Etude des propriétés du calcul de réécriture: du rho calcul au rhoEpsilon calcul

Germain Faure

Etude des propriétés du calcul de réécriture : du $\rho_{\emptyset}$-calcul au $\rho_{\varepsilon \mathrm{V}}$-calcul

Magistère Informatique et Modélisation
ENS Lyon

Germain Faure

Abstract

Le ρ-calcul intégre dans un cadre uniforme et simple la réécriture du premier ordre et le λ-calcul tout en permettant d'exprimer le non-déterminisme. Nous introduisons ici un nouveau calcul dans lequel l'échec est distingué de l'ensemble vide et dans lequel le test de l'échec est possible. Si nous munissons ce calcul de l'appel par valeur, nous obtenons un calcul confluent et dans lequel le `first` est exprimable (et par conséquent les stratégies d'évaluation le sont aussi). Nous conclurons sur un bref aperçu des perspectives et des questions restant ouvertes.

Abstract

The ρ-calculus integrates in a uniform and simple setting first-order rewriting, λ-calculus and non-deterministic computations. In the calculus here introduced, the empty set is not an overloading term i.e. failure is distinguished from the empty set. Moreover, failure can here be checked. If the call-by-value evaluation mechanism is added to the calculus, a confluent calculus in which the `first` can be expressed is obtained (and consequently evaluation strategies can be expressed too). Finally, we open up new horizons and we conclude with a short overview of open questions.

Introduction

Si je devais exprimer en quelques mots et de manière totalement informelle le travail que j'ai effectué durant mon stage je crois que je citerais Freud : "de quelle manière que l'on s'y prenne, on s'y prend toujours mal" en rajoutant que l'essentiel étant d'arriver à un compromis satisfaisant. Mon stage m'a en effet amené à construire une extension du $\rho_{\emptyset}$-calcul. Pendant toute cette construction, nous avons donc dû combiner différents choix sémantiques, syntaxiques... Pour être plus précis, mon stage s'est déroulé principalement en quatres parties :

- prise de connaissance de l'état actuel de la recherche dans le ρ-calcul ;
- construction d'un nouveau calcul : le ρ_{ε}-calcul ;
- preuve de confluence du ρ_{ε}-calcul ;
- expression du `first` dans le ρ_{ε}-calcul.

La structure du rapport sera donc naturellement la suivante : après une brève présentation du laboratoire et des objectifs de stage, nous exposerons dans le premier chapitre les différents ρ-calculs introduits (base de mon stage). Dans un deuxième chapitre, nous motiverons et expliciterons la construction du ρ_{ε}-calcul. Puis nous nous intéresserons à sa confluence pour enfin dans un troisième chapitre examiner l'exprimabilité du `first` dans un tel calcul. Nous conclurons sur les questions laissées ouvertes et les perspectives de notre travail.

Présentation du stage

Présentation du projet PROTHEO

Le projet PROTHEO est un projet du LORIA et de l'INRIA Lorraine. J'ai effectué mon stage dans cette équipe du LORIA (Laboratoire Lorrain de Recherche en Informatique et Automatisme) à Nancy. L'objectif de cette équipe est la conception et la réalisation d'outils pour la spécification et la vérification de logiciels. Le travail de l'équipe peut se résumer principalement en trois points :

- réalisation d'un environnement permettant de prototyper de tels outils ;
- réalisation des démonstrateurs spécialisés sur les preuves par récurrence ou dans certaines théories équationnelles ;
- mise en oeuvre de techniques de preuves spécifiques privilégiant l'utilisation des contraintes et des règles de réécriture.

Pour ma part, je prenais place dans l'équipe Protheo, suite aux travaux effectués par Horatiu Cirstea et Claude Kirchner. Je fus encadré principalement par ce dernier et dans la dernière partie de mon stage par Luigi Liquori.

Présentation des objectifs de stage

En début de stage et suite aux travaux déjà effectués sur le ρ-calcul, trois questions m'étaient proposées :

1. le `first` est-il exprimable dans le ρ-calcul?
2. quel lien peut-on faire entre le ρ-calcul et le ρ_{MP}-calcul?
3. peut-on construire à l'image de l'isomorphisme de Curry-Howard (λ-calcul et déduction naturelle) une logique correspondant au ρ-calcul?

Il va de soi que nous n'espérions pas répondre à toutes ces questions dans la durée du stage tout en sachant que la troisième représente un travail conséquent et de longue haleine. Au fur et à mesure de l'avancée du stage, celui-ci a pris une orientation légérement différente. En réfléchissant à l'exprimabilité du `first` dans le ρ-calcul, il nous a semblé constructif de voir quels pourraient être les ingrédients nécessaires à cette exprimabilité. C'est ce qui nous a amené à construire le ρ_ε-calcul, d'en étudier les propriétés et enfin d'examiner l'exprimabilité du `first` dans ce calcul.

Chapter 1

Présentation des différents ρ-calculs

Afin de déterminer la base de notre travail, nous présentons dans ce chapitre le ρ_T-calcul, le $\rho_\emptyset$-calcul , le $\rho_T^{1^{st}}$-calcul, le ρ_{MP}-calcul introduits dans [Cir00, CK01a, CK01b, CKL01].

1.1 Présentation et syntaxe générales du ρ-calcul

Le ρ-calcul a été introduit à l'origine pour que tous les ingrédients de la réécriture soient des objets explicites. La notion principale est donc celle de termes de réécriture et on introduit une notion très générale qui est celle de règles de réécriture. Les règles, l'application de règles, les résultats deviennent donc des concepts explicites. Insistons sur le fait que dans le ρ-calcul les termes, les règles, les applications de règles et par conséquent les stratégies d'applications de ces règles sont traitées au niveau objet.

Dans le ρ-calcul, on peut explicitement représenter l'application d'une règle de réécriture (par exemple $a \to b$) à un terme (par exemple a) comme un objet ($[a \to b](a)$ qui s'évalue en $\{b\}$). Donc le symbole d'application d'une règle $@$ (où @ note la place des arguments de l'opérateur) fait partie de la syntaxe du ρ-calcul. Mais l'application d'une règle peut échouer comme par exemple $[a \to b](c)$ qui s'évalue en l'ensemble vide $\emptyset$ ou peut se réduire en un ensemble ayant plus d'un élément comme nous le verrons par la suite. Bien sûr les variables peuvent être utilisées dans les règles de réécriture (exemple: $[f(x) \to x](f(a))$ qui par le mécanisme d'évaluation se réduit en $\{a\}$). En effet, lorsque l'on évalue l'expression, la variable x de l'exemple est instanciée en a par le mécanisme classique de filtrage, de la même manière que dans la réécriture ([KK99]). De même, $@ \to @$ fait partie de la syntaxe du calcul. C'est un abstracteur puissant dont la relation avec la λ-abstraction du λ-calcul ([Bar84]) est assez intuitive : une λ-abstraction $\lambda x.t$ peut être représentée par la règle de réécriture $x \to t$. Ainsi le β-redex $(\lambda x.t\ u)$ n'est rien d'autre que le ρ-terme $[x \to t](u)$. De même, le λ-calcul avec motifs ([PJ87]) s'exprime facilement dans le ρ-calcul . Notons que lorsque l'on construit les abstractions dans un premier temps, il n'y a aucune restriction comme nous le verrons dans les exemples suivants.

Exemple 1.1

- $[f(x) \to g(x,y)](f(a))$ *s'évalue en* $g(a,y)$ *en laissant* y *libre.*
- $[x \to (f(y) \to g(x,y))](a)$ *notons que la variable* x *est libre dans* $f(y) \to g(x,y)$ *mais liée dans l'expression initiale. Ainsi un objet peut se réécrire en une règle de réécriture .*
- $[a \to a](a)$ *s'évalue en* $\{a\}$ *et ne crée donc pas de nouveau redex.*

Donc les objets du ρ-calcul sont construits sur une signature, un ensemble de variable, un opérateur d'abstraction $@ \to @$, un opérateur d'application $@$ et on peut considérer des ensembles de tels objets. On peut donc ainsi exprimer le non-déterminisme par l'intermédiaire d'ensembles de résultats. Par exemple, si l'on suppose que le $+$ est commutatif, l'application de la règle $x + y \to x$ au terme $a + b$ nous donne $\{a, b\}$. On définit donc les termes du ρ-calcul par :

Définition 1.1 *Etant donnés un ensemble de variables $\mathcal{X}$ et un ensemble de symboles $\mathcal{F} = \bigcup_{i\geq 0} \mathcal{F}_i$ tel que pour tout m, $\mathcal{F}_m$ est le sous-ensemble de symboles d'arité m. Nous supposons que chaque symbole a une arité unique, c'est-'a-dire que les $\mathcal{F}_m$ sont disjoints.*

L'ensemble de ρ-termes de base, noté $\varrho(\mathcal{F}, \mathcal{X})$, est le plus petit ensemble tel que :

- *les variables de $\mathcal{X}$ sont des ρ-termes,*
- *si $t_1, \ldots, t_n$ sont des ρ-termes et $f \in \mathcal{F}_n$ alors $f(t_1, \ldots, t_n)$ est un ρ-terme,*
- *si $t_1, \ldots, t_n$ sont des ρ-termes alors $\{t_1, \ldots, t_n\}$ est un ρ-terme,*
- *si t et u sont des ρ-termes alors $[t](u)$ est un ρ-terme,*
- *si t et u sont des ρ-termes alors $t \to u$ est un ρ-terme.*

On appelle position fonctionnelle *d'un ρ-terme t, toute position p du terme tel que $t(p) \in \mathcal{F}$. Les sous-termes $t_1, \ldots, t_n$ d'un terme fonctionnel $t = f(t_1, \ldots, t_n)$ sont appelés les* arguments *de t ou, par abus de langage, les arguments de f. Les symboles de $\mathcal{F}_0$ sont appelés* constantes.

En notation BNF les ρ-termes peuvent être définis par :

$$\textbf{ρ-termes} \qquad t \quad ::= \quad x \;\mid\; f(t, \ldots, t) \;\mid\; \{t, \ldots, t\} \;\mid\; [t](t) \;\mid\; t \to t$$

Exemple 1.2 *On considère $\mathcal{F}_0 = \{a, b, c\}$, $\mathcal{F}_1 = \{f, g\}$, $\mathcal{F} = \mathcal{F}_0 \cup \mathcal{F}_1$ et x,y des variables de $\mathcal{X}$. Quelques exemples de ρ-termes :*

- *$[a \to b](a)$ dénote l'application de la règle de réécriture $a \to b$ au terme a. Nous verrons que ce terme s'évalue en $\{b\}$.*
- *$[a \to a](a)$ dénote l'application de la règle de réécriture $a \to a$ au terme a. Ce terme s'évalue en $\{a\}$.*
- *$[f(x, y) \to g(x, y)](f(a, b))$ une application classique d'une règle de réécriture .*
- *$[x \to x + y](a)$ une règle de réécriture avec une variable libre y et nous verrons plus tard pourquoi ce terme se réécrit en $\{a + y\}$*
- *$[y \to [x \to x + y](b)]([x \to x](a))$ un ρ-terme correspondant au λ-terme $(\lambda y.((\lambda x.x + y)b))\ ((\lambda x.x\ a))$. Dans la règle de réécriture $x \to x + y$ la variable y est libre mais dans la règle de réécriture $y \to [x \to x + y](b)$, elle est liée.*
- *$[x \to [x](x)](x \to [x](x))$ représentant le λ-terme bien connu $(\omega\omega)$.*
- *$[[(x \to x + 1) \to (1 \to x)](a \to a + 1)](1)$ un ρ-terme plus compliqué sans correspondance avec une règle de réécriture standard ou un λ-terme.*

1.2 La définition du ρ_T-calcul

En utilisant les notions présentées dans les Annexes A, B (filtrage et règles d'évaluation) et dans les sections précédentes nous donnons la définition générale du ρ_T-calcul .

Définition 1.2 *Etant donnés un ensemble de symboles de fonctions $\mathcal{F}$, un ensemble de variables $\mathcal{X}$ et une théorie T sur les termes de $\varrho(\mathcal{F}, \mathcal{X})$ avec un problème de filtrage décidable, nous appelons ρ_T-calcul (ou calcul de réécriture) un calcul défini par :*

- *un sous-ensemble non vide $\varrho\_(\mathcal{F}, \mathcal{X})$ de l'ensemble de termes $\varrho(\mathcal{F}, \mathcal{X})$,*
- *l'application (d'ordre supérieur) de substitution aux termes*

- *une théorie T,*
- *l'ensemble de règles d'évaluation $\mathcal{E}$: Fire, Congruence, Congruence_fail, Distrib, Batch, Switch$_L$, Switch$_R$, OpOnSet, Flat (Figure 1.1),*
- *une stratégie d'évaluation $\mathcal{S}$ qui guide l'application des règles d'évaluation.*

Fire	$[l \to r](t)$	$\Longrightarrow$	$r\langle\langle Solution(l \ll^?_T t)\rangle\rangle$
Congruence	$[f(u_1,\ldots,u_n)](f(v_1,\ldots,v_n))$	$\Longrightarrow$	$\{f([u_1](v_1),\ldots,[u_n](v_n))\}$
Congruence_fail	$[f(u_1,\ldots,u_n)](g(v_1,\ldots,v_m))$	$\Longrightarrow$	$\emptyset$
Distrib	$[\{u_1,\ldots,u_n\}](v)$	$\Longrightarrow$	$\{[u_1](v),\ldots,[u_n](v)\}$
Batch	$[v](\{u_1,\ldots,u_n\})$	$\Longrightarrow$	$\{[v](u_1),\ldots,[v](u_n)\}$
Switch$_L$	$\{u_1,\ldots,u_n\} \to v$	$\Longrightarrow$	$\{u_1 \to v,\ldots,u_n \to v\}$
Switch$_R$	$u \to \{v_1,\ldots,v_n\}$	$\Longrightarrow$	$\{u \to v_1,\ldots,u \to v_n\}$
OpOnSet	$f(v_1,\ldots,\{u_1,\ldots,u_m\},\ldots,v_n)$	$\Longrightarrow$	$\{f(v_1,\ldots,u_1,\ldots,v_n),\ldots,f(v_1,\ldots,u_m,\ldots,v_n)\}$
Flat	$\{u_1,\ldots,\{v_1,\ldots,v_n\},\ldots,u_m\}$	$\Longrightarrow$	$\{u_1,\ldots,v_1,\ldots,v_n,\ldots,u_m\}$

Figure 1.1: Les règles d'évaluation du ρ_T-calcul

1.3 Définition du $\rho_\emptyset$-calcul

On s'intéresse ici à une instance du ρ_T-calcul : le $\rho_\emptyset$-calcul .

Définition 1.3 *Etant donné un ensemble de symboles de fonctions $\mathcal{F}$, un ensemble de variables $\mathcal{X}$, nous appelons $\rho_\emptyset$-calcul un calcul défini par :*

- *un sous-ensemble $\varrho_\emptyset(\mathcal{F},\mathcal{X})$ de l'ensemble de termes $\varrho(\mathcal{F},\mathcal{X})$ tel que toutes les règles de réécriture soient de la forme $l \to r$ avec $l \in \mathcal{T}_\mathcal{F}(\mathcal{X})$,*
- *l'application (d'ordre supérieur) de substitution aux termes,*
- *la théorie $\emptyset$ (filtrage syntaxique),*
- *les règles d'évaluation Fire, Congruence, Congruence_fail, Distrib, Batch, Switch$_L$, Switch$_R$, OpOnSet, Flat,*
- *une stratégie d'évaluation $\mathcal{S}$ qui guide l'application des règles d'évaluation.*

Le $\rho_\emptyset$-calcul par abus est appelé le ρ-calcul . Cette restriction du ρ_T-calcul est intéressante principalement pour trois raisons :

1. elle encode le λ-calcul [Bar84];
2. sa confluence est établie [Cir00, CK01a];

3. elle donne une bonne sémantique au langage ELAN (pour une présentation du langage ELAN comme langage de systèmes de déduction voir [Vit94, KKV95, BKK+96], pour une présentation du ρ-calcul comme sémantique au langage ELAN voir [Cir00, CK01b]).

1.4 Le `first`

1.4.1 Intérêt du `first`

Comme nous l'avons signalé auparavant, dans le ρ-calcul, les termes, les règles, les applications de règles sont traités au niveau objet et par conséquent nous souhaiterions de même pouvoir exprimer les stratégies d'application de ces règles au même niveau. L'application d'une règle de réécriture de la réécriture standard est décrite explicitement dans le ρ-calcul par un ρ-terme approprié. Par exemple, l'application de la règle de réécriture $a \to b$ au terme a est représentée par le ρ-terme $[a \to b](a)$. La réduction de cette application en le terme b dans la réécriture correspond à la réduction du ρ-terme $[a \to b](a)$ en le ρ-terme $\{b\}$ dans le ρ-calcul. Mais une réduction en réécriture peut impliquer l'application de plusieurs règles et puisque nous voulons représenter dans le ρ-calcul toute réduction de la réécriture et pas seulement les réductions nécessitant un seul pas de réécriture alors nous voulons répondre à la question : *étant donnée une théorie de réécriture $\mathcal{R}$ existe-il un ρ-terme $\xi_{\mathcal{R}}$ tel que pour tout terme u, si u se réduit en le terme v dans la théorie de réécriture $\mathcal{R}$ alors $[\xi_{\mathcal{R}}](u)$ se ρ-réduit en un ensemble contenant le ρ-terme v ?*

Nous voulons donner une méthode pour construire le terme $\xi_{\mathcal{R}}(u)$ sans connaître la dérivation de u en v dans la théorie $\mathcal{R}$. Ce problème s'avère plus difficile parce que les propriétés du système de réécriture, comme la terminaison et la confluence, ne sont pas connues *a priori*. La définition de stratégies (de normalisation) est faite en général au méta-niveau et nous voulons montrer que le ρ-calcul est assez puissant pour nous permettre la représentation de telles dérivations au niveau objet du calcul. Ce que nous apportons ici, grâce à la puissance du filtrage et à l'utilisation du non-déterminisme, est la facilité d'exprimer en utilisant les ρ-termes, des stratégies (de normalisation) guidant l'application d'une ou plusieurs règles de réécriture. Nous pouvons ainsi exprimer les stratégies dans un formalisme uniforme combinant les mécanismes standards de réécriture et les techniques d'ordre supérieur. Pour calculer la forme normale d'un terme u par rapport à un système de réécriture $\mathcal{R}$, les règles de réécriture de $\mathcal{R}$ sont appliquées *répétitivement* à une position *quelconque* de u jusqu'à ce qu'aucune règle de $\mathcal{R}$ ne soit plus *applicable*. Par conséquent, les ingrédients nécessaires pour définir une telle stratégie sont :

- un opérateur d'itération qui applique *répétitivement* des règles de réécriture,
- un opérateur de parcours de termes qui applique une règle de réécriture à une position *quelconque* d'un terme,
- un opérateur testant si un ensemble de règles de réécriture est *applicable* sur un terme.

Ce que nous voulons par exemple c'est arriver à construire un ρ-terme qui réalise la normalisation *innermost* par rapport à un ensemble de règles de réécriture $\mathcal{R}$. Pour pouvoir exprimer ce terme nous avons besoin d'exprimer :

1. l'opérateur $repeat*$ décrivant l'application d'un terme tant que le résultat de l'application n'est pas $\emptyset$;
2. l'opérateur $Once_{bu}$ décrivant l'application d'un terme une fois et de manière bottom-up.

Avec ces deux opérateurs on peut exprimer $im(\mathcal{R})$: $im(\mathcal{R}) = repeat*(Once_{bu}(\mathcal{R}))$. Nous pouvons donc représenter la normalisation d'un terme u dans une théorie de réécriture par le ρ-terme : $\xi_{\mathcal{R}}(u) = [im(\mathcal{R})](u)$. Que ce soit pour l'opérateur $Once_{bu}$ ou pour l'opérateur $repeat*$ il apparaît clairement la nécessité de pouvoir tester l'échec dans l'application d'une règle. Nous définissons ainsi un opérateur qui cherche les termes qui ne mènent pas à un échec.

1.4.2 Définition du `first`

Nous introduisons un nouvel opérateur n-aire appelé $first$ qui a pour rôle de sélectionner parmi ses arguments le premier terme dont l'application à un ρ-terme donné n'est pas réduite en $\emptyset$. Nous voulons donc que l'application d'un ρ-terme $first(s_1, \ldots, s_n)$ à un terme t retourne le résultat de la première application sans échec d'un de ses arguments au terme t. Si la réduction de tout terme $[s_i](t)$, $i = 1, \ldots, k-1$, mène à $\emptyset$ et que le terme $[s_k](t)$ ne se réduit pas en $\emptyset$, alors $[first(s_1, \ldots, s_n)](t)$ est réduit en le même terme que le terme $[s_k](t)$. Ainsi, lorsque la réduction de tous les termes $[s_i](t)$, $i = 1, \ldots, k-1$ termine et mène à $\emptyset$ et que la réduction de $[s_k](t)$ ne termine pas alors la réduction du terme $[first(s_1, \ldots, s_n)](t)$ ne termine pas. Nous pouvons résumer la description de l'opérateur $first$ donnée ci-dessus par les deux règles d'évaluation $First'$ et $First''$:

$$
\begin{array}{lll}
First' \quad [first(s_1, \ldots, s_n)](t) & \Longrightarrow & \{u_k \downarrow\} \\
 & & \text{si } [s_i](t) \xrightarrow{*}_\rho \emptyset,\ i = 1, \ldots, k-1 \\
 & & \quad [s_k](t) \xrightarrow{*}_\rho u_k \downarrow \neq \emptyset,\ u_k \downarrow \ clos,\ sans\ radical \\
First'' \quad [first(s_1, \ldots, s_n)](t) & \Longrightarrow & \emptyset \\
 & & \text{si } [s_i](t) \xrightarrow{*}_\rho \emptyset,\ i = 1, \ldots, n
\end{array}
$$

Dans les deux règles précédentes la condition qu'un terme soit réduit en $\emptyset$ ou en un terme clos en forme normale ($u_k \downarrow$) est testée au méta-niveau du calcul. En procédant de cette manière, des opérations intrinsèques au niveau objet doivent être exécutées au méta-niveau et donc, les réductions au niveau objet ne contiennent plus toute l'information sous-jacente. Pour rendre explicite le fonctionnement de l'opérateur $first$ nous introduisons un nouvel opérateur n-aire "$\langle\rangle$" qui, intuitivement, sélectionne les termes non vides. Contrairement à un terme de la forme $\{t_1, \ldots, t_n\}$ où la virgule est supposée respecter les axiomes des ensembles (associativité, commutativité, idempotence), nous ne faisons aucune supposition sur un terme $\langle t_1, \ldots, t_n \rangle$ où l'ordre des arguments est essentiel dans l'évaluation.

1.4.3 Le ρ_T^{1st}-calcul

Nous allons maintenant définir le ρ_T^{1st}-calcul comme une extension du ρ_T-calcul que l'on enrichit avec l'opérateur `first`. On définit l'ensemble des ρ^{1st}-termes.

Définition 1.4 *L'ensemble des ρ^{1st}-termes étend l'ensemble $\varrho(\mathcal{F}, \mathcal{X})$ de ρ-termes de base (Définition 1.1) comme étant le plus petit ensemble tel que :*

- *les éléments de $\mathcal{T}_\mathcal{F}(\mathcal{X})$ sont des ρ^{1st}-termes,*
- *si $t_1, \ldots, t_n$ sont des ρ^{1st}-termes et $f \in \mathcal{F}_n$ alors $f(t_1, \ldots, t_n)$ est un ρ^{1st}-terme,*
- *si $t_1, \ldots, t_n$ sont des ρ^{1st}-termes alors $\{t_1, \ldots, t_n\}$ est un ρ^{1st}-terme,*
- *si t et u sont des ρ^{1st}-termes alors $[t](u)$ est un ρ^{1st}-terme,*
- *si t et u sont des ρ^{1st}-termes alors $t \to u$ est un ρ^{1st}-terme.*
- *si $t_1, \ldots, t_n$ sont des ρ^{1st}-termes alors $first(t_1, \ldots, t_n)$ est un ρ^{1st}-terme,*
- *si $t_1, \ldots, t_n$ sont des ρ^{1st}-termes alors $\langle t_1, \ldots, t_n \rangle$ est un ρ^{1st}-terme.*

Les règles d'évaluation décrivant l'opérateur $first$ et l'opérateur auxiliaire $\langle\rangle$ sont présentées dans la Figure 1.2. La description des deux opérateurs utilise une condition qui teste si un terme est (évalué en) un ensemble vide.

Nous étendons maintenant la Définition 1.2 du ρ_T-calcul en considérant les nouveaux opérateurs et les règles d'évaluation correspondantes.

Définition 1.5 *Etant donnés un ensemble de symboles de fonctions $\mathcal{F}$, un ensemble de variables $\mathcal{X}$, une théorie T sur les termes du $\varrho^{1st}(\mathcal{F}, \mathcal{X})$ avec un problème de filtrage décidable, nous appelons ρ_T^{1st}-calcul un calcul défini par :*

- *un sous-ensemble non vide $\varrho_-^{1st}(\mathcal{F}, \mathcal{X})$ de l'ensemble de termes $\varrho^{1st}(\mathcal{F}, \mathcal{X})$,*

$First$	$[first(s_1,\ldots,s_n)](t)$	$\Longrightarrow$	$\langle [s_1](t),\ldots,[s_n](t)\rangle$
$First\_fail$	$\langle \emptyset, t_1,\ldots,t_n\rangle$	$\Longrightarrow$	$\langle t_1,\ldots,t_n\rangle$
$First\_success$	$\langle t, t_1,\ldots,t_n\rangle$	$\Longrightarrow$	$\{t\}$ si t *ne contient pas de radical et de variable libre et n'est pas* $\emptyset$
$First\_single$	$\langle\rangle$	$\Longrightarrow$	$\{\}$

Figure 1.2: Les règles d'évaluation de l'opérateur $first$

- *l'application (d'ordre supérieur) de substitution aux termes,*
- *une théorie T,*
- *l'ensemble de règles d'évaluation : $Fire$, $Congruence$, $Congruence\_fail$, $Distrib$, $Batch$, $Switch_L$, $Switch_R$, $OpOnSet$, $Flat$, $First$, $First\_fail$, $First\_success$ et $First\_single$,*
- *une stratégie d'évaluation $\mathcal{S}$ qui guide l'application des règles d'évaluation.*

On voudrait voir maintenant s'il est possible d'exprimer une stratégie de normalisation dans le $\rho_T^{1^{st}}$-calcul.

1.4.4 Stratégies de normalisation

Nous avons besoin comme d'habitude de :

- l'application d'un terme à une profondeur donnée, à la position la plus profonde ne menant pas à un échec, à la position la moins profonde ne menant pas à un échec;
- la répétition de l'application d'une règle.

Pour exprimer des opérations récursives, il nous sera nécessaire de pouvoir exprimer un combinateur de point fixe.

Expression du combinateur de point fixe : le combinateur de point fixe de Turing

Nous avons choisi comme dans [Cir00] d'utiliser le point classique du λ-calcul aussi appelé point fixe de Turing ([Tur37]) qui est défini dans le λ-calcul par :
$\Theta_\lambda = (A_\lambda\ A_\lambda)$ où

$$A_\lambda = \lambda xy.y(xxy)$$

De manière tout à fait triviale, on obtient son expression dans ρ-calcul : $\Theta = [A](A)$ avec

$$A = x \to (y \to [y]([[x](x)](y))).$$

Bien évidemment, cela peut donner lieu à des réductions infinies. Il convient donc de pouvoir maîtriser ces réductions par des stratégies comme expliqué dans [Cir00].

Opérateurs de congruence générique

Nous pouvons définir grâce aux règles de *Congruence*, l'application d'un terme r à tous les sous-termes d'un terme $t = f(t_1,\ldots,t_n)$. En effet, on a bien :

$$[f(r,\ldots,r)](f(t_1,\ldots,t_n)) \overset{Congruence}{\longrightarrow} \{f([r](t_1),\ldots,[r](t_n))\}$$

Mais ce qu'on voudrait c'est deux opérateurs génériques Φ et Ψ qui vérifient les énoncés de la Figure 1.3.

$$Traverse\_seq \quad [\Phi(r)](f(u_1,\ldots,u_n)) \implies \langle\{f([r](u_1),\ldots,u_n)\},\ldots,\{f(u_1,\ldots,[r](u_n))\}\rangle$$

$$Traverse\_par \quad [\Psi(r)](f(u_1,\ldots,u_n)) \implies \{f([r](u_1),\ldots,[r](u_n))\}$$

Figure 1.3: Congruence générique

La question que l'on se pose est de savoir si l'on peut exprimer dans le $\rho_T^{1^{st}}$-calcul ces deux opérateurs. On se donne une signature finie $\mathcal{F}$ et on note par $\mathcal{F}_0 = \{c_1,\ldots,c_n\}$ l'ensemble de symboles de fonctions constantes et par $\mathcal{F}_+ = \{f_1,\ldots,f_m\}$ l'ensemble de symboles de fonctions non constantes.

Lemme 1.1 *Les applications des termes $\Phi(r)$ et $\Psi(r)$ à un terme $t = f_k(t_1,\ldots,t_n)$ peuvent être décrites dans le $\rho_T^{1^{st}}$-calcul.*

Preuve : les deux opérateurs $\Phi(r)$ et $\Psi(r)$ peuvent être exprimés par :

$$\Phi'(r) \triangleq first(f_1(r,id,\ldots,id),\ldots,f_1(id,\ldots,id,r),\ldots,f_m(r,id,\ldots,id),\ldots,f_m(id,\ldots,id,r))$$

$$\Psi(r) \triangleq \{c_1,\ldots,c_n,f_1(r,\ldots,r),\ldots,f_m(r,\ldots,r)\}$$

$$avec\ c_i \in \mathcal{F}_0,\ i=1,\ldots,n,\ et\ f_j \in \mathcal{F}_+,\ j=1,\ldots,m$$

L'opérateur Φ' ne correspond donc pas exactement à la description de l'opérateur Φ donnée dans la Figure 1.3 mais le résultat obtenu en appliquant les termes $\Phi(r)$ et $\Phi'(r)$ à un terme $f_k(u_1,\ldots,u_p)$ est le même.

La preuve est présentée dans [Cir00]. □

BottomUp

Lemme 1.2 *L'opérateur BottomUp décrivant l'application d'un terme r à tous les sous-termes d'un autre terme t d'une manière bottomp-up est exprimable dans $\rho_T^{1^{st}}$-calcul.*

Preuve : Il suffit de poser :

$$G_{bu}(r) \triangleq f \to (x \to [first(\Psi(f),id);first(r,id)](x))\ et\ BottomUp(r) \triangleq [\Theta](G_{bu}(r))$$

□

bottom-up

Lemme 1.3 *L'opérateur $Once_{bu}$ décrivant l'application d'un terme r une fois d'une manière bottom-up est exprimable dans le $\rho_T^{1^{st}}$-calcul.*

Preuve : Il suffit de poser :

$$H_{bu}(r) \triangleq f \to (x \to [first(\Phi(f),r)](x))\ et\ Once_{bu}(r) \triangleq [\Theta](H_{bu}(r)).$$

□

Répétition

Lemme 1.4 *L'opérateur repeat* décrivant l'application répétée d'un terme tant que le résultat de l'application n'est pas une erreur, est exprimable dans le $\rho_T^{1^{st}}$-calcul.*

Preuve : Il suffit de poser :

$$J(r) \triangleq f \to (x \to [first(r; f,\ id)](x)) \text{ et } repeat*(r) \triangleq [\Theta](J(r))$$

□

Opérateur de normalisation

On peut, comme mentionné ci-dessus, définir $im(\mathcal{R})$ par : $im(\mathcal{R}) = repeat*(Once_{bu}(\mathcal{R}))$. On définit de même l'opérateur de normalisation *innermost* d'un terme u dans une théorie de réécriture $\mathcal{R}$ par le ρ-terme : $\xi_{\mathcal{R}}(u) = [im(\mathcal{R})](u)$. Il va de soi que l'on pourrait définir de manière similaire l'opérateur $Once_{td}$ décrivant l'application d'un terme une fois et de manière *top-down* et donc l'opérateur de normalisation *outermost*.

1.5 Le ρ_{MP}-calcul

Nous abordons maintenant un calcul un peu différent appelé *Rho Calcul* ou *ρ_{MP}-calcul* et introduit dans [CKL01]. Même si l'on veut toujours faire des ingrédients de la réécriture des concepts explicites, on aborde un calcul à la fois plus simple et plus "libre". Les ensembles ne sont pas comme dans les versions antérieures du ρ-calcul "built-in" mais les axiomes vérifiés pas la structure des résultats est maintenant un paramètre du calcul. La syntaxe des termes du Rho Calcul est définie par :

$$\begin{array}{rcll} t & ::= & a \mid X \mid t \to t \mid t \bullet t & \textit{plain terms} \\ & & \textit{null} \mid t, t & \textit{structured terms} \end{array}$$

On remarque que comme dans les calculs précédents on dispose d'un opérateur d'abstraction $@ \to @$ et d'un symbole d'application noté $@ \bullet @$ au lieu de $@$. Insistons sur le fait qu'ici la virgule ne vérifie aucun axiome et que la constante *null* représente la structure vide. Dans la suite, nous noterons $t(t_1 \dots t_n) \triangleq t \bullet t_1 \dots \bullet t_n$ et $(t_i)^{i=1\dots n} \triangleq t_1, \dots, t_n$.

Exemple 1.3 *Nous donnons quelques exemples de termes du Rho Calcul :*

- *le terme $(f(X) \to X) \bullet f(a)$ représente l'application de la règle de réécriture $f(X) \to X$ appliquée au terme $f(a)$;*
- *le terme $X \to Y \to X$ correspond au λ-terme $\lambda(X)\lambda(Y)X$;*
- *le terme $(1, 2, 3)$ dénote tout simplement une structure ternaire ;*
- *le terme $(cx \to 0, cy \to 0)$ dénote l'enregistrement à deux champs, cx et cy.*

Un paramètre important du ρ_{MP}-calcul est la théorie de filtrage $\mathbb{T}$. Nous allons définir les problèmes de filtrage mais auparavant nous allons donner quelques exemples de théories.

Exemple 1.4 • *La théorie vide $\mathbb{T}_\emptyset$ qui définit l'égalité à α-conversion près est définie par les règles d'inférence suivantes :*

$$\frac{t_1 = t_2,\ t_2 = t_3}{t_1 = t_2}\ (Trans) \qquad \frac{t_1 = t_2}{t_2 = t_1}\ (Sym)$$

$$\frac{t_1 = t_2}{t_3[t_1]_p = t_3[t_2]_p}\ (Ctx) \qquad \frac{\mathcal{X} \subseteq \mathcal{V}}{t = \alpha_{\mathcal{X}}(t)}\ (Refl_\alpha)$$

où $t_2[t_1]_p$ dénote le terme t_1 avec le terme t_2 à la position p.

- *Pour un symbole binaire f on définit $\mathbb{T}_{Set(f,nil)}$ par $\mathbb{T}_\emptyset$ plus les règles d'inférence suivantes :*

$$\frac{}{f(t_1\ t_2) = f(t_2\ t_1)}\ (Comm) \qquad \frac{}{f(f(t_1\ t_2)t_3) = f(t_1\ f(t_2\ t_3))}\ (Assoc) \qquad \frac{}{f(t\ t) = t}\ (Idem)$$

$$\frac{}{f(t\ nil) = t}\ (Zero_R) \qquad \frac{}{f(nil\ t) = t}\ (Zero_L)$$

Un cas important est quand la "," et "null", opérateurs du calcul, vérifient les axiomes des ensembles ; dans ce cas nous notons cette théorie $\mathbb{T}_{Set}$.

Nous allons maintenant définir quelques notions relatives au filtrage.

Définition 1.6 *Pour une théorie donnée $\mathbb{T}$ sur les termes du Rho Calcul :*

1. *une $\mathbb{T}$-matching équation est une formule de la forme $t_1 \ll_{\mathbb{T}} t_2$;*
2. *une substitution σ est une solution de la $\mathbb{T}$-matching équation $t_1 \ll_{\mathbb{T}} t_2$ si $\sigma t_1 =_{\mathbb{T}} t_2$;*
3. *un $\mathbb{T}$-matching système est une conjonction de $\mathbb{T}$-matching équations ;*
4. *une substitution σ est une solution d'un $\mathbb{T}$-matching système si c'est une solution de toutes les $\mathbb{T}$-matching équations ;*
5. *on définit la fonction $\mathcal{S}ol$ sur un $\mathbb{T}$-matching système $\mathcal{S}$ comme la fonction retourant une liste ordonnée par $\prec$ des solutions de toutes les $\mathbb{T}$-matching équations de $\mathcal{S}$.*

Notons que quand l'algorithme de filtrage échoue la fonction $\mathcal{S}$ol retourne la liste vide. La solution du filtrage de deux termes du premier ordre quand elle existe peut être trouvée par un algorithme simple comme celui de G. Huet [Hue76]. Dans $\mathbb{T}_\emptyset$, la solution du filtrage entre de termes t_1 et t_2 peut être trouver en utilisant le système de réécriture suivant où $\wedge$ est supposé comme étant associatif et commutatif et où $\diamond_1, \diamond_2$ peuvent être soit un symbole constant soit l'un des symboles préfixes suivants : "," ou "$\bullet$" ou "$\rightarrow$".

Decompose	$\diamond_1(t_1 \dots t_n) \ll_{\mathbb{T}_\emptyset} \diamond_2(t'_1 \dots t'_m)$	$\rightsquigarrow$	$\begin{cases} \bigwedge_{i=1\dots n}(t_i \ll_{\mathbb{T}_\emptyset} t'_i)\ si\ \diamond_1 = \diamond_2\ et\ n = m \\ \mathbb{F}\ sinon \end{cases}$
Merge	$(X \ll_{\mathbb{T}_\emptyset} t) \wedge (X \ll_{\mathbb{T}_\emptyset} t')$	$\rightsquigarrow$	$\begin{cases} (X \ll_{\mathbb{T}_\emptyset} t)\ si\ t =_{\mathbb{T}_\emptyset} t' \\ \mathbb{F}\ sinon \end{cases}$
Clash	$t \ll_{\mathbb{T}_\emptyset} X$	$\rightsquigarrow$	$\mathbb{F}\ si\ t \notin \mathcal{V}$
Propagate	$\mathbb{F} \wedge (t \ll_{\mathbb{T}_\emptyset} t')$	$\rightsquigarrow$	$\mathbb{F}$

Figure 1.4:

A partir d'un système de matching $\mathcal{S}$, l'application des règles ci-dessus termine et retourne $\mathbb{F}$ s'il n'y a pas de substitution solution du système ; sinon elle retourne un système $\mathcal{S}'$ en "forme nomale" d'où l'on peut facilement extraire la solution du problème de matching comme expliqué dans [KK99].

Sémantique opérationnelle

Pour un ordre $\prec$ donné et une théorie $\mathbb{T}$, la sémantique opérationnelle est définie par les règles d'inférence de la Figure 1.5.

L'idée centrale de la règle *Fire* est que l'application d'une règle de réécriture $t_1 \rightarrow t_2$ au terme t_3 en position de tête consiste à appliquer toutes les solutions de la $\mathbb{T}$-matching équation de filtrage dans l'ordre de la liste (ordonnée par $\prec$) retournée par la fonction $\mathcal{S}ol(t_1 \ll_{\mathbb{T}} t_3)$ au terme t_3. Quand il n'y a pas de

$$
\begin{array}{llll}
Fire & (t_1 \to t_2) \bullet t_3 & \mapsto_{\mathbb{T}} & \left\{ \begin{array}{l} null \text{ si } t_1 \ll_{\mathbb{T}} t_3 \text{ n'a pas de solution} \\ \sigma_1 t_2, \ldots, \sigma_n t_2 \text{ où } \sigma_i \in Sol(t_1 \ll_{\mathbb{T}} t_3) \ (n \leq \infty) \end{array} \right. \\
Distrib & (t_1, t_2) \bullet t_3 & \mapsto_{\mathbb{T}} & t_1 \bullet t_3, t_2 \bullet t_3 \\
Distrib' & null \bullet t & \mapsto_{\mathbb{T}} & null
\end{array}
$$

Figure 1.5: Règles d'évaluation du Rho Calcul

solution à cette équation, la constante spéciale *null* est renvoyée. Il est important de noter que si t_1 est une variable alors la règle *Fire* est exactement la β-règle du λ-calcul([Bar84]). En résumé on peut dire que par rapport aux autres versions du calcul, on a modifié la notation de l'application mais surtout, les règles d'évaluation ont été simplifiées d'un côté et généralisées d'un autre et ceci de manière à pouvoir gérer des structures génériques. Il est important de noter que ce calcul permet de coder par exemple le calcul orienté objet de Abadi et Cardelli ([AC96]) de manière naturelle et très simple. On peut ainsi exprimer de nouveaux objets comme des méthodes paramétrées. Pour plus de renseignements sur le ρ_{MP}-calcul, nous renvoyons le lecteur à [CKL01].

Chapter 2

Le $\rho_{\mathcal{E}}$-calcul et le $\rho_{\mathcal{E}\mathrm{V}}$-calcul

Nous introduisons ici un nouveau calcul : le ρ_{ε}-calcul. Ce calcul est né lors du travail que nous réalisions avec Claude Kirchner sur l'exprimabilité du `first` dans le $\rho_{\emptyset}$-calcul. Notre idée était la suivante : au lieu d'enrichir le ρ-calcul avec l'opérateur `first`, quels pourraient être les ingrédients à rajouter pour que le `first` soit exprimable dans une extension du $\rho_{\emptyset}$-calcul? La première difficulté que nous avons rencontrée a été de construire un calcul dans lequel l'ensemble vide ne soit plus surchargé de sens. La deuxième difficulté a été de combiner la propriété de strictité à la possibilité de tester l'échec. Nous construisons ainsi un calcul :

- simulant des comportements de type exceptions ;
- confluent avec l'appel par valeur ;
- dans lequel, si l'appel par valeur est la stratégie d'évaluation utilisée alors le `first` est exprimable.

2.1 Le `first` peut-il s'exprimer dans le $\rho_{\emptyset}$-calcul ?

Le choix qui a été fait dans le $\rho_{\emptyset}$-calcul et plus généralement dans le ρ-calcul est de représenter un ensemble vide de résultats de la même manière que l'on représente l'échec dans l'application d'une règle $l \to r$ à un terme t. On a par exemple : $[a \to \ \{\}\ ](\mathrm{a}) \longrightarrow_{\rho} \{\}$ où $\{\}$ représente un ensemble vide de résultats, et $[a \to \ \{\}\ ](\mathrm{b}) \longrightarrow_{\rho} \{\}$ où $\{\}$ représente un échec dans l'application de la règle $a \to \ \{\}$ à b. Dans le ρ-calcul ainsi défini, il ne nous est donc pas possible de tester si le résultat de l'application d'une règle est un échec ou non. Cette condition est pourtant nécessaire si l'on veut définir le `first` dans le ρ-calcul puisque, rappelons-le, le `first` est définit par :

$$
\begin{array}{lll}
First' \quad [first(s_1,\ldots,s_n)](t) & \Longrightarrow & \{u_k\downarrow\} \\
 & & \text{si } [s_i](t) \xrightarrow{*}_{\rho} \{\}\ , 1 \leqslant i \leqslant k-1 \\
 & & \quad [s_k](t) \xrightarrow{*}_{\rho} u_k \downarrow \neq \{\}\ ,\ u_k\downarrow \ clos,\ en\ forme\ normale \\
First'' \quad [first(s_1,\ldots,s_n)](t) & \Longrightarrow & \{\} \\
 & & \text{si } [s_i](t) \xrightarrow{*}_{\rho} \{\}\ , 1 \leqslant i \leqslant n
\end{array}
$$

(où $\{\}$ représenterait un échec).

La condition dans $First'$ indiquant que u_k doit être en forme normale et clos peut paraître à première vue restrictive. En fait, c'est uniquement parce qu'elle inclut des conditions sur la forme des résultats afin de ne pas altérer les propriétés de confluence et d'assurer une définition correcte du `first`. On évitera grâce à ces restrictions les situations suivantes :

- $[first(x \to [a](b), [l \to r](t))](a) \xrightarrow{*}_{\rho} [a](b) \xrightarrow{*}_{\rho} \{\}$
- $[x \to [first(y \to [x](y))](b)](a) \xrightarrow{*}_{\rho} [x \to [x](b)](a) \xrightarrow{*}_{\rho} [a](b) \xrightarrow{*}_{\rho} \{\}$

Le `first` paraît donc difficilement exprimable dans le $\rho_\emptyset$-calcul. Essayons d'enrichir ce calcul afin de pouvoir distinguer :

- les échecs dans l'application de règles ;
- les ensembles vides de résultats.

2.2 Vers un premier enrichissement du calcul

Pour pouvoir distinguer un ensemble vide de résultats d'un échec, introduisons le symbole $\perp$ (n'apparaissant pas dans la signature de notre calcul) utilisé pour indiquer l'échec dans l'application d'une règle $l \to r$. Redéfinissons tous les opérateurs impliqués dans le calcul.

On note classiquement $\mathcal{S}olution(l \ll^?_\emptyset t)$ l'ensemble des substitutions obtenues en filtrant syntaxiquement l et t comme dans [Cir00]. Nous considérons une application de l'ensemble des substitutions au niveau méta du calcul représentée par l'opérateur binaire "$\_\langle\!\langle\_\rangle\!\rangle$" dont le comportement est décrit par les méta-règles suivantes :

$$\begin{array}{lllll} Propagate & r\langle\!\langle\{\sigma_1,\ldots,\sigma_n\}\rangle\!\rangle & \rightsquigarrow & \{\sigma_1 r,\ldots,\sigma_n r\} \\ & & & \mathsf{si}\ n>0 \\ PropagateEmpty & r\langle\!\langle\ \{\}\ \rangle\!\rangle & \rightsquigarrow & \{\perp\} \end{array}$$

Ayant réajustée la définition de "$\_\langle\!\langle\_\rangle\!\rangle$", la règle $Fire$ construite sur l'opérateur modifié, ne change pas. On obtient la règle décrite dans la Figure 2.1.

$$Fire \quad [l \to r](t) \quad \Longrightarrow \quad r\langle\!\langle \mathcal{S}olution(l \ll^?_\emptyset t)\rangle\!\rangle$$

Figure 2.1: Règle d'évaluation $Fire$ dans le $\rho_\emptyset$-calcul étendu

Dans le $\rho_\emptyset$-calcul, en considérant les règles sur les ensembles pour un nombre nul d'élément, le comportement de l'ensemble vide est totalement explicite. Par contre, ayant introduit le nouveau symbole $\perp$, nous devons définir la manière d'opérer avec celui-ci. On a donc besoin des règles d'évaluation décrites dans Figure 2.2 qui décrivent une propagation de $\perp$ (nous verrons que dans l'état actuel du calcul, elle n'est pas stricte).

$$\begin{array}{lllll} AppBottomOperand & [v](\perp) & \Longrightarrow & \{\perp\} \\ AppBottomOperator & [\perp](v) & \Longrightarrow & \{\perp\} \\ RuleBottomL & \perp \to v & \Longrightarrow & \{\perp\} \\ RuleBottomR & v \to \perp & \Longrightarrow & \{\perp\} \\ OpOnbottom & f(t_1,\ldots,t_k,\perp,t_{k+1},\ldots,t_n) & \Longrightarrow & \{\perp\} \\ FlatBottom & \{t_1,\ldots,\perp,\ldots,t_n\} & \Longrightarrow & \{t_1,\ldots,t_n\} \\ & & & \mathsf{si}\ n>0 \end{array}$$

Figure 2.2: Règles d'évaluation pour l'opérateur $\perp$

On notera que la condition $n > 0$ dans la règle $FlatBottom$ de la Figure 2.2 est une condition indispensable pour éviter que $\{\perp\} \xrightarrow{*}_\rho \{\}$, ce qui nous supprimerait la distinction recherchée! En rajoutant aux règles du $\rho_\emptyset$-calcul les règles d'évaluation de la Figure 2.2, on obtient le $\rho_\perp$-calcul dont les règles sont présentées dans la Figure 2.3.

$Fire$	$[l \to r](t)$	$\Longrightarrow$	$r\langle\!\langle Solution(l \ll^?_\emptyset t)\rangle\!\rangle$
$Congruence$	$[f(t_1,\ldots,t_n)](f(u_1,\ldots,u_n))$	$\Longrightarrow$	$\{f([t_1](u_1),\ldots,[t_n](u_n))\}$
$CongruenceFail$	$[f(t_1,\ldots,t_n)](g(u_1,\ldots,u_n))$	$\Longrightarrow$	$\{\bot\}$
$Distrib$	$[\{u_1,\ldots,u_n\}](v)$	$\Longrightarrow$	si $n>0$, $\{[u_1](v),\ldots,[u_n](v)\}$
			si $n=0$, $\{\bot\}$
$Batch$	$[v](\{u_1,\ldots,u_n\})$	$\Longrightarrow$	si $n>0$, $\{[v](u_1),\ldots,[v](u_n)\}$
			si $n=0$, $\{\bot\}$
$SwitchR$	$u \to \{v_1,\ldots,v_n\}$	$\Longrightarrow$	$\{u \to v_1,\ldots,u \to v_n\}$
			si $n>0$
$OpOnSet$	$f(v_1,\ldots,\{u_1,\ldots,u_n\},\ldots,v_m)$	$\Longrightarrow$	$\{f(v_1,\ldots,u_i,\ldots,v_m)\}_{1\leqslant i\leqslant n}$
			si $n>0$
$Flat$	$\{u_1,\ldots,\{v_1,\ldots,v_m\},\ldots,u_n\}$	$\Longrightarrow$	$\{u_1,\ldots,v_1,\ldots,v_m,\ldots,u_n\}$
$AppBottomOperand$	$[v](\bot)$	$\Longrightarrow$	$\{\bot\}$
$AppBottomOperator$	$[\bot](v)$	$\Longrightarrow$	$\{\bot\}$
$RuleBottomL$	$\bot \to v$	$\Longrightarrow$	$\{\bot\}$
$RuleBottomR$	$v \to \bot$	$\Longrightarrow$	$\{\bot\}$
$OpOnBottom$	$f(t_1,\ldots,t_k,\bot,t_{k+1},\ldots,t_n)$	$\Longrightarrow$	$\{\bot\}$
$FlatBottom$	$\{t_1,\ldots,\bot,\ldots,t_n\}$	$\Longrightarrow$	$\{t_1,\ldots,t_n\}$
			si $n>0$

Figure 2.3: Règles d'évaluation du $\rho_\bot$-calcul

Dans la suite, nous parlerons de *strictité* du calcul pour dénoter la propagation stricte de l'erreur. Regardons maintenant de plus près les conséquences de l'ajout de telles règles dans notre calcul. Pour définir le `first`, nous avons besoin d'un terme exprimant le fait suivant : *si* $res = \bot$ *alors faire* c; ce qui ne peut s'exprimer dans l'état actuel de notre calcul uniquement par le terme $[\bot \to c](res)$ et si $res = \bot$ le filtrage réussit et $[\bot \to c](res)$ s'évalue en $\{c\}$. Il apparaît donc nécessaire de réaliser le filtrage du résultat res de manière plus soutenue, par l'intermédiaire d'une nouvelle fonction (n'apparaissant pas dans la signature) introduite pour pouvoir distinguer la propagation de l'erreur du "rattrapage" de celle-ci.

Une des propriétés forte du $\rho_\emptyset$-calcul était sa strictité. Malheurement, cette propriété ne peut pas être conservée dans notre nouveau calcul si l'on veut avoir $[\bot \to c](\bot) \longrightarrow_{Fire} \{c\}$. Nous devons donc modifier notre système. Notons de plus que la propagation de l'erreur n'est pas totalement assurée avec les termes contenant $\{\}$. Par exemple, $f(\ \{\}\ , \bot) \longrightarrow_\rho \{\}$ est un comportement que nous voudrions éviter pour que la propagation de l'erreur soit totale et pour ne pas altérer les propriétés de confluence, puisque l'on a aussi $f(\ \{\}\ , \bot) \longrightarrow_\rho \{\bot\}$.

En résumé, il apparaît donc nécessaire de modifier et/ou enrichir nos règles d'évaluation pour permettre :

- la strictité du calcul ;
- la non-altération des propriétés de confluence;
- le rattrapage d'erreur (afin de pouvoir définir le `first`).

2.3 Vers une version plus adaptée du calcul : le ρ_ε-calcul

2.3.1 Le sens donné à l'ensemble vide

Dans le ρ_ε-calcul, l'ensemble vide représente à la fois un ensemble vide de résultat et un échec dans l'application d'une règle. Le choix qui a été fait en étendant le calcul est de donner une unique signification à l'ensemble vide : représenter un ensemble vide de résultats comme un résultat à part entière. On pourrait finalement penser que l'ensemble vide correspondrait à un terme $*^{void}$ de la même manière que

dans le système $\mathbb{T}$ de Gödel. Dans cette perspective, ce terme ne peut pas s'appliquer et ne peut pas être appliquer à un autre terme. On veut que :

$$[v](\{\}) \xrightarrow{*}_\rho \{\bot\} \textbf{ et } [\{\}](v) \xrightarrow{*}_\rho \{\bot\}$$

Par contre, $f(t_1, \ldots, \{\}, \ldots, t_n)$ ne doit pas générer un échec mais représenter un résultat sous forme normale (si tous les tx_i sont en forme normale). En effet, même si l'on veut interdire que l'ensemble vide soit appliqué, il est intéressant qu'il puisse être l'argument d'une fonction sans pour autant mener à un échec. Nous verrons par la suite que cela permet d'obtenir une plus grande expressivité (cf. *exn*). On modifie donc $Distrib, Batch, OpOnSet$ en conséquence et on obtient les règles présentées dans la Figure 2.4.

$Distrib$	$[\{u_1, \ldots, u_n\}](v)$	$\Longrightarrow$	si $n > 0$, $\{[u_1](v), \ldots, [u_n](v)\}$ si $n = 0$, $\{\bot\}$
$Batch$	$[v](\{u_1, \ldots, u_n\})$	$\Longrightarrow$	si $n > 0$, $\{[v](u_1), \ldots, [v](u_n)\}$ si $n = 0$, $\{\bot\}$
$OpOnSet$	$f(v_1, \ldots, \{u_1, \ldots, u_n\}, \ldots, v_m)$	$\Longrightarrow$	$\{f(v_1, \ldots, u_i, \ldots, v_m)\}_{1 \leqslant i \leqslant n}$ si $n > 0$

Figure 2.4: Règles d'évaluation $Distrib, Batch, OpOnSet$ du nouveau calcul

2.3.2 Nouvelle extension du calcul

Pour pouvoir définir le `first`, nous voudrions pouvoir rattrapper un échec. Il nous faut donc (si on veut garder la strictité) un nouvel opérateur, de manière à ce que l'on ait

$$[operateur(\bot) \to u](operateur(\bot)) \longrightarrow_\rho u$$

Nous introduisons ce nouvel opérateur `exn` régi par la règle suivante :

Exn	$exn(t)$	$\Longrightarrow$	$\{t\}$ si $\{t\}\downarrow \neq \{\bot\}$ $t\downarrow$ *est clos*

Nous avons mentionné le fait que $f(\ \{\}\ , \bot) \longrightarrow_\rho \{\}$. Ce problème est résolu grâce aux restrictions présentées dans la Figure 2.4.

2.3.3 Définition du ρ_ε-calcul

Dans le $\rho_\emptyset$-calcul, seuls étaient autorisés pour les membres gauches des abstractions les termes du premier ordre. Cela nous permettait de définir un calcul suffisament expressif et, grâce à des stratégies d'évaluation particulières, confluent. En introduisant $\bot$ et *exn*, nous voudrions définir une extension du $\rho_\emptyset$-calcul qui garde ces propriétés. Modulo une stratégie d'évaluation raisonnable, nous souhaitons un calcul confluent dont l'expressivité soit celle du $\rho_\emptyset$-calcul enrichi du rattrapage de l'échec. On définit donc une notion de terme du premier ordre plus adaptée.

Définition 2.1 *Un ρ_ε-terme t est dit du ε-premier ordre si t est construit à l'aide de la grammaire suivante (où $f \in \mathcal{F}$) :*

$$\textbf{\textit{ρ_ε-terme du ε-premier ordre}} \qquad t \quad ::= \quad x \ \mid \ f(t, \ldots, t) \ \mid \ exn(\bot)$$

On définit ainsi le $\rho_\emptyset$-calcul étendu ou ρ_ε-calcul. Pour cela, commençons par étendre toutes les notions relatives aux positions afin de pouvoir distinguer les positions fonctionnelles, les positions du type $exn, \perp$.

Définition 2.2 *On définit :*

- $\mathcal{P}os(t)$ *l'ensemble des positions de* t *;*
- $\mathcal{FP}os(t)$ *l'ensemble des positions fonctionnelles de* t *par :* $\mathcal{FP}os(t) = \{p \in \mathcal{P}os(t) | t(p) \in \mathcal{F}\}$ *;*
- $\mathcal{F}_\varepsilon$ *l'ensemble défini par :* $\mathcal{F}_\varepsilon = \mathcal{F} \cup \{exn, \perp\}$ *;*
- $\mathcal{P}os_{\{\}}(t) = \{p \in \mathcal{P}os(t) | t(p) = \{\}\}$
- $\mathcal{P}os_\perp(t) = \{p \in \mathcal{P}os(t) | t(p) = \perp\}$
- $\mathcal{P}os_\varepsilon(t) = \{p \in \mathcal{P}os(t) | t(p) = exn \text{ ou } t(p) = \perp\}$
- $\mathcal{F}_\varepsilon\mathcal{P}os(t)$ *l'ensemble des positions* ε*-fonctionnelles de* t *défini par :* $\mathcal{F}_\varepsilon\mathcal{P}os(t) = \mathcal{FP}os(t) \cup \mathcal{P}os_\varepsilon(t)$

Définition 2.3 *Etant donné un ensemble de symboles de fonctions* $\mathcal{F}$*, un ensemble de variables* $\mathcal{X}$*, nous appelons* ρ_ε*-calcul le calcul défini par :*

- *un sous-ensemble* $\varrho(\mathcal{F}_\varepsilon, \mathcal{X})$ *de l'ensemble de termes* $\varrho(\mathcal{F}_\varepsilon, \mathcal{X})$ *tel que toutes les règles de réécriture soient de la forme* $l \to r$ *avec* l *du* ε*-premier ordre ;*
- *l'application (d'ordre supérieur) de substitution aux termes ;*
- *la théorie* $\emptyset$ *(filtrage syntaxique) avec l'algorithme de filtrage ci-dessus ;*
- *les règles d'évaluation de la figure 2.5 ;*
- *une stratégie d'évaluation* $\mathcal{S}$ *qui guide l'application des règles d'évaluation.*

Fait 2.1 *Soit* t *un terme du* ρ_ε*-calcul. Pour tous les réduits de* t*, toutes leurs règles de réécriture ont un membre gauche du* ε*-premier ordre .*

Preuve : La preuve est immédiate en raisonnant par l'absurde. □

La règle *RuleBottomL* est donc inutile dans le ρ_ε-calcul, elle ne fait donc pas partie de l'ensemble des règles d'évaluation.

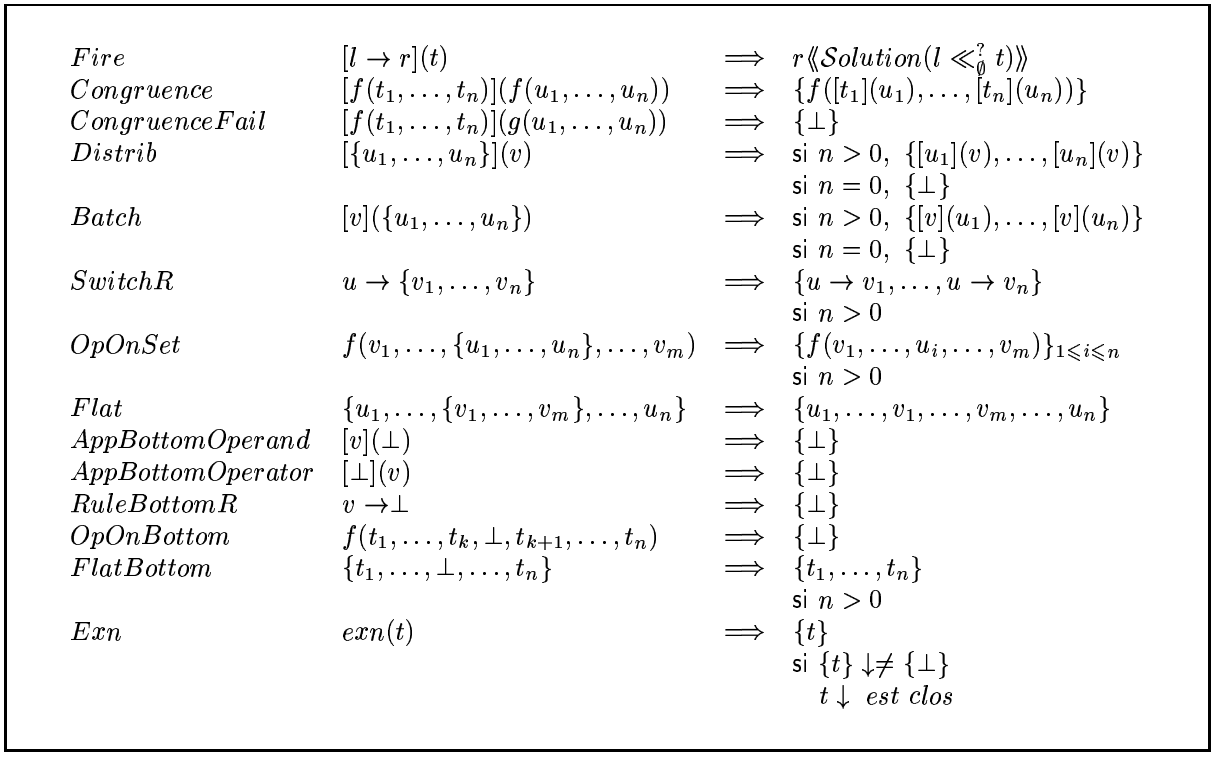
Pour conclure ce chapitre, signalons qu'il nous reste à examiner la confluence du ρ_ε-calcul. On voudrait, comme dans le $\rho_\emptyset$-calcul, exprimer une stratégie d'évaluation raisonnable nous garantissant la confluence. Signalons pour finir que le rôle et le comportement de $exn(\perp)$ devra être précisé et éventuellement contrôlé par la stratégie d'évaluation.

2.4 Le ρ_ε-calcul par l'exemple

Nous allons donner maintenant deux exemples de l'utilisation de $exn(\perp)$: un premier exemple illustrant le rattrapage de l'erreur.

$$[\{True \to P_{err}, False \to P_{normal}\}]([first(exn(\perp) \to True, x \to False)](exn(M)))$$

Si M est évalué en une erreur alors P_{err} est évalué. Sinon, P_{normal} est évalué. P peut par exemple réaliser une division et si l'on effectue une division par zéro P_{err} est executé sinon c'est P_{normal} qui l'est. Avec le `first` et le $exn(\perp)$, nous sommes en mesure de mettre en place tout un mécanisme du type : si l'évaluation se passe sans problème alors on évalue la "suite" du terme, sinon on déclanche un comportement exceptionnel. Nous ne nous étendrons pas sur tout l'intérêt du traitement des exceptions. Nous verrons que le $exn(\perp)$ nous permettra de définir le `first` dans le ρ_ε-calcul et donc des stratégies de normalisation. A l'aide du `first`, nous donnons maintenant l'expression des principaux opérateurs de normalisation qui nous



$Fire$	$[l \to r](t)$	$\Longrightarrow$	$r\langle\!\langle Solution(l \ll^?_\emptyset t)\rangle\!\rangle$
$Congruence$	$[f(t_1,\ldots,t_n)](f(u_1,\ldots,u_n))$	$\Longrightarrow$	$\{f([t_1](u_1),\ldots,[t_n](u_n))\}$
$CongruenceFail$	$[f(t_1,\ldots,t_n)](g(u_1,\ldots,u_n))$	$\Longrightarrow$	$\{\bot\}$
$Distrib$	$[\{u_1,\ldots,u_n\}](v)$	$\Longrightarrow$	si $n>0$, $\{[u_1](v),\ldots,[u_n](v)\}$
			si $n=0$, $\{\bot\}$
$Batch$	$[v](\{u_1,\ldots,u_n\})$	$\Longrightarrow$	si $n>0$, $\{[v](u_1),\ldots,[v](u_n)\}$
			si $n=0$, $\{\bot\}$
$SwitchR$	$u \to \{v_1,\ldots,v_n\}$	$\Longrightarrow$	$\{u \to v_1,\ldots,u \to v_n\}$
			si $n>0$
$OpOnSet$	$f(v_1,\ldots,\{u_1,\ldots,u_n\},\ldots,v_m)$	$\Longrightarrow$	$\{f(v_1,\ldots,u_i,\ldots,v_m)\}_{1\leqslant i\leqslant n}$
			si $n>0$
$Flat$	$\{u_1,\ldots,\{v_1,\ldots,v_m\},\ldots,u_n\}$	$\Longrightarrow$	$\{u_1,\ldots,v_1,\ldots,v_m,\ldots,u_n\}$
$AppBottomOperand$	$[v](\bot)$	$\Longrightarrow$	$\{\bot\}$
$AppBottomOperator$	$[\bot](v)$	$\Longrightarrow$	$\{\bot\}$
$RuleBottomR$	$v \to \bot$	$\Longrightarrow$	$\{\bot\}$
$OpOnBottom$	$f(t_1,\ldots,t_k,\bot,t_{k+1},\ldots,t_n)$	$\Longrightarrow$	$\{\bot\}$
$FlatBottom$	$\{t_1,\ldots,\bot,\ldots,t_n\}$	$\Longrightarrow$	$\{t_1,\ldots,t_n\}$
			si $n>0$
Exn	$exn(t)$	$\Longrightarrow$	$\{t\}$
			si $\{t\}\downarrow \neq \{\bot\}$
			$t\downarrow$ *est clos*

Figure 2.5: Règles d'évaluation du ρ_ε-calcul

permettront d'exprimer les stratégies de normalisation *innermost* et *outermost*. Pour plus de lisibilité dans les expressions des ρ_ε-termes nous noterons $id \triangleq x \to x$ et $(u;\, v) = (x \to [v]([u](x)))$.

- Un opérateur de congruence générique

  $$\Phi(r) \triangleq first(f_1(r,id,\ldots,id),\ldots,f_1(id,\ldots,id,r),\ldots,f_m(r,id,\ldots,id),\ldots,f_m(id,\ldots,id,r))$$

  Exemples :

  $[\Phi(a \to c)](f(a,b)) \Rightarrow_{\rho_\varepsilon} \{f(c,b)\}$
  $[\Phi(a \to b)](c) \Rightarrow_{\rho_\varepsilon} \{\bot\}$

- Un combinateur de point fixe

  $$\Theta = [A](A)\ avec\ A = x \to (y \to [y]([[x](x)](y))).$$

- Une application bottom-up

  $$Once_{bu}(r) \triangleq [\Theta](H_{bu}(r))\ avec\ H_{bu}(r) \triangleq f \to (x \to [first(\Phi(f),r)](x))$$

  Exemple : $[Once_{bu}(a \to b)](f(a,g(a)) \Rightarrow_{\rho_\varepsilon} \{f(b,g(a))\})$

- Opérateur de répétition

  $$repeat*(r) \triangleq [\Theta](J(r))\ avec\ J(r) \triangleq f \to (x \to [first(r;f,\ id)](x))$$

  Exemple : $[repeat*(\{a \to b, b \to c\})](a) \Rightarrow_{\rho_\varepsilon} \{c\}$

- Opérateurs de normalisation

$$im(r) \triangleq repeat*(Once_{bu}(r)) \text{ et } om(r) \triangleq repeat*(Once_{td}(r))$$

Exemples de normalisation : On s'intéresse à la normalisation de $f(a, g(a))$ par la stratégie de normalisation *innermost* dans le système $\mathcal{R} = \{a \to b, f(x, g(x)) \to x\}$. On a : $[im(\mathcal{R})](f(a, g(a))) \Rightarrow_{\rho_\varepsilon} \{b\}$. On s'intéresse maintenant à la normalisation de $f(a, a)$ par la stratégie de normalisation *innermost* et pour la stratégie *outermost* dans le système $\mathcal{R}' = \{a \to b, a \to c, f(x, x) \to x\}$.
On a : $[im(\mathcal{R}')](f(a, a)) \Rightarrow_{\rho_\varepsilon} \{b, f(c, b), f(b, c), c\}$; $[om(\mathcal{R}')](f(a, a)) \Rightarrow_{\rho_\varepsilon} \{b, c\}$. Nous voulons maintenant réaliser le rattrapage d'échec sur les arguments d'une fonction f : on veut "coder" le terme réalisant le comportement suivant :

si les deux arguments de f s'évalue en une erreur *alors* évaluer $P_{err\ 1\&2}$

sinon si le premier argument de f est une *alors* évaluer P_{err1} erreur

sinon si le deuxième argument de f *alors* évaluer p_{err2} est une erreur

sinon évaluer P_{normal}

Ceci peut s'exprimer dans le ρ_ε-calcul par :
$[\{True \to P_{err1\&2}, False \to T_1\}]([first(f(exn(\bot), exn(\bot)) \to True, x \to False)](f(exn(M_1), exn(M_2))))$
où $T_1 \triangleq [\{True \to P_{err1}, False \to T_2\}]([first(f(exn(\bot), x) \to True, x \to False)](f(exn(M_1), M_2)))$
où $T_2 \triangleq [\{True \to P_{err2}, False \to P_{normal}\}]([first(f(x, exn(\bot)) \to True, x \to False)](f(M_1, exn(M_2))))$

2.5 Sur la confluence du ρ_ε-calcul

Dans [Cir00], on montre qu'il est possible de munir le $\rho_\emptyset$-calcul d'une stratégie d'évaluation de sorte que celui-ci soit confluent. Nous voudrions généraliser ceci au ρ_ε-calcul. A première vue, la généralisation peut paraître facile mais plusieurs difficultés viennent se greffer à celles déjà présentes :

- l'existence de la règle Exn ;
- l'existence d'un nombre plus important de possibilités pour obtenir un terme se réduisant en $\{\bot\}$ (cf. $Distrib$, $Batch$) ;
- le rôle de $exn(\bot)$.

L'analyse de ce rôle et des exemples ci-dessous nous amènerons ainsi à prouver la confluence du ρ_ε-calcul pour une stratégie d'évaluation basée sur l'appel par valeur. La preuve généralisera celle présentée dans [Cir00] tout en gardant une structure similaire. Dans un second temps, nous montrerons que si l'on munit le ρ_ε-calcul de l'appel par valeur, alors celui est confluent. Dans le sens d'appel par valeur, nous entendons que $Fire$ est appliquée à $[l \to r](t)$ que si t est une valeur et que Exn est appliquée à $exn(t)$ de même que si t est une valeur.

2.5.1 Le sens de $exn(\bot)$ et la non-confluence du ρ_ε-calcul

Le sens de $exn(\bot)$

$exn(\bot)$ a pour unique but de réaliser le rattrapage d'erreur et ainsi de "simuler" un comportement de type exception. Soit l'erreur est rattrapée comme par exemple dans $[exn(\bot) \to c](exn([a](b)))$, soit elle n'est pas rattrapée et dans ce cas-là, on ne voudrait pas que $exn(\bot)$ soit propagé comme pour $\bot$. Par exemple, $[x \to c](exn(\bot)) \Longrightarrow_{\rho_\varepsilon} \{c\}$. Ce comportement ne nous pose aucun problème. cela permet d'exprimer le fait que l'on veut ignoner l'exception qui a été levée. Par contre; un terme de la forme $[\{True \to P_{err}, False \to P_{normal}\}]([first(exn(\bot) \to True, x \to False)](exn(M)))$ nous permet bien de "cibler" l'évaluation selon qu'il y ait échec ou non. Donc nous allons dans notre stratégie d'évaluation autoriser les réductions du type $[x \to P](exn(\bot)) \Longrightarrow_{\rho_\varepsilon} \{P\}$ puisque du point de vue sémantique, si cet exn avait été placé dans le terme de départ, il bloquait nécessairement la propagation de $\bot$. Donc le résultat est bien conforme à ce que l'on attend.

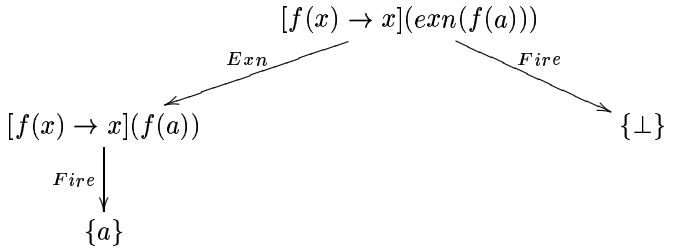
La non-confluence du ρ_ε-calcul

Nous allons analyser à travers des exemples la non-confluence du ρ_ε-calcul. Nous commencons par citer sans détail les mêmes exemples que ceux mentionnés dans [Cir00].

1. Variable potentiellement instanciée : $[x \to [a \to b](x)](a)$
2. Terme non réduit : $[f(x) \to x](\{f(a)\})$
3. Sous-terme non réduit : $[a \to b]([a \to a](a))$
4. Non-linéarité à gauche : $[f(x,x) \to x](f(a,[a \to a](a))$
5. Non-linéarité à droite : $[x \to f(x,x)](\{a,b\})$

Il semble intéressant de mentionner les problèmes dus à l'application de $Fire$ sur $[l \to r](t)$ si la règle Exn est applicable à t.

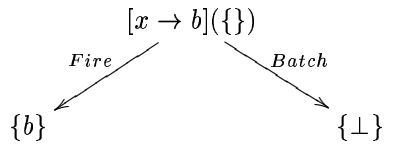
Exemple 2.1 *[Termes non suffisament réduit par rapport à Exn]*



Il convient donc d'appliquer $Fire$ à $[l \to r](t)$ uniquement si t est suffisament réduit par rapport à Exn (pas nécessairement entièrement réduit mais réduit aux positions correspondant à une position ε-fonctionnelle dans l). Ainsi l'échec du filtrage ne peut qu'être dû à des "clashs" de symboles ne disparaissant pas par ρ_ε-réduction.

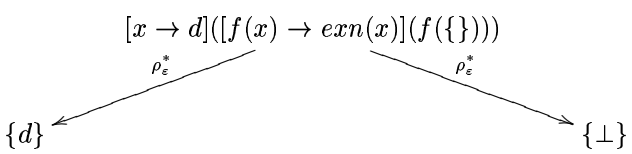
Par rapport au $\rho_\emptyset$-calcul , savoir si un terme peut se réduire en $\{\bot\}$ est quelque chose de plus difficile. En effet, si l'on était sûr que le terme t ne contenait pas $\{\}$ et qu'aucun échec n'était possible alors on pourrait assurer que t ne se réduirait pas en $\{\}$. Dans le ρ_ε-calcul, la difficulté vient du fait que $Distrib$ et $Batch$ peuvent mener à $\{\bot\}$. Il faut donc pour que mon terme ne se réduise pas en $\{\bot\}$, tester (entre autres) s'il ne contient pas d'application de la forme $[u](v)$ où u ou v pourrait se réduire en $\{\}$. C'est un test un peu délicat.

Exemple 2.2 *[Opérande d'une application égale à $\{\}$]*



Le problème de confluence est ici facile à résoudre mais ce n'est pas le cas dans l'exemple suivant puisqu'il faut tester si l'opérande de l'application ne se réduit pas en l'ensemble vide.

Exemple 2.3 *[Opérande se* réduisant *en $\{\}$]*



Notons que l'on montre trivialement qu'il est nécessaire que l'ensemble vide soit présent dans t pour que t se réduise en $\{\}$. On pourrait penser à première vue que tester si $t \longrightarrow_{\rho_\varepsilon} t'$ et $t' \xrightarrow{*}_{\rho_\varepsilon} \{\}$ est un test pouvant s'effectuer en analysant les positions des variables l, r et les positions de $\{\}$ dans r et t mais ce test n'est pas valide comme indiqué dans l'exemple suivant.

Exemple 2.4 *Considérons la réduction :*

$$\begin{array}{ll} & t \triangleq [f(x) \to [h(y) \to y](h(x))](f(\{\})) \\ \xrightarrow{*}_\rho & [f(x) \to x](f(\{\})) \\ \xrightarrow{*}_\rho & \{\} \end{array}$$

Nous verrons qu'en fait il suffit d'imposer l'évaluation de certaines des applications présentes dans le terme t.

Exemple 2.5 (Instanciation par $\{\}$) $[f(x) \to c](f(\{\})) \Longrightarrow_{\rho_\varepsilon} \{c\}$

Bien que cela ne nous pose pas de problèmes pour la confluence, mais étant donné le sens que l'on a donné à l'ensemble vide, il serait judicieux que lors du filtrage aucune variable ne puisse être instanciée par $\{\}$. Cette condition peut facilement être obtenue si la règle EmptyInstanciation est rajoutée à l'algorithme de filtrage de la même manière que pour le $\rho_{\varepsilon\mathrm{v}}$-calcul (cf. Section 2.6).

2.5.2 Confluence du ρ_ε-calcul avec la stratégie d'évaluation $ConfStrat$

Définition 2.4 *Nous appelons* ConfStrat *la stratégie d'évaluation qui consiste à appliquer*

1. *Fire sur $[l \to r](t)$ uniquement si*
   - (a) *t est clos et du ε-premier ordre ou*
   - (b) i. *l est linéaire ;*
     - ii. *t est suffisament réduit par rapport à l par la règle Exn :*
       $$\forall p \in \mathcal{P}os(t) \cap \mathcal{F}_\varepsilon \mathcal{P}os(l), \big(t(p) = exn \Rightarrow t(p.0) = \bot\big)$$
     - iii. *l ε-subsume faiblement t :* $\forall p\big(p \in \mathcal{F}_\varepsilon \mathcal{P}os(l) \cap \mathcal{P}os(t) \Rightarrow t(p) \in \mathcal{F}_\varepsilon\big)$
     - iv. *la propagation de $\bot$ a eu lieu pour t :* $\forall p \in \mathcal{P}os(t), \big(t(p) = \bot \Rightarrow (p = p'i) \wedge (t(p') = exn)\big)$
     - v. *$\{\}$ se situe à des positions satisfaisantes dans t :*
       $$\forall p \in \mathcal{P}os(t), \big(t(p) = \{\} \Rightarrow (p = p'i) \wedge ((t(p') = \to) \vee (t(p') \in \mathcal{F}))\big)$$
       et pour tout sous-terme de la forme $[u \to v](w)$: $(v \neq \{\}) \wedge (v \neq x) \wedge (v \neq \{x\})$
     - vi. *pour tout sous-terme $[u](v)$, u est une abstraction ;*
     - vii. *pour tout sous-terme de la forme $[u \to v](w)$ u subsume w ;*
     - viii. *t ne contient pas d'ensemble ayant plus d'un élément.*
2. *Exn à t uniquement si :*
   - (a) *t est clos ;*
   - (b) *la propagation de $\bot$ a eu lieu :* $\forall p \in \mathcal{P}os(t), \big(t(p) = \bot \Rightarrow (p = p'i) \wedge (t(p') = exn)\big)$
   - (c) *t ne contient pas d'application.*

Théorème 2.1 *Le ρ_ε-calcul munit de la stratégie d'évaluation* ConfStrat *est confluent.*

Preuve : cf. Annexe C □

2.6 Le $\rho_{\varepsilon \mathrm{v}}$-calcul

Nous présentons ici un résultant intéressant du ρ_ε-calcul : celui-ci est conflluent lorsqu'on le munit de l'appel par valeur ; on parle ainsi du $\rho_{\varepsilon \mathrm{v}}$-calcul. Commençons par définir l'ensemble des valeurs.

$$\textbf{valeur} \qquad v \quad ::= \quad c \mid f(v,\ldots,v) \mid f(v,\ldots,\{\},\ldots,v) \mid v \to v \mid exn(\bot)$$

Dans la suite de ce chapitre, l, r, t désigneront toujours des termes du ρ_ε-calcul et $v, v_1, \ldots, v_n$ des valeurs du ρ_ε-calcul. Afin, que lors du filtrage aucune variable ne soit instanciée en $\{\}$, nous allons modifier l'algorithme de filtrage syntaxique en rajoutant la règle *EmptyInstanciation* comme présenté dans la Figure 2.6.

Decomposition	$(f(t_1,\ldots,t_n) \ll^?_\emptyset f(t'_1,\ldots,t'_n)) \wedge P$	$\mapsto\!\!\!\!\to$	$\bigwedge_{i=1\ldots n} t_i \ll^?_\emptyset t'_i \wedge P$
SymbolClash	$(f(t_1,\ldots,t_n) \ll^?_\emptyset g(t'_1,\ldots,t'_m)) \wedge P$	$\mapsto\!\!\!\!\to$	$\mathbf{F}$ si $f \neq g$
MergingClash	$(x \ll^?_\emptyset t) \wedge (x \ll^?_\emptyset t') \wedge P$	$\mapsto\!\!\!\!\to$	$\mathbf{F}$ si $t \neq t'$
EmptyInstanciation	$(x \ll^?_\emptyset \{\}) \wedge P$	$\mapsto\!\!\!\!\to$	$\mathbf{F}$
SymbolVariableClash	$(f(t_1,\ldots,t_n) \ll^?_\emptyset x) \wedge P$	$\mapsto\!\!\!\!\to$	$\mathbf{F}$ si $x \in \mathcal{X}$

Figure 2.6: *TheCallByValueSyntacticMatching* - Règles pour le filtrage syntaxique gérant l'ensemble vide

On définit $Fire_v$ et Exn_v deux variantes des règles $Fire$ et Exn par :

$$\begin{array}{lll} Fire_v & [l \to r](v) & \Longrightarrow \quad r\langle\!\langle Solution(l \ll^?_\emptyset v)\rangle\!\rangle \\ Exn_v & exn(v) & \Longrightarrow \quad v \end{array}$$

On définit le $\rho_{\varepsilon \mathrm{v}}$-calcul comme le ρ_ε-calcul mais utilisant les règles $Fire_v, Exn_v$ au lieu des règles $Fire, Exn$ et utilisant l'algorithme de filtrage de la Figure 2.6. On pourrait montrer de la même manière que l'on a montré que le ρ_ε-calcul était confluent avec la stratégie *ConfStrat* que le $\rho_{\varepsilon \mathrm{v}}$-calcul est confluent. On notera que ce résultat est intéressant par le fait qu'il ouvre les portes vers une implémentation du $\rho_{\varepsilon \mathrm{v}}$-calcul raisonnable puisque l'appel par valeur est une technique de réduction maintenant bien connue. On notera de même que le ρ_ε-calcul muni de la stratégie d'évaluation *ConfStrat* possède la propriété de strictité, le $\rho_{\varepsilon \mathrm{v}}$-calcul la possède aussi. Nous signalerons pour conclure ce chapitre que les exceptions ne sont pas nommées dans le calcul que l'on a introduit et qu'ainsi lorqu'une exception est levée, c'est la première de la pile qui est rattrapée. Ainsi il semblerait plus riche de définir *exn* comme un symbole binaire. On retrouve le problème rencontré par Milner lors de ses premières implémentations de ML ([Rob78, WL96]).

Chapter 3

Le `first` dans le $\rho_{\varepsilon \mathrm{V}}$-calcul

Nous rappelons que le ρ_{ε}-calcul a été introduit dans un premier temps pour enrichir le $\rho_{\emptyset}$-calcul de manière à ce que le `first` soit exprimable. Toutes les ambigüités qui nous laissent croire en la non-exprimabilité du `first` dans le $\rho_{\emptyset}$-calcul ne sont plus présentes dans le ρ_{ε}-calcul. Ainsi, après avoir introduit un calcul permettant le test de l'échec et dans lequel il y a bien distinction entre un ensemble vide de résultat et un échec tous les ingrédients pour exprimer le `first` sont présents. L'idée sous-jacente du `first` est de calculer l'argument suivant uniquement si le précédent se réduit en un échec. Ceci est difficilement exprimable dans le ρ_{ε}-calcul. Nous avons donc résolu ce problème en représentant le résultat de $[first(t_1, \ldots, t_n)](r)$ par un ensemble de n termes. A chaque fois au plus un seul de ces termes ne se réduira pas en $\{\bot\}$. Notons $u_1, \ldots, u_n$ les termes de cet ensemble. Chaque terme u_i est calculé en fonction des $i-1$ précédents : si tous les $i-1$ premiers termes se réduisent en $\{\bot\}$ alors nous calculons $[t_i](r)$ sinon u_i doit être affecté d'un terme se réduisant en $\{\bot\}$. Les u_i sont donc définis par :

$$u_1 = [t_1](x)$$

$$u_2 = [exn(\bot) \to [t_2](x)](exn(u_1))$$

$$u_3 = [exn(\bot) \to [exn(\bot) \to [t_3](x)](exn(u_2))](exn(u_1))$$

$$u_4 = [exn(\bot) \to [exn(\bot) \to [exn(\bot) \to [t_4](x)](exn(u_3))](exn(u_2))](exn(u_1))$$

$$\vdots$$

$$u_n = [exn(\bot) \to [exn(\bot) \to [exn(\bot) \to [\ldots \to [t_n](x)](exn(u_{n-1}))\ldots](exn(u_3))](exn(u_2))](exn(u_1))$$

On définit donc $MyFirst$ par :

$$MyFirst(t_1, \ldots, t_n) \quad \triangleq \quad x \to \{u_1, \ldots, u_n\}$$

Cette définition n'est pas valide si nous ne nous plaçons pas dans un calcul confluent. En effet, si le calcul n'est pas confluent vu que pour réduire u_i, nous avons besoin de réduire les $u_1, \ldots, u_{i-1}$ il serait tout à fait possible qu'une réduction de u_{i_0} aboutisse à $\{\bot\}$ une fois et à $v_{i_0} \downarrow \neq \{\bot\}$ une autre fois (i.e. quand nous calculons un u_j pour $j > i_0$), dans ce cas-là le résultat n'aurait pas le sens recherché. C'est pour cette raison que nous nous plaçons dans le $\rho_{\varepsilon \mathrm{V}}$-calcul, calcul confluent. Nous allons maintenant montrer la validité de notre définition dans ce calcul.

Lemme 3.1 *Si* $\forall i [t_i](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$, *alors* $[MyFirst(t_1, \ldots, t_n)](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$ *et* $[first(t_1, \ldots, t_n)](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$ *où* `first` *est défini précisemment ici avec les deux règles* $First', First''$ *présentées dans la section 2.1.*

Preuve : cf. Annexe D □

Lemme 3.2 *Si* $\exists l$ *tel que* $\forall i \leq l-1 \;\; [t_i](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$ *et* $[t_l](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{v_l\} \downarrow \neq \{\bot\}$ *avec* v_l *clos alors* $[MyFirst(t_1, \ldots, t_n)](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{v_l\} \downarrow$ *et* $[first(t_1, \ldots, t_n)](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{v_l\} \downarrow$.

Preuve : cf. Annexe D □

Remarquons que si *si* $\exists l$ tel que $\forall i \leq l-1$ $[t_i](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$ et $[t_l](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{v_l\} \neq \{\bot\}$ avec v_l non clos alors l'évaluation de $[MyFirst(t_1,\ldots,t_n)](r)$ est en quelque sorte bloquée par la stratégie d'évaluation (appel par valeur). L'évaluation de $[first(t_1,\ldots,t_n)](r)$ est de même bloquée comme indiquée dans la définition donnée en 2.1. Ces résultats de validité étant prouvé, le `first` sera supposé défini par `MyFirst` et répondra aux mêmes règles et on peut donc énoncer le résultat suivant :

Théorème 3.1 *Le* `first` *est exprimable dans le* $\rho_{\varepsilon \mathrm{v}}$*-calcul.*

Nous allons maintenant donner deux exemples illustrant la définition que l'on vient de donner du `first`.

Exemple 3.1 *Considérons la réduction suivante :*

$$\begin{array}{ll} & [x \to [first(y \to [y](x), y \to c)](b)](a) \\ \triangleq & [x \to \{[y \to [y](x)](b), [exn(\bot) \to [y \to c](b)](exn([y \to [y](x)](b)))\}](a) \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & [x \to \{[b](x), [exn(\bot) \to [y \to c](b)](exn([b](x)))\}](a) \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & \{[b](a), [exn(\bot) \to [y \to c](b)](exn([b](a)))\} \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & \{\{\bot\}, [y \to c](b)\} \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & \{c\} \end{array}$$

On a aussi la réduction :

$$\begin{array}{ll} & [x \to [first(y \to [y](x), y \to c)](b)](a) \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & [first(y \to [y](a), y \to c)](b) \\ \triangleq & \{[y \to [y](a)](b), [exn(\bot) \to [y \to c](b)](exn([y \to [y](a)](b)))\} \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & \{\bot, \{c\}\} \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & \{c\} \end{array}$$

On voit à travers cet exemple que grâce à la confluence du ρ_{ε}*-calcul, les réductions aboutissent au même résultat ; on a donc un calcul confluent où le* `first` *est exprimable.*

Exemple 3.2 *On considère maintenant la réduction :*

$$\begin{array}{ll} & x \to [first(y \to [x](y), y \to c)](b) \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & x \to \{[x](b), [exn(\bot) \to [y \to c](b)](exn([x](b)))\} \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & \{x \to [x](b), x \to [exn(\bot) \to c](exn([x](b)))\} \end{array}$$

L'évalutation est maintenant "bloquée" et ce résultat est sous forme normale.

Exemple 3.3 *On reprend l'exemple présenté en 2.4 et on vérifie que la définition du* `first` *est bien valide sur cet exemple.*

$$\begin{array}{ll} & [\{True \to P_{err}, False \to P_{normal}\}]([first(exn(\bot) \to True, x \to False)](exn(M))) \\ \triangleq & [\{True \to P_{err}, False \to P_{normal}\}](\{[exn(\bot) \to True](exn(M)), \\ & \qquad [exn(\bot) \to [x \to False](exn(M))](exn([exn(\bot) \to True](exn(M))))\}) \end{array}$$

La stratégie d'évaluation d'appel par valeur nous oblige à évaluer M *avant d'appliquer* $Fire$*.*
Premier cas: si $M \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$ *alors on a :*

$$\begin{array}{ll} & [\{True \to P_{err}, False \to P_{normal}\}](\{[exn(\bot) \to True](exn(\{\bot\})), \\ & \qquad [exn(\bot) \to [x \to False](exn(\{\bot\}))](exn([exn(\bot) \to True](exn(\{\bot\}))))\}) \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & [\{True \to P_{err}, False \to P_{normal}\}](\{True\}) \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & \{P_{err}\} \end{array}$$

ce qui est bien le résultat "attendu".

Deuxième cas : si $M \xrightarrow{*}_{\rho_{\varepsilon v}} M'$ *avec* M' *en forme normale et* $M' \neq \{\bot\}$ *alors on a :*

$$\begin{array}{ll} & [\{True \to P_{err}, False \to P_{normal}\}](\{[exn(\bot) \to True](exn(M')), \\ & \qquad [exn(\bot) \to [x \to False](exn(M'))](exn([exn(\bot) \to True](exn(M'))))\}) \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & [\{True \to P_{err}, False \to P_{normal}\}](\{False\}) \\ \xrightarrow{*}_{\rho_{\varepsilon v}} & \{P_{normal}\} \end{array}$$

et donc dans ce cas aussi nous avons bien le comportement souhaité.

Remarque 3.1 *Les réductions au méta niveau permettent d'associer au calcul du* `first` *une implémentation dans un environnement multi-threads. On crée ainsi n processus légers communiquant entre eux et calculant chacun un* u_i*. Dès que l'on a trouvé un* u_{i_0} *dont une forme normale n'est pas* $\{\bot\}$*, cette information est communiquée aux threads calculant les* u_j *pour* $i_0 < j \leqslant n$ *et ces threads peuvent conclure quant à la valeur qu'ils doivent renvoyer puisque le filtrage échoue sans qu'il soit nécessaire d'évaluer le reste des* u_j*.*

Ainsi, la définition

$$u_i = [exn(\bot) \to [exn(\bot) \to [exn(\bot) \to [\ldots](exn(u_1))\ldots](exn(u_{i-3}))](exn(u_{i-2}))](exn(u_{i-1}))$$

bien qu'équivalente, ne permetterait pas cette implémentation efficace dans un environnement multi-threads puisqu'on a besoin de tous *les* u_k *précédents pour calculer* u_i*. Pour implémenter cela à l'aide d'une machine multiprocesseurs à mémoire commune :*

- *il faut créer n threads tel que chacun calcule un* u_i *;*
- *on peut alors stocker l'identifiant (i.e. de type* `pthread_t` *dans POSIX threads) de chacun de ces threads dans un tableau global ;*
- *lorsqu'un thread i obtient une valeur qui n'est pas* $\{\bot\}$*, alors il faut qu'il stoppe tous les threads j* $(j > i)$ *au moyen d'une simple primitive* `pthread_cancel(pid)`*, et qu'il positionne à* $\{\bot\}$ *le résultat de ces threads.*

En contexte distribué, c'est un peu plus délicat et il faut gérer proprement la distribution des threads sur les différentes machines, et les communications ne peuvent plus se faire par mémoire physique partagée : il faut alors utiliser soit un paradigme de type send/recv, soit de type RPC, soit encore de la mémoire partagée virtuelle... Des environnements tels que PM2 du LIP de l'ENS-Lyon permettent de faire cela aisément.

Conclusion

Nous avons introduit un nouveau calcul permettant la gestion des exceptions et la définition dans ce même calcul de stratégies de normalisation. Ce calcul possède des propriétes intéressantes comme la confluence et la strictité qui en font un calcul où "rien de mauvais" ne peut arriver. Tout à la fin de mon stage nous nous sommes demandés si le ρ_ε-calcul pourrait être programmable dans le ρ_{MP}-calcul. Nous ne sommes pas arriver à énoncer un résultat répondant entièrement à la question mais il semblerait que le ρ_ε-calcul soit programmable dans le ρ_{MP}-calcul avec appel par valeur (modulo une légère modification de l'algorithme de filtrage). En d'autres termes il semble possible d'obtenir un système de réécriture simulant le ρ_ε-calcul comme une "bibliothèque" que nous rajouterions au ρ_{MP}-calcul. A la fin de stage plusieurs questions restent ouvertes :

- peut-on construire un système de type pour le ρ_ε-calcul de manière à prouver la forte normalisation du calcul pour les termes bien typés ?
- le ρ_{MP}-calcul avec appel par valeur est-il confluent ?
- peut-on réaliser dans ce ρ_{MP}-calcul avec appel par valeur un travail similaire au λ-calcul de Plotkin ?
- peut-on construire à l'image de l'isomorphisme de Curry-Howard une logique relative au ρ-calcul ?

Toutes ces questions restent pour l'instant sans réponse exacte et vont mener très prochainement à des travaux. Quoiqu'il en soit avec le ρ-cube, on commence à mesurer l'expressivité et la richesse du ρ-calcul. Je conclurai ce rapport en disant encore une fois tout le plaisir que j'ai eu à réaliser ce travail et j'espère très prochainement pouvoir aller plus loin dans ce domaine.

Remerciements

Il me semble important de signaler que j'ai pris un grand plaisir à travailler avec Claude Kirchner et à construire le ρ_ε-calcul. Chaque choix qui devait être fait lors de la construction du calcul a mené à de nombreuses discussions enrichissantes. Je l'en remercie ainsi que Luigi Liquori. Je remercie aussi Raymond Namyst pour l'échange que nous avons eu sur l'implémentation du `first` dans un environnement multi-threads.

Appendix A

Le filtrage

Nous suivons la présentation du filtrage faite dans [Cir00]. La liaison entre paramètres formels et actuels est basée sur le filtrage qui est donc un composant fondamental du ρ-calcul. Nous définissons d'abord les problèmes de filtrage dans un cadre général :

Définition A.1 *Etant donnée une théorie T sur les ρ-termes, une* T-équation de filtrage *est une formule de la forme $t \ll^?_T t'$, où t et t' sont des ρ-termes. Une substitution σ est une solution de l'équation $t \ll^?_T t'$ si $T \models \sigma(t) = t'$. Un* T-système de filtrage *est une conjonction de T-équations de filtrage. Une substitution est une solution d'un T-système de filtrage P si c'est une solution de toutes les T-équations de filtrage. Nous notons par* $\mathbf{F}$ *un T-système de filtrage sans solution. Un T-système de filtrage est appelé* trivial *quand toute substitution est solution du système.*

Nous définissons Solution($\mathcal{S}$) pour un T-système de filtrage $\mathcal{S}$ comme étant la fonction qui retourne l'ensemble de toutes les solutions de $\mathcal{S}$ quand $\mathcal{S}$ n'est pas trivial et $\{\mathbb{ID}\}$, où $\mathbb{ID}$ est la substitution identité, quand $\mathcal{S}$ est trivial.

On appelle *T-filtre* de t à t' une substitution σ qui est une solution d'un problème de filtrage $t \ll^?_T t'$. Si une telle substitution existe, on dit que le terme t *subsume* le terme t'.

Remarquons que si l'algorithme de filtrage échoue (i.e. retourne $\mathbf{F}$) alors la fonction *Solution* retourne l'ensemble vide.

Puisqu'en général on peut considérer des théories arbitraires sur les ρ-termes, le T-filtrage est en général indécidable, même lorsqu'on se restreint à des théories équationnelles du premier ordre. Afin de surmonter ce problème d'indécidabilité, on peut penser à utiliser des contraintes, comme dans la résolution d'ordre supérieur avec contraintes ou dans la déduction avec contraintes.

Nous sommes principalement intéressés par les cas décidables. Parmi les théories décidables, on peut mentionner le filtrage d'ordre supérieur avec motif qui est décidable et unitaire comme conséquence de la décidabilité de l'unification avec motif, le filtrage linéaire d'ordre supérieur, le filtrage d'ordre supérieur qui est décidable jusqu'au quatrième ordre (le problème de décision du cas général étant encore ouvert), beaucoup de théories équationnelles du premier ordre comprenant l'associativité, la commutativité, la distributivité et la majorité de leur combinaisons.

Par exemple, quand la théorie T est vide, la substitution résultant du filtrage syntaxique entre les termes t et t', quand elle existe, est unique et peut être calculée par un algorithme récursif simple donné, par exemple, par G. Huet [Hue76]. Cette substitution peut également être calculée par l'ensemble de règles *SyntacticMatching* où on suppose que le symbole $\wedge$ est associatif et commutatif.

Proposition A.1 *[KK99] La forme normale de tout problème de filtrage $t \ll^?_\emptyset t'$ calculée par les règles SyntacticMatching existe et est unique. Après avoir enlevé de la forme normale toute équation dupliquée et toute équation triviale de la forme $x \ll^?_\emptyset x$, si le système résultant est :*

1. *$\mathbf{F}$, alors il n'y a pas de filtre de t à t' et $Solution(t \ll^?_\emptyset t') = Solution(\mathbf{F}) = \emptyset$,*
2. *de la forme $\bigwedge_{i\in I} x_i \ll^?_\emptyset t_i$ avec $I \neq \emptyset$, alors la substitution $\sigma = \langle x_i/t_i\rangle_{i\in I}$ est l'unique filtre de t à t' et $Solution(t \ll^?_\emptyset t') = Solution(\bigwedge_{i\in I} x_i \ll^?_\emptyset t_i) = \{\sigma\}$,*

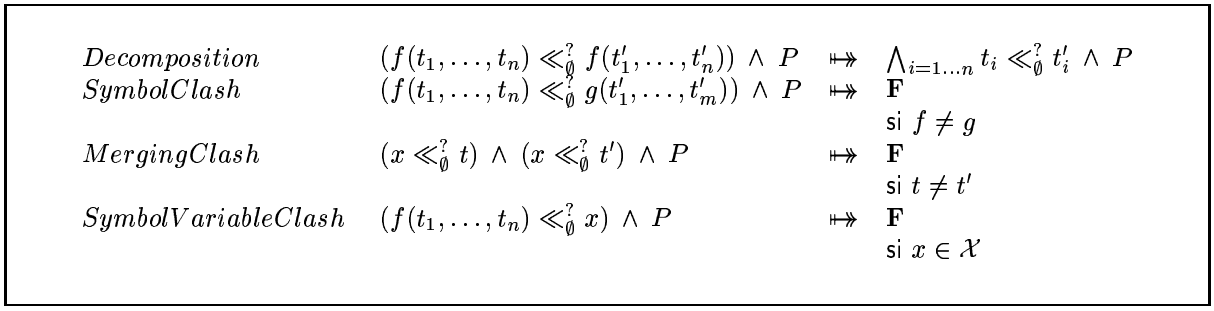


Figure A.1: *SyntacticMatching* - Règles pour le filtrage syntaxique

3. *vide, alors t et t' sont identiques et* $\mathcal{S}olution(t \ll^?_\emptyset t) = \{\mathbb{ID}\}$.

Exemple A.1 *Si nous considérons le problème de filtrage $(f(x,g(x,y)) \ll^?_\emptyset f(a,g(a,b))$, nous appliquons d'abord la règle de filtrage Decomposition et nous obtenons le système avec les deux équations $(x \ll^?_\emptyset a)$ et $(g(x,y) \ll^?_\emptyset g(a,b))$. Quand nous appliquons la même règle de nouveau pour la deuxième équation nous obtenons $(x \ll^?_\emptyset a)$ et $(y \ll^?_\emptyset b)$. Ainsi, l'équation initiale est réduite en $(x \ll^?_\emptyset a) \wedge (x \ll^?_\emptyset a) \wedge (y \ll^?_\emptyset b)$ et donc $\mathcal{S}olution(f(x,g(x,y)) \ll^?_\emptyset f(a,g(a,b)) = \{\langle x/a, y/b\rangle\}$.*

Pour le problème de filtrage $(f(x,x) \ll^?_\emptyset f(a,b))$ nous appliquons, comme avant, la règle Decomposition et nous obtenons le système $(x \ll^?_\emptyset a) \wedge (x \ll^?_\emptyset b)$. Ce dernier système est réduit par la règle de filtrage MergingClash en **F** *et ainsi, $\mathcal{S}olution(f(x,x) \ll^?_\emptyset f(a,b)) = \emptyset$.*

Cet algorithme de filtrage syntaxique peut être étendu d'une façon naturelle quand on suppose qu'un symbole, comme le + par exemple, est commutatif. Dans ce cas-ci, l'ensemble précédent de règles devrait être complété avec :

$$\begin{array}{ll} CommDec & (t_1 + t_2) \ll^? (t'_1 + t'_2) \wedge P \twoheadrightarrow \\ & ((t_1 \ll^? t'_1 \wedge t_2 \ll^? t'_2) \vee (t_1 \ll^? t'_2 \wedge t_2 \ll^? t'_1)) \wedge P \end{array}$$

où la disjonction a les propriétés habituelles.

Il est clair que dans ce dernier cas le nombre de filtres peut être exponentiel par rapport à la taille des membres gauches des équations de filtrage initiales.

Exemple A.2 *Quand on filtre modulo la commutativité un terme comme $x+y$, avec + défini comme étant commutatif, contre le terme $a+b$, la règle CommDec mène à*

$$((x \ll^? a \wedge y \ll^? b) \vee (x \ll^? b \wedge y \ll^? a))$$

et ainsi, $\mathcal{S}olution(x + y \ll^? a + b) = \{\langle x/a, y/b\rangle, \langle x/b, y/a\rangle\}$.

Comme nous l'avons déjà dit, la théorie T est un paramètre du ρ-calcul et ainsi nous notons, par exemple, par $\rho_\emptyset$-calcul le ρ-calcul avec une théorie de filtrage vide (filtrage syntaxique), par ρ_C-calcul le ρ-calcul avec une théorie de filtrage commutative, par ρ_A-calcul le ρ-calcul avec une théorie de filtrage associative, par ρ_{AC}-calcul le ρ-calcul avec une théorie de filtrage associative-commutative.

Appendix B

Les règles d'évaluation du ρ-calcul

On veut s'intéresser aux règles d'évaluation du ρ-calcul. Nous supposons donnée une théorie T sur les ρ-termes pour laquelle le filtrage est décidable. Les règles d'évaluation du ρ_T-calcul décrivent principalement l'application d'un ρ-terme sur un autre ρ-terme et indiquent le comportement des différents opérateurs du calcul quand les arguments sont des ensembles.

B.1 Les règles d'évaluation

B.1.1 L'application d'une règle en tête dans le ρ_T-calcul

L'application d'une règle de réécriture $l \to r$ sur un terme t est décrite par la règle d'évaluation $Fire$ présentée dans la Figure B.1. La règle $Fire$, comme toutes les règles d'évaluation du calcul, peut être appliquée à n'importe quelle position d'un ρ-terme.

$$Fire \quad [l \to r](t) \quad \Longrightarrow \quad \{\sigma_1 r, \ldots, \sigma_n r, \ldots\} \quad \text{avec } \sigma_i \in Solution(l \ll^?_T t)$$

Figure B.1: La règle d'évaluation $Fire$ du ρ_T-calcul

L'idée centrale est que l'application d'une règle de réécriture $l \to r$ à la position de tête d'un terme t, écrite $[l \to r](t)$, consiste à remplacer le terme r par $\{\sigma_1 r, \ldots, \sigma_n r, \ldots\}$. Par conséquent, quand le filtrage échoue, menant à un ensemble vide de substitutions, le résultat de l'application de la règle $Fire$ est l'ensemble vide.

Notons que, comme dans λ-calcul, une application peut toujours être évaluée, mais contrairement au λ-calcul, l'ensemble de résultats peut être vide. Plus généralement, quand on filtre modulo une théorie T, l'ensemble de filtres correspondants peut être vide, un singleton (comme dans la théorie vide), un ensemble fini (comme dans le cas d'une théorie associative-commutative) ou infini).Nous avons ainsi choisi de représenter les résultats de l'application d'une règle de réécriture à un terme par un ensemble. Un ensemble vide signifie que la règle de réécriture $l \to r$ ne s'applique pas sur le terme t dans le sens d'un échec pour le filtrage entre les termes l et t.

B.1.2 Les règles $Congruence$ du ρ_T-calcul

Afin de pousser l'application de règles de réécriture plus profondément dans les termes, nous introduisons les deux règles d'évaluation $Congruence$ présentées dans la Figure B.2.

Nous pouvons simuler les réductions utilisant les règles d'évaluation $Congruence$ pour les applications d'un terme avec un symbole de tête fonctionnel à un terme de la même forme en transformant le terme initial et utilisant la règle d'évaluation $Fire$.

$$
\begin{array}{lllll}
Congruence & [f(u_1,\ldots,u_n)](f(v_1,\ldots,v_n)) & \Longrightarrow & \{f([u_1](v_1),\ldots,[u_n](v_n))\} \\
Congruence\_fail & [f(u_1,\ldots,u_n)](g(v_1,\ldots,v_m)) & \Longrightarrow & \emptyset
\end{array}
$$

Figure B.2: Les règles d'évaluation *Congruence* du ρ_T-calcul

B.1.3 Le traitemenent des ensembles dans le ρ_T-calcul

Les réductions correspondant aux termes contenant des ensembles sont définies par les règles d'évaluation présentées dans la Figure B.3. Ces règles décrivent la propagation des ensembles par rapport aux constructeurs de ρ-termes : les règles *Distrib* et *Batch* pour l'application, $Switch_L$ et $Switch_R$ pour l'abstraction et *OpOnSet* pour les fonctions.

$$
\begin{array}{lllll}
Distrib & [\{u_1,\ldots,u_n\}](v) & \Longrightarrow & \{[u_1](v),\ldots,[u_n](v)\} \\
Batch & [v](\{u_1,\ldots,u_n\}) & \Longrightarrow & \{[v](u_1),\ldots,[v](u_n)\} \\
Switch_L & \{u_1,\ldots,u_n\} \to v & \Longrightarrow & \{u_1 \to v,\ldots,u_n \to v\} \\
Switch_R & u \to \{v_1,\ldots,v_n\} & \Longrightarrow & \{u \to v_1,\ldots,u \to v_n\} \\
OpOnSet & f(v_1,\ldots,\{u_1,\ldots,u_m\},\ldots,v_n) & \Longrightarrow & \{f(v_1,\ldots,u_1,\ldots,v_n),\ldots,f(v_1,\ldots,u_m,\ldots,v_n)\}
\end{array}
$$

Figure B.3: Les règles d'évaluation *Set* du ρ_T-calcul

B.1.4 Aplatir les ensembles dans le ρ_T-calcul

La règle d'évaluation *Flat* présentée dans la Figure B.4 décrit la propagation des ensembles par rapport aux ρ-termes de type ensemble et l'élimination des symboles d'ensemble redondants.

$$
Flat \quad \{u_1,\ldots,\{v_1,\ldots,v_n\},\ldots,u_m\} \Longrightarrow \{u_1,\ldots,v_1,\ldots,v_n,\ldots,u_m\}
$$

Figure B.4: La règle d'évaluation *Flat* du ρ_T-calcul

B.2 La stratégie d'évaluation

La définition générale d'une stratégie est donnée dans [KKV95] mais nous spécialisons cette définition dans le cas du ρ-calcul.

Définition B.1 *Une* stratégie d'évaluation *dans le ρ-calcul est un sous-ensemble de toutes les dérivations possibles.*

La stratégie $\mathcal{S}$ guidant l'application des règles d'évaluation du ρ_T-calcul peut être cruciale pour obtenir les bonnes propriétés pour le calcul. Dans une première étape, la propriété principale analysée est la confluence

du calcul et nous verrons que si la règle *Fire* est appliquée sans aucune restriction et à n'importe quelle position d'un ρ-terme alors le calcul n'est pas confluent.

Appendix C

Sur la confluence du $\rho_{\mathcal{E}}$-calcul

Nous reprenons le cheminement de la preuve de confluence du $\rho_{\emptyset}$-calcul présentée dans [Cir00]. Mais les deux conditions suivantes sont plus délicates à obtenir et nécessitent une analyse plus détaillée des termes sur lesquels *Fire et Exn* sont appliquées : *le terme t est tel que si le filtrage $l \ll^?_{\emptyset} t$ échoue alors pour tout terme t' obtenu en instanciant ou ρ-réduisant t le filtrage $l \ll^?_{\emptyset} t'$ échoue* et *le terme t est tel que t ne peut pas être réduit en un échec ou en un ensemble.* Ces conditions sont difficiles à obtenir puisque :

1. il existe un plus grand nombre de possibilités pour obtenir un échec ;
2. il faut gérer le comportement très précis que l'on a donné à l'ensemble vide (il ne peut pas être appliqué et ne peut pas être instancié pendant le fitrage ...)
3. contrairement à l'ensemble vide dans le $\rho_{\emptyset}$-calcul , $\perp$ peut être présent dans un terme même si dans celui-ci la propagation de l'échec a eu lieu (présence de sous-termes de la forme $exn(\perp)$) ;
4. si nous appliquons $Fire$ sur $[l \to r](t)$ et que t n'est pas suffisament réduit par rapport à la règle Exn alors un échec de filtrage peut être dû à un "clash" entre un symbole fonctionnel de l et le symbole exn considéré. Dans ce cas-là, il est possible que cet exn disparaisse par ρ_{ε}-réductions (application de la règle Exn) et donc que la propriété énoncée ci-dessus n'est pas validée ;

Nous n'allons pas montrer la confluence du ρ_{ε}-calcul muni de la stratégie d'évaluation *ConfStrat* mais nous donnons la preuve de la confluence du ρ_{ε}-calcul avec les règles $Fire$ et Exn transformées en règles conditionnelles. Les conditions de ces règles sont exactement les conditions de la définition de *ConfStrat* et donc la preuve reste identique. L'idée générale de la preuve est la suivante : nous considérons les règles d'évaluation *Fire et Exn* que l'on transforme en règles conditionnelles avec les conditions définies dans *ConfStrat*. Nous définissons la relation *FireCong* comme la relation induite par les règles $Fire, Congruence, CongruenceFail$. Les autres règles induisent ainsi la relation *Calculus*. Nous dénotons par $\longrightarrow_F$ la relation *FireCong* et par $\longrightarrow_C$ la relation *Calculus*. L'idée est de prouver la confluence de la relation $\xrightarrow{*}_C \longrightarrow_F \xrightarrow{*}_C$. Donc nous allons prouver la forte confluence de $\longrightarrow_C$ (en introduisant une notion de réduction parallèle), la confluence et terminaison de $\longrightarrow_F$ (conflence locale et interprétation polynomiale) et enfin la compatibilité des deux relations (i.e. le lemme de Yokoutchi).

C.1 Notions préliminaires

Dans les exemples de la section 2.5.1, nous avons vu que l'application de la règle $Fire$ à un terme non suffisament réduit peut mener à des résultats non-confluents. Nous voudrions définir des conditions sur la structure des termes, qui nous évitent de telles situations. Nous souhaiterions de plus que ces conditions soient efficacement implémentables. Dans le même temps, nous souhaitons remplacer la condition de l'application de la règle Exn ($\{x\}\downarrow \neq \{\perp\}$) par une condition structurelle sur x.

C.1.1 Termes ρ_ε-préfiltables

Tout d'abord, nous introduisons une définition permettant de garantir la "cohérence" du filtrage en imposant des conditions sur la structure des termes l et t. Ce que l'on veut, c'est préserver l'échec de filtrage par ρ_ε-réductions. Pour cela on se donne les notions suivantes :

Définition C.1 *On dit que t est ε_{exn}-réduit par rapport à l si :*

$$\forall p \in \mathcal{P}os(t) \cap \mathcal{F}_\varepsilon\mathcal{P}os(l), \Big(t(p) = exn \Rightarrow t(p.0) = \bot\Big)$$

Lorsque cela sera évident dans le contexte, par abus de langage, nous dirons t est ε_{exn}-réduit sans préciser par rapport à quel autre terme.

Définition C.2 *On dit que l ε-subsume faiblement t si :*

$$\forall p\Big(p \in \mathcal{F}_\varepsilon\mathcal{P}os(l) \cap \mathcal{P}os(t) \Rightarrow t(p) \in \mathcal{F}_\varepsilon\Big)$$

Remarque C.1 *Si l ε-subsume faiblement t alors le filtrage $(l \ll^?_\emptyset t)$ ne peut pas échouer à cause de l'application des règles de filtrage $SymbolVariableClash$. De plus, on peut remarquer que si l ε-subsume faiblement t, à toute position non-ε-fonctionelle de t correspond dans l une position variable.*

Définition C.3 *Soient l du ε-premier ordre et t ρ_ε-terme. On dit que (l,t) est ρ_ε-préfiltrable si :*

1. *t est clos et du ε-premier ordre ou*
2. *(a) l est linéaire ;*
   (b) t est ε_{exn}-réduit par rapport à l ;
   (c) l ε-subsume faiblement t.
   (d) $\mathcal{V}\mathcal{P}os(l) \cap \mathcal{P}os_{\{\}}(t) = \emptyset$

Par abus de langage, nous dirons que l et t sont ρ_ε-préfiltables à la place de (l,t) ρ_ε-préfiltrable.

Remarque C.2 *Il est clair que pour tout terme l de la forme $f(l_1,\ldots,l_n)$ du ε-premier ordre et tout ρ_ε-terme t de la forme $t = g(t_1,\ldots,t_m)$ tels que l,t soient ρ_ε-préfiltables, les termes l_i,t_i sont ρ_ε-préfiltables pour tout $1 \leqslant i \leqslant min(m,n)$.*

Remarque C.3 *Si l est linéaire, $(l \ll^?_\emptyset t)$ ne peut pas échouer à cause de la règle $MergingClash$ et donc si l et t sont ρ_ε-préfiltables, par le cas (2) de la définition on en conclut grâce à la Remarque C.1 qu'un échec de filtrage peut seulement être dû à la règle $SymbolClash$.*

Lemme C.1 *Etant donnés les termes ρ_ε-préfiltables l,t si le filtrage $(l \ll^?_\emptyset t)$ échoue alors pour tout terme t' obtenu en ρ_ε-réduisant ou instanciant le terme t, le filtrage $(l \ll^?_\emptyset t')$ échoue.*

Preuve : Si t est clos et du ρ_ε-premier ordre, alors t en forme normale et donc $t' = t$ et le résultat est immédiat.

Raisonnons par induction sur la structure du terme t. D'après la Remarque C.3, l'échec de filtrage ne peut-être causé que par la règle $SymbolClash$ et donc $t = f(t_1,\ldots,t_n)$ et $l = g(l_1,\ldots,l_m)$ avec l du ε-premier ordre .

1. si $f \neq g$
   (a) si $f = \bot$ alors nécessairement $t' = t$ et le résultat est immédiat ;
   (b) si $f = exn$ c'est-à-dire $t = exn(t_1)$ comme t est ε_{exn}-réduit, t' est forcément de la même forme que t et donc $t' = exn(t'_1)$ et donc le filtrage $(l \ll^?_\emptyset t')$ échoue aussi ;
   (c) si $f \in \mathcal{F}$ alors t' est forcément de la même forme que t ou est un ensemble. Dans les deux cas, il y a échec du filtrage.

2. si $f = g$ où $f \in \mathcal{F}_\varepsilon$
   Nécessairement $f \neq \perp$ puisque sinon $g = \perp$ et par hypothèse g est du ρ_ε-premier ordre. Les l_i t_i sont ρ_ε-préfiltables par la Remarque C.2 et $t' = f(t'_1, \ldots, t'_n)$ comme t est ε_{exn}-réduit, avec t'_i obtenu en ρ_ε-réduisant ou instanciant t_i. Puisque $(l \ll^?_\emptyset t)$ échoue, un des filtrages $(l_i \ll^?_\emptyset t_i)$ échoue et par induction le filtrage correspondant $(l_i \ll^?_\emptyset t'_i)$ et donc $(l \ll^?_\emptyset t')$ échoue.

□

C.1.2 Termes ρ_ε-safes

La notion de termes ρ_ε-safes est introduite pour expliciter des conditions structurelles sur t pour appliquer la règle *Fire* uniquement si :

1. t ne se réduit pas en $\{\}$;
2. t ne se réduit pas $\{\perp\}$ et n'est pas égal à $\perp$;
3. t ne se réduit pas en un ensemble ayant plus d'un élément.

La dernière condition est simple à obtenir puisque il suffit d'imposer que t ne contienne aucun ensemble ayant plus d'un élément. La première et la deuxième condition sont finement liées. En effet, pour que t ne se réduise pas en $\{\perp\}$, il faut imposer en autre qu'il ne contienne aucun terme de la forme $[u](v)$ où u et v pourraient se réduire en $\{\}$, qu'aucune application ne va mener à l'échec et que $\perp$ n'est présent que "sous" un *exn*. Il est nécesaire que :

1. pour tout sous-terme de la forme $[u](v)$, u est une abstraction ;
2. pour tout sous-terme de la forme $[u \to w](v)$, u subsume v ;
3. $\forall p \Big(t(p) = \perp \Rightarrow (p = p'i) \wedge (t(p') = exn) \Big)$ (donc $t \neq \perp$)

Mais ceci n'est pas suffisant car les règles *Distrib*, *Batch* peuvent mener à $\{\perp\}$. Pour ce qui est de la règle *Distrib*, la première condition empêche tout sous-terme de la forme $[u](v)$ où $u \xrightarrow{*}_{\rho_\varepsilon} \{\}$ puisque u ne peut être qu'une abstraction. Dans le ρ_ε-calcul, trois règles peuvent donner un résultat se réduisant en $\{\}$: *Fire*, *Flat* et *Exn*. La condition délicate à imposer est celle pour la règle *Fire*. Pour la règle *Flat*, il suffit que l'on ait pour t : $\forall p \in \mathcal{P}os(t), \Big((t(p) = \{\}) \wedge (p = p'i) \Rightarrow t(p') \neq \{\} \Big)$. Pour la règle *Exn* : $\forall p \in \mathcal{P}os(t), \Big((t(p) = \{\}) \wedge (p = p'i) \Rightarrow t(p') = exn \Big)$. Si ces deux conditions sont vérifiées, on a dans t aucun élément de la forme $\{\ldots\{\}\ldots\}$ et aucun élément de la forme exn($\{\ldots\{\}\ldots\}$). On suppose ces deux conditions vérifiées pour déterminer les conditions à imposer pour que *Fire* ne renvoie pas un résultat se réécrivant en $\{\}$. *Fire* appliquée à $[l \to r](t)$ renvoie un terme se réduisant en $\{\}$ si $r\langle\langle \mathcal{S}olution(l \ll^?_\emptyset t) \rangle\rangle \xrightarrow{*}_{\rho_\varepsilon} \{\}$ c'est-à-dire $\{\sigma r\} = \{\ldots\{\}\ldots\}$ et donc vu les hypothèses que l'on s'est données, cette condition peut s'exprimer par : $r = \{\}$ ou $r = x$ et x est instancié par un terme se réduisant en $\{\}$ ou $r = \{x\}$ et x est instancié par un terme se réduisant en $\{\}$. La condition x instancié en $\{\}$ n'est pas triviale à déterminer. Dans un premier temps, nous avons cherché une condition testant cette potentielle instanciation, mais très vite nous nous sommes rendu compte que celle-ci était :

- très lourde à implémenter ;
- ne correspondait pas à "l'esprit" du calcul avec appel par valeur dans le sens où on montre qu'il suffit d'imposer que mon terme soit suffisament évalué pour tester *facilement* si *Fire* est applicable.

Donnons-nous un peu de vocabulaire.

Définition C.4 *On dit que t est $\varepsilon_\perp$-réduit si :*

$$\forall p \in \mathcal{P}os(t), \Big(t(p) = \perp \Rightarrow (p = p'i) \wedge (t(p') = exn) \Big)$$

Définition C.5 *On dit que t est $\{\}$-réduit si :*

$$\forall p \in \mathcal{P}os(t), \Big(t(p) = \{\} \Rightarrow (p = p'i) \wedge ((t(p') = \rightarrow) \vee (t(p') \in \mathcal{F}))\Big)$$

On notera que si t est $\{\}$-réduit, t ne contient aucun sous-terme de la forme $[u](\{\})$.

Définition C.6 *On dit que t est $\{\}$-adéquate si pour tout sous-terme de la forme $[u \rightarrow v](w)$:*

$$(v \neq \{\}) \wedge (v \neq x) \wedge (v \neq \{x\})$$

Définition C.7 *On dit que t est $\varepsilon_{\{\}}$-réduit si t est $\{\}$-réduit et si t est $\{\}$-adéquate.*

Remarque C.4 *Si t est $\varepsilon_{\{\}}$-réduit, t ne contient des sous-termes égaux à l'ensemble vide uniquement si dans ce cas-là leur père dans l'arbre correspondant au terme t est un symbole fonctionnel.*

Nous sommes maintenant en mesure de définir la notion de termes ρ_ε-safes.

Définition C.8 *On dit que t est ρ_ε-safe s'il satisfait les conditions suivantes :*

1. *t ne contient pas d'ensemble ayant plus d'un élément ;*
2. *t est $\varepsilon_\perp$-réduit ;*
3. *t est $\varepsilon_{\{\}}$-réduit ;*
4. *pour tout sous-terme de la forme $[u](v)$, u est une abstraction ;*
5. *pour tout sous-terme de la forme $[u \rightarrow v](w)$, u subsume w ;*

Si tous les termes du codomaine d'une substitution σ sont ρ_ε-safes, nous disons que σ est ρ_ε-safe.

Lemme C.2 *Soit t un terme ρ_ε-safe. Alors t ne peut pas être égal à $\perp$, t ne peut se réduire ni en $\{\}$, ni en $\{\perp\}$ et ni en un ensemble ayant plus d'un élément.*

Preuve : Soit t un terme ρ_ε-safe. Puisque le fitrage est syntaxique, *Fire* ne peut pas mener à un ensemble ayant plus d'un élément sans que le terme de départ n'en contienne un; ce qui prouve que t ne peut se réduire en un ensemble ayant plus d'un élément.

On ne peut pas faire apparaître de nouveau symbole $\perp$ car l'apparition de ce symbole peut être dû :

1. aux règles de propagation de $\perp$ présentées dans la Figure 2.2, mais dans ce cas-là la condition (2) de la définition de ρ_ε-safe ne serait pas vérifiée pour t ;
2. à la règle *Fire* mais ceci est impossible puisque pour toute application $[u \rightarrow v](w)$, u subsume w ;
3. à la règle *CongruenceFail* mais ceci est impossible par la condition (4) de la définition de t ρ_ε-safe ;
4. à la règle *Distrib* mais ceci est impossible car sinon t contiendrait un terme de la forme $[u](v)$ avec $u \xrightarrow{*}_{\rho_\varepsilon} \{\}$ et donc t contiendrait une application $[u](v)$ où u serait une application ce qui est absurde par la condition (4) de la définition de t ρ_ε-safe ;
5. à la règle *Batch* mais cela signifirait que dans t il y a un terme de la forme $[u](v)$ avec $v \xrightarrow{*}_{\rho_\varepsilon} \{\}$ mais ceci est impossible. En effet, seules les règles *Flat*, *Exn et Fire* peuvent donner $\{\ldots\{\}\ldots\}$. Mais si *Flat* ou *Exn* sont appliquées et donnent un terme se réduisant en $\{\}$ c'est que t n'est pas $\{\}$-réduit, ce qui est absurde. De plus, si *Fire* appliquée à $[l \rightarrow r](s)$ donne comme résultat $\{\ldots\{\}\ldots\}$ c'est que $\{\sigma(r)\} = \{\ldots\{\}\ldots\}$ soit $\sigma(r) = \{\ldots\{\}\ldots\}$ et vu les conditions imposées (t $\varepsilon_{\{\}}$-réduit) à t c'est équivalent à $r = \{\}$ ou $r = x$ et x est instancié par $\{\}$ ou encore $r = \{x\}$ et x est instancié par $\{\}$. Mais aucun des cas ne peut se présenter puisque t est $\varepsilon_{\{\}}$-réduit.

On vient de montrer dans (5) qu' aucun sous-terme de t ne peut se réduire en $\{\}$ et donc a fortiori t ne peut se réduire en $\{\}$. Notons de plus que t ne peut pas être égal à $\perp$ puisque t est $\varepsilon_\perp$-réduit. Ce qui achève la preuve. $\square$

Définition C.9 *Le couple de ρ_ε-termes (l,t) est* ρ_ε-calculable *si les conditions suivantes sont satisfaites :*

- *les termes l,t sont ρ_ε-préfiltables;*
- *le terme t est ρ_ε-safe.*

Par abus de langage, nous dirons que l et t sont ρ_ε-calculables à la place de (l,t) ρ_ε-calculable.

Proposition C.1 *Etant donnés les termes ρ_ε-calculables l,t, si $\sigma l = t$, alors σ est ρ_ε-safe.*

Proposition C.2 *Etant donnés des termes l,t ρ_ε-calculables et une substitution σ ρ_ε-safe, les termes $l,\sigma t$ sont ρ_ε-calculables.*

Remarque C.5 *Pour obtenir cette proposition, il est important de ne pas avoir imposer à t d'être totalement réduit par rapport à Exn. Par exemple, soit $l = x$, $t = f(y)$ et $\sigma = \langle y/exn(f(a))\rangle$. l,t sont ρ_ε-calculables et σ est ρ_ε-safe. Pourtant $l,\sigma t$ ne sont pas ρ_ε-calculables.*

C.1.3 Une condition structurelle pour l'application de la règle Exn

Pour définir la notion de terme ρ_ε-safe, nous avons réfléchi à des conditons structurelles à imposer pour qu'un terme ait entre autre la propriété de ne pas se réduire en $\{\perp\}$ (ou être égal à $\perp$). Nous pourrions adapter ces conditions à la règle Exn de manière à imposer uniquement que t ne se réduise pas en $\{\perp\}$ (ou $\perp$) et soit clos. Mais le sens que nous avons donné à exn ne nous amène pas à effectuer de tels choix. En effet, exn a été introduit pour le rattrapage de l'échec. L'idée est donc d'évaluer x puis, selon qu'il est égal à $\perp$ ou non d'appliquer la règle Exn. Lorsque l'on s'intéresse à un terme $exn(t)$, on va commencer par calculer une forme normale de t pour les règles de déduction. Puis le test de réductibilité en $\{\perp\}$ ou $\perp$ est facilement réalisable puisque il suffit d'imposer à t d'être $\varepsilon_\perp$-réduit.

Définition C.10 *On dit que t est Exn-réductible si t vérifie les conditions suivantes :*

- *t est clos ;*
- *t ne contient aucune application ;*
- *t est $\varepsilon_\perp$-réduit.*

On définit ainsi une variante de la règle Exn appelée Exn_c :

$$Exn_c \quad exn(t) \implies \{t\} \quad \text{si } t \text{ est } Exn-\text{réductible}$$

C.1.4 Relations induites par les règles d'évaluation

On voudrait comme dans [Cir00], décomposer l'ensemble des règles d'évaluation du ρ_ε-calcul en deux ensembles afin que l'évaluation se comporte comme dans la déduction modulo [DHK98]. On scinde donc les règles en deux ensembles :

- les règles de déduction : $Fire_c$, $Congruence$, $CongruenceFail$;
- les règles de calcul : $Distrib$, $Batch$, $SwitchR$, $OpOnSet$, $Flat$, $AppBottomOperand$, $AppBottomOperator$, $RuleBottomR$, $OpOnSet$, $FlatBottom$, Exn_c.

Définition C.11 *Nous considérons la relation sur l'ensemble des termes du ρ_ε-calcul appelée Calculus induite par les règles de calcul.*

Les relations suivantes sont induites par la relation Calculus :

$\longrightarrow_C$ *est la fermeture compatible de la relation Calculus,*

$\xrightarrow{*}_C$ *est la fermeture réflexive, transitive de* $\longrightarrow_C$ *(la réduction engendrée par Set),*

$\xleftrightarrow{*}_C$ *est la relation d'équivalence engendrée par* $\xrightarrow{*}_C$.

L'application de la règle d'évaluation *Fire* est guidée par une stratégie qui tient compte des conditions présentées dans la section précédente et qui peut être exprimée explicitement en transformant les règles *Fire* et *Exn* en règle conditionnelle :

$$\begin{array}{lll} Fire_c \quad [l \to r](t) & \Longrightarrow & r\langle\!\langle Solution(l \ll^?_\emptyset t)\rangle\!\rangle \\ & & \text{si } l, t \text{ sont } \rho_\varepsilon - \text{calculables} \\ Exn_c \quad exn(t) & \Longrightarrow & \{t\} \\ & & \text{si } t \text{ est } Exn - \text{réductible} \end{array}$$

Définition C.12 *Nous considérons la relation appelée FireCong sur les termes du ρ_ε-calcul induite par la règle d'évaluation $Fire_c$ et les règles d'évaluation Congruence et CongruenceFail.*

Nous considérons les relations suivantes induites par les relations FireCong et Calculus

$\longrightarrow_F$ *est la fermeture compatible de la relation FireCong,*

$\xrightarrow{*}_F$ *est la fermeture réflexive, transitive de* $\longrightarrow_F$,

$\longrightarrow_{F/C}$ *est la relation* $\longrightarrow_F$ *modulo la relation* $\xleftrightarrow{*}_C$ *définie de façon standard ([ASU72]) : étant donnés deux ρ-termes u, v nous avons $u \longrightarrow_{F/C} v$ ssi il existe deux ρ-termes u', v' tels que $u \xleftrightarrow{*}_C u'$, $u' \longrightarrow_F v'$ et $v \xleftrightarrow{*}_C v'$,*

$\xrightarrow{*}_{F/C}$ *est la fermeture réflexive, transitive de* $\longrightarrow_{F/C}$.

C.2 Propriétés des relations C

Nous allons montrer que $\longrightarrow_C$ termine et est confluente. Pour cela, nous allons montrer la terminaison et la confluence locale puis nous conclurons à l'aide du lemme de Newman sur la confluence de $\longrightarrow_C$.

Lemme C.3 *La relation* $\longrightarrow_C$ *termine.*

Preuve : Nous utilisons les interprétations polynômiales suivantes :

$P(c) = 2$ pour c constante de $\mathcal{F}_0$

$P(exn(t)) = P(t) + 3$

$P(\bot) = 2$

$P(f(t_1, \ldots, t_n)) = \Pi_{i=1}^n P(t_i)$ pour $f \in \mathcal{F}$

$P(\{u_1, \ldots, u_n\}) = \Sigma_{i=1}^n P(u_i) + 2$

$P(u \to v) = P([u](v)) = 4P(u)P(v)$.

On utilise l'ordre standard sur les entiers naturels. La vérification des inégalités pour toutes les règles concernées est immédiate.

□

Lemme C.4 *La relation $\longrightarrow_C$ est localement confluente.*

Preuve : On montre que toutes les paires critiques sont convergentes et l'on conclut quant à la locale confluence par le lemme de Knuth-Bendix. On a :

1. la règle *Flat* a une paire critique avec elle même :

$$\langle \{u_1, \ldots, u_n, \{v_1, \ldots, v_m\}, \ldots, t\}, \{\{u_1, \ldots, u_n\}, v_1, \ldots, v_m, \ldots, t\} \rangle$$

2. la règle *OpOnSet* a une paire critique avec elle même :

$$\langle \{f(u_1, \ldots, \{v_1, \ldots, v_m\}), \ldots, f(u_n, \ldots, \{v_1, \ldots, v_m\})\},$$
$$\{f(\{u_1, \ldots, u_n\}, \ldots, v_1), \ldots, f(\{u_1, \ldots, u_n\}, \ldots, v_m)\} \rangle$$

3. les règles *Distrib* et *Batch* mènent à une paire critique convergente :

$$\langle \{[u_1](\{v_1, \ldots, v_m\}), \ldots, [u_n](\{v_1, \ldots, v_m\})\}, \{[\{u_1, \ldots, u_n\}](v_1), \ldots, [\{u_1, \ldots, u_n\}](v_m)\} \rangle$$

4. les règles *Distrib* et *Flat* mènent à une paire critique convergente :

$$\langle [\{u_1, \ldots, v_1, \ldots, v_m, \ldots, u_n\}](v), \{[u_1](v), \ldots, [\{v_1, \ldots, v_m\}](v), \ldots, [u_n](v)\} \rangle$$

5. les règles *Distrib* et *FlatBottom* mènent à une paire critique convergente :

$$\langle \{[u_1](v), \ldots, [u_k](v), [\bot](v), [u_{k+1}](v), \ldots, [u_n](v)\}, [\{u_1, \ldots, u_k, u_{k+1}, \ldots, u_n\}](v) \rangle$$

6. les règles *Batch* et *Flat* mènent à une paire critique convergente :

$$\langle [v](\{u_1, \ldots, v_1, \ldots, v_m, \ldots, u_n\}), \{[v](u_1), \ldots, [v](\{v_1, \ldots, v_m\}), \ldots, [v](u_n)\} \rangle$$

7. les règles *Batch* et *FlatBottom* mènent à une paire critique convergente :

$$\langle \{[v](u_1), \ldots, [v](u_k), [v](\bot), [v](u_{k+1}), \ldots, [v](u_n)\}, [v](\{u_1, \ldots, u_k, u_{k+1}, \ldots, u_n\}) \rangle$$

8. les règles *OpOnSet* et *Flat* mènent à une paire critique convergente :

$$\langle f(\{u_1, \ldots, u_n, \ldots, v_m\}), \{f(\{u_1, \ldots, u_n\}), \ldots, f(v_m)\} \rangle$$

9. les règles *OpOnSet* et *FlatBottom* mènent à une paire critique convergente :

$$\langle \{f(u_1, \ldots, v_1, \ldots, u_n), \ldots, f(u_1, \ldots, v_l, \ldots, u_n), f(u_1, \ldots, \bot, \ldots, u_n), f(u_1, \ldots, v_{l+1}, \ldots, u_n), \ldots\},$$
$$f(u_1, \ldots, \{v_1, \ldots, v_l, v_{l+1}, \ldots, v_m\}, \ldots, u_n) \rangle$$

10. les règles *SwitchR* et *Flat* mènent à une paire critique convergente :

$$\langle u \to \{v_1, \ldots, v_n, \ldots, w_m\}, \{u \to \{v_1, \ldots, v_n\}, \ldots, u \to w_m\} \rangle$$

11. les règles *SwitchR* et *FlatBottom* mènent à une paire critique convergente :

$$\langle u \to \{v_1, \ldots, v_k, v_{k+1}, \ldots, v_n\}, \{u \to v_1, \ldots, u \to v_k, u \to \bot, u \to v_{k+1}, \ldots, u \to v_n\}\rangle$$

12. les règles *Batch* et *AppBottomOperator* mènent à une paire critique convergente :

$$\langle \{[\bot](v_1), \ldots, [\bot](v_n)\}, \{\bot\}\rangle$$

13. les règles *OpOnSet* et *OpOnBottom* mènent à une paire critique convergente :

$$\langle \{f(u_1, \ldots, \bot, \ldots, v_1, \ldots, u_n), \ldots, f(u_1, \ldots, \bot, \ldots, v_m, \ldots, u_n)\}, \{\bot\}\rangle$$

14. les règles *Flat* et *FlatBottom* mènent à une paire critique convergente :

$$\langle \{u_1, \ldots, \{v_1, \ldots, v_k, v_{k+1}, \ldots, v_m\}, \ldots, u_n\}, \{u_1, \ldots, v_1, \ldots, v_k, \bot, v_{k+1}, \ldots, v_m, \ldots, u_n\}\rangle$$

15. les règles *Distrib* et *AppBottomOperand* mènent à une paire critique convergente :

$$\langle \{[u_1](\bot), \ldots, [u_n](\bot)\}, \{\bot\}\rangle$$

□

Lemme C.5 *La relation $\longrightarrow_C$ est confluente.*

Preuve : Les lemmes C.3 et C.4 nous permettent d'appliquer le lemme de Newman et de conclure. □

C.3 Propriétés des relations *FireCong*

La relation $\longrightarrow_F$ est trivialement non-terminante comme montré par le contre exemple classique $[\omega_{\rho_\varepsilon}](\omega_{\rho_\varepsilon})$ où $\omega_{\rho_\varepsilon} = x \to [x](x)$. Ce terme se réduit en une étape en lui-même et on obtient ainsi une chaîne de réduction infinie. Dans cette section nous montrerons que la relation $\longrightarrow_F$ est confluente et pour cela nous nous inspirerons de la preuve de confluence du $\rho_\emptyset$-calcul donné dans [Cir00].

On commence par rappeler le Lemme suivant :

Lemme C.6 *[Bar84] Etant données une relation $\longrightarrow$ et sa fermeture transitive $\xrightarrow{*}$. Si $\longrightarrow$ est fortement confluente alors $\xrightarrow{*}$ est fortement confluente.*

On introduit comme dans la preuve de confluence du λ-calcul une notion de réduction parallèle comme celle de *Tait & Martin-Löf.*

Définition C.13 *La relation $\longrightarrow_{F_\parallel}$ est définie par les règles suivantes :*

1. $t \longrightarrow_{F_\parallel} t$,
2. $u_i \longrightarrow_{F_\parallel} u'_i$, $1 \leqslant i \leqslant n \quad \Rightarrow \quad \{u_1, \ldots, u_n\} \longrightarrow_{F_\parallel} \{u'_1, \ldots, u'_n\}$,
3. $u_i \longrightarrow_{F_\parallel} u'_i$, $1 \leqslant i \leqslant n \quad \Rightarrow \quad f(u_1, \ldots, u_n) \longrightarrow_{F_\parallel} f(u'_1, \ldots, u'_n)$
4. $u \longrightarrow_{F_\parallel} u'$, $v \longrightarrow_{F_\parallel} v' \quad \Rightarrow \quad u \to v \longrightarrow_{F_\parallel} u' \to v'$
5. $u \longrightarrow_{F_\parallel} u'$, $v \longrightarrow_{F_\parallel} v' \quad \Rightarrow \quad [u](v) \longrightarrow_{F_\parallel} [u'](v')$,
6. $l \longrightarrow_{F_\parallel} l'$, $t \longrightarrow_{F_\parallel} t'$, $r \longrightarrow_{F_\parallel} r' \Rightarrow$
   $[l \to r](t) \longrightarrow_{F_\parallel} r'\langle\!\langle Solution(l' \ll^?_\emptyset t')\rangle\!\rangle$ *(Fire$_c$)*
   si l, t sont ρ_ε-calculables
7. $u_i \longrightarrow_{F_\parallel} u'_i$, $v_i \longrightarrow_{F_\parallel} v'_i$, $1 \leqslant i \leqslant n \Rightarrow$
   $[f(u_1, \ldots, u_n)](f(v_1, \ldots, v_n)) \longrightarrow_{F_\parallel} \{f([u'_1](v'_1), \ldots, [u'_n](v'_n))\}$, *(Congruence)*

8. $u_i \longrightarrow_{F_\parallel} u'_i$, $v_i \longrightarrow_{F_\parallel} v'_i$, $1 \leqslant i \leqslant n \Rightarrow$
$[f(u_1, \ldots, u_n)](g(v_1, \ldots, v_m)) \longrightarrow_{F_\parallel} \{\bot\}$ *(CongruenceFail)*
si $f \neq g$

D'une façon similaire à la relation $\longrightarrow_F$, *la relation* $\longrightarrow_{F_\parallel}$ *modulo la relation* $\overset{*}{\longleftrightarrow}_C$ *est notée* $\longrightarrow_{F_\parallel/C}$ *et la fermeture réflexive et transitive de* $\longrightarrow_{F_\parallel/C}$ *est notée* $\overset{*}{\longrightarrow}_{F_\parallel/C}$.

Nous allons prouver la confluence forte de la relation $\longrightarrow_{F_\parallel}$. Puis, nous allons prouvé que $\overset{*}{\longrightarrow}_F$ est la fermeture transitive de $\longrightarrow_{F_\parallel}$ et nous en déduirons la confluence forte de la relation $\overset{*}{\longrightarrow}_F$. Etant donné que tout membre gauche d'une abstraction est du ε premier ordre, dans le point (6) de la définition ci-dessus, si on applique $Fire_c$ à $[l \to r](t)$ pour le terme l' tel que $l \longrightarrow_{F_\parallel} l'$ nous avons $l' = l$.

Lemme C.7 *Etant donnés les* ρ_ε*-termes* l, t, l', t' *tels que* $l \longrightarrow_F l'$ *et* $t \longrightarrow_F t'$. *Alors,*

1. *si* l, t *sont* ρ_ε*-préfiltables alors* l', t' *sont* ρ_ε*-préfiltables;*
2. *si* t *est* ρ_ε*-safe alors* t' *est* ρ_ε*-safe;*
3. *si* l, t *sont* ρ_ε*-calculables alors* l', t' *sont* ρ_ε*-calculables.*

Preuve : On montre successivement les trois points du lemme.

Preuve de (1) **:** Soient l, t ρ_ε-préfiltables montrons que l', t' sont ρ_ε-préfiltables. Par définition du ρ_ε-calcul, l est du ε-premier ordre et donc $l = l'$.
$1^{er} cas$
Si t est clos et du ε-premier ordre $t' = t$ et le résultat est immédiat.
$2^{eme} cas$
Si l est linéaire, ε-subsume faiblement t et si t est ε_{exn}-réduit.
Il faut montrer que l, t' sont ρ_ε-préfiltables c'est-à dire que :

- l est linéaire : par hypothèse ;
- t' est ε_{exn}-réduit : c'est évident puisque si on réalise une réduction en utilisant $\longrightarrow_F$ à une position p, alors à cette position on a une application et donc dans l à la position correspondante on a une variable. Donc cette position n'appartient pas à $\mathcal{F}_\varepsilon\mathcal{P}os(t)$ et donc t' est ε_{exn}-réduit.
- l ε-subsume faiblement t' : par définition de $\longrightarrow_F$. Puisque l, t sont ρ_ε-préfiltables, alors l ε-subsume faiblement t et donc toute position ε-fonctionnelle du terme l est une position ε-fonctionnelle du terme t ou n'est pas une position de t. Soit u et u' tels que $u \longrightarrow_F u'$. Remarquons que le symbole de tête d'un terme u n'est pas le même que celui de u' seulement si u est une application. Dans ce cas-là, le symbole de tête de u' est un ensemble. Notons p la position à laquelle a été effectuée la réduction. $t_{|p}$ est donc une application et $l_{|p}$ est une variable et donc toute position ε-fonctionnelle de l existant dans t existe dans t' et n'a pas été modifiée. Donc l ε-subsume faiblement t'.

□

Preuve de (2) **:** Soit t un terme ρ_ε-safe. Montrons que si t' est définit par $t \longrightarrow_F t'$ alors t' est ρ_ε-safe. On va montrer successivement les 5 points de la définition.

1. même raisonnement que dans le Lemme C.2 ;
2. même raisonnement que dans le Lemme C.2 ;
3. il faut montrer que pour tout sous-terme de t' de la forme $[u \to v](w)$
   (a) t est $\{\}$-réduit par le même raisonnement que dans le Lemme C.2 ;
   (b) $v \neq \{\}$: sinon dans ce cas soit $v = \{\}$ dans t (absurde), ou alors t contenait un terme qui pouvait se réduire en $\{\}$ (absurde)

(c) $v \neq x$: sinon dans ce cas soit $v = x$ dans t (absurde), ou alors v est le résultat de l'application de l'une des règles (absurde puisque le résultat est toujours un ensemble)

(d) $v \neq \{x\}$: sinon dans ce cas soit $v = \{x\}$ dans t (absurde), soit v est le résultat de l'application de l'une des trois règles. Seule $Fire_c$ peut donner ce résultat. v est donc le résultat de l'application de $Fire_c$ à $[a \to b](c)$ est dans ce cas-là c'est que $\sigma b = x$ avec $x \in Var(\sigma)$ ou $x \notin Var(\sigma)$. Dans les deux cas, b était une variable dans t (absurde).

4. s'il existait dans t' un sous-terme de la forme $[u](v)$ où u n'est pas une abstraction cela signifierait qu'un tel terme existait dans t ce qui absurde ;
5. puisque t est ρ_ε-safe, tout sous-ensemble $[u \to v](w)$ de t est tel que u subsume w. En utilisant la définition de $\longrightarrow_C$ de la même manière que précédemment, nous obtenons que u subsume w' avec $w \longrightarrow_F w'$. Ainsi tout sous-terme de t' de la forme $[a \to b](c)$, a subsume c.

□

Preuve de (3) **:** Immédiat par (1) et (2). □

□

Lemme C.8 *Etant donnés les ρ-termes l, t, l', t' tels que $l \longrightarrow_{F_\parallel} l'$ et $t \longrightarrow_{F_\parallel} t'$. Alors,*

- *si l, t sont ρ_ε-préfiltables alors l', t' sont ρ_ε-préfiltables,*
- *si t est ρ_ε-safe alors t' est ρ_ε-safe,*
- *si l, t sont ρ_ε-calculables alors l', t' sont ρ_ε-calculables.*

Preuve : Similaire au lemme précedent. □

Lemme C.9 *Etant donnés les ρ-termes l, t et t' tels que l, t soient ρ_ε-préfiltables et $t \longrightarrow_{F_\parallel} t'$.*

a. Si $\langle x_1/u_1, \ldots, x_n/u_n\rangle$ est le résultat de $(l \ll^?_\emptyset t)$ et $\langle x_1/v_1, \ldots, x_n/v_n\rangle$ est le résultat de $(l \ll^?_\emptyset t')$ (où n est le nombre de variables de l), alors $u_i \longrightarrow_{F_\parallel} v_i$ (i=1...n).

b. $Solution(l \ll^?_\emptyset t) = \emptyset$ ssi $Solution(l \ll^?_\emptyset t') = \emptyset$.

Preuve : Si t est clos et du ε-premier ordre , $t' = t$ et le résultat est trivial. Sinon procédons par induction sur la structure du ρ_ε-terme t :

Cas de Base : $t = x$

a. puisque dans de cas-là $t' = t$ le résultat est trivial ;
b. forcément $l = y$ et donc le filtrage $(l \ll^?_\emptyset t)$ $(= (l \ll^?_\emptyset t'))$ ne peut pas échouer

Induction

Si $t = [t_1](t_2)$, $t = \{t_1, \ldots, t_q\}$ et $t = t_1 \to t_2$ alors $l = x$ et donc $\sigma = \langle x/t\rangle$ et $\sigma' = \langle x/t'\rangle$ et le résultat est trivial. Il en est de même pour le cas $t = f(t_1, \ldots, t_q)$ et $l = x$. Supposons donc maintenant que $t = f(t_1, \ldots, t_n)$ et $l = g(l_1, \ldots, l_m)$.

a. Si le filtrage n'échoue pas alors $l = f(l_1, \ldots, l_n)$ et les l_i, t_i sont ρ_ε-préfiltables. Par induction si $t_i \longrightarrow_{F_\parallel} t'_i$ et la solution de $(l_i \ll^?_\emptyset t_i)$ est $\langle x_{1i}/u_{1i}, \ldots, x_{mi}/u_{mi}\rangle$ alors la solution de $(l_i \ll^?_\emptyset t'_i)$ est $\langle x_{1i}/u'_{1i}, \ldots, x_{mi}/u'_{mi}\rangle$ avec $u_{ki} \longrightarrow_{F_\parallel} u'_{ki}$ avec $1 \leqslant k \leqslant m$ et $1 \leqslant i \leqslant q$. Puisque t' est ε_{exn}-réduit, $t' = f(t'_1, \ldots, t'_q)$ avec $t_i \longrightarrow_{F_\parallel} t'_i$ et la première règle de filtrage appliquée est *Decomposition* : $(l \ll^?_\emptyset t') = (\bigwedge_{i=1,\ldots,q} l_i \ll^?_\emptyset t'_i)$. Comme l est linéaire, les (x_{ki}) sont différents et donc $MergingClash$ ne peut pas être appliquée et on a donc la propriété recherchée.

b. Comme vu précédemment, comme l, t sont ρ_ε-préfiltables, l'échec de filtrage ne peut être dû qu'à $SymbolClash$ soit en position de tête, soit en position plus profonde. Si nous avons $l = f(l_1, \ldots, l_q)$ et $t = g(t_1, \ldots, t_n)$ $(f \neq g)$ comme t est ε_{exn}-réduit, t' est de la forme $t' = g(t'_1, \ldots, t'_n)$ avec $t_i \longrightarrow_{F_\parallel} t'_i$, on a : $(l \ll^?_\emptyset t)$ et $(l \ll^?_\emptyset t')$ échouent. Si nous avons $l = f(l_1, \ldots, l_q)$ et $t = f(t_1, \ldots, t_q)$ comme t est ε_{exn}-réduit, t' est de la forme $t' = f(t'_1, \ldots, t'_n)$ puisque par induction $(l_i \ll^?_\emptyset t_i)$ échoue ssi $(l_i \ll^?_\emptyset t'_i)$ échoue, on a $(l \ll^?_\emptyset t)$ échoue ssi $(l \ll^?_\emptyset t')$ échoue

□

Lemme C.10 *Etant donnés les ρ-termes l, t, r et t', r' tels que l, t soient ρ_ε-calculables et $t \longrightarrow_{F_\parallel} t'$, $r \longrightarrow_{F_\parallel} r'$. Si les problèmes de filtrage $(l \ll^?_\emptyset t)$ et $(l \ll^?_\emptyset t')$ ont comme solutions les substitutions σ et σ' respectivement alors, $\sigma r \longrightarrow_{F_\parallel} \sigma' r'$.*

Preuve : Commençons par remarquer que comme vu précédemment, le filtrage $(l \ll^?_\emptyset t)$ échoue ssi le filtrage $(l \ll^?_\emptyset t')$ échoue.

On se place dans le cas où $(l \ll^?_\emptyset t)$ n'échoue pas et donc le filtrage $(l \ll^?_\emptyset t')$ non plus. On note $\sigma = \langle x_1/s_1, \ldots, x_m/s_m \rangle$ et $\sigma = \langle x_1/s'_1, \ldots, x_m/s'_m \rangle$. Par le Lemme C.9 on sait que $s_i \longrightarrow_{F_\parallel} s'_i$. Procédons par induction sur la structure du terme r en considérant toutes les réductions possibles $r \longrightarrow_{F_\parallel} r'$.

1. $r = x$ et $r' = x$
2. $r = \{u_1, \ldots, u_n\}$ et $r' = \{u'_1, \ldots, u'_n\}$ avec $u_i \longrightarrow_{F_\parallel} u'_i$
3. $r = f(u_1, \ldots, u_n)$ et $r' = f(u'_1, \ldots, u'_n)$ avec $u_i \longrightarrow_{F_\parallel} u'_i$
4. $r = u \to v$ et $r' = u' \to v'$ avec $u \longrightarrow_{F_\parallel} u'$, $v \longrightarrow_{F_\parallel} v'$
5. $r = [u](v)$ et $r' = [u'](v')$ avec $u \longrightarrow_{F_\parallel} u'$, $v \longrightarrow_{F_\parallel} v'$
6. $r = [u \to v](w)$ avec u, w ρ_ε-calculables et $r' = v' \langle\!\langle Solution(u' \ll^?_\emptyset w') \rangle\!\rangle$,et $u \longrightarrow_{F_\parallel} u'$, $v \longrightarrow_{F_\parallel} v'$, $w \longrightarrow_{F_\parallel} w'$.
7. $r = [f(u_1, \ldots, u_n)](f(v_1, \ldots, v_n))$ et $r' = \{f([u'_1](v'_1), \ldots, [u'_n](v'_n))\}$ avec $u_i \longrightarrow_{F_\parallel} u'_i$, $v_i \longrightarrow_{F_\parallel} v'_i$ pour tout $1 \leqslant i \leqslant n$.
8. $r = [f(u_1, \ldots, u_n)](g(v_1, \ldots, v_m)$ et $r' = \{\bot\}$.

Pour le cas $r = x$, $\sigma r = s_i$ et $\sigma' r' = \sigma' r = s'_i$ et on conclut immédiatement par le Lemme C.9.

L'induction s'applique de manière tout à fait naturelle sauf pour le cas (6) qui est plus délicat à traiter. Nous devons prouver que

$$\sigma([u \to v](w)) \longrightarrow_{F_\parallel} \sigma' \{\mu v'\} \tag{C.1}$$

avec $\mu \in Solution(u' \ll^?_\emptyset w') = Solution(u \ll^?_\emptyset w')$

Par α-convertion, nous supposons que u ne contient aucune variable de σ et aucune variable de σ'. On a donc $\sigma([u \to v](w)) = [u \to \sigma(v)](\sigma w)$. Puisque σ représente la solution du problème entre deux termes ρ_ε-calculables, σ est ρ_ε-safe (Proposition C.1) et comme de plus u, w sont ρ_ε-calculables, nous obtenons que $u, \sigma(w)$ sont ρ_ε-calculables (Proposition C.2).

Si $Solution(u \ll^?_\emptyset \sigma w) = \emptyset$ alors $[u \to \sigma v](\sigma w) \longrightarrow_{F_\parallel} \{\bot\}$. Puisque les termes $u, \sigma w$ sont ρ_ε-calculables, $(u \ll^?_\emptyset \sigma w)$ ne peut échouer qu'à cause de symboles ε-fonctionnels différents à la même position des termes u et σw et donc des termes u et w (puisque u, w sont ρ_ε-calculables). Comme w est ε_{exn}-réduit, il ne peut être réduit par $\longrightarrow_{F_\parallel}$ qu'en un terme de la même forme et donc le filtrage $(u \ll^?_\emptyset w')$ échoue et donc $v' \langle\!\langle Solution(u \ll^?_\emptyset w') \rangle\!\rangle = \{\bot\}$.

Si le filtrage $(u \ll^?_\emptyset \sigma w)$ n'échoue pas, nous pouvons appliquer l'induction aux sous-termes v et w et nous avons $\sigma v \longrightarrow_{F_\parallel} \sigma' v'$ et $\sigma w \longrightarrow_{F_\parallel} \sigma' w'$. Ainsi, nous obtenons la réduction :

$$\sigma([u \to v](w)) = [u \to \sigma v](\sigma w) \longrightarrow_{F_\parallel} \{\mu'(\sigma' v')\} \tag{C.2}$$

avec $\mu' \in \mathcal{S}olution(u \ll^?_\emptyset \sigma' w')$.

En utilisant les réductions (C.1) et (C.2), l'égalité suivante nous permet de conclure la preuve :

$$\{\mu'(\sigma' v')\} = \sigma'\{\mu v'\}$$

Il faut donc montrer que $\mu'(\sigma' v') = \sigma'(\mu v')$.

En effet, nous supposons que $\sigma' = \langle x_1/t_1, \ldots, x_n/t_n\rangle$ et $\mu = \langle y_1/s_1, \ldots, y_m/s_m\rangle$. Il est clair que $\mu' = \langle y_1/\sigma' s_1, \ldots, y_m/\sigma' s_m\rangle$. Puisque u ne contient aucune variable de σ' nous déduisons que y_j n'est pas une variable de t_i et $x_i \neq y_j$ pour tous $j = 1 \ldots m$, $i = 1 \ldots n$. Ainsi, nous avons $\mu'(\sigma' v') = \langle y_1/\sigma' s_1, \ldots, y_m/\sigma' s_m\rangle(\langle x_1/t_1, \ldots, x_n/t_n\rangle v')$ et puisque y_j n'est pas une variable de t_i,

$$\mu'(\sigma' v') = \langle y_1/\sigma' s_1, \ldots, y_m/\sigma' s_m, x_1/t_1, \ldots, x_n/t_n\rangle v'.$$

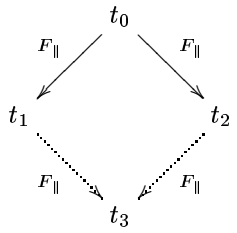
Pour le deuxième terme nous prenons $\sigma'(\mu v') = \langle x_1/t_1, \ldots, x_n/t_n\rangle(\langle y_1/s_1, \ldots, y_m/s_m\rangle v')$ et puisque $x_i \neq y_j$ et y_j n'est pas une variable de t_i,

$$\sigma'(\mu v') = \langle x_1/t_1, \ldots, x_n/t_n, y_1/\langle x_1/t_1, \ldots, x_n/t_n\rangle s_1, \ldots, y_m/\langle x_1/t_1, \ldots, x_n/t_n\rangle s_m\rangle v' =$$
$$\langle x_1/t_1, \ldots, x_n/t_n, y_1/\sigma' s_1, \ldots, y_m/\sigma' s_m\rangle v'$$

L'égalité $\mu'(\sigma' v') = \sigma'(\mu v')$ est valide et ainsi, le lemme est prouvé.

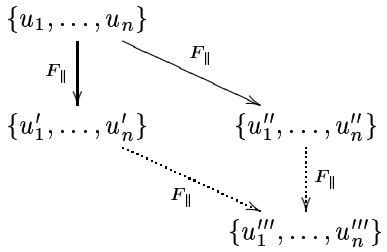
□

Lemme C.11 *($\longrightarrow_{F_\parallel}$ est fortement confluente) Etant donnés les termes t_0, t_1, t_2 tels que $t_0 \longrightarrow_{F_\parallel} t_1$ et $t_0 \longrightarrow_{F_\parallel} t_2$. Alors, il existe un terme t_3 tel que $t_1 \longrightarrow_{F_\parallel} t_3$ et $t_2 \longrightarrow_{F_\parallel} t_3$:*

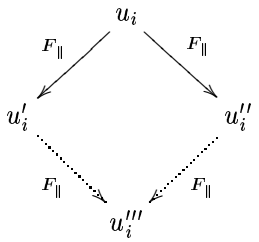


Preuve : Nous montrons le lemme par induction sur la structure du terme t_0.

1. $t_0 = x$
   Par définition de la relation $\longrightarrow_{F_\parallel}$, $t_0 = t_1 = t_2$ et nous pouvons choisir $t_3 = t_0$.
2. $t_0 = \{u_1, \ldots, u_n\}$
   Nous avons $t_1 = \{u'_1, \ldots, u'_n\}$, $t_2 = \{u''_1, \ldots, u''_n\}$ avec $u_i \longrightarrow_{F_\parallel} u'_i$ et $u_i \longrightarrow_{F_\parallel} u''_i$, $i = 1 \ldots n$. Par induction, il existe les termes u'''_i tels que $u'_i \longrightarrow_{F_\parallel} u'''_i$, $u''_i \longrightarrow_{F_\parallel} u'''_i$, $i = 1 \ldots n$. Ainsi, nous pouvons choisir $t_3 = \{u'''_1, \ldots, u'''_n\}$.
   Les diagrammes correspondants sont présentés ci-dessous :



puisque



3. $t_0 = f(u_1, \ldots, u_n)$ avec $f \in \mathcal{F}_\varepsilon$

   Nous avons $t_1 = f(u'_1, \ldots, u'_n)$ et $t_2 = f(u''_1, \ldots, u''_n)$ avec $u_i \longrightarrow_{F_\parallel} u'_i$ et $u_i \longrightarrow_{F_\parallel} u''_i$, $i = 1 \ldots n$. Par induction, il existe les termes u'''_i tels que $u'_i \longrightarrow_{F_\parallel} u'''_i$, $u''_i \longrightarrow_{F_\parallel} u'''_i$, $i = 1 \ldots n$. Ainsi, nous pouvons choisir $t_3 = f(u'''_1, \ldots, u'''_n)$.

4. $t_0 = u_0 \to v_0$

   u_0 est du ε-premier ordre et par définition de $\longrightarrow_{F_\parallel}$, $t_1 = u_0 \to v_1$ et $t_2 = u_0 \to v_2$ avec $v_0 \longrightarrow_{F_\parallel} v_1$ et $v_0 \longrightarrow_{F_\parallel} v_2$. Par induction, il existe v_3 tel que $v_1 \longrightarrow_{F_\parallel} v_3$ et $v_2 \longrightarrow_{F_\parallel} v_3$ et donc nous obtenons $t_3 = u_0 \to v_3$.

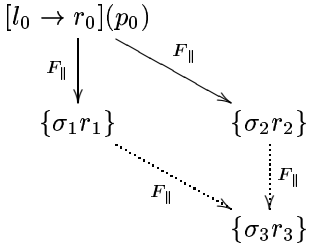
5. $t_0 = [u_0](v_0)$

   Nous avons plusieurs possibilités pour choisir les termes t_1, t_2 selon les propriétés des termes u_0, v_0 décrites dans la définition de la relation $\longrightarrow_{F_\parallel}$:

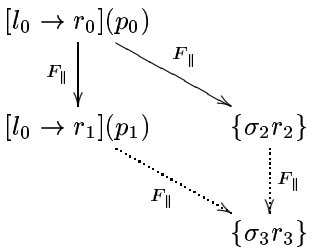
   (a) $t_1 = [u_1](v_1)$ et $t_2 = [u_2](v_2)$ avec $u_0 \longrightarrow_{F_\parallel} u_1$, $u_0 \longrightarrow_{F_\parallel} u_2$, $v_0 \longrightarrow_{F_\parallel} v_1$, $v_0 \longrightarrow_{F_\parallel} v_2$. Par induction, il existe les termes u_3, v_3 tels que $u_1 \longrightarrow_{F_\parallel} u_3$, $u_2 \longrightarrow_{F_\parallel} u_3$ et $v_1 \longrightarrow_{F_\parallel} v_3$, $v_2 \longrightarrow_{F_\parallel} v_3$. Ainsi, nous obtenons $t_3 = [u_3](v_3)$.

   (b) $t_0 = [l_0 \to r_0](p_0)$ avec l_0, p_0 ρ_ε-calculables. Conformément au Lemme C.9 seulement trois cas peuvent se présenter (i.e. le cas $t_0 = [l_0 \to r_0](p_0)$, $t_1 = \{\bot\}$ et $t_2 = \{\sigma(r_1)\}$ est impossible ici):

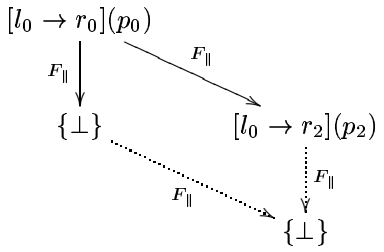
      i. Si nous avons $r_0 \longrightarrow_{F_\parallel} r_1$, $p_0 \longrightarrow_{F_\parallel} p_1$, $r_0 \longrightarrow_{F_\parallel} r_2$, $p_0 \longrightarrow_{F_\parallel} p_2$, $t_1 = \{\sigma_1 r_1\}$ et $t_2 = \{\sigma_2 r_2\}$, avec σ_1 solution de $(l_0 \ll^?_\emptyset p_1)$ et σ_2 solution de $(l_0 \ll^?_\emptyset p_2)$. Par induction il existe des termes r_3, p_3 tels que $r_1 \longrightarrow_{F_\parallel} r_3$, $r_2 \longrightarrow_{F_\parallel} r_3$, $p_1 \longrightarrow_{F_\parallel} p_3$ et $p_2 \longrightarrow_{F_\parallel} p_3$. Puisque l_0, p_1 et l_0, p_2 sont ρ_ε-calculables par le Lemme C.8 alors, nous pouvons utiliser le Lemme C.10 et on pose $t_3 = \{\sigma_3 r_3\}$ avec σ_3 solution du filtrage $(l_0 \ll^?_\emptyset p_3)$ (qui n'échoue pas par le Lemme C.9 et le fait que $(l_0 \ll^?_\emptyset p_0)$ n'échoue pas) :



      ii. Si nous avons $r_0 \longrightarrow_{F_\parallel} r_1$, $p_0 \longrightarrow_{F_\parallel} p_1$, $r_0 \longrightarrow_{F_\parallel} r_2$, $p_0 \longrightarrow_{F_\parallel} p_2$ et $t_1 = \{\sigma_1 r_1\}$, $t_2 = [l_0 \to r_2](p_2)$ avec σ_1 solution de $(l_0 \ll^?_\emptyset p_1)$. En raisonnant de la même façon que dans le cas précédent nous obtenons :

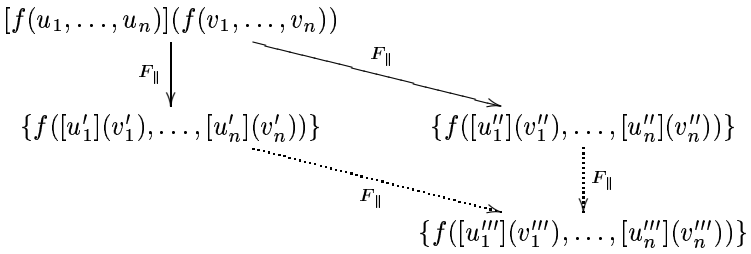


      iii. Si nous avons $r_0 \longrightarrow_{F_\parallel} r_1$, $p_0 \longrightarrow_{F_\parallel} p_1$, $r_0 \longrightarrow_{F_\parallel} r_2$, $p_0 \longrightarrow_{F_\parallel} p_2$ $t_1 = \{\bot\}$ et $t_2 = [l_0 \to r_2](p_2)$. Par le Lemme C.9, si le fitrage $(l_0 \ll^?_\emptyset p_1)$ échoue alors, le filtrage $(l_0 \ll^?_\emptyset p_0)$ échoue et donc le filtrage $(l_0 \ll^?_\emptyset p_2)$ échoue. Par le lemme C.8, comme l_0, t_0 sont ρ_ε-calculables, l_0, p_2 sont ρ_ε-calculables et donc $[l_0 \to r_2](p_2) \longrightarrow_{F_\parallel} \{\bot\}$.

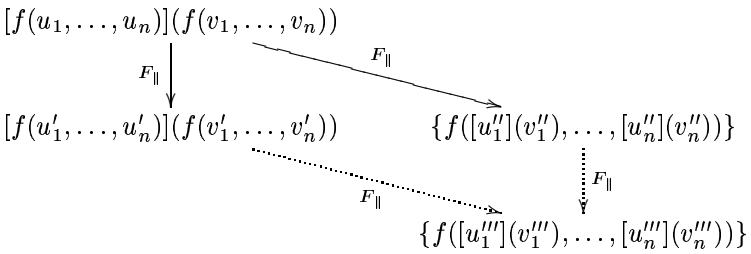


(c) $t_0 = [f(u_1, \ldots, u_n)](f(v_1, \ldots, v_n))$ avec $f \in \mathcal{F}$

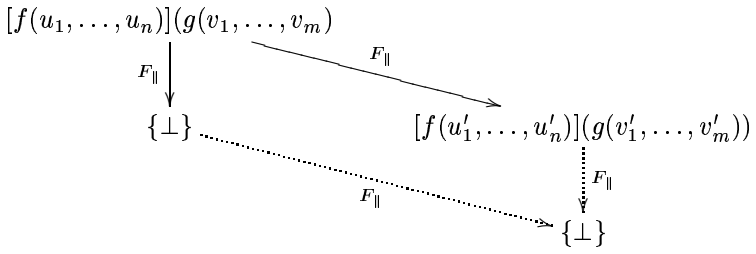
i. Nous considérons $t_1 = \{f([u'_1](v'_1), \ldots, [u'_n](v'_n))\}$, $t_2 = \{f([u''_1](v''_1), \ldots, [u''_n](v''_n))\}$ avec $u_i \longrightarrow_{F_\parallel} u'_i$, $v_i \longrightarrow_{F_\parallel} v'_i$ et $u_i \longrightarrow_{F_\parallel} u''_i$, $v_i \longrightarrow_{F_\parallel} v''_i$, $i = 1 \ldots n$. Par induction, il existe les termes u'''_i, v'''_i tels que $u'_i \longrightarrow_{F_\parallel} u'''_i$, $u''_i \longrightarrow_{F_\parallel} u'''_i$ et $v'_i \longrightarrow_{F_\parallel} v'''_i$, $v''_i \longrightarrow_{F_\parallel} v'''_i$, $i = 1 \ldots n$. Par conséquent, nous pouvons choisir $t_3 = \{f([u'''_1](v'''_1), \ldots, [u'''_n](v'''_n))\}$.



ii. Maintenant, nous considérons les termes $t_1 = [f(u'_1, \ldots, u'_n)](f(v'_1, \ldots, v'_n))$ et $t_2 = \{f([u''_1](v''_1), \ldots, [u''_n](v''_n))\}$ tels que nous avons $f(u_1, \ldots, u_n) \longrightarrow_{F_\parallel} f(u'_1, \ldots, u'_n)$, $f(v_1, \ldots, v_n) \longrightarrow_{F_\parallel} f(v'_1, \ldots, v'_n)$ et $u_i \longrightarrow_{F_\parallel} u''_i$, $v_i \longrightarrow_{F_\parallel} v''_i$, $i = 1 \ldots n$. Par conséquent, nous devons avoir $u_i \longrightarrow_{F_\parallel} u'_i$ et $v_i \longrightarrow_{F_\parallel} v'_i$, $i = 1 \ldots n$. Par induction, il existe les termes u'''_i, v'''_i tels que $u'_i \longrightarrow_{F_\parallel} u'''_i$, $u''_i \longrightarrow_{F_\parallel} u'''_i$ et $v'_i \longrightarrow_{F_\parallel} v'''_i$, $v''_i \longrightarrow_{F_\parallel} v'''_i$, $i = 1 \ldots n$. Par conséquent, nous pouvons choisir $t_3 = \{f([u'''_1](v'''_1), \ldots, [u'''_n](v'''_n))\}$.



(d) $t_0 = [f(u_1, \ldots, u_n)](g(v_1, \ldots, v_m))$ avec $f, g \in \mathcal{F}$
Si nous avons $t_1 = \{\bot\}$, $t_2 = [f(u'_1, \ldots, u'_n)](g(v'_1, \ldots, v'_m))$ avec $u_i \longrightarrow_{F_\parallel} u'_i$, $i = 1 \ldots n$, $v_j \longrightarrow_{F_\parallel} v'_j$, $j = 1 \ldots m$ alors nous pouvons choisir $t_3 = \{\bot\}$.



□

Lemme C.12 *La relation $\overset{*}{\longrightarrow}_F$ est la fermeture transitive de la relation $\longrightarrow_{F_\parallel}$.*

Preuve : Nous allons prouver les inclusions suivantes :

$$\longrightarrow_F \quad \subseteq \quad \longrightarrow_{F_\parallel} \quad \subseteq \quad \overset{*}{\longrightarrow}_F$$

et dans ce cas, puisque $\xrightarrow{*}_F$ est la fermeture transitive de la relation $\longrightarrow_F$, $\xrightarrow{*}_F$ est la fermeture transitive de la relation $\longrightarrow_{F_\parallel}$.

Nous devons donc prouver les deux inclusions :

$$\longrightarrow_F \subseteq \longrightarrow_{F_\parallel} \quad et \quad \longrightarrow_{F_\parallel} \subseteq \xrightarrow{*}_F$$

Il est clair que $\longrightarrow_F \subseteq \longrightarrow_{F_\parallel}$.

Pour prouver que $\longrightarrow_{F_\parallel} \subseteq \xrightarrow{*}_F$ le seul cas non-trivial est celui correspondant à la règle d'évaluation $Fire_c$. On doit montrer que :

> si $u \xrightarrow{*}_F u'$, $v \xrightarrow{*}_F v'$ et $w \xrightarrow{*}_F w'$ avec les termes u, w ρ_ε-calculables alors $[u \to w](v) \xrightarrow{*}_F v'\langle\!\langle Solution(u' \ll^?_\emptyset w')\rangle\!\rangle$

Il est clair que $[u \to w](v) \xrightarrow{*}_F [u' \to w'](v')$ et puisque, par le Lemme C.8, les termes u', w' sont ρ_ε-calculables alors, en utilisant la définition de la relation $\longrightarrow_F$, nous déduisons que $[u' \to w'](v') \longrightarrow_F v'\langle\!\langle Solution(u' \ll^?_\emptyset w')\rangle\!\rangle$. Ainsi, par transitivité $[u \to w](v) \xrightarrow{*}_F v'\langle\!\langle Solution(u' \ll^?_\emptyset w')\rangle\!\rangle$, ce qui achève la preuve.

□

Théorème C.1 *($\xrightarrow{*}_F$ est fortement confluente, $\longrightarrow_F$ est confluente) Si $t \xrightarrow{*}_F u$ et $t \xrightarrow{*}_F v$ alors il existe un terme w tel que $u \xrightarrow{*}_F w$ et $v \xrightarrow{*}_F w$.*

Preuve : Conformément au Lemme C.12, la relation $\xrightarrow{*}_F$ est la fermeture transitive de la relation $\longrightarrow_{F_\parallel}$. Le Lemme C.11 montre la confluence forte de la relation $\longrightarrow_{F_\parallel}$. Ainsi, par le Lemme C.6, la relation $\xrightarrow{*}_F$ est fortement confluente et par conséquent, la relation $\longrightarrow_F$ est confluente.

□

C.4 La confluence

C.4.1 Confluence de la relation $\longrightarrow_{F/C}$

Lemme C.13 *Etant donnés deux ρ-termes l, t et t' tels que l, t soient ρ_ε-préfiltables, $t \xrightarrow{*}_C t'$ et l, t' soient ρ_ε-préfiltables.*

a. Si $\langle x_1/u_1, \ldots, x_n/u_n\rangle$ est le résultat de $(l \ll^?_\emptyset t)$ et $\langle x_1/v_1, \ldots, x_n/v_n\rangle$ est le résultat de $(l \ll^?_\emptyset t')$ (avec n le nombre de variables de l), alors $u_i \xrightarrow{*}_C v_i$.

b. $Solution(l \ll^?_\emptyset t) = \emptyset$ ssi $Solution(l \ll^?_\emptyset t') = \emptyset$.

Preuve : Si t est du ε-premier ordre, alors $t' = t$ et le résultat est trivialement vrai.

On raisonne par induction sur la structure du terme t.

Cas de base : $t = x$.

Puisque l, t sont ρ_ε-préfiltables, l ε-subsume faiblement t et donc l est forcément une variable

a. $t' = t$ et le résultat est évident ;

b. puisque l est une variable, $(l \ll^?_\emptyset t) = (l \ll^?_\emptyset t')$ ne peut pas échouer.

Induction

Les cas que nous devons analyser sont $t = \{t_1, \ldots, t_q\}$, $t = f(t_1, \ldots, t_q)$, $t = t_1 \to t_2$ et $t = [t_1](t_2)$. Si $t = \{t_1, \ldots, t_q\}$, $t = t_1 \to t_2$ ou $t = [t_1](t_2)$ alors la position de tête de l est une position variable

soit $l = x$, le lemme est clairement vrai. Le cas où $l = x$ et $t = f(t_1, \ldots, t_q)$ où $f \in \mathcal{F}_\varepsilon$ est trivial. Si $t = f(t_1, \ldots, t_q)$ et l n'est pas une variable nous analysons successivement les deux points :

a. si le filtrage n'échoue pas alors le terme l doit être de la forme $l = f(l_1, \ldots, l_q)$ et la solution de $(l \ll^?_\emptyset t) = (\bigwedge_{i=1,\ldots,q} l_i \ll^?_\emptyset t_i)$ est $\langle x_1/u_1, \ldots, x_n/u_n \rangle$. Puisque l, t' sont ρ_ε-préfiltables, t' est forcément de la forme $t' = f(t'_1, \ldots, t'_n)$ et $t_i \xrightarrow{*}_C t'_i$ et l_i, t'_i ρ_ε-préfiltables, pour $i = 1 \ldots q$.
Par hypothèse d'induction, si $t_i \xrightarrow{*}_C t'_i$ et la solution de $(l_i \ll^?_\emptyset t_i)$ est la substitution $\langle x_{1i}/u_{1i}, \ldots, x_{mi}/u_{mi} \rangle$, alors la solution de $(l_i \ll^?_\emptyset t'_i)$ est $\langle x_{1i}/u'_{1i}, \ldots, x_{mi}/u'_{mi} \rangle$ avec $u_{ki} \xrightarrow{*}_C u'_{ki}$.
La première règle de filtrage appliquée pour filtrer l' et t' est *Decomposition* et nous obtenons $(l \ll^?_\emptyset t') = (\bigwedge_{i=1,\ldots,q} l_i \ll^?_\emptyset t'_i)$. Comme l est linéaire, les variables (x_{ki}) sont différentes et donc la règle de filtrage $MergingClash$ ne peut pas être appliquée et la propriété est donc vérifiée.

b. L'échec peut être obtenu seulement en appliquant la règle de filtrage $SymbolClash$ à la position de tête ou à des positions plus profondes.
Nous pouvons donc avoir $l = g(l_1, \ldots, l_q)$ et $t' = f(t'_1, \ldots, t'_q)$ avec $t_i \xrightarrow{*}_C t'_i$ et $f \neq g$ et ainsi, $(l \ll^?_\emptyset t)$ et $(l \ll^?_\emptyset t')$ échouent. Si $l = f(l_1, \ldots, l_q)$ alors l'échec est obtenu à une position plus profonde. Puisque par induction, le problème $(l_i \ll^?_\emptyset t_i)$ mène à un échec de filtrage ssi le problème $(l_i \ll^?_\emptyset t'_i)$ mène à un échec alors, $(l \ll^?_\emptyset t)$ échoue ssi $(l \ll^?_\emptyset t')$ échoue.

□

Lemme C.14 *Etant donnés les ρ-termes l, t, r et t' tels que l, t soient ρ_ε-préfiltables, $t \xrightarrow{*}_C t'$ et l, t' soient ρ_ε-préfiltables. Si les problèmes de filtrage $(l \ll^?_\emptyset t)$ et $(l \ll^?_\emptyset t')$ ont comme solutions les substitutions σ et σ' respectivement alors, $\sigma r \xrightarrow{*}_C \sigma' r$.*

Preuve : Nous procédons par induction sur la structure du terme r.

Le cas de base, $r = x$ suit immédiatement par le Lemme C.13.

Tous les autres cas sont traités facilement en utilisant l'hypothèse d'induction. Par exemple, si $r = \{u_1, \ldots, u_m\}$ nous avons par induction $\sigma u_i \xrightarrow{*}_C \sigma' u_i$, $i = 1 \ldots m$, et puisque la relation $\xrightarrow{*}_C$ est fermée par contexte, $\sigma\{u_1, \ldots, u_m\} = \{\sigma u_1, \ldots, \sigma u_m\} \xrightarrow{*}_C \{\sigma' u_1, \ldots, \sigma' u_m\} = \sigma'\{u_1, \ldots, u_m\}$.

□

Lemme C.15 *Etant donnés les ρ-termes l, t, r et r' tels que $r \xrightarrow{*}_C r'$. Si la substitution σ est la solution du problème de filtrage $(l \ll^?_\emptyset t)$, alors $\sigma r \xrightarrow{*}_C \sigma r'$.*

Preuve : Nous raisonnons par induction sur le nombre de pas de réduction $r = r_1 \longrightarrow_C \ldots \longrightarrow_C r_n = r'$. Il est suffisant de montrer que : $r_n \longrightarrow_C r_{n+1} \Rightarrow \sigma r_n \longrightarrow_C \sigma r_{n+1}$. On montre donc que $t \longrightarrow_C t' \Rightarrow \sigma t \xrightarrow{*}_C \sigma t'$ en utilisant une induction sur la structure du terme t. Le cas de base $t = x$ est évident puisque $t' = x$ et donc $\sigma x \xrightarrow{*}_C \sigma x$. Nous considérons toutes les formes d'un ρ_ε-terme et toutes les réductions $\longrightarrow_C$ possibles.

1. $t = \{u_1, \ldots, u_m\}$
   (a) Si $t = \{u_1, \ldots, u_m\}$ et $t' = \{u'_1, \ldots, u_m\}$, avec $u_1 \longrightarrow_C u'_1$, nous obtenons, par induction, $\sigma u_1 \xrightarrow{*}_C \sigma u'_1$ et donc, $\sigma t = \{\sigma u_1, \ldots, \sigma u_m\} \xrightarrow{*}_C \{\sigma u'_1, \ldots, \sigma u_m\} = \sigma t'$. Dans le cas où nous avons à la place de u_1 un autre sous-terme u_k tel que $u_k \longrightarrow_C u'_k$ la preuve est similaire.
   (b) Si $t = \{u_1, \ldots, \{v_1, \ldots, v_n\}, \ldots, u_m\}$ avec $n > 0$ et $t' = \{u_1, \ldots, v_1, \ldots, v_n, \ldots, u_m\}$ alors, $\sigma t = \{\sigma u_1, \ldots, \{\sigma v_1, \ldots, \sigma v_n\}, \ldots, \sigma u_m\} \xrightarrow{*}_C \{\sigma u_1, \ldots, \sigma v_1, \ldots, \sigma v_n, \ldots, \sigma u_m\} = \sigma t'$.
   (c) Si $t = \{u_1, \ldots, \bot, \ldots, u_n\}$ et $t' = \{u_1, \ldots, u_n\}$ alors, $\sigma t = \{\sigma u_1, \ldots, \bot, \ldots, u_n\} \xrightarrow{*}_C \{\sigma u_1, \ldots, u_n\} = \sigma t'$.
2. $t = exn(t_1)$
   (a) Si $t = exn(t_1)$ et $t' = exn(t'_1)$ avec $t_1 \xrightarrow{*}_C t'_1$, alors par hypothèse d'induction nous obtenons $\sigma t_1 \xrightarrow{*}_C \sigma t'_1$. Or $\sigma(exn(t_1)) = exn(\sigma t_1) \xrightarrow{*}_C exn(\sigma t'_1) = \sigma(exn(t'_1)) = \sigma t'$

(b) Si $t = exn(t_1)$ et $t' = t_1$ alors cela signifie que t est Exn-réductible. Donc t est clos et donc $\sigma t = \sigma(exn(t_1)) = exn(\sigma t_1) = exn(t_1) \xrightarrow{*}_C t_1 = \sigma t_1 = \sigma t'$.

3. $t = \perp$

   Si $t = \perp$ alors $t' = \perp$ et le résultat est immédiat.

4. $t = f(u_1, \ldots, u_m)$ où $f \in \mathcal{F}$

   (a) Si $t = f(u_1, \ldots, u_m)$ et $t' = f(u'_1, \ldots, u_m)$, avec $u_1 \longrightarrow_C u'_1$, nous obtenons, par induction, $\sigma u_1 \xrightarrow{*}_C \sigma u'_1$ et donc, $\sigma t = f(\sigma u_1, \ldots, \sigma u_m) \xrightarrow{*}_C f(\sigma u'_1, \ldots, \sigma u_m) = \sigma t'$. Dans le cas où nous avons à la place de u_1 un autre sous-terme u_k tel que $u_k \longrightarrow_C u'_k$ la preuve est similaire.

   (b) Si nous considérons le terme $t = f(u_1, \ldots, \{v_1, \ldots, v_n\}, \ldots, u_m)$ avec $n > 0$ et le terme $t' = \{f(u_1, \ldots, v_1, \ldots, u_m), \ldots, f(u_1, \ldots, v_n, \ldots, u_m)\}$ alors nous obtenons la réduction $\sigma t = f(\sigma u_1, \ldots, \{\sigma v_1, \ldots, \sigma v_n\}, \ldots, \sigma u_m) \xrightarrow{*}_C$
   $\{f(\sigma u_1, \ldots, \sigma v_1, \ldots, \sigma u_m), \ldots, f(\sigma u_1, \ldots, \sigma v_n, \ldots, \sigma u_m)\} = \sigma t'$

   (c) Si $t = f(\perp, u_2, \ldots, u_m)$ et $t' = \{\perp\}$ alors $\sigma t = f(\perp, \sigma u_2, \ldots, \sigma u_m) \xrightarrow{*}_C \{\perp\} = t' = \sigma t'$. La preuve est similaire dans le cas où $\perp$ est à la place d'un autre u_k.

5. $t = u \to v$

   (a) Si $t = u \to v$ et $t' = u' \to v$ par définition du ρ_ε-calcul, $u = u'$ comme u est du ε-premier ordre .

   (b) Si $t = u \to v$ et $t' = u \to v'$ la preuve est similaire au point 4a.

   (c) Si $t = u \to \{v_1, \ldots, v_m\}$ et $t' = \{u \to v_1, \ldots, u \to v_m\}$ la preuve est similaire au point 4b.

   (d) Si $t = u \to \perp$ et $t' = \{\perp\}$, $\sigma t = \sigma(u) \to \perp \xrightarrow{*}_C \{\perp\} = t' = \sigma t'$

6. $t = [u](v)$

   (a) Si $t = [u](v)$ et $t' = [u'](v)$ la preuve est similaire au point 4a.

   (b) Si $t = [u](v)$ et $t' = [u](v')$ la preuve est similaire au point 4a.

   (c) Si $t = [\perp](v)$ et $t' = \{\perp\}$, alors la preuve est similaire au point 5d.

   (d) Si $t = [u](\perp)$ et $t' = \{\perp\}$, alors la preuve est similaire au point 5d.

   (e) Si $t = [\{u_1, \ldots, u_m\}](v)$ avec $m > 0$ et $t' = \{[u_1](v), \ldots, [u_m](v)\}$, alors la preuve est similaire au point 4b.

   (f) Si $t = [\{\}](v)$ et $t' = \{\perp\}$, alors $\sigma t = [\{\}](\sigma(v)) \xrightarrow{*}_C \{\perp\} = t' = \sigma t'$

   (g) Si $t = [u](\{v_1, \ldots, v_m\})$ avec $m > 0$ et $t' = \{[u](v_1), \ldots, [u](v_m)\}$, alors la preuve est similaire au point 4b.

   (h) Si $t = [u](\{\})$ et $t' = \{\perp\}$, alors la preuve est similaire au point 6f.

□

Lemme C.16 *Etant donnés les ρ-termes l, t, r et t', r' tels que l, t soient ρ_ε-préfiltables $t \xrightarrow{*}_C t'$, $r \xrightarrow{*}_C r'$ et l, t' soient ρ_ε-préfiltables. Si les problèmes de filtrage $(l \ll^?_\emptyset t)$ et $(l \ll^?_\emptyset t')$ ont comme solutions les substitutions σ et σ' respectivement alors, $\sigma r \xrightarrow{*}_C \sigma' r'$.*

Preuve : Le résultat est obtenu immédiatement par transitivité en utilisant le Lemme C.14 et le Lemme C.15.

□

Lemme C.17 *Etant donnés les ρ-termes $l, \{t\}, r$ tels que $l, \{t\}$ soient ρ_ε-calculables. Si les problèmes de filtrage $(l \ll^?_\emptyset \{t\})$ et $(l \ll^?_\emptyset t)$ ont comme solutions les substitutions σ et σ' respectivement alors, $\sigma r \xrightarrow{*}_C \{\sigma' r\}$.*

Preuve : Puisque les termes $l, \{t\}$ sont ρ_ε-calculables, ils sont ρ_ε-préfiltables et donc l est une variable notée x. Ainsi, la solution de $(x \ll^?_\emptyset \{t\})$ est $\langle x/\{t\}\rangle = \sigma$ et la solution de $(x \ll^?_\emptyset t)$ est $\langle x/t\rangle = \sigma'$.

Nous procédons par induction sur la structure du ρ_ε-terme r.

Le cas de base : $r = x$, avec $x \in \mathcal{X}$.

Nous avons $\sigma x = \langle x/\{t\}\rangle x = \{t\}$, $\{\sigma' x\} = \{\langle x/t\rangle x\} = \{t\}$ et $\{t\} \xrightarrow{*}_C \{t\}$.

Induction :

1. $r = \{r_1, \ldots, r_m\}$
   Nous avons $\sigma r = \sigma\{r_1, \ldots, r_m\} = \{\sigma r_1, \ldots, \sigma r_m\}$ et $\{\sigma' r\} = \{\sigma'\{r_1, \ldots, r_m\}\} = \{\{\sigma' r_1, \ldots, \sigma' r_m\}\}$. Par induction, $\sigma r_i \xrightarrow{*}_C \{\sigma' r_i\}$ et donc, $\{\sigma r_1, \ldots, \sigma r_m\} \xrightarrow{*}_C \{\{\sigma' r_1\}, \ldots, \{\sigma' r_m\}\}$. En plus, nous avons $\{\{\sigma' r_1\}, \ldots, \{\sigma' r_m\}\} \xrightarrow{*}_C \{\{\sigma' r_1, \ldots, \sigma' r_m\}\}$ et ainsi, $\{\sigma r_1, \ldots, \sigma r_m\} \xrightarrow{*}_C \{\{\sigma' r_1, \ldots, \sigma' r_m\}\}$.
2. $r = f(r_1, \ldots, r_m)$ avec $f \in \mathcal{F}_\varepsilon$
   Nous procédons de la même façon que dans le premier cas.
3. $r = u \to v$
   Nous procédons de la même façon que dans le premier cas.
4. $r = [u](v)$
   Nous procédons de la même façon que dans le premier cas.

□

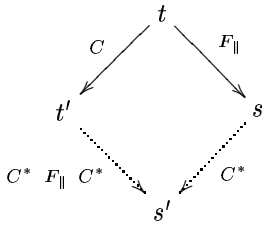
Lemme C.18 *Etant donnés les ρ-termes $l, f(u_1, \ldots, \{t\}, \ldots, u_m), r$ tels que $l, f(u_1, \ldots, \{t\}, \ldots, u_m)$ soient ρ_ε-calculables. Si les substitutions σ et σ' sont les solutions des problèmes de filtrage $(l \ll^?_\emptyset f(u_1, \ldots, \{t\}, \ldots, u_m))$ et $(l \ll^?_\emptyset f(u_1, \ldots, t, \ldots, u_m))$ respectivement alors, $\sigma r \xrightarrow{*}_C \{\sigma' r\}$.*

Preuve : Puisque les termes $l, f(u_1, \ldots, \{t\}, \ldots, u_m)$ sont ρ_ε-préfiltables alors, soit l est une variable, soit l est de la forme $f(l_1, \ldots, l_m)$.

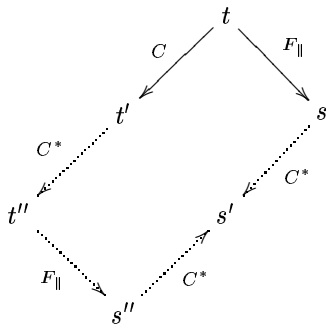
Si l est une variable alors la preuve est similaire à celle du Lemme C.17.

Si $l = f(l_1, \ldots, l_m)$, puisque $l, f(u_1, \ldots, \{t\}, \ldots, u_m)$ sont ρ_ε-préfiltables alors l_i, u_i $(i = 1 \ldots m)$ sont ρ_ε-préfiltables. Par conséquent, si $\{t\}$ est le k-ième argument alors $l_k = x$ et la preuve est similaire à celle du Lemme C.17 □

Lemme C.19 *(diagramme de Yokouchi) Etant donnés les ρ_ε-termes t, t' et s tels que $t \longrightarrow_C t'$ et $t \longrightarrow_{F_\parallel} s$. Alors, il existe un terme s' tel que $t' \xrightarrow{*}_C \longrightarrow_{F_\parallel} \xrightarrow{*}_C s'$ et $s \xrightarrow{*}_C s'$:*



Preuve : Nous pouvons reformuler le lemme en décrivant toutes les étapes intermédiaires : *Etant donnés les ρ_ε-termes t, t' et s tels que $t \longrightarrow_C t'$ et $t \longrightarrow_{F_\parallel} s$. Alors, il existe les termes s', s'' et t'' tel que $t' \xrightarrow{*}_C t''$, $t'' \longrightarrow_{F_\parallel} s''$, $s'' \xrightarrow{*}_C s'$ et $s \xrightarrow{*}_C s'$.*



Nous procédons par induction sur la structure du terme t et nous analysons toutes les réductions $\longrightarrow_C$ et $\longrightarrow_{F_\parallel}$ possibles.

Le cas de base :

Si t est une variable toutes les réductions sont triviales.

Induction :

Nous présentons par la suite tous les cas possibles.

1. t est un ensemble.
   Nous avons trois possibilités pour réduire t en t' :
   (a) $t = \{u_1, \ldots, u_k, \ldots, u_n\}$ et $t' = \{u_1, \ldots, v_k, \ldots, u_n\}$ avec $u_k \longrightarrow_C v_k$.
   Si $u_i \longrightarrow_{F_\parallel} u_i^1$, $i = 1 \ldots n$, alors, nous avons :

$$\begin{array}{ccc} \{u_1, \ldots, u_k, \ldots, u_n\} & \xrightarrow{F_\parallel} & \{u_1^1, \ldots, u_k^1, \ldots, u_n^1\} \\ \downarrow C & & \vdots C^* \\ \{u_1, \ldots, v_k, \ldots, u_n\} & & \{u_1^1, \ldots, u_k^3, \ldots, u_n^1\} \\ \vdots C^* & & \uparrow C^* \\ \{u_1, \ldots, w_k, \ldots, u_n\} & \overset{F_\parallel}{\cdots\!\!\rightarrow} & \{u_1^1, \ldots, u_k^2, \ldots, u_n^1\} \end{array} \quad \text{puisque} \quad \begin{array}{ccc} u_k & \xrightarrow{F_\parallel} & u_k^1 \\ \downarrow C & & \vdots C^* \\ v_k & & u_k^3 \\ \vdots C^* & & \uparrow C^* \\ w_k & \overset{F_\parallel}{\cdots\!\!\rightarrow} & u_k^2 \end{array}$$

   est obtenu par induction.
   (b) $t = \{u_1, \ldots, \{v_1, \ldots, v_n\}, \ldots, u_m\}$ et $t' = \{u_1, \ldots, v_1, \ldots, v_n, \ldots, u_m\}$.
   Si $u_i \longrightarrow_{F_\parallel} u_i^1$, $v_j \longrightarrow_{F_\parallel} v_j^1$, $i = 1 \ldots m$, $j = 1 \ldots n$, alors, nous avons :

$$\begin{array}{ccc} \{u_1, \ldots, \{v_1, \ldots, v_n\}, \ldots, u_m\} & \xrightarrow{F_\parallel} & \{u_1^1, \ldots, \{v_1^1, \ldots, v_n^1\}, \ldots, u_m^1\} \\ \downarrow C & & \vdots C \\ \{u_1, \ldots, v_1, \ldots, v_n, \ldots, u_m\} & \overset{F_\parallel}{\cdots\!\!\rightarrow} & \{u_1^1, \ldots, v_1^1, \ldots, v_n^1, \ldots, u_m^1\} \end{array}$$

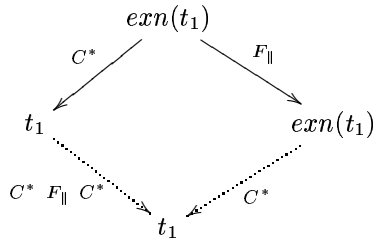
   (c) $t = \{u_1, \ldots, u_k, \bot, u_{k+1}, \ldots, u_m\}$ et $t' = \{u_1, \ldots, u_k, u_{k+1}, \ldots, u_m\}$.Si $u_i \longrightarrow_{F_\parallel} u_i^1$ pour $1 \leqslant i \leqslant m$, alors nous avons :

$$\begin{array}{ccc} \{u_1, \ldots, u_k, \bot, u_{k+1}, \ldots, u_m\} & \xrightarrow{F_\parallel} & \{u_1^1, \ldots, u_k^1, \bot, u_{k+1}^1, \ldots, u_m^1\} \\ \downarrow C & & \vdots C \\ \{u_1, \ldots, u_k, u_{k+1}, \ldots, u_m\} & \overset{F_\parallel}{\cdots\!\!\rightarrow} & \{u_1^1, \ldots, u_k^1, u_{k+1}^1, \ldots, u_m^1\} \end{array}$$

2. t est de la forme $exn(t_1)$
   (a) $t = exn(u)$ et $t' = exn(u')$ avec $u \longrightarrow_C u'$. Nous avons :

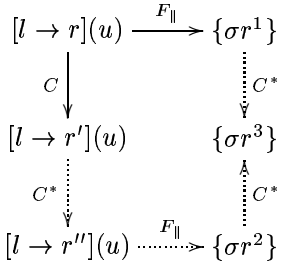
$$\begin{array}{ccc} exn(u) & \xrightarrow{F_\parallel} & exn(u^1) \\ \downarrow C & & \vdots C^* \\ exn(u') & & exn(u^3) \\ \vdots C^* & & \uparrow C^* \\ exn(u'') & \overset{F_\parallel}{\cdots\!\!\rightarrow} & exn(u^2) \end{array} \quad \text{puisque} \quad \begin{array}{ccc} u & \xrightarrow{F_\parallel} & u^1 \\ \downarrow C & & \vdots C^* \\ u' & & u^3 \\ \vdots C^* & & \uparrow C^* \\ u'' & \overset{F_\parallel}{\cdots\!\!\rightarrow} & u^2 \end{array}$$

   s'obtient par induction.
   (b) $t = exn(t_1)$ et $t' = t_1$, alors cela signifie que la règle Exn_c a été appliquée à t et donc que t_1 est Exn-réductible et donc a fortiori, que t_1 ne contient pas d'application. Donc pour la relation $\longrightarrow_{F_\parallel}$, t est en forme normale. On a donc :

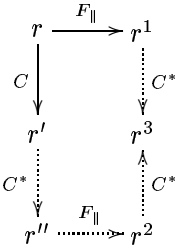


3. $t = \perp$
   alors $t' = s = \perp$ et résultat est trivial.
4. t est de la forme $f(u_1, \ldots, u_n)$ avec $f \in \mathcal{F}$
   (a) $t = f(u_1, \ldots, u_k, \ldots, u_m)$ et $t' = f(u_1, \ldots, v_k, \ldots, u_m)$ avec $u_k \longrightarrow_C v_k$
   Similaire au cas 1a.
   (b) $t = f(u_1, \ldots, \{v_1, \ldots, v_n\}, \ldots, u_m)$, avec $n > 0$
   $t' = \{f(u_1, \ldots, v_1, \ldots, u_m), \ldots, f(u_1, \ldots, v_n, \ldots, u_m)\}$.
   Similaire au cas 1b.
   (c) $t = f(u_1, \ldots, u_k, \perp, u_{k+1}, \ldots, u_m)$, $t' = \{\perp\}$
   Similaire au cas 1c.
5. t est de la forme $u \to v$
   (a) $t = u \to v$, $t' = u' \to v$ avec $u \longrightarrow_C u'$. $u' = u$ puisque u est nécessairement du ε-premier ordre .
   (b) $t = u \to v$, $t' = u \to v'$ avec $v \longrightarrow_C v'$.
   Similaire au cas 1a.
   (c) $t = u \to \perp$ et $t' = \{\perp\}$
   Similaire au cas 1c.
   (d) $t = l \to \{r_1, \ldots, r_n\}$, $t' = \{l \to r_1, \ldots, l \to r_n\}$.
   Similaire au cas 1b.
6. t est de la forme $[v](u)$ et les règles d'évaluation $Fire_c$ et $Congruence$ ne sont pas appliquées à la position de tête
   (a) $t = [v](u)$, $t' = [v'](u)$ avec $v \longrightarrow_C v'$.
   Similaire au cas 1a.
   (b) $t = [v](u)$, $t' = [v](u')$ avec $u \longrightarrow_C u'$.
   Similaire au cas 1a.
   (c) $t = [\{r_1, \ldots, r_n\}](u)$, $t' = \{[r_1](u), \ldots, [r_n](u)\}$ avec $n > 0$
   Similaire au cas 1b.
   (d) $t = [\{\}](u)$, $t' = \{\perp\}$ alors $t \longrightarrow_{F_\|} [\{\}](u^1)$ et $[\{\}](u^1) \xrightarrow{*}_C \{\perp\}$.
   (e) $t = [r](\{u_1, \ldots, u_n\})$, $t' = \{[r](u_1), \ldots, [r](u_n)\}$ avec $n > 0$
   Similaire au cas 1b.
   (f) $t = [u](\{\})$, $t' = \{\perp\}$
   Similaire au cas 6d.
   (g) $t = [v](\perp)$, $t' = \{\perp\}$ alors $t \longrightarrow_{F_\|} [v^1](\perp)$ et $[v^1](\perp) \xrightarrow{*}_C \{\perp\}$.
   (h) $t = [\perp](v)$, $t' = \{\perp\}$
   Similaire au cas 6g.
7. t est de la forme $[l \to r](u)$ avec l, u ρ_ε-calculables.
   (a) $t = [l \to r](u)$, $t' = [l' \to r](u)$ avec $l \longrightarrow_C l'$,
   Si l, u sont ρ_ε-calculables alors $l = l'$ et le lemme est trivialement vrai.
   (b) $t = [l \to r](u)$, $t' = [l \to r'](u)$ avec $r \longrightarrow_C r'$,

Nous considérons les termes r^1, u^1 tels que $r \longrightarrow_{F_\parallel} r^1$, $u \longrightarrow_{F_\parallel} u^1$. Le cas où $(l \ll^?_\emptyset u^1)$ échoue est trivial. Sinon soit σ la substitution telle que $\sigma = \mathit{Solution}(l \ll^?_\emptyset u^1)$. Alors, nous avons :



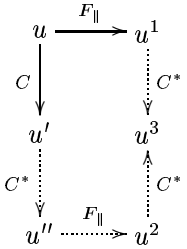
puisque par induction



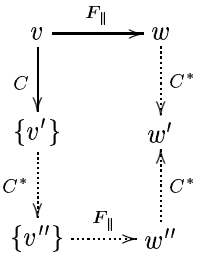
et nous appliquons le Lemme C.15.

(c) $t = [l \to r](u)$, $t' = [l \to r](u')$ avec $u \longrightarrow_C u'$,

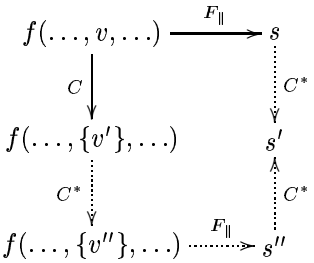
i. Nous considérons d'abord que l, u sont ρ_ε-calculables et l, u' sont ρ_ε-calculables et nous obtenons par induction :



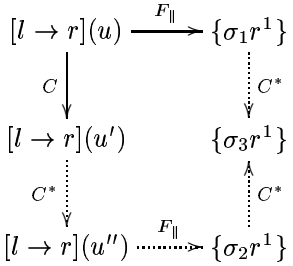
Nous voulons avoir l, u'' ρ_ε-calculables et le seul cas où la propriété n'est pas vraie est obtenu pour les termes de la forme $u = f(\ldots, v, \ldots)$, $u' = f(\ldots, \{v'\}, \ldots)$ et $u'' = \{f(\ldots, v', \ldots)\}$ mais nous avons



et donc



Par conséquent, nous pouvons trouver un terme u'' satisfaisant le diagramme précédent et tel que si l, u' sont ρ_ε-calculables alors l, u'' sont ρ_ε-calculables. Nous considérons $r \longrightarrow_{F_\parallel} r^1$ et en utilisant le Lemme C.14 nous obtenons :



avec $\sigma_1, \sigma_2, \sigma_3$ telles que σ_1 solution de $(l \ll^?_\emptyset u^1)$, σ_2 solution de $(l \ll^?_\emptyset u^2)$ et σ_3 solution de $(l \ll^?_\emptyset u^3)$. Ce cas inclut le cas où $t = [l \to r](exn(t_1))$ et $t' = [l \to r](t_1)$ Si le filtrage $(l \ll^?_\emptyset u^1)$ échoue alors, en utilisant le Lemme C.13 nous obtenons que le filtrage $(l \ll^?_\emptyset u^2)$ échoue et le lemme est clairement vrai.

ii. Nous traitons maintenant le cas où l, u sont ρ_ε-calculables mais l, u' ne sont pas ρ_ε-calculables. Si $t = [l \to r](f(\ldots, \{u\}, \ldots))$ et $t' = [l \to r](\{f(\ldots, u, \ldots)\})$ alors, l doit être une variable ou de la forme $l = f(l_1, \ldots, l_n)$ avec l_i, u_i $(i = 1 \ldots n)$ ρ_ε-calculables. Dans le dernier cas, la position dans le terme l correspondant à la position du terme $\{u\}$ dans le terme $f(\ldots, \{u\}, \ldots)$ est une position variable. Ainsi, dans les deux cas les termes $l, f(\ldots, u, \ldots)$, $i = 1 \ldots n$, sont ρ_ε-calculables.
Nous considérons $r \longrightarrow_{F_\parallel} r^1$, $u \longrightarrow_{F_\parallel} u^1$ et les substitutions σ, σ' telles que nous avons σ solution de $(l \ll^?_\emptyset f(\ldots, \{u^1\}, \ldots))$ et σ' solution de $(l \ll^?_\emptyset f(\ldots, u^1, \ldots))$. Puisque $l, f(\ldots, \{u\}, \ldots)$ sont ρ_ε-calculables, par le Lemme C.8 $l, f(\ldots, \{u^1\}, \ldots)$ sont ρ_ε-calculables et nous obtenons, en utilisant le Lemme C.18,

$$\begin{array}{ccc}
[l \to r](f(\ldots, \{u\}, \ldots)) & \xrightarrow{F_\parallel} & \{\sigma r^1\} \\
\Big\downarrow{\scriptstyle C} & & \\
[l \to r](\{f(\ldots, u, \ldots)\}) & & \Big\downarrow{\scriptstyle C^*} \\
\Big\downarrow{\scriptstyle C} & & \\
\{[l \to r](f(\ldots, u, \ldots))\} & \xrightarrow{F_\parallel} & \{\{\sigma' r^1\}\}
\end{array}$$

Le filtrage $(l \ll^?_\emptyset f(\ldots, \{u^1\}, \ldots))$ peut échouer seulement à cause des symboles fonctionnels différents à la même position des termes l et $f(\ldots, \{u^1\}, \ldots)$ et dans ce cas, le filtrage $(l \ll^?_\emptyset f(\ldots, u^1, \ldots))$ échoue aussi et le lemme est trivialement vrai.

(d) $t = [l \to \{r_1, \ldots, r_n\}](u)$, $t' = [\{l \to r_1, \ldots, l \to r_n\}](u)$.
Si nous considérons $r_i \longrightarrow_{F_\parallel} r_i^1$, $i = 1 \ldots n$, $u \longrightarrow_{F_\parallel} u^1$ et la substitution σ telle que σ solution de $(l \ll^?_\emptyset u^1)$, puisque l, u sont ρ_ε-calculables, nous obtenons le diagramme :

$$\begin{array}{ccc}
[l \to \{r_1, \ldots, r_n\}](u) & \xrightarrow{F_\parallel} & \{\sigma\{r_1^1, \ldots, r_n^1\}\} \\
\Big\downarrow{\scriptstyle C} & & \Big\downarrow{\scriptstyle C^*} \\
[\{l \to r_1, \ldots, l \to r_n\}](u) & & \{\sigma r_1^1, \ldots, \sigma r_n^1\} \\
\Big\downarrow{\scriptstyle C} & & \Big\uparrow{\scriptstyle C^*} \\
\{[l \to r_1](u), \ldots, [l \to r_n](u)\} & \xrightarrow{F_\parallel} & \{\{\sigma r_1^1\}, \ldots, \{\sigma r_n^1\}\}
\end{array}$$

Le cas où $\mathcal{S}olution(l \ll^?_\emptyset u^1) = \emptyset$ est trivial.

(e) $t = [l \to r](\{u\})$, $t' = \{[l \to r](u)\}$.
Nous considérons $u \longrightarrow_{F_\parallel} u^1$, $r \longrightarrow_{F_\parallel} r^1$ et les substitutions σ, σ' telles que σ solution de $(l \ll^?_\emptyset \{u^1\})$ et σ solution de $(l \ll^?_\emptyset u^1)$. Puisque $l, \{u\}$ sont ρ_ε-calculables, par le Lemme C.8 $l, \{u^1\}$ sont ρ_ε-calculables l est une variable et donc, les filtrages $(l \ll^?_\emptyset \{u^1\})$ et $(l \ll^?_\emptyset u^1)$ n'échouent pas. En utilisant le Lemme C.17, nous obtenons le diagramme :

$$\begin{array}{ccc}
[l \to r](\{u\}) & \xrightarrow{F_\parallel} & \{\sigma r^1\} \\
\Big\downarrow{\scriptstyle C} & & \Big\downarrow{\scriptstyle C^*} \\
\{[l \to r](u)\} & \xrightarrow{F_\parallel} & \{\{\sigma' r^1\}\}
\end{array}$$

(f) $t = [l \to \bot](u)$, $t' = [\{\bot\}](u)$ Si le filtrage $(l \ll^?_\emptyset u)$ échoue $t \longrightarrow_{F_\parallel} \{\bot\}$, si le filtrage n'échoue pas, on a aussi $t \longrightarrow_{F_\parallel} \{\bot\}$. On a donc dans les deux cas :

$$\begin{array}{ccc} [l \to \bot](u) & \xrightarrow{F_\parallel} & \{\bot\} \\ \Big\downarrow{\scriptstyle C} & & \vdots{\scriptstyle C^*} \\ [\{\bot\}](u) & \overset{C^*}{\dashrightarrow} & \{\bot\} \end{array}$$

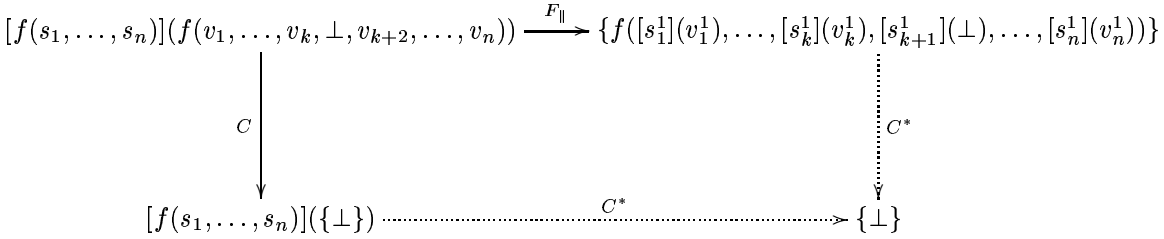
8. t est de la forme $[f(u_1, \ldots, u_n)](f(v_1, \ldots, v_n))$ avec $f \in \mathcal{F}$ et nous appliquons la règle d'évaluation *Congruence* à la position de tête.
   (a) $t = [f(u_1, \ldots, u_n)](f(v_1, \ldots, v_n))$, $t' = [f(u'_1, \ldots, u_n)](f(v_1, \ldots, v_n))$ et $u_1 \longrightarrow_C u'_1$.
   Si $u_i \longrightarrow_{F_\parallel} u_i^1$, $v_i \longrightarrow_{F_\parallel} v_i^1$, $i = 1 \ldots n$, alors, nous avons :

$$\begin{array}{ccc} [f(u_1, \ldots, u_n)](f(v_1, \ldots, v_n)) & \xrightarrow{F_\parallel} & \{f([u_1^1](v_1^1), \ldots, [u_n^1](v_n^1))\} \\ \Big\downarrow{\scriptstyle C} & & \vdots{\scriptstyle C^*} \\ [f(u'_1, \ldots, u_n)](f(v_1, \ldots, v_n)) & & \{f([u_1^3](v_1^1), \ldots, [u_n^1](v_n^1))\} \\ \vdots{\scriptstyle C^*} & & \vdots{\scriptstyle C^*} \\ [f(u''_1, \ldots, u_n)](f(v_1, \ldots, v_n)) & \overset{F_\parallel}{\dashrightarrow} & \{f([u_1^2](v_1^1), \ldots, [u_n^1](v_n^1))\} \end{array}$$

   puisque par induction le diagramme est vrai pour le terme u_1 et ses réduits.
   (b) $t = [f(u_1, \ldots, u_n)](f(v_1, \ldots, v_n))$, $t' = [f(u_1, \ldots, u_n)](f(v'_1, \ldots, v_n))$ et $v_1 \longrightarrow_C v'_1$.
   Similaire au cas 8a.
   (c) $t = [f(s_1, \ldots, \{u_1, \ldots, u_n\}, \ldots, s_m)](f(v_1, \ldots, v_m))$,
   $t' = [\{f(s_1, \ldots, u_1, \ldots, s_m), \ldots, f(s_1, \ldots, u_n, \ldots, s_m)\}](f(v_1, \ldots, v_m))$.
   Pour simplicité, nous supposons que $t = [f(\{u_1, \ldots, u_n\})](f(v))$ et donc, $t' = [\{f(u_1), \ldots, f(u_n)\}](f(v))$. Le cas général est traité exactement de la même façon.
   Si nous considérons $v^1 \longrightarrow_{F_\parallel} v^1$, $u_i^1 \longrightarrow_{F_\parallel} u_i^1$, $i = 1 \ldots n$, nous obtenons :

$$\begin{array}{ccc} [f(\{u_1, \ldots, u_n\})](f(v)) & \xrightarrow{F_\parallel} & \{f([\{u_1^1, \ldots, u_n^1\}](v^1))\} \\ \Big\downarrow{\scriptstyle C} & & \vdots{\scriptstyle C} \\ [\{f(u_1), \ldots, f(u_n)\}](f(v)) & & \{f(\{[u_1^1](v^1), \ldots, [u_n^1](v^1)\})\} \\ \vdots & & \vdots{\scriptstyle C} \\ & & \{\{f([u_1^1](v^1)), \ldots, f([u_n^1](v^1))\}\} \\ \vdots{\scriptstyle C} & & \vdots{\scriptstyle C} \\ & & \{f([u_1^1](v^1)), \ldots, f([u_n^1](v^1))\} \\ \vdots & & \vdots{\scriptstyle C^*} \\ \{[f(u_1)](f(v)), \ldots, [f(u_n)](f(v))\} & \overset{F_\parallel}{\dashrightarrow} & \{\{f([u_1^1](v^1))\}, \ldots, \{f([u_n^1](v^1))\}\} \end{array}$$

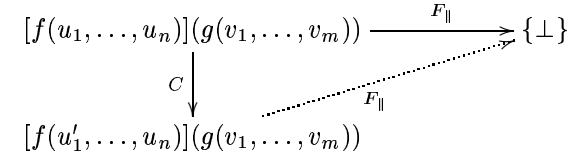
   (d) $t = [f(u_1, \ldots, u_m)](f(s_1, \ldots, \{v_1, \ldots, v_n\}, \ldots, s_m))$,
   $t' = [f(u_1, \ldots, u_m)](\{f(s_1, \ldots, v_1, \ldots, s_m), \ldots, f(s_1, \ldots, v_n, \ldots, s_m)\})$.
   Similaire au cas 8c.
   (e) $t = [f(s_1, \ldots, s_n)](f(v_1, \ldots, v_k, \bot, v_{k+2}, \ldots, v_n))$, $t' = [f(s_1, \ldots, s_n)](\{\bot\})$. Si $s_i \longrightarrow_{F_\parallel} s_i^1$ pour $1 \leqslant i \leqslant n$ et si $v_j \longrightarrow_{F_\parallel} v_j^1$ pour $j \in [\![1, k]\!] \cup [\![k+2, n]\!]$



(f) $t = [f(s_1,\ldots,\bot,\ldots,s_n)](f(v_1,\ldots,v_n))$, $t' = [\{\bot\}](f(s_1,\ldots,s_n))$
Similaire au cas 8e.

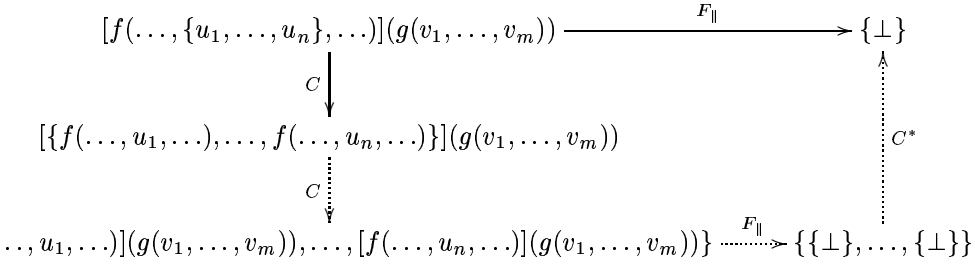
9. t est de la forme $[f(u_1,\ldots,u_n)](g(v_1,\ldots,v_m))$ avec $f,g \in \mathcal{F}$ et nous appliquons la règle d'évaluation *CongruenceFail* à la position de tête.

(a) $t = [f(u_1,\ldots,u_n)](g(v_1,\ldots,v_m))$, $t' = [f(u_1',\ldots,u_n)](g(v_1,\ldots,v_m))$, $u_1 \longrightarrow_C u_1'$.
Nous obtenons immédiatement :



(b) $t = [f(u_1,\ldots,u_n)](g(v_1,\ldots,v_m))$, $t' = [f(u_1,\ldots,u_n)](g(v_1',\ldots,v_m))$, $v_1 \longrightarrow_C v_1'$.
Similaire au cas 9a.

(c) $t = [f(\ldots,\{u_1,\ldots,u_n\},\ldots)](g(v_1,\ldots,v_m))$,
$t' = [\{f(\ldots,u_1,\ldots),\ldots,f(\ldots,u_n,\ldots)\}](g(v_1,\ldots,v_m))$.
Nous obtenons immédiatement :



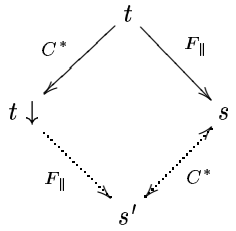
(d) $t = [f(u_1,\ldots,u_n)](g(\ldots,\{v_1,\ldots,v_m\},\ldots))$,
$t' = [f(u_1,\ldots,u_n)](\{g(\ldots,v_1,\ldots),\ldots,g(\ldots,v_m,\ldots)\})$.
Similaire au cas 9c.

(e) $t = [f(s_1,\ldots,s_n)](g(v_1,\ldots,v_k,\bot,v_{k+2},\ldots,v_n))$, $t' = [f(s_1,\ldots,s_n)](\{\bot\})$.
Similaire au cas 8e.

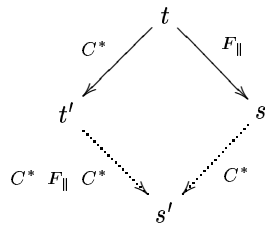
(f) $t = [f(s_1,\ldots,\bot,\ldots,s_n)](g(v_1,\ldots,v_n))$, $t' = [\{\bot\}](g(s_1,\ldots,s_n))$
Similaire au cas 8e.

□

Lemme C.20 *Etant donnés les termes t, s et $t\!\downarrow$, avec $t\!\downarrow$ représentant la forme normale du terme t par rapport à la relation $\xrightarrow{*}_C$. Si $t \longrightarrow_{F_\parallel} s$ alors, il existe un terme s' tel que $t\!\downarrow \longrightarrow_{F_\parallel} s'$ et $s \xleftrightarrow{*}_C s'$:*

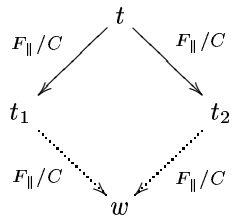


Preuve : Puisque la relation $\longrightarrow_C$ est terminante, nous pouvons utiliser une induction sur le nombre d'étapes dans la réduction $t \stackrel{*}{\longrightarrow}_C t'$ et obtenir, en utilisant le Lemme C.19 :

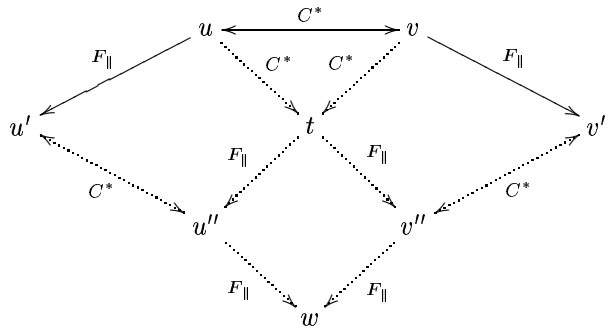


Si $t' = t\downarrow$ alors, t' ne peut pas être réduit en utilisant la relation $\stackrel{*}{\longrightarrow}_C$ et donc, nous obtenons le diagramme du lemme. □

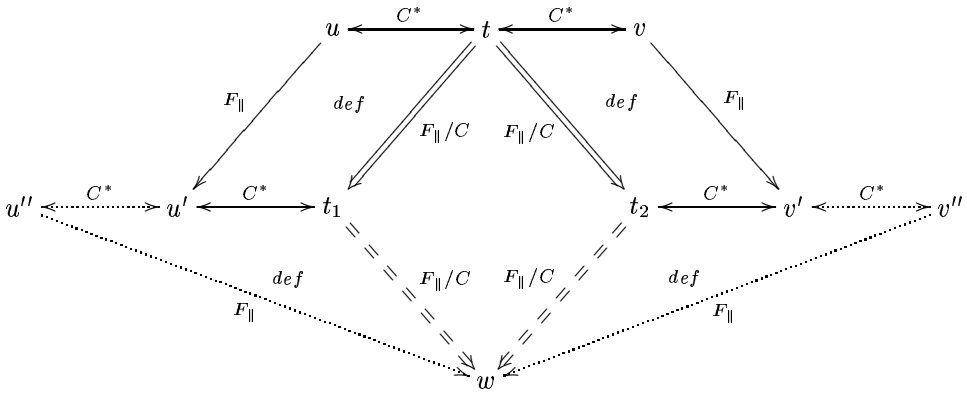
Lemme C.21 *($\longrightarrow_{F_\parallel/C}$ est fortement confluente) Etant donnés les termes t, t_1, t_2 tels que $t \longrightarrow_{F_\parallel/C} t_1$ et $t \longrightarrow_{F_\parallel/C} t_2$. Alors, il existe un terme w tel que $u \longrightarrow_{F_\parallel/C} w$ et $v \longrightarrow_{F_\parallel/C} w$:*



Preuve : Puisque la relation $\longrightarrow_C$ a la propriété de Church-Rosser, si nous considérons deux termes u, v tels que $u \stackrel{*}{\longleftrightarrow}_C v$ alors, le termes u, v ont la même forme normale et nous notons cette forme normale t. Nous utilisons le Lemme C.11 et Lemme C.20 et nous obtenons :



En utilisant ce dernier diagramme et la définition de la relation $\longrightarrow_{F_\parallel/C}$ nous obtenons :



□

Lemme C.22 *La relation $\xrightarrow{*}_{F/C}$ est la fermeture transitive de la relation $\longrightarrow_{F_\parallel/C}$.*

Preuve : Conformément au Lemme C.12 nous avons $\longrightarrow_F \subseteq \longrightarrow_{F_\parallel} \subseteq \xrightarrow{*}_F$. Nous devons prouver que $\longrightarrow_{F/C} \subseteq \longrightarrow_{F_\parallel/C}$ et $\longrightarrow_{F_\parallel/C} \subseteq \xrightarrow{*}_{F/C}$.

Pour l'inclusion $\longrightarrow_{F/C} \subseteq \longrightarrow_{F_\parallel/C}$ il suffit de voir que pour tous termes u', v' tels que $u' \longrightarrow_F v'$ nous avons $u' \longrightarrow_{F_\parallel} v'$ et donc, si

$$\begin{array}{ccc} u & \xrightarrow{F/C} & v \\ {\scriptstyle C^*}\vdots & & \vdots{\scriptstyle C^*} \\ u' & \xrightarrow{F} & v' \end{array} \quad \text{alors} \quad \begin{array}{ccc} u & \xrightarrow{F_\parallel/C} & v \\ {\scriptstyle C^*}\vdots & & \vdots{\scriptstyle C^*} \\ u' & \xrightarrow{F_\parallel} & v' \end{array}$$

Pour $\longrightarrow_{F_\parallel/C} \subseteq \xrightarrow{*}_{F/C}$ nous devons prouver que si

$$\begin{array}{ccc} u & \xrightarrow{F_\parallel/C} & v \\ {\scriptstyle C^*}\vdots & & \vdots{\scriptstyle C^*} \\ u' & \xrightarrow{F_\parallel} & v' \end{array} \quad \text{alors}$$

$$\begin{array}{ccc} u & \xrightarrow{F/C} & u_2 \\ {\scriptstyle C^*}\vdots & & \vdots{\scriptstyle C^*} \\ t_1 & \xrightarrow{F} & t_2 \end{array} \quad \begin{array}{ccc} u_2 & \xrightarrow{F/C} & u_3 \\ {\scriptstyle C^*}\vdots & & \vdots{\scriptstyle C^*} \\ t'_2 & \xrightarrow{F} & t_3 \end{array} \quad \cdots \quad \begin{array}{ccc} u_{n-1} & \xrightarrow{F/C} & v \\ {\scriptstyle C^*}\vdots & & \vdots{\scriptstyle C^*} \\ t'_{n-1} & \xrightarrow{F} & t_n \end{array}$$

Puisque $\longrightarrow_{F_\parallel} \subseteq \xrightarrow{*}_F$, nous pouvons choisir $t_i = t'_i$, $2 \leqslant i \leqslant n-1$ pour prouver l'inclusion.

□

Théorème C.2 *($\xrightarrow{*}_{F/C}$ est fortement confluente, $\longrightarrow_{F/C}$ est confluente) Etant donnés les ρ_ε-termes t, u, v tels que $t \xrightarrow{*}_{F/C} u$ et $t \xrightarrow{*}_{F/C} v$. Alors, il existe un terme w tel que $u \xrightarrow{*}_{F/C} w$ et $v \xrightarrow{*}_{F/C} w$.*

Preuve : Conformément au Lemme C.22, la relation $\xrightarrow{*}_{F/C}$ est la fermeture transitive de la relation $\longrightarrow_{F_\parallel/C}$. Le Lemme C.21 montre la confluence forte de la relation $\longrightarrow_{F_\parallel/C}$. Ainsi, par le Lemme C.6, la relation $\xrightarrow{*}_{F/C}$ est fortement confluente et par conséquent, la relation $\longrightarrow_{F/C}$ est confluente. □

C.4.2 Confluence de la relation $\xrightarrow{*}_C \longrightarrow_F \xrightarrow{*}_C$

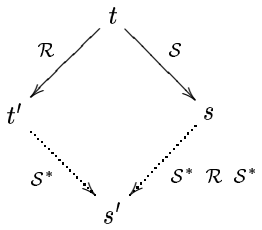
Jusqu'à maintenant nous avons considéré que les règles d'évaluation du ρ-calcul sont soit des règles de *déduction* soit des règles de *calcul* et nous avons analysé la relation induite par les règles de *déduction* modulo la congruence générée par les règles de *calcul*.

Une deuxième approche consiste à considérer la relation habituelle induite par les règles d'évaluation du calcul correspondant à la relation $\xrightarrow{*}_C \longrightarrow_F \xrightarrow{*}_C$. Dans cette section nous nous concentrons sur la confluence de cette relation. La preuve est réalisée en utilisant le Lemme de Yokouchi [YH90].

On rappelle pour cela le Lemme de Yokouchi.

Lemme C.23 *(Yokouchi [YH90])*

Etant donnés deux relations $\longrightarrow_\mathcal{R}$ et $\longrightarrow_\mathcal{S}$ telles que $\longrightarrow_\mathcal{S}$ est confluente et terminante, $\longrightarrow_\mathcal{R}$ est fortement confluente et le diagramme suivant est satisfait :



Alors la relation $\xrightarrow{*}_{\mathcal{S}} \longrightarrow_{\mathcal{R}} \xrightarrow{*}_{\mathcal{S}}$ *est confluente.*

Lemme C.24 *La relation* $\xrightarrow{*}_{C} \longrightarrow_{F_{\|}} \xrightarrow{*}_{C}$ *est fortement confluente.*

Preuve : Les hypothèses pour le Lemme de Yokouchi ont été prouvées dans le Lemme C.3 et le Lemme C.5 (confluence et terminaison de la relation $\longrightarrow_{C}$), le Lemme C.11 (confluence forte de la relation $\longrightarrow_{F_{\|}}$) et le Lemme C.19 (le diagramme de Yokouchi).

□

Théorème C.3 *La relation* $\xrightarrow{*}_{C} \longrightarrow_{F} \xrightarrow{*}_{C}$ *est confluente.*

Preuve : Conformément au Lemme C.22 nous avons :

$$\longrightarrow_{F} \subseteq \longrightarrow_{F_{\|}} \subseteq \xrightarrow{*}_{F}$$

et donc

$$\xrightarrow{*}_{C} \longrightarrow_{F} \xrightarrow{*}_{C} \subseteq \xrightarrow{*}_{C} \longrightarrow_{F_{\|}} \xrightarrow{*}_{C} \subseteq (\xrightarrow{*}_{C} \longrightarrow_{F} \xrightarrow{*}_{C})^{*}$$

où $(\xrightarrow{*}_{C} \longrightarrow_{F} \xrightarrow{*}_{C})^{*}$ représente la fermeture transitive de $\xrightarrow{*}_{C} \longrightarrow_{F} \xrightarrow{*}_{C}$.

Puisque dans le Lemme C.24 nous montrons la confluence forte de la relation $\xrightarrow{*}_{C} \longrightarrow_{F_{\|}} \xrightarrow{*}_{C}$ alors, la relation $\xrightarrow{*}_{C} \longrightarrow_{F} \xrightarrow{*}_{C}$ est confluente.

□

Appendix D

Preuve des lemmes sur la validité du `first`

Lemme D.1 *Si* $\forall i [t_i](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$, *alors* $[MyFirst(t_1,\ldots,t_n)](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$ *et* $[first(t_1,\ldots,t_n)](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$ *où* **`first`** *est défini précisemment ici avec les deux règles* $First', First''$ *présentées dans la section 2.1.*

Preuve : Supposons que $\forall i [t_i](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$. Alors, par $First''$, $[first(t_1,\ldots,t_n)](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$.

Montrons que $[MyFirst(t_1,\ldots,t_n)](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$.

$[MyFirst(t_1,\ldots,t_n)](r) \triangleq [x\{u_i\}](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{< x/r > \{u_i\}\}\ u_1 \triangleq [t_1](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$

Si $u_j = \{\bot\}$ pour tout $j < i$, alors $u_i = \{\bot\}$. En effet, on a la réduction :

$$
\begin{array}{ll}
 & u_i \triangleq [exn(\bot) \to [exn(\bot) \to [\ldots \to [t_i](r)](exn(u_{i-1}))\ldots](exn(u_2)](exn(u_1))) \\
\xrightarrow{*}_{\rho_{\varepsilon v}} & \{[t_i](r)\} \\
\xrightarrow{*}_{\rho_{\varepsilon v}} & \{\bot\}
\end{array}
$$

□

Lemme D.2 *Si* $\exists l$ *tel que* $\forall i \leq l-1\ \ [t_i](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{\bot\}$ *et* $[t_l](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{v_l\} \downarrow \neq \{\bot\}$ *avec* v_l *clos alors* $[MyFirst(t_1,\ldots,t_n)](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{v_l\}\downarrow$ *et* $[first(t_1,\ldots,t_n)](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{v_l\}\downarrow$.

Preuve : On suppose que $\exists l$ tel que $\forall i \leq l-1\ \ [t_i](r) \longrightarrow_{\rho_{\varepsilon v}} \{\bot\}$ et $[t_l](r) \longrightarrow_{\rho_{\varepsilon v}} v_l \neq \{\bot\}$ et v_l clos.

Par $First'$, $[first(t_1,\ldots,t_n)](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{v_l\}$.

Montrons que $[MyFirst(t_1,\ldots,t_n)](r) \xrightarrow{*}_{\rho_{\varepsilon v}} \{v_l\}$.

On a :

$$
\begin{array}{ll}
 & u_1 \triangleq [t_1](r) \\
\longrightarrow_{\rho_{\varepsilon v}} & \{\bot\} \\
 & \vdots \\
 & u_i \triangleq [exn(\bot) \to [exn(\bot) \to [\ldots \to [t_i](r)](exn(u_{i-1}))\ldots](exn(u_2)](exn(u_1))) \\
\xrightarrow{*}_{\rho_{\varepsilon v}} & \{[t_i](r)\} \\
\xrightarrow{*}_{\rho_{\varepsilon v}} & \{\bot\} \\
 & \vdots \\
 & u_{l-1} \triangleq [exn(\bot) \to [exn(\bot) \to [\ldots \to [t_{l-1}](r)](exn(u_{l-2}))\ldots](exn(u_2)](exn(u_1))) \\
\xrightarrow{*}_{\rho_{\varepsilon v}} & \{[t_{l-1}](r)\}
\end{array}
$$

$$
\begin{array}{ll}
\xrightarrow{*}_{\rho_{\varepsilon v}} & \{\bot\} \\
 & \vdots \\
 & u_l \triangleq [exn(\bot) \to [exn(\bot) \to [\ldots \to [t_l](r)](exn(u_{l-1}))\ldots](exn(u_2))](exn(u_1))) \\
\xrightarrow{*}_{\rho_{\varepsilon v}} & \{[t_l](r)\} \\
\xrightarrow{*}_{\rho_{\varepsilon v}} & \{v_l\} \neq \{\bot\} \\
 & u_{l+1} \triangleq [exn(\bot) \to [exn(\bot) \to [\ldots \to [t_{l+1}](r)](exn(u_l))\ldots](exn(u_2))](exn(u_1))) \\
\xrightarrow{*}_{\rho_{\varepsilon v}} & \{\bot\}\ (\textit{échéc du filtrage}) \\
\vdots &
\end{array}
$$

et donc

$$
\begin{array}{ll}
 & [Myfirst(t_1, \ldots, t_n)](r) \\
\xrightarrow{*}_{\rho_{\varepsilon v}} & \{\{\bot\}, \ldots, \{\bot\}, \{v_l\}, \{\bot\}, \ldots, \{\bot\}\} \\
\xrightarrow{*}_{\rho_{\varepsilon v}} & \{v_l\}
\end{array}
$$

□

Bibliography

[AC96] Abadi and Cardelli. *A Theory of Objects*. Springer Verlag, 1996.

[ASU72] A. V. Aho, R. Sethi, and J. D. Ullman. Code optimization and finite Church-Rosser systems. In *Proceedings of Courant Computer Science*, pages 89–105. Prentice Hall, 1972.

[Bar84] H. P. Barendregt. *The Lambda-Calculus, its syntax and semantics*. Studies in Logic and the Foundation of Mathematics. Elsevier Science Publishers B. V. (North-Holland), Amsterdam, 1984. Second edition.

[BKK+96] P. Borovanský, C. Kirchner, H. Kirchner, P.-E. Moreau, and M. Vittek. ELAN: A logical framework based on computational systems. In J. Meseguer, editor, *Proceedings of the first international workshop on rewriting logic*, volume 4 of *Electronic Notes in TCS*, Asilomar (California), September 1996.

[Cir00] H. Cirstea. *Calcul de réécriture : fondements et applications.* Thèse de Doctorat d'Université, Université Henri Poincaré – Nancy 1, France, October 2000.

[CK01a] H. Cirstea and C. Kirchner. The rewriting calculus — Part I. *igpl*, 9(3):427–463, May 2001.

[CK01b] H. Cirstea and C. Kirchner. The rewriting calculus — Part II. *igpl*, 9(3):465–498, May 2001.

[CKL01] H. Cirstea, C. Kirchner, and L. Liquori. Matching power. *RTA2001*, May 2001.

[DHK98] G. Dowek, T. Hardin, and C. Kirchner. Theorem proving modulo. Rapport de Recherche 3400, Institut National de Recherche en Informatique et en Automatique, April 1998. `ftp://ftp.inria.fr/INRIA/publication/RR/RR-3400.ps.gz`.

[Hue76] G. Huet. *Résolution d'equations dans les langages d'ordre 1,2, ...,ω*. Thèse de Doctorat d'Etat, Université de Paris 7 (France), 1976.

[KK99] C. Kirchner and H. Kirchner. Rewriting, solving, proving. A preliminary version of a book available at `www.loria.fr/~ckirchne/rsp.ps.gz`, 1999.

[KKV95] C. Kirchner, H. Kirchner, and M. Vittek. Designing constraint logic programming languages using computational systems. In P. Van Hentenryck and V. Saraswat, editors, *Principles and Practice of Constraint Programming. The Newport Papers.*, chapter 8, pages 131–158. The MIT press, 1995.

[PJ87] S. Peyton-Jones. *The implementation of functional programming languages*. Prentice Hall, Inc., 1987.

[Rob78] M. Robin. A theory of type polymorphism in programming. *JCSS*, 17:348–375, 1978.

[Tur37] A. M. Turing. The φ-functions in λ-K-conversion. *The Journal of Symbolic Logic*, 2:164, 1937.

[Vit94] M. Vittek. ELAN*: Un cadre logique pour le prototypage de langages de programmation avec contraintes*. theseu, nancyUHP, October 1994.

[WL96] Weiss and Leroy. *Le langage Caml.* InterEditions, 1996.

[YH90] H. Yokouchi and T. Hikita. A rewriting system for categorical combinators with multiple arguments. *SIAM Journal of Computing*, 19(1), February 1990.

Contents