Confluence de calcul à motifs

Pierre Caserta

Nancy-Université
Université Henri Poincaré

Université Henri Poincaré – Nancy I

Confluence de calcul à motifs

Rapport

26 Juin 2008

dans le cadre du

Master Recherche option Maîtrise du Logiciel de l'Université Henri Poincaré – Nancy I

par

Pierre Caserta

Membres du Jury

Encadrant:	Horatiu Cirstea
Membres:	N. Carbonnell
	D. Galmiche
	C. Godart
	D. Méry
	G. Perrier

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Remerciements

Je tiens tout d'abord à remercier Horatiu Cirstea de m'avoir donné l'opportunité d'effectuer un stage au sein de l'équipe PAREO et de m'avoir encadré durant ce stage en sachant me donner toutes les indications dont j'avais besoin et en étant toujours disponible, tout en me laissant une grande liberté.

Je remercie tous les membres de l'équipe PAREO pour leur accueil.

Je remercie également Didier Galmiche pour avoir répondu à mes questions concernant le déroulement du stage.

Je remercie enfin tous ceux qui m'ont aidé dans la rédaction de ce rapport.

Table des matières

Remerciements **1**

Introduction **1**

1 Le λ-calcul dynamique à motifs et preuve de la confluence **3**

1.1 Syntaxe et sémantique opérationnelle 3

1.2 Conditions pour la confluence du λ-calcul dynamique à motifs 7

1.2.1 Les conditions que doit respecter la fonction $\mathcal{S}ol$ 8

1.3 Confluence de la relation $\beta_{\shortparallel}$ 11

1.4 Yokouchi-Hikita modulo 15

1.5 Confluence seulement avec des motifs linéaires 17

2 Instances du λ-calcul dynamique à motifs **20**

2.1 Le λ-calcul à motifs avec filtrage unitaire 20

2.2 Le calcul à motifs purs avec filtrage unitaire 21

2.3 Le calcul de réécriture modulo la commutativité 23

2.3.1 Confluence de la sous relation 23

Conclusion et perspectives **27**

Bibliographie **28**

Introduction

La réécriture est un modèle de calcul basé sur la notion de règles. Une règle décrit une transformation réalisée sur des termes. Les notions de règles et de relations de réécriture jouent donc un rôle fondamental dans ce contexte, comme dans celui de la transformation de programmes [Vis05], de la sécurité des protocoles [Rus06], des politiques de contrôles d'accès [HJM06], etc. Ces notions sont aussi particulièrement pertinentes pour manipuler des structures de données vérifiant des invariants [Rei07, HJM06]. Le calcul de réécriture (aussi appellé ρ-calcul) est un concept permettant de modéliser les principes de transformation et de simplification. Il sert en particulier de fondement au calcul symbolique, à la sémantique opérationnelle des langages de programmation, notamment aux langages fonctionnels, et à la déduction automatique.

Toutes ces applications de la réécriture utilisent explicitement ou implicitement la notion de stratégies [Bor, Vis01, KK04, Kir04, MOMV06], c'est-à-dire contrôler la manière dont le calcul évolue. La notion de stratégie s'est imposée universellement dans les langages à base de règles (par exemple Tom, Elan, Maude), certains langages en ayant même fait un véritable paradigme de programmation (Stratego).

Il est particulièrement intéressant de remarquer que l'ingrédient fondamental de la réécriture est le filtrage (défini dans la définition 7), car il permet d'écrire des programmes hautement déclaratifs, comme c'est la cas dans le langage Tom [BBK+06]. Le filtrage est présent dans la plupart de langages fonctionnels et intervient donc dans des applications couvrant des domaines variés, comme par exemple le web [Bal06, Fri06] ou la programmation parallèle [CMV+06]. Cependant, et de manière tout à fait surprenante, le λ-calcul, modèle de calcul particulièrement bien étudié, repose sur une notion triviale de filtrage. Par exemple, S. Peyton-Jones [Pey87] insista sur l'importance d'étudier une généralisation du λ-calcul avec du filtrage, afin de proposer pour les langages fonctionnels un paradigme incluant de manière primitive le filtrage. Parce que le filtrage dans le λ-calcul est trivial, le λ-calcul nécessite de nombreux encodages tant il est vrai qu'il n'est pas adapté pour décrire de manière simple des mécanismes de calcul.

Introduit par H. Cirstea et C. Kirchner [Cir00] pour expliciter les mécanismes de la réécriture et des stratégies, le calcul de réécriture ou ρ-calcul est une généralisation du λ-calcul avec filtrage et agrégation de termes. L'abstraction sur les variables est étendue en une abstraction sur les motifs et ce calcul, bien qu'ayant été introduit pour des motivations différentes, partage avec les autres calculs à base de motifs un certain nombre de points communs que nous expliquerons dans le chapitre 1. Dans sa définition la plus générale, le calcul de réécriture permet l'utilisation du filtrage modulo une théorie équationnelle, utilisant l'agrégation pour collecter les différents résultats possibles. L'agrégation de termes, souvent appelée structure, peut donc être assimilée à un ensemble de terme.

Les calculs à motifs représentent une catégorie de calculs qui permettent de faire des abstractions sur des motifs (et pas seulement sur des variables comme pour le λ-calcul) ce qui est intéressant dans la mesure où cela permet d'augmenter l'expressivité du calcul. Le ρ-calcul fait partie des calculs à motif. Il en existe d'autres tel que le λ-calcul à motifs [vO90] et le calcul à motifs purs [JK06]. Tous ces calculs peuvent être la base de l'implémentation d'un langage de programmation fonctionnel, chacun ayant ces avantages et ces défauts. Par exemple le calcul à motifs purs permet de définir des fonctions génériques (c'est-à-dire qui peuvent s'appliquer quelque soit l'objet sur lequel on l'applique). Tous ces calculs à motifs peuvent être intanciés par un calcul plus général qui est le λ-calcul dynamique à motifs [Fau07]. Le λ-calcul dynamique à motifs est un calcul où les motifs sont dynamiques, c'est-à-dire qu'ils peuvent être instanciés et réduits.

Nous proposons une preuve de confluence générique où la manière dont le filtrage est réalisé est axiomatisée. Nous montrons que cette approche s'applique à différents calculs avec motifs tels que le λ-calcul à motifs, calcul à motifs purs dans le cas d'un filtrage simple et le calcul de réécriture pour un filtrage prennant en compte la commutativité. Nous étudions ensuite la propriété de confluence des calculs avec motifs et nous montrons que cette étude peut être réalisée de manière axiomatique sur les propriétés des algorithmes de filtrage utilisés dans ces calculs. Nous isolons ainsi les propriétés implicitement utilisées dans les différentes preuves de confluence des calculs avec motifs. Ces calculs sont tous confluents sous certaines conditions et dans le cas simple où aucune théorie de filtrage n'est utilisée [Fau07].

Dans ce rapport, nous proposons une preuve de confluence générique qui poura être instanciée pour les différents calculs. Pour cela, nous nous intéresserons au λ-calcul dynamique à motifs qui axiomatise la manière dont l'abstraction est réduite. Nous utilisons cette preuve pour l'étendre au cas où le filtrage est fait modulo une théorie, ici la commutativité. Intuitivement il faut que le filtrage soit stable par substitution, par réduction et par équivalence. Nous caractérisons aussi une classe d'algorithmes de filtrage qui conduisent à des calculs non confluents.

La structure du rapport sera donc naturellement la suivante : nous exposerons dans le premier chapitre le λ-calcul dynamique à motifs. Dans une deuxième chapitre, nous donnerons la preuve de confluence du λ-calcul dynamique à motifs et nous expliciterons la preuve de confluence du λ-calcul à motifs, du calcul à motifs purs et du calcul de réécriture avec filtrage commutatif définis comme des instances du λ-calcul dynamique à motifs. Nous conclurons sur les questions laissées ouvertes et les perspectives de notre travail.

Chapitre 1

Le λ-calcul dynamique à motifs et preuve de la confluence

Le λ-calcul dynamique à motifs est un calcul plus général que les autres calculs à motifs tel que le λ-calcul à motifs, le calcul de réécriture ou le calcul à motifs purs, c'est-à-dire que tous ces calculs peuvent être vus comme des instances du λ-calcul dynamique à motifs. Dans ce chapitre, on va présenter la syntaxe et la sémantique du λ-calcul dynamique à motifs.

1.1 Syntaxe et sémantique opérationnelle

On définit ci-dessous comment sont formé les termes qui compose le λ-calcul dynamique à motifs. Ces termes sont appelé des λ-termes.

Définition 1 (Les termes)**.** *Les termes qui composent le λ-calcul dynamique à motifs sont appelés des λ-termes. L'ensemble des λ-termes est noté Λ.*
On définit un λ-terme T grâce à une grammaire de la façon suivante :

$$\begin{array}{rcll} T & ::= & x & \textit{(Variable)} \\ & | & c & \textit{(Constante)} \\ & | & \lambda_\theta T.T & \textit{(Abstraction)} \\ & | & T\ T & \textit{(Application)} \\ & | & T \wr T & \textit{(Structure)} \end{array}$$

On donne quelques notations supplémentaires concernant les termes.
- des variables : $x, y...$ sont des λ-termes, l'ensemble des variables est noté $\mathbb{V}$
- des constantes : $a, b...$ sont des λ-termes, l'ensemble des constantes est noté $\mathbb{C}$
- des applications : $(A\ B)$ est un λ-terme si A et B sont des λ-termes.
- des abstractions : $\lambda_\theta A.B$ est un λ-terme si A est un λ-terme (aussi appelé motif) et B un λ-terme (appelé corps de l'abstraction). L'ensemble θ est un sous ensemble des variables de de A et représente l'ensemble des variables liées par l'abstraction (voir définition 2)
- des structures (ou agrégation) : $A \wr B$ est un λ-terme qui représente une collection de λ-terme séparés par l'opérateur de structure qui est un opérateur associatif et commutatif.

L'application peut être vue ainsi : si A est une fonction et si V est son argument, alors $(U\ V)$ représente l'application à V de la fonction U. L'abstraction $\lambda_\theta A.B$ peut être interprétée comme la formalisation de la fonction qui à A associe B, où B contient en général des occurrences

de A.
Un λ-terme algébrique est un terme sous la forme : $(\dots((f\ A_1)A_2)\dots)A_n$ où f est une constante et A_i est un λ-terme. Ce terme est également noté $f(A_1, A_2, \dots, A_n)$, un λ-terme algébrique peut être aussi une constante ou une variable.

Exemple 1 (λ-termes)**.**
- *Le terme $\lambda_x xz.xy$ est un λ-terme qui représente une abstraction. Ce λ-terme a pour motif le λ-terme (xy), pour corps le λ-terme (xz) et cette abstraction lie la variable x.*
- *Le terme $\lambda_{x,y} f(x,y).x$ est un λ-terme qui a pour motif le λ-terme $f(x,y)$, pour corps le λ-terme x et cette abstraction lie les λ-termes x et y.*

La notion de variable libre et de variable liée est en fait un moyen de déterminer quelles sont les variables qui vont être remplacées par une étape de réduction. Les variables liées d'une abstraction de la forme $\lambda_\theta P.A$ sont énumérées dans l'ensemble de variables θ, on dit que ces variables sont liées par l'abstraction.

Définition 2 (Variables libres et variables liées)**.** *L'ensemble des variables libres et des variables liées d'un λ-terme, sont définis tel que :*

$\forall c \in \mathbb{C}$	$\mathtt{fv}(c)$	$\triangleq$	$\emptyset$	$\mathtt{bv}(c)$	$\triangleq$	$\emptyset$
$\forall x \in \mathbb{V}$	$\mathtt{fv}(x)$	$\triangleq$	$\{x\}$	$\mathtt{bv}(x)$	$\triangleq$	$\emptyset$
$\forall A, B \in \Lambda, \forall \theta \subseteq \mathbb{V}$	$\mathtt{fv}(\lambda_\theta A.B)$	$\triangleq$	$(\mathtt{fv}(A) \cup \mathtt{fv}(B)) \backslash \theta$	$\mathtt{bv}(\lambda_\theta A.B)$	$\triangleq$	$\mathtt{bv}(A) \cup \mathtt{bv}(B) \cup \theta$
	$\mathtt{fv}(A\ B)$	$\triangleq$	$\mathtt{fv}(A) \cup \mathtt{fv}(B)$	$\mathtt{bv}(A\ B)$	$\triangleq$	$\mathtt{bv}(A) \cup \mathtt{bv}(B)$
	$\mathtt{fv}(A \wr B)$	$\triangleq$	$\mathtt{fv}(A) \cup \mathtt{fv}(B)$	$\mathtt{bv}(A \wr B)$	$\triangleq$	$\mathtt{bv}(A) \cup \mathtt{bv}(B)$

Définition 3 (α-conversion)**.** *Deux λ-termes qui ne diffèrent que par un renommage de leurs variables liées sont dis α-convertibles. L'α-conversion est une relation d'équivalence entre λ-termes.*

Exemple 2.
- $(\lambda_x x.xy) \equiv \lambda_z z.za$
- *l'α-conversion est parfois nécéssaire avant d'effectuer certaines substitutions ainsi le λ-terme $(\lambda_x x.xy)\{y \leftarrow x\}$ doit subir un renommage de la variable x avant d'effectuer la substitution. Le résultat est donc $(\lambda_z z.zx)$ (et non $(\lambda_x x.xx)$, qui est totalement différent).*

Remarque : l'α-conversion doit être définie avec précaution avant la substitution. Ainsi dans le terme $\lambda_x x.\lambda_y y.xy\ \lambda_z z.z$, on ne pourra pas renommer x en y sans α-conversion préalable (on obtiendrait $\lambda_y y.\lambda_y y.yy\ \lambda z.z$) en revanche on peut renommer x en z pour obtenir $\lambda_z z.\lambda_y y.zy\ \lambda_z z.z$.

Dans ce rapport tous les calculs décrit respectent la convention d'hygyène de Barendregt [Bar84], stipulant que les variables libres et les variables liées ont des noms différents et que toutes les variables liées de même nom sont liées par le même lieurs.

Une substitution permet de remplacer une variable x par un λ-terme A, dans un λ-terme A. Par exemple le λ-terme B contient la variable x, la substitution σ est définie de manière à remplacer toutes les variables x par le λ-terme A. Si on applique la substitution σ à B (noté $B\sigma$) alors toutes les occurences de x dans B sont remplacées par A.

Définition 4 (Substitution)**.** *Une substitution est une fonction partielle de $\mathbb{V}$ dans Λ (nous utilisons la notation postfixée).*

On note $\sigma = \{x_1 \leftarrow A_1, \ldots, x_n \leftarrow A_n\}$ la substitution qui à chaque variable x_i associe le terme A_i. L'ensemble $\{x_1, \ldots, x_n\}$ est appelé le domaine de σ et est noté $\mathcal{D}om(\sigma)$. $\mathcal{R}an(\sigma)$ représente l'union des ensembles $\mathtt{fv}(x\sigma)$ où $x \in \mathcal{D}om(\sigma)$. La composition de deux substitutions σ et τ est notée $\sigma \circ \tau$. Elle est définie telle que $x(\sigma \circ \tau) = (x\tau)\sigma$. On note $\mathtt{id}$ la substitution vide. La restriction d'une substitution σ à un ensemble de variables θ est noté $\sigma_{|\theta}$.

L'application d'une substitution σ à un terme A est inductivement définie par :

$$\forall x, y \in \mathbb{V}, A_1, A_2 \in \Lambda$$

$$\begin{array}{lcll} x\sigma & \triangleq & A & \text{si } \sigma = \{\ldots, x \leftarrow A, \ldots\} \\ y\sigma & \triangleq & y & \text{si } y \notin \mathcal{D}om(\sigma) \\ (\lambda_\theta A_1.A_2)\sigma & \triangleq & \lambda_\theta(A_1\sigma).(A_2\sigma) & \text{si } \mathcal{D}om(\sigma) \cap \mathcal{V}ar(A_1) = \emptyset \\ (A_1\ A_2)\sigma & \triangleq & (A_1\sigma)\ (A_2\sigma) & \\ (A_1 \wr A_2)\sigma & \triangleq & (A_1\sigma) \wr (A_2\sigma) & \end{array}$$

Définition 5 (Les théories de filtrage)**.** *On appelle théorie de filtrage par des axiomes sous forme de règles d'inférence :*

$$\frac{E_1 \qquad \ldots \qquad E_n}{E}$$

où $E_1, \ldots, E_n, E$ sont des équations.

Exemple 3. *Nous donnons ci-dessous des exemples de théories $\mathbb{T}$ définis de manière équationnelle. [CKL01]*

– La théorie vide $\mathbb{T}_\emptyset$ de l'égalité (comme α-conversion) est définie par les règles suivantes :

$$\frac{t_1 = t_2 \qquad t_2 = t_3}{t_1 = t_3}\ (Tra) \qquad \frac{t_1 = t_2}{t_2 = t_1}\ (Sym) \qquad \frac{t_1 = t_2}{t_3[t_1]_p = t_3[t_2]_p}\ (Ctx) \qquad \frac{}{t = t}\ (Ref)$$

où $t_1, \ldots, t_n$ sont des termes et $t_1[t_2]_p$ représente le terme t_1 avec le terme t_2 à la position p.

– la théorie de la commutativité $\mathbb{T}_{C(f)}$ (resp. Associativité $\mathbb{T}_{A(f)}$) est définie comme $\mathbb{T}_\emptyset$ plus les règles suivantes :

$$\frac{}{f(t_1\ t_2) = f(t_2\ t_1)}\ (Com) \qquad \frac{}{f(f(t_1\ t_2)\ t_3) = f(t_1\ f(t_2\ t_3))}\ (Ass)$$

– la théorie de l'idempotence $\mathbb{T}_{I(f)}$ est définie comme $\mathbb{T}_\emptyset$ plus l'axiome $f(t\ t) = t$.

On a besoin de définir une relation d'équivalence entre deux λ-termes. Cette relation d'équivalence est défini modulo une théorie de filtrage.

Définition 6 (Congruence entre les termes)**.** *Les λ-termes sont considérés modulo une relation d'équivalence qui inclus l'α-conversion. Cette relation est notée $\sim_{\mathbb{T}}$ où $\mathbb{T}$ est une théorie de filtrage tel que l'associativité, la commutativité ou d'autres théories (voir définition5).*

Par exemple, si on considère la commutativité alors on a $a+b \sim_C b+a$ mais $a+b \not\equiv b+a$. Remarque : La relation d'équivalence $\sim_{\mathbb{T}_\emptyset}$ est identique à la relation d'équivalence syntaxique $\equiv$ modulo l'α-conversion.

L'évaluation d'une réduction dans le λ-calcul dynamique à motifs dépend beaucoup de l'opération de filtrage qui est fondamentale. Quand on applique une étape de réduction, cette

opération remplace les variables liées par l'abstraction par leurs valeurs correspondantes. Le λ-calcul dynamique à motifs permet de faire des abstractions sur des motifs (c'est-à-dire élément placé après le symbole λ) et le filtrage peut être fait syntaxiquement ou modulo une théorie (voir définition 5). On doit définir une relation de congruence permettant d'avoir des motifs filtrable modulo une théorie qui est un paramètre important du calcul. Nous allons d'abord définir l'opération de filtrage et donner ensuite quelques exemples d'algorithme de filtrage pour différents ensembles de motif.

Définition 7 (Filtrage)**.** *Si $\mathbb{T}$ est une théorie définie par un ensemble d'axiome alors le filtrage est définit de la façon suivante :*

- *une équation de filtrage (aussi appelée problème de filtrage) est une paire notée $P \lessdot^{\mathbb{T}} A$ où P un motif (c'est-à-dire un terme algébrique), A un λ-terme.*
- *une substitution σ est une solution du problème de filtrage $P \lessdot^{\mathbb{T}} A$ ssi $P\sigma \sim_{\mathbb{T}} A$.*

L'ensemble des solutions de $P \lessdot^{\mathbb{T}} A$ est noté : $\mathcal{S}ol(P \lessdot^{\mathbb{T}} A)$.

Exemple 4 (Filtrage syntaxique)**.** *On décompose tout problème de filtrage $P \lessdot^{\mathbb{T}} A$ en une conjonction de contraintes de filtrages notées $P_i \ll_{\mathbb{T}} A_i$. Cette notation est utilisée lorsqu'on décrit l'algorithme de résolution du filtrage. Nous considérons des contraintes de filtrage de la forme $A \ll^? B$ et nous autorisons les conjonctions et disjonctions vides avec l'opérateur associatif et commutatif $\wedge$ (avec pour élément neutre* `id`*). Les règles suivantes terminent et sont confluentes. Elles peuvent être utilisées pour résoudre des filtrages syntaxiques (non linéaires) :*

$$\begin{array}{lcl} A \ll^? A \wedge M & \to & M \\ (A_1\ A_2 \ll^? B_1\ B_2) \wedge M & \to & (A_1 \ll^? B_1) \wedge (A_2 \ll^? B_2) \wedge M \end{array}$$

On décrit ci-dessus les différents comportements de la fonction $\mathcal{S}ol$:

- *$\mathcal{S}ol(P \lessdot A) = \{\sigma\}$ avec $\sigma = \{x_i \leftarrow A_i\}_{i\in I}$ si le résultat est de la forme $\bigwedge_{i\in I\neq\emptyset}(x_i \ll^? A_i)$ avec $A_i = A_j$ si $x_i = x_j$*
- *$\mathcal{S}ol(P \lessdot A) = \{$*`id`*$\}$ si le résultat une conjonction vide*
- *$\mathcal{S}ol(P \lessdot A) = \emptyset$ sinon*

Exemple 5 (Filtrage commutatif)**.** *Soit f un symbole binaire qui vérifie la commutativité, on a :*

$$\forall x, \forall y \quad f\ x\ y = f\ y\ x$$

L'algorithme de filtrage syntaxique peut être étendu avec les règles suivantes :

$$\begin{array}{lcl} (f(A_1\ A_2) \ll_C f(B_1\ B_2)) \wedge M & \to & (A_1 \ll_C B_1) \wedge (A_2 \ll_C B_2) \wedge M \\ & & \vee \\ & & (A_1 \ll_C B_2) \wedge (A_2 \ll_C B_1) \wedge M \end{array}$$

N'importe quelle contrainte de filtrage $P \ll^? A$ respecte les trois comportements cités ci-dessous modulo l'associativité et la commutativité du $\vee$ et du $\wedge$, on obtient ainsi le résultat suivant :

1. $\mathcal{S}ol(P \lessdot A) = \{\sigma_j\}_{j\in J}$ avec $\sigma_j = \{x_i \leftarrow A_i\}_{i\in I_j}$ si la forme du résultat est disjonctive alors $\bigvee_{j\in J}\bigwedge_{i\in I_j}(x_i \ll^? A_i)$ avec $A_i = A_k$ si $x_i = x_k$, et $\sigma_j =$ `id` si $I_j = \emptyset$
2. $\mathcal{S}ol(P \lessdot A) =$ {`id`} si le résultat est une conjonction d'ensemble vide
3. $\mathcal{S}ol(P \lessdot A) = \emptyset$ sinon.

Définition 8 (Sémantique)**.** *Les réductions ont pour but de modifier les λ-termes, cette étape s'appelle une β-réduction. Le λ-calcul dynamique à motifs n'a qu'une seule règle de réduction qui définit la manière dont l'abstraction est appliquée. Cette règle est donnée ci-dessous, elle est paramétrable par une fonction, notée $\mathcal{S}ol(A \lessdot_\theta B)$ où A représente le motif, B représente l'argument et θ représente l'ensemble des variables à considérer par le filtrage. La fonction $\mathcal{S}ol$ retourne un ensemble qui représente la solution du filtrage entre A et B (voir définition 7). On définit la β-réduction de la façon suivante :*

$$(\beta) \quad (\lambda_\theta A.B)C \quad \to \quad B\sigma_1 \wr \ldots \wr B\sigma_n \quad \text{où } \{\sigma_1, \ldots, \sigma_n\} = \mathcal{S}ol(A \lessdot_\theta C)$$

Différentes instances du λ-calcul dynamique à motifs sont obtenues lorsque l'on donne une définition concrète de la fonction $\mathcal{S}ol$.

Exemple 6 (Instance du λ-calcul grâce au λ-calcul dynamique à motifs)**.** *. Le λ-calcul peut être vu comme le λ-calcul dynamique à motifs avec*

$$\sigma = \mathcal{S}ol(A \lessdot_\theta C) \text{ si } A \text{ est une variable } x,\ \theta = \{x\} \text{ et } A\sigma \equiv C$$

Dans la plupart des cas la fonction correspond à un algorithme de filtrage du motif mais il peut être encore plus général [HLL07, KKM07].

Exemple 7 (Filtrage)**.** *Dans cet exemple on remarque que l'ensemble des variables à considérer pour le filtrage est constitué uniquement de la variable x, ainsi le résultat du filtrage ne contient que la variable x.*

$$\mathcal{S}ol\Big((a+b) \lessdot_x^{\emptyset} (x+y)\Big) = \{x \leftarrow a\}$$

Le résultat de la fonction $\mathcal{S}ol$ filtrant modulo une théorie équationnelle peut renvoyer plusieurs solutions et donc plusieurs ensembles de substitutions regroupés dans un même ensemble.

1.2 Conditions pour la confluence du λ-calcul dynamique à motifs

Le λ-calcul dynamique à motifs n'est pas confluent si aucune restriction n'est imposée sur la fonction $\mathcal{S}ol$. Par exemple, le terme $(\lambda_x(x\ a).x)\ ((\lambda_y y.y)\ a)$ où la variable x qui est active (c'est-à-dire pas de sous-terme de la forme xA où x est libre) dans $(x\ a)$ se réduit en $\lambda_y y.y$ ou à $(\lambda_x(x\ a).x)\ a$ qui ne sont pas joignable.

La confluence est obtenue uniquement lorsque les motifs sont linéaires (voir section 1.5).

Néanmoins, la confluence est définie pour quelques définitions spécifiques de la fonction $\mathcal{S}ol$ comme par exemple dans l'exemple 6 lorsqu'on définit le λ-calcul grâce au λ-calcul dynamique à motifs. Dans cette section nous allons donner les conditions nécessaires pour garantir la confluence du λ-calcul dynamique à motifs.

Définition 9 (La réduction β-parallèle)**.** *La réduction β-parallèle (noté $\twoheadrightarrow$) est inductivement définie sur l'ensemble des λ-termes de la façon suivante :*

$$\frac{}{A \twoheadrightarrow A} \qquad \frac{A \twoheadrightarrow A' \qquad B \twoheadrightarrow B'}{AB \twoheadrightarrow A'B'} \qquad \frac{A \twoheadrightarrow A' \qquad B \twoheadrightarrow B'}{\lambda_\theta A.B \twoheadrightarrow \lambda_\theta A'.B'}$$

$$\frac{A \twoheadrightarrow A' \qquad B \twoheadrightarrow B' \qquad C \twoheadrightarrow C'}{(\lambda_\theta A.B)C \twoheadrightarrow B'\sigma'_1 \wr \ldots \wr B'\sigma'_n} \text{ IF } \{\sigma'_1, \ldots, \sigma'_n\} = \mathcal{S}ol(A' \lessdot_\theta C')$$

De plus, nous remarquons que la définition de la réduction parallèle ne coïncide pas avec la définition classiques des notions de développement. Par exemple, si on utilise la fonction $\mathcal{S}ol$ qui calcule les substitutions qui satisfont le problème de filtrage entre deux arguments (comme dans l'exemple 4 mais sans utiliser la dernière règle) alors on a :

$$\frac{f\,x \twoheadrightarrow f\,x \qquad x \twoheadrightarrow x \qquad (\lambda y.f\,y)\,a \twoheadrightarrow f\,a}{(\lambda(f\,x).x)((\lambda y.f\,y)\,a) \twoheadrightarrow a}$$

La substitution $\{x \leftarrow a\}$ résout le problème de filtrage syntaxique entre les termes $f\,x$ et $f\,a$ et même si le terme initial ne contient aucune abstraction, il peut quand même être réduit en utilisant la réduction parallèle. On étend la définition de la réduction parallèle à des substitutions qui ont le même domaine en définissant $\sigma \twoheadrightarrow \sigma'$ si pour tout x dans le domaine de σ, nous avons $x\sigma \twoheadrightarrow x\sigma'$.

Définition 10 (Relation d'équivalence)**.** *La relation d'équivalence est inductivement définie sur l'ensemble des λ-termes comme ceci :*

$$\frac{}{A \sim_{\mathbb{T}} A} \qquad \frac{A \sim_{\mathbb{T}} A' \qquad B \sim_{\mathbb{T}} B'}{AB \sim_{\mathbb{T}} A'B'} \qquad \frac{A \sim_{\mathbb{T}} A' \qquad B \sim_{\mathbb{T}} B'}{\lambda_\theta A.B \sim_{\mathbb{T}} \lambda_\theta A'.B'}$$

$$\frac{A \sim_{\mathbb{T}} B}{B \sim_{\mathbb{T}} A} \qquad \frac{A \sim_{\mathbb{T}} B \qquad B \sim_{\mathbb{T}} C}{A \sim_{\mathbb{T}} C} \qquad \frac{A \sim_{\mathbb{T}} A' \qquad B \sim_{\mathbb{T}} B' \qquad C \sim_{\mathbb{T}} C'}{(\lambda_\theta A.B)C \sim_{\mathbb{T}} (\lambda_\theta A'.B')C'}$$

Remarque : Deux termes équivalents modulo une théorie sont stables par substitution [KK99].

$$\frac{x \sim_{\mathbb{T}} y}{\sigma(x) \sim_{\mathbb{T}} \sigma(y)}$$

Définition 11 (Cohérence entre deux relation)**.** *Deux relation sont cohérentes modulo une relation d'équivalence $\sim_{\mathbb{T}}$ si : $\forall A, B, A' \in \Lambda$ tels que $A \to_\delta B$ et $A \sim_{\mathbb{T}} A'$, $\exists B' \in \Lambda$ tel que $B \sim_{\mathbb{T}} B'$ et $A' \to_\delta B'$.*

$$\begin{array}{ccc} A & \sim_{\mathbb{T}} & A' \\ {\scriptstyle\delta}\downarrow & & \vdots{\scriptstyle\delta} \\ B & \sim_{\mathbb{T}} & B' \end{array}$$

1.2.1 Les conditions que doit respecter la fonction $\mathcal{S}ol$

La fonction $\mathcal{S}ol$ est celle qui permet de renvoyer les solutions d'un filtrage. Si aucune condition n'est impossée sur cette fonction le λ-calcul dynamique à motifs n'est pas confluent.

$\forall A, C, A', C' \in \Lambda$

$$\mathbf{H_0}: \quad \mathcal{S}ol(A \lessdot_\theta C) = \{\sigma_i\}_{i=1}^n \implies \begin{cases} \forall i & \mathcal{D}om(\sigma_i) = \theta \\ & \mathcal{R}an(\sigma_i) \subseteq \mathtt{fv}(C) \end{cases}$$

$$\mathbf{H_1}: \quad \mathcal{S}ol(A \lessdot_\theta C) = \{\sigma_i\}_{i=1}^n \implies \begin{cases} \forall \tau, \mathcal{V}ar(\tau) \cap \theta = \emptyset, \\ \mathcal{S}ol(A\tau \lessdot_\theta C\tau) = \{(\tau \circ \sigma_i)_{|\theta}\}_{i=1}^n \end{cases}$$

$$\mathbf{H_2}: \quad \begin{cases} \mathcal{S}ol(A \lessdot_\theta C) = \{\sigma_i\}_{i=1}^n \\ A \twoheadrightarrow A' \qquad C \twoheadrightarrow C' \end{cases} \implies \begin{cases} \mathcal{S}ol(A' \lessdot C') = \{\sigma'_j\}_{j=1}^n \\ \forall i, j \ \sigma_i \twoheadrightarrow \sigma'_j \end{cases}$$

$$\mathbf{H_3}: \quad \begin{cases} \mathcal{S}ol(A \lessdot_\theta C) = \{\sigma_i\}_{i=1}^n \\ A \sim_{\mathbb{T}} A' \qquad C \sim_{\mathbb{T}} C' \end{cases} \implies \begin{cases} \mathcal{S}ol(A' \lessdot C') = \{\sigma'_j\}_{j=1}^n \\ \forall i, j \ \sigma_i \sim_{\mathbb{T}} \sigma'_j \end{cases}$$

FIGURE 1.1 – Les conditions pour avoir la confluence du λ-calcul dynamique à motifs

Préservation des variables libres

La première condition, est que l'ensemble des variables libres doit être préservé par la réduction (certaines variables libres peuvent disparaître mais aucune ne peut apparaître durant la réduction). Par exemple, les variables libres du terme $(\lambda_\theta A.B)C$ doivent inclure celles de $B\sigma$ avec $\sigma = \mathcal{S}ol(A \lessdot_\theta C)$. De plus, la substitution σ doit instancier toutes les variables liées (par l'abstraction) dans B, qui sont en réalité les variables dans l'ensemble θ. De plus, les variables libres de σ doivent toujours être présentes dans C (ou libres dans A). Cette condition est réalisée par l'hypothèse $\mathbf{H_0}$ figure 1.1.

Exemple 8. *Si on considère la fonction $\mathcal{S}ol$ comme un algorithme de filtrage unitaire, l'exemple qui ne vérifie pas $\mathbf{H_0}$ serait un algorithme particulier. (par exemple la fonction $\mathcal{S}ol$ qui retourne la substitution $\{x \leftarrow y\}$ pour tout problème).*
Quand on considère un filtrage non unitaire il y a beaucoup d'exemples qui ne vérifient pas $\mathbf{H_0}$. Par exemple, l'algorithme qui résout le filtrage d'ordre supérieur ou le filtrage avec des théories non régulières [KK99] (c'est-à-dire comme par exemple $x \times 0 = 0$) ne vérifie pas $\mathbf{H_0}$.

Stable par substitution

Dans le λ-calcul dynamique à motifs, quand une abstraction est appliquée l'argument utilisé peut être réduit. Une abstraction peut attendre qu'un argument se réduise pour être instancié. Après avoir calculé la substitution correspondante (si elle existe) on réduit l'application. D'un autre coté, si on veut pouvoir effectuer les réductions en parallèle, on doit obtenir le même résultat si la réduction a été faite séquenciellement. Cette condition est donnée en hypothèse $\mathbf{H_1}$ dans la figure 1.1.

Exemple 9. *Si on considère que $\mathcal{S}ol$ définit un algorithme de filtrage naïf qui ne prend pas en compte les variables dans θ tel que $\mathcal{S}ol(a \lessdot_\emptyset b)$ n'a pas de solution et $\mathcal{S}ol(x \lessdot_\emptyset y) = \{x \leftarrow y\}$, alors l'hypothèse $\mathbf{H_1}$ est clairement pas vérifié (si on prend $\tau = \{x \leftarrow a, y \leftarrow b\}$).*
Si on a une réduction de la forme $(\lambda_\theta(A\tau).B)(C\tau) \to (B(\tau \circ \sigma_1)) \wr \ldots \wr (B(\tau \circ \sigma_n))$ où

$\{(\tau \circ \sigma_1), \ldots, (\tau \circ \sigma_n)\} = \mathcal{S}ol(A\tau \prec_\theta C\tau)$ *on a bien* $\mathcal{S}ol(A\tau \prec_\theta C\tau) = \{(\tau \circ \sigma_i)_{|\theta}\}_{i=1}^n$ *si la condition* $\mathcal{V}ar(\tau) \cap \theta = \emptyset$ *est respecté.*

Stable par réduction

Quand on applique une abstraction, les arguments peuvent ne pas être entièrement réduits. Si $\mathcal{S}ol$ produit une substitution σ alors la sous séquence de réduction ne devrait pas mener vers un échec de la fonction $\mathcal{S}ol$. De plus, la substitution qui est éventuellement obtenue devrait être dérivable de σ. Cette condition est définie formellement dans l'hypothèse $\mathbf{H_2}$ dans la figure 1.1.

Exemple 10. *La fonction* $\mathcal{S}ol$ *utilisant un filtrage syntaxique (voir exemple 4) ne satisfait pas cette hypothèse. Si nous prenons* $I \triangleq (\lambda y.y)$ *alors* $\mathcal{S}ol(f(x,x) \prec_x f(I\,I, I\,I)) = \{x \leftarrow I\,I\}$ *mais* $\mathcal{S}ol(f(x,x) \prec_x f(I\,I, I))$ *n'a pas de solution. De la même façon, cette hypothèse n'est pas satisfaite par l'algorithme de filtrage qui autorise la décomposition d'une application qui contient une variable libre active. Par exemple, nous pouvons avoir* $\mathcal{S}ol(x\,a \prec_x (\lambda y.y)a) = \{x \leftarrow \lambda y.y\}$ *mais* $\mathcal{S}ol(x\,a \prec_x a)$ *n'a pas de solution pour tous les algorithmes de filtrage du premier ordre.*

Stable par équivalence

Lorsqu'on introduit une relation d'équivalence $\sim_{\mathbb{T}}$, la fonction $\mathcal{S}ol$ doit prendre en compte et retourner les solutions équivalentes pour un problème de filtrage. Cette propriété est formellement définie dans l'hypothèse $\mathbf{H_3}$. Intuitivement, cette propriété garantie que la relation de réduction est compatible avec la relation d'équivalence.

Exemple 11. *Si on suppose que le symbole* $\times$ *satisfait l'axiome* $x \times 0 = 0$ *alors la fonction* $\mathcal{S}ol$ *définie avec un filtrage syntaxique ne vérifie pas* $\mathbf{H_3}$. *Dans ce cas, nous avons* $\mathcal{S}ol(x \times y \prec_{x,y} a \times 0) = \{\ \{x \leftarrow a, y \leftarrow 0\}\ \}$ *mais* $\mathcal{S}ol(x \times y \prec_{x,y} 0)$ *n'a pas de solution.*

L'axiome qui définit la théorie est $(x \times 0) = 0$

$$\begin{array}{ccc} (\lambda_{x,0}(x \times y).0)(a \times 0) & \sim & (\lambda_{x,0}(x \times y).0)0 \\ {\scriptstyle\beta}\downarrow & & \\ 0 & & \end{array}$$

Dans cette section, nous allons d'abord prouver la confluence du λ-calcul dynamique à motifs sous les hypothèses $\mathbf{H_0}, \mathbf{H_1}, \mathbf{H_2}, \mathbf{H_3}$. La preuve utilise les techniques standards de réduction parallèle introduites par Tait et Martin-Löf. Nous allons d'abord montrer que la clôture transitive et réflexive de la réduction parallèle et une étape de réduction sont les mêmes. Nous montrerons ensuite que la réduction parallèle a la propriété du diamant, c'est-à-dire pour tout terme A, B et C si A se réduit en une étape en B et A se réduit en une étape en C alors il existe un terme D tel que B se réduit en une étape en D et C se réduit en une étape en D, et nous en déduirons la confluence de chaque étape de réduction.

Les quatres hypothèses données dans la section précédente sont utilisées et la preuve pour montrer que la réduction parallèle a la propriété du diamant utilise en particulier les lemmes 2 et 3. Par ailleurs, nous pouvons montrer que la clôture transitive et réflexive de la relation $\twoheadrightarrow$ est égale à $\rightarrow_\beta^*$ indépendamment des propriétés de $\mathcal{S}ol$.

1.3 Confluence de la relation $\beta_{||}$

Lemme 1. *L'inclusion suivante est vérifiée.* $\rightarrow_\beta \subseteq \twoheadrightarrow \subseteq \rightarrow_\beta^*$.

Démonstration. Pour la première inclusion on suppose que la réduction $A \rightarrow_\beta B$ apparaît en tête d'un terme. Ce qui veut dire que $A \equiv (\lambda_\theta A_1.A_2)A_3$ et $B \equiv A_2\{\sigma_i\}_{i=1}^n$ où $\{\sigma_i\}_{i=1}^n = \mathcal{S}ol(A_1 \lessdot_\theta A_3)$. On a $A \twoheadrightarrow B$ car la relation $\twoheadrightarrow$ est réflexive. Dans les autres cas (où la réduction n'a pas lieu en position de tête), la preuve est déduite du fait que la relation $\twoheadrightarrow$ est compatible (c'est-à-dire si $U \twoheadrightarrow V$ avec $U, V \in \Lambda$ alors $W\ U \twoheadrightarrow W\ V$ avec $W \in \Lambda$). Pour la seconde inclusion, on prouve par induction que si $A \twoheadrightarrow B$ alors $A \rightarrow_\beta^* B$.

- Si $A \equiv B$ alors le résultat est déduit du fait que la relation $\rightarrow_\beta^*$ est réflexive.
- Si $A \equiv A_1A_2$ et $B \equiv B_1B_2$ avec $A_1 \twoheadrightarrow B_1$ et $A_2 \twoheadrightarrow B_2$ alors par hypothèse d'induction appliquée à A_1 et A_2 on a, $A_1 \rightarrow_\beta^* B_1$ et $A_2 \rightarrow_\beta^* B_2$. on conclut que $A_1A_2 \rightarrow_\beta^* B_1B_2$.
- Si $A \equiv \lambda_\theta A_1.A_2$ et $B \equiv \lambda_\theta B_1.B_2$ avec $A_1 \twoheadrightarrow B_1$ et $A_2 \twoheadrightarrow B_2$ on procède comme le cas précédent.
- Si $A \equiv (\lambda_\theta A_1.A_2)A_3$ et $B \equiv B_2\sigma_1 \wr \ldots \wr B_2\sigma_n$ avec $A_1 \twoheadrightarrow B_1$ et $A_2 \twoheadrightarrow B_2$ et $A_3 \twoheadrightarrow B_3$ et $\{\sigma_1, \ldots, \sigma_n\} = \mathcal{S}ol(B_1 \lessdot_\theta B_3)$ alors par hypothèse d'induction, on a $A_1 \rightarrow_\beta^* B_1$ et $A_2 \rightarrow_\beta^* B_2$ et $A_3 \rightarrow_\beta^* B_3$ on prouve pour conclure que

$$\begin{array}{ll} & (\lambda_\theta A_1.A_2)\, A_3 \\ \rightarrow_\beta^* & (\lambda_\theta B_1.B_2)\, B_3 \\ \mapsto & B_2\sigma_1 \wr \ldots \wr B_2\sigma_n. \end{array}$$

□

La preuve que la réduction parallèle a la propriété du diamant dépend du lemme suivant, qui est en fait une généralisation de l'hypothèse hT.

Lemme 2 (Lemme fondamental)**.** *Pour tout terme C et C', et pour toute substitution σ et σ',tel que $C \twoheadrightarrow C'$ et $\sigma \twoheadrightarrow \sigma'$ on a $C\sigma \twoheadrightarrow C'\sigma'$.*

Démonstration. La preuve est faite par induction.

- Si $C \equiv C'$ alors le résultat est immédiat grâce à $\mathbf{H_2}$ et en utilisant une induction triviale sur C.
- Si $C \equiv C_1C_2$ et $C' \equiv C_1'C_2'$ avec $C_1 \twoheadrightarrow C_1'$ et $C_2 \twoheadrightarrow C_2'$. Par hypothèse d'induction appliquée à C_1 et C_2, on a $C_1\sigma \twoheadrightarrow C_1'\sigma'$ et $C_2\sigma \twoheadrightarrow C_2'\sigma'$ alors on a $(C_1C_2)\sigma \twoheadrightarrow (C_1'C_2')\sigma'$ ce qui veut dire que $C\sigma \twoheadrightarrow C'\sigma'$. Ce qui conclut le cas.
- Si $C \equiv (\lambda_\theta C_1.C_2)$ et $C' \equiv (\lambda_\theta C_1'.C_2')$ avec $C_1 \twoheadrightarrow C_1'$ et $C_2 \twoheadrightarrow C_2'$. Ce cas est identique au cas précédent. En fait par hypothèse d'induction appliquée à C_1 et C_2, on a $C_1\sigma \twoheadrightarrow C_1'\sigma'$ et $C_2\sigma \twoheadrightarrow C_2'\sigma'$ et enfin on obtient $(\lambda_\theta C_1.C_2)\sigma \twoheadrightarrow (\lambda_\theta C_1'.C_2')\sigma'$ ce qui veut dire que $C\sigma \twoheadrightarrow C'\sigma'$. Ce qui conclut le cas.
- Si $C \equiv (\lambda_{\theta'} C_1.C_2)C_3$ et $C' \equiv C_2'\tau_1' \wr \ldots \wr C_2'\tau_n'$ avec $C_1 \twoheadrightarrow C_1'$ et $C_2 \twoheadrightarrow C_2'$ et $C_3 \twoheadrightarrow C_3'$ et $\{\tau_i'\}_{i=1}^n = \mathcal{S}ol(C_1' \lessdot_{\theta'} C_3')$. On veut prouver que $C\sigma \equiv (\lambda_{\theta'} C_1\sigma.C_2\sigma)C_3\sigma \twoheadrightarrow (C_2'\tau_1' \wr \ldots \wr C_2'\tau_n')\sigma' \equiv C'\sigma'$ Par hypothèse d'induction appliquée à C_1, C_2 et C_3 on a

$$C_1\sigma \twoheadrightarrow C_1'\sigma' \quad \text{et} \quad C_2\sigma \twoheadrightarrow C_2'\sigma' \quad \text{et} \quad C_3\sigma \twoheadrightarrow C_3'\sigma'.$$

On vérifie que $\mathcal{D}om(\sigma') \cap \theta' = \emptyset$ même par α-conversion, et on peut appliquer $\mathbf{H_1}$ et ainsi avoir $\mathcal{S}ol(C_1'\sigma' \lessdot_{\theta'} C_3'\sigma') = \{(\sigma' \circ \tau_i')\}_{i=1}^n$. alors, on a

$$(\lambda_{\theta'} C_1\sigma.C_2\sigma)C_3\sigma \twoheadrightarrow (C_2'\sigma')\{(\sigma' \circ \tau_i')_{|\theta'}\}_{i=1}^n$$

Donc, pour conclure la preuve il suffit de montrer que $\{(\sigma' \circ \tau'_i)\}_{i=1}^n \equiv \{(\sigma' \circ \tau'_i)_{|\theta'}\}_{i=1}^n \circ \sigma'$. La preuve est divisée en deux cas.

1. Si x appartient à l'ensemble θ' alors

$$\{(\sigma' \circ \tau'_i)\}_{i=1}^n \quad : \quad x \xrightarrow{(\sigma' \circ \tau'_1)} x(\sigma' \circ \tau'_1) \xrightarrow{(\sigma' \circ \tau'_2)} x\{(\sigma' \circ \tau'_i)\}_{i=1}^2 \xrightarrow{\{(\sigma' \circ \tau'_i)\}_{i=3}^n} x\{(\sigma' \circ \tau'_i)\}_{i=1}^n$$

$$\{(\sigma' \circ \tau'_i)_{|\theta'}\}_{i=1}^n \circ \sigma' \quad : \quad x \xrightarrow[(1)]{\sigma'} x \xrightarrow{\{(\sigma' \circ \tau'_i)_{|\theta'}\}_{i=1}^n} x\{(\sigma' \circ \tau'_i)\}_{i=1}^n$$

2. Si x n'appartient pas à l'ensemble θ' alors

$$\{(\sigma' \circ \tau'_i)\}_{i=1}^n \quad : \quad x \xrightarrow[(3)]{\tau'_1} x \xrightarrow{\sigma'} x\sigma' \xrightarrow{\{(\sigma' \circ \tau'_i)\}_{i=2}^n} x\sigma'$$

$$\{(\sigma' \circ \tau'_i)_{|\theta'}\}_{i=1}^n \circ \sigma' \quad : \quad x \xrightarrow{\sigma'} x\sigma' \xrightarrow[(2)]{\{(\sigma' \circ \tau'_i)_{|\theta'}\}_{i=1}^n} x\sigma'$$

Les étapes marquées par (1) et (2) sont possibles grâce à α-conversion ; on vérifie que $\mathcal{D}om(\sigma') \cap \theta' = \emptyset$ et $\mathcal{R}an(\sigma') \cap \theta' = \emptyset$. L'étape labelée (3) est vraie en appliquant $\mathbf{H_0}$, on a $\mathcal{D}om(\tau'_i) = \theta'$.

□

La preuve que la réduction parallèle a la propriété du diamant dépend également du lemme suivant, qui est une généralisation de l'hypothèse hTh.

Lemme 3 (Lemme fondamental)**.** $\forall C, C' \in \Lambda,\ \forall \sigma, \sigma' : \mathbb{V} \to \Lambda,\ \forall \mathbb{T}$*, tels que* $C \sim_{\mathbb{T}} C'$ *et* $\sigma \sim_{\mathbb{T}} \sigma'$ *on a* $C\sigma \sim_{\mathbb{T}} C'\sigma'$.

Exemple 12 (Stable par substitutions équivalentes)**.** *Soit deux termes* $x+y$ *et* $y+x$ *et deux substitutions* $\sigma = \{\{x \leftarrow a+b, y \leftarrow c\}\}$ *et* $\sigma' = \{\{x \leftarrow b+a, y \leftarrow c\}\}$ *avec* $\sigma \ \sim_C \ \sigma'$ *et* $x+y \ \sim_C \ y+x$*. Alors* $(x+y)\sigma \ \sim_C \ (y+x)\sigma'$ *ce qui égal à* $a+b+c \ \sim_C \ b+a+c$

Démonstration. La preuve est faire par induction.

- Si $C \equiv C'$ alors le résultat est immédiat grâce à une induction triviale sur C.
- Si $C \equiv C_1C_2$ et $C' \equiv C'_1C'_2$ avec $C_1 \sim_{\mathbb{T}} C'_1$ et $C_2 \sim_{\mathbb{T}} C'_2$. Par hypothèse d'induction appliquée à C_1 et C_2, on a $C_1\sigma \sim_{\mathbb{T}} C'_1\sigma'$ et $C_2\sigma \sim_{\mathbb{T}} C'_2\sigma'$ et alors on a $(C_1C_2)\sigma \sim_{\mathbb{T}} (C'_1C'_2)\sigma'$ ce qui veut dire que $C\sigma \sim_{\mathbb{T}} C'\sigma'$. Ce qui conclut le cas.
- Si $C \equiv (\lambda_\theta C_1.C_2)$ et $C' \equiv (\lambda_\theta C'_1.C'_2)$ avec $C_1 \sim_{\mathbb{T}} C'_1$ et $C_2 \sim_{\mathbb{T}} C'_2$. Ce cas est identique au cas précédent. En fait, par hypothèse d'induction appliquée à C_1 et C_2, on a $C_1\sigma \sim_{\mathbb{T}} C'_1\sigma'$ et $C_2\sigma \sim_{\mathbb{T}} C'_2\sigma'$ et alors on obtient $(\lambda_\theta C_1.C_2)\sigma \sim_{\mathbb{T}} (\lambda_\theta C'_1.C'_2)\sigma'$ ce qui veut dire que $C\sigma \sim_{\mathbb{T}} C'\sigma'$. Ce qui conclut le cas.
- Si $C \equiv (\lambda_\theta C_1.C_2)C_3$ et $C' \equiv (\lambda_\theta C'_1.C'_2)C'_3$ avec $C_1 \sim_{\mathbb{T}} C'_1$ et $C_2 \sim_{\mathbb{T}} C'_2$ et $C_3 \sim_{\mathbb{T}} C'_3$. Ce cas est identique au cas précédent. Par hypothèse d'induction appliquée à C_1, C_2 et C_3, on a $C_1\sigma \sim_{\mathbb{T}} C'_1\sigma'$, $C_2\sigma \sim_{\mathbb{T}} C'_2\sigma'$ et $C_3\sigma \sim_{\mathbb{T}} C'_3\sigma'$ et après on obtient $((\lambda_\theta C_1.C_2)C_3)\sigma \sim_{\mathbb{T}} ((\lambda_\theta C'_1.C'_2)C'_3)\sigma'$ ce qui veut dire que $C\sigma \sim_{\mathbb{T}} C'\sigma'$. Ce qui conclut le cas.

□

Lemme 4 (Propriété du diamant pour la relation $\twoheadrightarrow$ modulo $\sim_{\mathbb{T}}$)**.** *La relation* $\twoheadrightarrow$ *satisfait la propriété du diamant qui est :* $\forall A, B, C \in \Lambda$ *si* $A \twoheadrightarrow B$ *et* $A \twoheadrightarrow C$ *alors il existe deux termes* $D' \in \Lambda$ *et* $D'' \in \Lambda$ *tel que* $B \twoheadrightarrow D'$ *et* $C \twoheadrightarrow D''$ *et* $D' \sim_{\mathbb{T}} D''$.

Démonstration. La preuve se fait par induction sur la structure de A.

- Si $A \equiv x$ ou $A \equiv c$ alors le résultat est évident car on a nécessairement $A \equiv B \equiv C$.
- Si $A \equiv (\lambda_\theta A_1.A_2)$ alors :
  on a $B \equiv (\lambda_\theta B_1.B_2)$ avec $A_1 \twoheadrightarrow B_1$ et $A_2 \twoheadrightarrow B_2$
  et $C \equiv (\lambda_\theta C_1.C_2)$ avec $A_1 \twoheadrightarrow C_1$ et $A_2 \twoheadrightarrow C_2$.
  On applique l'hypothèse d'induction à A_1 et à A_2 et on conclut qu'il existe deux termes D'_1 et D'_2 tel que
  on a $B_1 \twoheadrightarrow D'_1$ et $C_1 \twoheadrightarrow D''_1$ avec $D'_1 \sim_{\mathbb{T}} D''_1$
  et $B_2 \twoheadrightarrow D'_2$ et $C_2 \twoheadrightarrow D''_2$ avec $D'_2 \sim_{\mathbb{T}} D''_2$
  On conclut que

$$\begin{array}{rrcl} & B \equiv (\lambda_\theta B_1.B_2) & \twoheadrightarrow & (\lambda_\theta D'_1.D'_2) \equiv D' \\ \text{et} & C \equiv (\lambda_\theta C_1.C_2) & \twoheadrightarrow & (\lambda_\theta D''_1.D''_2) \equiv D'' \text{ avec } D' \sim_{\mathbb{T}} D''. \end{array}$$

- Si $A \equiv A_1 A_2$ alors il y a trois cas qui dépendent de la dernière règle utilisée pour prouver que $A \twoheadrightarrow B$ et $A \twoheadrightarrow C$.

  1. Si $B \equiv B_1 B_2$ et $C \equiv C_1 C_2$ avec $A_1 \twoheadrightarrow B_1$ et $A_2 \twoheadrightarrow B_2$ et $A_1 \twoheadrightarrow C_1$ et $A_2 \twoheadrightarrow C_2$ alors la preuve est similaire à la précédente.
  2. Si $A \equiv (\lambda_\theta A_1.A_2)A_3$ et $B \equiv B_2\sigma_1 \wr \ldots \wr B_2\sigma_n$ et $C \equiv C_2\tau_1 \wr \ldots \wr C_2\tau_n$ avec

$$\left\{\begin{array}{lll} A_1 \twoheadrightarrow B_1 & \text{et} & A_1 \twoheadrightarrow C_1 \\ A_2 \twoheadrightarrow B_2 & \text{et} & A_2 \twoheadrightarrow C_2 \\ A_3 \twoheadrightarrow B_3 & \text{et} & A_3 \twoheadrightarrow C_3 \end{array}\right. \qquad \text{et} \qquad \left\{\begin{array}{l} \mathcal{S}ol(B_1 \lessdot_\theta B_3) = \{\sigma_i\}_{i=1}^n \\ \mathcal{S}ol(C_1 \lessdot_\theta C_3) = \{\tau_j\}_{j=1}^n \end{array}\right.$$

     On applique l'hypothèse d'induction sur A_1, A_2 et A_3 et on obtient

$$\begin{array}{lllll} B_1 \twoheadrightarrow D'_1 & \text{et} & C_1 \twoheadrightarrow D''_1 & \text{avec} & D'_1 \sim_{\mathbb{T}} D''_1 \\ B_2 \twoheadrightarrow D'_2 & \text{et} & C_2 \twoheadrightarrow D''_2 & \text{avec} & D'_2 \sim_{\mathbb{T}} D''_2 \\ B_3 \twoheadrightarrow D'_3 & \text{et} & C_3 \twoheadrightarrow D''_3 & \text{avec} & D'_3 \sim_{\mathbb{T}} D''_3 \end{array}$$

     En appliquant $\mathbf{H_2}$ on obtient

$$\{\sigma'_i\}_{i=1}^n = \mathcal{S}ol(D'_1 \lessdot_\theta D'_3) \quad \text{et} \quad \{\tau'_j\}_{j=1}^n = \mathcal{S}ol(D''_1 \lessdot_\theta D''_3)$$

     En appliquant le lemme 2 on obtient $\forall i\ B_2\sigma_i \twoheadrightarrow D'_2\sigma'_i \equiv D'$ et $\forall j\ C_2\tau_j \twoheadrightarrow D''_2\tau'_j \equiv D''$ avec $D'_2 \sim_{\mathbb{T}} D''_2$.
     En appliquant $\mathbf{H_3}$ on a $D'_1 \sim_{\mathbb{T}} D''_1$ et $D'_3 \sim_{\mathbb{T}} D''_3$, donc $\forall i \sigma'_i \sim_{\mathbb{T}} \tau'_i$ on peut conclure que $D' \sim_{\mathbb{T}} D''$.
  3. Le dernier cas est $A \equiv (\lambda_\theta A_1.A_2)A_3$ et $B \equiv B_2\sigma_1 \wr \ldots \wr B_2\sigma_n$ et $C \equiv (\lambda_\theta C_1.C_2)C_3$ avec

$$\begin{array}{lllll} A_1 \twoheadrightarrow B_1 & \text{et} & A_1 \twoheadrightarrow C_1 \text{ et} & & \\ A_2 \twoheadrightarrow B_2 & \text{et} & A_2 \twoheadrightarrow C_2 \text{ et} & & \\ A_3 \twoheadrightarrow B_3 & \text{et} & A_3 \twoheadrightarrow C_3 \text{ et} & \mathcal{S}ol(B_1 \lessdot_\theta B_3) = \{\sigma_i\}_{i=1}^n & \end{array}$$

     est identique au cas précédent.

□

Lemme 5 (Cohérence de la relation $\twoheadrightarrow$ modulo $\sim_{\mathbb{T}}$)**.** *La relation $\twoheadrightarrow$ est cohérente modulo $\sim_{\mathbb{T}}$ si, pour tout terme A, B, A' tel que $A \twoheadrightarrow B$ et $A \sim_{\mathbb{T}} A'$, il existe B' tel que $A' \twoheadrightarrow B'$ et $B \sim_{\mathbb{T}} B'$.*

Démonstration. La preuve se fait par induction sur la structure de A.

- Si $A \equiv x$ ou $A \equiv c$ alors le résultat est évident car on a nécessairement $A \equiv B \equiv A' \equiv B'$.
- Si $A \equiv (\lambda_\theta A_1.A_2)$ alors
  on a $B \equiv (\lambda_\theta B_1.B_2)$ avec $A_1 \twoheadrightarrow B_1$ et $A_2 \twoheadrightarrow B_2$
  et $A' \equiv (\lambda_\theta A'_1.A'_2)$ avec $A_1 \sim_{\mathbb{T}} A'_1$ et $A_2 \sim_{\mathbb{T}} A'_2$.
  En appliquant l'hypothèse d'induction sur A, A', B on conclut qu'il existe un terme B' tel que
  on a $A'_1 \twoheadrightarrow B'_1$ et $A'_2 \twoheadrightarrow B'_2$ avec $B_1 \sim_{\mathbb{T}} B'_1$ et $B_2 \sim_{\mathbb{T}} B'_2$
  On conclut que
  $$\begin{array}{rrl} & A' \equiv (\lambda_\theta A'_1.A'_2) & \twoheadrightarrow \quad (\lambda_\theta B'_1.B'_2) \equiv B' \\ \text{et} & B \equiv (\lambda_\theta B_1.B_2) & \sim_{\mathbb{T}} \quad (\lambda_\theta B'_1.B'_2) \equiv B' \end{array}$$
- Si $A \equiv A_1 A_2$ alors il existe trois cas différents.
  1. Si $B \equiv B_1 B_2$ et $A' \equiv A'_1 A'_2$ avec $A_1 \twoheadrightarrow B_1$ et $A_2 \twoheadrightarrow B_2$ et $A_1 \sim_{\mathbb{T}} A'_1$ et $A_2 \sim_{\mathbb{T}} A'_2$ alors la preuve est identique au cas précédent.
  2. Si $A \equiv (\lambda_\theta A_1.A_2)A_3$ et $B \equiv A_2\sigma_1 \wr \ldots \wr A_2\sigma_n$ et $A' \equiv (\lambda_\theta A'_1.A'_2)A'_3$ avec
  $$\begin{cases} A_1 \twoheadrightarrow B_1 & \text{et} \quad A_1 \sim_{\mathbb{T}} A'_1 \\ A_2 \twoheadrightarrow B_2 & \text{et} \quad A_2 \sim_{\mathbb{T}} A'_2 \\ A_3 \twoheadrightarrow B_3 & \text{et} \quad A_3 \sim_{\mathbb{T}} A'_3 \end{cases} \qquad \text{et} \qquad \begin{cases} \mathcal{S}ol(A_1 \prec_\theta A_3) = \{\sigma_i\}_{i=1}^n \\ \mathcal{S}ol(A'_1 \prec_\theta A'_3) = \{\sigma'_j\}_{j=1}^n \end{cases}$$
  En appliquant l'hypothèse d'induction sur A, A', B on conclut qu'il existe un terme $B' \equiv A'_2\sigma'_1 \wr \ldots \wr A'_2\sigma'_n$ tel que $A' \twoheadrightarrow B'$ EN appliquant $\mathbf{H_3}$ on obtient $\forall i \sigma_i \sim_{\mathbb{T}} \sigma'_i$ et on a $A_2 \sim_{\mathbb{T}} A'_2$. En appliquant le Lemme 3 on obtient $\forall i\ A_2\sigma_i \sim_{\mathbb{T}} A'_2\sigma'_i$.On conclut que $B \sim_{\mathbb{T}} B'$.
  3. Le dernier cas est $A \equiv (\lambda_\theta A_1.A_2)A_3$ et $B \equiv (\lambda_\theta B_1.B_2)B_3$ et $A' \equiv (\lambda_\theta A'_1.A'_2)A'_3$avec
  $$\begin{array}{lll} A_1 \twoheadrightarrow B_1 & \text{et} & A_1 \sim_{\mathbb{T}} A'_1 \text{ et} \\ A_2 \twoheadrightarrow B_2 & \text{et} & A_2 \sim_{\mathbb{T}} A'_2 \text{ et} \\ A_3 \twoheadrightarrow B_3 & \text{et} & A_3 \sim_{\mathbb{T}} A'_3 \end{array}$$
  est identique au cas précédent.

□

Théorème 1 (Confluence)**.** *Le λ-calcul dynamique à motifs avec $\mathcal{S}ol$ satisfaisant* $\mathbf{H_0}$, $\mathbf{H_1}$, $\mathbf{H_2}$ *et* $\mathbf{H_3}$ *est confluent modulo $\sim_{\mathbb{T}}$.*

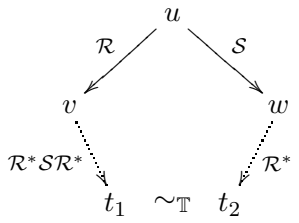
Démonstration. Le lemme 1 nous assure que la clôture réflexive et transitive des relations $\mapsto_{\mathcal{B}}$ et $\twoheadrightarrow$ sont les mêmes. Le Lemme 4 nous assure que, la relation $\twoheadrightarrow$ a la propriété du diamant modulo $\sim_{\mathbb{T}}$ et le lemme 5 on a la cohérence de la relation $\twoheadrightarrow$ modulo $\sim_{\mathbb{T}}$. On conclut que la relation $\mapsto_{\mathcal{B}}$ est confluente. □

1.4 Yokouchi-Hikita modulo

Le lemme de Yokouchi-Hikita modulo a été introduit par Germain Faure [Fau07]. La version présentée ci-dessous est une version modifiée du lemme car dans cette version la relation $\mathcal{S}$ a la propriété du diamant modulo une équivalence alors que dans la version originale la relation $\mathcal{S}$ a simplement la propriété du diamant.

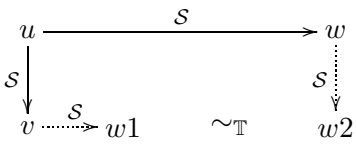
Lemme 6 (Yokouchi-Hikita modulo)**.** *On définit $\mathcal{R}$ et $\mathcal{S}$ qui sont deux relations appartenant au même ensemble $\mathcal{T}$ (qui représente un ensemble de règle de réduction) et $\sim_{\mathbb{T}}$ une relation d'équivalence telles que :*

- *$\mathcal{R}$ est fortement normalisable, c'est-à-dire, si toutes les réductions à partir d'un λ-terme sont finies.*
- *$\mathcal{R}$ est confluent modulo $\sim_{\mathbb{T}}$, c'est-à-dire, pour tout u, v, w dans $\mathcal{T}$ tel que $u\ \mathcal{R}^*\ v$ et $u\ \mathcal{R}^*\ w$ il existe t_1, t_2 dans $\mathcal{T}$ tel que $v\ \mathcal{R}^*\ t_1$, $w\ \mathcal{R}^*\ t_2$ et $t_1 \sim_{\mathbb{T}} t_2$*
- *$\mathcal{S}$ a la propriété du diamant modulo $\sim_{\mathbb{T}}$, c'est-à-dire, pour tout u, v, w dans $\mathcal{T}$ tel que $u\mathcal{S}v$ et $u\mathcal{S}w$ il existe un élément t dans $\mathcal{T}$ tel que $v\mathcal{S}t1$ et $w\mathcal{S}t2$ et $t1 \sim_{\mathbb{T}} t2$*
- *$\mathcal{R}$ et $\mathcal{S}$ sont cohérents modulo $\sim_{\mathbb{T}}$, c'est-à-dire, pour tout u, v, w tel que $u \sim_{\mathbb{T}} v$, $u\mathcal{R}w$ (resp. $u\mathcal{S}w$) il existe un t tel que $v\mathcal{R}t$ (resp. $v\mathcal{S}t$) et $w \sim_{\mathbb{T}} t$.*
- *le diagramme Yokouchi-Hikita modulo suivant est vérifié :*

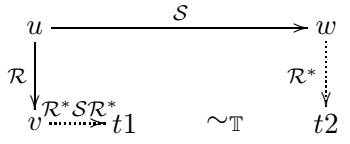


Alors la relation $\mathcal{R}^*\mathcal{S}\mathcal{R}^*$ est confluente modulo $\sim_{\mathbb{T}}$.

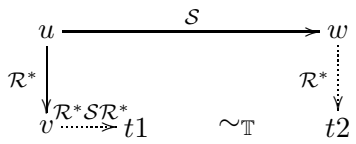
Démonstration. Soient $\mathcal{R}$ et $\mathcal{S}$ deux relations définies dans le même ensemble X, $\mathcal{R}$ et qui est confluent et fortement normalisable, et $\mathcal{S}$ qui est fortement confluent modulo $\sim_{\mathbb{T}}$, c'est-à-dire le diagramme suivant peut être utilisé, pour tout u, v, w dans X.



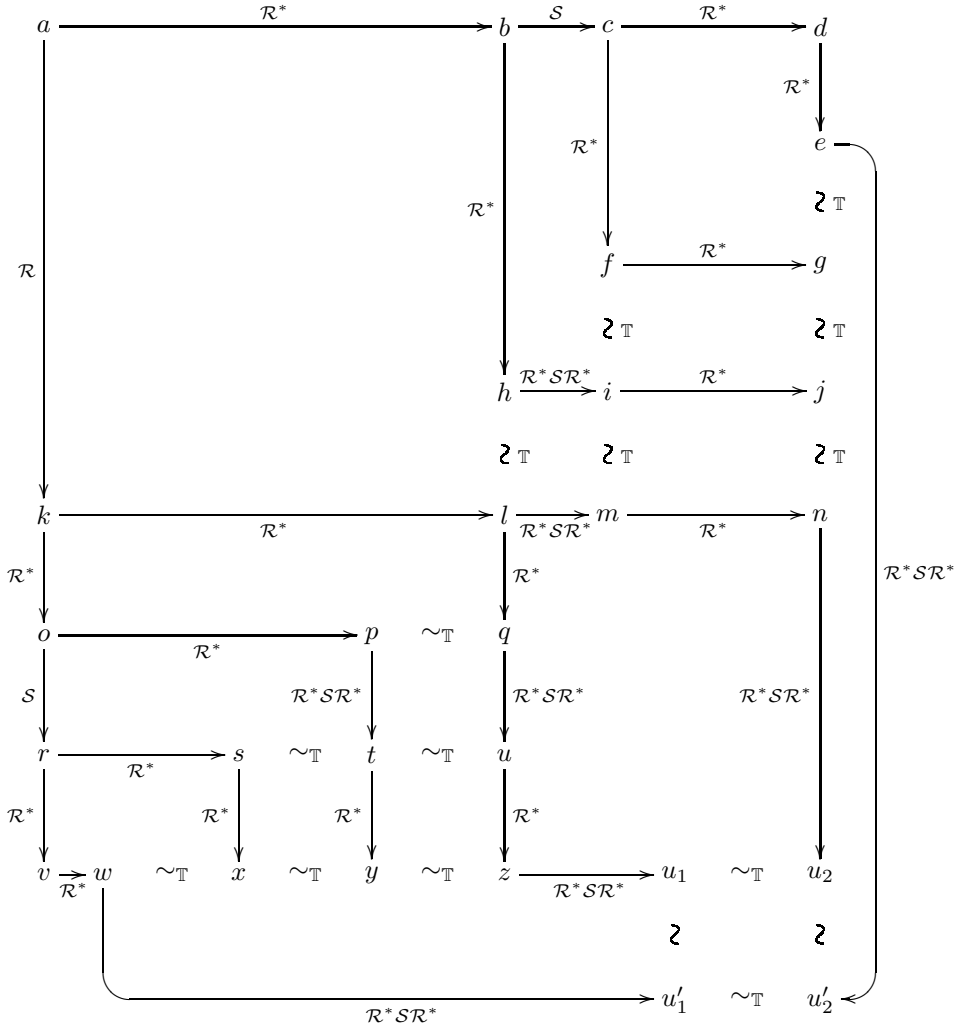
De plus, on suppose que le diagramme suivant est correct :



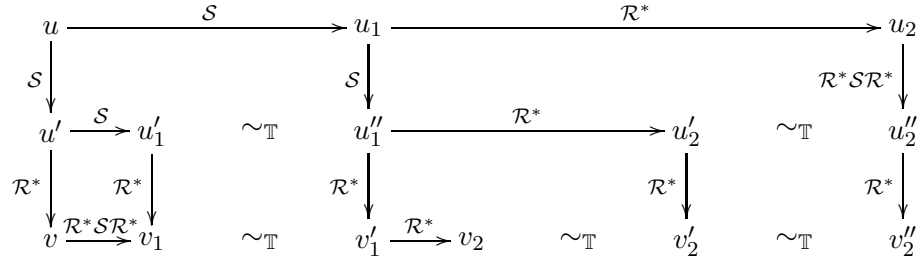
Par induction sur le nombre de rédution du λ-terme u par la relation R.



On conclut, par distinction sur le nombre de réduction par la relation R sur le λ-terme u, et on distingue respectivement le cas où ce nombre est différent de zéro et le cas où ce nombre est égal à zero.



et



□

Lemme 7. *La relation* $\twoheadrightarrow_\xi \;\twoheadrightarrow\!\!\!\!\!\!\!+\!\!\!+\; \twoheadrightarrow_\xi$ *est confluente modulo* $\sim_{\mathbb{T}}$.

Démonstration. Si toutes les conditions du lemme [?] sont respectées alors on prouve que cette relation est confluente modulo $\sim_{\mathbb{T}}$.

- $\mathcal{S}ol$ satisfait $\mathbf{H_0}, \mathbf{H_1}, \mathbf{H_2}, \mathbf{H_3}$
- la relation $\rightarrow_\xi$ est fortement normalisable (c'est-à-dire si toutes les réductions à partir d'un λ-terme sont finies) par le Lemme 9.
- la relation $\rightarrow_\xi$ est confluente modulo $\sim_{\mathbb{T}}$ par le lemme 12.
- la relation $\twoheadrightarrow\!\!\!\!\!\!\!+\!\!\!+$ a la propriété du diamant modulo $\sim_{\mathbb{T}}$ (lemme 4).
- les relations $\twoheadrightarrow\!\!\!\!\!\!\!+\!\!\!+$ et $\rightarrow_\xi$ sont cohérentes modulo $\sim_{\mathbb{T}}$. La cohérence de la relation $\twoheadrightarrow\!\!\!\!\!\!\!+\!\!\!+$ modulo $\sim_{\mathbb{T}}$ par le lemme 3 et la cohérence de $\rightarrow_\xi$ est obtenue par le lemme 10.
- Le diagramme de Yokouchi-Hikita modulo est vérifié par le lemme 6.

□

Lemme 8. *le* λ*-calcul dynamique à motifs est confluent modulo* $\sim_{\mathbb{T}}$

Démonstration. On applique le lemme de Yokouchi-Hikita's en prenant la relation $\mathcal{R}$ la relation induite par l'ensemble des règles de réduction ξ et la relation $\mathcal{S}$ qui est la relation $\twoheadrightarrow\!\!\!\!\!\!\!+\!\!\!+$. La relation $\twoheadrightarrow\!\!\!\!\!\!\!+\!\!\!+$ a la propriété du diamant grâce au lemme 4. Pour conclure la preuve, il est suffisant de remarquer que la clôture transitive et réflexive de la relation

$$\rightarrow_{\beta\cup\xi} \qquad \text{et} \qquad \twoheadrightarrow_\xi \;\twoheadrightarrow\!\!\!\!\!\!\!+\!\!\!+\; \twoheadrightarrow_\xi$$

sont les mêmes (conséquence du lemme 1). □

En fait ce théorème nous dit que tout calcul à motifs défini comme une instance du λ-calcul dynamique à motifs avec une fonction $\mathcal{S}ol$ particulière qui satisfait les conditions $\mathbf{H_0}$, $\mathbf{H_1}$, $\mathbf{H_2}$ et $\mathbf{H_3}$ est confluent.

1.5 Confluence seulement avec des motifs linéaires

On a vu précédement que les conditions imposées sur l'algorithme de filtrage sont fortes. Néanmoins, ces conditions sont imposées par beaucoup de calculs à motif. On décrit plus bas le contre-exemple le plus connu pour la confluence.

Par exemple, si le filtrage peut être fait sur des variables actives alors une réduction non confluente peut être obtenue dans les deux calculs suivants : le λ-calcul à motif [vO90] et dans le calcul de réécriture [CLW03]. De la même manière, des motifs non linéaires peuvent mener à une réduction non confluente comme le montre l'exemple de Klop [BK86] pour les systèmes d'ordre supérieur qui autorisent un filtrage non linéaire. C'est pour cela que dans le λ-calcul à motif ou dans le calcul de réécriture on considère seulement les motifs linéaires et dans le calcul à motifs purs le filtrage d'un terme non linéaire échoue.

Proposition 1 (Non confluence)**.** *Le* λ*-calcul dynamique à motifs avec* $\mathcal{S}ol$ *défini de telle manière que pour tous termes* A *et* B *et pour une constante* d.

$\mathcal{S}ol(A \ll_\emptyset A) = \mathtt{id}$
$\mathcal{S}ol(x \ll_x A) = \{x \leftarrow A\}$
$\mathcal{S}ol(d(x,y) \ll_{x,y} d(A,B)) = \{x \leftarrow A, y \leftarrow B\}$
n'est pas confluent.

Démonstration. L'idée principale de la preuve est de remarquer qu'on encode des motifs non linéaire en utilisant un motif linéaire dynamique (c'est-à-dire motif qui contient des variables qui ne sont pas liées par l'abstraction) comme indiqué dans la ci-dessous :

$$\lambda_x(d\,x\,x).\clubsuit \qquad \triangleq \qquad \lambda_{x,y}(d\,x\,y).(\lambda_\emptyset x.\clubsuit)y$$

où $\clubsuit$ représente un terme arbitraire.

On encode le contre exemple de Klop dans le calcul de réécriture. Soient d et e deux constantes. On oublie l'ensemble θ des variables liées par l'abstraction quand il est égal à l'ensemble des variables libres dans le motif. On note $\twoheadrightarrow_h$ la réduction de tête (c'est-à-dire la réduction la plus à gauche).

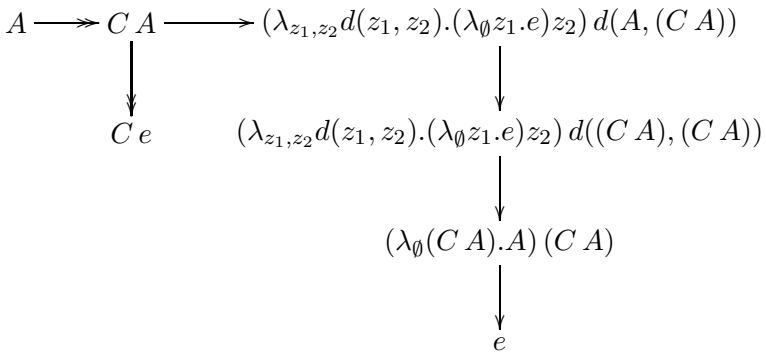
On note le combinateur de point fixe Y par

$$\mathsf{Y} \triangleq \Big(\lambda_y y.\lambda_x x.(x\,(y\,y\,x))\Big)\Big(\lambda_y y.\lambda_x x.(x\,(y\,y\,x))\Big)$$

Habituellement, pour tout terme A on a $\mathsf{Y}\,A \twoheadrightarrow_h A\,(\mathsf{Y}\,A)$. On définit les termes suivants :

$$\begin{array}{lcl} C' & \equiv & \big(\lambda_y y.(\lambda_x x.(\lambda_{z_1,z_2} d(z_1,z_2).(\lambda_\emptyset z_1.e)z_2)d(x,(y\,x)))\big) \\ C & \equiv & \mathsf{Y}C' \\ C & \twoheadrightarrow_h & \big(\lambda_y y.\lambda_x x.(\lambda_{z_1,z_2} d(z_1,z_2).(\lambda_\emptyset z_1.e)z_2)d(x,(y\,x))\big)\ C \\ & \twoheadrightarrow_h & \lambda_x x.(\lambda_{x,z} d(x,z).(\lambda_\emptyset z_1.e)z_2)d(x,(C\,x)) \\ A & \equiv & \mathsf{Y}\,C \end{array}$$

On obtient les réductions suivantes :



La constante e est en forme normale (c'est-à-dire ne peut plus être β-réduit). Si on considère la plus petite réduction partant de Ce alors dans cette réduction, l'abstraction de tête peut être réduit en :

$$\begin{array}{lcll} C\,e & \equiv & & (\mathsf{Y}C')\,e \\ & \twoheadrightarrow_h & h & C'\,(\mathsf{Y}\,C')\,e \\ & \twoheadrightarrow_h & h & (\lambda(d(z_1,z_2).(\lambda_\emptyset z_1.e)z_2)(d(e,(C\,e)) \\ & \mapsto_h & & (\lambda_\emptyset e.e)(C\,e) \end{array}$$

Mais le terme $(\lambda_\emptyset e.e)(C\,e)$ ne peut être réduit en tête seulement après avoir réduit $C\,e$ en e. On conclut que $C\,e$ ne peut être réduit en e. □

Comme conséquence, on obtient un calcul à motif avec la fonction $\mathcal{S}ol$ qui satisfait les conditions de la proposition 1 qui n'est pas confluent. Ce qui est surprenant car les deux dernières règles sont satisfaites par n'importe quel algorithme de filtrage syntaxique et la première peut être vue comme un choix raisonnable. D'un autre coté, ignorer les informations données par l'ensemble θ des variables liées au moment du filtrage peut engendrer d'étranges comportements. Il existe quelques tentatives de développement de calculs qui combinent ces trois conditions de la proposition 1 dans [BCKL03], mais elles n'ont pas été concluantes. En fait, la solution proposée dans [BCKL03] valide la première condition mais ne peut jamais autoriser la seconde condition ni la troisième.

Chapitre 2

Instances du λ-calcul dynamique à motifs

Comme nous l'avons vu dans le chapitre 1, beaucoup de calculs à motif étendent la règle β (ou un équivalent) par un ensemble de règles. Nous voullons montrer que les conditions imposées dans le but de prouver la confluence du λ-calcul dynamique à motifs peuvent être utilisées pour d'autre calculs qui peuvent être exprimés comme une instance du λ-calcul dynamique à motifs. Dans ce chapitre, nous donnons les instances du λ-calcul dynamique à motifs pour le λ-calcul à motifs et le calcul à motifs purs avec un filtrage unitaire. Nous donnons également l'instance du calcul de réécriture avec un filtrage modulo la commutativité. Tous ces calculs ont été prouvés confluents sous certaines conditions ; nous donnons la preuve de confluence génerale pour le λ-calcul dynamique à motifs. Notre approche permet d'avoir une méthode de preuve uniforme pour tous les calculs à motifs. Intuitivement, l'hypothèse induite dans la section 1.2.1 sous laquelle nous avons prouvé la confluence d'un calcul qui garantit une cohérence entre $\mathcal{S}ol$ et les autres règles du calcul (la cohérence est définie dans la définition 11). Le résultat obtenu peut être généralisé pour certains calculs qui utilisent la β-réduction avec une extension de l'ensemble des règles ξ qui satisfont des conditions de cohérence.

2.1 Le λ-calcul à motifs avec filtrage unitaire

Le λ-calcul à motifs est aussi appelé $\lambda\phi$-calcul a été introduit par Vincent van Oostrom [vO90]. Le λ-calcul à motifs est une extension du λ-calcul avec la possibilité de faire du filtrage sur des motifs et pas seulement sur des variables. L'ensemble Φ représente l'ensemble des motifs appartenant à l'ensemble des λ-termes noté Λ. Le λ-calcul à motifs peut être vu comme une instance du λ-calcul dynamique à motifs.
On définit la syntaxe et la sémantique du λ-calcul à motifs de la fonçon suivante :

Syntaxe du λ-calcul à motifs

$$P, T \quad ::= \quad x \mid c \mid \lambda P.T \mid T\,T \quad \text{où } x \in \mathbb{V}, c \in \mathbb{C}, T \in \mathbb{T} \text{ et } P \in \Phi$$

Sémantique du λ-calcul à motifs

$$(\beta_{\lambda_P}) \qquad (\lambda A.B)(A\sigma) \qquad \rightarrow_{\beta_{\lambda\Phi}} \qquad B\sigma$$

Le calcul n'est pas confluent (voir [vO90] Ex. 4.18) dans le cas général mais certaines restrictions imposées sur l'ensemble des motifs Φ permettent d'obtenir la confluence de la relation β_{λ_P}. Cette restriction est la condition RPC définie ci-dessous.

Définition 12 (RPC)**.** *L'ensemble des termes qui satisfont RPC est l'ensemble des termes du calcul qui :*

- *sont linéaires (chaque variable libre n'apparaît qu'une seule fois)*
- *sont en forme normale (c'est-à-dire ne peut plus être β-réduit)*
- *n'ont pas de variable active (c'est-à-dire pas de sous-terme de la forme xA où x est libre)*

On remarque que RPC est une reformulation de la condition $\mathbf{H_2}$ dans le cas particulier où la fonction $\mathcal{S}ol$ exécute le filtrage sur des motifs clos. Néanmoins, l'hypothèse $\mathbf{H_2}$ autorise le filtrage sur des motifs qui ne sont pas en forme normale (c'est-à-dire ne peut plus être β-réduit), ce qui n'est pas le cas de RPC.

Quelques exemples de réduction du λ-calcul à motifs.

Exemple 13.

$$((\lambda x.x)y) \equiv ((\lambda x.x)x\{x \leftarrow y\}) \rightarrow_{\beta_{\Lambda\Phi}} x\{x \leftarrow y\} \equiv y$$

Exemple 14. *Les paires et les projections peuvent être codées en λ-calcul à motifs par un filtrage direct sur l'encodage de la paire.*

$$\begin{array}{lll} \big(\lambda(\lambda z.(z\,x)\,y).x\big)\big(\lambda z.(z\,A)\,B\big) & \rightarrow_{\beta_{\Lambda\Phi}} & x\{x \leftarrow A,\ y \leftarrow B\} \\ & \equiv & A \end{array}$$

Proposition 2. *Le λ-calcul à motifs est confluent si l'ensemble des motifs respecte la condition RPC.*

Démonstration. Les hypothèses $\mathbf{H_0}$ et $\mathbf{H_1}$ sont immédiates. Pour prouver $\mathbf{H_2}$, on peut remarquer que si $P\sigma \twoheadrightarrow B$ avec $P \in$ RPC alors $\exists B', \sigma'$ tels que $B' \equiv P\sigma'$ avec $\sigma \twoheadrightarrow \sigma'$. Ce qui prouve qu'aucune abstraction ne peut interférer avec P dans $P\sigma$ si $P \in$ RPC, ce qui prouve que la condition $\mathbf{H_2}$ est satisfaite. On conclut la preuve en appliquant le théorème 1. □

2.2 Le calcul à motifs purs avec filtrage unitaire

Le calcul à motifs purs a été introduit par Barry Jay [JK06], ce calcul est une généralisation du λ-calcul car il utilise le filtrage sur des motifs et la β-réduction. Tous les termes du calcul à motifs purs sont des motifs. Par conséquent, on peut écrire des fonctions génériques qui peuvent être appliquées à n'importe quelle structure de données, ainsi la notion de polymorphisme peut être modélisée (c'est-à-dire une fonction peut être utilisée avec différents types, ce qui permet des implémentations plus abstraites et générales). La règle $(\beta_{\mathsf{pc}_{\mathsf{stk}}})$ peut être vue comme une instance de la règle (β) du λ-calcul dynamique à motifs.

Le filtrage dans le calcul à motifs purs est basé sur une notion de ϕ-structures et ϕ-filtrable définie de la façon suivante :

ϕ-structures (noté D) et ϕ-filtrable (noté E)

$$\begin{array}{lll} D & ::= & x\,(x \in \phi) \mid c \mid D\,A \\ E & ::= & D \mid \lambda_\theta A.B \end{array}$$

où A et B sont des termes arbitraires

Sémantique du calcul à motifs purs

$$
\begin{array}{llll}
(\beta_{\mathsf{pc}}) & (\lambda_\theta A.B)C & \to_{\beta_{ppc}} & B\sigma \\
 & & & \text{si } \sigma = \mathcal{S}ol(A \prec\!\!\!\prec_\theta C) \\
(\beta^{\mathsf{stk}}_{\mathsf{pc}}) & (\lambda_\theta A.B)C & \to_{\beta_{ppc}} & \lambda x.x \\
 & & & \text{si none} = \mathcal{S}ol(A \prec\!\!\!\prec_\theta C)
\end{array}
$$

Dans le calcul à motifs purs la fonction $\mathcal{S}ol$ est définie par les équations ci-dessous. Ces équations doivent être laissées dans l'ordre de définition :

$$
\begin{array}{lcl}
\mathcal{S}ol(x \prec\!\!\!\prec_\theta A) & = & \{x \leftarrow A\} \quad \text{si } x \in \theta \\
\mathcal{S}ol(c \prec\!\!\!\prec_\theta c) & = & \mathtt{id} \\
\mathcal{S}ol(A_1A_2 \prec\!\!\!\prec_\theta B_1B_2) & = & \mathcal{S}ol(A_1 \prec\!\!\!\prec_\theta B_1) \uplus \mathcal{S}ol(A_2 \prec\!\!\!\prec_\theta B_2) \\
 & & \text{si } A_1A_2 \text{ est une } \theta\text{-structure de données} \\
 & & \text{si } B_1B_2 \text{ est une structure de données} \\
\mathcal{S}ol(A_1 \prec\!\!\!\prec_\theta B_1) & = & \text{none} \\
 & & \text{si } A_1 \text{ est en forme de } \theta\text{-filtrage} \\
 & & \text{si } B_1 \text{ est en forme de filtrage}
\end{array}
$$

Remarque : l'union disjoint noté $\uplus$ est définit pour unifier des substitutions qui ont des domaines disjoints. Les domaines disjoints permettent de rendre déterministe l'opération de filtrage.

Exemple 15 (Elim)**.** *La fonction éliminatrice permet de retourner le première élément de n'importe quelle structure. Elle est définit de la façon suivante :*

$$\boldsymbol{elim} := \lambda_x x.\lambda_y(xy).y$$

Celle-ci peut s'appliquer à toutes les structures de données. Par exemple, avec la structure de donnée singleton qui est définie de la façon suivante : $\lambda_x x.cons\ x\ nil$.

$$(\lambda_x x.\lambda_y(xy).y)(\lambda_x x.(cons\ x\ nil)) \to_{\beta_{ppc}} \lambda_y((\lambda_x x.cons\ x\ nil)y).y \to_{\beta_{ppc}} \lambda_y(cons\ y\ nil).y$$

La structure de donnée successeur qui est définie de la façon suivante : $\lambda_x x.(succ\ x)$

$$(\lambda_x x.\lambda_y(xy).y)(\lambda_x x.(succ\ x)) \to_{\beta_{ppc}} \lambda_y((\lambda_x x.succ\ x)y).y \to_{\beta_{ppc}} \lambda_y(succ\ y).y$$

Ou encore la structure de données paire qui est définie de la façon suivante :
$pair = \lambda_x x.\lambda_y y.\lambda_f f.(f\ x\ y)$ *ou encore* $pair\ x\ y = \lambda_f f.(f\ x\ y)$*. On introduit une nouvelle constante "?", elle n'a pas de variable libre, elle n'est pas affectée par une substitution. La fonction pour récupérer le premier élément d'une paire s'écrit :* $elim(\lambda_x x.(pair\ x\ ?))$

$$(\lambda_x x.\lambda_y(xy).y)(\lambda_x x.\lambda_f f.(f\ x\ ?)) \to_{\beta_{ppc}} \lambda_y(\lambda_x x.((\lambda_f f.f\ x\ ?)y).y \to_{\beta_{ppc}} \lambda_y\lambda_f f.(f\ y\ ?).y$$

Proposition 3. *Le calcul à motifs purs est confluent.*

Démonstration. L'hypothèse $\mathbf{H_0}$ est vrai. L'hypothèse $\mathbf{H_1}$ (qui indique que la fonction $\mathcal{S}ol$ est stable par substitution) et l'hypothèse $\mathbf{H_2}$ (qui indique que la fonction $\mathcal{S}ol$ est stable par réduction) sont données dans [JK06]. On prouve que la relation $(\beta^{\mathsf{stk}}_{\mathsf{pc}})$ est localement confluente (simple induction). Il est facile de montrer la terminaison, on peut conclure que cette relation est confluente. Par induction on obtient que les relations $\twoheadrightarrow_{\beta_{\mathsf{pc}}}$ et $(\beta^{\mathsf{stk}}_{\mathsf{pc}})$ vérifient le diagramme de Yokouchi-Hikita, ce qui conclut la preuve. □

2.3 Le calcul de réécriture modulo la commutativité

Le calcul de réécriture appelé aussi ρ-calcul, a été introduit pour rendre explicite tous les éléments de la réécriture permettant ainsi d'ajouter des stratégies de réécriture [KK99]. Pour avoir le contrôle des règles de réécriture le ρ-calcul donne explicitement tous les éléments nécessaires, en particulier la notion de règle de réécriture, d'application et de résultat. La première version du ρ-calcul apparaît dans [CK01, Cir00, CK98] et se focalise sur l'encodage des règles de réécriture et sur la stratégie. Le ρ-calcul est une extension du λ-calcul avec une notion de filtrage de motif et de collection de termes. L'opération fondamentale du calcul de réécriture est l'opération de filtrage qui autorise a remplacer des variables liées par une autre valeurs. On autorise que le filtrage soit fait modulo une congruence entre les termes. Cette congruence utilisée lors du filtrage et un paramètre important du calcul et différente instances sont obtenues au moment de la définition de la relation de congruence. Il existe plusieurs versions du ρ-calcul, la version définit ici est le $\rho_{\rightarrow}$-calcul. La syntaxe et la sémantique opérationnelle du $\rho_{\rightarrow}$-calcul sont définies ci-dessous, $\mathbb{T}$ représentant la théorie que respecte le filtrage. Les motifs sont des motifs algébriques linéaires, c'est-à-dire une même variable n'apparaît qu'une seule fois dans un motif.

Syntaxe du $\rho_{\rightarrow}$-calcul

Motifs algébriques	P	$::=$	$x \mid c(P,\ldots,P) \qquad x \in \mathbb{V}, c \in \mathbb{C}.$
Termes	A	$::=$	$c \mid x \mid \lambda P.A \mid A\ A \mid A \wr A$

Sémantique du ρ-calcul

$$(\beta_\rho) \qquad (\lambda P.A)\ B \quad \rightarrow_{\beta_\rho} \quad A\sigma_1 \wr \ldots \wr A\sigma_n \qquad \text{avec } \forall i P\sigma_i \sim_{\mathbb{T}} B$$

$$(\delta) \qquad (A \wr B)\ C \quad \rightarrow_{\delta} \quad A\ C \wr B\ C$$

La règle (β_ρ) peut être considérée comme une instance de la règle (β) du λ-calcul dynamique à motifs.

$$(\beta_\rho) \qquad (\lambda P.A)B \quad \rightarrow_{\beta_\rho} \quad A\sigma_1 \wr \ldots \wr A\sigma_n \qquad \text{si } \{\sigma_i\}_{i=1}^{n} = \mathcal{S}ol(P \prec\!\!\prec B)$$

La règle (β_ρ) porte sur l'application d'une abstraction : quand on applique la règle $\lambda P.A$ sur un terme B, On cherche les solutions potentielles du filtrage entre P et B modulo la congruence $\mathbb{T}$. Quand des solutions existent, les substitutions correspondantes sont appliquées à A et le résultats sont séparés par l'agrégation. Une agrégation est ajoutée au calcul pour chaque sous-ensemble de substitutions. Si une règle n'a pas de solution de filtrage, le terme ne peut être réduit et représente la forme normale du terme. La règle (δ) porte sur l'application de la structure. Quand on applique une structure de la forme $A_1 \wr A_2$ à un terme B, On distribue simplement l'application de l'argument aux éléments de la structure.

2.3.1 Confluence de la sous relation

Lemme 9 (Forte normalisation de la relation $\rightarrow_\delta$)**.** *La relation $\rightarrow_\delta$ est fortement normalisable (c'est-à-dire si toutes les réductions à partir de d'un λ-terme sont finies).*

Démonstration. Chaque paire de λ-termes dans la relation $\rightarrow_\delta$ décroît (strictement). □

Lemme 10 (Cohérence de la relation $\rightarrow_\delta$ modulo $\sim_C$)**.** *La relation $\rightarrow_\delta$ est cohérente modulo une équivalence $\sim_C$, c'est-à-dire modulo la théorie de la commutativité.*

Démonstration. La preuve est faite par l'analyse des règles utilisées pour réduire le λ-terme A en B. Le diagramme des dérivations est facilement clôt pour la règle δ, toutes les exécutions partant du λ-terme A jusqu'à A' peuvent être produite quand on exécute la règle de réduction $\rightarrow_\delta$ de A' jusqu'à B'. □

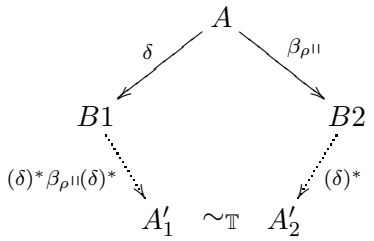
Lemme 11 (La relation $\rightarrow_\delta$ est localement confluente)**.** $\forall A, B_1, B_2 \in \Lambda$ *tels que* $A \rightarrow_\delta B_1$ *et* $A \rightarrow_\delta B_2$, $\exists B'_1, B'_2 \in \Lambda$ *tels que* $B_1 \twoheadrightarrow_\delta C$ *et* $B_2 \twoheadrightarrow_\delta C$ *:*

Démonstration. Il y a pas de chevauchement, c'est-à-dire on ne peut avoir deux réductions différentes en appliquant deux règles différentes sur un même λ-terme. □

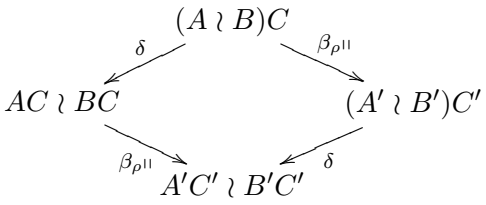
Lemme 12. *La relation* $\rightarrow_\delta$ *est confluente modulo* $\sim_{\mathbb{T}}$*.*

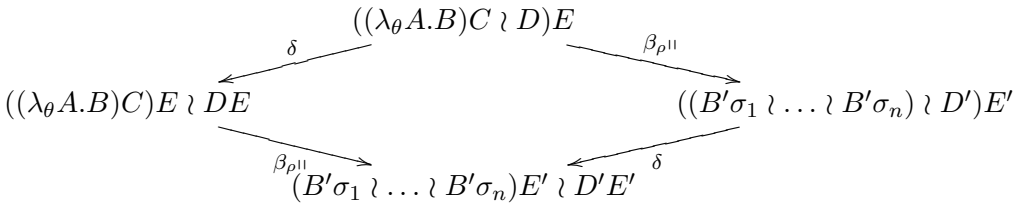
Démonstration. On prouve que cette relation est confluente pour toute théorie $\mathbb{T}$ en utilisant [Ohl98], la forte normalisation (lemme 9), la cohérence (lemme 10) et la confluence locale (lemme 11) de la relation $\rightarrow_\delta$. □

Lemme 13 (Le diagramme de Yokouchi-Hikita modulo $\sim_{\mathbb{T}}$)**.** $\forall A, B_1, B_2 \in \Lambda$*, il existe deux termes* A'_1, A'_2 *tels que les diagrammes suivants soient respectés :*



Démonstration. Quand les deux étapes de A vers B_1 et de A vers B_2 ne se chevauchent pas, la preuve est facile. On veut vérifier toutes les paires critiques. Une sous-expression d'un λ-terme réductible par la relation β_ρ ||, ne peut jamais se chevaucher avec un λ-terme réductible par la relation δ.





□

Définition 13 (Opération basique sur des ensembles)**.** *On doit définir quelques opérations sur les ensembles de substitution renvoyées par la fonction Sol. La priorité de l'opérateur* $\oplus$ *est plus grande que celle de l'opérateur* $\otimes$*.*

$$\{\{x_i \leftarrow a_i\}\} \oplus \{\{y_j \leftarrow b_j\}\} = \begin{cases} \{\{x_i \leftarrow a_i, y_j \leftarrow b_j\}\} & si\ \forall i,j\ x_i = y_i \Rightarrow a_i = b_j \\ \bot & sinon \end{cases}$$

$$\{\sigma_1 \dots \sigma_n\} \otimes \{\tau_1 \dots \tau_m\} = \{\sum_{i=1}^{n} \sum_{j=1}^{m} \{\sigma_i\} \oplus \{\tau_j\}\} tel\ que\ \sigma_i \oplus \tau_j \neq \bot$$

Si $n = 0$ *ou* $m = 0$ *alors* $(\ \_\ \otimes\ \_\ ) = \emptyset$.
$(\ \{\sigma\} \otimes\ \bot\ ) = \{\sigma\}$
$(\ \bot\ \oplus\ \bot\ ) = \bot$

Proposition 4 (Opération de filtrage commutatif)**.** *Quand on a un filtrage de la forme* $\mathcal{S}ol(f(P_1, P_2) \prec^C_\theta f(T_1, T_2))$ *avec* f *commutatif alors le résultat est :*

$$(\mathcal{S}ol(P_1 \prec^C_\theta T_1) \oplus \mathcal{S}ol(P_2 \prec^C_\theta T_2)) \otimes (\mathcal{S}ol(P_1 \prec^C_\theta T_2) \oplus \mathcal{S}ol(P_2 \prec^C_\theta T_1))$$

Exemple 16 (Résolution de $\mathcal{S}ol$)**.** .

$$\begin{aligned}
&\mathcal{S}ol(f(g(x,y),h(x,z))) \prec \mathcal{S}ol(f(g(a,b),h(d,a))) \\
=\ &((\mathcal{S}ol(g(x,y) \prec g(a,b))) \oplus (\mathcal{S}ol(h(x,z) \prec h(d,a)))) \\
&\otimes((\mathcal{S}ol(g(x,y) \prec h(d,a))) \oplus (\mathcal{S}ol(h(x,z) \prec g(a,b)))) \\
=\ &(((\mathcal{S}ol(x \prec a) \oplus \mathcal{S}ol(y \prec b)) \otimes (\mathcal{S}ol(x \prec b) \oplus \mathcal{S}ol(y \prec a))) \\
&\otimes((\mathcal{S}ol(x \prec d) \oplus \mathcal{S}ol(z \prec a)) \otimes (\mathcal{S}ol(x \prec a) \oplus \mathcal{S}ol(y \prec d)))) \otimes (\bot\ \oplus\ \bot) \\
=\ &((((\{\{x \leftarrow a\}\} \oplus \{\{y \leftarrow b\}\}) \otimes (\{\{x \leftarrow b\}\} \oplus \{\{y \leftarrow a\}\})) \\
&\otimes((\{\{x \leftarrow d\}\} \oplus \{\{z \leftarrow a\}\}) \otimes (\{\{x \leftarrow a\}\} \oplus \{\{y \leftarrow d\}\})))\otimes \bot \\
=\ &(\{\{x \leftarrow a, y \leftarrow b\}\} \otimes \{\{x \leftarrow b, y \leftarrow a\}\}) \\
&\otimes(\{\{x \leftarrow d, z \leftarrow a\}\} \otimes \{\{x \leftarrow a, y \leftarrow d\}\}) \\
=\ &(\{\{x \leftarrow a, y \leftarrow b\}\} \oplus \{\{x \leftarrow b, y \leftarrow a\}\}) \\
&\otimes(\{\{x \leftarrow d, z \leftarrow a\}\} \oplus \{\{x \leftarrow a, y \leftarrow d\}\}) \\
=\ &(\{\{x \leftarrow a, y \leftarrow b\}, \{x \leftarrow b, y \leftarrow a\}\}) \\
&\otimes(\{\{x \leftarrow d, z \leftarrow a\}, \{x \leftarrow a, y \leftarrow d\}\}) \\
=\ &\{\{\{x \leftarrow a, y \leftarrow b\} \oplus \{x \leftarrow d, z \leftarrow a\}\}, \{\{x \leftarrow a, y \leftarrow b\} \oplus \{x \leftarrow a, y \leftarrow d\}\}, \{\{x \leftarrow \\
&b, y \leftarrow a\} \oplus \{x \leftarrow d, z \leftarrow a\}\}, \{\{x \leftarrow b, y \leftarrow a\} \oplus \{x \leftarrow a, y \leftarrow d\}\}\} \\
=\ &\{\bot, \{x \leftarrow a, y \leftarrow b, z \leftarrow d\}, \bot, \bot\}
\end{aligned}$$

Proposition 5. *Le* ρ_C*-calcul avec* ξ *en paramètre du calcul et avec des motifs algébriques linéaires est confluent modulo la commutativité.*

Démonstration. On considère un filtrage de la forme : $\mathcal{S}ol(A \prec^C_\theta C)$ L'hypothèse $\mathbf{H_0}$ est immédiate. L'hypothèse $\mathbf{H_1}$ est prouvé par induction sur la structure de A.

- Si $A \equiv x$ alors $\theta = \{x\}$. On prend τ avec $\mathcal{V}ar(\tau) \cap \theta = \emptyset$ et par conséquent $x\tau = x$.
  $\mathcal{S}ol(A \prec^C_\theta C) = \sigma = \{x \leftarrow C\}$
  $\mathcal{S}ol(A\tau \prec^C_\theta C\tau) = \mathcal{S}ol(x\tau \prec^C_\theta C\tau) = \{x \leftarrow C\tau\} = \tau \circ \sigma$
- Si $A \equiv f(P_1, P_2))$ avec f commutatif alors nécessairement on a $C \equiv f(T_1, T_2)$
  En appliquant la proposition 4
  on a $\mathcal{S}ol(A \prec^C_\theta C) = (\mathcal{S}ol(P_1 \prec^C_\theta T_1) \oplus \mathcal{S}ol(P_2 \prec^C_\theta T_2)) \otimes (\mathcal{S}ol(P_1 \prec^C_\theta T_2) \oplus \mathcal{S}ol(P_2 \prec^C_\theta T_1))$

Si $\quad \mathcal{S}ol(A\tau \prec^C_\theta C\tau)$ et $\mathcal{V}ar(\tau) \cap \theta = \emptyset$ et $\theta = \mathcal{V}ar(A)$
alors $\quad \mathcal{S}ol(A\tau \prec^C_\theta C\tau) = \mathcal{S}ol(A \prec^C_\theta C\tau)$
$= (\mathcal{S}ol(P_1 \prec^C_\theta T_1\tau) \oplus \mathcal{S}ol(P_2 \prec^C_\theta T_2\tau))$
$\otimes(\mathcal{S}ol(P_1 \prec^C_\theta T_2\tau) \oplus \mathcal{S}ol(P_2 \prec^C_\theta T_1\tau))$
avec $\quad \mathcal{S}ol(P_1 \prec^C_\theta T_1\tau) = (\tau \circ \sigma^1_1)$
$\mathcal{S}ol(P_2 \prec^C_\theta T_2\tau) = (\tau \circ \sigma^2_2)$
$\mathcal{S}ol(P_1 \prec^C_\theta T_2\tau) = (\tau \circ \sigma^1_2)$
$\mathcal{S}ol(P_2 \prec^C_\theta T_1\tau) = (\tau \circ \sigma^2_1)$

On déduit que :

$((\tau \circ \sigma^1_1) \oplus (\tau \circ \sigma^2_2)) \otimes ((\tau \circ \sigma^1_2) \oplus (\tau \circ \sigma^2_1)) = \tau \circ (\sigma^1_1 \oplus \sigma^2_2 \otimes \sigma^1_2 \oplus \sigma^2_1) = \{(\tau \circ \sigma_i)_{|\theta}\}^n_{i=1}$

L'hypothèse **H₂** est prouvé par induction sur la structure de A.

– Si $A \equiv x$ et $A \twoheadrightarrow A'$ alors $A' \equiv x$. Donc si $C \twoheadrightarrow C'$ on a $\sigma = \{x \leftarrow C\}$ et $\sigma' = \{x \leftarrow C'\}$ donc $\sigma \twoheadrightarrow \sigma'$

– Si $A \equiv f(P_1, \ldots, P_n))$ avec f commutatif alors nécessairement
on a $\quad C \equiv f(T_1, \ldots, T_n)$
et $\quad \mathcal{S}ol(A \prec^C_\theta C) = (\mathcal{S}ol(P_1 \prec^C_\theta T_1) \oplus \mathcal{S}ol(P_2 \prec^C_\theta T_2))$
$\otimes(\mathcal{S}ol(P_1 \prec^C_\theta T_2) \oplus \mathcal{S}ol(P_2 \prec^C_\theta T_1)) = \{\sigma_i\}^n_{i=1}$
et $\quad \mathcal{S}ol(A' \prec^C_\theta C') = (\mathcal{S}ol(P'_1 \prec^C_\theta T'_1) \oplus \mathcal{S}ol(P'_2 \prec^C_\theta T'_2))$
$\otimes(\mathcal{S}ol(P'_1 \prec^C_\theta T'_2) \oplus \mathcal{S}ol(P'_2 \prec^C_\theta T'_1)) = \{\sigma'_j\}^n_{j=1}$
avec $\quad C \twoheadrightarrow C'$ et $A \twoheadrightarrow A'$

A et un motif algébrique linéaire donc $A \equiv A'$ et on conclut que
et $\quad \mathcal{S}ol(A' \prec^C_\theta C') = (\mathcal{S}ol(P_1 \prec^C_\theta T'_1) \oplus \mathcal{S}ol(P_2 \prec^C_\theta T'_2))$
$\otimes(\mathcal{S}ol(P_1 \prec^C_\theta T'_2) \oplus \mathcal{S}ol(P_2 \prec^C_\theta T'_1))$
Donc $\quad \forall i, j\ \sigma_i \twoheadrightarrow \sigma'_j$

L'hypothèse **H₃** est prouvé par induction sur la structure de A.

– Si $A \equiv x$ et $A \sim_C A'$ alors $A' \equiv x$. Donc si $C \sim_C C'$ on a $\sigma = \{x \leftarrow C\}$ et $\sigma' = \{x \leftarrow C'\}$ donc $\sigma \sim_C \sigma'$

– Si $A \equiv f(P_1, \ldots, P_n))$ avec f commutatif alors nécessairement
identique au cas précédent.
avec $\quad C \sim_C C'$ et $A \sim_C A'$

On conclut que
$\mathcal{S}ol(A' \prec^C_\theta C') = (\mathcal{S}ol(P_1 \prec^C_\theta T'_1) \oplus \mathcal{S}ol(P_2 \prec^C_\theta T'_2))$
$\otimes(\mathcal{S}ol(P_1 \prec^C_\theta T'_2) \oplus \mathcal{S}ol(P_2 \prec^C_\theta T'_1))$
donc $\quad \forall i, j\ \sigma_i \sim_C \sigma'_j$

La confluence de la relation (β_ρ) avec filtrage commutatif est par conséquent obtenue par le theorème 1. La relation δ termine. Elle est également localement confluente. Donc, les relations $\twoheadrightarrow_{\beta_\rho}$ et (δ) satisfont le diagramme de Yokouchi-Hikita (par induction sur la structure des termes) et le théorème 6 conclut la preuve.

□

Conclusion et perspectives

Le but du stage était de prouver la confluence des calculs à motifs en utilisant un filtrage non unitaire. Les travaux effectués montrent que cela est possible dans le cas où la théorie de filtrage utilisée est la commutativité. Bien sûr on imagine que la preuve est possible pour d'autres théories de filtrage. Les prochains travaux devraient tenir compte d'autres théories telles que l'associativité, puis mêler deux théories pour prendre en compte l'associativité et la commutativité. Le calcul de réécriture décrit comme une instance du λ-calcul dynamique à motifs dans la section 2.3 n'autorise pas l'agrégation en tant que motif. Il serait intéressant de prouver la confluence dans ce cas également.

Ces calculs avec motifs peuvent manipuler des motifs dynamiques, c'est-à-dire qui peuvent être instanciés et réduits. Ils conduisent à des langages fonctionnels avec de nouvelles formes de polymorphisme [Jay04]. Notre étude s'est limitée au cas du filtrage modulo la commutativité. L'utilisation du filtrage modulo une théorie équationnelle permet d'écrire des programmes hautement déclaratifs, comme c'est la cas dans le langage Tom [BBK+06] implémentant le filtrage sur les listes (filtrage associatif avec élément neutre). L'étude réalisée ici dans le cas d'un filtrage non unitaire a permis une compréhension nouvelle du mécanisme de filtrage et plus particulièrement dans son interaction avec les collections de résultats (indispensables dans le cas du filtrage non unitaire).

Ces travaux sont très formels, ils ont pour but de faire progresser notre connaissance des calculs à motifs car on souhaite approfondir la sémantique des stratégies de réécriture. Pour cela l'étude de la sémantique des calculs avec motifs, initialisée dans ce rapport, est intéressante.

Bibliographie

[Bal06] Vincent Balat. Ocsigen : typing web interaction with objective caml. In Andrew Kennedy and FranÃ§ois Pottier, editors, *ML*, pages 84–94. ACM, 2006.

[Bar84] Hendrik Pieter Barendregt. *The Lambda Calculus – Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.

[BBK+06] Émilie Balland, Paul Brauner, Radu Kopetz, Pierre-Étienne Moreau, and Antoine Reilles. The Tom manual, 2006. `http://tom.loria.fr/soft/release-2.4/manual-2.4/index.html`.

[BCKL03] Gilles Barthe, Horatiu Cirstea, Claude Kirchner, and Luigi Liquori. Pure patterns type systems. In *Principles of Programming Languages - POPL2003, New Orleans, USA*. ACM, January 2003.

[BK86] J A Bergstra and J W Klop. Conditional rewrite rules : Confluence and termination. *J. Comput. Syst. Sci.*, 32(3) :323–362, 1986.

[Bor] Peter Borovanský. Controlling rewriting : Study and implementation of a strategy formalism. pages 63–74.

[Cir00] Horatiu Cirstea. *Calcul de réécriture : fondements et applications*. Thèse de Doctorat d'Université, Université Henri Poincaré - Nancy I, 2000.

[CK98] Horatiu Cirstea and Claude Kirchner. Combining higher-order and first-order computation using ρ-calculus : Towards a semantics of ELAN. In Dov Gabbay and Maarten de Rijke, editors, *Frontiers of Combining Systems 2*, Research Studies, ISBN 0863802524, appeared in 1999, pages 95–120. Wiley, October 1998.

[CK01] Horatiu Cirstea and Claude Kirchner. Rewriting and Multisets in Rho-calculus and ELAN. *Romanian Journal of Information, Science and Technology*, 4(1-2) :33–48, 2001. ISSN : 1453-8245.

[CKL01] Horatiu Cirstea, Claude Kirchner, and Luigi Liquori. Matching power. In *RTA*, pages 77–92, 2001.

[CLW03] Horatiu Cirstea, Luigi Liquori, and Benjamin Wack. Typed recursive rewriting calculus. To be published. Available at `http ://www.loria.fr/~wack/papers/full-TypedRecursive.ps.gz`, 2003.

[CMV+06] Francois Clément, Vincent Martin, Arnaud Vodicka, Roberto Di Cosmo, and Pierre Weis. Domain decomposition and skeleton programming with ocamlp3l. *Parallel Computing*, 32(7-8) :539–550, 2006.

[Fau07] Germain Faure. *Structures et modÃ¨les de calculs de rÃ©Ã©criture*. PhD thesis, UniversitÃ© Henri PoincarÃ© & Loria Inria Nancy Grand Est, 2007.

[Fri06] Alain Frisch. Ocaml + xduce. In *ICFP*, pages 192–200, 2006.

[HJM06] ThÃ©rÃ¨se Hardin, Mathieu Jaume, and Charles Morisset. Access control and rewrite systems. In *Proceedings of the 1st International Workshop on Security and Rewriting Techniques (SecReT'06)*, Venice, Italy, July 2006.

[HLL07] Furio Honsell, Marina Lenisa, and Luigi Liquori. A framework for defining logical frameworks. In *Computation, Meaning and Logic*. Elsevier Science Publishers, 2007.

[Jay04] Barry Jay. The pattern calculus. *ACM Trans. Program. Lang. Syst.*, 26(6) :911–937, 2004.

[JK06] Barry Jay and Delia Kesner. Pure pattern calculus. In Peter Sestoft, editor, *ESOP*, volume 3924 of *Lecture Notes in Computer Science*, pages 100–114. Springer, 2006.

[Kir04] Claude Kirchner. Strategic rewriting. In *4th International Workshop on Reduction Strategies in Rewriting and Programming - WRS'2004, Aachen, Germany*, Jun 2004.

[KK99] Claude Kirchner and Hélène Kirchner. Rewriting, solving, proving, 1999.

[KK04] Claude Kirchner and Hélène Kirchner. Rule-based programming and proving : The ELAN experience outcomes. In Michael J. Maher, editor, *Advances in Computer Science - ASIAN 2004, Higher-Level Decision Making, 9th Asian Computing Science Conference, Dedicated to Jean-Louis Lassez on the Occasion of His 5th Cycle Birthday*, volume 3321 of *Lecture Notes in Computer Science*, pages 363–379, Chiang Mai, Thailand, December 8-10 2004. Springer.

[KKM07] Claude Kirchner, Radu Kopetz, and Pierre-Etienne Moreau. Anti-pattern matching. In Rocco De Nicola, editor, *ESOP*, volume 4421 of *Lecture Notes in Computer Science*, pages 110–124. Springer, 2007.

[MOMV06] Narciso MartÃ-Oliet, JosÃ© Meseguer, and Alberto Verdejo. Towards a strategy language for maude. In *P6th International Workshop on Rewriting Logic and its Applications - WRLA'06*, volume 117. Electronic Notes in Theoretical Computer Science, 2006.

[Ohl98] Ohlebusch. Modularity of termination for disjoint term graph rewrite systems : A simple proof. *BEATCS : Bulletin of the European Association for Theoretical Computer Science*, 66, 1998.

[Pey87] Simon Peyton-Jones. *Implementation of Functional Programming Languages*. Prentice-Hall, 1987.

[Pol95] Robert Pollack. Polishing up the Tait-Martin-Löf proof of the Church-Rosser theorem. 1995.

[Rei07] Antoine Reilles. Canonical abstract syntax trees. *Electr. Notes Theor. Comput. Sci.*, 176(4) :165–179, 2007.

[Rus06] MichaÃ■l Rusinowitch. Deciding protocol insecurity with rewriting techniques. In *Proceedings of the 1st International Workshop on Security and Rewriting Techniques (SecReT'06)*, Invited Talk, 2006.

[Vis01] Eelco Visser. Stratego : A language for program transformation based on rewriting strategies. System description of Stratego 0.5. In A. Middeldorp, editor,

Rewriting Techniques and Applications (RTA'01), volume 2051 of *Lecture Notes in Computer Science*, pages 357–361. Springer-Verlag, May 2001.

[Vis05] Eelco Visser. A survey of strategies in rule-based program transformation systems. *Journal of Symbolic Computation*, 40(1) :831–873, 2005. Special issue on Reduction Strategies in Rewriting and Programming.

[vO90] V. van Oostrom. Lambda calculus with patterns, 1990.