_z(v)+1) & = \\ \mathtt{M}_z(t[y/v]) + \mathtt{M}_x(t).(\mathtt{M}_z(u)+1) & = \mathtt{M}_z(w') \end{array}$$

    - $z \in \mathtt{fv}(t) \cap \mathtt{fv}(u) \setminus \mathtt{fv}(v)$

$$\begin{array}{ll} \mathtt{M}_z(w) & = \\ \mathtt{M}_z(t[x/u]) & = \\ \mathtt{M}_z(t) + \mathtt{M}_x(t).(\mathtt{M}_z(u)+1) & = \\ \mathtt{M}_z(t[y/v]) + \mathtt{M}_x(t).(\mathtt{M}_z(u)+1) & = \mathtt{M}_z(w') \end{array}$$

    - $z \in \mathtt{fv}(u) \cap \mathtt{fv}(v) \setminus \mathtt{fv}(t)$

$$\begin{array}{ll} \mathtt{M}_z(w) & = \\ \mathtt{M}_z(t[x/u]) + \mathtt{M}_y(t).(\mathtt{M}_z(v)+1) & = \\ \mathtt{M}_x(t).(\mathtt{M}_z(u)+1) + \mathtt{M}_y(t).(\mathtt{M}_z(v)+1) & = \\ \mathtt{M}_z(t[y/v]) + \mathtt{M}_x(t).(\mathtt{M}_z(u)+1) & = \mathtt{M}_z(w') \end{array}$$

    - $z \in \mathtt{fv}(v) \setminus \mathtt{fv}(t) \setminus \mathtt{fv}(u)$

$$\begin{array}{ll} \mathtt{M}_z(w) & = \\ \mathtt{M}_y(t).(\mathtt{M}_z(v)+1) & = \\ \mathtt{M}_z(t[y/v]) & = \mathtt{M}_z(w') \end{array}$$

  - Complexity

$$\begin{array}{ll} \mathtt{S}(w) & = \\ \mathtt{S}(t) + \mathtt{M}_x(t).\mathtt{S}(u) + \mathtt{M}_y(t[x/u]).\mathtt{S}(v) & = \\ \mathtt{S}(t) + \mathtt{M}_x(t).\mathtt{S}(u) + \mathtt{M}_y(t).\mathtt{S}(v) & = \\ \mathtt{S}(t) + \mathtt{M}_y(t).\mathtt{S}(v) + \mathtt{M}_x(t[y/v]).\mathtt{S}(u) & = \mathtt{S}(w') \end{array}$$

  - Interpretation

$$\mathtt{I}(w) = \mathtt{I}(t).(\mathtt{I}(u)+1).(\mathtt{I}(v)+1) = \mathtt{I}(w')$$

  - Last

$$\mathtt{L}(w) = \mathtt{L}(t).(\mathtt{L}(u)+1).(\mathtt{L}(v)+1) = \mathtt{L}(w')$$

- $\mathcal{W}_y(u)v \to_{\mathtt{AW}_1} \mathcal{W}_{y\setminus\mathtt{fv}(v)}(uv)$
  - Multiplicity

* $z \neq y$, $z \in \mathtt{fv}(u) \cap \mathtt{fv}(v)$ (other cases are similar),

$$\mathtt{M}_z(w) = \mathtt{M}_z(u) + \mathtt{M}_z(v) = \mathtt{M}_z(w')$$

* $z = y$
  · $y \in \mathtt{fv}(v)$

$$\mathtt{M}_z(w) = 1 + \mathtt{M}_z(v) > \mathtt{M}_z(v) = \mathtt{M}_z(w')$$

  · $y \notin \mathtt{fv}(v)$

$$\mathtt{M}_z(w) = 1 = \mathtt{M}_z(w')$$

– The complexity is obviously the same.

– Interpretation

$$\begin{array}{ll} \mathtt{I}(w) & = \\ 2.(\mathtt{I}(\mathcal{W}_y(u)) + \mathtt{I}(v)) + 2 & = \\ 2.(\mathtt{I}(u) + 1 + \mathtt{I}(v)) + 2 & > \\ 2.(\mathtt{I}(u) + \mathtt{I}(v)) + 2 + 1 & = \mathtt{I}(w') \end{array}$$

• $\mathcal{C}_x^{x_1|x_2}(t)[x/u] \rightarrow_{\mathtt{SCa}} \mathcal{C}_\Gamma^{\Delta|\Pi}(t[x_1/R_\Delta^\Gamma(u)][x_2/R_\Pi^\Gamma(u)])$ with $x_1, x_2 \in \mathtt{fv}^+(t)$, $\Gamma = \mathtt{fv}(u)$, Δ, Π fresh

– Multiplicity

* $z \notin \mathtt{fv}(u)$

$$\mathtt{M}_z(w) = \mathtt{M}_z(t) = \mathtt{M}_z(w')$$

* $z \in \mathtt{fv}(u)$ (and thus $z \notin \mathtt{fv}(t)$)

$$\begin{array}{ll} \mathtt{M}_z(w) & = \\ (\mathtt{M}_{x_1}(t) + \mathtt{M}_{x_2}(t) + 1).(\mathtt{M}_z(u) + 1) & > \\ \mathtt{M}_{x_1}(t).(\mathtt{M}_z(u) + 1) + \mathtt{M}_{x_2}(t).(\mathtt{M}_z(u) + 1) + 1 & = \\ \mathtt{M}_{z_1}(t[x_1/R_\Delta^\Gamma(u)][x_2/R_\Pi^\Gamma(u)]) + \mathtt{M}_{z_2}(t[x_1/R_\Delta^\Gamma(u)][x_2/R_\Pi^\Gamma(u)]) + 1 & = \\ \mathtt{M}_z(w') & \end{array}$$

– Complexity

$$\begin{array}{ll} \mathtt{S}(w) & = \\ \mathtt{S}(t) + (\mathtt{M}_y(t) + \mathtt{M}_z(t) + 1).\mathtt{S}(u) & > \\ \mathtt{S}(t) + \mathtt{M}_y(t).\mathtt{S}(u) + \mathtt{M}_z(t).\mathtt{S}(u) & = (z \notin \mathtt{fv}(R_\Delta^\Gamma(u))) \\ \mathtt{S}(t[y/R_\Delta^\Gamma(u)]) + \mathtt{M}_z(t[y/R_\Delta^\Gamma(u)]).\mathtt{S}(u) & = \\ \mathtt{S}(t[y/R_\Delta^\Gamma(u)][z/R_\Pi^\Gamma(u)]) & = \mathtt{S}(w') \end{array}$$

– Interpretation

• $\mathcal{C}_a^{a_1|a_2}(t[x/u]) \rightarrow_{\mathtt{CS}} t[x/\mathcal{C}_a^{a_1|a_2}(u)]$ with $a_1, a_2 \in \mathtt{fv}^+(u)$

– Multiplicity

* $z \neq a$

· $z \notin \mathtt{fv}(u)$ (and thus $z \in \mathtt{fv}(t)$)

$$\mathtt{M}_z(w) = \mathtt{M}_z(t) = \mathtt{M}_z(w')$$

· $z \in \mathtt{fv}(u)$ (and thus $z \notin \mathtt{fv}(t)$)

$$\mathtt{M}_z(w) = \mathtt{M}_x(t).(\mathtt{M}_z(u) + 1) = \mathtt{M}_z(w')$$

∗ $z = a$

$$\begin{array}{ll}
\mathtt{M}_z(w) & = \\
\mathtt{M}_{a_1}(t[x/u]) + \mathtt{M}_{a_2}(t[x/u]) + 1 & = \\
\mathtt{M}_x(t).(\mathtt{M}_{a_1}(u) + 1) + \mathtt{M}_x(t).(\mathtt{M}_{a_2}(u) + 1) + 1 & = \\
\mathtt{M}_x(t).(\mathtt{M}_{a_1}(u) + \mathtt{M}_{a_2}(u) + 1) + \mathtt{M}_x(t) + 1 & > \\
\mathtt{M}_x(t).(\mathtt{M}_{a_1}(u) + \mathtt{M}_{a_2}(u) + 1) + \mathtt{M}_x(t) & = \\
\mathtt{M}_x(t).((\mathtt{M}_{a_1}(u) + \mathtt{M}_{a_2}(u) + 1) + 1) & = \\
\mathtt{M}_x(t).(\mathtt{M}_a(\mathcal{C}_a^{a_1|a_2}(u)) + 1) & = \\
\mathtt{M}_z(w') &
\end{array}$$

– Complexity

$$\mathtt{S}(w) = \mathtt{S}(t) + \mathtt{M}_x(t).\mathtt{S}(u) = \mathtt{S}(w')$$

– Interpretation

$$\mathtt{I}(w) = 2.\mathtt{I}(t).(\mathtt{I}(u) + 1) > \mathtt{I}(t).(2.\mathtt{I}(u) + 1) = \mathtt{I}(w')$$

- The inductive cases are easy. Let consider the interesting one : $t[y/u] \to t[y/u']$ with $z \in \mathtt{fv}(u) \cap \mathtt{fv}(t)$.

  – Multiplicity. By induction hypothesis we get $\mathtt{M}_z(u') \leq \mathtt{M}_z(u)$

  ∗ $z \in \mathtt{fv}(u')$

$$\begin{array}{ll}
\mathtt{M}_z(t[y/u]) & = \\
\mathtt{M}_z(t) + \mathtt{M}_y(t).(\mathtt{M}_z(u) + 1) & \geq \\
\mathtt{M}_z(t) + \mathtt{M}_y(t).(\mathtt{M}_z(u') + 1) & = \mathtt{M}_z(t[y/u'])
\end{array}$$

  ∗ $z \notin \mathtt{fv}(u')$

$$\mathtt{M}_z(t[y/u]) = \mathtt{M}_z(t) + \mathtt{M}_y(t).(\mathtt{M}_z(u) + 1) > \mathtt{M}_z(t) = \mathtt{M}_z(t[y/u'])$$

  – Complexity is straightforward (as the other cases) since $\mathtt{M}_x(t) = 1$ when $x \notin \mathtt{fv}(t)$.

  – Interpretation is also straightforward.

□

We conclude this section by relating adding and removing resources :

Lemma 5.3.10. *Let $\emptyset \neq \mathcal{A} \subseteq \mathcal{R}$. If $t \in \mathcal{T}_{\mathcal{A}}$ is in $\mathcal{A}$-normal form then $\mathtt{w} \in \mathcal{A}$ implies $t \equiv_{\mathcal{A}} \mathcal{W}_{\mathtt{fv}(t)\setminus\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t))}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t)))$ and $\mathtt{w} \notin \mathcal{A}$ implies $t \equiv_{\mathcal{A}} \mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t))$.*

Proof. By induction on $\mathtt{size}(t)$.

- If $t = x$, then $x = \mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(x))$ and $\mathtt{fv}(t) \setminus \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t)) = \emptyset$
- If $t = \lambda x.u$, then we reason by cases.
  - $\mathtt{w} \in \mathcal{A}$. We know $u \equiv_{\mathcal{A}} \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u))}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(u)))$ by the i.h. But t is in $\mathcal{A}$-normal form, so $\mathtt{fv}(u) \setminus \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u)) \subseteq \{x\}$, otherwise it can be reduced by $\mathtt{LW}$. Now, if $\mathtt{fv}(u) \setminus \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u)) = \emptyset$, then also $\mathtt{fv}(t) \setminus \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t)) = \emptyset$ and the claim $t \equiv_{\mathcal{A}} \mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(\lambda x.u))$ immediately holds. Otherwise, $\mathtt{fv}(u) \setminus \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u)) = \{x\}$ and $t \equiv_{\mathcal{A}} \lambda x.\mathcal{W}_x(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(u))) = \mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t))$.
  - $\mathtt{w} \notin \mathcal{A}$. Then $\lambda x.u \equiv_{\mathcal{A}}$ *(i.h.)* $\lambda x.\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(u)) = \mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(\lambda x.u))$.
- If $t = u\ v$, then we reason by cases.
  - $\mathtt{w} \in \mathcal{A}$. Then,

    $$t \equiv_{\mathcal{A}} \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u))}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(u)))\ \mathcal{W}_{\mathtt{fv}(v)\setminus\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(v))}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(v)))$$

    by the i.h. But t is an $\mathcal{A}$-normal form, thus $\mathtt{fv}(u) \setminus \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u)) = \mathtt{fv}(v) \setminus \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(v)) = \emptyset$, (otherwise it could be reduced by $\mathtt{AW_l}$ or $\mathtt{AW_r}$). Hence, $\mathtt{fv}(t) = \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t))$ and $t \equiv_{\mathcal{A}} \mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(u))\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(v))$. If $\mathtt{c} \in \mathcal{A}$ then $t \equiv_{\mathcal{A}} \mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t))$ since $\mathtt{RR}_{\mathcal{A}}(u)$ and $\mathtt{RR}_{\mathcal{A}}(v)$ have no variable in common. If $\mathtt{c} \notin \mathcal{A}$ then $t \equiv_{\mathcal{A}} \mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t))$ by definition of the function $\mathtt{AR}_{\mathcal{A}}(\_)$.
  - $\mathtt{w} \notin \mathcal{A}$. Then, $t \equiv_{\mathcal{A}} \mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(u))\ \mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(v))$ by i.h. We have $t \equiv_{\mathcal{A}} \mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t))$ since $\mathtt{RR}_{\mathcal{A}}(u)$ and $\mathtt{RR}_{\mathcal{A}}(v)$ have no variable in common.
- If $t = \mathcal{W}_x(u)$, then $t \equiv_{\mathcal{A}} \mathcal{W}_x(\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u))}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(u))))$ by the i.h. This last term is equal to $\mathcal{W}_{\mathtt{fv}(t)\setminus\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t))}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t)))$ since $x \in \mathtt{fv}(t)$ but $x \notin \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t))$.
- If $t = \mathcal{C}_x^{y|z}(u)$, then $t \equiv_{\mathcal{A}} \mathcal{C}_x^{y|z}(\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u))}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(u))))$ by the i.h. We know also that $y, z \in \mathtt{fv}^+(u)$ since otherwise t could be reduced by $\mathtt{CW_2}$ or $\mathtt{CW_1}$. We now reason by cases.
  - $\mathtt{w} \in \mathcal{A}$. Since t is in $\mathcal{A}$-normal form, we have $\mathtt{fv}(u) \setminus \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u)) = \emptyset$, otherwise t could be reduced by $\mathtt{CW_2}$ or $\mathtt{CW_1}$. Thus we get $t \equiv_{\mathcal{A}} \mathcal{C}_x^{y|z}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(u)))$. But t is well-formed, so that $y, z \in \mathtt{fv}(u)$ and $x \notin \mathtt{fv}(u)$. Since $y, z \in \mathtt{fv}^+(u)$, then $y, z \in \mathtt{fv}^+(\mathtt{RR}_{\mathcal{A}}(u)) \subseteq \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u))$ and also $x \notin \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(u))$.

    Since $\mathtt{c} \in \mathcal{A}$, then by definition $\mathtt{RR}_{\mathcal{A}}(t) = \mathtt{S}_x^{y,z}(\mathtt{del}_{y,z}(\mathtt{RR}_{\mathcal{A}}(u)))$, so that $x \in \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t))$ and we get $\mathtt{fv}(t) = \mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t))$.

Notice that $\mathtt{RR}_{\mathcal{A}}(u)$ can be neither a variable (otherwise t would not be well-formed) nor an abstraction (otherwise t could be reduced by $\mathtt{CL}$), so that $\mathtt{RR}_{\mathcal{A}}(u) = w\ v$, and thus $\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(u)) = \mathcal{C}_{\Phi}^{\Upsilon|\Psi}(R_{\Upsilon}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(w))\ R_{\Psi}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(v)))$ for $\Phi = \mathtt{fv}(w) \cap \mathtt{fv}(v)$ and Υ and Ψ fresh sets of variables.

Hence, $t \equiv_{\mathcal{A}} \mathcal{C}_x^{y|z}(\mathcal{C}_{\Phi}^{\Upsilon|\Psi}(R_{\Upsilon}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(w))\ R_{\Psi}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(v))))$.

Now it would suffice that $y \in \mathtt{fv}(w) \setminus \mathtt{fv}(v)$ and $z \in \mathtt{fv}(v) \setminus \mathtt{fv}(w)$ (the symmetric case is similar) to prove that this term is in fact:

$$
\begin{array}{ll}
\mathcal{C}_x^{y|z}(\mathcal{C}_{\Phi}^{\Upsilon|\Psi}(R_{\Upsilon}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(w))\ R_{\Psi}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(v)))) & = \\
\mathcal{C}_x^{y|z}(\mathcal{C}_{\Phi}^{\Upsilon|\Psi}(\mathtt{AR}_{\mathcal{A}}(R_{\Upsilon,y}^{\Phi,x}(R_x^y(w)))\ \mathtt{AR}_{\mathcal{A}}(R_{\Psi,z}^{\Phi,x}(R_x^z(v))))) & = \\
\mathcal{C}_x^{y|z}(\mathcal{C}_{\Phi}^{\Upsilon|\Psi}(R_{\Upsilon,y}^{\Phi,x}(\mathtt{AR}_{\mathcal{A}}(R_x^y(w)))\ R_{\Psi,z}^{\Phi,x}(\mathtt{AR}_{\mathcal{A}}(R_x^z(v))))) & = \\
\mathtt{AR}_{\mathcal{A}}(R_x^y(w)\ R_x^z(v)) & =_{L.\ 5.3.2:2} \\
\mathtt{AR}_{\mathcal{A}}(\mathtt{S}_x^{y,z}(\mathtt{RR}_{\mathcal{A}}(u))) & =_{L.\ 5.3.4:1} \\
\mathtt{AR}_{\mathcal{A}}(\mathtt{S}_x^{y,z}(\mathtt{del}_{y,z}(\mathtt{RR}_{\mathcal{A}}(u)))) & = \\
\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t)) &
\end{array}
$$

By well-formedness we know that $y, z \in \mathtt{fv}(w\ v)$.

Suppose that one of them, say y, is both in w and in v. Then $y \in \Phi$, so that

$$t \equiv_{\mathcal{A}} \mathcal{C}_x^{y|z}(\mathcal{C}_{\Phi',y}^{(\Upsilon',y')|(\Psi',y'')}(R_{\Upsilon}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(w)))\ R_{\Psi}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(v)))$$

which we can rearrange using $\equiv_{\mathtt{CC}_{\mathcal{A}}}$ into

$$t \equiv_{\mathcal{A}} \mathcal{C}_x^{y|y''}(\mathcal{C}_{\Phi',y}^{(\Upsilon',y')|(\Psi',z)}(R_{\Upsilon}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(w))\ R_{\Psi}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(v))))$$

if $z \in \mathtt{fv}(w) \setminus \mathtt{fv}(v)$, or into

$$t \equiv_{\mathcal{A}} \mathcal{C}_x^{y|y'}(\mathcal{C}_{\Phi',y}^{(\Upsilon',z)|(\Psi',y'')}(R_{\Upsilon}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(w))\ R_{\Psi}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(v))))$$

if $z \in \mathtt{fv}(v) \setminus \mathtt{fv}(w)$, or into

$$t \equiv_{\mathcal{A}} \mathcal{C}_x^{y|z}(\mathcal{C}_{\Phi'',y,z}^{(\Upsilon'',y',y'')|(\Psi'',z',z'')}(R_{\Upsilon}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(w))\ R_{\Psi}^{\Phi}(\mathtt{AR}_{\mathcal{A}}(v))))$$

if $z \in \mathtt{fv}(v) \cap \mathtt{fv}(w)$.

In the first (resp. second and third) case, t can be $\mathtt{CA}_{\mathtt{L}}$ (resp. $\mathtt{CA}_{\mathtt{R}}$ and ($\mathtt{CA}_{\mathtt{L}}$ or $\mathtt{CA}_{\mathtt{R}}$))-reduced on $\mathcal{C}_y^{y'|z}()$ (resp. $\mathcal{C}_y^{z|y''}()$ and ($\mathcal{C}_y^{y'|z'}()$ or $\mathcal{C}_z^{y''|z''}()$)). In both cases, it contradicts the fact that t is in $\mathcal{A}$-normal form. Hence, $y \notin \Phi$ (and similarly $z \notin \Phi$).

Now suppose that both y and z are on the same side, say in w. Then t can be $\mathtt{CA}_{\mathtt{L}}$-reduced on $\mathcal{C}_x^{y|z}()$. Similarly, they cannot be both in v. Hence one of them is only in w, and the other is only in v, as required.

– $\mathtt{w} \notin \mathcal{A}$. Then, we have $y, z \in \mathtt{fv}(u)$, otherwise t could be reduced by $\mathtt{CGc}$. The reasoning is then similar to the previous case except that here $\mathtt{RR}_{\mathcal{A}}(u)$ cannot be a variable otherwise it would be $\mathtt{CGc}$-reducible; and $y, z \in \mathtt{RR}_{\mathcal{A}}(u)$ by the i.h. and the fact that $\mathtt{AR}_{\mathcal{A}}()$ preserves free variables.

□

To illustrate Lemma 5.3.10 let us consider the term $t = \mathcal{W}_w(\lambda x.\mathcal{C}_x^{y|z}(y\ z))$. Then, $\mathtt{RR}_{\{\mathtt{C},\mathtt{W}\}}(t) = \lambda x.x\ x$, $\mathtt{AR}_{\{\mathtt{C},\mathtt{W}\}}(\mathtt{RR}_{\{\mathtt{C},\mathtt{W}\}}(t)) = \lambda x.\mathcal{C}_x^{y|z}(y\ z)$. We can conclude since $\mathtt{fv}(t) \setminus \mathtt{fv}(\mathtt{RR}_{\{\mathtt{C},\mathtt{W}\}}(t)) = w$.

Corollary 5.3.11. *Let $\emptyset \neq \mathcal{A} \subseteq \mathcal{R}$. Then, the unique $\mathcal{A}$-normal form of $t \in \mathcal{T}_{\mathcal{A}}$ is $\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t))$ if $\mathtt{w} \notin \mathcal{A}$, and $\mathcal{W}_{\mathtt{fv}(t)\setminus\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t))}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t)))$ if $\mathtt{w} \in \mathcal{A}$.*

Proof. Suppose $\mathtt{w} \in \mathcal{A}$. Termination of $\to_{\mathcal{A}}$ (Lemma 5.3.9) implies that there is t' in $\mathcal{A}$-normal form such that $t \to_{\mathcal{A}}^* t'$. By Lemma 5.1.20, $\mathtt{fv}(t) = \mathtt{fv}(t')$ and by Theorem 5.3.7, $\mathtt{RR}_{\mathcal{A}}(t) = \mathtt{RR}_{\mathcal{A}}(t')$. Since t' is in $\mathcal{A}$-normal form, then $t' \equiv_{\mathcal{A}} \mathcal{W}_{\mathtt{fv}(t')\setminus\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t'))}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t')))$ by Lemma 5.3.10 and thus we have that $t' \equiv_{\mathcal{A}} \mathcal{W}_{\mathtt{fv}(t)\setminus\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t))}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t)))$. To show uniqueness, let us consider two $\mathcal{A}$-normal forms t'_1 and t'_2 of t. By the previous remark, both t'_1 and t'_2 are congruent to the term $\mathcal{W}_{\mathtt{fv}(t)\setminus\mathtt{fv}(\mathtt{RR}_{\mathcal{A}}(t))}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(t)))$ which concludes the case. The case $\mathtt{w} \notin \mathcal{A}$ is similar. □

5.4 Untyped Properties

We first show PSN for all the calculi of the prismoid. The proof will be split in two different subcases, one for each base. This dissociation comes from the fact that redexes are erased by β-reduction in base $\mathfrak{B}_I$ while they are erased by $\mathtt{SGc}$ and/or $\mathtt{SW_1}$-reduction in base $\mathfrak{B}_E$.

Theorem 5.4.1 (PSN)**.** *Let $\mathcal{B} \subseteq \mathcal{S}$ and $\mathcal{A} = \mathcal{B} \setminus \{\mathtt{s}\}$. If $t \in \mathcal{T}_{\emptyset}$ & $t \in \mathcal{SN}_{\emptyset}$, then $\mathtt{AR}_{\mathcal{A}}(t) \in \mathcal{SN}_{\mathcal{B}}$.*

Proof. There are three cases, one for $\mathfrak{B}_I$ and two subcases for $\mathfrak{B}_E$.

- Suppose $\mathtt{s} \notin \mathcal{B}$. We first show that $u \in \mathcal{T}_{\mathcal{B}}$ & $\mathtt{RR}_{\mathcal{B}}(u) \in \mathcal{SN}_{\emptyset}$ imply $u \in \mathcal{SN}_{\mathcal{B}}$. For that we apply Theorem 2.1.6 with $R_1 = \to_{\beta}$, $R_2 = \to_{\mathcal{B}\setminus\beta}$, $R = \to_{\beta}$ and $g = \mathtt{RR}_{\mathcal{B}}(\_)$, using Theorem 5.3.7 and Lemma 5.3.9. Take $u = \mathtt{AR}_{\mathcal{B}}(t)$. Then $\mathtt{RR}_{\mathcal{B}}(\mathtt{AR}_{\mathcal{B}}(t)) =_{L.5.3.8} t \in \mathcal{SN}_{\emptyset}$ by hypothesis. Thus, $\mathtt{AR}_{\mathcal{B}}(t) \in \mathcal{SN}_{\mathcal{B}}$.

- Suppose $\mathcal{B} = \{\mathtt{s}\}$. The proof of $\mathtt{AR}_{\mathtt{s}}(t) = t \in \mathcal{SN}_{\mathtt{s}}$ is given in Chapter 3 using a modular proof technique to show PSN of calculi with full composition.

- Suppose $\mathtt{s} \in \mathcal{B}$. Then $\mathcal{B} = \{\mathtt{s}\} \cup \mathcal{A}$. We show that $u \in \mathcal{T}_{\mathcal{B}}$ & $\mathtt{RR}_{\mathcal{A}}(u) \in \mathcal{SN}_{\mathtt{s}}$ imply $u \in \mathcal{SN}_{\mathcal{B}}$. For that we apply Theorem 2.1.6 with $R_1 = \to_{\mathtt{s}}$, $R_2 = \to_{\mathcal{B}\setminus\mathtt{s}}$, $R = \to_{\mathtt{s}}$ and $g = \mathtt{RR}_{\mathcal{A}}(\_)$, using Theorem 5.3.7 and Lemma 5.3.9.

  Now, take $u = \mathtt{AR}_{\mathcal{A}}(t)$. We have $\mathtt{RR}_{\mathcal{A}}(\mathtt{AR}_{\mathcal{A}}(t)) =_{L.5.3.8} t \in \mathcal{SN}_{\emptyset}$ by hypothesis and $t \in \mathcal{SN}_{\mathtt{s}}$ by the previous point. Thus, $\mathtt{AR}_{\mathcal{A}}(t) \in \mathcal{SN}_{\mathcal{B}}$.

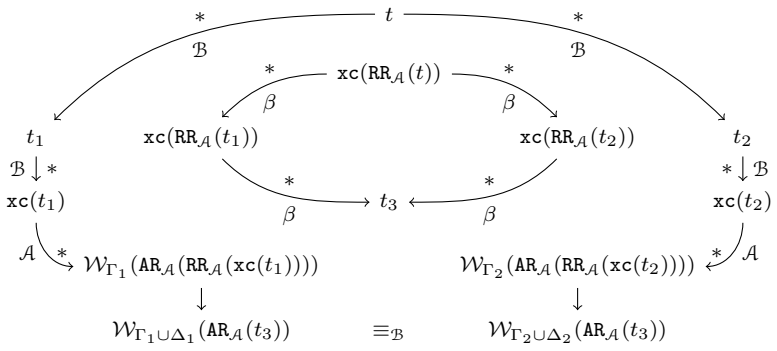


Figure 5.3: Confluence diagram

□

Confluence of each calculus of the prismoid is based on that of the $\lambda_\emptyset$-calculus [Bar84]. For any $\mathcal{A} \subseteq \mathcal{R}$, consider $\mathtt{xc} : \mathcal{T}_{\{\mathtt{s}\}\cup\mathcal{A}} \mapsto \mathcal{T}_\mathcal{A}$ which replaces explicit by implicit substitution.

$$
\begin{array}{lcl@{\qquad}lcl}
\mathtt{xc}(y) & := & y & \mathtt{xc}(\mathcal{W}_y(t)) & := & \mathcal{W}_y(\mathtt{xc}(t)) \\
\mathtt{xc}(t\ u) & := & \mathtt{xc}(t)\ \mathtt{xc}(u) & \mathtt{xc}(\mathcal{C}_y^{y_1|y_2}(t)) & := & \mathcal{C}_y^{y_1|y_2}(\mathtt{xc}(t)) \\
\mathtt{xc}(\lambda y.t) & := & \lambda y.\mathtt{xc}(t) & \mathtt{xc}(t[y/u]) & := & \mathtt{xc}(t)\{y/\mathtt{xc}(u)\}
\end{array}
$$

Lemma 5.4.2. *Let $t \in \mathcal{T}_\mathcal{B}$. Then $t \to_\mathcal{B}^* \mathtt{xc}(t)$.*

Proof. By induction on $\mathtt{size}(t)$ using Lemma 5.1.22. □

Lemma 5.4.3. *Let $t \in \mathcal{T}_\mathcal{B}$. Then $\mathtt{RR}_{\mathcal{B}\setminus\mathtt{s}}(\mathtt{xc}(t)) = \mathtt{xc}(\mathtt{RR}_{\mathcal{B}\setminus\mathtt{s}}(t))$.*

Proof. By induction on $\mathtt{size}(t)$ using Lemma 5.3.6. □

Lemma 5.4.4. *Let $t \in \mathcal{T}_\mathtt{s}$. If $t \to_\mathtt{s} u$, then $\mathtt{xc}(t) \to_\beta^* \mathtt{xc}(u)$.*

Proof. By induction on $t \to_\mathtt{s} u$ using the simplified (but equivalent) notion of substitution on s-terms given in Section 5.1. □

Theorem 5.4.5 (Confluence)**.** *Every calculus $\lambda_\mathcal{B}$ of the prismoid is confluent modulo $\equiv_\mathcal{B}$.*

Proof. The proof is diagrammatically described in Figure 5.3.

Let $t \to_\mathcal{B}^* t_1$ and $t \to_\mathcal{B}^* t_2$. We remark that $\mathcal{B} = \mathcal{A}$ or $\mathcal{B} = \{\mathtt{s}\} \cup \mathcal{A}$, with $\mathcal{A} \subseteq \mathcal{R}$. We have $\mathtt{RR}_\mathcal{A}(t) \to_{\mathcal{B}\setminus\mathcal{A}}^* \mathtt{RR}_\mathcal{A}(t_i)$ (i=1,2) by Theorem 5.3.7. Furthermore $\mathtt{xc}(\mathtt{RR}_\mathcal{A}(t)) \to_\beta^* \mathtt{xc}(\mathtt{RR}_\mathcal{A}(t_i))$ (i=1,2) by Lemma 5.4.4 and $\mathtt{xc}(\mathtt{RR}_\mathcal{A}(t_i)) \to_\beta^* t_3$ (i=1,2) for some $t_3 \in \mathcal{T}_\emptyset$ by confluence of the λ-calculus [Bar84]. We also have $\mathtt{AR}_\mathcal{A}(\mathtt{RR}_\mathcal{A}(\mathtt{xc}(t_i))) =_{L.\ 5.4.3} \mathtt{AR}_\mathcal{A}(\mathtt{xc}(\mathtt{RR}_\mathcal{A}(t_i))) \to_\mathcal{A}^* \mathcal{W}_{\Delta_i}(\mathtt{AR}_\mathcal{A}(t_3))$ for some Δ_i (i=1,2) by Theorem 5.2.4.

$$\frac{}{x : T \vdash_{\mathcal{B}} x : T}$$

$$\frac{\Gamma \vdash_{\mathcal{B}} t : U}{\Gamma \setminus_{\mathcal{B}} x : T \vdash_{\mathcal{B}} \lambda x.t : T \to U} \qquad \frac{\Gamma \vdash_{\mathcal{B}} u : U \quad \Delta \vdash_{\mathcal{B}} t : T}{\Gamma \uplus_{\mathcal{B}} (\Delta \setminus_{\mathcal{B}} x : U) \vdash_{\mathcal{B}} t[x/u] : T}\ (\mathtt{s} \in \mathcal{B})$$

$$\frac{\Gamma \vdash_{\mathcal{B}} t : T \to U \quad \Delta \vdash_{\mathcal{B}} u : T}{\Gamma \uplus_{\mathcal{B}} \Delta \vdash_{\mathcal{B}} tu : U} \qquad \frac{\Gamma \vdash_{\mathcal{B}} t : T}{\Gamma; x : U \vdash_{\mathcal{B}} \mathcal{W}_x(t) : T}\ (\mathtt{w} \in \mathcal{B})$$

$$\frac{\Gamma \vdash_{\mathcal{B}} t : T}{x : U; (\Gamma \setminus_{\mathcal{B}} \{y : U, z : U\}) \vdash_{\mathcal{B}} \mathcal{C}_x^{y|z}(t) : T}\ (\mathtt{c} \in \mathcal{B})$$

Figure 5.4: Typing rules

Lemmas 5.4.2 and Corollary 5.3.11 give $t_i \to_{\mathcal{B}}^* \mathtt{xc}(t_i) \to_{\mathcal{A}}^* \mathcal{W}_{\Gamma_i}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(\mathtt{xc}(t_i))))$ for some Γ_i (i=1,2). Then we get $\mathcal{W}_{\Gamma_i}(\mathtt{AR}_{\mathcal{A}}(\mathtt{RR}_{\mathcal{A}}(\mathtt{xc}(t_i)))) \to_{\mathcal{A}}^* \mathcal{W}_{\Gamma_i \cup \Delta_i}(\mathtt{AR}_{\mathcal{A}}(t_3))$ (i=1,2). Now, $\to_{\mathcal{A}}^* \subseteq \to_{\mathcal{B}}^*$ so in order to close the diagram we reason as follows.

If $\mathtt{w} \notin \mathcal{B}$, then $\Gamma_1 \cup \Delta_1 = \Gamma_2 \cup \Delta_2 = \emptyset$ and we are done. If $\mathtt{w} \in \mathcal{B}$, then $\to_{\mathcal{B}}$ preserves free variables by Lemma 5.1.20 so that $\mathtt{fv}(t) = \mathtt{fv}(t_i) = \mathtt{fv}(\mathcal{W}_{\Gamma_i \cup \Delta_i}(\mathtt{AR}_{\mathcal{A}}(t_3)))$ (i=1,2) which gives $\Gamma_1 \cup \Delta_1 = \Gamma_2 \cup \Delta_2$. □

5.5 Typing

We now introduce simply typed terms for all the calculi of the prismoid, and show that they all enjoy strong normalisation.

Definition 5.5.1 (Prismoid typing rules)**.** *Typing rules described in Figure 5.5 extend the inductive rules for well-formed terms (Section 5.1) with type annotations.*

Thus, typed terms are necessarily well-formed and each set of sorts $\mathcal{B}$ has its own set of typing rules. A term $t \in \mathcal{T}_{\mathcal{B}}$ with type T is written $t \in \mathcal{T}_{\mathcal{B}}^T$.

Lemma 5.5.2. *If $\Gamma \vdash_{\mathcal{B}} t : T$, then*

1. $\mathtt{fv}(t) = \mathtt{dom}(\Gamma)$,
2. $\Lambda; R_{\mathcal{S}}^{\mathtt{dom}(\Pi)}(\Pi) \vdash_{\mathcal{B}} R_{\mathcal{S}}^{\mathtt{dom}(\Pi)}(t) : T$, *where* $\Gamma = \Lambda; \Pi$ *and* $\mathcal{S}$ *is a fresh set of variables.*
3. $\mathtt{RR}_{\mathcal{A}}(t) \in \mathcal{T}_{\mathcal{B}\setminus\mathcal{A}}^T$, *for every* $\mathcal{A} \subseteq \mathcal{R}$.

Proof. By induction on $\Gamma \vdash_{\mathcal{B}} t : T$. □

Theorem 5.5.3 (Subject Reduction)**.** *If $t \in \mathcal{T}_{\mathcal{B}}^T$ & $t \to_{\mathcal{B}} u$, then $u \in \mathcal{T}_{\mathcal{B}}^T$.*

Proof. By induction on the reduction relation using Lemma 5.5.2. The proof is very similar to that of Lemma 5.1.20.

We consider interesting cases when the reduction/equivalence relation is at the root:

- $\mathcal{C}_x^{y|z}(s)[x/v] \to_{\mathtt{SCa}} \mathcal{C}_\Gamma^{\Delta|\Pi}(s[y/R_\Delta^\Gamma(v)][z/R_\Pi^\Gamma(v)])$, with $\Gamma = \mathtt{fv}(u)$ & Δ, Π fresh. Since $\mathtt{c} \in \mathcal{B}$ we know that $\uplus_\mathcal{B}$ is disjoint union so that the type derivation of t looks like:

$$\dfrac{\Gamma \vdash v : C \qquad \dfrac{\Lambda \vdash s : T}{x : C; \Lambda \setminus\!\!\setminus_\mathcal{B} \{y : C, z : C\} \vdash \mathcal{C}_x^{y|z}(s) : T}}{\Gamma; (\Lambda \setminus\!\!\setminus_\mathcal{B} \{y : C, z : C\}) \vdash \mathcal{C}_x^{y|z}(s)[x/v] : T}$$

We then construct the following type derivation:

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash v : C}{\Pi \vdash R_\Pi^\Gamma(v) : C} \qquad \dfrac{\dfrac{\Gamma \vdash v : C}{\Delta \vdash R_\Delta^\Gamma(v) : C} \qquad \Lambda \vdash s : T}{\Delta; (\Lambda \setminus\!\!\setminus_\mathcal{B} y : C) \vdash s[y/R_\Delta^\Gamma(v)] : T}}{\Pi; \Delta; ((\Lambda \setminus\!\!\setminus_\mathcal{B} y : C) \setminus\!\!\setminus_\mathcal{B} z : C) \vdash s[y/R_\Delta^\Gamma(v)][z/R_\Pi^\Gamma(v)] : T}}{\Gamma; (\Lambda \setminus\!\!\setminus_\mathcal{B} y : C \setminus\!\!\setminus_\mathcal{B} z : C) \vdash \mathcal{C}_\Gamma^{\Delta|\Pi}(s[y/R_\Delta^\Gamma(v)][z/R_\Pi^\Gamma(v)]) : T}$$

We conclude since $\Lambda \setminus\!\!\setminus_\mathcal{B} \{y : C, z : C\} = \Lambda \setminus\!\!\setminus_\mathcal{B} y : C \setminus\!\!\setminus_\mathcal{B} z : C$.

- $\mathcal{C}_w^{y|z}(t)[x/u] \to_{\mathtt{SCb}} \mathcal{C}_w^{y|z}(t[x/u])$, with $x \neq w$ & $y, z \notin \mathtt{fv}(u)$.

$$\dfrac{\Gamma \vdash u : C \qquad \dfrac{\Delta \vdash t : T}{\Delta \setminus\!\!\setminus_\mathcal{B} \{y : B; z : B\}; w : B \vdash \mathcal{C}_w^{y|z}(t) : T}}{\Gamma \uplus_\mathcal{B} ((\Delta \setminus\!\!\setminus_\mathcal{B} \{y : B; z : B\}; w : B) \setminus\!\!\setminus_\mathcal{B} x : C) \vdash \mathcal{C}_w^{y|z}(t)[x/u] : T}$$

$$\dfrac{\dfrac{\Gamma \vdash u : C \qquad \Delta \vdash t : T}{\Gamma \uplus_\mathcal{B} (\Delta \setminus\!\!\setminus_\mathcal{B} x : C) \vdash t[x/u] : T}}{(\Gamma \uplus_\mathcal{B} (\Delta \setminus\!\!\setminus_\mathcal{B} x : C)) \setminus\!\!\setminus_\mathcal{B} \{y : B; z : B\}; w : B \vdash \mathcal{C}_w^{y|z}(t[x/u]) : T}$$

We conclude since

$$\begin{array}{lcl}
\Gamma \uplus_\mathcal{B} ((\Delta \setminus\!\!\setminus_\mathcal{B} \{y : B; z : B\}; w : B) \setminus\!\!\setminus_\mathcal{B} x : C) & = & (x \neq w) \\
\Gamma \uplus_\mathcal{B} (\Delta \setminus\!\!\setminus_\mathcal{B} \{y : B; z : B\} \setminus\!\!\setminus_\mathcal{B} x : C; w : B) & = & (w \notin \Gamma) \\
\Gamma \uplus_\mathcal{B} (\Delta \setminus\!\!\setminus_\mathcal{B} x : C \setminus\!\!\setminus_\mathcal{B} \{y : B; z : B\}); w : B & = & (y, z \notin \Gamma) \\
(\Gamma \uplus_\mathcal{B} (\Delta \setminus\!\!\setminus_\mathcal{B} x : C)) \setminus\!\!\setminus_\mathcal{B} \{y : B; z : B\}; w : B & &
\end{array}$$

- $\mathcal{C}_w^{y|z}(\mathcal{W}_y(t)) \to_{\mathtt{CW_1}} R_w^z(t)$.

$$\dfrac{\dfrac{\Gamma \vdash t : T}{\Gamma; y : B \vdash \mathcal{W}_y(t) : T}}{w : B; \Gamma \setminus\!\!\setminus_{\mathcal{B}} z : B \vdash \mathcal{C}_w^{y|z}(\mathcal{W}_y(t)) : T}$$

Then Lemma 5.5.2 allows to conclude $\Gamma \setminus\!\!\setminus_{\mathcal{B}} z : B; w : B \vdash R_w^z(t) : T$.

- $\mathcal{C}_w^{x|z}(\mathcal{C}_x^{y|p}(s)) \equiv_{\mathtt{CC}_{\mathcal{A}}} \mathcal{C}_w^{x|y}(\mathcal{C}_x^{z|p}(s))$.

$$\dfrac{\dfrac{\Gamma \vdash s : T}{\Gamma \setminus\!\!\setminus_{\mathcal{B}} \{y : B; p : B\}; x : B \vdash \mathcal{C}_x^{y|p}(s) : T}}{\Gamma \setminus\!\!\setminus_{\mathcal{B}} \{y : B; p : B\} \setminus\!\!\setminus_{\mathcal{B}} z : B; w : B \vdash \mathcal{C}_w^{x|z}(\mathcal{C}_x^{y|p}(s)) : T}$$

$$\dfrac{\dfrac{\Gamma \vdash s : T}{\Gamma \setminus\!\!\setminus_{\mathcal{B}} \{z : B; p : B\}; x : B \vdash \mathcal{C}_x^{z|p}(s) : T}}{\Gamma \setminus\!\!\setminus_{\mathcal{B}} \{z : B; p : B\} \setminus\!\!\setminus_{\mathcal{B}} y : B; w : B \vdash \mathcal{C}_w^{x|y}(\mathcal{C}_x^{y|p}(s)) : T}$$

We conclude since $\Gamma \setminus\!\!\setminus_{\mathcal{B}} \{y : B; p : B\} \setminus\!\!\setminus_{\mathcal{B}} z : B = \Gamma \setminus\!\!\setminus_{\mathcal{B}} \{z : B; p : B\} \setminus\!\!\setminus_{\mathcal{B}} y : B$

- $\mathcal{C}_{x'}^{y'|z'}(\mathcal{C}_x^{y|z}(t)) \equiv_{\mathtt{CC}_{\mathcal{C}}} \mathcal{C}_x^{y|z}(\mathcal{C}_{x'}^{y'|z'}(t))$, with $x \neq y', z'$ & $x' \neq y, z$.

$$\dfrac{\dfrac{\Gamma \vdash t : T}{\Gamma \setminus\!\!\setminus_{\mathcal{B}} \{y : B; z : B\}; x : B \vdash \mathcal{C}_x^{y|z}(t) : T}}{\Gamma \setminus\!\!\setminus_{\mathcal{B}} \{y : B; z : B\} \setminus\!\!\setminus_{\mathcal{B}} \{y' : C; z' : C\}; x : B; x' : C \vdash \mathcal{C}_{x'}^{y'|z'}(\mathcal{C}_x^{y|z}(t)) : T}$$

$$\dfrac{\dfrac{\Gamma \vdash t : T}{\Gamma \setminus\!\!\setminus_{\mathcal{B}} \{y' : C; z' : C\}; x' : C \vdash \mathcal{C}_{x'}^{y'|z'}(t) : T}}{\Gamma \setminus\!\!\setminus_{\mathcal{B}} \{y' : B; z' : B\} \setminus\!\!\setminus_{\mathcal{B}} \{y : B; z : B\}; x : B; x' : B \vdash \mathcal{C}_x^{y|z}(\mathcal{C}_{x'}^{y'|z'}(t)) : T}$$

We conclude since $\Gamma \setminus\!\!\setminus_{\mathcal{B}} \{y : B; z : B\} \setminus\!\!\setminus_{\mathcal{B}} \{y' : C; z' : C\} = \Gamma \setminus\!\!\setminus_{\mathcal{B}} \{y' : C; z' : C\} \setminus\!\!\setminus_{\mathcal{B}} \{y : B; z : B\}$.

- $s[x/u][y/v] \equiv_{\mathtt{SS}_{\mathcal{C}}} s[y/v][x/u]$, with $y \notin \mathtt{fv}(u)$ & $x \notin \mathtt{fv}(v)$.

$$\dfrac{\Gamma \vdash v : B \qquad \dfrac{\Pi \vdash u : C \qquad \Delta \vdash s : T}{\Pi \uplus_{\mathcal{B}} (\Delta \setminus\!\!\setminus_{\mathcal{B}} x : C) \vdash s[x/u] : T}}{\Gamma \uplus_{\mathcal{B}} ((\Pi \uplus_{\mathcal{B}} (\Delta \setminus\!\!\setminus_{\mathcal{B}} x : C)) \setminus\!\!\setminus_{\mathcal{B}} y : B) \vdash s[x/u][y/v] : T}$$

$$\dfrac{\Pi \vdash u : C \qquad \dfrac{\Gamma \vdash v : B \qquad \Delta \vdash s : T}{\Gamma \uplus_{\mathcal{B}} (\Delta \setminus\!\!\setminus_{\mathcal{B}} y : B) \vdash s[y/v] : T}}{\Pi \uplus_{\mathcal{B}} ((\Gamma \uplus_{\mathcal{B}} (\Delta \setminus\!\!\setminus_{\mathcal{B}} y : B)) \setminus\!\!\setminus_{\mathcal{B}} x : C) \vdash s[y/v][x/u] : T}$$

We conclude since $y \notin \Pi$ and $x \notin \Gamma$ imply

$\Gamma \uplus_{\mathcal{B}} ((\Pi \uplus_{\mathcal{B}} (\Delta \setminus\!\!\setminus_{\mathcal{B}} x : C)) \setminus\!\!\setminus_{\mathcal{B}} y : B) = \Gamma \uplus_{\mathcal{B}} \Pi \uplus_{\mathcal{B}} (\Delta \setminus\!\!\setminus_{\mathcal{B}} x : C \setminus\!\!\setminus_{\mathcal{B}} y : B) = \Pi \uplus_{\mathcal{B}} ((\Gamma \uplus_{\mathcal{B}} (\Delta \setminus\!\!\setminus_{\mathcal{B}} y : B)) \setminus\!\!\setminus_{\mathcal{B}} x : C)$.

Now, we consider the reduction/equivalence relation is not at the root. We just consider one case as all the other ones are similar.

If $t = uv \Rightarrow_{\mathcal{B}_1} u'v$, with $u \Rightarrow_{\mathcal{B}_1} u'$, then we have

$$\frac{\Gamma \vdash_{\mathcal{B}} u : B \to T \qquad \Delta \vdash_{\mathcal{B}} v : B}{\Gamma \uplus_{\mathcal{B}} \Delta \vdash_{\mathcal{B}} uv : T}$$

The i.h. gives $\Gamma' \vdash_{\mathcal{B}} u' : B \to T$ with $\Gamma' \subseteq \Gamma$. Then $\Gamma' \uplus_{\mathcal{B}} \Delta \subseteq \Gamma \uplus_{\mathcal{B}} \Delta$ and we thus conclude. □

Corollary 5.5.4 (Strong Normalisation)**.** *Let $t \in \mathcal{T}_{\mathcal{B}}^{T}$, then $t \in \mathcal{SN}_{\mathcal{B}}$.*

Proof. Let $\mathcal{A} \subseteq \mathcal{R}$ so that $\mathcal{B} = \mathcal{A}$ or $\mathcal{B} = \mathcal{A} \cup \{\mathtt{s}\}$. It is well-known that (simply) typed $\lambda_{\emptyset}$-calculus is strongly normalising (see for example [Bar84]). It is also straightforward to show that PSN for the $\lambda_{\mathtt{s}}$-calculus implies strong normalisation for well-typed $\mathtt{s}$-terms (see for example [Kes07]). By Theorem 5.3.7 any infinite $\mathcal{B}$-reduction sequence starting at t can be projected into an infinite $(\mathcal{B} \setminus \mathcal{A})$-reduction sequence starting at $\mathtt{RR}_{\mathcal{A}}(t)$. By Lemma 5.5.2 $\mathtt{RR}_{\mathcal{A}}(t)$ is a well-typed $(\mathcal{B} \setminus \mathcal{A})$-term, that is, a well-typed term in $\lambda_{\emptyset}$ or $\lambda_{\mathtt{s}}$. This leads to a contradiction.

□

5.6 Appendix

The $\lambda_{\emptyset}$-calculus

Rules :

$$(\beta) \quad (\lambda x.t)\ u \ \to\ t\{x/u\}$$

The $\lambda_{\mathtt{C}}$-calculus

Equations :

$$\begin{array}{llll}
(\mathtt{CC}_{\mathcal{A}}) & \mathcal{C}_w^{x|z}(\mathcal{C}_x^{y|p}(t)) & \equiv & \mathcal{C}_w^{x|y}(\mathcal{C}_x^{z|p}(t)) \\
(\mathtt{C}_{\mathcal{C}}) & \mathcal{C}_x^{y|z}(t) & \equiv & \mathcal{C}_x^{z|y}(t) \\
(\mathtt{CC}_{\mathcal{C}}) & \mathcal{C}_{x'}^{y'|z'}(\mathcal{C}_x^{y|z}(t)) & \equiv & \mathcal{C}_x^{y|z}(\mathcal{C}_{x'}^{y'|z'}(t)) \qquad x \neq y', z' \ \&\ x' \neq y, z
\end{array}$$

Rules :

$$\begin{array}{llll}
(\beta) & (\lambda x.t)\ u & \to & t\{x/u\} \\
(\mathtt{CL}) & \mathcal{C}_w^{y|z}(\lambda x.t) & \to & \lambda x.\mathcal{C}_w^{y|z}(t) \\
(\mathtt{CA_L}) & \mathcal{C}_w^{y|z}(t\ u) & \to & \mathcal{C}_w^{y|z}(t)\ u \qquad y, z \notin fv(u) \\
(\mathtt{CA_R}) & \mathcal{C}_w^{y|z}(t\ u) & \to & t\ \mathcal{C}_w^{y|z}(u) \qquad y, z \notin fv(t) \\
(\mathtt{CGc}) & \mathcal{C}_w^{y|z}(t) & \to & R_w^{z}(t) \qquad y \notin fv(t)
\end{array}$$

The $\lambda_{\mathtt{S}}$-calculus

Equations :

$(\mathtt{SS}_{\mathcal{C}})$	$t[x/u][y/v]$	$\equiv$	$t[y/v][x/u]$	$y \notin fv(u)$ & $x \notin fv(v)$

Rules :

$(\mathtt{B})$	$(\lambda x.t)\ u$	$\to$	$t[x/u]$	
$(\mathtt{V})$	$x[x/u]$	$\to$	u	
$(\mathtt{SGc})$	$t[x/u]$	$\to$	t	$x \notin fv(t)$
$(\mathtt{SDup})$	$t[x/u]$	$\to$	$t_{[y]_x}[x/u][y/u]$	$\lvert t\rvert_x > 1$ & y fresh
$(\mathtt{SL})$	$(\lambda y.t)[x/u]$	$\to$	$\lambda y.t[x/u]$	
$(\mathtt{SA_L})$	$(t\ v)[x/u]$	$\to$	$t[x/u]\ v$	$x \notin fv(v)$
$(\mathtt{SA_R})$	$(t\ v)[x/u]$	$\to$	$t\ v[x/u]$	$x \notin fv(t)$
$(\mathtt{SS})$	$t[y/v][x/u]$	$\to$	$t[y/v[x/u]]$	$x \notin fv(t)$ & $x \in fv(v)$

The $\lambda_{\mathtt{W}}$-calculus

Equations :

$(\mathtt{WW}_{\mathcal{C}})$	$\mathcal{W}_x(\mathcal{W}_y(t))$	$\equiv$	$\mathcal{W}_y(\mathcal{W}_x(t))$

Rules :

(β)	$(\lambda x.t)\ u$	$\to$	$t\{x/u\}$	
$(\mathtt{LW})$	$\lambda x.\mathcal{W}_y(t)$	$\to$	$\mathcal{W}_y(\lambda x.t)$	$x \neq y$
$(\mathtt{AW_l})$	$\mathcal{W}_y(u)v$	$\to$	$\mathcal{W}_{y\setminus \mathtt{fv}(v)}(uv)$	
$(\mathtt{AW_r})$	$u\mathcal{W}_y(v)$	$\to$	$\mathcal{W}_{y\setminus \mathtt{fv}(u)}(uv)$	

The $\lambda_{\mathtt{CS}}$-calculus

Equations :

$(\mathtt{CC}_{\mathcal{A}})$	$\mathcal{C}_w^{x\vert z}(\mathcal{C}_x^{y\vert p}(t))$	$\equiv$	$\mathcal{C}_w^{x\vert y}(\mathcal{C}_x^{z\vert p}(t))$	
$(\mathtt{C}_{\mathcal{C}})$	$\mathcal{C}_x^{y\vert z}(t)$	$\equiv$	$\mathcal{C}_x^{z\vert y}(t)$	
$(\mathtt{CC}_{\mathcal{C}})$	$\mathcal{C}_{x'}^{y'\vert z'}(\mathcal{C}_x^{y\vert z}(t))$	$\equiv$	$\mathcal{C}_x^{y\vert z}(\mathcal{C}_{x'}^{y'\vert z'}(t))$	$x \neq y', z'$ & $x' \neq y, z$
$(\mathtt{SS}_{\mathcal{C}})$	$t[x/u][y/v]$	$\equiv$	$t[y/v][x/u]$	$y \notin \mathtt{fv}(u)$ & $x \notin \mathtt{fv}(v)$

Rules :

$(\mathtt{B})$	$(\lambda x.t)\ u$	$\to$	$t[x/u]$	
$(\mathtt{CL})$	$\mathcal{C}_w^{y\vert z}(\lambda x.t)$	$\to$	$\lambda x.\mathcal{C}_w^{y\vert z}(t)$	
$(\mathtt{CA_L})$	$\mathcal{C}_w^{y\vert z}(tu)$	$\to$	$\mathcal{C}_w^{y\vert z}(t)u$	$y, z \notin \mathtt{fv}(u)$
$(\mathtt{CA_R})$	$\mathcal{C}_w^{y\vert z}(tu)$	$\to$	$t\mathcal{C}_w^{y\vert z}(u)$	$y, z \notin \mathtt{fv}(t)$
$(\mathtt{CGc})$	$\mathcal{C}_w^{y\vert z}(t)$	$\to$	$R_w^z(t)$	$y \notin \mathtt{fv}(t)$
$(\mathtt{V})$	$x[x/u]$	$\to$	u	
$(\mathtt{SGc})$	$t[x/u]$	$\to$	t	$x \notin \mathtt{fv}(t)$
$(\mathtt{SL})$	$(\lambda y.t)[x/u]$	$\to$	$\lambda y.t[x/u]$	
$(\mathtt{SA_L})$	$(tv)[x/u]$	$\to$	$t[x/u]v$	$x \notin \mathtt{fv}(v)$
$(\mathtt{SA_R})$	$(tv)[x/u]$	$\to$	$tv[x/u]$	$x \notin \mathtt{fv}(t)$
$(\mathtt{SS})$	$t[x/u][y/v]$	$\to$	$t[x/u[y/v]]$	$y \notin \mathtt{fv}(t)$ & $y \in \mathtt{fv}^+(u)$
$(\mathtt{SCa})$	$\mathcal{C}_x^{y\vert z}(t)[x/u]$	$\to$	$\mathcal{C}_\Gamma^{\Delta\vert\Pi}(t[y/R_\Delta^\Gamma(u)][z/R_\Pi^\Gamma(u)])$	$\begin{cases} y, z \in \mathtt{fv}^+(t) \\ \Gamma = \mathtt{fv}(u) \\ \Delta, \Pi \text{ fresh} \end{cases}$
$(\mathtt{CS})$	$\mathcal{C}_w^{y\vert z}(t[x/u])$	$\to$	$t[x/\mathcal{C}_w^{y\vert z}(u)]$	$y, z \in \mathtt{fv}^+(u)$
$(\mathtt{SCb})$	$\mathcal{C}_w^{y\vert z}(t)[x/u]$	$\to$	$\mathcal{C}_w^{y\vert z}(t[x/u])$	$x \neq w$ & $y, z \notin \mathtt{fv}(u)$

The $\lambda_{\mathtt{CW}}$-calculus

Equations :

$(\mathtt{CC}_{\mathcal{A}})$	$\mathcal{C}_w^{x\|z}(\mathcal{C}_x^{y\|p}(t))$	$\equiv$	$\mathcal{C}_w^{x\|y}(\mathcal{C}_x^{z\|p}(t))$	
$(\mathtt{C}_{\mathcal{C}})$	$\mathcal{C}_x^{y\|z}(t)$	$\equiv$	$\mathcal{C}_x^{z\|y}(t)$	
$(\mathtt{CC}_{\mathcal{C}})$	$\mathcal{C}_{x'}^{y'\|z'}(\mathcal{C}_x^{y\|z}(t))$	$\equiv$	$\mathcal{C}_x^{y\|z}(\mathcal{C}_{x'}^{y'\|z'}(t))$	$x \neq y', z'$ & $x' \neq y, z$
$(\mathtt{WW}_{\mathcal{C}})$	$\mathcal{W}_x(\mathcal{W}_y(t))$	$\equiv$	$\mathcal{W}_y(\mathcal{W}_x(t))$	

Rules :

(β)	$(\lambda x.t)\ u$	$\to$	$t\{x/u\}$	
$(\mathtt{LW})$	$\lambda x.\mathcal{W}_y(t)$	$\to$	$\mathcal{W}_y(\lambda x.t)$	$x \neq y$
$(\mathtt{AW_l})$	$\mathcal{W}_y(u)v$	$\to$	$\mathcal{W}_{y\setminus\mathtt{fv}(v)}(uv)$	
$(\mathtt{AW_r})$	$u\mathcal{W}_y(v)$	$\to$	$\mathcal{W}_{y\setminus\mathtt{fv}(u)}(uv)$	
$(\mathtt{CL})$	$\mathcal{C}_w^{y\|z}(\lambda x.t)$	$\to$	$\lambda x.\mathcal{C}_w^{y\|z}(t)$	
$(\mathtt{CA_L})$	$\mathcal{C}_w^{y\|z}(tu)$	$\to$	$\mathcal{C}_w^{y\|z}(t)u$	$y, z \notin \mathtt{fv}(u)$
$(\mathtt{CA_R})$	$\mathcal{C}_w^{y\|z}(tu)$	$\to$	$t\mathcal{C}_w^{y\|z}(u)$	$y, z \notin \mathtt{fv}(t)$
$(\mathtt{CW_1})$	$\mathcal{C}_w^{y\|z}(\mathcal{W}_y(t))$	$\to$	$R_w^z(t)$	
$(\mathtt{CW_2})$	$\mathcal{C}_w^{y\|z}(\mathcal{W}_x(t))$	$\to$	$\mathcal{W}_x(\mathcal{C}_w^{y\|z}(t))$	$x \neq y, z$
$(\mathtt{CGc})$	$\mathcal{C}_w^{y\|z}(t)$	$\to$	$R_w^z(t)$	$y \notin \mathtt{fv}(t)$

The $\lambda_{\mathtt{SW}}$-calculus

Equations :

$(\mathtt{WW}_{\mathcal{C}})$	$\mathcal{W}_x(\mathcal{W}_y(t))$	$\equiv$	$\mathcal{W}_y(\mathcal{W}_x(t))$	
$(\mathtt{SS}_{\mathcal{C}})$	$t[x/u][y/v]$	$\equiv$	$t[y/v][x/u]$	$y \notin \mathtt{fv}(u)$ & $x \notin \mathtt{fv}(v)$

Rules :

$(\mathtt{B})$	$(\lambda x.t)\ u$	$\to$	$t[x/u]$	
$(\mathtt{LW})$	$\lambda x.\mathcal{W}_y(t)$	$\to$	$\mathcal{W}_y(\lambda x.t)$	$x \neq y$
$(\mathtt{AW_l})$	$\mathcal{W}_y(u)v$	$\to$	$\mathcal{W}_{y\setminus\mathtt{fv}(v)}(uv)$	
$(\mathtt{AW_r})$	$u\mathcal{W}_y(v)$	$\to$	$\mathcal{W}_{y\setminus\mathtt{fv}(u)}(uv)$	
$(\mathtt{V})$	$x[x/u]$	$\to$	u	
$(\mathtt{SGc})$	$t[x/u]$	$\to$	t	$x \notin fv(t)$
$(\mathtt{SDup})$	$t[x/u]$	$\to$	$t_{[y]_x}[x/u][y/u]$	$\|t\|_x^+ > 1$ & y fresh
$(\mathtt{SL})$	$(\lambda y.t)[x/u]$	$\to$	$\lambda y.t[x/u]$	
$(\mathtt{SA_L})$	$(t\ v)[x/u]$	$\to$	$t[x/u]\ v$	$x \notin fv(v)$
$(\mathtt{SA_R})$	$(t\ v)[x/u]$	$\to$	$t\ v[x/u]$	$x \notin fv(t)$
$(\mathtt{SS})$	$t[y/v][x/u]$	$\to$	$t[y/v[x/u]]$	$x \notin fv(t)$ & $x \in fv(v)$
$(\mathtt{SW_1})$	$\mathcal{W}_x(t)[x/u]$	$\to$	$\mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(t)$	
$(\mathtt{SW_2})$	$\mathcal{W}_y(t)[x/u]$	$\to$	$\mathcal{W}_{y\setminus\mathtt{fv}(u)}(t[x/u])$	$x \neq y$
$(\mathtt{SW})$	$t[x/\mathcal{W}_y(u)]$	$\to$	$\mathcal{W}_{y\setminus\mathtt{fv}(t)}(t[x/u])$	

The $\lambda_{\mathtt{csw}}$-calculus

Equations :

$(\mathtt{CC}_{\mathcal{A}})$	$\mathcal{C}_w^{x\mid z}(\mathcal{C}_x^{y\mid p}(t))$	$\equiv\ \mathcal{C}_w^{x\mid y}(\mathcal{C}_x^{z\mid p}(t))$	
$(\mathtt{C}_{\mathcal{C}})$	$\mathcal{C}_x^{y\mid z}(t)$	$\equiv\ \mathcal{C}_x^{z\mid y}(t)$	
$(\mathtt{CC}_{\mathcal{C}})$	$\mathcal{C}_{x'}^{y'\mid z'}(\mathcal{C}_x^{y\mid z}(t))$	$\equiv\ \mathcal{C}_x^{y\mid z}(\mathcal{C}_{x'}^{y'\mid z'}(t))$	$x \neq y', z'$ & $x' \neq y, z$
$(\mathtt{WW}_{\mathcal{C}})$	$\mathcal{W}_x(\mathcal{W}_y(t))$	$\equiv\ \mathcal{W}_y(\mathcal{W}_x(t))$	
$(\mathtt{SS}_{\mathcal{C}})$	$t[x/u][y/v]$	$\equiv\ t[y/v][x/u]$	$y \notin \mathtt{fv}(u)$ & $x \notin \mathtt{fv}(v)$

Rules :

$(\mathtt{B})$	$(\lambda x.t)\ u$	$\to t[x/u]$	
$(\mathtt{V})$	$x[x/u]$	$\to u$	
$(\mathtt{SDup})$	$t[x/u]$	$\to t_{[y]_x}[x/u][y/u]$	$\lvert t\rvert_x^+ > 1$ & y fresh
$(\mathtt{SL})$	$(\lambda y.t)[x/u]$	$\to \lambda y.t[x/u]$	
$(\mathtt{SA_L})$	$(tv)[x/u]$	$\to t[x/u]v$	$x \notin \mathtt{fv}(v)$
$(\mathtt{SA_R})$	$(tv)[x/u]$	$\to tv[x/u]$	$x \notin \mathtt{fv}(t)$
$(\mathtt{SS})$	$t[x/u][y/v]$	$\to t[x/u[y/v]]$	$y \notin \mathtt{fv}(t)$ & $y \in \mathtt{fv}^+(u)$
$(\mathtt{SW_1})$	$\mathcal{W}_x(t)[x/u]$	$\to \mathcal{W}_{\mathtt{fv}(u)\setminus\mathtt{fv}(t)}(t)$	
$(\mathtt{SW_2})$	$\mathcal{W}_y(t)[x/u]$	$\to \mathcal{W}_{y\setminus\mathtt{fv}(u)}(t[x/u])$	$x \neq y$
$(\mathtt{LW})$	$\lambda x.\mathcal{W}_y(t)$	$\to \mathcal{W}_y(\lambda x.t)$	$x \neq y$
$(\mathtt{AW_l})$	$\mathcal{W}_y(u)v$	$\to \mathcal{W}_{y\setminus\mathtt{fv}(v)}(uv)$	
$(\mathtt{AW_r})$	$u\mathcal{W}_y(v)$	$\to \mathcal{W}_{y\setminus\mathtt{fv}(u)}(uv)$	
$(\mathtt{SW})$	$t[x/\mathcal{W}_y(u)]$	$\to \mathcal{W}_{y\setminus\mathtt{fv}(t)}(t[x/u])$	
$(\mathtt{SCa})$	$\mathcal{C}_x^{y\mid z}(t)[x/u]$	$\to \mathcal{C}_\Gamma^{\Delta\mid\Pi}(t[y/R_\Delta^\Gamma(u)][z/R_\Pi^\Gamma(u)])$	$\begin{cases} y, z \in \mathtt{fv}^+(t) \\ \Gamma = \mathtt{fv}(u) \\ \Delta, \Pi \text{ fresh} \end{cases}$
$(\mathtt{CL})$	$\mathcal{C}_w^{y\mid z}(\lambda x.t)$	$\to \lambda x.\mathcal{C}_w^{y\mid z}(t)$	
$(\mathtt{CA_L})$	$\mathcal{C}_w^{y\mid z}(tu)$	$\to \mathcal{C}_w^{y\mid z}(t)u$	$y, z \notin \mathtt{fv}(u)$
$(\mathtt{CA_R})$	$\mathcal{C}_w^{y\mid z}(tu)$	$\to t\mathcal{C}_w^{y\mid z}(u)$	$y, z \notin \mathtt{fv}(t)$
$(\mathtt{CS})$	$\mathcal{C}_w^{y\mid z}(t[x/u])$	$\to t[x/\mathcal{C}_w^{y\mid z}(u)]$	$y, z \in \mathtt{fv}^+(u)$
$(\mathtt{SCb})$	$\mathcal{C}_w^{y\mid z}(t)[x/u]$	$\to \mathcal{C}_w^{y\mid z}(t[x/u])$	$x \neq w$ & $y, z \notin \mathtt{fv}(u)$
$(\mathtt{CW_1})$	$\mathcal{C}_w^{y\mid z}(\mathcal{W}_y(t))$	$\to R_w^z(t)$	
$(\mathtt{CW_2})$	$\mathcal{C}_w^{y\mid z}(\mathcal{W}_x(t))$	$\to \mathcal{W}_x(\mathcal{C}_w^{y\mid z}(t))$	$x \neq y, z$

Chapter 6

Towards Real Life

Contents

In this Chapter we present two different works which remind us that explicit substitutions were originally created by a need in the formal design of functional programming languages, and that was only afterwards that questions such as PSN or metaconfluence were raised.

In the first section, we try to give ideas to answer the following question: if the calculi with explicit substitutions are closer to practical implementations, then to what extent are they closer compared to the λ-calculus? To achieve this, we conjecture bounds on the lengths of reduction for a simple calculus with explicit substitutions.

In the second section, we present a formalisation of the $\lambda\mathtt{j}$-calculus in Coq, explain why it is so complex.

6.1 Complexity

Exact bounds for lengths of reductions are known for the simply typed λ-calculus. In the case of typed λ-calculi with explicit substitutions, the cost of the reduction is not constant w.r.t. the β-reduction and thus the bound rises. We conjecture both lower and upper bounds depending on the degree and the size of the term from the point of view of maximal strategies.

It is interesting to quantify the cost brought by the explicitation of the substitution, as it is more realistic. To quantify it, we will consider bounds for lengths of reductions. Following works of Schwichtenberg ([Sch82], [Sch91]), based on ideas from [How80], Beckmann [Bec01] showed exact bounds (upper and lower) for lengths of reductions in the case of λ-calculus, which depend on the size and the degree of a term. By bound, we actually mean given a set of terms having same degree and same size (or height), we pick up a term with a longest reduction (there can be several terms with the same length of reduction) and give bounds on this reduction. For a term t with degree n and size N, the bound is an exponential tower of 2 of height n and N at the top, namely $2^{\cdot^{\cdot^{\cdot^{2^N}}}}$.

Beckmann's proof for upper bound proceeds as follows: embed a term t in a tree structure, according to its size and its degree; rewrite the tree into an another one using a simpler grammar. In this simple tree, the number of its nodes is a bound of the longest reduction.

In this section, we will proceed as for the λ-calculus and give upper and lower bounds for a calculus with explicit substitutions. We will use maximal strategies which can be described by inductive definitions on terms choosing which redex to fire in order to have the longest reduction. They are well known and immediately give constructions needed to naturally extend Beckmann's proof and thus keep the same scheme for the upper bound proof. However, we will have to deal with the complexification induced by the number of rules. In our case, the bound will still be an exponential tower of same height but with 4 instead of 2.

Related works: Upper bounds weaker than Schwichtenberg's were also found by [Loa98]. Miranda gave in [dM03] an historical overview and made clearer the link between expanded head trees and maximal strategies. In [Sch01] the complexity of the decision problem, whether a term t reduces to a term t', was studied for levels ranging from 1 to 3. In [Xi99], a bound similar to the one in [Sch91] was found, using bounds extracted from a standardisation theorem proof. In [BW00] bounds for Gödel System T were ginve using the same technique as in [Bec01]. Beckman's result was strengthened in [AJ05] using continuous normalisation.

6.1.1 The calculus with explicit substitutions

We will consider the $\lambda\mathtt{x}^-$-calculus with the simple typing system given in Definition 2.3.3. From now one, we will only consider well-typed terms. We introduce the key notion, the maximal strategy, $F^{\mathtt{x}}_{\infty}()$:

Definition 6.1.1. *Let $C[\![\_]\!]$ be a head-context of the shape $\_\ [\overline{x}/\overline{u}]\ \overline{v}$ where $\overline{v} = v_1 \ldots v_m$, $t[\overline{x}/\overline{u}] := t[x_1/u_1]\ldots[x_n/u_n]$; and $D[\![\_]\!]$ a head-context of the shape $\_\ \overline{v}$. If t is a $\lambda\mathtt{x}^-$ term, then $F_{\infty}(t)$ is inductively defined on t, provided that t is not in*

normal form:

$$
\begin{array}{lcll}
F^{\mathtt{x}}_{\infty}(D[\![(\lambda x.t)\ u]\!]) & := & D[\![t[x/u]]\!] & \\
F^{\mathtt{x}}_{\infty}(C[\![(\lambda y.t)[x/u]]\!]) & := & C[\![\lambda y.t[x/u]]\!] & \\
F^{\mathtt{x}}_{\infty}(C[\![(v\ w)[x/u]]\!]) & := & C[\![v[x/u]\ w[x/u]]\!] & \\
F^{\mathtt{x}}_{\infty}(C[\![x[x/t]]\!]) & := & C[\![t]\!] & \\
F^{\mathtt{x}}_{\infty}(C[\![z[x/u]]\!]) & := & \begin{cases} C[\![z[x/F^{\mathtt{x}}_{\infty}(u)]]\!] & \textit{if } u \notin \mathcal{NF}_{\lambda\mathtt{x}^-} \\ C[\![z]\!] & \textit{otherwise} \end{cases} & x \neq z \\
F^{\mathtt{x}}_{\infty}(\lambda x.t) & := & \lambda x.F^{\mathtt{x}}_{\infty}(t) & \\
F^{\mathtt{x}}_{\infty}(xt_1...t_n) & := & xt_1...F^{\mathtt{x}}_{\infty}(t_j)...t_n & \forall i < j, t_i \in \mathcal{NF}_{\lambda\mathtt{x}^-}
\end{array}
$$

Using the context C instead of D in $F^{\mathtt{x}}_{\infty}(D[\![(\lambda x.t)\ u]\!])$ would lead to perform the step $((\lambda x.t)u)[y/v] \to_B t[x/u][y/v]$ instead of $((\lambda x.t)\ u)[y/v] \to_{SApp} (\lambda x.t)[y/v]\ u[y/v]$ whereas only the last one is maximal.

6.1.2 Upper bounds

We are interested in giving bounds on the following functions:

$$dl_n(N) := max\{\eta_{\lambda\mathtt{x}^-}(t) : t \text{ a term}, g(t) \leq n, |t| \leq N\}$$

$$dh_n(N) := max\{\eta_{\lambda\mathtt{x}^-}(t) : t \text{ a term}, g(t) \leq n, h(t) \leq N\}$$

i.e. given a set of $\lambda\mathtt{x}^-$-terms with same degree and same size (or height), we pick up a term with a longest reduction and we give bounds on this reduction.

In [Bec01], the exact bounds for the λ-calculus are $dh(N) = 2^{\Theta(N)}_{n+1}$ and $dl(N) = 2^{\Theta(N)}_{n}$ where $\Theta(N)$ is the usual notion to compare functions growth [CLRS01] and 2^m_n is inductively defined by $2^m_{n+1} = 2^{2^m_n}$ and $2^m_0 = m$.

We want to embed each term in a tree representing its longest reduction in the sense that it is possible to define a route in the tree that mimics choices of the maximal reduction. Such trees will be called expanded head tree or more simply eht. Then we count the number of nodes in this tree, and deduce the longest possible reduction. However, we cannot embed a term directly in this kind of tree because doing so, we do not know how to guess the height of the resulting tree whereas this is an indispensable information to estimate the number of nodes.

Therefore we use another kind of trees (associated binary trees, abt) , via an embedding depending on the degree and the size of the term.

Both kinds of trees are defined by induction. Associated binary trees use one more rule than expanded head trees, namely the (Cut)-rule. As cut-elimination in logic, we will remove cuts level by level, quantifying the increase of nodes in this operation. When all cuts have been removed, we obtain an eht and we have the desired property i.e. the number of nodes is a bound. Since we know how many nodes we have when we embed, and how many we add when we eliminate cuts, then we finally deduce the number of nodes and conclude.

The general scheme of the proof can be graphically summarised as:

$$\boxed{term} \xrightarrow[\text{Lemma 6.1.16}]{\text{embedding}} \text{abt} \xrightarrow[\text{Lemma 6.1.15}]{\text{cut-elim}} \text{eht} \xrightarrow[\text{Lemma 6.1.17}]{\text{estimation}} \boxed{bound}$$

To show the connection between the trees we are going to use and maximal strategies, we first define both of them for the λ-calculus. We give the definition of an associated binary tree as in [Bec01].

Definition 6.1.2 (Associated Binary Tree for the λ-calculus)**.** *An abt $\vdash^{\alpha}_{\rho} t$ for the λ-term t with $\alpha, \rho \in \mathbb{N}$ is inductively defined (in a bottom-up fashion) by the following rules:*

$$\begin{array}{rl}
(\underline{\beta}) & \textit{If } \vdash^{\alpha}_{\rho} t\{x/u\}\ \overline{v} \textit{ and } \vdash^{\alpha}_{\rho} u, \textit{ then } \vdash^{\alpha+1}_{\rho} (\lambda x.t)u\ \overline{v} \\
(Lam) & \textit{If } \vdash^{\alpha}_{\rho} t, \textit{ then } \vdash^{\alpha+1}_{\rho} \lambda x.t \\
(Var) & \textit{If } \vdash^{\alpha}_{\rho} t_1, ..., \vdash^{\alpha}_{\rho} t_n\ (n \geq 0), \textit{ then } \vdash^{\alpha+n}_{\rho} xt_1...t_n \\
(Cut) & \textit{If } \vdash^{\alpha}_{\rho} t \textit{ with } lv(t) \leq \rho \textit{ and } \vdash^{\alpha}_{\rho} u, \textit{ then } \vdash^{\alpha+1}_{\rho} t\ u
\end{array}$$

With rule (Var)we have $\vdash^{\alpha}_{\rho} x$ for any x, α, ρ.

In $\vdash^{\alpha}_{\rho} t$, α corresponds to the height of the tree and ρ to the level up to which it is allowed to use the rule (Cut). In the sequel, we will often use the following rule, admissible thanks to the particular case of (Var)where we have $\vdash^{\alpha}_{\rho} x$ for any α, ρ and variable x: if $\vdash^{\alpha}_{\rho} t$ and $\alpha \leq \alpha', \rho \leq \rho'$, then $\vdash^{\alpha'}_{\rho'} t$.

We could also have defined binary rules, like for instance β, if $\vdash^{\alpha}_{\rho} t\{x/u\}\ \overline{v}$ and $\vdash^{\beta}_{\rho} v$, then $\vdash^{max(\alpha,\beta)+1}_{\rho} (\lambda x.t)u\ \overline{v}$. Doing so, we remove the need of this admissible rule, but it is less convenient to use.

We specify an equivalent definition of $F^{\beta}_{\infty}(\_)$ for the λ-calculus:

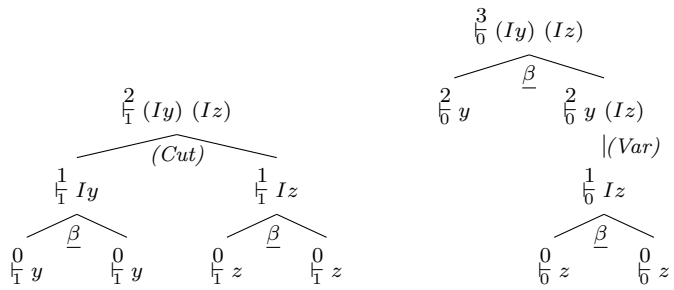
Definition 6.1.3. *The maximal strategy $F^{\beta}_{\infty}(\_)$ on a λ-term t s.t. $t \notin \mathcal{NF}_{\lambda}$ is the following:*

$$\begin{array}{lcl}
F^{\beta}_{\infty}(D[\![(\lambda x.t)\ u]\!]) & := & \begin{cases} D[\![(\lambda x.t)\ F^{\beta}_{\infty}(u)]\!] & \textit{if } x \notin \mathtt{fv}(t) \textit{ and } u \notin \mathcal{NF} \\ D[\![t\{x/u\}]\!] & \textit{otherwise} \end{cases} \\
F^{\beta}_{\infty}(\lambda x.t) & := & \lambda x.F^{\beta}_{\infty}(t) \\
F^{\beta}_{\infty}(x\ t_1\ ...\ t_n) & := & x\ t_1\ ...F^{\beta}_{\infty}(t_i)\ ...\ t_n \forall i < j, t_i \in \mathcal{NF}_{\beta}
\end{array}$$

A condition giving two cases in the strategy will give a binary node in the associated binary tree, to keep track of all possibilities.

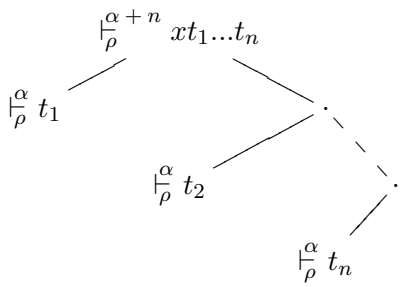
In general, all rules used to build the abt which are underlined (for example $\underline{\beta}$ in Definition 6.1.2) are the ones performed by $F_{\infty}(\_)$, reading downward in the tree. Rules (Lam)and (Var)are looking for a leftmost-outermost redex to contract, as in $F_{\infty}(\_)$ except that rules in the strategy do not deconstruct. Rules (Lam), (Var)and (Cut)will be the same in our case, and would be also the same for other systems with explicit substitutions. The underlined rules are specific to the maximal strategy in the system considered. Due to the fact that the (Cut)-rule overlaps rules $\underline{\beta}$ and (Var), for any term t, there can be several abts.

Example 6.1.4. *Two different abts for the term* $(Iy)\ (Iz)$ *where* $I \equiv \lambda x.x$ *are:*



In the tree on the left, as $\rho = 1$*, we can use the rule (Cut)to obtain* $(Iy)\ (Iz)$ *(since* $lv((Iy)) = 1$ *as* $I\ y$ *is of type* $\alpha \to \alpha$*) but not to obtain* Iy *from* I *and* y *(since* $lv(I) = 2$*).*

Example 6.1.5. *The shape of a term in head normal form is like:*



The (Var)rule is designed with $\vdash^{\alpha+n}_{\rho}\ x\ t_1...t_n$ *and not* $\vdash^{\alpha+1}_{\rho}\ xt_1...t_n$ *because it is easier to count nodes in trees with a fixed number of children, as in Lemma 6.1.17.*

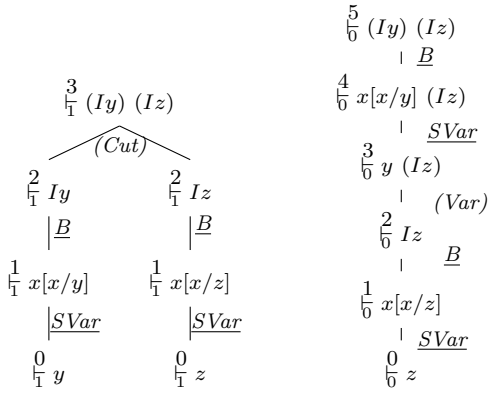
We now give the definition of associated binary trees, which is adapted from $F^{\mathtt{x}}_{\infty}(\_)$ to the case of $\lambda\mathtt{x}^{-}$.

Definition 6.1.6 (Associated Binary Tree)**.** *An abt* $\vdash^{\alpha}_{\rho} t$ *for the* $\lambda\mathtt{x}^{-}$ *term* t *with* $\alpha, \rho \in \mathbb{N}$ *is inductively defined by the following rules where* $C[\![\_]\!]$ *and* $D[\![\_]\!]$ *are as defined for* $F^{\mathtt{x}}_{\infty}(\_)$*:*

$\underline{B}$	*If* $\vdash^{\alpha}_{\rho} D[\![t[x/u]]\!]$*, then* $\vdash^{\alpha+1}_{\rho} D[\![(\lambda x.t)u]\!]$
$\underline{SLam}$	*If* $\vdash^{\alpha}_{\rho} C[\![\lambda y.t[x/u]]\!]$*, then* $\vdash^{\alpha+1}_{\rho} C[\![(\lambda y.t)[x/u]]\!]$
$\underline{SApp}$	*If* $\vdash^{\alpha}_{\rho} C[\![v[x/u]\ w[x/u]]\!]$*, then* $\vdash^{\alpha+1}_{\rho} C[\![(v\ w)[x/u]]\!]$
$\underline{SVar}$	*If* $\vdash^{\alpha}_{\rho} C[\![t]\!]$*, then* $\vdash^{\alpha+1}_{\rho} C[\![x[x/t]]\!]$
$\underline{SGc}$	*If* $\vdash^{\alpha}_{\rho} C[\![z]\!]$ *and* $\vdash^{\alpha}_{\rho} u$*, then* $\vdash^{\alpha+1}_{\rho} C[\![z[x/u]]\!]$
λ	*If* $\vdash^{\alpha}_{\rho} t$*, then* $\vdash^{\alpha+1}_{\rho} \lambda x.t$
(Var)	*If* $\vdash^{\alpha}_{\rho} t_1, ..., \vdash^{\alpha}_{\rho} t_n$*, then* $\vdash^{\alpha+n}_{\rho} xt_1...t_n$
(Cut)	*If* $\vdash^{\alpha}_{\rho} t$ *with* $lv(t) \leq \rho$ *and* $\vdash^{\alpha}_{\rho} u$*, then* $\vdash^{\alpha+1}_{\rho} t\ u$

Notice that B, unlike β in λ-calculus, does not erase its argument and thus $\underline{B}$ is unary. However SGc does and $\underline{SGc}$ is binary to keep a copy of the erased subterm in the tree. As for the λ-calculus, underlined rules correspond to the maximal strategy.

Example 6.1.7. *Two different ABT for the term $(Iy)\ (Iz)$, as in Example 6.1.4:*



Definition 6.1.8 (Expanded Head Tree)**.** *An abt $\vdash^{\alpha}_{0} t$ associated to t is called its expanded head tree.*

An expanded head tree does not use the rule (Cut)and thus is unique, unlike an arbitrary abt.

Definition 6.1.9 (Number of nodes)**.** *The number of nodes $\#t$ of an expanded head tree $\vdash^{\alpha}_{0} t$ is defined by induction on t:*

$$
\begin{array}{lcl}
\#D[\![(\lambda x.t)u]\!] & := & \#D[\![t[x/u]]\!] + 1 \\
\#C[\![(\lambda y.t)[x/u]]\!] & := & \#C[\![\lambda y.t[x/u]]\!] + 1 \\
\#C[\![(v\ w)[x/u]]\!] & := & \#C[\![v[x/u]\ w[x/u]]\!] + 1 \\
\#C[\![x[x/u]]\!] & := & \#C[\![u]\!] + 1 \\
\#C[\![z[x/u]]\!] & := & \#C[\![z]\!] + \#u + 1 \\
\#\lambda x.t & := & \#t + 1 \\
\#(x\ t_1 ... t_n) & := & \sum_{i=1}^{n} \#t_i
\end{array}
$$

We easily notice that for a given eht $\vdash^{\alpha}_{0} t$ we have that for each maximal step performed on t there is a corresponding edge (and thus a node) in the tree. Actually for most of the steps in the strategy, we just go down in the tree. If $t = C[\![z[x/u]]\!]$ we have all the nodes of the subtree associated to u for the maximal reduction of u and nodes of $C[\![z]\!]$ for the rest of the remaining steps.

Lemma 6.1.10. *Let t be a $\lambda\mathtt{x}^{-}$-term such that $\vdash^{\alpha}_{0} t$. Then $\#t$ is a bound on the longest reduction from t.*

Notice that without a proof of maximality for $F^{\mathtt{x}}_{\infty}(\_)$ we would have had to do like in [Bec01] with its main lemma: if $t \to_{\lambda\mathtt{x}^{-}} t'$ then $\#t' < \#t$ by case analysis on

the reduction rule. However we could not deduce a proof of maximality for $F^{\mathtt{x}}_{\infty}(\_)$ from this lemma since $F^{\mathtt{x}}_{\infty}(\_)$ only visits a subset of the tree.

We now want to embed each $\lambda\mathtt{x}^{-}$ term in an abt. To do this, we need some technical lemmas:

We conjecture this lemma

Conjecture 6.1.11 (Explicit Renaming)**.** *If* $\vdash^{\alpha}_{\rho} t$*, then* $\vdash^{3\alpha+1}_{\rho} t[y/z]$*.*

We are not able to prove it by induction on the derivation of $\vdash^{\alpha}_{\rho} t$*. Indeed, in the case where we have* $\vdash^{\alpha+1}_{\rho} D[\![(\lambda x.v)u]\!]$ *coming from* $\vdash^{\alpha}_{\rho} D[\![v[x/u]]\!]$*. We write the context* $D[\![v[x/u]]\!][y/z]$ *like* $(v[x/u]\ t_1...t_n)[y/z]$*. We cannot apply the rule* $\underline{B}$ *since we do not have a* D *context anymore and are thus stuck.*

Example 6.1.12. *Consider an expanded head tree* $\vdash^{1}_{0} x_0\ x_1$*. We have for instance* $\vdash^{4}_{0} (x_0\ x_1)[y/z]$ *Notice that, as the tree is filiform, then the height itself bounds the longest reduction.*

Lemma 6.1.13 (Appending)**.** *If* $\vdash^{\alpha}_{\rho} t$ *and* $t\ y$ *is a typable term, then* $\vdash^{3\alpha}_{\rho} t\ y$*.*

Proof. By induction on $\vdash^{\alpha}_{\rho} t$. We detail less as the reasoning is quite similar to the previous one.

- $\underline{B}$: By induction $\vdash^{3\alpha}_{\rho} D[\![v[x/u]]\!]\ y$ and then by $\underline{B}$, $\vdash^{3\alpha+1}_{\rho} D[\![(\lambda x.v)\ u]\!]\ y$ which concludes since $3\alpha+1 \leq 3\alpha+3$.
- Rules $\underline{SLam}$, $\underline{SApp}$, $\underline{SVar}$ and $\underline{SGc}$ are straightforward by induction.
- (Lam): By Lemma 6.1.11, $\vdash^{3\alpha+1}_{\rho} t'[x/y]$, then by the rule $\underline{B}$, $\vdash^{3\alpha+2}_{\rho} (\lambda x.t')\ y$.
- (Var): By hypothesis, $\vdash^{\alpha}_{\rho} t_1$, ..., $\vdash^{\alpha}_{\rho} t_n$. With the rule (Var), we also have $\vdash^{\alpha}_{\rho} y$ and we conclude by a last application of (Var)$\vdash^{\alpha+n+1}_{\rho} xt_1...t_ny$.
- (Cut): $\vdash^{\alpha+1}_{\rho} t\ u$ comes from $\vdash^{\alpha}_{\rho} t$ with $lv(t) \leq \rho$ and $\vdash^{\alpha}_{\rho} u$. We have that $lv(tu) \leq \rho$ since $lv(tu) \leq lv(t)$. With (Var), we have $\vdash^{\alpha+1}_{\rho} y$ and we can apply the rule (Cut)to get $\vdash^{\alpha+2}_{\rho} t\ u\ y$.

□

We now extend the implicit substitution to $\lambda\mathtt{x}^{-}$-terms: $t[x/u]\{y/v\} := t\{y/v\}[x/u\{y/v\}]$.

Lemma 6.1.14 (Implicit Substitution)**.** *If* $\vdash^{\alpha}_{\rho} t$ *and* $\vdash^{\beta}_{\rho} v$ *and* $lv(v) \leq \rho$*, then* $\vdash^{\alpha+\beta}_{\rho} t\{y/v\}$*. We restrict ourselves to the case where there is no node generated by* $\underline{SGc}$ *with* $C[\![y]\!]$ *as a left premise in the tree of* t*.*

Proof. By induction on $\vdash^{\alpha}_{\rho} t$. Let $t^* := t\{y/v\}$.

- $\underline{B}$: By induction hypothesis $\vdash_{\rho}^{\alpha+\beta} D^*[\![t^*[x/u^*]]\!]$, hence by the B-Rule we have $\vdash_{\rho}^{\alpha+\beta+1} D[\![(\lambda x.t^*)\ u^*]\!]$.

- Rules $\underline{SLam}$, $\underline{SApp}$ and $\underline{SVar}$ are straightforward by induction.

- For the case $\underline{SGc}$ we have $\vdash_{\rho}^{\alpha+1} C[\![z[x/u]]\!]$ coming from $\vdash_{\rho}^{\alpha} u$ and $\vdash_{\rho}^{\alpha} C[\![z]\!]$ with $z \neq y$. By induction hypothesis $\vdash_{\rho}^{\alpha+\beta} C^*[\![z]\!]$ and $\vdash_{\rho}^{\alpha+\beta} u^*$. We have $\vdash_{\rho}^{\alpha+\beta+1}$ $C^*[\![z[x/u^*]]\!]$ by $\underline{SGc}$ that is $\vdash_{\rho}^{\alpha+\beta+1} (C[\![z[x/u]]\!])^*$ thanks to our restriction.

- (Lam): By induction hypothesis $\vdash_{\rho}^{\alpha+\beta} t^*$ and $\vdash_{\rho}^{\alpha+\beta+1} \lambda x.t^*$ by the (Lam)-Rule.

- (Var): By induction hypothesis $\vdash_{\rho}^{\alpha+\beta} t_i^*$ hence $\vdash_{\rho}^{\alpha+\beta+n} x\ t_1^*...t_n^*$ by (Var), if $x \neq y$. Otherwise, we can weaken the hypothesis and get $\vdash_{\rho}^{\alpha+\beta} v$. Then we can apply n cuts and get $\vdash_{\rho}^{\alpha+\beta+n} v\ t_1^*...t_n^*$

- (Cut): By induction hypothesis, $\vdash_{\rho}^{\alpha+\beta} t^*$ and $\vdash_{\rho}^{\alpha+\beta} u^*$. As $lv(t) \leq \rho$ we have $lv(t^*) \leq \rho$ and we can apply (Cut)and get $\vdash_{\rho}^{\alpha+\beta+1} t^*u^*$.

□

Lemma 6.1.15 (Cut Elimination)**.** *If* $\vdash_{\rho+1}^{\alpha} t$*, then* $\vdash_{\rho}^{4^\alpha} t$*.*

Proof. All cases are easy by induction except (Cut). Suppose we have $\vdash_{\rho+1}^{\alpha+1} t\ u$ coming from $\vdash_{\rho+1}^{\alpha} t$ and $\vdash_{\rho+1}^{\alpha} u$. Then, by induction hypothesis, $\vdash_{\rho}^{4^\alpha} t$ and $\vdash_{\rho}^{4^\alpha} u$ with $lv(t) \leq \rho + 1$, hence $lv(u) \leq \rho$. By Lemma 6.1.13, $\vdash_{\rho}^{3 \cdot 4^\alpha} t\ x$ with $x \notin \mathtt{fv}(t)$ so we can apply Lemma 6.1.14 and get $\vdash_{\rho}^{4 \cdot 4^\alpha} t\ u$ i.e. $\vdash_{\rho}^{4^{\alpha+1}} t\ u$. □

We now have all necessary lemmas to embed each $\lambda\mathtt{x}^-$ term:

Lemma 6.1.16 (Embedding Lemma)**.** *Let* t *be a* $\lambda\mathtt{x}^-$ *term. If* $g(t) \leq \rho$*, then* $\vdash_{\rho}^{4^{|t|}} t$*.*

Proof. By induction on t.

- $t = x$, $\vdash_{\rho}^{4^{|x|}} x$ by (Var)and weakening.

- $t = \lambda x.u$, $\vdash_{\rho}^{4^{|t|-1}} u$ by induction hypothesis and thus $\vdash_{\rho}^{4^{|t|-1}+1} t$ by (Lam). We can conclude since $4^{|t|-1} + 1 \leq 4^t$ (as $|t| \geq 2$).

- $t = u\ v$, $\vdash_{\rho}^{4^{|t|-1}} u$ and $\vdash_{\rho}^{4^{|t|-1}} v$ by induction hypothesis. By (Cut), $\vdash_{\rho}^{4^{|t|-1}+1} u\ v$ and we conclude as in previous item.

- $t = u[x/v]$, $\vdash_{\rho}^{4^{|t|-2}} u$ and $\vdash_{\rho}^{4^{|t|-2}} v$ by induction hypothesis. By Lemma 6.1.11, $\vdash_{\rho}^{3 \cdot 4^{|t|-2}+1} u[x/y]$ with $y \notin \mathtt{fv}(u)$. Thus, as $lv(v) \leq \rho$ (since v is a subterm of t and $g(t) \leq \rho$) we can apply Lemma 6.1.14 and get $\vdash_{\rho}^{4 \cdot 4^{|t|-2}+1} u[x/v]$ i.e. $\vdash_{\rho}^{4^{|t|-1}+1} u[x/v]$. We can conclude since $4^{|t|-1}+1 \leq 4^{|t|}$ (since $|t| \geq 3$).

□

Since our tree is binary and we know its height, we can immediately estimate the number of nodes:

Lemma 6.1.17 (Estimate Lemma)**.** *If* $\vdash_{0}^{\alpha} t$ *then* $\#t \leq 2^{\alpha}$.

Theorem 6.1.18 (Upper bound)**.** $dl_n(N) \leq 2^{4_n^{|t|}}$ *and* $dh_n(N) \leq 2^{2^{4_n^{|t|}}}$.

Proof. Using the three key lemmas (embedding, cut elimination, estimation), we get that the height of a tree is such that: $height \leq 4_n^{|t|}$. As the tree is binary we have: $\#t \leq 2^{4_{g(t)}^{|t|}}$. Hence by Lemma 6.1.10,

$$dl_n(N) \leq 2^{4_n^{|t|}}$$

As previously noticed, we can derive $dh_n(N)$ since for any term, $|t| \leq 2^{h(t)}$:

$$dh_n(N) \leq 2^{2^{4_n^{|t|}}}$$

□

This result seems reasonable compared to the λ-calculus. However it could be much stronger with a lower bound closed to it.

6.1.3 Lower bounds

Since $\lambda\mathtt{x}^-$ simulates the λ-calculus, the lower bound for the λ-calculus is also a valid lower bound for $\lambda\mathtt{x}^-$. We briefly give terms used in [Bec01] to show optimality.

Let ι be a ground type. For $n \in \mathbb{N}$, let $o(n)$ be a type s.t. $o(0) = \iota$ and $o(n+1) = o(n) \to o(n)$. Notice that $lv(o(n)) = n$. Let $[u]^k(v)$ be the term denoting the k-fold iteration of u applied to v, i.e. $u(...(u\ v)...)$ with k u s. Let σ be a type and $\overline{N}^{\sigma}$ be the generalized Church numeral $\lambda f^{\sigma\to\sigma}\lambda x.[f]^N(x)$. Fix some variable 0 of type ι, and d of type $\iota \to \iota \to \iota$. Let $I = \lambda x^{\iota}.x$ and $D = \lambda x^{\iota}.dxx$. Let $T_k(u)$ be a tree-like term of height k defined as $T_0(u) = u$ and $T_{k+1} = d\ T_k(u)\ T_k(u)$.

We adapt a lemma from [Bec01]:

Lemma 6.1.19. *Let k be a natural number. Then,*

- $[\overline{2}^{\sigma}]^k\ (u)\ v \to^*_{\lambda\mathtt{x}^-} [u]^{2^k}\ (v)$

- $[D]^k(u) \to_{\lambda\mathtt{x}^-} T_k(u)$
- $T_k(I0) \to^{2^{k+1}}_{\lambda\mathtt{x}^-} T_k(0)$

Proof. By induction on k. □

Let $\overline{2}^{o(-1)}$ be D and $\overline{2}^{o(-2)}$ be $I0$. Then we can define A^n_N with degree $n+1$ and size $\mathrm{O}(N)$ [CLRS01], independently of n. Using Lemma 6.1.19 we have:

$$\begin{array}{lll} A^n_N & := & [\overline{2}^{o(n-1)}]^N\ \overline{2}^{o(n-2)}\ \overline{2}^{o(n-3)} ... \overline{2}^{o(-2)} \\ & \to^*_{\lambda\mathtt{x}^-} & [\overline{2}^{o(n-2)}]^{2^N}\ \overline{2}^{o(n-3)}\ ... \overline{2}^{o(-2)} \to_{\lambda\mathtt{x}^-} ... \\ & \to^*_{\lambda\mathtt{x}^-} & [D]^N_{2_n}(I0) \to^*_{\lambda\mathtt{x}^-} T_{2_n}(N)(I0) \\ & \to^{2^N_{n+2}}_{\lambda\mathtt{x}^-} & T_{2_n}(0) \end{array}$$

Thus we have $dl_n(N) \geq 2^N_{n+1}$ i.e. the bound for λ-calculus with one more floor. We can have the same reasoning for $dh_n(N)$ and find a similar bound $dh_n(N) \geq 2^N_{n+2}$. Both bounds are really under the upper bound. We conjecture that both upper and lower bounds can be improved. However, with the approach with terms A^N_n there is no hope in improving the lower because 2^N_n is not significant wrt exponential towers of 4 or even 3. The main difficulty comes from the fact that the maximal strategy for A^n_{N+1} or A^{n+1}_N does not share many reductions with A^n_N and thus it is difficult to find an induction hypothesis.

To investigate, we developed a program which performs the maximal reduction for $\lambda\mathtt{x}^-$. We analyse the size of the terms during reduction and conjecture that the optimal bound is a tower of 3 of height n. it is possible to find shapes of A^n_N graph in those of A^n_{N+1} or A^{n+1}_N in groups of three.

In Figures 6.1.3, 6.1.3 and 6.1.3, we reproduce interesting graphs where the x-coordinates correspond to the number of steps, y-coordinates to the size of the term.

Notice that in the Figure 6.1.3, it is only the beginning of the reduction, and the last part on the right may indicate that it is only a small part. It was impossible to investigate more in this direction due to the fact that reductions take definitely too much time (several days), even for terms A^n_N with $N, n \leq 3$.

To compare, we also reproduce a graph in Figure 6.1.3 for the case of the λ-calculus. Notice that peaks appear here group by 2.

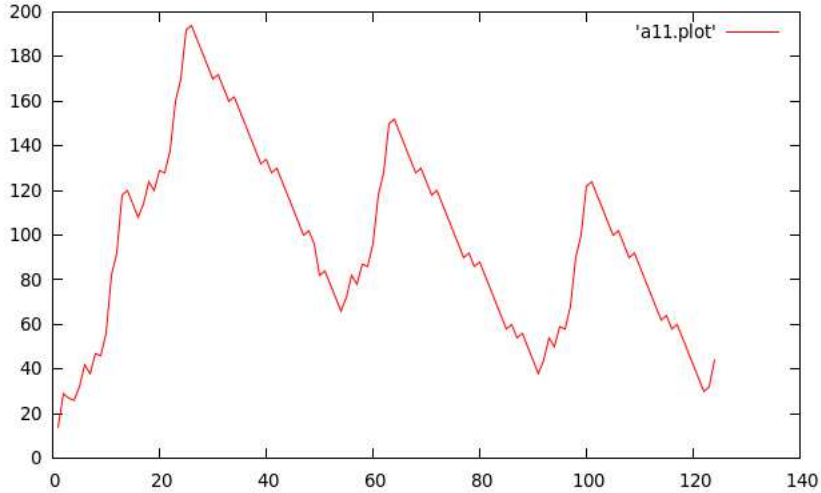


Figure 6.1: Entire reduction of A_1^1

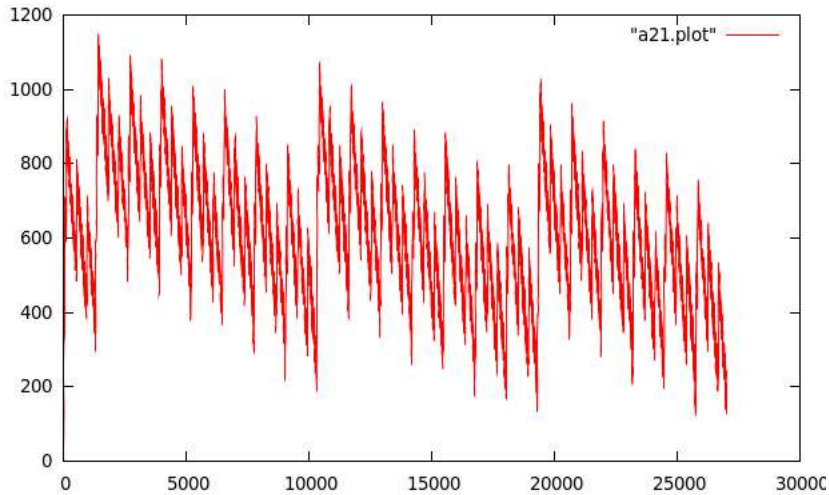


Figure 6.2: Entire reduction of A_2^1

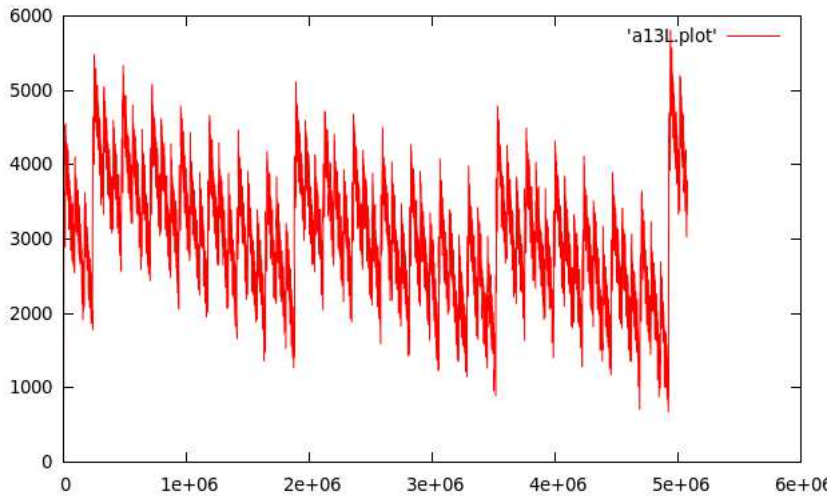


Figure 6.3: First steps of reduction of A_1^3

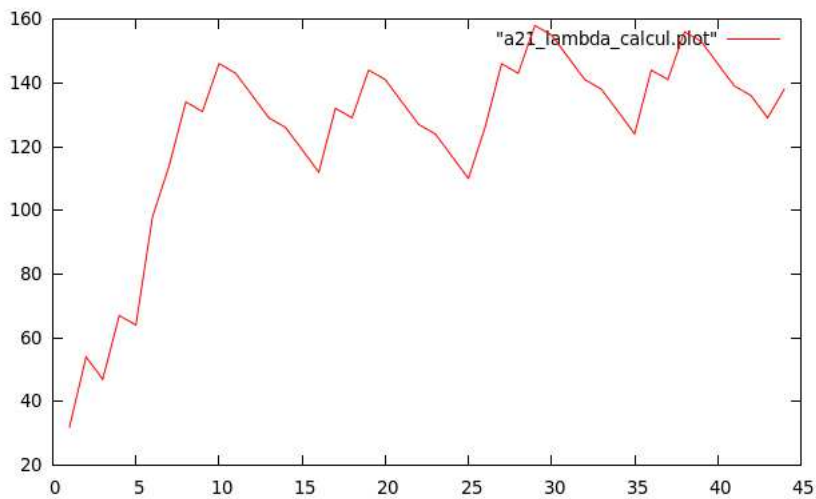


Figure 6.4: Entire reduction of A_2^1 (λ-calculus)

6.2 Formalisation

In this section, we formalise $\lambda\mathtt{j}$ in Coq. The formalisation in its own is interesting since $\lambda\mathtt{j}$ has been defined in its original version with names and not with indexes, and the non-deterministic replacement operation of $\lambda\mathtt{j}$ has not been mathematically defined in [AK10].

The implementation in Coq has been developped in collaboration with Stéphane Zimmermann. Certain technical details are omitted here for a better comprehension. The full development can be browsed or downloaded at https://www.pps.jussieu.fr/~renaud/coq.

6.2.1 Representation of Bound Variables

When studying the theory of calculi with explicit substitutions we have the choice between calculi with indexes or names, because there is not a major influence on the properties that a calculus can enjoy. However, when dealing with implementations, it is quite unrealistic to have simply names and α-conversion. Nevertheless, we still have the choice between several representation of binders.

1. The first possibility is of course to use De Bruijn indexes or levels [Cré07, LRD95]. It removes the need of α-conversion, but it is difficult to manipulate them since indexes or levels have to be shifted when they cross a binder. Both levels and indexes notations are unique for a given term. An advantage of indexes over levels are is that they are context-independant i.e. no index update is required when embedding a term in another. However, when the context is fixed, updates of levels are easier.

2. Another possibility is to use nominal logic [GP02]. It is mainly available for the proof assistant Isabelle [Isa], even if an approach with the Coq assistant has been tried [ABW07].

3. The locally *nameless* representation [Cha11, Gor93], based itself on the locally *named* representation [MP93] is a good compromise between names and indexes. Neither α-conversion nor indexes shifting is needed. This is done using the following grammar of *preterms*:

   $$t ::= \mathtt{trm\_bvar}\ i \mid \mathtt{trm\_fvar}\ x \mid \mathtt{trm\_abs}\ t \mid \mathtt{app}\ t\ t \mid \mathtt{sub}\ t\ t$$

   Free and bound variables have now their own constructors. Bound variables are handled as for De Bruijn indexes and free variables are represented using names, also called *atoms*, as done in the usual theory. Atoms can be implemented with any datatype which supports a comparison function and a generator. We will thus use natural numbers in our implementation. For the sake of clarity, we will however use names in the sequel.

   Furthermore, it seems that the induction principles obtained with this technique are stronger than those obtained automatically with the nominal approach of Isabelle [ACP+08].

Given the familiarity of the author with Coq, and the advantages of the locally nameless representation, we choose this combination.

There exist some attempts to generate and prove all the infrastructure related to the technicalities of the locally nameless representation directly from simple specifications (such as those of Ott [SNO+10]) containing a grammar for terms and a set of rewriting rules. However, as those softwares are still under development ([AW10, SNO+10]), we will thus not use them. Instead, we will adapt all the lemmas needed for the locally nameless infrastructure of the λ-calculus [Cha11].

6.2.2 Preterms, and their Operations

The formal grammar of *preterms* is the following:

```
Inductive pterm : Set :=
  | pterm_bvar : nat -> pterm
  | pterm_fvar : var -> pterm
  | pterm_abs  : pterm -> pterm
  | pterm_app  : pterm -> pterm -> pterm
  | pterm_sub : pterm -> pterm -> pterm.
```

We will use the notation `t`[`u`] to represent `pterm_sub t u`.

We can define the notion of free variables on preterms:

```
Fixpoint fv (t : pterm) : vars :=
  match t with
  | pterm_bvar i    => {}
  | pterm_fvar x    => {{x}}
  | pterm_abs t1    => (fv t1)
  | pterm_app t1 t2 => (fv t1) \u (fv t2)
  | pterm_sub t1 t2 => (fv t1) \u (fv t2)
  end.
```

where {} is the empty set, and {{`x`}} the singleton $\{x\}$.

Even if there is no need to update indexes, there are still several operations to be formally defined. In particular, when a proof needs to consider a subterm, there is some work to do. Indeed, if for the named case we can manipulate the subterm t of $\lambda x.t$, it is not the case for locally nameless terms. For instance, if we take the subterm `trm_bvar` 0 of `trm_abs trm_bvar` 0 (representing $\lambda x.x$), we have a preterm which does not correspond to any λ-term. We would indeed like to have `trm_fvar x` instead.

We will later only consider the subset of locally nameless *terms* satisfying a certain predicate, in order to have a bijection between locally nameless terms and λ-terms equivalence classes. In a first time, to be the most general possible, we will define operations on preterms.

To look to the subterm `t` under an abstraction (or a substitution), we thus need to transform the references to the binder into a fresh free variable `x`. This is the purpose of the *variable opening* operation. Formally, we need to replace all `trm_bvar i` with i equal to the number of binders enclosing the bound variable.

To be more general, we directly define the `open` operation extended to preterms, and not only to variables:

```
Definition open t u := open_rec 0 u t.
```

with open_rec being the generalisation of open to any depth, not only 0:

```
Fixpoint open_rec (k : nat) (u : pterm) (t : pterm) : pterm :=
  match t with
  | pterm_bvar i    => if k = i then u else (pterm_bvar i)
  | pterm_fvar x    => pterm_fvar x
  | pterm_abs t1    => pterm_abs (open_rec (S k) u t1)
  | pterm_app t1 t2 => pterm_app (open_rec k u t1)
                                 (open_rec k u t2)
  | pterm_sub t1 t2 => pterm_sub (open_rec (S k) u t1)
                                 (open_rec k u t2)
  end.
```

We will use the following notations:

```
Notation "{ k ~> u } t" := open_rec k u t.
Notation "t ^^ u" := open t u.
Notation "t ^ x" := open t (pterm_fvar x).
```

Symmetrically, we can define the close operation:

```
Fixpoint close_rec  (k : nat) (x : var) (t : pterm)  : pterm :=
  match t with
  | pterm_bvar i    => pterm_bvar i
  | pterm_fvar x'   => if x' = x then
                           pterm_bvar k
                       else
                           pterm_fvar x'
  | pterm_abs t1    => pterm_abs (close_rec (S k) x t1)
  | pterm_app t1 t2 => pterm_app (close_rec k x t1)
                                 (close_rec k x t2)
  | pterm_sub t1 t2 => (close_rec (S k) x t1)[close_rec k x t2]
  end.

Definition close t x := close_rec 0 x t.
```

As expected we have:

```
Lemma close_open_var: forall x t, x \notin fv t -> close (t^x) x = t.
```

We can also define the implicit substitution on preterms:

```
Fixpoint subst (z : var) (u : pterm) (t : pterm) : pterm :=
  match t with
  | pterm_bvar i    => pterm_bvar i
  | pterm_fvar x    => if x = z then u else (pterm_fvar x)
  | pterm_abs t1    => pterm_abs (subst z u t1)
  | pterm_app t1 t2 => pterm_app (subst z u t1) (subst z u t2)
  | pterm_sub t1 t2 => pterm_sub (subst z u t1) (subst z u t2)
  end.
Notation "[ z ~> u ] t" := (subst z u t).
```

Notice that the implicit substitution can be defined by means of the closing and opening operations:

```
Lemma subst_as_close_open : forall t x u, term t ->
                        [x ~> u] t = (close t x) ^^ u.
```

6.2.3 Locally Closed Terms and Bodies

The grammar of preterms contains much more objects than those in λj. We will thus restrict preterms to locally closed terms, or simply terms. The idea is to have for any bound variable `trm_bvar i` deep inside a preterm at least i binders above it. There are several ways to perform this restriction. The simplest one is to define a term, for instance in the case of the abstraction, by using a statement like "`trm_abs t` is a term if there exists a free variable `x` such that `t ^x` is a term". We will however follow the method suggested in [Cha11] which uses cofinite quantification. The idea is to change the previous definition into: `trm_abs t` is a term if for any set of variables, there exists a free variable `x` which is not in this set such that `t ^x` is a term. Doing so, we have stronger induction principles which facilitates the development.

```
Inductive term : pterm -> Prop :=
  | term_var : forall x,
      term (pterm_fvar x)
  | term_abs : forall L t1,
      (forall x, x \notin L -> term (t1 ^ x)) ->
      term (pterm_abs t1)
  | term_app : forall t1 t2,
      term t1 ->
      term t2 ->
      term (pterm_app t1 t2)
  | term_sub : forall L t1 t2,
     (forall x, x \notin L -> term (t1 ^ x)) ->
      term t2 ->
      term (pterm_sub t1 t2).
```

We have now for each `trm_bvar i` in a term `t`, at least i binders above it, and thus the operation `open` is the identity on a term:

```
Lemma open_rec_term : forall t u,  term t -> forall k, t = {k ~> u}t.
```

We also need the predicate body on preterms, stating that if we open a preterm `t` (at level 0) with a fresh variable, we obtain a locally closed term:

```
Definition body t := exists L, forall x, x \notin L -> term (t ^ x).
```

For instance, `trm_bvar` 0 is a body since (`trm_bvar` 0)^`x` = `trm_fvar x` which is a term. In the same way, `trm_abs trm_bvar` 1 is also a body since (`trm_abs trm_bvar` 1)^`x` = {0 ~> `trm_fvar x`}(`trm_abs trm_bvar` 1) = `trm_abs` ({1 ~> `trm_fvar x`}`trm_bvar` 1) = `trm_abs trm_fvar x`. Of course, every term is also a body by Lemma `open_rec_term`.

6.2.4 Number of Occurrences of Variables

As the λj-calculus is based on the notion of multiplicity, we need to be able to count the number of free variables:

```
Fixpoint fv_occ (x: var) (t: pterm) : nat  :=
 match t  with
```

```
 | pterm_bvar k => 0
 | pterm_fvar z => if z = x then 1 else 0
 | pterm_abs t' => fv_occ x t'
 | pterm_app t1 t2 => fv_occ x t1 + fv_occ x t2
 | pterm_sub t1 t2 => fv_occ x t1 + fv_occ x t2
end.
```

We can now wonder if this notion is enough to express for instance the dereliction rule, stated in the original system as $t[x/u] \to_{\mathtt{w}} t\{x/u\}$ when $|t|_x = 1$. With our approach, this gives `t[s]` $\to$ `t ^^ s` with a condition now a little bit more complex. There are two options:

- The first one, which only uses the number of occurrences of free variables, needs to introduce a fresh variable x and is:

```
forall L, forall x \notin L, fv_occ x t = 1
```

- As it is not convenient to manipulate fresh variables, and furthermore not necessary, we count the number of bound variables. The condition simply becomes `bv_occ t` $= 1$.

The idea of `bv_occ` is to count the number of `trm_bvar` 0 (modulo the fact that they are under abstractions). For instance `bv_occ`(`trm_abs trm_bvar` 1) $= 1$, and `bv_occ`(`app` (`trm_bvar` 0)(`trm_bvar` 0)) $= 2$. Once again, we have to generalize this notion to any level. We thus have the general definition:

```
Fixpoint bv_occ_k (k : nat) (t : pterm) : nat :=
  match t with
  | pterm_bvar i    => if k = i then 1 else 0
  | pterm_fvar x    => 0
  | pterm_abs t1    => bv_occ_k (S k) t1
  | pterm_app t1 t2 => (bv_occ_k k t1) + (bv_occ_k k t2)
  | pterm_sub t1 t2 => (bv_occ_k (S k) t1) + (bv_occ_k k t2)
  end.
```

and the particular case (corresponding to `bv_occ`), when we consider the number of bound occurrences at the root of a term:

```
Definition bv_occ (t : pterm)  : nat := bv_occ_k 0 t.
```

6.2.5 The Non-deterministic Replacement

The non-deterministic replacement in the contraction rule

$$t[x/u] \to_{\mathtt{c}} t_{[y]_x}[x/u][y/u] \text{ with } |t|_x \geq 2$$

is defined in [AK10] as "we write $t_{[y]_x}$ for the non-deterministic replacement of i $(1 \leq i \leq n-1)$ occurrences of x in t by a *fresh* variable y".

This definition is not satisfying since there are several notions which need to be precisely defined from a mathematical point of view. We propose to first define a relation `ndr_n t x y t' n` which means that `t'` is obtained from `t` by renaming `n` occurrences of `x` in `y`. For practical reasons, the relation is defined on preterms.

```
Inductive ndr_n : pterm -> var -> var -> pterm -> nat -> Prop :=
  | ndr_bvar    : forall x y z,
                    ndr_n (pterm_bvar z) x y (pterm_bvar z) 0
  | ndr_varnone : forall x y z,
                    ndr_n (pterm_fvar z) x y (pterm_fvar z) 0
  | ndr_varsome : forall x y,
                    ndr_n (pterm_fvar x) x y (pterm_fvar y) 1
  | ndr_abs     : forall t t' x y  n, ndr_n t x y t' n ->
                    ndr_n (pterm_abs t) x y (pterm_abs t') n
  | ndr_app     : forall t t' u u' x y n n' n'', n'' = n + n' ->
                    ndr_n t x y u n -> ndr_n t' x y u' n' ->
                    ndr_n (pterm_app t t') x y (pterm_app u u') n''
  | ndr_sub     : forall t t' x y u u' n n' n'', n'' = n + n' ->
                    ndr_n (t) x y (t') n -> ndr_n u x y u' n' ->
                    ndr_n (t[u]) x y (t'[u']) n''.
```

Notice that the non-determinism is only apparent on free variables, and then propagated by the context.

We can now restrict ourselves to the case where there is the right number of replacements, that is at least one, and strictly less than the number of occurrences of the variable to be replaced:

```
Definition ndr t x y t' := exists n, (1 <= n) /\
                           (n < fv_occ x t) /\
                           ndr_n t x y t' n.
```

We can illustrate the notion of this non-deterministic replacement with what would be the replacement $(x\ x[z/x])_{x \rightsquigarrow y} = y\ x[z/y]$ in the original version of λj. In our case, we have to check:

```
ndr ((pterm_app (pterm_fvar x) (pterm_fvar x))[pterm_fvar x])
    x y
    ((pterm_app (pterm_fvar y) (pterm_fvar x))[pterm_fvar y])
```

This is true since it exists 2 such that:

The non-deterministic replacement preserves the well-formed terms:

```
Lemma ndr_preservation_term : forall t x y t',
                                term t -> ndr t x y t' -> term t'.
```

6.2.6 The Rules of λj

We first introduce rewriting definitions independently of λj, so our development is as generic as possible.

6.2.6.1 General Rewriting Rules

First of all we define general definitions for rewriting. Given a relation `Red` over preterms (of type `p_term -> p_term -> Prop`), we define how to build the contextual closure, and the reflexive and transitive closure:

```
Inductive contextual_closure Red : pterm -> pterm -> Prop :=
  | redex : forall t s, Red t s -> contextual_closure Red t s
```

```
  | beta_app_left : forall t t' u, term u ->
        contextual_closure Red t t' ->
        contextual_closure Red (pterm_app t u) (pterm_app t' u)
  | beta_app_right : forall t u u', term t ->
        contextual_closure Red u u' ->
        contextual_closure Red (pterm_app t u) (pterm_app t u')
  | beta_abs : forall t t' L, (forall x, x \notin L ->
        contextual_closure Red (t^x) (t'^x)) ->
        contextual_closure Red (pterm_abs t) (pterm_abs t')
  | beta_subst_left : forall t t' u L, term u ->
        (forall x, x \notin L -> contextual_closure Red (t^x) (t'^x))
        -> contextual_closure Red  (pterm_sub t u) (pterm_sub t' u)
  | beta_subst_right : forall t u u', body t ->
        contextual_closure Red u u' ->
        contextual_closure Red  (pterm_sub t u) (pterm_sub t u').

Inductive star_closure Red : pterm -> pterm -> Prop :=
  | reflexive_reduction : forall t, star_closure Red t t
  | one_step_reduction : forall t u, Red t u -> star_closure Red t u
  | transitive_reduction : forall t u v, Red t u ->
                star_closure Red u v -> star_closure Red t v.
```

6.2.6.2 Rules of λj

We redefine the rules of λj, using the locally nameless representation. Since our goal is to show Full Composition, it is enough to restrict ourselves to the rule B where the list of substitutions is empty i.e. $(\lambda x.t)\ u \to_{\mathtt{B}} t[x/u]$. As usually done, we split the rules of λj into B and the rules j.

```
Inductive B_red : pterm -> pterm -> Prop :=
 | rule_B : forall t s, body t -> term s ->
        B_red (pterm_app (pterm_abs t) s) (t[s]).
Notation "t ->B u" := (B_red t u).

Definition B_ctxt_red t u := (contextual_closure B_red) t u.
Notation "t -->B u" := (B_ctxt_red t u).
```

We then define all the rules of j, depending on the number of occurrences:

```
Inductive j_red : pterm -> pterm -> Prop :=
 | rule_w : forall t s, body t -> bv_occ t = 0 -> term s ->
                j_red (t[s]) t
 | rule_d: forall t s, body t-> term s -> bv_occ t = 1 ->
                j_red (t[s])  (t^^s)
 | rule_c : forall t s x y t', body t -> term s ->
                bv_occ t >= 2 ->
                x \notin fv t ->
                y \notin fv t ->
                ndr (t^x) x y t' ->
                j_red (t[s])  ((close ((close t' x)[s]) y) [s]).
Notation "t ->j u" := (j_red t u).

Definition j_ctxt_red t u := (contextual_closure j_red) t u.
Notation "t -->j u" := (j_ctxt_red t u).
```

As we have mentioned before, we use the notion of `bv_occ_0` instead of `fv_occ` in the dereliction rule.

Of course, we have preservation of well-formed terms:

```
Lemma preservation_terms_after_B : forall t t', term t -> t -->B t'
                                 -> term t'.
Lemma preservation_terms_after_j : forall t t', term t -> t -->j t'
                                 ->  term t'.
```

The contraction rule deserves some attention. First of all, it is important to notice that as we have chosen to consider the inductive definition `ndr_n t x y t'` with `x` a free variable in `t`, we need to open the body `t` with a fresh variable `x` to state the relation between `t^x` and `t'`.

6.2.6.3 Examples

We can illustrate the contraction rule with the reduction which is $(z\ \lambda a.z)[z/b] \to_{\mathtt{c}} (z'\ \lambda a.z)[z/b][z'/b]$ in the original system. We thus want to have a relation between the terms (`app` (`trm_bvar` 0)(`trm_abs` `trm_bvar` 1))[`trm_fvar` b] and (`app` (`trm_bvar` 1)(`trm_abs` `trm_bvar` 1))[`trm_fvar` b][`trm_fvar` b]. In the contraction rule , t is (`app` (`trm_bvar` 0)(`trm_abs` `trm_bvar` 1)). If we open it with x we thus have (`app` (`trm_fvar` `x`)(`trm_abs` `trm_fvar` `x`)) (don't forget to add one under a binder). Then we can show that the relation

```
ndr (pterm_app (pterm_fvar x) (pterm_abs pterm_fvar x))
        x y
    (pterm_app (pterm_fvar y) (pterm_abs pterm_fvar x))
```

holds. Indeed, there exists 1 such that $|t|_x > 1$ and

```
ndr_n (pterm_app (pterm_fvar x) (pterm_abs pterm_fvar x))
        x y
    (pterm_app (pterm_fvar y) (pterm_abs pterm_fvar x)) 1
```

which easily holds.

Now, if we close (`app` (`trm_fvar` `y`)(`trm_abs` `trm_fvar` `x`)) with x this gives (`app` (`trm_fvar` `y`)(`trm_abs` `trm_bvar` 1)). The last step is to close (`app` (`trm_fvar` `y`)(`trm_abs` `trm_bvar` 1))[`trm_fvar` b] with y which gives (`app` (`trm_bvar` 1)(`trm_abs` `trm_bvar` 1))[`trm_fvar` b][`trm_fvar` b] and we can thus conclude.

6.2.7 The Full Composition Proof

We first give the statement and the proof of Full Composition as stated in the original article:

Theorem 6.2.1 (Full Composition)**.** *Let t, u be terms. Then $t[x/u] \to_{\mathtt{j}}^* t\{x/u\}$*

Proof. By induction on the number of free occurrences of x in t.

- If $|t|_x = 0$, then $t[x/u] \to_{\mathtt{w}} t = t\{x/u\}$

- If $|t|_x = 1$, then $t[x/u] \to_{\mathtt{d}} t\{x/u\}$
- If $|t|_x \geq 2$, then

$$\begin{array}{ll} t[x/u] & \to_{\mathtt{c}} \\ t_{x\rightsquigarrow y}[y/u][x/u] & \to_{\mathtt{j}}^* \text{(i.h.)} \\ t_{x\rightsquigarrow y}\{y/u\}[x/u] & \to_{\mathtt{j}}^* \text{(i.h.)} \\ t_{x\rightsquigarrow y}\{y/u\}\{x/u\} & = t\{x/u\} \end{array}$$

□

The structure of the proof in our development will be the same. The statement of the Full Composition theorem is of course not the same, but still very simple, thanks to the opening operation which is the equivalent of the substitution operation in the original system on bound variables:

```
Theorem full_composition : forall t u,  term (t[u]) ->
                                        (t[u]) -->j* (t^^u).
```

As it is the most complex proof of our development, we will give a sketch of the proof. We will proceed by an induction on the number of bound variables at level 0 in `t`. Let n be this number.

- If $n = 0$, we have to show that `t^^`$u = t$ to apply the weakening rule. This is possible using Lemma `open_rec_term`, at the condition that t is a term. This is indeed the case since $n = 0$ (and that t is a body by hypothesis) as stated by the following lemma:

  ```
  Lemma bv_occ_body_term : forall t, body t -> bv_occ t = 0 ->
                           term t.
  ```

  Notice that this lemma is not true if `t` is not a body (take for instance `trm_bvar` 1).

- If $n = 1$, we can immediately apply the dereliction rule.
- If $n > 1$,

  We can now split the number n of bound variables at level 0 in t into two numbers $n1$ and $n2$. The number $n1$ will represent the number of variables replaced, and $n2$ those unchanged.

  Before showing properties about the non-deterministic replacement, we need to show that for every term `t`, there always exists a `t`$'$ which is its non-deterministic replacement:

  ```
  Lemma ndr_n_exists : forall t x y n, n <= fv_occ x t ->
          exists t', ndr_n t x y t' n.
  ```

  Thus if we pick-up two fresh variables `a` and `b`, we can obtain `t`$'$ s.t. the relation `ndr_n (t^a) a b t' n1` holds.

  We then go through several steps, using the transitive composition:

```
Lemma transitive_closure_composition : forall Red t u v,
                star_closure Red t u -> star_closure Red u v ->
                star_closure Red t v.
```

- t[u] → j (close (close t′ a[u]) b[u])
  In this subcase, we simply apply the contraction rule which easily holds.
- (close (close t′ a[u]) b[u]) → j (close (close t′ a[u]) b)^^u In this subcase, can apply the induction hypothesis which states:

  ```
  IHn1 : forall t : pterm, bv_occ t = n1 -> forall u : pterm,
                 term (t [u]) -> (t [u]) -->j* (t ^^ u)
  ```

  We thus have to show:

  ```
  bv_occ (close (close t' a [u]) b) = n1
  ```

  and

  ```
  term (close (close t' a [u]) b [u])
  ```

  Those two goals are automatically solved using tactics which saturate the context with all the informations which can be obtained using lemmas concerning free variables and the non-deterministic replacement such as:

  ```
  Lemma ndr_n_fv_occ : forall t t' x y n, ~(x = y) ->
                 ndr_n t x y t' n ->
                 fv_occ y t' = fv_occ y t + n.

  Lemma ndr_n_fv_occ_x : forall t t' x y n, ~(x = y) ->
                 ndr_n t x y t' n ->
                 fv_occ x t' = fv_occ x t - n.
  ```

  and lemmas relating the number of free variables and the number of bound variables such as

  ```
  Lemma bv_occ_close_decompose : forall t y,
          bv_occ t + fv_occ y t = bv_occ (close t y).
  ```

- (close (close t′ a[u]) b)^^u = close ([b ∼> u]t′) a[u]) This subcase is done using Lemma subst_as_close_open and some lemmas concerning the implicit substitution:

  ```
  Lemma subst_fresh : forall x t u,   x \notin fv t ->
                 [x ~> u] t = t.
  Lemma subst_close : forall t x y u,
      x \notin fv u -> y \notin fv u ->
      x <> y ->  [y ~> u] (close t x) = close ([y ~> u]t) x.
  ```

- close ([b ∼> u]t′) a[u]) → j (close ([b ∼> u]t′) a)^^u Once again, we apply an induction hypothesis dealing with n2 (the number of variables not replaced) and not n1 (the number of variables replaced). We thus have to show that:

  ```
  bv_occ (close ([b ~> u]t') a) = n2
  ```

and

```
term (close ([b ~> u]t') a [u])
```

The tactics which we have defined can handle those goals automatically, using the same principle of context saturation with the lemmas defined previously.

– (close ([b ∼> u]t′) a)^^u = ([a ∼> u]([b > u]t′)) This step is straightforward using Lemma subst_as_close_open.

– ([a ∼> u]([b > u]t′)) = t^^u This last step proceeds like that:

```
[a ~> u]([b ~> u]t')
```

is transformed into

```
[b ~> u]([a ~> u]t')
```

thanks to the following lemma:

```
Lemma fv_subst_commute : forall t x y u v, x <>y ->
        x \notin fv v ->
        y \notin fv u ->
        ([y~>v][x~>u]t) = ([x~>u][y~>v]t).
```

Then

```
t ^^ u
```

is transformed into

```
[a ~> u]t ^ a
```

thanks to the following lemma:

```
Lemma subst_intro : forall x t u,
  x \notin (fv t) ->
  t ^^ u = [x ~> u](t ^ x).
```

The last lemma allows us to conclude:

```
Lemma ndr_subst : forall t t' u x y n,
        x <>y ->
        y \notin fv t ->
        y \notin fv u ->
        ndr_n t x y t' n ->
        [x~>u]t = [y~>u]([x~>u]t').
```

Chapitre 7

Conclusion

Lors de cette thèse nous avons abordé différents aspects autour des substitutions et ressources explicites, ainsi que de la réécriture. Nous proposons de conclure et d'évoquer des perspectives de recherches en cours ou à venir selon quatre grand thèmes de la thèse.

Réécriture et logique Nous avons à travers le prisme des ressources établi un cadre homogène pour définir des λ-calculs capable de contrôler l'affaiblissement, la contraction, et la substitution linéaire. Le comportement opérationnel du prisme des ressources n'est pas seulement basé sur la propagation structurelle, mais également sur la décroissance de la multiplicité des variables affectées par les substitutions. Tous les calculs ont les propriétés de simulation de la β-réduction, de confluence, de PSN, ainsi que de normalisation forte des termes (simplement) typés. Diverses directions de recherche sont envisageables pour poursuivre le travail sur le prisme des ressources.

- La première est bien entendu la question de la métaconfluence pour l'ensemble des calculs du prisme. Avec une approche comme celle proposée au chapitre 4.2, qui consiste à spécifier les formes normales par rapport à la duplication des substitutions explicites, nous conjecturons qu'il soit possible d'obtenir une réponse positive. L'utilisation de cette méthode permettrait d'ailleurs de mieux la comprendre, pour en arriver à une simplification.

- On peut également considérer ce que deviendrait le prisme si l'on remplaçait $\lambda_{\mathtt{s}}$ par un autre calcul, tel $\lambda\mathtt{j}$. En particulier, il serait intéressant d'étudier les liens entre $\lambda\mathtt{j}$ avec affaiblissement et contraction explicite et le formalisme graphique de $\lambda\mathtt{j}$ [AG09] où contractions et affaiblissements sont par nature, déjà explicites. On pourrait ainsi espérer une simplification de l'ensemble du prisme. D'autre part, la démarche permettant d'atteindre l'équilibre entre les différents calculs, les simulations et les propriétés serait facilitée. En effet, le fait que cet équilibre fût si difficile à obtenir pour le prisme est du en grande partie au fait que la propagation était aussi structurelle, contrairement à $\lambda\mathtt{j}$ où la propagation est à distance par multiplicité.

- Une autre possibilité est de changer la logique ou bien encore sa présentation. On pourrait par exemple obtenir un prisme en ayant un calcul des séquents intuitionniste avec affaiblissement et contraction explicite $l\lambda^{\mathtt{Gtz}}$ [GILv09]

(également appelé $\lambda_{®}^{\mathtt{Gtz}}$ dans [GILL11], où un système avec types intersection est proposé). De la même manière, on pourrait utiliser le calcul classique ${}^{*}\mathcal{X}$ [Ž07] où affaiblissements et contractions sont explicites.

- Un exercice difficile serait de transformer le prisme des ressources en un cube. Rappelons que notre formalisme prend la forme d'un prisme car il comporte deux bases (selon le fait que la substitution soit explicite ou non) et que la projection ne permet pas de passer de l'une à l'autre. Le but serait d'avoir des opérations d'enrichissement et d'appauvrissement à partir de n'importe quel sommet du cube vers n'importe quel autre.

- Nous avons privilégié lors du développement du prisme mettre l'accent sur les relations d'enrichissement et d'appauvrissement entre tous les calculs. De ce fait, nous avons mis de côté un certain aspect de la logique car nous n'avons pas abordé la question de la projection des réductions de chaque calcul du prisme dans celles d'un formalisme graphique tel que les réseaux de preuves [Gir87], ou les $\lambda\mathtt{j}$-dags [AG09]. Bien que cet exercice soit facile à réaliser pour certains calculs, comme par exemple le λ-calcul ou le $\lambda\mathtt{csw}$-calcul, il l'est impossible pour d'autres, comme par exemple $\lambda\mathtt{s}$. Pour atteindre cet objectif, un moyen possible serait de transformer le prisme en cube (ainsi la projection dans un formalisme graphique passerait par le $\lambda\mathtt{csw}$-calcul) même si on peut également imaginer d'autres approches.

Réécriture

- Dans le chapitre 3 nous avons étudié la PSN pour le calcul $\lambda_{\mathtt{s}}$ grâce à une technique modulaire qui sépare les propriétés communes à tous les calculs et celles spécifiques à chacun d'entre eux. Dans un cadre plus général, on peut se poser la question de la PSN pour des calculs d'ordre supérieur (exprimés grâce au formalisme des ERS [Kha90] par exemple) avec substitutions explicites. En effet, il est possible pour tout système de réécriture d'ordre supérieur, où la substitution est implicite, de la rendre explicite. Dès lors, on doit ajouter un ensemble de règles qui vont propager les substitutions explicites par rapport à tous les constructeurs de ce calcul. Cette idée a déjà été étudiée [Blo97] mais la caractérisation des systèmes d'ordre supérieur pour lesquels PSN est vérifiée est très restrictive. La version plus évoluée pour montrer PSN avec la technique qui utilise les étiquettes [AK10] serait une bonne piste de départ pour obtenir PSN pour une plus large classe de systèmes.

- Du côté des substitutions explicites seules, de nombreuses possibilités existent. Pour continuer l'étude de la confluence axiomatique faite au chapitre 4, on pourrait l'étendre à la métaconfluence. Celle-ci ne pourrait pas facilement englober l'ensemble des calculs décrits dans cette thèse, car il faut caractériser la structure des formes normales. Or il apparaît qu'elles peuvent être fortement différentes selon le fait que l'on considère un calcul avec propagation selon

la structure des termes, la multiplicité, ou un mélange des deux. On pourrait dans un premier temps se concentrer sur les calculs plus traditionnels avec propagation selon la structure. Par la suite, nous pourrions généraliser le travail sur la métaconfluence de $\lambda\mathtt{j}$ réalisé au chapitre 4 qui illustre bien le comportement des calculs propageant les substitutions explicites à distance selon la multiplicité. Ainsi nous aurions une preuve axiomatique pour ce genre de calculs qui pourraient s'adapter à divers formalismes [Mil07, Ó Conchúir06].

- Un autre problème apparemment simple, mais qui résiste depuis longtemps, est de montrer la normalisation du calcul de propagation des substitutions de $\lambda\mathtt{ex}$ grâce à un argument donné simplement par une mesure arithmétique. La difficulté vient de la combinaison de plusieurs facteurs. Premièrement on propage des substitutions même si elle sont inutiles (dans le sens où elles ne lient pas de variable). De ce fait, on ne peut pas se reposer sur une certaine décroissance du nombre de variables affectées par une substitution. Deuxièmement, il faut définir des mesures qui soient stables par rapport à l'équation permutant les substitutions indépendantes. Dernièrement, on duplique des substitutions. Ce comportement est similaire à beaucoup de calculs, mais dans le cas de $\lambda\mathtt{ex}$ cela implique de parfois dupliquer une substitution pour en créer une autre inutile (dans la règle de composition notamment).

- Malgré l'abondance d'articles traitant des substitutions explicites dans la littérature, il existe néanmoins encore plusieurs aspects non explorés.

  La standardisation [CF58] fait partie des théorèmes classiques et bien connus en λ-calcul mais qui n'a pas son penchant du côté des calculs avec substitutions explicites. Il existe des approches [Mel05] mais celles-ci reposent principalement sur la standardisation même du λ-calcul. Une direction intéressante serait de s'affranchir de toute notion du λ-calcul afin de donner des réductions standards qui seraient plus spécifiques aux calculs avec substitutions explicites. Le $\lambda\mathtt{j}$-calcul, par sa propagation à distance, semble un bon candidat. La réduction linéaire de tête [DR03] pourrait permettre de définir une telle notion de standardisation, qui fonctionnerait pour un calcul avec substitution explicite, mais pas pour le λ-calcul.

  De la même manière, les stratégies normalisante définies pour les calculs avec substitutions explicites (comme celle du chapitre 6.1 pour $\lambda\mathtt{x}^{-}$) reproduisent ce qui est fait en λ-calcul, c'est à dire de toujours contracter le redex externe le plus à gauche. On pourrait de la même manière que pour la standardisation, essayer de définir une approche plus spécifique aux calculs avec substitutions explicites. Une fois encore, il semble que la réduction linéaire de tête pourrait être à la base d'une telle stratégie normalisante.

Complexité Dans le chapitre 4.2 nous avons étudié la complexité du $\lambda\mathtt{x}^{-}$-calcul, plus près des implémentations que le λ-calcul. Nous avons ainsi conjecturé l'écart

séparant la complexité de ces deux calculs. Ici encore, de nombreuse directions de recherche sont possibles.

- La première est bien évidemment de montrer cette conjecture. Pour cela, plusieurs voies sont possibles. La première consiste à détourner le problème en trouvant un autre calcul qui se prête mieux à l'exercice que $\lambda\mathtt{x}^-$. Des essais non concluant ont été fait avec $\lambda\mathtt{j}$. Une autre possibilité serait de conserver $\lambda\mathtt{x}^-$ mais d'ajouter une règle donnant à la substitution explicite le même statut que le Cut. Cette règle permettrait de crée directement une substitution explicite sans effectuer la propagation maximale. Il faudrait par la suite bien évidemment éliminer cette règle, comme on l'a fait avec la règle Cut.

- Nous n'avons pas encore parlé de la duplication et de son coût. La façon la plus naïve de l'implémenter dans la règle $(t\ v)[x/u] \to_{SApp} t[x/u]\ v[x/u]$, est de réellement dupliquer le terme u et avoir deux occurrences de celui-ci en mémoire. Cette opération étant en $\mathrm{O}(|u|)$ cela ne changerait pas tellement la borne déjà énorme, la passant au carré approximativement. Au lieu de dupliquer naïvement, on pourrait aller dans l'autre direction et considérer le partage, voire même le partage optimal [Lév78]. Bien que cela étant bien étudié pour le λ-calcul (à la fois d'un point de vue typé et non typé), et même du point de vue de la complexité [ACM04], il n'existe actuellement pas de travail pour les substitutions explicites.

- Comme nous sommes concernés par les descriptions d'implémentations, nous pourrions nous restreindre au cas faible, celui où il n'y a pas de réduction sous les abstractions, ainsi qu'au cas où il n'y a pas de variables libres. Nous pourrions également combiner cette restriction avec le partage, comme c'est fait pour le λ-calcul [BLM05] ou certains calculs avec substitutions explicites [BLR96]. Il serait ainsi intéressant de comparer les résultats obtenus avec les bornes extraites des machines à environnement utilisées dans les langages tel OCaml.

- D'un point de vue plus théorique, il serait intéressant de connaitre les bornes d'un calcul avec composition totale comme $\lambda\mathtt{ex}$. Une autre direction pourrait être de généraliser les résultats à n'importe quel calcul. Par exemple, étant donné un langage et sa stratégie maximale, la notion d'arbre binaire associé pourrait être obtenu. Les lemmes menant de la notion d'arbre binaire à la borne maximale semblent également généralisables. De ce fait, on pourrait obtenir une procédure qui, étant donné un calcul et une stratégie maximale, donnerait une borne maximale de réduction.

- Un autre exercice serait de s'intéresser à la complexité implicite. Plutôt que de chercher la borne de tel ou tel calcul, il est intéressant de se fixer une classe de complexité réaliste (polynomiale ou simplement exponentiel) et de chercher alors à pouvoir exprimer uniquement des programmes ayant cette complexité.

On peut y arriver en restreignant la grammaire des termes, ou bien le système de types.

Formalisation Dans le chapitre 6.2 nous avons développé une formalisation de $\lambda\mathtt{j}$ ainsi que montré la propriété de composition totale.

- Une première direction de recherche conduite actuellement est de simplifier l'opération de remplacement non-déterministe en l'effectuant directement sur les variables liées, et non sur les variables libres. Ainsi, non seulement la règle de contraction se simplifie, mais également toute la bureaucratie nécessaire pour faire le lien entre le nombre de variables libres et celui de variables liées lorsque l'on fait un renommage non-déterministe.

- Dans un second temps, il semble intéressant de généraliser la règle `B` qui déclenche le calcul afin de pouvoir considérer une liste de substitutions entre l'abstraction et l'argument, c'est à dire $(\lambda x.t)\mathtt{L}\ u \to t[x/u]\mathtt{L}$ où `L` est une liste, comme défini dans [AK10]. Bien que cela ne soit pas nécessaire pour obtenir la propriété de composition totale, il faut considérer cette généralisation pour capturer les $\lambda\mathtt{j}$-dags [AG09] et l'ensemble de leurs propriétés. Les notions de super-développement [vR93], ou encore de `XL`-développement [AK10] ont notamment besoin de la règle `B` à distance.

- Il est intéressant de noter que la certification du calcul prend une forme pyramidale, la base étant constituée de nombreux lemmes techniques rendus nécessaires par le besoin de spécifier en détail le comportement de toute opération par rapport à toute autre. Par dessus, se trouvent de nombreux lemmes un peu plus haut niveau, notamment à propos du remplacement non-déterministe. Et finalement tout au sommet la preuve de composition totale. De ce fait, il serait maintenant beaucoup plus rapide de montrer d'autres propriétés telles que la confluence et la PSN (car la preuve de cette dernière propriété se fait simplement par induction, contrairement aux preuves de PSN pour la plupart des calculs avec substitutions explicites dits structurels). La suite du développement serait également facilitée par l'usage des tactiques déjà disponibles permettant de savoir si un terme est bien formé.

Bibliography

[ABR00] Ariel Arbiser, Eduardo Bonelli, and Alejandro Ríos. Perpetuality in a lambda calculus with explicit substitutions and composition. Workshop Argentino de Informática Teórica (WAIT), JAIIO, 2000.

[ABW07] Brian E. Aydemir, Aaron Bohannon, and Stephanie Weirich. Nominal reasoning techniques in coq (extended abstract). *Electr. Notes Theor. Comput. Sci.*, 174(5):69–77, 2007.

[ACCL90] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. In *ACM Symposium on Principles of Programming Languages (POPL)*, pages 31–46. ACM, 1990.

[ACM04] Andrea Asperti, Paolo Coppola, and Simone Martini. (optimal) duplication is not elementary recursive. *Inf. Comput.*, 193(1):21–56, 2004.

[ACP+08] Brian E. Aydemir, Arthur Charguéraud, Benjamin C. Pierce, Randy Pollack, and Stephanie Weirich. Engineering formal metatheory. In *ACM Symposium on Principles of Programming Languages (POPL)*, pages 3–15. ACM, 2008.

[AG09] Beniamino Accattoli and Stefano Guerrini. Jumping boxes. In *Computer Science Logic (CSL)*, volume 5771 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 2009.

[AJ05] Klaus Aehlig and Felix Joachimski. Continuous normalization for the lambda-calculus and Gödel's T. *Annals of Pure and Applied Logic*, 133, 2005.

[AK10] Beniamino Accattoli and Delia Kesner. The structural lambda-calculus. In *Proceedings of the European Association for Computer Science Logic (CSL)*, Lecture Notes in Computer Science. Springer-Verlag, 2010.

[AW10] Brian Aydemir and Stephanie Weirich. Lngen: Tool support for locally nameless representations. Technical Report MS-CIS-10-24, Computer and Information Science, University of Pennsylvania, 2010.

[Bar84] Henk Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984. Revised Edition.

[Bar92] Henk Barendregt. Lambda calculus with types. In Samson Abramsky, Dov Gabbay, and Thomas Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 117–309. Oxford University Press, 1992.

[BBLRD96] Zine-El-Abidine Benaissa, Daniel Briaud, Pierre Lescanne, and Jocelyne Rouyer-Degli. $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 6(5):699–722, 1996.

[Bec01] Arnold Beckmann. Exact bounds for lengths of reductions in typed lambda-calculus. *Journal of Symbolic Logic*, 66(3):1277–1285, 2001.

[Big] Bigloo. `http://www-sop.inria.fr/indes/fp/Bigloo`.

[BK82] Jan Bergstra and Jan-Willem Klop. Strong normalization and perpetual reductions in the lambda calculus. *Information Processing and Cybernetics*, 18:403–417, 1982.

[BLM05] Tomasz Blanc, Jean-Jacques Lévy, and Luc Maranget. Sharing in the weak lambda-calculus. In Middeldorp et al. [MvOvRdV05], pages 70–87.

[Blo97] Roel Bloo. *Preservation of Termination for Explicit Substitution*. PhD thesis, Eindhoven University of Technology, 1997.

[BLR96] Zine-El-Abidine Benaissa, Pierre Lescanne, and Kristoffer Rose. Modeling sharing and recursion for weak reduction strategies using explicit substitution. In *Programming Languages: Implementations, Logics, and Programs (PLILP)*, volume 1140 of *Lecture Notes in Computer Science*, pages 393–407. Springer, 1996.

[BR95] Roel Bloo and Kristoffer Rose. Preservation of strong normalization in named lambda calculi with explicit substitution and garbage collection. In *Computer Science in the Netherlands (CSN)*, pages 62–72, 1995.

[Bro] Institute Brouwer. The automath archive. All publications concerning the projet are available on `http://www.win.tue.nl/automath`.

[BT07] Clark Barrett and Cesare Tinelli. CVC3. In *Proceedings of the International Conference on Computer Aided Verification (CAV)*, volume 4590 of *Lecture Notes in Computer Science*, pages 298–302. Springer-Verlag, 2007.

[BW00] Arnold Beckmann and Andreas Weiermann. Analyzing Gödel's T via expanded head reduction trees. *Mathematical Logic Quarterly*, 46(4):517–536, 2000.

[CCK06] Sylvain Conchon, Evelyne Contejean, and Johannes Kanig. Ergo: a theorem prover for polymorphic first-order logic modulo theories. http://ergo.lri.fr/papers/ergo.ps, 2006.

[CDCV81] M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Functional characters of solvable terms. *Mathematical Logic Quarterly*, 27:45–58, 1981.

[CF58] Haskell Curry and Robert Feys. *Combinatory Logic*, volume 1. North-Holland, 1958.

[Cha11] Arthur Charguéraud. The locally nameless representation. *Journal of Automated Reasoning*, 2011. To appear.

[CHL92] Pierre-Louis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Confluence properties of weak and strong calculi of explicit substitutions. Technical Report 1617, INRIA-Rocquencourt, 1992.

[CHL96] Pierre-Louis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *Journal of the ACM*, 43(2):362–397, March 1996.

[Chu33] Alonzo Church. A set of postulates for the foundation of logic (second paper). *Annals of Mathematics, Series 2*, 34:839–364, 1933.

[Chu36] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(2):56–68, 1940.

[Chu41] Alonzo Church. *The Calculi of Lambda Conversion*. Princeton University Press, 1941.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms, second edition, 2001.

[Com] CompCert. The compcert project. All publications concerning the projet are available on http://compcert.inria.fr/.

[Coq] The Coq Proof Assistant. http://coq.inria.fr/.

[Cré07] Pierre Crégut. Strongly reducing variants of the krivine abstract machine. *Higher-Order and Symbolic Computation*, 20(3):209–230, 2007.

[Cur88] Pierre-Louis Curien. The $\lambda\rho$-calculi: an abstract framework for closures, 1988. unpublished (preliminary version printed as LIENS report).

[Dav] René David. A short proof of the strong normalization of the simply typed lambda calculus. Available as http://www.lama.univ-savoie.fr/~david/.

[dB72] Nicolaas G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae*, 5(35):381–392, 1972.

[dB78] Nicolaas G. de Bruijn. A namefree lambda calculus with facilities for internal definition of expressions and segments. Technical Report 78-WSK-03, Eindhoven University of Technology, 1978.

[dB87] Nicolaas G. de Bruijn. Generalizing automath by means of a lambda-typed lambda calculus. In Edgar G.K. Lopez-Escobar David W. Kueker and Carl H. Smith, editors, *Mathematical Logic and Theoretical Computer Science*, number 106 in Lecture Notes in Pure and Applied Mathematics, page 71–92. Marcel Dekker, 1987.

[DCKP00] Roberto Di Cosmo, Delia Kesner, and Emmanuel Polonovski. Proof nets and explicit substitutions. In *Foundations of Software Science and Computation Structures (FOSSACS)*, volume 1784 of *Lecture Notes in Computer Science*, pages 63–81. Springer-Verlag, 2000.

[DG01] René David and Bruno Guillaume. A λ-calculus with explicit weakening and explicit substitution. *Mathematical Structures in Computer Science*, 11:169–206, 2001.

[DHKP96] Gilles Dowek, Thérèse Hardin, Claude Kirchner, and Frank Pfenning. Higher-order unification via explicit substitutions: the case of higher-order patterns. In *Joint International Conference and Symposium on Logic Programming*, pages 259–273, 1996.

[dM03] Jolie G. de Miranda. Normalization bounds in the simply typed lambda calculus, 2003. Private communication.

[DR03] Vincent Danos and Laurent Regnier. How abstract machines implement head linear reduction. Submitted, 2003.

[dV85] Roel C. de Vrijer. A direct proof of the finite developments theorem. *Journal of Symbolic Logic*, 50(2):339–343, 1985.

[dV87] Roel C. de Vrijer. Exactly estimating functionals and strong normalization. *Indagationes Mathematicae*, 1987.

[FMS05] Maribel Fernández, Ian Mackie, and François-Régis Sinot. Lambda-calculus with director strings. *Applicable Algebra in Engineering, Communication and Computing*, 15(6):393–437, 2005.

[Gac08] Andrew Gacek. The abella interactive theorem prover (system description). In *International Conference on Automated Reasoning (IJCAR)*, number 5195 in Lecture Notes in Computer Science. Springer-Verlag, 2008.

[GILL11] Silvia Ghilezan, Jelena Ivetic, Pierre Lescanne, and Silvia Likavec. Intersection types for the resource control lambda calculi. In *International Colloquium of Theoretical Aspects of Computing (ICTAC)*, volume 6916 of *Lecture Notes in Computer Science*, pages 116–134. Springer, 2011.

[GILv09] Silvia Ghilezan, Jelena Ivetic, Pierre Lescanne, and Dragiša Žunić. Intuitionistic sequent-style calculus with explicit structural rules. In *International Tbilisi Symposium on Logic, Language, and Computation (TbiLLC)*, volume 6618 of *Lecture Notes in Computer Science*, pages 101–124. Springer, 2009.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50, 1987.

[GL98] Jean Goubault-Larrecq. A proof of weak termination of typed lambda sigma-calculi. In Thorsten Altenkirch, Wolfgang Naraschewski, and Bernhard Reus, editors, *Proceedings of the International Workshop Types for Proofs and Programs*, volume 1512 of *Lecture Notes in Computer Science*, pages 134–151. Springer-Verlag, December 1998.

[Gor93] Andrew D. Gordon. A mechanisation of name-carrying syntax up to alpha-conversion. In Jeffrey J. Joyce and Carl-Johan H. Seger, editors, *Higher Order Logic Theorem Proving and its Applications*, volume 780 of *Lecture Notes in Computer Science*, pages 413–425. Springer, 1993.

[GP02] Murdoch Gabbay and Andrew M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13(3-5):341–363, 2002.

[Gue11] Nicolas Guenot. Nested proof search as reduction in the λ-calculus. In *International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*. ACM, 2011.

[Har89] Thérèse Hardin. Confluence results for the strong pure categorical logic CCL; λ-calculi as subsystems of CCL. *Theoretical Computer Science*, 65, 1989.

[Has] *The Haskell language*. http://www.haskell.org/.

[Her94] Hugo Herbelin. A λ-calculus structure isomorphic to sequent calculus structure. In *Annual Conference of the European Association for Computer Science Logic (CSL)*, volume 933 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[HL89] Thérèse Hardin and Jean-Jacques Lévy. A confluent calculus of substitutions. In *France-Japan Artificial Intelligence and Computer Science Symposium*, 1989.

[How80] William Alvin Howard. Ordinal analysis of terms of finite type. *Journal of Symbolic Logic*, 45(3):493–504, 1980.

[Hue76] Gérad Huet. *Résolution d'équations dans les langages d'ordre* $1, 2, \ldots, \omega$. Thèse de doctorat d'état, Université Paris VII, 1976.

[Isa] The Isabelle theorem prover. http://isabelle.in.tum.de/.

[Kes96] Delia Kesner. Confluence properties of extensional and non-extensional λ-calculi with explicit substitutions. In *Procedings of the International Conference on Rewriting Techniques and Applications (RTA)*, volume 1103 of *Lecture Notes in Computer Science*, pages 184–199. Springer-Verlag, 1996.

[Kes07] Delia Kesner. The theory of explicit substitutions revisited. In *Proceedings of the EACSL Annual Conference on Computer Science and Logic (CSL)*, volume 4646 of *Lecture Notes in Computer Science*, pages 238–252. Springer-Verlag, 2007.

[Kes09] Delia Kesner. A theory of explicit substitutions with safe and full composition. *Logical Methods in Computer Science*, pages 1–29, 2009.

[Kha90] Zurab Khasidashvili. Expression reduction systems. In *Proceedings of IN Vekua Institute of Applied Mathematics*, volume 36, Tbilisi, 1990.

[KL05] Delia Kesner and Stéphane Lengrand. Extending the explicit substitution paradigm. In *Proceedings of the International Conference on Rewriting Techniques and Applications (RTA)*, volume 3467 of *Lecture Notes in Computer Science*, pages 407–422. Springer-Verlag, 2005.

[KL07] Delia Kesner and Stéphane Lengrand. Resource operators for lambda-calculus. *Information and Computation*, 205(4):419–473, 2007.

[Kle36] Stephen C. Kleene. λ-definability and recursiveness. *Duke Math Journal*, pages 340–353, 1936.

[KLN05] Fairouz Kamareddine, Twan Laan, and Rob Nederpelt. Pure type systems with parameters and definitions. In *A Modern Perspective on Type Theory*, volume 29 of *Applied Logic Series*, pages 255–310. Springer Netherlands, 2005.

[KR95] Fairouz Kamareddine and Alejandro Ríos. The confluence of the λs_e-calculus via a generalized interpretation method, 1995. Draft.

[KR97] Fairouz Kamareddine and Alejandro Ríos. Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *Journal of Functional Programming*, 7(4):395–420, 1997.

[KR09] Delia Kesner and Fabien Renaud. The prismoid of resources. In *Mathematical Foundations of Computer Science (MFCS)*, volume 5734 of *Lecture Notes in Computer Science*, pages 464–476. Springer, 2009.

[KR11] Delia Kesner and Fabien Renaud. A prismoid framework for languages with resources. *Theoretical Computer Science*, In Press, Accepted Manuscript, 2011.

[Les94a] Pierre Lescanne. De bruijn's c-lambda-xi-phi calculus of explicit substitutions revisited. Available at `http://perso.ens-lyon.fr/pierre.lescanne/publications.html#classic`, 1994.

[Les94b] Pierre Lescanne. From λ_σ to λ_υ, a journey through calculi of explicit substitutions. In *ACM Symposium on Principles of Programming Languages (POPL)*, pages 60–69. ACM, 1994.

[Lév78] Jean-Jacques Lévy. *Réductions correctes et optimales dans le lambda-calcul*. PhD thesis, Université Paris VII, France, 1978.

[LIS] Common LISP. `http://common-lisp.net`.

[LM99] Jean-Jacques Lévy and Luc Maranget. Explicit substitutions and programming languages. In *Foundations of Software Technology and Theoretical Computer Science*, volume 1738 of *Lecture Notes in Computer Science*, pages 181–200. Springer-Verlag, 1999.

[Loa98] Ralph Loader. Notes on simply typed lambda calculus, 1998.

[LRD95] Pierre Lescanne and Jocelyne Rouyer-Degli. Explicit substitutions with de Bruijn levels. In Jieh Hsiang, editor, *6th International Conference on Rewriting Techniques and Applications (RTA)*, volume 914 of *Lecture Notes in Computer Science*, pages 294–308. Springer-Verlag, April 1995.

[Mel95] Paul-André Melliès. Typed λ-calculi with explicit substitutions may not terminate. In *Proceedings of the International Conference of Typed Lambda Calculus and Applications (TLCA)*, volume 902 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[Mel05] Paul-André Melliès. Axiomatic rewriting theory I: A diagrammatic standardization theorem. In Middeldorp et al. [MvOvRdV05], pages 554–638.

[Mil78] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Science*, 17(3):348–375, 1978.

[Mil07] Robin Milner. Local bigraphs and confluence: Two conjectures: (extended abstract). *Electronic Notes in Theoretical Computer Science*, 175(3):65–73, 2007.

[MP93] James McKinna and Robert Pollack. Pure type systems formalized. In *International Conference on Typed Lambda Calculi and Applications (TLCA)*, volume 664 of *Lecture Notes in Computer Science*, pages 289–305. Springer, 1993.

[MRZ10] Ariel Mendelzon, Alejandro Ríos, and Beta Ziliani. Swapping: a natural bridge between named and indexed explicit substitution calculi. In *Proceedings of the International Workshop on Higher-Order Rewriting (HOR)*, volume 49 of *EPTCS*, pages 1–15, 2010.

[MvOvRdV05] Aart Middeldorp, Vincent van Oostrom, Femke van Raamsdonk, and Roel C. de Vrijer, editors. *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion of His 60th Birthday*, volume 3838 of *Lecture Notes in Computer Science*. Springer, 2005.

[Nad02] Gopalan Nadathur. The suspension notation for lambda terms and its use in metalanguage implementations. *Electronic Notes in Theoretical Computer Science*, 67:35–48, 2002.

[Ned92] Robert. P. Nederpelt. The fine-structure of lambda calculus. Technical Report Computing Science Notes 92/07, Eindhoven University of Technology, Department of Mathematics and Computer Science, 1992.

[Oca] The Objective Caml language. `http://caml.inria.fr/`.

[Ó Conchúir06] Shane Ó Conchúir. Proving PSN by simulating non-local substitutions with local substitution. In *Proceedings of the Workshop on Higher-Order Rewriting (HOR)*, pages 37–42, August 2006. Proc. available as `http://hor.pps.jussieu.fr/06/proc/proc.html`.

[RBL09] Kristoffer Rose, Roel Bloo, and Frédéric Lang. On explicit substitution with names, 2009. IBM Research Report.

[Reg91] Laurent Regnier. *λ-calcul et réseaux*. PhD thesis, Université de Paris VII, 1991. Thèse de doctorat de mathématiques.

[Ren] Fabien Renaud. Metaconfluence of λj: dealing with non-deterministic replacements. Submitted.

[Ros92] Kristoffer Rose. Explicit cyclic substitutions. In *Proceedings of the International Workshop on Conditional Term Rewriting Systems (CTRS)*, volume 656 of *Lecture Notes in Computer Science*, pages 36–50. Springer-Verlag, 1992.

[Ros96] K.H. Rose. *Operational Reduction Models for Functional Programming Languages*. PhD thesis, University of Copenhagen, 1996.

[Sch82] Helmut Schwichtenberg. Complexity of normalization in the pure typed lambda calculus. In A. S. Troelstra and D. Van Dalen, editors, *The L. E. J. Brouwer Centenary Symposium*, Amsterdam, 1982. North-Holland.

[Sch91] Helmult Schwichtenberg. An upper bound for reduction sequences in the typed lambda-calculus. *Archive for Mathematical Logic*, 30(5):405–408, 1991.

[Sch01] Aleksy Schubert. The complexity of beta-reduction in low orders. In *Proceedings of the International Conference of Typed Lambda Calculus and Applications (TLCA)*, volume 2044 of *Lecture Notes in Computer Science*, pages 400–414. Springer-Verlag, 2001.

[SML] SML/NJ. Standard ML of new jersey. http://www.smlnj.org.

[SNO+10] Peter Sewell, Francesco Zappa Nardelli, Scott Owens, Gilles Peskine, Thomas Ridge, Susmit Sarkar, and Rok Strnisa. Ott: Effective tool support for the working semanticist. *J. Funct. Program.*, 20(1):71–122, 2010.

[SP94] Paula Severi and Erik Poll. Pure type systems with definitions. In *Logical Foundations of Computer Science*, volume 813 of *Lecture Notes in Computer Science*, pages 316–328. Springer-Verlag, 1994.

[Ter03] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[Tur37] Alan M. Turing. Computability and lambda-definability. *Journal of Symbolic Logic*, 2(4):153–163, 1937.

[vO01] Vincent van Oostrom. Net-calculus. Notes available on http://www.phil.uu.nl/~oostrom/oudonderwijs/cmitt/03-04/net.ps, 2001.

[vO07] Vincent van Oostrom. Random descent. In *International Conference on Rewriting Techniques and Applications (RTA)*, volume 4533 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 2007.

[vO08a] Vincent van Oostrom. Confluence by decreasing diagrams. In *International Conference on Rewriting Techniques and Applications (RTA)*, volume 5117 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2008.

[vO08b] Vincent van Oostrom. The z property. Slides available on http://www.phil.uu.nl/~oostrom/publication/rewriting.html, 2008.

[vR93] Femke van Raamsdonk. Confluence and superdevelopments. In *International Conference on Rewriting Techniques and Applications (RTA)*, volume 690 of *Lecture Notes in Computer Science*, pages 168–182. Springer-Verlag, 1993.

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

[vRSSX99] Femke van Raamsdonk, Paula Severi, Morten Heine Sorensen, and Hongwei Xi. Perpetual reductions in λ-calculus. *Information and Computation*, 149(2), 1999.

[Ž07] Dragiša Žunić. *Computing with Sequents and Diagrams in Classical Logic - Calculi* $^*\mathcal{X}$, $^d\mathcal{X}$ *and* $^{©}\mathcal{X}$. PhD thesis, ENS Lyon, 2007.

[Xi99] Hongwei Xi. Upper bounds for standardizations and an application. *Journal of Symbolic Logic*, 64(1):291–303, 1999.

[YH90] Hirofumi Yokouchi and Teruo Hikita. A rewriting system for categorical combinators with multiple arguments. *SIAM Journal on Computing*, 19(1):78–97, 1990.

Les ressources explicites vues par la théorie de la réécriture

Cette thèse s'articule autour de la gestion de ressources explicites dans les langages fonctionnels, en mettant l'accent sur des propriétés de calculs avec substitutions explicites raffinant le λ-calcul. Dans une première partie, on s'intéresse à la propriété de préservation de la β-normalisation forte (PSN) pour le calcul $\lambda_{\mathtt{s}}$. Dans une seconde partie, on étudie la propriété de confluence pour un large ensemble de calculs avec substitutions explicites. Après avoir donné une preuve générique de confluence basée sur une série d'axiomes qu'un calcul doit satisfaire, on se focalise sur la métaconfluence de $\lambda\mathtt{j}$, un calcul où le mécanisme de propagation des substitutions utilise la notion de multiplicité, au lieu de celle de structure. Dans la troisième partie de la thèse on définit un prisme des ressources qui généralise de manière paramétrique le λ-calcul dans le sens où non seulement la substitution peut être explicite, mais également la contraction et l'affaiblissement. Cela donne un ensemble de huit calculs répartis sur les sommets du prisme pour lesquels on prouve de manière uniforme plusieurs propriétés de bon comportement comme par exemple la simulation de la β-réduction, la PSN, la confluence, et la normalisation forte pour les termes typés. Dans la dernière partie de la thèse on montre différentes ouvertures vers des domaines plus pratiques. On s'intéresse à la complexité d'un calcul avec substitutions en premier lieu. On présente des outils de recherche et on conjecture des bornes maximales. Enfin, on finit en donnant une spécification formelle du calcul $\lambda\mathtt{j}$ dans l'assistant à la preuve Coq.
Mots clés : Substitutions explicites, ressources, métaconfluence, complexité, formalisation

---

Explicit resources from the rewriting theory point of view

This thesis deals with the management of explicit resources in functional languages, stressing on properties of calculi with explicit substitutions refining the λ-calculus. In the first part, we are concerned with the preservation property of β-strong normalisation (PSN) for the $\lambda_{\mathtt{s}}$-calculus, a language among the eight calculi of the prismoid of resources defined thereafter. In the second part, we study the confluence property for a large set of calculi with explicit substitutions. After having given a generic proof of confluence based on a series of axioms that a calculus must fulfill to verify this property, we focalise on the metaconfluence of $\lambda\mathtt{j}$, a calculus where the propagation mechanism of substitutions uses the notion of multiplicity, whereas the traditional way is the structural propagation. In the third part, we define a prismoid of resources which generalise in a parametric way the λ-calculus in the sense that not only the substitution can be explicit, but also the contraction and the weakening. This gives a set of eight calculi spread over the vertices of the prismoid for which we prove in a uniform way several properties of good behavior as the simulation of β-reduction, PSN, confluence, and strong normalisation for typed terms. In the last part of the thesis we show different opening up to more practical domains. First, we are concerned with the complexity of a calculus with substitutions. We present research tools and conjecture on maximal bounds for reductions of the $\lambda\mathtt{x}^{-}$-calculus . Finally, we give a formal specification of the $\lambda\mathtt{j}$-calculus within the proof assistant Coq.
Keywords: Explicit substitutions, resources, metaconfluence, complexity, formalisation