Un typage pour assurer une borne sur la complexité des programmes

LEVENTIS Thomas

Table des matières

I Introduction **2**

II Logiques linéaires et représentations graphiques **3**

1 La logique linéaire LL **3**
1.1 Nouveaux connecteurs 3
1.2 Règles de réduction 3
1.3 Logiques dérivées 4

2 Représentation graphique d'une preuve **4**
2.1 Transformation d'une preuve en graphe 4
2.2 Aspect d'un réseau de preuve 5

3 EAL,LAL : des logiques bornées **5**
3.1 Logique affine élémentaire 6
3.2 Logique affine légère 7

4 Graphe de partage d'un λ-terme **7**
4.1 Partage d'un graphe 8
4.2 Propagation des multiplexeurs 8

III Le problème : réduction d'un λ-terme typable dans EAL ou LAL **9**

5 Les systèmes de typages étudiés **10**
5.1 Typage dans EAL 10
5.2 Typage dans LAL 11

6 Les réductions de réseaux et de graphes partagés **12**
6.1 Elimination des coupures dans une preuve de EAL 12
6.2 Réduction de Lamping d'un λ-terme 13

IV Eléments de résolution **14**

7 Faire le lien entre élimination des coupures et reécriture de graphe **14**

8 Modifications de UN et correspondance avec EAL et SG **15**
8.1 Etude de la propagation des lifts 15
8.2 Problèmes liés au comptage du coût de duplication dans EAL 16
8.3 Un système de coloration des lifts 18

V Conclusion **22**

Première partie
Introduction

Un domaine de recherche actuel est la construction d'un langage de programmation dans lequel le type des programmes donne une borne sur leur complexité.

Dans ce but, des logiques bornées ont vu le jour, telles que la *logique affine élémentaire* (EAL) ou la *logique affine légère* (LAL), dans lesquelles on dispose de bornes sur le nombre d'étapes nécessaires pour éliminer les coupures d'une preuve : ce temps est borné par une fonction élémentaire de la taille de la preuve dans EAL, et par un fonction polynomiale dans LAL. Par correspondance de Curry-Howard, on dispose ainsi de stratégies bornées pour réduire un λ-terme typable dans EAL ou LAL.

Cependant, toute réduction d'un λ-terme typable en logique bornée ne correspond pas à une élimination des coupures d'une preuve de cette logique. Or Lamping a proposé un système de réduction des λ-termes, optimal au sens de Lévy, auquel on aimerait pouvoir appliquer ce résultat.

Une borne sur la complexité de l'algorithme de Lamping a déjà été prouvé par Baillot, Coppola et Dal Lago en utilisant des principes de géométrie d'interaction. Le travail que j'ai effectué, en collaboration avec Marco SOLIERI et sous la tutelle de Stefano GUERRINI, a consisté à chercher une nouvelle preuve de ce résultat, qui soit la plus simple possible et ne fasse pas appel à la géométrie d'interaction.

La partie II présente la logique linéaire et des graphes partagés de manière générale. Une part conséquente de mon stage ayant consisté en la découverte de ces domaines, sans avoir encore formulé de question précise, cette présentation va au-delà de ce qu'il est nécesssaire de comprendre pour aborder le problème traité dans le reste de ce rapport.

La partie III présente le problème abordé, en définissant rigoureusement certains points qui n'ont été qu'évoqués dans la partie précédente.

Dans la partie IV est décrit le travail que nous avons effectué pour répondre à ce problème.

Deuxième partie

Logiques linéaires et représentations graphiques

1 La logique linéaire LL

La logique linéaire, introduite par Jean-Yves GIRARD [4], est souvent décrite comme une logique "avec ressources". L'implication linéaire $\multimap$ "consomme" son hypothèse, si bien qu'à partir de A, A $\multimap$ B et A $\multimap$ C on peut obtenir B et on peut obtenir C, mais pas les deux.

1.1 Nouveaux connecteurs

En plus des connecteurs usuels &, $\oplus$, $\exists$ et $\forall$, appelés les *connecteurs additifs*, on introduit le *tenseur* $\otimes$ et le *par* $\parr$. L'implication linéaire A $\multimap$ B correspond à $A^\perp \parr B$, et $\otimes$ est le dual de $\parr$. $\multimap$, $\parr$ et $\otimes$ sont les *connecteurs multiplicatifs* de la logique linéaire.

Une logique purement linéaire est cependant extrêmement faible. Pour compenser ce déficit, on introduit des connecteurs unaires ! (bien sûr !) et ? (pourquoi pas ?), appelés *exponentiels*, permettant d'échapper à cette contrainte. ?A correspond à une "source" de A illimitée, et ! est le connecteur dual. Les formules ?A échappent aux contraintes de linéarité et peuvent être affaiblies ou dupliquées librement. La formule !A s'obtient quant à elle en prouvant A avec uniquement des hypothèses de la forme ?B.

1.2 Règles de réduction

Voici les règles de la logique linéaire, où $A^\perp$ désigne le dual de A :

Règles d'identité :

$$\frac{}{\vdash A, A^\perp}\ \text{Axiome} \qquad \frac{\vdash \Gamma, A \qquad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta}\ \text{Coupure}$$

Règles additives :

$$\frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}\ \& \qquad \frac{\vdash \Gamma, A_i}{\vdash \Gamma, A_1 \oplus A_2}\ \oplus_i$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \forall x : A}\ \forall,\ x \notin \mathrm{FV}(\Gamma) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, \exists x : A}\ \exists$$

Règles multiplicatives :

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\ \otimes \qquad \frac{\vdash \Gamma, A_1, A_2}{\vdash \Gamma, A_1 \parr A_2}\ \parr$$

Règles exponentielles :

$$\frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\ \text{Contraction} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\ \text{Affaiblissement}$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\ \text{Déréliction} \qquad \frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\ \text{Promotion}$$

1.3 Logiques dérivées

La logique linéaire LL n'est pas toujours considérée dans son ensemble. On se restreint parfois par exemple à sa partie multiplicative, ce qui donne la logique MLL, où aux parties multiplicative et exponentielle, qui donnent la logique MELL.

Il est également possible d'ajouter à LL une règle d'affaiblissement valable pour n'importe quelle formule, et non uniquement pour les formules ?A : $\dfrac{\vdash \Gamma}{\vdash \Gamma, A}$

On parle alors de logique affine.

Les logiques EAL et LAL que nous allons étudier sont des logiques affines particulières. Dans le cas de LAL, on introduit en plus un nouveau connecteur §, dérivé de !. Une première description de ces logiques sera faite dans la section II.3, et le détail de leurs règles sera donné dans la partie III.

2 Représentation graphique d'une preuve

2.1 Transformation d'une preuve en graphe

La représentation d'une preuve au moyen d'un arbre de dérivation présente des inconvénients, en ce sens qu'elle contient plus d'informations que nécessaire à l'étude de l'élimination des coupures. Ainsi il n'est pas nécessaire de différencier les preuves suivantes :

$$\dfrac{\vdash \Gamma, A \qquad \dfrac{\vdash \Delta, B, C \qquad \vdash \Theta, D}{\vdash \Delta, \Theta, B, C \otimes D}}{\vdash \Gamma, \Delta, \Theta, A \otimes B, C \otimes D} \quad \text{et} \quad \dfrac{\dfrac{\vdash \Gamma, A \qquad \vdash \Delta, B, C}{\vdash \Gamma, \Delta, A \otimes B, C} \qquad \vdash \Theta, D}{\vdash \Gamma, \Delta, \Theta, A \otimes B, C \otimes D}$$

Pour avoir une représentation non séquentielle des preuves, on utilise des graphes, dont les sommets correspondent aux connecteurs logiques et où une les arêtes sont étiquetées par des formules logiques.

On ne s'intéresse ici qu'à la logique linéaire multiplicative exponentielle (MELL), c'est-à-dire LL sans les connecteurs additifs. Les graphes correspondant aux règles de MLL sont décrits dans la figure 1.

La partie exponentielle, et plus précisément la règle de promotion, pose plus de problèmes car elle n'est pas locale : la contrainte est que toutes les formules autres que celle promue soient sous la forme "pourquoi pas ?". On introduit donc une notion de *boîte*, chaque boîte possédant un port principal correspondant à la formule promue et un nombre quelconque de ports auxiliaires dont les arêtes sont étiquetées par des formules ?B.

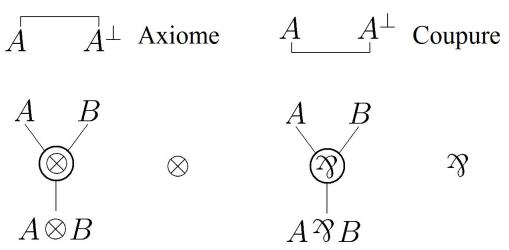


Figure 1 : Parties identitaire et multiplicative des réseaux de preuve

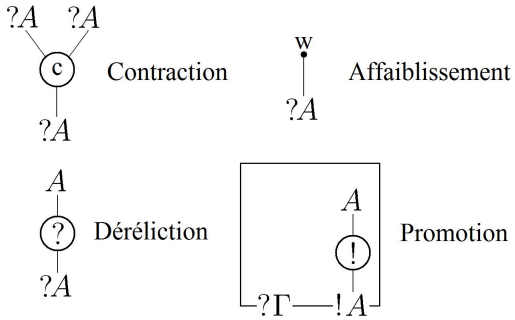


Figure 2 : Partie exponentielle des réseaux de preuve

Remarque : Les réseaux tels que décrits ici ne sont pas exactement des graphes. En effet il existe des arêtes dont l'une des extrémités n'est reliée à aucun noeud. Pour une définition rigoureuse des réseaux, on les considère comme des hypergraphes où les "noeuds" sont des hyperarcs et les "arêtes" sont des sommets de degré 1 ou 2 (voir [5]).

2.2 Aspect d'un réseau de preuve

Des critères de correction ont été établis par Danos et Regnier [6] pour caractériser les graphes utilisant ces noeuds qui correspondent effectivement à l'image d'une preuve de MELL. De tels graphes sont appelés *réseaux de preuve*.

Pour le problème qui nous intéresse, nous n'avons pas besoin de détailler ces critères. Le seul point indispensable pour comprendre à quoi ressemble un réseau de preuve est la *condition d'imbrication des boîtes* : dans un réseau de preuve, étant donné deux boîtes B et B', on a soit B $\subset$ B', soit B $\supset$ B', soit B $\cap$ B' $= \emptyset$. De plus B et B' n'ont pas le même port principal.

La structure de boîte permet de définir des *niveaux* dans un réseau de preuve. Le niveau d'une arête correspond au nombre de boîtes dans lequel elle est incluse. La notion de niveau est facilement compréhensible intuitivement, nous n'allons donc pas en donner une définition rigoureuse.

3 EAL,LAL : des logiques bornées

Pour notre problème, nous ne nous intéressons qu'à deux logiques particulières dérivées de la logique linéaire, les logiques EAL et LAL. Leur point commun est d'avoir été construites de telle sorte que pour toute preuve Π, la nombre d'étapes pour éliminer les

coupures de Π est borné par une fonction, élémentaire pour EAL, polynomiale pour LAL, de la taille de Π.

Lorsque l'on considère le système de reécriture de graphe correspondant à l'élimination des coupures, on s'aperçoit que la règle permettant de construire des réductions longues à partir d'un réseau de taille réduite est la règle suivante :

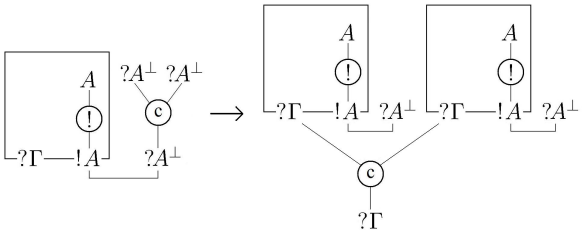


Figure 3 : Duplication d'une boîte dans LL

Il s'agit en effet de la seule règle qui ne diminue pas la taille du réseau, mais qui au contraire la fait croître. C'est donc sur la forme des boîtes que jouent les logiques EAL et LAL pour limiter le nombre de duplications et donc la complexité de l'élimination des coupures.

3.1 Logique affine élémentaire

L'idée de la logique affine élémentaire EAL est de limiter l'augmentation des niveaux des noeuds des réseaux lors de l'élimination des coupures. En effet le niveau d'un noeud peut changer lorsque l'on utilise la réduction suivante :

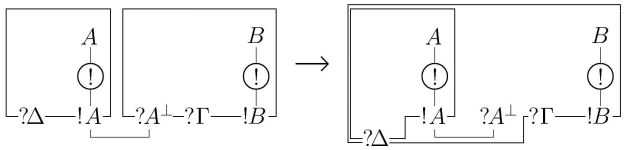


Figure 4 : Augmentation du niveau d'un noeud dans LL

Dans EAL, on considère donc les mêmes règles que dans la logique affine en changeant simplement la règle de promotion, qui devient :

$$\frac{\vdash \Gamma, A}{\vdash ?\Gamma, !A}\,\text{Promotion}$$

L'élimination des coupures entre deux boîtes devient alors :

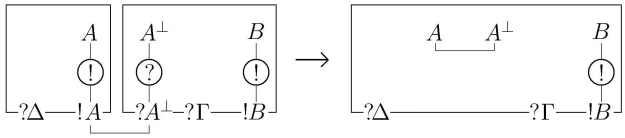


Figure 5 : Conservation du niveau dans EAL

En effectuant ce changement, on obtient une logique dans laquelle le niveau ne peut pas changer :

Proposition 1 : Pour tout réseau de preuve de EAL, le niveau des noeuds ne change pas au cours de l'élimination des coupures.

Grâce à cette propriété, on peut borner le nombre de fois où une boîte peut être dupliquée en fonction de son niveau, pour obtenir le théorème suivant :

Théorème 1 : Pour tout δ il existe une fonction élémentaire f_δ telle que pour tout réseau N de EAL de profondeur maximale δ, pour toute réduction $\rho : N \to N'$, $c(\rho) \leq f_\delta(\|N\|)$ et $\|N'\| \leq f_\delta(\|N\|)$, où c est la fonction de coût dans EAL.

3.2 Logique affine légère

La logique affine légère conserve également les règles de la logique affine mais impose une contrainte encore plus forte sur la règle de promotion. Cette règle devient :

$$\frac{\vdash B, A}{\vdash ?B, !A} \qquad\qquad \frac{\vdash A}{\vdash !A}$$

Une promotion ne peut être effectuée que si le résultat A n'est obtenu qu'à partir d'au plus une hypothèse B. Cependant avec cette version de la promotion, on obtient une logique très faible. On utilise donc un nouveau connecteur §, qui est une version affaiblie des connecteurs ? et !, avec la règle suivante :

$$\frac{\vdash \Gamma, \Delta}{\vdash \S\Gamma, ?\Delta}$$

On dispose alors de résultats similaires à ceux dans EAL :

Proposition 2 : Pour tout réseau de preuve de LAL , le niveau des noeuds ne change pas au cours de l'élimination des coupures.

Théorème 2 : Pour tout δ il existe une fonction polynomiale g_δ telle que pour tout réseau N de LAL de profondeur maximale δ, pour toute réduction $\rho : N \to N'$, $c(\rho) \leq g_\delta(\|N\|)$ et $\|N'\| \leq g_\delta(\|N\|)$.

4 Graphe de partage d'un λ-terme

De même que les preuves peuvent être représentées sous forme de réseau, les λ-termes sont habituellement représentés par des arbres utilisant les noeuds λ et @. Cependant si l'on utilise cette représentation, la règle de β-réduction ne correspond pas à une transformation élémentaire de l'abre correspondant.

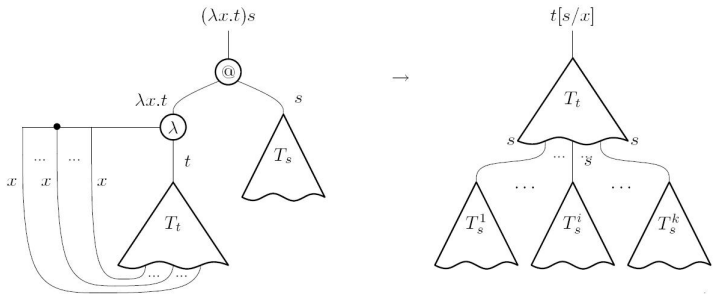
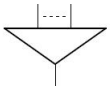


Figure 6 : β-réduction instantanée

La représentation par *graphe de partage* utilise un nouveau type de noeud permettant de partager une partie du graphe d'un λ-terme entre plusieurs sous-termes. Cette nouvelle représentation a le double avantage de permettre de tranformer la β-réduction en transformation élémentaire et de faire le lien entre un graphe de λ-terme typé et le réseau d'une preuve de son type.

4.1 Partage d'un graphe

Lors d'une β-réduction, chaque occurence de la variable liée doit être remplacée par l'argument passé à la λ-abstraction. Pour effectuer cette opération de manière élémentaire sur les graphes, on ne duplique pas le sous-arbre de l'argument en le recopiant une fois pour chaque occurence de la variable, mais on dit que toutes les occurences de la variable se partagent le même sous-arbre.

En pratique on utilise des multiplexeurs, que l'on représente ainsi : 
La β-réduction s'exprime alors de la manière suivante :

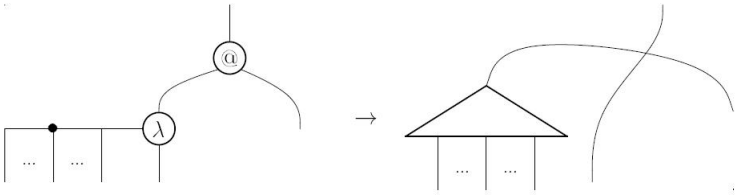


Figure 7 : β-réduction élémentaire

4.2 Propagation des multiplexeurs

Simplement modifier la β-réduction n'est pas suffisant. En effet un graphe contenant de nombreux multiplexeurs n'est pas facile à lire et, plus important, ce système n'est pas complet vis-à-vis du λ-calcul usuel. Une β-réduction peut normalement créer de nouveaux rédexs, or ici la présence des multiplexeurs s'y oppose.

Il est donc nécessaire d'ajouter des règles de propagation des multiplexeurs, qui traduisent correctement la notion de partage. Pour des multiplexeurs binaire, ces règles sont :

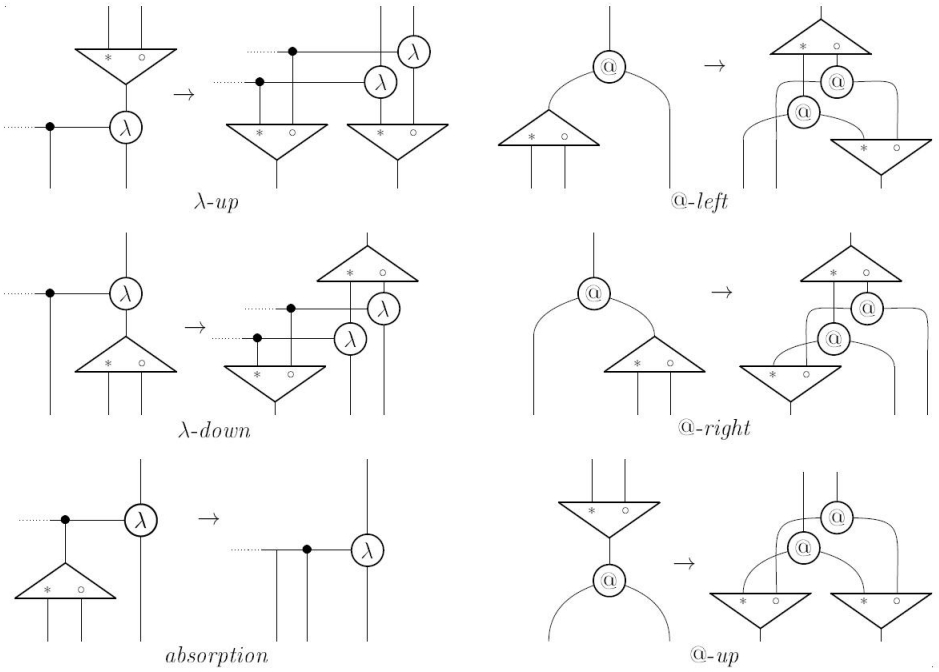


Figure 8 : Propagation des multiplexeurs

Le seul cas problématique est celui de deux multiplexeurs se faisant face. Deux choix sont alors possibles :

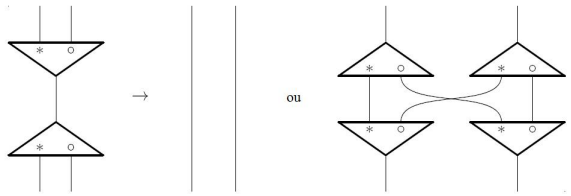


Figure 9 : Rencontre de multiplexeurs

Le problème est de savoir si les multiplexeurs proviennent de la même β-réduction, c'est-à-dire s'ils délimitent la même zone de partage, ou non. Si c'est la cas ils s'éliminent, sinon il se croisent en se dupliquant.

Pour différencier ces deux cas, il est possible de nommer les multiplexeurs introduits par chaque β-réduction. Une autre solution est d'utiliser un lien avec les réseaux. En faisant ainsi, on peut associer un niveau à chaque multiplexeur, qui ne change jamais au cours de la propagation, et on montre alors que deux multiplexeurs se faisant face s'anihilent si et seulement si ils ont le même niveau.

Le lien entre graphe partagé et réseau est bien plus complexe dans le cas général que dans le cas des logiques EAL et LAL, nous ne le détaillerons donc que dans ces cas particuliers.

Troisième partie

Le problème : réduction d'un λ-terme typable dans EAL ou LAL

5 Les systèmes de typages étudiés

5.1 Typage dans EAL

On considère le système de typage suivant :

$$\frac{}{x : A \vdash x : A}\,\mathrm{A} \qquad \frac{\Gamma \vdash t : A \qquad \Delta, x : A \vdash u : B}{\Gamma, \Delta \vdash u\{t/x\} : B}\,\mathrm{U}$$

$$\frac{\Gamma \vdash t : B}{\Gamma, x : A \vdash t : B}\,\mathrm{W} \qquad \frac{\Gamma, x :!A, y :!A \vdash t : B}{\Gamma, z :!A \vdash t\{z/x, z/y\} : C}\,\mathrm{X}$$

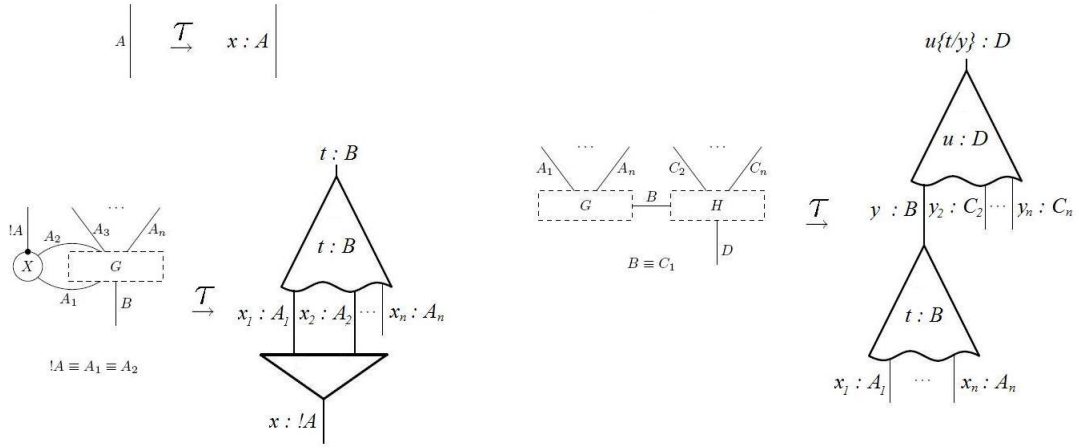
$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B}\,\mathrm{R}_{\multimap} \qquad \frac{\Gamma \vdash t : A \qquad \Delta, x : B \vdash u : C}{\Gamma, \Delta, y : A \multimap B \vdash u\{yt/x\} : C}\,\mathrm{L}_{\multimap}$$

$$\frac{\Gamma \vdash t : A}{!\Gamma \vdash t :!A}\,\mathrm{P}_!$$

$$\frac{\Gamma \vdash t : A \qquad \alpha \notin FV(\Gamma)}{\Gamma \vdash t : \forall\alpha.A}\,\mathrm{R}_{\forall} \qquad \frac{\Gamma, x : A\{B/\alpha\} \vdash t : C}{\Gamma, x : \forall\alpha.A \vdash t : C}\,\mathrm{L}_{\forall}$$

$$\frac{\Gamma \vdash t : A\{\mu\alpha.A/\alpha\}}{\Gamma \vdash t : \mu\alpha.A}\,\mathrm{R}_{\mu} \qquad \frac{\Gamma, x : A\{\mu\alpha.A/\alpha\} \vdash t : B}{\Gamma, x : \mu\alpha.A \vdash t : B}\,\mathrm{L}_{\mu}$$

Chaque preuve du type d'un λ-terme est une preuve de EAL, dont on peut construire le réseau. On construit ensuite une transformation τ_{EAL} des réseaux dans les graphes partagés, qui au réseau d'un preuve de typage associe une représentation du λ-terme typé. On définit τ_{EAL} ainsi, où $t = \tau_{EAL}(G)$ et $u = \tau_{EAL}(H)$:



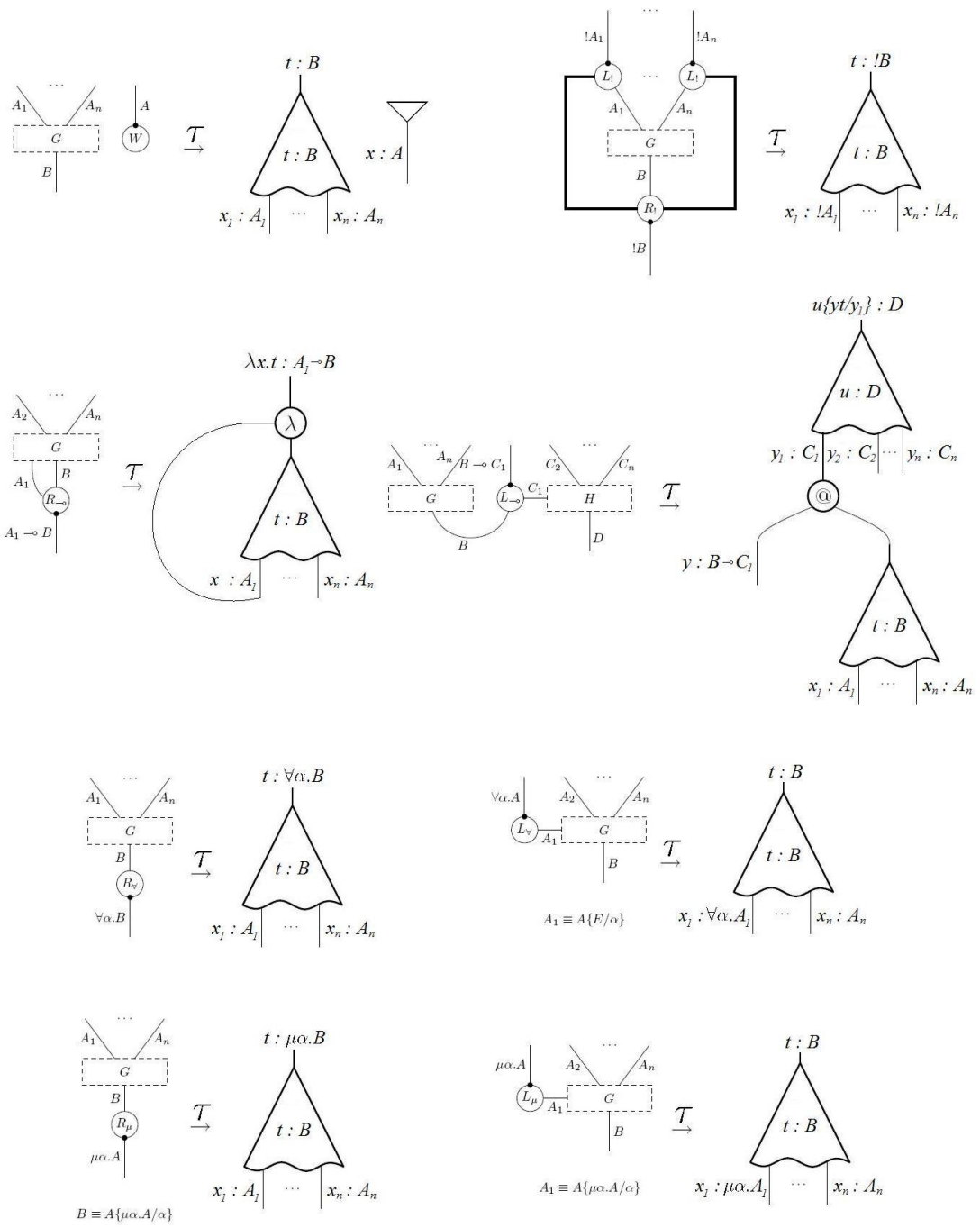


Figure 10 : Transformation τ de EAL dans SG

5.2 Typage dans LAL

Le typage dans LAL utilise les règles A, U, W, X, $R_\multimap$, $L_\multimap$, $R_\forall$, $L_\forall$, R_μ et L_{mu} définies précédemment, et la règle $P_!$ devient :

$$\frac{\vdash t : A}{\vdash t :!A}\,P_!^1 \qquad \frac{x : B \vdash t : A}{x :!B \vdash t :!A}\,P_!^2 \qquad \frac{\Gamma, \Delta \vdash t : A}{\S\Gamma, !\Delta \vdash t :\S A}\,P_\S$$

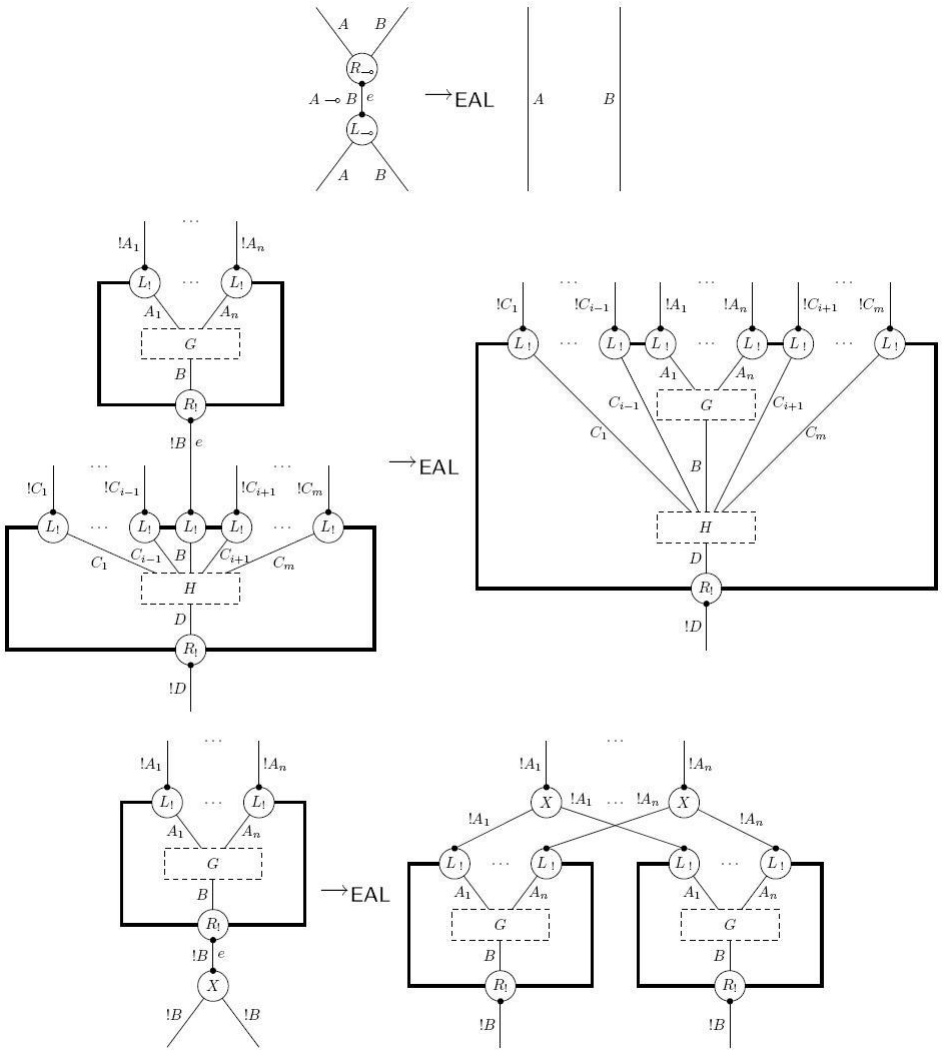
On peut définir une fonction τ_{LAL} similaire à τ_{EAL}. Cependant si dans une preuve de LAL Π on remplace tous les symboles § par des symboles !, on obtient une preuve Π' correcte dans EAL, et l'on a $\tau_{LAL}(\Pi) = \tau_{EAL}(\Pi')$. Comme de plus cette transformation de Π vers Π' est correcte vis-à-vis de l'élimination des coupures dans LAL et EAL, on peut se contenter de raisonner dans EAL. Toute borne obtenue en fonction de f_δ sur les termes typables dans EAL est valable dans LAL en remplaçant f_δ par g_δ.

Par la suite on se place donc dans EAL, et il sera aisément vérifié que la borne élémentaire obtenue pour la réduction des λ-termes de EAL devient une borne polynomiale en remplaçant f_δ par g_δ. Pour simplifier on utilisera la notation τ pour désigner τ_{EAL}.

6 Les réductions de réseaux et de graphes partagés

6.1 Elimination des coupures dans une preuve de EAL

Le système d'élimination des coupures dans EAL est :



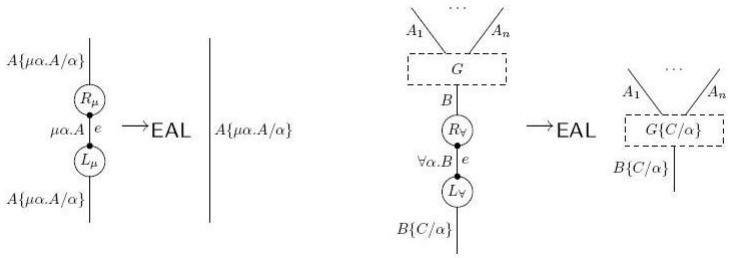


Figure 11 : Réductions dans EAL

Toute étape de réduction a un coût de 1, sauf lorsque l'on duplique une boîte contenant un réseau G, auquel cas le coût est $\|G\|+1$. On sait alors que le coût de la réduction d'un réseau N de profondeur maximale δ est borné par $f_\delta(\|N\|)$.

6.2 Réduction de Lamping d'un λ-terme

Lamping a montré que le système de reécriture des graphes partagés dont les règles sont les règles λ-up, @-left, les règles d'échange et d'élimination des multiplexeurs et la β-réduction définies à la section II.4 (où le niveau d'un multiplexeur dans un terme $\tau(N)$ est le niveau du noeud X correspondant dans N) est optimal au sens de Lévy, c'est-à-dire que toute normalisation de λ-terme s'effectue en utilisant un nombre minimal de fois la β-réduction. Cependant il n'affirme rien quant au coût réel de la réduction, qui correspond au nombre total d'étapes effectuées.

Notre objectif est de borner la longueur de n'importe quelle réduction, pas uniquement les réductions optimales. On considère le système de réduction des graphes partagés dans son ensemble, à une modification près : les multiplexeurs ne sont pas créés par la β-réduction mais préexistent, et par conséquent la règle d'absorption doit être suprimée. De plus les multiplexeurs utilisés, étant obtenus par application de la fonction τ à un noeud X, seront tous binaires (les arités plus élevées pouvant être assimilés à des arbres de multiplexeurs binaires) ou d'arité nulle. Le système obtenu est :

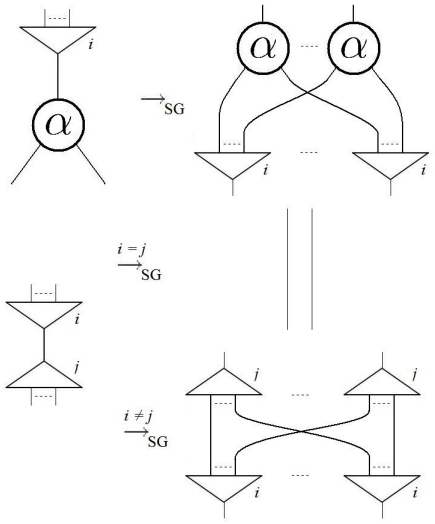


Figure 12 : Règles de réduction de UN

où si $\alpha = \lambda$, le port d'arrivée du multiplexeur n'est pas celui correspondant à la variable de l'abstraction. On appelle ces règles les règles de *propagation*, d'*annihilation* et d'*échange*. La β-réduction est quant à elle identique à son équivalent dans EAL, c'est-à-dire l'élimination des coupures entre les noeuds $R_{\multimap}$ et $L_{\multimap}$.

Le résultat que nous voulons prouver est :

Théorème 3 : Pour tout type α dans EAL, il existe une fonction élémentaire h_α telle que pour tout λ-terme t de type α, si G un graphe de t et $G \rightarrow^n_{SG} G'$ alors $n \leq h_\alpha(\|G\|)$.

Quatrième partie
Eléments de résolution

7 Faire le lien entre élimination des coupures et reécriture de graphe

L'idée principale pour prouver ce résultat est d'associer à chaque réduction de graphe une réduction dans les réseaux de coût comparable, afin de pouvoir utiliser le théorème 1. La principale différence entre les deux réductions est que dans le cas des réseaux, la duplication des noeuds par un noeud de contraction X s'effectue de manière globale, tandis que dans les graphes partagés toutes les duplications sont élémentaires.

On introduit donc un système intermédiaire, que l'on appelle les *réseaux non partagés* (UN), dans lequel le coût de la duplication de boîtes est compté par des étapes élémentaires. L'étape de duplication réelle a donc un coût de 1, mais introduit des noeuds semblables à des multiplexeurs unaires, qui en se propageant vont "compter" la taille de la boîte dupliquée.

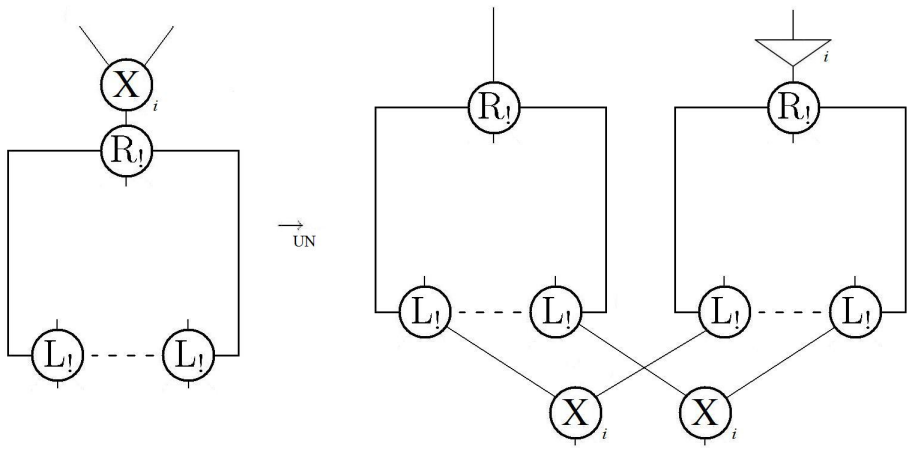
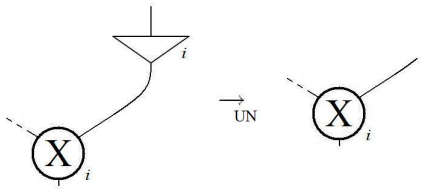


Figure 13 : Création des lifts

Ces noeuds sont appelés des *lifts*. Leurs règles de propagation et d'échange, sont les mêmes que pour les multiplexeurs classiques, avec une règle supplémentaire :



Ce système à l'avantage de pouvoir effectuer à la fois des duplications de boîte, comme dans les réseaux, et un parcours pas-à-pas des noeuds dupliqués, comme dans les graphes partagés. Le lien avec les réseaux en particulier est très fort, puisque les réductions dans UN sont des réductions de EAL décorées" par des lifts. Cependant la définition du coût d'une réduction étant différente dans UN et dans EAL, la comparaison de ces coûts n'est pas immédiate.

Pour lier UN à SG, l'idée est d'associer tout d'abord à chaque multiplexeur un noeud de contraction X. Lorsqu'un multiplexeur se propage, on peut montrer que le noeud X associé forme un rédex avec une boîte, et l'on peut donc effectuer une duplication, ce qui crée un lift. On associe alors ce lift au multiplexeur, les deux se propageant selon des règles similaires. Cependant cette correspondance qui semble naturelle n'est pas aisée à définir précisément.

8 Modifications de UN et correspondance avec EAL et SG

Le fait que les réseaux non partagés comptent correctement le coût réel de la réduction d'un réseau de EAL n'est pas immédiat. En effet rien n'impose aux lifts de parcourir leur boîte dès qu'elle a été copiée, si bien que le nombre de noeuds comptés peut être différent du nombre de noeuds présents au moment de la copie. Pour cette raison il est nécessaire de rajouter des éléments aux réseaux non partagés, tout en vérifiant que l'on maintient une certaine cohérence avec le comportement de SG.

8.1 Etude de la propagation des lifts

Pour bien comprendre comment fonctionnent les réseaux non partagés, et pour répérer les problèmes qui se posent dans la correspondance avec EAL et SG, il est nécessaire de bien connaître le comportement global des lifts au cours d'une réduction de UN.

Dans cette partie, on parlera de noeud pour désigner tout noeud différent d'un lift.

Comme le système UN n'est pas encore totalement défini, il n'est pas possible de prouver des propriétés précises concernant ce comportement. Les résultats suivants seront donc principalement descriptifs.

Intéressons nous au cas d'une unique duplication, et donc où un seul lift est créé par une duplication, les autres provenant des règles de propagation. Avant tout, les lifts doivent compter la taille de la boîte dupliquée. Ce décompte correspond aux règles de propagation, et par conséquent chaque noeud de la boîte doit intervenir une fois et une seule dans une règle de propagation. On doit donc pouvoir à tout moment séparer la boîte en deux parties : d'une part les noeuds comptés et d'autre part les noeuds à compter. Il semble de plus naturel de penser que ces deux parties doivent être séparées par des lifts, de telle sorte que :

- tout lift fait face à un noeud non compté (ou à l'extérieur de la boîte) et derrière se trouve un noeud compté (ou l'extérieur)
- deux noeuds adjacents sont tous deux comptés ou tous deux à compter.

Schématiquement, la boîteà l'aspect décrit dans la figure 14.

Il peut être surprenant de voir que les noeuds comptés ne forment pas un ensemble connexe. Ceci provient de la β réduction, qui peut isoler des lifts.

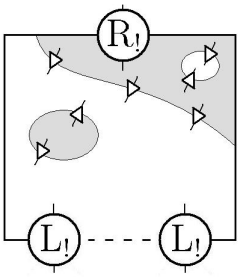


Figure 14 : Aspect d'une boîte en cours de comptage

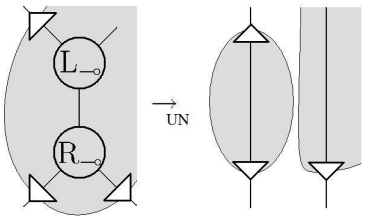
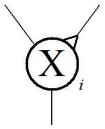


Figure 15 : Isolement de lifts

Dans le cas général où plusieurs duplications sont effectuées, les choses sont plus compliquées. Le comportement précédent est toujours valide si l'on seulement considère les lifts introduits par l'une des duplications, si ce n'est qu'une boîte en train d'être comptée peut être dupliquée en tant qu'élément d'une boîte plus grande, si bien que le comptage s'effectue non plus dans une seule mais dans deux boîtes à la fois.

8.2 Problèmes liés au comptage du coût de duplication dans EAL

Limite de propagation des lifts Un problème que pose UN tel que décrit précédemment est que les lifts cessent de se propager lorsqu'ils rencontrent un copie du noeud X qui les a créés. Or ce noeud peut être impliqué dans une nouvelle étape de duplication avant d'avoir été rejoint par les lifts, ce qui accroît le domaine parcouru par les lifts avant de disparaître, et donc augmente de manière non voulue le coût de la réduction. Il serait possible de faire correspondre les lifts non pas à des noeuds X mais aux noeuds de sortie de boîte $L_!$, mais le même problème se pose alors lorsque l'on effectue la fusion de deux boîtes.

La solution que j'ai proposée est de considérer une variante 
des noeuds de contraction X, des noeuds dits *en attente* :

Un tel noeud se comporte comme un noeud de contraction classique, si ce n'est que lorsqu'il duplique une boîte, il ne crée pas de lift. Ses règles d'introduction et d'élimination sont :

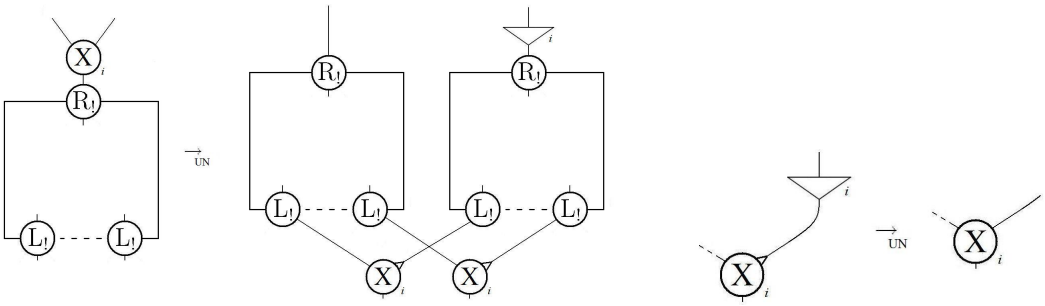


Figure 16 : Création et activation de noeuds X en attente

De cette manière lorsqu'un noeud en attente duplique une boîte, cela signifie qu'il existe un lift qui aurait dû le rejoindre, et par conséquent ce lift ne s'arrêtera pas une fois son précédent travail terminé mais il commencera à compter le coût de cette nouvelle duplication.

Remarque : Si dans une boîte il n'y a pas de chemin entre le noeud $R_!$ et un noeud $L_!$, ce qu'il est possible de réaliser en utilisant la règle d'affaiblissement, une partie de la boîte ne sera pas parcourue par les lifts. De plus le noeud X correspondant restera toujours en attente, si bien que les coûts des duplications qu'il effectuera ne seront pas pris en compte. Cependant dans SG, ces duplications n'ont en réalité pas lieu, on peut donc penser qu'il est normal de ne pas en compter le coût.

De manière générale, les problèmes liés aux affaiblissements n'ont pas été étudiés en détail durant ce stage.

Elimination de noeuds Même si les lifts se propagent à l'intérieur de limites bien définies, le contenu du domaine qu'ils parcourent peut évoluer. Les cas les plus simples sont ceux de l'élimination des connecteurs $\multimap$, $\forall$ et μ. A l'exception du fait que la β-réduction modifie la structure de la boîte et peut par exemple isoler des lifts tel qu'il a été vu plus tôt, ces règles consistent simplement à effacer deux noeuds.

Comme tout noeud est normalement compté au plus une fois par un lift (car il n'est créé que par une seule étape de duplication), la variation du coût d'une réduction de UN par rapport à une réduction de EAL due à l'élimination de noeuds avant qu'ils ne soient comptés est bornée par $2 * \#\{\textit{étapes d'élimination de noeuds}\}$.

Duplication de boîtes imbriquées La principale difficulté survient lorsque, si B et B' sont deux boîtes, B est en cours de comptage et l'on duplique B' avec B $\subset$ B' ou B' $\subset$ B. Ce qui se passe alors est que tout ou une partie de la boîte la plus petite sera pris en compte dans le coût des deux duplications, celle qui a créé B et celle de B'.

Si B $\subset$ B', une solution apparemment simple est de dire que lorsque l'on copie une boîte, on ne copie pas les lifts qu'elle contient, ou bien on les "noircit" et l'on dit que la propagation d'un lift noir se fait avec un coût nul, afin d'éviter de compter le contenu de la copie de B une fois de trop.

Le cas B' $\subset$ B est plus complexe, et remet même en cause la solution du cas précédent. Trois cas de figure sont possibles :

- B' a déjà été comptée dans B, auquel cas il n'y a pas de problème.
- B' n'a pas été comptée dans B, B' et sa copie vont donc toutes deux être parcourues par des lifts de B. Il est alors possible de ne pas introduire de lift pour la duplication de B', ou bien de dire d'un façon ou d'une autre que la copie de B' est "fermée" aux lifts de B, pour que le coût de la duplication soit compté soit par les lifts parcourant B, soit par ceux de la duplication de B', mais pas les deux.
- B' est en train d'être comptée par les lifts de B. Il ne faut alors pas copier les lifts de B avec B', mais également préciser que ces lifts ne doivent pas par la suite se propager dans la copie de B', ce qui implique un remaniement des règles de propagation.

Marco et Stefano ont proposé diverses solutions utilisant des lifts blancs, noirs ou gris qui, selon leur couleur, ont ou non un coût de propagation nul. Dans tous les cas, la difficulté est ensuite de faire le lien avec les graphes partagés, et de s'assurer que pour chaque multiplexeur on puisse trouver un lift associé dont le coût de propagation est non nul. J'ai personnellement mis au point un système qui, il me semble, permet de lier les complexités des systèmes EAL et SG.

8.3 Un système de coloration des lifts

Annotation des noeuds. On se donne initialement un réseau N_0 et le graphe correspondant $G_0 = \tau(N_0)$. On associe alors :

- à chaque noeud de G_0 un nom qui lui est propre parmi un ensemble E, et l'on donne le même nom à son antécédent dans N_0
- à chaque noeud de G_0 et N_0 un *historique* $h \subset E$, avec initialement $h = \emptyset$
- à tout multiplexeur, noeud X, et par la suite à tout lift, un *chemin*, qui est un mot $c \in \{0,1\}^*$, avec initialement $c = \epsilon$
- à tout noeud et lift de N_0 un *futur* $f \subset E$, avec initialement $f = \emptyset$
- à tout noeud X en attente un nom (a) avec $a \in E$.

On appelle *annotation* l'ensemble des informations associées à un noeud. Les noeuds $R_!$, $L_!$, $R_\forall$, $L_\forall$, R_μ et L_μ de N_0 n'ont pas besoin d'être annotés.

On considère les règles pour SG et UN :

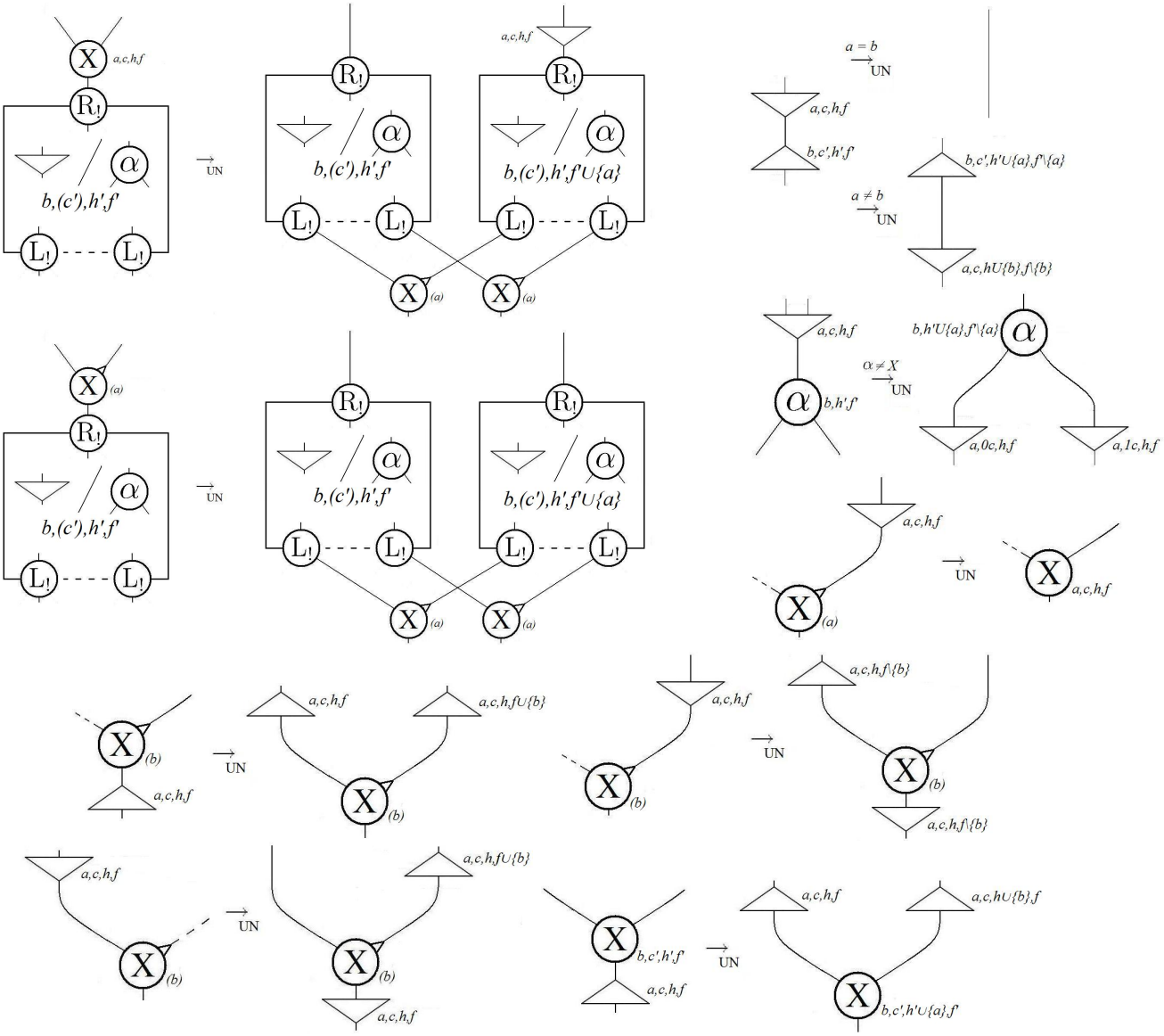


Figure 17 : Régles d'annotation dans UN

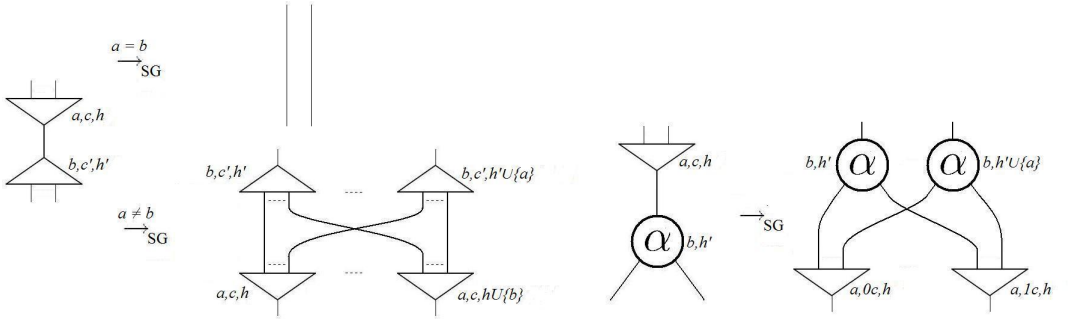


Figure 18 : Régles d'annotation dans SG

On conserve par ailleurs toutes les règles de réduction ne concernant pas les lifts, les noeuds X ni les multiplexeurs. Comme on ne s'intéresse pas aux affaiblissements, il n'y a pas de multiplexeur d'arité nulle.

L'idée est d'identifier les différents noeuds, de telle sorte que la seule donnée des annotations suffise à savoir exactement comment un noeud a été créé. Ainsi :

- le nom désigne le noeud dont il provient dans le graphe d'origine
- l'historique indique par qui il a été copié
- dans le cas des noeuds "mobiles", le chemin permet de retracer le parcours du noeud
- dans UN, comme les duplications s'effectuent en deux temps (un temps pour la copie effective et un temps de comptage du coût de la duplication), le futur indique si un noeud est dans une copie en train d'être comptée.

Les noeuds $R_!$, $L_!$, $R_\forall$, $L_\forall$, R_μ et L_μ ne sont pas annotés car ils n'ont pas d'image dans SG et, étant binaires, ils n'ont aucune influence sur la dynamique des lifts, et leur évolution ne nous intéresse donc pas.

Le résultat fondamental suivant concernant les annotations n'a pas encore été prouvé :

Proposition 3 : Si $N_0 \multimap^*_{UN} N'$ et $G_0 \multimap^*_{SG} G'$, étant deux noeuds distincts α_1 et α_2 tous deux dans N' ou tous deux dans G', les annotations de α_1 et α_2 sont distinctes.

Simulation d'une réduction de SG. On définit alors une fonction τ qui, étant donnés un réseau N et un graphe G annotés, associe :

- à tout lift ou noeud X de N d'annotation a, c, h, f l'ensemble des multiplexeurs de G d'annotation a, c, h
- à tout autre noeud de N d'annotation a, h, f l'ensemble des noeuds λ ou @ de G d'annotation a, h.

Remarque : Dans le cadre de la proposition 3, τ associe à un noeud de N' au plus un noeud de G'. En particulier si on considère N_0 et G_0, τ correspond à la fonction τ_{EAL}, d'où son appellation.

On définit ensuite la *simulation* $s(\rho)$ dans UN d'une réduction ρ de SG par induction : $s(\epsilon_{SG}) = \epsilon_{UN}$ et $s(\rho r) = s(\rho)\sigma$ où σ est définie par :

- si r est une β-réduction entre deux noeuds α_1 et α_2, on effectue les β-réductions entre les éléments de $\tau^{-1}(\alpha_1)$ et $\tau^{-1}(\alpha_2)$ ayant le même futur.
- si r est la propagation d'un multiplexeur m avec un noeud α, on effectue une duplication pour tout noeud X dans $\tau_{-1}(m)$. $\tau_{-1}(m)$ ne contient alors plus que des lifts, et chacun effectue une étape de propagation avec un noeud de $\tau_{-1}(\alpha)$.
- si r est l'annihilation de multiplexeurs, on effectue l'annihilation des lifts corrrespondants
- si r est un échange de multiplexeurs, on effectue l'échange ou la propagation à travers un noeud X des lifts correspondants.

Dans tous les cas on peut également avoir à effectuer un certain nombre d'éliminations de coupures entre noeuds binaires, de propagations de lifts avec des noeuds binaires ou d'activations de noeuds X en attente.

Le fait que la notion de simulation est bien définie n'est pas évident. Cela vient du fait que, si $G_0 \to_\rho G'$ et $N_0 \to_{s(\rho} N'$:

- pour tout nom $a \in E$, l'ensemble des noeuds dont le futur contient a est délimité par les ports droits de noeuds en attente (a) et par des lifts a
- le même ensemble dans lequel on ôte a des futurs des noeuds est aussi un sous-graphe de N', et les arêtes qui étaient liées au port droit d'un noeud (a) sont liées au port gauche de ce même noeud
- connaissant N', on peut retrouver G' en fusionnant ces deux ensembles pour tous les noeuds en attente et en effaçant les noeuds binaires.

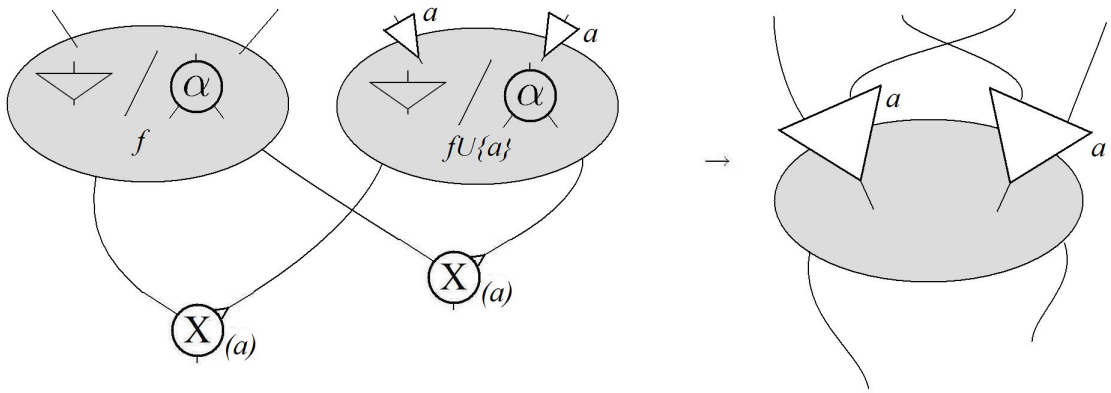


Figure 19 : Réductions parallèles dans UN et SG

Un autre problème est de montrer qu'il possible de faire abstraction des noeuds binaires. Cela provient de ce que si l'on considère une chaîne de noeuds binaires dont les extrémités ont le même type, il est possible d'effacer ces noeuds par une succession de réductions.

Coût d'une réduction Il est possible de définir le coût d'une réduction de UN de telle sorte que pour toute réduction ρ de SG, le coût de $s(\rho)$ soit exactement la longueur de ρ.

On définit $c(\sigma)$, où σ est une réduction de UN, comme le nombre d'étapes de σ n'impliquant que des noeuds dont le futur est $\emptyset$ (les noeuds n'ayant pas de futur n'ont donc aucune influence sur le coût de la réduction). En ce qui concerne la propagation des lifts, et pour reprendre les idées développées dans de précédents modèles, un lift est considéré comme blanc si son futur est vide, et comme noir sinon.

Du comportement de N' au cours d'une simulation décrit ci-dessus, on voit que pour chaque noeud $a, (c), h$ de G', il existe $F \subset E$ tel que $\tau_{-1}(a, (c), h) = \{a, (c), h, f \mid f \subset F\}$. Par conséquent il existe toujours un et un seul noeud de $\tau_{-1}(a, (c), h)$ de futur vide. Comme les noeuds sont rassemblés selon leur futur dans N', pour tout rédex dans G' il existe exactement un rédex correspondant dans N' dans lequel les noeuds impliqués ont un futur vide.

L'intérêt de la notion de futur est d'assurer que lorsque deux multiplexeurs se croisent dans SG, il est possible d'associer au moins un lift à chacun d'entre eux. Ceci est la principale difficulté de la simulation dans UN, évoquée à la fin de la partie IV.8.2, puisque l'on crée un multiplexeur dans SG sans qu'il n'y ait de création équivalente dans UN. Ici, lorsque deux multiplexeurs a et b se croisent, on associe à la copie de a tous les lifts

précédemment associés à a qui avaient b dans leur futur, et l'on a vu qu'exactement l'un d'entre eux à un futur vide, et donc un coût de propagation non nul.

Alors chaque fois que l'on effectue une réduction dans SG, il y a exactement une étape de la simulation dans UN dont le coût vaut 1, les autres étapes ayant un coût nul, d'où $c(s(\rho)) = \#\rho$.

Correspondance avec EAL En changeant légèrement la définition du coût d'une réduction de UN, on peut obtenir le même résultat concernant EAL.

Cette fois toute étape de réduction ne concernant pas les lifts doit être prise en compte, et a donc un coût de 1. La seule difficulté pour faire le lien entre EAL et UN est d'assurer que les lifts ne comptent pas plusieurs fois un même noeud. Ceci peut être fait en disant que les étapes de propagation n'ont un coût que si le noeud concerné a un futur à un seul élément ou, de manière équivalente, si le lift qui se propage a un futur vide.

On obtient alors un coût c' tel que, pour toute réduction N $\rightarrow^*_{UN\sigma}$ N' et pour toute réduction équivalente N $\rightarrow^*_{EAL\sigma'}$ N" (c'est-à-dire la même réduction sans création ni propagation de lift), on a $c'(\sigma) = cot(\sigma')$.

Remarque : Comme dit à la section IV.8.2, on a $c'(\sigma) \leq cot(\sigma') \leq 3 * c'(\sigma)$ si les étapes d'élimination de noeuds ont un coût de 1, mais on peut obtenir une égalité si l'in considère qu'éliminer deux noeuds dont les futurs sont non vides à un coût de 3.

Le problème est que c' ne doit pas tenir compte des étapes d'échange et d'annihilation des lifts. Par conséquent on ne peut pas affirmer que $c(\sigma) \leq c'(\sigma)$, ce qui permettrait de borner la longueur de toute réduction de SG par la longeur d'un réduction de EAL, et donc par une fonction élémentaire.

Cinquième partie
Conclusion

Au cours de nos travaux, nous avons imaginé de nombreuses variantes du système UN qui permettent avec plus ou moins de précision de comparer les coûts des réductions de SG et EAL. Cependant aucune n'est parfaite. En effet tous ces sytèmes ont pour principe de décomposer le coût d'une duplication dans EAL en étapes élémentaires, or ceci permet certes de ce coût avec le coût de propagation des multiplexeurs dans SG, mais ne dit sur les coûts d'échange et d'annihilation de multiplexeurs, qui n'ont aucun équivalent dans EAL.

Ce travail n'est pas vain pour autant. L'étude de ces systèmes permet de bien comprendre les similitudes de comportement entre les preuves de la logique linéaire d'un côté et les représentations partagées de λ-termes de l'autre. Qui plus est, il est peut-être

possible d'utiliser les réseaux non partagés pour borner ce coût qui nous échappe pour l'instant. Nous y travaillons toujours, notre piste principale étant d'essayer de montrer que le coût des échanges peut être reporté sur d'autres étapes de réductions, par exemple sur les étapes de propagation, durant lesquelles les lifts sont créés.

Références

[1] Andrea Asperti, Luca Roversi Intuitionistic Light Affine Logic *ACMTransactions on Computational Logic*, 3(1) :1–39, 2002

[2] P. Baillot, P. Coppola, U. Dal Lago. Light Logics and Optimal Reduction : Completeness and Complexity. *Information and Computation*, 209(2) :118–142, 2011.

[3] Vincent Danos, Jean-Baptiste Joinet. Linear Logic and elementary time *Information and Computation*, 183 :123–137, 2003

[4] Jean-Yves Girard Linear logic : Its syntax and semantics *Advances in Linear Logic*, 1–42, 1995

[5] Stefano Guerrini. Theoretical and Practical Issues of Optimal Implementations of Functionnal Languages Ph.D. Thesis, 1996

[6] Stefano Guerrini Proof-nets and the λ-calculus *Linear Logic in Computer Science*, 316 :65–118, 2004

[7] Stefano Guerrini Sharing Implementations of Graph Rewriting Systems Version préliminaire