Presentation et evaluation de la complexite en moyenne des algorithmes d'unification

Luc Albert

INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 1212

Programme 1
Programmation, Calcul Symbolique
et Intelligence Artificielle

PRESENTATION ET EVALUATION DE LA COMPLEXITE EN MOYENNE DES ALGORITHMES D'UNIFICATION

Luc ALBERT

Avril 1990

★RR-1212★

PRESENTATION ET EVALUATION DE LA COMPLEXITE EN MOYENNE DES ALGORITHMES D'UNIFICATION

Luc ALBERT *

Résumé. *L'unification dans les langages du premier ordre est une opération fondamentale en calcul symbolique et en programmation logique. On dispose de beaucoup d'algorithmes d'unification, mais il n'y a pas de consensus quant à savoir lequel est le meilleur à utiliser en pratique. L'algorithme linéaire de Paterson Wegman a la plus faible complexité dans le cas le pire, néanmoins son implémentation nécessite un prétraitement important. C'est aussi le cas, dans une moindre mesure, de l'algorithme de Martelli Montanari [MM82] et finalement, c'est souvent l'algorithme de Robinson [Rob71] qui est employé dans beaucoup d'applications malgré sa complexité exponentielle dans le cas le pire. Cette situation s'explique de plusieurs façons : un fait important est que, dans la pratique, les sous-problèmes d'unification ne sont pas indépendants et les algorithmes d'unification linéaires ne sont pas très performants sur les séquences d'unification-desunification [MU86]. Dans cet article, nous présentons des arguments théoriques basés sur l'étude de la complexité en moyenne. Nous montrons, tout d'abord, que la famille des paires d'arbres binaires unifiables est exponentiellement négligeable devant la famille des paires d'arbres binaires formés à partir de l symboles de fonction binaires, c constantes et v variables. Nous analysons les différentes causes d'échec et nous obtenons des évaluations asymptotiques et numériques. Nous généralisons ensuite les résultats de [DL89] pour ces familles de termes et nous montrons qu'une légère modification de l'algorithme de Herbrand-Robinson a une complexité moyenne constante sur les paires d'arbres quelconques. D'autre part, nous montrons que l'algorithme de Martelli Montanari et une de ses améliorations ont une complexité moyenne linéaire sur les paires d'arbres quelconques. Les échecs par clash ne sont en effet pas suffisants pour assurer une complexité moyenne constante, il est nécessaire de disposer d'un test d'occurrence efficace (i.e. qui ne nécessite pas un parcours complet des sous-arbres). Enfin, dans la dernière partie, nous généralisons nos résultats sur la probabilité d'occurrence à un modèle avec un nombre infini de variables.*

PRESENTATION AND AVERAGE CASE ANALYSIS OF UNIFICATION ALGORITHMS

Abstract. *Unification in first-order languages is a central operation in symbolic computation and logic programming. Many unification algorithms have been proposed in the past, however there is no consensus on which algorithm is the best to use in practice. While Paterson and Wegman's linear unification algorithm has the lowest time complexity in the worst case, it necessitates an important overhead to be implemented. This is true also, although less importantly, for Martelli and Montanari's algorithm [MM82], and Robinson's algorithm [Rob71] is finally retained in many applications despite its exponential worst-case time complexity. There are many explanations for that situation, one important argument is that in practice unification subproblems are not independent, and linear unification algorithms do not perform well on sequences of unify-deunify operations [MU86]. In this paper we present average case complexity theoretic arguments. We show first that the family of unifiable pairs of binary trees is exponentially negligible with respect to the family of arbitrary pairs of binary trees formed over l binary function symbols, c constants and v variables. We analyze the different causes of failure and get asymptotical and numerical evaluations. Then we extend the previous results of [DL89] to these families of trees, we show that a slight modification of Herbrand-Robinson's algorithm has a constant average cost on random pairs of trees. On the other hand, we show that various variants of Martelli and Montanari's algorithm all have a linear average cost on random pairs of trees. The point is that failures by clash are not sufficient to lead to a constant average cost, an efficient occur check (i.e. without a complete traversal of subterms) is necessary. In the last section we extend the results on the probability of the occur check in presence of an unbounded number of variables.*

* Institut National de Recherche en Informatique et Automatique, Domaine de Voluceau, Rocquencourt, BP 105, 78150 Le Chesnay Cedex France. mail : albert@inria.inria.fr

PRESENTATION ET EVALUATION DE LA COMPLEXITE EN MOYENNE DES ALGORITHMES D'UNIFICATION

Luc ALBERT [1]

Résumé. *L'unification dans les langages du premier ordre est une opération fondamentale en calcul symbolique et en programmation logique. On dispose de beaucoup d'algorithmes d'unification, mais il n'y a pas de consensus quant à savoir lequel est le meilleur à utiliser en pratique. L'algorithme linéaire de Paterson Wegman a la plus faible complexité dans le cas le pire, néanmoins son implémentation nécessite un prétraitement important. C'est aussi le cas, dans une moindre mesure, de l'algorithme de Martelli Montanari [MM82] et finalement, c'est souvent l'algorithme de Robinson [Rob71] qui est employé dans beaucoup d'applications malgré sa complexité exponentielle dans le cas le pire. Cette situation s'explique de plusieurs façons : un fait important est que, dans la pratique, les sous-problèmes d'unification ne sont pas indépendants et les algorithmes d'unification linéaires ne sont pas très performants sur les séquences d'unification-desunification [MU86]. Dans cet article, nous présentons des arguments théoriques basés sur l'étude de la complexité en moyenne. Nous montrons, tout d'abord, que la famille des paires d'arbres binaires unifiables est exponentiellement négligeable devant la famille des paires d'arbres binaires formés à partir de l symboles de fonction binaires, c constantes et v variables. Nous analysons les différentes causes d'échec et nous obtenons des évaluations asymptotiques et numériques. Nous généralisons ensuite les résultats de [DL89] pour ces familles de termes et nous montrons qu'une légère modification de l'algorithme de Herbrand-Robinson a une complexité moyenne constante sur les paires d'arbres quelconques. D'autre part, nous montrons que l'algorithme de Martelli Montanari et certaines de ses améliorations ont une complexité moyenne linéaire sur les paires d'arbres quelconques. Les échecs par clash ne sont en effet pas suffisants pour assurer une complexité moyenne constante, il est nécessaire de disposer d'un test d'occurrence efficace (i.e. qui ne nécessite pas un parcours complet des sous-arbres). Nous obtenons en effet que, dans un modèle avec un nombre de variables bornés, l'occurrence est une cause d'échec numériquement inférieure mais de même ordre que le clash. Enfin, dans la dernière partie, nous présentons un développement combinatoire du problème pour des termes arborescents avec une infinité de variables au renommage près. Ceci nous permet de généraliser nos résultats sur la probabilité d'occurrence à un modèle avec un nombre de variables non borné.*

[1] Institut National de Recherche en Informatique et Automatique, Domaine de Voluceau, Rocquencourt, BP 105, 78150 Le Chesnay Cedex France. mail : albert@inria.inria.fr

I. Introduction

La résolution d'équations sur des termes d'un langage du premier ordre est une opération fondamentale en calcul symbolique. Herbrand [Her30] a été le premier à étudier ce problème dans ses travaux sur la théorie de la démonstration et c'est Robinson qui le baptisa *unification* dans ses travaux fondateurs sur la démonstration automatique en logique du premier ordre [Rob65]. Aujourd'hui, l'unification des termes du premier ordre est une opération centrale pour une grande diversité de programmes allant des systèmes de démonstration automatique [CL73], [KB70], [HO80], aux interpréteurs pour langages de programmation logique [Kow74], [Col84], au contrôle de types pour les langages fonctionnels [Mil78], [Mai89], aux analyseurs syntaxiques en langage naturel [Col78], aux systèmes d'apprentissage, etc... Tous ces domaines d'application ont motivé la recherche d'algorithmes d'unification efficaces ainsi que les extensions de l'unification dans les langages d'ordre supérieur [Hue75], [Hue76], l'unification dans les théories équationnelles [Plo72], [FH86], [BJSS89] par exemple l'unification en présence d'opérateurs associatifs, commutatifs [Sti75], [Fag84], etc...

L'algorithme d'unification de Robinson [Rob65] a pour entrée deux termes du premier ordre et produit en sortie : soit une substitution la plus générale sur les variables des termes d'entrée qui égalise ces derniers, soit un échec s'ils ne sont pas unifiables. Il est bien connu que la complexité dans le cas le pire de l'algorithme de Robinson est exponentielle en la taille des termes d'entrée, même si la substitution produite est représentée sous une forme triangulaire et si l'on utilise des techniques de partage [Rob71] afin de représenter les termes par des graphes orientés acycliques (Directed Acyclic Graphs ou DAG) (des travaux récents [FSS90] ont montré que le gain moyen du partage maximum était de l'ordre de $n/\sqrt{\ln n}$ avec n la taille du terme à compacter). La raison de cette complexité exponentielle dans le cas le pire pour cet algorithme est que , même si la taille du DAG reste constante, ce dernier, sans technique de marquage, est encore parcouru comme un arbre. Lorsque l'on utilise des techniques de marquage, la complexité dans le cas le pire devient alors quadratique en la taille des données d'entrée [VZ75], [CB83]. En améliorant ces idées, Martelli et Montanari [MM82] ont obtenu un algorithme d'unification en $O(n + v \ln v)$ où n est la taille des termes en entrée et v leur nombre de variables distinctes; de leur côté, Paterson et Wegman ont obtenu un *algorithme linéaire* [PW78] (découvert également par Martelli et Montanari de façon indépendante [MM76]).

En considérant l'unification comme un cas particulier de clôture sur des classes de termes, Huet [Hue76] a obtenu un algorithme d'unification quasi-linéaire basé sur l'algorithme bien connu du *set-union-find* [AHU74] (gestion dynamique d'équivalence). Des variantes et des implémentations en sont données dans [Fag83], [Col84], [Jaf84] et [EG88]. L'algorithme de Huet réalise l'unification sur des arbres réguliers [Cou83], *i.e.* sur des termes finis ou infinis représentés par un graphe fini qui peut être cyclique. L'unification sur les arbres réguliers a conduit à l'implémentation d'une variante du langage de programmation logique Prolog qui manipule des expressions infinies [Col82]. On peut utiliser l'algorithme de Huet pour réaliser l'unification sur des termes finis en rajoutant un test de circularité en phase finale. Etant linéaire, ce test ne change pas la complexité dans le cas le pire de l'algorithme, c'est à dire $O(n^2)$, $O(n \ln(n))$ ou $O(nG(n))$, où G est une fonction à croissance très lente (un inverse de la fonction d'Ackermann), suivant les différentes stratégies utilisées dans l'algorithme du set-union-find (tableau des nœuds pères, équilibrage et compression du parcours).

[DKM84] ont montré que l'unification est un problème P-complet en temps. Ce résultat signifie que l'unification est intrinsèquement un problème séquentiel et que l'on n'obtiendra pas de gain significatif avec une implémentation parallèle. Ce résultat est encore valide pour l'unification sur les arbres réguliers infinis, mais dans ce cas il faut noter que l'algorithme séquentiel de Paterson Wegman ne s'applique plus. L'existence d'un algorithme d'unification linéaire pour les expressions infinies reste un problème ouvert (il en est de même pour tester l'équivalence des automates finis déterministes).

D'un point de vue pratique, il n'y a pas de consensus quant à savoir quel algorithme d'unification est le meilleur à utiliser. L'algorithme de Paterson Wegman a la plus faible complexité dans le cas le pire, néanmoins son implémentation nécessite un prétraitement important. C'est aussi le cas dans une moindre mesure de l'algorithme de Martelli Montanari en raison de la phase d'initialisation des compteurs d'occurrence des variables dans les termes (cf. partie II). L'algorithme de Huet dans sa version en $O(n^2)$ ne présente pas un prétraitement très important en comparaison avec l'algorithme de Robinson qui, d'autre part, révèle son comportement exponentiel uniquement sur des exemples pathologiques mais *pas dans la pratique*. De plus, dans les applications où les cas de non-unifiabilité sont prépondérants, la capacité à détecter efficacement les échecs peut se révéler plus importante que la complexité dans le cas le pire. Toutes ces raisons expliquent pourquoi l'algorithme de Robinson reste encore employé dans beaucoup de systèmes de démonstration automatique, dans Prolog (avec usuellement l'omission discutable du test d'occurrence, cf. partie II), dans le contrôle de de types [CAML89], etc ...

Dans cet article, nous essayons de préciser les arguments précédents en analysant *en moyenne* les algorithmes d'unification dans un modèle de distribution uniforme c'est à dire avec l'hypothèse que tous les termes de même taille sont équiprobables. Dans la partie II, nous détaillons l'algorithme non déterministe de Herbrand à partir duquel nous déduisons les algorithmes de Robinson et de Martelli Montanari. Dans la partie III, nous montrons que la famille des paires d'arbres unifiables, formés à partir de l symboles fonctionnels binaires, c constantes et v variables, est exponentiellement négligeable devant la famille des paires d'arbres binaires quelconques. En conséquence, nous analysons avec précision les différentes causes d'échec dont nous obtenons des estimations asymptotiques et numériques.

Dans la partie IV, nous généralisons les précédents résultats de [DL89] et [ACDT90] pour ces familles d'arbres et nous obtenons qu'une légère modification de l'algorithme de Herbrand-Robinson a un coût en moyenne constant sur les paires d'arbres quelconques. Ensuite nous montrons que le coût en moyenne de l'algorithme de Martelli Montanari est linéaire. Et ce résultat est encore valide lorsque les décompositions sont réalisées en même temps que les compactifications et lorsque l'initialisation des compteurs est menée de pair avec la détermination de la première frontière (à la place d'une phase de prétraitement). La complexité en moyenne de l'algorithme de Paterson Wegman sur des paires d'arbres quelconques reste un problème ouvert.

Sur les paires d'arbres unifiables, la complexité en moyenne de l'algorithme de Robinson a été étudiée dans [CDS89] dans le cas d'arbres binaires construits à partir d'un symbole interne et de *deux variables*. Dans ce cas très particulier, les substitutions possibles sont triviales : $\sigma : x \longleftarrow T(y)$, $\sigma : y \longleftarrow T(x)$ ou $\sigma : x \longleftrightarrow y$ et *il n'y a pas de composition des substitutions*. Ceci est le phénomène fondamental qui, dans le cas général, rompt l'uniformité de distribution sur les arbres et empêche de dénombrer exactement les paires d'arbres unifiables. Avec seulement deux variables, on peut compter directement les paires d'arbres unifiables qui sont de deux types: même structure binaire et égalité à $x = y$ près ou bien une partie commune avec aux feuilles des deux arbres, en correspondance, pour chaque occurrence de la variable x (resp. y) un sous-arbre $T(y)$ (resp. $T(x)$) au même emplacement dans l'autre arbre. Ces deux types d'arbres partitionnent en quantité égale la famille des paires d'arbres unifiables. On obtient alors pour l'algorithme de Robinson une complexité moyenne linéaire sur les paires d'arbres unifiables (étant quadratique dans le cas le pire). Sur les paires d'arbres unifiables, dans un cadre général, nous pouvons simplement affirmer que la complexité moyenne est *au moins linéaire*.

Enfin, dans la partie V, nous étudions la probabilité d'occurrence dans un modèle où le nombre de variables n'est pas fini, ce qui se rapproche plus de la réalité pour beaucoup d'applications. Ainsi, nous considérons des arbres binaires formés à partir d'un symbole fonctionnel et d'un ensemble dénombrable de variables. Le nombre de tels termes, *au renommage des variables près*, est $C_n B_{n+1}$ avec C_n le $n^{ième}$ nombre de Catalan et B_{n+1} le $n+1^{ième}$ nombre de Bell. Nous montrons que le

nombre moyen de variables distinctes dans un tel termes de taille n est $n/\ln(n)(1+O(1))$ (ce qui tend vers l'infini avec n et justifie donc l'étude de cette partie où le nombre de variables est non borné). Nous obtenons alors que, pour un nombre de variables constant comme dans la partie III ou pour un nombre de variables de l'ordre du nombre moyen de variables distinctes dans un terme au renommage près, la probabilité d'occurrence d'une variable dans un terme tend vers 1 quand la taille du terme tend vers l'infini. Elle tend vers 0 dès que le nombre de variables disponibles est superlinéaire ($n^{1+\alpha}$ avec $\alpha > 0$).

II. Présentation des algorithmes de base et modèle d'étude

2.1. Présentation des algorithmes d'unification

2.1.1 L'algorithme de Herbrand-Robinson

Dans cette partie, nous présentons les algorithmes d'unifications de base dans un cadre général. Soit F un ensemble fini de symboles fonctionnels d'arité fixée α (les constantes sont les symboles d'arité 0). Soit V un ensemble *infini dénombrable* de variables. Nous notons $T(F,V)$ l'ensemble des *termes du premier ordre*, c'est à dire la F-algèbre libre sur V, définie par :

i) $V \subset T(F,V)$

ii) $M_1,...,M_n \in T(F,V) \implies f(M_1,...,M_n) \in T(F,V)$
si $f \in F,\ \alpha(f) = n$

Les substitutions, notées par σ, ρ, θ,..., sont les F-endomorphismes de $T(F,V)$ à domaine fini sur V. On les représente par un ensemble fini de substitutions élémentaires (qui correspondent à une suite d'opérations à effectuer en parallèle). Par exemple, avec $\sigma = \{x \leftarrow a, y \leftarrow g(b,b)\}$, et $M = g(x,f(y))$, on a $M\sigma = g(a,f(g(b,b)))$. La composition des substitutions est la composition usuelle des applications :

$$\sigma o\rho = \{x_1 \leftarrow x_1\sigma\rho,...,x_n \leftarrow x_n\sigma\rho,\ y_1 \leftarrow y_1\rho,...,y_m \leftarrow y_m\rho\}$$

où $D(\sigma) = \{x_1,...,x_n\}$ et $D(\rho) - D(\sigma) = \{y_1,...,y_m\}$ avec D le domaine d'une substitution. Par la suite, nous nous intéresserons aux *substitutions idempotentes* pour lesquelles on pourra donner une *représentation séquentielle plus compacte.* Une substitution σ est idempotente ($\sigma o\sigma = \sigma$) si et seulement si $D(\sigma) \cap I(\sigma) = \emptyset$, avec $I(\sigma)$ l'ensemble des variables introduites par σ.

Les substitutions induisent un *préordre de filtrage* sur $T(F,V)$. On a :

$$M \leq N \ ssi\ \exists\sigma\ M\sigma = N$$

Dans ce cas, nous disons que M est plus général que N, ou que N est une instance de M. Par exemple, $g(x,f(y)) \leq g(g(a,a),f(b))$. L'équivalence engendrée par ce préordre $\leq$ est *l'equivalence de renommage* des variables sur $T(F,V)$:

$$M \equiv N \ ssi\ M \leq N\ et N \leq M$$

Par exemple, $g(x,f(y)) \equiv g(u,f(v)) \equiv g(y,f(x))$. Ainsi, lors du dénombrement des familles de termes avec variables, nous nous intéresserons au nombre des classes de termes de taille donnée *au*

renommage près (cf. partie V). Le préordre de filtrage sur les termes induit un préordre de filtrage sur les substitutions. Nous disons qu'une substitution σ est plus générale qu'une substitution ρ, $\sigma \leq \rho$ si $\exists\theta\ \sigma o\theta = \rho$. Ceci va permettre de dégager la notion de termes unifiables et d'unificateur principal.

DEFINITION : Deux termes M et N sont *unifiables* si $\exists\sigma\ M\sigma = N\sigma$.
Nous notons $U(M,N)$ l'ensembles des unificateurs de deux termes $U(M,N) = \{\sigma \mid \sigma M = \sigma N\}$.

THÉORÈME 1 : *(Unification [Rob65]) Si deux termes M et N sont unifiables alors ils possédent un unificateur principal (en anglais most general unifier) (mgu) σ tel que :*
- *1) $\sigma \in U(M,N)$,*
- *2) $\forall\rho \in U(M,N)\ \sigma \leq \rho$. Les unificateurs principaux de deux termes sont equivalents modulo $\equiv$.*

Par exemple, considérons $M = f(x, g(y,x))$ et $N = f(h(y), g(u, h(u)))$, l'unificateur principal de M et N est $\{x \leftarrow h(y), u \leftarrow y\}$.

PREUVE : La preuve du théorème est basée sur un algorithme d'unification.
ENTREE : deux termes M et N.
SORTIE: unifiable? Si oui unificateur principal.
Nous décrivons tout d'abord *l'algorithme d'unification non-déterministe de Herbrand* [Her30], qui est à la base un algorithme de simplification d'un système d'équations Γ, *i.e.* un multi-ensemble de paires de termes, à l'aide des trois règles suivantes :

Dec $(f(T_1,...,T_n), f(T'_1,...,T'_n)) \cup \Gamma \Longrightarrow (T_1,T'_1) \cup ... \cup (T_n,T'_n) \cup \Gamma$

Triv $(v,v) \cup \Gamma \Longrightarrow \Gamma$

Var $(v,T) \cup \Gamma \Longrightarrow (v,T) \cup \Gamma\sigma$ qui s'applique si $v \in V(\Gamma)$, $v \notin V(T)$, et à ce moment produit pour σ : $\{v \leftarrow T\}$
(la même règle s'applique à $(T,v) \cup \Gamma$).

L'objet du test $v \in V(\Gamma)$ dans la troisième règle est d'appliquer cette règle uniquement aux équations qui ne sont pas en forme résolue. D'autre part, le test $v \notin V(T)$ est une opération fondamentale appelée *test d'occurrence*. Il élimine les équations de la forme $v = T[v]$ qui n'ont pas de solutions sur les termes finis. Une paire (v,T) est en *forme résolue* dans Γ si v n'a pas d'autres occurrences dans Γ. Un système Γ est en forme résolue si toutes ses équations sont en forme résolue. Enfin, une variable est en forme résolue dans Γ si elle a une unique occurrence dans Γ dans une paire en forme résolue.

L'intérêt de cet ensemble de règles est qu'il est correct et complet quelque soit l'ordre d'application des règles. Ce fait va autoriser différentes stratégies de résolution et donc différents algorithmes.

THÉORÈME 2 : **(Correction)** *Si $\Gamma \Longrightarrow^* \Gamma' = \{(v_1,T_1),...,(v_n,T_n)\}$ est en forme résolue alors $\sigma_{\Gamma'} = \{v_1 \leftarrow T_1, ..., v_n \leftarrow T_n\}$ est un unificateur principal idempotent de Γ.*

PREUVE : Par récurrence sur le nombre d'étapes de transformation (simplification) et en remarquant que si $\Gamma \Longrightarrow \Gamma'$ alors $U(\Gamma) = U(\Gamma')$. ∎

THÉORÈME 3 : **(Terminaison et Complétude)** *Toute suite de transformations termine.*

$$\Gamma = \Gamma_0 \Longrightarrow \Gamma_1 \Longrightarrow ... \Longrightarrow \Gamma_n$$

De plus, si Γ *est unifiable, et si* $\theta \in U(\Gamma)$ *alors* $\Gamma \Longrightarrow^* \Gamma'$ *avec* Γ' *en forme résolue et* $\sigma_{\Gamma'} \leq \theta$.

PREUVE : La preuve de terminaison de toute suite de transformations est basée sur une mesure de complexité, $c(\Gamma) = (n_v, s)$, où n_v est le nombre de variables qui ne sont pas en forme résolue et s est la somme des tailles des termes dans Γ. En considérant l'ordre lexicographique sur (n_v, s), il est facile de vérifier que chaque transformation fait décroître strictement la complexité du système.

La complétude est une conséquence directe du fait que les transformations sont correctes et que si Γ est unifiable et qu'aucune règle ne s'applique à Γ' alors Γ' est en forme résolue. ∎

L'algorithme de Robinson est obtenu à partir de l'algorithme non déterministe de Herbrand en gérant le système Γ comme une pile. On transforme Γ avec la règle qui s'applique à l'équation en sommet de pile (en ignorant les équations en forme résolue). Si aucune règle ne s'applique l'algorithme s'arrête en échec dû soit à une *occurrence* soit à un *clash* si la règle **Dec** ne s'applique pas. De cette manière, on effectue un parcours en profondeur gauche (préordre) des termes.

L'algorithme de Robinson a une complexité exponentielle pour l'exemple bien connu suivant : $M = f(v_0, v_1, ..., v_{n-1}, v_0)$, $N = f(g(v_1, v_1), g(v_2, v_2), ..., g(v_n, v_n), v_0)$.
Chaque variable v_i, $i < n$ est substituée par un terme de taille $2^{n-i+1} - 1$, et l'unification du dernier argument v_0 nécessite $2^{n+1} - 1$ comparaisons. D'autre part, si l'ensemble des variables V est de cardinal v *fini*, la complexité dans le cas le pire de l'algorithme de Robinson est polynômiale d'ordre v. Sur M et $N = f(g(v_1, ..., v_1), ..., g(v_{v-1}, ..., v_{v-1}), v_0)$ l'algorithme de Robinson nécessite $O(n^v)$ comparaisons.

DEFINITION : Nous disons qu'une paire de termes est *égalisable* si elle est réductible par les règles **Dec** et **Triv** en un système de paires qui est soit en forme résolue soit réductible par la règle **Var**.

Ainsi, les paires d'arbres non-égalisables échouent à l'unification avant l'application par la règle **Var** d'une quelconque substitution, c'est à dire soit par un clash dans les premières décompositions, soit par une *occurrence directe i.e.* l'occurrence d'une feuille variable dans le sous-arbre correspondant.

En partie III, nous montrerons que la famille des paires de termes unifiables est exponentiellement négligeable devant la famille des paires d'arbres quelconques. Pour cette raison, nous nous intéressons à une variante de l'algorithme de Herbrand-Robinson où l'application des substitutions dans Γ (avec la règle **Var**) est retardée jusqu'à ce que les termes initiaux aient été décomposés et que l'on ait vérifié les échecs directs par clash ou occurrence. Nous appellerons cette variante *l'algorithme de Robinson avec retard* (RR).
Sur les paires d'arbres non-égalisables, nous avons les quatre causes d'échecs suivantes :

- clash à la racine,
- occurrence directe d'une variable,
- décomposition à la racine et échec direct dans les arguments de gauche
- ou échec direct dans les arguments de droite si les arguments de gauche sont égalisables.

Ceci nous permet de détailler l'expression du coût de l'algorithme :
$u_{RR}(c, c') = u_{RR}(c, T) = u_{RR}(T, c) = u_{RR}(f(T_1, T_2), g(T'_1, T'_2)) = 1$ (clash)
$u_{RR}(v, T) = u_{RR}(T, v) = occ(T)$ (coût du test d'occurrence)
$u_{RR}(f(T_1, T_2), f(T'_1, T'_2)) = 1 + u_{RR}(T_1, T'_1)$
si (T_1, T'_1) sont non-égalisables;
$u_{RR}(f(T_1, T_2), f(T'_1, T'_2)) = 1 + 2(|T_1| + |T'_1| + 1) + u_{RR}(T_2, T'_2)$,
si (T_1, T'_1) sont égalisables;

avec v (resp. c, T) une variable (resp. une constante, un terme composé), et en notant $|T|$ le nombre de nœuds internes de T (le nombre total de nœuds de l'arbre binaire T, internes et externes, étant donc $2|T|+1$). Dans la partie IV, nous obtiendrons à partir de ces formules, que le coût moyen de l'algorithme de Robinson avec retard sur les substitutions est constant c_{RR} sur la famille des paires d'arbres quelconques.

2.1.2 Les algorithmes d'unification linéaires

Les algorithmes d'unification linéaires sont basés sur une représentation compacte des substitutions, *i.e.* la forme triangulaire :

PROPOSITION : *Toute substitution idempotente peut être représentée par une substitution séquentielle sous la forme "triangulaire" :* $[x_1 \leftarrow T_1, ..., x_n \leftarrow T_n]$ *où* $\forall i,\ 1 \leq i \leq n\ \forall j \leq i\ x_j \notin V(T_i)$.

Ainsi, l'unificateur principal qui était de taille exponentielle pour l'exemple "pathologique" précédent, peut être représenté sous la forme triangulaire linéaire suivante :
$[v_o \leftarrow g(v_1, v_1), ..., v_{n-1} \leftarrow g(v_n, v_n)]$.
Les formes triangulaires correspondent en fait à une représentation des substitutions par des indirections sur les feuilles variables d'un DAG constituant les termes à unifier. On peut également partager les sous-arbres à partir des indirections sur les variables après substitution. Finalement, afin d'obtenir un algorithme d'unification linéaire, nous pouvons reposer le problème de l'unification de la façon suivante :
ENTREE : deux termes M et N (ou un DAG avec deux nœuds particuliers M et N)
SORTIE : unifiable ? Si oui unificateur principal sous forme triangulaire (ou une représentation du DAG après l'unification)

L'idée des algorithmes de Martelli Montanari et de Paterson Wegman est d'appliquer la règle **Var** uniquement sur une variable qui n'a pas d'autre occurrence dans le système. Une telle variable existe nécessairement si le système est unifiable. En procédant ainsi, on intègre le test d'occurrence et on élimine les substitutions. On conserve les reg s **Dec** et **Triv** tandis que l'on remplace la règle **Var** par les deux règles :

Merge $(v, v') \cup \Gamma \Longrightarrow (v, v') \cup \Gamma\sigma$
si $v \in V(\Gamma)$, $v' \in V(\Gamma)$, $v \neq v'$, $\sigma = \{v \leftarrow v'\}$,

Select $\{(v, T_1), ..., (v, T_n)\} \cup \Gamma \cup R \Longrightarrow \{(v, C), (T_1, T_2), ..., (T_1, T_n)\} \cup \Gamma \cup R$ (avec R l'ensemble des paires qui sont déjà en forme résolue)
si $n \geq 2$, $v \notin V(\Gamma)$, et si pour $1 \leq i \leq n$, $v \notin V(T_i)$, $T_i \notin V$,
où C désigne la partie commune des termes $T_1, ..., T_n$
(la même reg s'applique aux (T_i, v)).

Nous insistons sur le fait qu'il est nécessaire, pour obtenir une forme résolue de taille linéaire, de conserver pour v dans la règle **Select** la partie commune des T_i et non l'un d'entre eux, fut il le plus petit. Par exemple, avec

$$\Gamma = \{f(x_1, f(x_2, ...f(x_n, a)...))\ ,\ f(f(...f(a, x_n)..., x_2), x_1)\}$$

l'unificateur principal en forme résolue est $[x_1 \leftarrow f(x_2, x_2), ..., x_{n-1} \leftarrow f(x_n, x_n), x_n \leftarrow f(a, a)]$, alors que si l'on ne conserve pas les parties communes pour les x_i, on obtient une forme résolue de taille quadratique. On détermine naturellement la partie commune des T_i dans la règle **Select** conjointement avec les décompositions des T_i par la règle **Dec**.

Le contrôle est ((**Dec** ∪ **Triv** ∪ **Merge**)* **Select**)*. C'est à dire que l'on décompose complétement les équations (phase appelée *détermination de la première frontière*) avant de sélectionner un ensemble de paires avec une variable commune, appelée *multiéquation.* Nous appelons cet algorithme non-déterministe *l'algorithme de Herbrand avec oracle HO*. L'oracle détermine lorsque la règle **Select** peut être appliquée, c'est à dire quand une variable v n'a pas d'autre occurrence dans Γ.

Les preuves de correction, terminaison et complétude suivent en tout point celles de l'algorithme de Herbrand ci-avant. On démontre en particulier la terminaison avec la même mesure de complexité. La propriété fondamentale de cet algorithme due à [MM82] est que si Γ est formé de paires (v, T) qui ne sont pas réductibles par la règle **Select** alors il y a un cycle dans la relation d'occurrence pour les termes de Γ, ce qui entraîne que Γ n'est pas unifiable.

Sur la famille des paires d'arbres non-égalisables, l'expression du coût de l'algorithme de Herbrand avec oracle u_{HO} est identique à celle de l'algorithme de Robinson avec retard u_{RR} sauf sur les paires (*variable*, *terme*) où le test d'occurrence n'est plus nécessaire puisque l'oracle détermine l'existence ou la non-existence d'une multi-équation à réduire. On a ainsi :
$u_{HO}(v, T) = u_{HO}(T, v) = 1$ (pas de test d'occurrence, pas de sélection)
Dans la partie IV, nous montrons que le coût moyen de l'algorithme de Herbrand avec oracle sur la famille des paires d'arbres non-égalisables est constant égal à $c_{HO} \leq c_{RR}$.

On peut déduire les algorithmes de Martelli Montanari et de Paterson Wegman à partir de l'algorithme de Herbrand avec oracle, la différence entre ces deux algorithmes provenant de la façon dont est implémenté l'oracle.

Dans l'algorithme de Martelli Montanari, on associe à chaque variable un compteur qui indique le nombre ses occurrences dans Γ. Quand le compteur atteint 0, la règle **Select** peut être appliquée. De cette façon, la complexité dans le cas le pire est en $O(n + v \log v)$ où n est la taille des termes et v est le nombre de variables *distinctes* dans les termes [MM82]. L'initialisation des compteurs nécessite un parcours complet des termes initiaux à unifier et a donc un *coût linéaire*. Ceci entraîne donc un surcoût (linéaire) sur les paires de termes non-égalisables par rapport à l'algorithme de Robinson avec retard. Afin d'avoir une complexité moyenne constante sur les paires d'arbres quelconques, on peut essayer de mener de pair la phase d'initialisation avec les premières décompositions ce qui consiste, lorsqu'on atteint une feuille variable, à poursuivre le parcours du sous-terme correspondant afin d'initialiser correctement les compteurs. Pour cette variante de l'algorithme de Martelli Montanari, l'expression du coût u_{MM} est identique à celle de l'algorithme de Robinson avec retard sauf à nouveau sur les paires (*variable*, *terme*) en raison de l'initialisation des compteurs. Ainsi, on a :
$u_{MM}(v, T) = u_{MM}(T, v) = 2|T| + 1$

Nous montrerons dans la partie IV, que malgré cette amélioration le coût moyen sur la famille des paires d'arbres non-égalisables est encore *linéaire* en fonction de la taille des termes en entrée. De même, le mélange des décompositions (**Dec**) avec les compactifications (**Select**), comme Martelli et Montanari le suggèrent dans [MM82] afin d'améliorer l'efficacité sur des données non unifiables, ne modifie pas la complexité moyenne théorique. La seule façon d'obtenir une complexité moyenne constante sur des paires d'arbres quelconques est d'ajouter un test d'occurrence pour les variables à l'initialisation des compteurs après la détermination de la première frontière.

Dans l'algorithme de Paterson Wegman, l'oracle est implémenté par un parcours "bottom-up" judicieux du DAG. Ceci nécessite un prétraitement assez coûteux (gestion de pointeurs arrières ...) mais permet une complexité dans le cas le pire en $O(n)$. Le problème de déterminer si la complexité moyenne de cet algorithme sur des paires d'arbres quelconques est constant ou non reste ouvert.

2.2. Le modèle des termes arborescents

Nous considérons des arbres binaires $\mathcal{B}$ construits à partir de l symboles binaires, v variables et c constantes.

Exemple : Avec f et g des symboles fonctionnels, a une constante, x et y des variables, on peut construire le terme $T = f(x\ ,\ g(a\ ,\ y))$

On définit la taille d'un terme comme le nombre de nœuds internes de sa représentation arborescente.

Exemple : On a ici $|T| = 2$.

Pour faciliter la compréhension, nous noterons les familles de paires d'arbres par *une double lettre majuscule*. Ainsi $\mathcal{BB}$ désignera la famille des paires d'arbres quelconques de $\mathcal{B}$.

III. Les familles caractéristiques de paires d'arbres

3.1. Définitions

Notre but est de compter le nombre de couples d'arbres unifiables puis de dénombrer les différentes causes d'échec à l'unification.

Pour dénombrer la famille des couples d'arbres unifiables $\mathcal{UU}$, on va *encadrer* celle-ci par une famille de couples d'arbres trivialement unifiables, la famille $\mathcal{VB}$ définie récursivement par :

$$\begin{aligned}\mathcal{VB} &= (v,v') + (c,c) + (c,v) + (v,c) \\ &\quad + (v,T) + (T,v) \qquad \text{avec } v \notin T \text{ et } |T| > 1\end{aligned}$$

comme borne inférieure et avec la famille des paires d'arbres *égalisables* $\mathcal{EE}$ qui contient la famille des paires d'arbres unifiables. Dans $\mathcal{EE}$ il n'y a ni clash ni occurrence directe. Les causes d'échec ne peuvent provenir que d'occurrences ou de clashs indirects dus aux substitutions. Cette famille est définie comme suit :

$$\begin{aligned}\mathcal{EE} &= (v,v') + (c,c) + (c,v) + (v,c) \\ &\quad + (v,T) + (T,v) \qquad \text{avec } v \notin T \text{ et } |T| > 1 \\ &\quad + (f(T1,T2), f(T'1,T'2)) \qquad \text{avec } (T1,T'1) \text{ et } (T2,T'2) \in \mathcal{EE}\end{aligned}$$

Exemple : Ainsi $f\ (\ f(x,y)\ ,\ z\ )$ et $f\ (\ f(f(y,z),f(z,y))\ ,\ f(w,w)\ )$ forment une paire d'arbres non égalisable car en occurrence directe $(y \leftarrow f(z,y))$;
alors que $f\ (\ f(x,y)\ ,\ z\ )$ et $f\ (\ f(f(y,z),f(z,x))\ ,\ f(w,w)\ )\ \}$ forment une paire d'arbres égalisable mais non unifiable en raison d'une occurrence indirecte (substitution $\sigma\ : x \leftarrow f(y,z)$).

On a bien l'encadrement :

$$\mathcal{VB} \subset \mathcal{UU} \subset \mathcal{EE}$$

Du point de vue dénombrement, on verra que les familles $\mathcal{EE}$ et $\mathcal{VB}$ ont une cardinalité de même ordre asymptotique. On dispose donc de l'ordre asymptotique exact de $\mathcal{UU}$. De plus on montrera que les constantes d'encadrement sont très voisines (la limite de leur rapport tend vers 1), on dispose donc ainsi d'un encadrement numérique très correct du nombre de paires d'arbres unifiables.

A présent, nous pouvons nous intéresser aux causes d'échec à l'unification. Nous pouvons définir l'ensemble complémentaire $\mathcal{FF}$ des couples d'arbres non égalisables :

$$\mathcal{FF} = (c,c') + (v,T) + (T,v) \qquad \text{avec } v \in T \text{ et } |T| \geq 1$$

$$
\begin{array}{lll}
+ (c,T) + (T,c) & & \text{avec } T \in \mathcal{B} \text{ et } |T| \geq 1 \\
+ (f(T1,T2), g(T'1,T'2)) & & \text{avec } T1,T2,T'1,T'2 \in \mathcal{B} \\
+ (f(T1,T2), f(T'1,T'2)) & & \text{avec } (T1,T'1) \in \mathcal{FF} \text{ et } (T2,T'2) \in \mathcal{BB} \\
+ (f(T1,T2), f(T'1,T'2)) & & \text{avec } (T1,T'1) \in \mathcal{EE} \text{ et } (T2,T'2) \in \mathcal{FF}
\end{array}
$$

Dans $\mathcal{FF}$ se produisent des clashs et des occurrences directes. Afin de préciser les causes d'échecs à l'unification, on va distinguer les couples d'arbres non égalisables à cause d'une *occurrence* directe à gauche $\mathcal{FO}$ ou a cause d'un *clash* direct à gauche $\mathcal{FC}$. "A gauche" signifiant que la première cause d'échec détectée par un algorithme utilisant un parcours d'arbre gauche (profondeur d'abord) est soit un clash soit une occurrence (notons qu'une paire d'arbres peut échouer à l'unification pour les deux raisons, la détection de la première cause d'échec dépend de l'algorithme utilisé).

On définit ainsi :

$$
\begin{array}{lll}
\mathcal{FO} = (v,T) + (T,v) & & \text{avec } v \in T \text{ et } |T| \geq 1 \\
\quad + (f(T1,T2), f(T'1,T'2)) & & \text{avec } (T1,T'1) \in \mathcal{FO} \text{ et } (T2,T'2) \in \mathcal{BB} \\
\quad + (f(T1,T2), f(T'1,T'2)) & & \text{avec } (T1,T'1) \in \mathcal{EE} \text{ et } (T2,T'2) \in \mathcal{FO} \\
\mathcal{FC} = (c,c') + (c,T) + (T,c) & & \text{avec } T \in \mathcal{B} \text{ et } |T| \geq 1 \\
\quad + (f(T1,T2), g(T'1,T'2)) & & \text{avec } T1,T2,T'1,T'2 \in \mathcal{B} \\
\quad + (f(T1,T2), f(T'1,T'2)) & & \text{avec } (T1,T'1) \in \mathcal{FC} \text{ et } (T2,T'2) \in \mathcal{BB} \\
\quad + (f(T1,T2), f(T'1,T'2)) & & \text{avec } (T1,T'1) \in \mathcal{EE} \text{ et } (T2,T'2) \in \mathcal{FC}
\end{array}
$$

On a bien :

$$\mathcal{FF} = \mathcal{FO} + \mathcal{FC} = \mathcal{BB}/\mathcal{EE}$$

Dans $\mathcal{FO}$ ne se produisent pas de clashs directs à gauche et dans $\mathcal{FC}$ ne se produisent pas d'occurrences directes à gauche mais (puisque dans $\mathcal{EE}$ et dans $\mathcal{BB}$ peuvent se produire des échecs indirects), il peut y avoir des occurrences ou des clashs indirects. On va donc pour affiner cette étude introduire la famille des paires d'arbres qui *n'échouent pas par clash direct* : $\mathcal{NC}$. Cette famille reste de même importance que $\mathcal{BB}$ et son complémentaire dans $\mathcal{BB}$ fournira une borne supérieure de l'ensemble des arbres qui échouent par clash. Cette famille peut être définie comme suit :

$$
\begin{array}{lll}
\mathcal{NC} = (v,v') + (c,c) + (c,v) + (v,c) & & \\
\quad + (v,T) + (T,v) & & \text{avec } |T| \geq 1 \\
\quad + (f(T1,T2), f(T'1,T'2)) & & \text{avec } (T1,T'1) \text{ et } (T2,T'2) \in \mathcal{NC}
\end{array}
$$

Remarque : Dans $\mathcal{BB} \setminus \mathcal{NC}$ on compte en plus par rapport à $\mathcal{FC}$ les clashs directs détectés par un parcours gauche après une occurrence directe.

Remarque : $\mathcal{NC}$ ne contient pas $\mathcal{FO}$ car dans $\mathcal{FO}$ il peut y avoir des clashs directs (encore une fois détectés par un parcours gauche après une occurrence directe).

Remarque : La famille $\mathcal{NO}$ des paires d'arbres qui n'échouent pas par occurrence directe, étant négligeable devant $\mathcal{BB}$, ne présente pas d'intérêt du point de vue dénombrement (son complémentaire fournit pour $\mathcal{FO}$ la borne supérieure triviale $\mathcal{BB}$).

3.2. Analyse algébrique

Pour chacune des familles précédentes, nous allons déterminer le nombre x_n d'arbres T ou de paires d'arbres (T_1,T_2) de taille n (on pose $|(T_1,T_2)| = |T_1| + |T_2|$). Classiquement, la série génératrice correspondant à la famille $\mathcal{X}$ est donnée par :

$$X(z) = \sum_{n \geq 0} x_n z^n = \sum_{T \in \mathcal{X}} z^{|T|}$$

(Cf. [FV87] pour la théorie et l'utilisation des séries génératrices).
On notera en romain la série génératrice correspondant à la famille notée par une lettre italique.

Ainsi à partir des définitions des familles précédentes, on obtient les équations fonctionnelles et les séries génératrices suivantes : $B(z) = (v+c) + lzB(z)^2$, soit :

$$B(z) = -\frac{\sqrt{1-4lz(v+c)}\ -1}{2lz}$$

La série génératrice des couples d'arbres quelconques est :

$$BB(z) = B(z)^2 = \frac{(\sqrt{1-4lz(v+c)}\ -1)^2}{4l^2z^2}$$

On introduit quelques séries utiles pour les dénombrements qui vont suivre, ainsi la série génératrice des arbres de taille au moins 1 : $B^+(z) = lzB(z)^2$. La série génératrice des arbres ne contenant pas une variable donnée : $B_{v-1}(z) = v - 1 + c + lzB_{v-1}(z)^2$, soit :

$$B_{v-1}(z) = -\frac{\sqrt{1-4lz(v+c-1)}\ -1}{2lz}$$

et les arbres de taille au moins 1 sans une variable donnée : $B^+_{v-1}(z) = lzB_{v-1}(z)^2$.
On définit aussi les arbres contenant une variable donnée :

$$C(z) = 1 + lzC(z)B(z) + lzB_{v-1}(z)C(z)$$

soit

$$C(z) = (\frac{\sqrt{1-4lz(v+c)}}{2} + \frac{\sqrt{1-4lz(v+c-1)}}{2})^{-1}$$

et on a donc les arbres avec une variable donnée de taille au moins 1 :

$$C^+(z) = lzC(z)B(z) + lzB_{v-1}(z)C(z)$$

soit

$$C^+(z) = -\frac{\sqrt{1-4lz(v+c)}\ -1}{\sqrt{1-4lz(v+c)}\ +\sqrt{1-4lz(v+c-1)}} - \frac{\sqrt{1-4lz(v+c-1)}\ -1}{\sqrt{1-4lz(v+c)}+\sqrt{1-4lz(v+c-1)}}$$

La série génératrice des paires d'arbres trivialement unifiables vérifie :

$$VB(z) = v^2 + c + 2vc + 2vB^+_{v-1}(z)$$

On peut enfin obtenir la série génératrice des couples d'arbres égalisables $EE(z)$ c'est à dire :

$$EE(z) = v^2 + c + 2vc + 2vB^+_{v-1}(z) + lz^2EE(z)^2$$

soit

$$EE(z) = -\frac{\sqrt{1+4v(v-2)lz^2-4zv-4lcz^2+4zv\sqrt{1-4lz(v+c-1)}}-1}{2lz^2}$$

On dispose aussi des équations donnant les séries génératrices $FF(z)$, $FO(z)$, $FC(z)$ et $NC(z)$ (les expressions explicites sont trop longues pour être écrites) :

$$FF(z) = (c^2-c) + 2vC^+(z) + 2cB^+(z) + (l^2-l)z^2B(z)^4 + lz^2FF(z)(EE(z)+BB(z))$$
$$FO(z) = 2vC^+(z) + lz^2FO(z)(BB(z)+EE(z))$$
$$FC(z) = (c^2-c) + 2cB^+(z) + (l^2-l)z^2B(z)^4 + lz^2FC(z)(BB(z)+EE(z))$$
$$NC(z) = (v+c)^2 + 2vB^+(z) + lz^2NC(z)^2$$

On vérifie aisément que

$$BB(z) = EE(z) + FF(z) \qquad \text{et} \qquad FF(z) = FO(z) + FC(z)$$

3.3. Analyse asymptotique

On va utiliser l'analyse de singularité. Nous connaissons le fait que

$$[z^n](1-z/\rho)^\alpha = \frac{n^{-\alpha-1}}{\rho^n\Gamma(-\alpha)}\left(1+O\left(\frac{1}{n}\right)\right)$$

et les lemmes de transfert (Cf. [FV87]).

La singularité de plus petit module des séries génératrices $BB(z)$, $FF(z)$, $FO(z)$, $FC(z)$ est $z=\frac{1}{4l(v+c)}$. Par contre la singularité de plus petit module des séries génératrices $EE(z)$, $VB(z)$ (et donc de $UU(z)$) est $z=\frac{1}{4l(v+c-1)}$.

Remarque : Précisons que c'est effectivement le cas pour $EE(z)$ dès que ($c\geq 0, l\geq 3$) ou ($c\geq 2$, $l\geq 1$) ou ($c\geq 1, l\geq 2$) ou ($v\geq 3, c\geq 0, l\geq 2$) soit pour toutes les valeurs (ou presque) de (v,c,l); pour les valeurs de (v,c,l) inférieures qui gardent un sens (on sait que $v\geq 2$) c'est à dire $v=2$, $c=0$, $l=2$ ou $0\leq c\leq 1$ et $l=1$ on a :

- soit la plus petite singularité ρ de $\mathcal{EE}$ comprise entre $\frac{1}{4l(v+c)}$ et $\frac{1}{4l(v+c-1)}$ ce qui entraîne que $\mathcal{EE}$ reste exponentiellement négligeable devant $\mathcal{BB}$ mais avec un ordre asymptotique supérieur à celui de $\mathcal{VB}$,
- soit on modifie légèrement la famille $\mathcal{EE}$ pour conserver l'encadrement adéquat ce qui a pour effet de modifier la constante de l'encadrement.

On a :

$$BB(z) = -8(v+c)^2\sqrt{1-4lz(v+c)}\ +O(z-\frac{1}{4l(v+c)}\ )$$

$$FF(z) = -8(v+c)^2\sqrt{1-4lz(v+c)}\ +O(z-\frac{1}{4l(v+c)}\ )$$

$$FO(z) = -16\ v\ l\ (v+c)^2\ \frac{1}{f}\left(1+\frac{(2\sqrt{v+c}-1)}{f}\right)\sqrt{1-4lz(v+c)}+O(z-\frac{1}{4l(v+c)}\ )$$

$$FC(z) = \frac{-8(v+c)^2}{f}\left(2(v+c)(l-1)+2cl+\frac{(3lc^2-lc+2vcl+(v+c)^2(l-1))}{f}\right)$$
$$\times\sqrt{1-4lz(v+c)}+O(z-\frac{1}{4l(v+c)}\ )$$

avec

$$f=(v+c)(2l-1)+\sqrt{l(4l(v+c)^2+4v\sqrt{v+c}-3v^2-4vc-2v-c)}$$

$$NC(z) = \frac{-8(v+c)^2vl^{\frac{1}{2}}}{\sqrt{4l(v+c)^2-3v^2-4vc-c}}\sqrt{1-4lz(v+c)}+O(z-\frac{1}{4l(v+c)}\ )$$

D'autre part, on a :

$$EE(z) = \frac{-8\ \sqrt{l}\ v\ (v+c-1)^2}{\sqrt{4l(v+c-1)^2-3v^2+2v-4vc-c}}\sqrt{1-4lz(v+c-1)}+O(z-\frac{1}{4l(v+c-1)}\ )$$

$$VB(z) = -4(v+c-1)\ v\ \sqrt{1-4lz(v+c-1)}+O(z-\frac{1}{4l(v+c-1)}\ )$$

On en déduit :

$$[z^n]BB(z) = 4(v+c)^2(4l(v+c))^n \frac{n^{-3/2}}{\sqrt{\pi}}\left(1+O\left(\frac{1}{n}\right)\right) = [z^n]FF(z)$$

($\mathcal{EE}$ est exponentiellement négligeable devant $\mathcal{BB}$ et $\mathcal{FF}$).

$$[z^n]FO(z) = 8\ v\ l\ (v+c)^2\ \frac{1}{f}\left(1+\frac{(2\sqrt{v+c}-1)}{f}\right)(4l(v+c))^n\frac{n^{-3/2}}{\sqrt{\pi}}\left(1+O\left(\frac{1}{n}\right)\right)$$

$$[z^n]FC(z) = \frac{4(v+c)^2}{f}\left(2(v+c)(l-1)+2cl+\frac{(3lc^2-lc+2vcl+(v+c)^2(l-1))}{f}\right)$$
$$\times(4l(v+c))^n\frac{n^{-3/2}}{\sqrt{\pi}}\left(1+O\left(\frac{1}{n}\right)\right)$$

$$[z^n]NC(z) = \frac{4(v+c)^2v\sqrt{l}}{\sqrt{4l(v+c)^2-3v^2-4vc-c}}(4l(v+c))^n\frac{n^{-3/2}}{\sqrt{\pi}}\left(1+O\left(\frac{1}{n}\right)\right)$$

Et

$$[z^n]VB(z) = 2(v+c-1)v\frac{(4l(v+c-1))^n}{\sqrt{\pi}}n^{-3/2}\left(1+O\left(\frac{1}{n}\right)\right)$$
$$= C_1\frac{(4l(v+c-1))^n}{\sqrt{\pi}}n^{-3/2}\left(1+O\left(\frac{1}{n}\right)\right)$$

$$[z^n]EE(z) = \frac{4\sqrt{l}\ (v+c-1)^2v}{\sqrt{4l(v+c-1)^2-3v^2+2v-4vc-c}}\frac{(4l(v+c-1))^n}{\sqrt{\pi}}n^{-3/2}\left(1+O\left(\frac{1}{n}\right)\right)$$
$$= C_2\frac{(4l(v+c-1))^n}{\sqrt{\pi}}n^{-3/2}\left(1+O\left(\frac{1}{n}\right)\right)$$

3.4. Résultats

3.4.1 Les paires d'arbres unifiables

On a l'encadrement asymptotique souhaité pour $[z^n]UU(z)$:

$$C_1\frac{(4l(v+c-1))^n}{\sqrt{\pi}}n^{-3/2}\left(1+O\left(\frac{1}{n}\right)\right) \le [z^n]UU(z) \le C_2\frac{(4l(v+c-1))^n}{\sqrt{\pi}}n^{-3/2}\left(1+O\left(\frac{1}{n}\right)\right)$$

On dispose de l'odre asymptotique correct de $[z^n]UU(z)$, et de plus, numériquement, on constate que $C_2/C_1 \approx 1$! On a $1 \le C_2/C_1 \le 1.2$ et on a même :

$$\lim_{l\to\infty}\frac{C_2}{C_1} = 1.$$

L'encadrement numérique est donc d'autant meilleur que l est grand (à partir de 3 ou 4 déjà : pour $c=2$, $v=2$, $l=3$ on a $C_2/C_1 = 1.10$, pour $c=3$, $v=3$, $l=6$ on a $C_2/C_1 = 1.05$, pour $c=10$, $v=4$, $l=10$ on a $C_2/C_1 = 1.01$).

Pour résumer, on a le théorème suivant :

THÉORÈME 4 : *En considérant une distribution uniforme sur les paires d'arbres binaires avec l symboles internes, v variables et c constantes, alors :*

- *le nombre de paires d'arbres unifiables de taille n est*

$$\Theta\left((4l(v+c-1))^n n^{\frac{3}{2}}\right)$$

- *la proportion entre le nombre de paires d'arbres unifiables et le nombre de paires d'arbres binaires quelconques est*

$$\Theta\left(\left(\frac{v+c-1}{v+c}\right)^n\right)$$

(Θ : "de l'ordre asymptotique de").

ce qui signifie que pour n assez grand presque toutes les paires d'arbres binaires sont non-unifiables et que l'on dispose d'un bon encadrement numérique du nombre de paires d'arbres unifiables.

3.4.2 Les causes d'échec

Passons à présent aux causes d'échec. On a une idée du rapport entre le nombre de couples qui échouent par clash et par occurrence directe.

En notant

$$\tau_c = \frac{|FC|}{|BB|} = \frac{|FC|}{|FF|} \quad et \quad \tau_o = \frac{|FO|}{|BB|} = \frac{|FO|}{|FF|}$$

on peut représenter les variations de ces rapports.

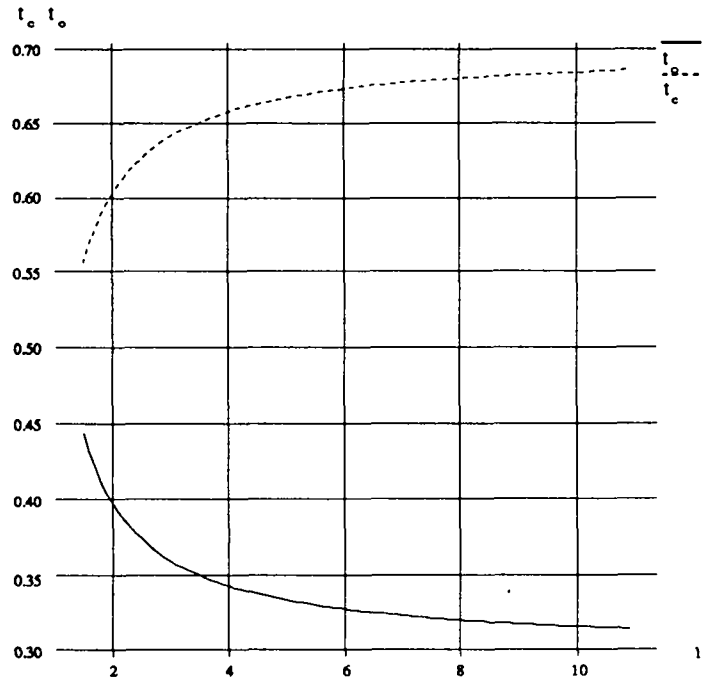

Figure 1 : Variation de τ_c et τ_o pour $c = 2$, $v = 3$, $2 \leq l \leq 10$

La figure 1 nous montre qu'à c et v fixés, la proportion de clashs directs augmente avec l et est nettement prépondérante devant la proportion d'occurrences.

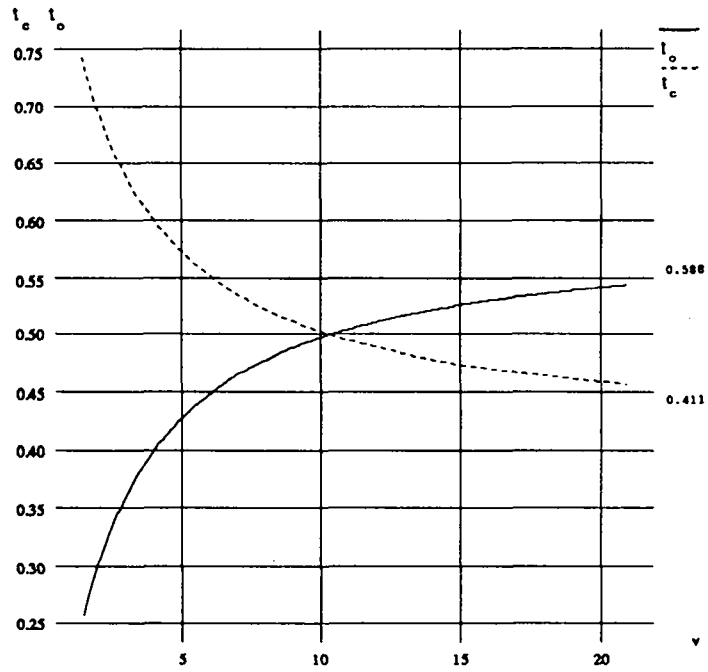

Figure 2 : Variation de τ_c et τ_o pour $c = 2$, $l = 3$, $2 \leq v \leq 20$

A l et c fixés, l'occurrence directe augmente avec v; elle devient *légèrement* supérieure aux causes d'échecs par clash lorsque v est grand devant l.

On peut directement considérer le rapport :

$$\tau_{c/o} = \frac{|FC|}{|FO|}$$

et le représenter en 3D dans les cas $c = 2$ et $c = 10$:

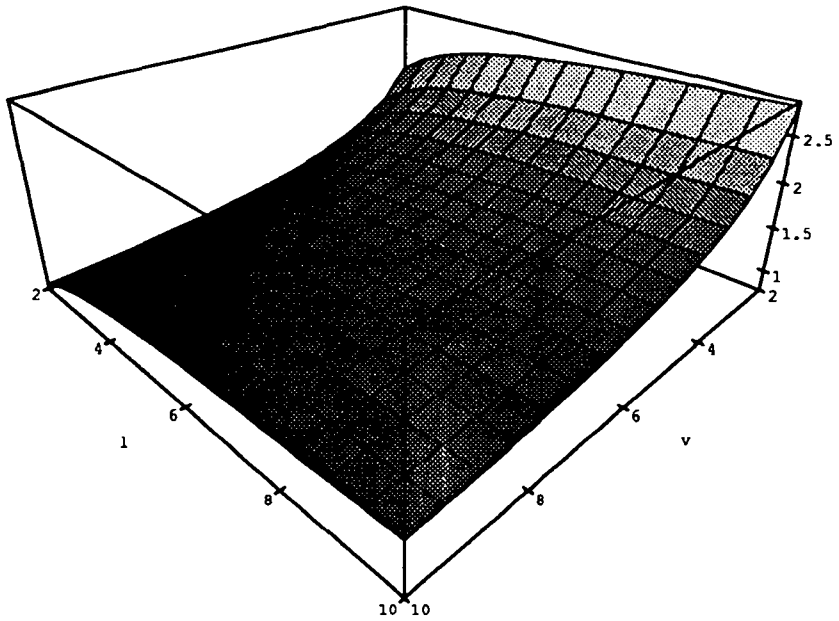

Figure 3 : Variation de $\tau_{c/o}$ pour $c = 2$, $2 \leq v \leq 10$ et $2 \leq l \leq 10$

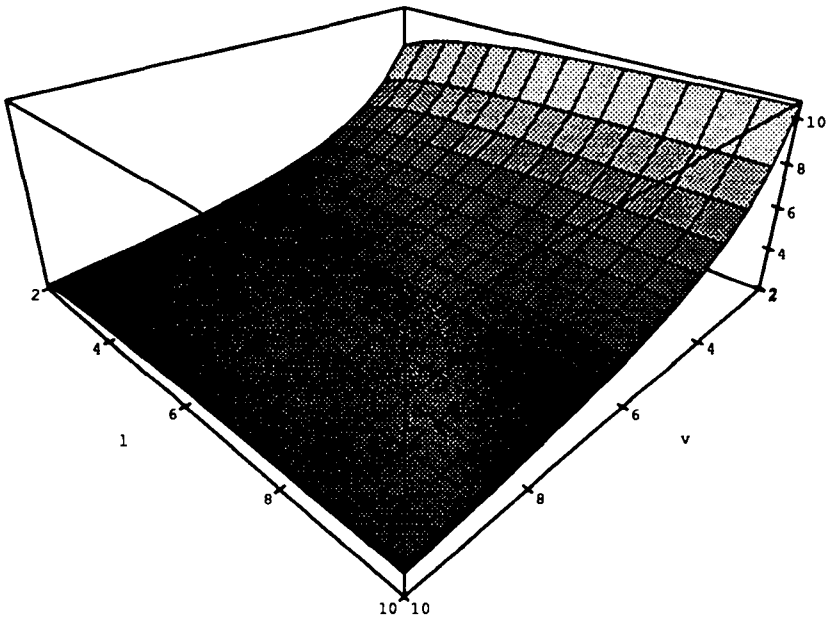

Figure 4 : Variation de $\tau_{c/o}$ pour $c = 10$, $2 \leq v \leq 10$ et $2 \leq l \leq 10$

Les figures 3 et 4 nous montrent que : numériquement, on a toujours plus de clashs que d'occurrences et ceci est d'autant plus marqué que c et l sont grands (rapport qui peut valoir 10 par exemple). On peut remarquer que quand v est petit, le nombre de clashs reste encore prépondérant alors qu'on peut penser qu'un petit nombre de variables entraîne une répétition de celles-ci aux feuilles des termes et donc accroît la probabilité d'échec par occurrence. En fait un autre phénomène l'emporte sur ce dernier : quand v est petit, en raison du modèle de distribution uniforme, les feuilles sont majoritairement occupées par des constantes d'où l'importance croissante du clash. *On remarque ainsi qu'il est fondamental de considérer des termes avec constantes.*

Enfin, afin d'étudier précisément les causes d'échec par clash, on peut également définir le rapport :

$$\tau_{c+} = 1 - \frac{|NC|}{|BB|} = 1 - \frac{v\sqrt{l}}{\sqrt{4l(v+c)^2 - 3v^2 - 4vc - c}}$$

Ceci va nous permettre de regarder un encadrement des paires d'arbres qui échouent par clash direct, en représentant sur le même schéma les proportions par rapport à $\mathcal{BB}$ des familles $\mathcal{FC}$ et $\mathcal{BB} \setminus \mathcal{NC}$.

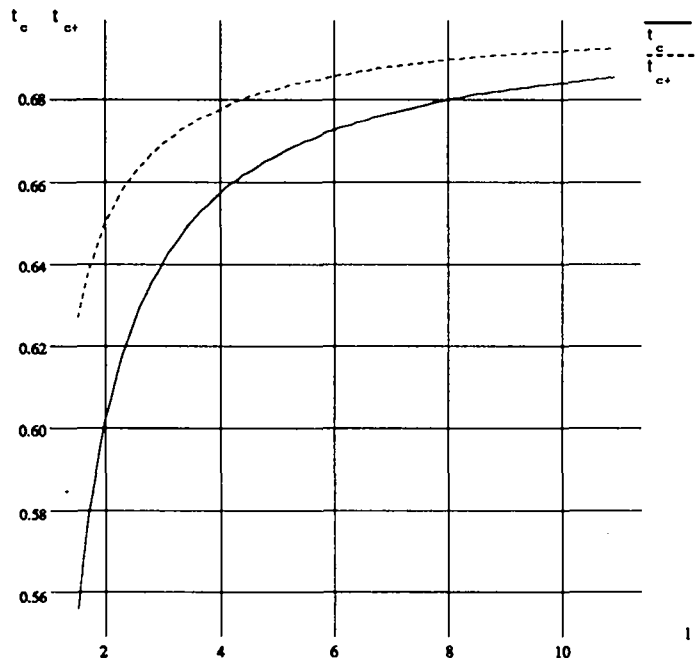

Figure 5 : Variation de τ_c et τ_{c+} pour $c = 2$, $v = 3$ et $2 \leq l \leq 10$

La proportion des clashs directs se situe entre les deux courbes de la figure 5. On constate donc qu'on dispose d'un bon encadrement de la proportion de la cause d'échec par clash, encadrement qui est d'autant meilleur que l est grand.

On peut également regarder le comportement aux limites de ces différentes quantités. On a en effet :

$$\lim_{l \to \infty} \tau_c = \frac{v + 2c}{2(v+c)} \underset{v \to \infty}{\longrightarrow} \frac{1}{2}^+$$

$$\lim_{v \to \infty} \tau_c = \frac{-5l + 2\sqrt{l^3(4l-3)} + 4l^2 + 1 - 2\sqrt{l(4l-3)}}{(2l - 1 + \sqrt{l(4l-3)})^2} \underset{l \to \infty}{\longrightarrow} \frac{1}{2}^-$$

et

$$\lim_{l \to \infty} \tau_o = \frac{v}{2(v+c)} \underset{v \to \infty}{\longrightarrow} \frac{1}{2}^-$$

$$\lim_{v \to \infty} \tau_o = \frac{2l}{2l - 1 + \sqrt{l(4l-3)}} \underset{l \to \infty}{\longrightarrow} \frac{1}{2}^+$$

et

$$\lim_{l\to\infty} \tau_{c/o} = \frac{v+2c}{v} \underset{v\to\infty}{\longrightarrow} 1^+$$

et

$$\lim_{v\to\infty} \tau_{c/o} = \frac{-5l + 2\sqrt{l^3(4l-3)} + 4l^2 + 1 - 2\sqrt{l(4l-3)}}{2l(2l-1+\sqrt{l(4l-3)})} \underset{l\to\infty}{\longrightarrow} 1^-$$

$$\lim_{l\to\infty} \tau_{c+} = \frac{v+2c}{2(v+c)} \underset{v\to\infty}{\longrightarrow} \frac{1}{2}^+$$

et

$$\lim_{v\to\infty} \tau_{c+} = \frac{\sqrt{4l-3}-\sqrt{l}}{\sqrt{4l-3}} \underset{l\to\infty}{\longrightarrow} \frac{1}{2}^-$$

Remarque : τ_c et τ_{c+} ont même valeur limite quand $l \longrightarrow \infty$, ceci montre bien la qualité de l'encadrement des clashs directs.

En conclusion de cette partie, on a le théorème :

THÉORÈME 5 : *Les familles de paires d'arbres non unifiables en raison d'un échec direct à gauche $\mathcal{FF}$, $\mathcal{FO}$ et $\mathcal{FC}$ ainsi que la famille des paires d'arbres ne produisant pas de clash direct $\mathcal{NC}$ sont de même ordre de grandeur que la famille des paires d'arbres quelconques $\mathcal{BB}$. Ce modèle, où l'ensemble des variables est de taille finie v, fournit une bonne estimation des causes d'échec par clash. Avec ce modèle, l'occurrence n'est pas une cause d'échec négligeable devant le clash. Néanmoins numériquement on constate qu'elle est d'importance moindre.*

Interprétons ce résultat combinatoire sur le phénomène d'occurrence. L'analyse asymptotique nous conduit à considérer des arbres binaires de grande taille, avec donc de nombreuses feuilles, ce qui impose des répétitions de variables identiques et donc une proportion d'occurrence non négligeable. Dans Prolog, (où il n'y a pas de test d'occurrence et où l'on unifie beaucoup de termes de taille assez petite, cf. [SS86]), il y a une génération automatique de variables (on renomme tout d'abord les deux termes à unifier pour qu'ils n'aient aucune variable commune puis on gère les égalités) et l'on peut disposer potentiellement d'une infinité de variables (*i.e.* le nombre de variables peut croître de concert avec la taille n de la paire de termes) et ce fait est non modélisable avec v fini.

Il est donc intéressant de reconsidérer le phénomène de l'occurrence (le problème du clash étant comme on l'a vu bien maîtrisé avec le modèle v fini). Dans la partie V, nous reprendrons cette étude en changeant de structure et en modélisant les termes à unifier par des motifs arborescents avec une infinité de variables (au renommage près). On regardera alors l'influence sur la probabilité d'occurrence (on démontrera d'ailleurs que le nombre moyen de variables distinctes, dans de tels motifs au renommage des variables près, tend vers l'infini avec la taille du terme, ceci étant la justification théorique de cette étude avec v infini).

IV. Coût moyen de l'unification

Dans cette partie nous allons présenter le coût en moyenne des différents algorithmes exposés dans la partie II. En section 4.1, nous précisons la notion de coût moyen d'un algorithme sur une famille de données puis en utilisant les résultats de la partie précédente nous démontrons que le coût moyen d'un algorithme d'unification sur la famille des paires d'arbres quelconques $\mathcal{BB}$ est le même que sur la famille $\mathcal{FF}$ des paires d'arbres qui échouent de façon directe. Nous démontrons

ensuite que le coût du test d'occurrence d'une variable dans un terme est en moyenne constant. En section 4.2, nous calculons le coût moyen exact de l'algorithme de Robinson avec retard soit la constante c_{RR}. En section 4.3, nous calculons le coût moyen exact de l'algorithme de Herbrand avec oracle sur la règle **Select** soit la constante c_{HO} (ce qui modélise également l'algorithme de Martelli Montanari *aprés la phase de prétraitement linéaire d'initialisation des compteurs*). En section 4.4, nous analysons la complexité moyenne de l'amélioration de l'algorithme de Martelli-Montanari où la phase d'initialisation des compteurs est menée de pair avec la phase de détermination de la première frontière (où on n'effectue pas une phase de prétraitement de coût linéaire) et nous obtenons que le coût sur des paires d'arbres quelconques reste en moyenne encore linéaire. Enfin dans la section 4.5, nous tirons quelques conclusions de tous ces résultats.

4.1. Coût moyen

4.1.1 Définition

Le coût moyen passé à unifier deux termes de taille totale n appartenant à une famille de paire de termes $\mathcal{X}\mathcal{X}$ est défini par :

$$\overline{\tau_n} = \frac{\sum_{|T_1|+|T_2|=n} \tau(T_1, T_2)}{xx_n}$$

où $\tau(T_1, T_2)$ est le coût de l'unification des deux termes T_1 et T_2 et xx_n le nombre de paires de termes de la famille $\mathcal{X}\mathcal{X}$ de taille globale n.

4.1.2 Coût sur $\mathcal{B}\mathcal{B}$ = coût sur $\mathcal{F}\mathcal{F}$

On établit le résultat suivant :

LEMME : *Le coût moyen d'un algorithme d'unification sur des paires d'arbres quelconques est le même que sur des paires d'arbres non-unifiables par échec direct.*

Ceci revient à dire que le coût moyen de l'algorithme va être celui de détecter un échec direct. En effet, ceci vient du fait que, comme nous l'avons vu, les paires d'arbres égalisables (et donc les paires d'arbres unifiables) sont exponentiellement négligeables devant les paires d'arbres quelconques. Or le coût de tout algorithme d'unification, dans un modèle où le nombre de variables est fini, est borné polynômialement dans le cas le pire ($O(n^v)$ exactement). Ceci entraîne donc que la contribution prépondérante avec un modèle de distribution uniforme sur les paires de termes va être celle des paires non-unifiables en raison d'un échec direct.

PREUVE : On peut en effet décomposer :

$$\sum_{T,T' \in \mathcal{B}\mathcal{B}} \tau(T,T') z^{|T|+|T'|} = \sum_{T,T' \in \mathcal{F}\mathcal{F}} \tau(T,T') z^{|T|+|T'|} + \sum_{T,T' \in \mathcal{E}\mathcal{E}} \tau(T,T') z^{|T|+|T'|}$$

En bornant le coût de l'unification de deux termes T et T' de $\mathcal{E}\mathcal{E}$ par $M(|T| + |T'|)^v$ on a

$$\sum_{T,T' \in \mathcal{E}\mathcal{E}\mathcal{E}} \tau(T,T') z^{|T|+|T'|} \leq M \sum_{T,T' \in \mathcal{E}\mathcal{E}\mathcal{E},\ |T|+|T'|=n} ee_n n^v z^n$$

avec ee_n le nombre de paires de termes de $\mathcal{E}\mathcal{E}$ de taille globale n.

En décomposant x^v dans la base des polynômes $\{1, x, x(x-1), \ldots, x(x-1)\ldots(x-v+1)\}$, on exprime facilement le membre droit de l'inégalité précédente sous la forme

$$M\left(\sum_{i=0}^{v} \beta_i z^i EE(z)^{[i]}\right)$$

avec β_i des constantes et $EE(z)^{[i]}$ la derivée $i^{ième}$ de $EE(z)$. Les $EE(z)^{[i]}$ ayant toute même rayon de convergence, leur plus petite singularité reste $z = 1/4l(v+c-1)$ et donc $[z^n]EE(z)^{[i]} \sim \lambda n^{-3/2}(4l(v+c-1))^n$ ce qui reste exponentiellement négligeable devant $[z^n]BB(z)$ et c'est donc la contribution des éléments de $\mathcal{FF}$ qui est prépondérante dans le coût moyen total. ∎

4.1.3 Coût moyen du test d'occurrence

Le coût moyen du test d'occurrence d'une variable x dans un terme T de $\mathcal{B}$ peut être défini récursivement de la façon suivante :

$occ(c) = occ(v) = 1$

$$occ(f(T_1, T_2)) = \begin{cases} 1 + occ(T_1), & \text{si } x \in T_1; \\ 1 + occ(T_1) + occ(T_2), & \text{sinon.} \end{cases}$$

Ainsi si l'on pose $O(z) = \sum_{T \in \mathcal{B}} occ(T) z^{|T|}$ on peut tout de suite décomposer $O(z) = O_1(z) + O_2(z)$ avec $O_1(z) = \sum_{x \in T} occ_1(T) z^{|T|}$ et $O_2(z) = \sum_{x \notin T} occ_2(T) z^{|T|}$; occ_1 et occ_2 étant definis récursivement de façon analogue à occ.

On obtient alors les séries génératrices :

$$O_1(z) = v + c - 1 + l(zB_{v-1}(z)^2 + 2zB_{v-1}(z)O_1(z))$$

$$O_2(z) = 1 + l(zC(z)B(z) + zB(z)O_2(z) + zB_{v-1}(z)C(z) + zC(z)O_1(z) + zB_{v-1}(z)O_2(z))$$

Les expressions explicites de $O_1(z)$ et $O_2(z)$ sont un peu trop longues pour être écrites, on obtient asymptotiquement :

$$O(z) = -4(v+c)^{3/2}(\sqrt{v+c}+1)\sqrt{1-4lz(v+c)} + O(z - \frac{1}{4l(v+c)}) = O_2(z)$$

D'où un coût moyen du test d'occurrence constant égal à

$$k_o = \frac{[z^n]O(z)}{[z^n]B(z)} = \frac{[z^n]O_2(z)}{[z^n]C(z)} = 2\sqrt{v+c}(\sqrt{v+c}+1)$$

Remarque : On retrouve bien $k_o = 4 + 2\sqrt{2}$ cas particulier où $l = v = c = 1$ traité dans [DL89].

4.2. Coût moyen de l'algorithme de Robinson avec retard

Nous allons déterminer le coût moyen exact de l'algorithme de Robinson avec retard c'est à dire l'algorithme d'unification où l'on retarde au maximum l'application des substitutions. Cet algorithme va donc tout d'abord détecter les causes d'échec direct. Nous savons d'après le lemme de la section 4.1.2 que le coût moyen de l'algorithme sur des paires d'arbres quelconques va coïncider avec son coût moyen sur les paires d'arbres non-unifiables en raison d'un échec direct.

Nous pouvons reprendre le détail de l'expression récursive du coût de l'algorithme $u_{RR}(T,T')$ sur une paire de termes $(T,T') \in \mathcal{FF}$:

$u_{RR}(c,c') = 1$ clash

$u_{RR}(v,T) = u_{RR}(T,v) = occ_2(T)$ coût du test d'occurrence avec $v \in T$ et $|T| \geq 1$.

$u_{RR}(c,T) = u_{RR}(T,c) = 1$ clash avec $T \in \mathcal{B}$ et $|T| \geq 1$

$u_{RR}(f(T_1,T_2),g(T_1',T_2')) = 1$ clash avec $T_1, T_2, T_1', T_2' \in \mathcal{B}$

$u_{RR}(f(T_1,T_2),f(T_1',T_2')) = 1 + u_{RR}(T_1,T_1')$ avec $(T_1,T'1) \in \mathcal{FF}$ et $(T_2,T_2') \in \mathcal{BB}$

$u_{RR}(f(T_1,T_2),f(T_1',T_2')) = 1+(2|T_1|+1+2|T_1'|+1)+u_{RR}(T_2,T_2')$ parcours de $(T_1,T'1) \in \mathcal{EE}$ et coût sur $(T_2,T_2') \in \mathcal{FF}$

Nous pouvons exprimer $u_{RR}(T,T')$ pour $|T|+|T'| = n$ comme le $n^{ième}$ coefficient de la série génératrice $U_{RR}(z)$ qui vérifie l'équation algébrique :

$$\begin{aligned} U_{RR}(z) = {} & c^2 - c + 2v(O_2(z)-1) + 2cB^+(z) + (l^2-l)z^2B^4(z) \\ & + l(z^2B^2(z)FF(z) + z^2B^2U_{RR}(z)) \\ & + l(z^2EE(z)FF(z) + 2z^2FF(z)(EE(z) + zEE'(z)) + z^2EE(z)U_{RR}(z)) \end{aligned}$$

(puisque $u_{RR}(v,T)$ est le coût du test d'occurrence sur T on a bien $\sum_{(v,T),|T|\geq 1, v\in T} u_{RR}(v,T)z^{|T|} = v\sum_{|T|\geq 1} occ_2(T)z^{|T|} = v(O_2(z)-1)$).

L'expression explicite de $U_{RR}(z)$ est encore trop longue pour être écrite.

La singularité de plus petit module de $U_{RR}(z)$ est $z = \frac{1}{4l(v+c)}$. Nous pouvons alors effectuer l'analyse asymptotique qui fournit :

$$U_{RR}(z) = K_1\sqrt{1-4lz(v+c)} + O(z-\frac{1}{4l(v+c)})$$

d'où

$$[z^n]U_{RR}(z) = K_2(4l(v+c))^n n^{-3/2}\left(1+O\left(\frac{1}{n}\right)\right)$$

avec K_1 et K_2 des constantes dependants de l, v et c.

Le coefficent $[z^n]U_{RR}(z)$ est donc de même ordre que $[z^n]FF(z) = [z^n]BB(z)$, on a le résultat :

THÉORÈME 6 : *Le coût moyen de l'algorithme de Robinson avec retard sur des paires d'arbres quelconques est constant égal à c_{RR}.*

Nous pouvons donner l'expression la plus "compacte" de la constante c_{RR} obtenue avec le système de calcul symbolique MAPLE ([CGC+88]) :

```
2*l^(5/2)*(-8*Y^(1/2)*l*X^(1/2)*v^2+16*Y^(1/2)*l^2*X^(1/2)*v^3+16*Y^(1
/2)*l^2*X^(1/2)*c^2*v-4*Y^(1/2)*X^(1/2)*c*v^2+32*Y^(1/2)*l^2*X^(1/2)*c
*v^2-2*c*l*X^(1/2)*v*Y^(1/2)+32*Y^(1/2)*l^(5/2)*c^3*v+96*Y^(1/2)*l^(5/2
)*c^2*v^2+96*Y^(1/2)*l^(5/2)*c*v^3+8*l^(3/2)*Y^(1/2)*v^3*c+24*l^(3/2)*
```

```
Y^(1/2)*c^2*v+24*l^(3/2)*Y^(1/2)*c*v^2-16*Y^(1/2)*l^(1/2)*v^4-32*Y^(1/
2)*l^(1/2)*v^3*c-16*l^(1/2)*Y^(1/2)*c^2*v^2+4*l^(3/2)*v^4+8*l^(1/2)*v
^4-8*l^(3/2)*v^3-72*l^(3/2)*c^2*v^2-40*l^(3/2)*c^3*v-144*l^(3/2)*v^4*
c+128*c*l^(5/2)*v^4-28*l^(3/2)*c*v^2-184*l^(3/2)*v^3*c^2+16*l^(5/2)*v^
4+32*l^(5/2)*v^5+144*c^2*l^(5/2)*v^2+192*c^2*l^(5/2)*v^3+112*c^3*l^(5
/2)*v+128*c^3*l^(5/2)*v^2-96*l^(3/2)*v^2*c^3-16*l^(3/2)*v*c^4-28*c^2*l
^(3/2)*v+80*c*l^(5/2)*v^3-28*l^(3/2)*c*v^3-40*l^(3/2)*v^5-8*c^3*l^(3/2
)+16*l^(1/2)*v^2*c^3+20*l^(1/2)*v^3*c+16*l^(1/2)*v^2*c^2+4*l^(1/2)*v*c
^3+40*l^(1/2)*v^4*c+44*l^(1/2)*v^3*c^2+32*c^4*l^(5/2)*v+12*l^(1/2)*v^
5+32*c*l^2*X^(1/2)*v^2+16*c^3*l^2*X^(1/2)*v-8*l*X^(1/2)*v*c^3+40*c^2*l
^2*X^(1/2)*v-32*l*X^(1/2)*v^2*c^2+16*c^3*l^2*X^(1/2)-38*l*X^(1/2)*v^3
*c+48*c*l^2*X^(1/2)*v^3+c^2*X^(1/2)*v+X^(1/2)*v^3+2*X^(1/2)*v*c^3-8*X
^(1/2)*c^3*l+6*X^(1/2)*v^2*c^2+2*l*X^(1/2)*v^2+2*c*X^(1/2)*v^2+6*X^(1
/2)*v^3*c+16*l^2*X^(1/2)*v^4-14*l*X^(1/2)*v^4-26*c*l*X^(1/2)*v^2-26*c^
2*l*X^(1/2)*v+l*X^(1/2)*v*c+48*c^2*l^2*X^(1/2)*v^2-9*l*X^(1/2)*v^3+8*l
^2*X^(1/2)*v^3-2*Y^(1/2)*X^(1/2)*v^3+32*Y^(1/2)*l^(5/2)*v^4+2*X^(1/2)
*v^4+8*l^(3/2)*Y^(1/2)*v^4+8*Y*l*X^(1/2)*v^2-2*Y^(1/2)*X^(1/2)*c^2*v+2
*l*X^(1/2)*v^3*Y^(1/2)+32*c^4*l^(5/2))/(2*l^(3/2)*v+2*l^(3/2)*c+X^(1/2)
*l-l^(1/2)*v-l^(1/2)*c)^3/X^(1/2)
```

avec $X = 4l(v+c)^2 - 3v^2 - 4vc - c + 4v\sqrt{v+c} - 2v$ et $Y = v + c$.

Nous pouvons préciser le comportement de c_{RR} en fonction de l, v et c.
Ainsi le terme dominant en l est :

$$\frac{2v^2 + v + 2v\sqrt{v+c} + 2vc + 2c}{2v + 2c} + O(\frac{1}{l})$$

le terme dominant en v est :

$$\frac{4l^{5/2}(\sqrt{4l-3} - 20l^{3/2} - 7\sqrt{4l-3}\,l + 16l^{5/2} + 6\sqrt{l} + 8l^2\sqrt{4l-3})}{(2l^{3/2} + \sqrt{4l-3}\,l - \sqrt{l})^3\sqrt{4l-3}} v + O(v^{1/2})$$

(on retrouve bien le comportement linéaire en fonction du nombre de variables de [ACDT90]),
et le terme dominant en fonction de c est :

$$4\frac{(4lv + 4l - v)l}{(4l-1)^2} + O(\frac{1}{c^{1/2}})$$

On remarque donc que la complexité diminue quand l et c augmentent ce qui s'explique par une proportion de clashs plus grande; elle augmente avec v qui fait diminuer la probabilité du clash au profit de l'occurrence.

On peut par exemple donner les valeurs numériques :

c_{RR} se comporte comme $3.75 + 1.9\frac{1}{l} + O(\frac{1}{l^2})$ à v et c fixés égaux à 2,

c_{RR} se comporte comme $1.3v + 1.45v^{1/2} + 0.3 + O(\frac{1}{v^{1/2}})$ à l et c fixés égaux à 2,

c_{RR} se comporte comme $3.6 + 2.9\frac{1}{c^{1/2}} + 0.1\frac{1}{c} - 3.2\frac{1}{c^{3/2}} + O(\frac{1}{c^2})$ à l et v fixés égaux à 2,

et finalement, on a $c_{RR} \simeq 4.97$ pour $l = 2$, $v = 2$ et $c = 2$.

4.3. Coût moyen de l'algorithme de Herbrand avec oracle

Nous allons déterminer le coût moyen exact de l'algorithme de Herbrand avec oracle c'est à dire en considérant que l'algorithme dispose d'un moyen de déterminer lorsque la paire (v, T) du système Γ fait partie de l'unificateur principal mis sous forme triangulaire autrement dit lorsque la règle **Select** s'applique (cet oracle pouvant être le résultat d'une initialisation de compteurs sur les variables comme dans la phase de prétraitement lineaire de Martelli Montanari ou d'un parcours arrière de pointeurs comme dans *Paterson Wegman*). A nouveau, d'après le lemme de la section 4.1.2, nous savons que le coût moyen de l'algorithme sur des paires d'arbres quelconques va coïncider avec son coût moyen sur les paires d'arbres non-unifiables en raison d'un échec direct.

L'expression récursive $u_{HO}(T, T')$ du coût moyen de l'algorithme sur une paire de termes $(T, T') \in \mathcal{FF}$ est identique à celle de $u_{RR}(T, T')$ à la seule différence que :

$u_{HO}(v, T) = u_{HO}(T, v) = 1$ puisque que c'est l'oracle qui détermine l'application ou la non application de **Select**.

A nouveau, nous pouvons exprimer $u_{HO}(T, T')$ pour $|T| + |T'| = n$ comme le $n^{ième}$ coefficient de la série génératrice $U_{HO}(z)$ qui vérifie l'équation algébrique :

$$\begin{aligned} U_{HO}(z) = {} & c^2 - c + 2vC^+(z) + 2cB^+(z) + (l^2 - l)z^2B^4(z) \\ & + l(z^2B^2(z)FF(z) + z^2B^2U_{HO}(z)) \\ & + l(z^2EE(z)FF(z) + 2z^2FF(z)(EE(z) + zEE'(z)) + z^2EE(z)U_{HO}(z)) \end{aligned}$$

L'expression explicite de $U_{HO}(z)$ est encore trop grande pour être écrite.

La singularité de plus petit module de $U_{HO}(z)$ est à nouveau $z = \frac{1}{4l(v+c)}$. Nous pouvons alors effectuer l'analyse asymptotique qui fournit

$$U_{HO}(z) = K_3\sqrt{1 - 4lz(v + c)} + O(z - \frac{1}{4l(v + c)})$$

d'où

$$[z^n]U_{HO}(z) = K_4(4l(v + c))^n n^{-3/2}\left(1 + O\left(\frac{1}{n}\right)\right)$$

avec K_3 et K_4 des constantes dependants de l, v et c.

Le coefficent $[z^n]U_{HO}(z)$ est encore de même ordre que $[z^n]FF(z) = [z^n]BB(z)$, on a donc le résultat :

THÉORÈME 7 : *Le coût moyen de l'algorithme de Herbrand avec oracle sur des paires d'arbres quelconques est constant égal à* c_{HO}.

Nous pouvons donner l'expression la plus "compacte" de la constante c_{HO} obtenue à nouveau avec le système MAPLE (les constantes X et Y sont les mêmes que pour c_{RR}) :

```
4*Y^(1/2)*l^(5/2)*(-13*l^(1/2)*v^3-2*l^(1/2)*v^2+20*l^(3/2)*v^3-12*l*Y
^(1/2)*X^(1/2)*v*c+8*l^2*Y^(1/2)*X^(1/2)*c^2+2*l^(1/2)*Y^(1/2)*c*v-12*l
^(3/2)*c*Y^(1/2)*v+8*l^(1/2)*Y^(1/2)*v^2-8*l^(3/2)*Y^(1/2)*v^2+10*l^(1
/2)*Y^(1/2)*v^3-4*l^(3/2)*c^2*Y^(1/2)+16*l^(5/2)*Y^(1/2)*c^3+48*l^(5/2
)*Y^(1/2)*v*c^2-44*l^(3/2)*c*Y^(1/2)*v^2-24*l^(3/2)*c^2*Y^(1/2)*v+48*l
^(5/2)*Y^(1/2)*v^2*c+12*l^(1/2)*Y^(1/2)*c*v^2+16*l^(5/2)*Y^(1/2)*v^3-4
*Y*X^(1/2)*v-20*l^(3/2)*Y^(1/2)*v^3+8*l^2*Y^(1/2)*X^(1/2)*v^2+16*l^2*Y
```

```
^(1/2)*X^(1/2)*v*c-4*l*Y^(1/2)*X^(1/2)*c^2-8*l*Y^(1/2)*X^(1/2)*v^2+8*l*
Y*X^(1/2)*v+2*Y^(1/2)*X^(1/2)*v*c+2*l*X^(1/2)*v^3+2*Y^(1/2)*X^(1/2)*v^2
+2*l*c^2*X^(1/2)*v-X^(1/2)*v^3+4*l*c*X^(1/2)*v^2-X^(1/2)*c^2*v+3*X^(1/
2)*v^2-6*l*c*X^(1/2)*v-6*l*X^(1/2)*v^2+3*X^(1/2)*c*v-2*X^(1/2)*c*v^2+20*
l^(3/2)*v*c^2+12*l^(3/2)*v^3*c-15*l^(1/2)*v^2*c-l^(1/2)*v*c^2-4*l^(1/2
)*v^2*c^2+40*l^(3/2)*v^2*c-l^(1/2)*v*c-7*l^(1/2)*v^3*c+4*l^(3/2)*v*c^3
+12*l^(3/2)*v^2*c^2+4*l^(3/2)*v^4-3*l^(1/2)*v^4)/(2*l^(3/2)*v+2*l^(3/2
)*c+X^(1/2)*l-l^(1/2)*v-l^(1/2)*c)^3/X^(1/2)
```

Nota : On a bien $c_{HO} \leq c_{RR}$.

Nous pouvons préciser le comportement de c_{HO} en fonction de l, v et c.
Ainsi le terme dominant en l est :

$$1 + O(\frac{1}{l})$$

le terme dominant en v est :

$$\frac{4l^{5/2}(-3\sqrt{l} - \sqrt{4l-3} + 2\sqrt{4l-3}\, l + 4l^{3/2})}{(2l^{3/2} + \sqrt{4l-3}\, l - \sqrt{l}\,)^3 \sqrt{4l-3}} v^{1/2} + O(1)$$

et le terme dominant en fonction de c est :

$$\frac{16l^2}{(4l-1)^2} + O(\frac{1}{c^{1/2}})$$

On remarque à nouveau que le coût moyen de l'algorithme diminue quand l et c augmentent; cette fois-ci son augmentation avec v n'est plus qu'en $v^{1/2}$ ce qui s'explique par le fait que l'on ne fait plus le test d'occurrence, les variables ont donc moins d'influence sur la complexité.

On peut par exemple donner les valeurs numériques :
c_{HO} se comporte comme $1 + 0.70\frac{1}{l} + O(\frac{1}{l^2})$ à v et c fixés égaux à 2,
c_{HO} se comporte comme $0.21v^{1/2} + 1.18 + O(\frac{1}{v^{1/2}})$ à l et c fixés égaux à 2,
c_{HO} se comporte comme $1.30 + 0.32\frac{1}{c^{1/2}} + O(\frac{1}{c})$ à l et v fixés égaux à 2,
et finalement, on a $c_{HO} \simeq 1.47$ pour $l = 2$, $v = 2$ et $c = 2$.

4.4. Coût moyen de l'algorithme de Martelli-Montanari avec initialisation retardée des compteurs

Nous allons déterminer le coût moyen exact de l'algorithme de Martelli-Montanari avec initialisation retardée des compteurs c'est à dire la variante de l'algorithme de Martelli-Montanari dans laquelle la détermination des compteurs (qui sont nécessaires pour savoir si la règle **Select** s'applique) n'est pas realisée lors d'une phase de prétraitement linéaire mais directement conjointement pendant la détermination de la première frontière des termes à unifier. On pourrait penser que cette amélioration de l'algorithme de Martelli Montanari va conduire à un coût moyen constant sur des paires d'arbres quelconques (alors que celui-ci pour l'algorithme classique est nécessairement linéaire en raison de la phase de prétraitement); nous allons voir que cette variante ne modifie pas le comportement moyen de l'algorithme de Martelli Montanari et reste linéaire en moyenne.

Encore une fois, d'après le lemme de la section 4.1.2, nous savons que le coût moyen de l'algorithme sur des paires d'arbres quelconques va coïncider avec son coût moyen sur les paires d'arbres non-unifiables en raison d'un échec direct.

L'expression récursive $u_{MM}(T,T')$ du coût moyen de l'algorithme sur une paire de termes $(T,T') \in \mathcal{FF}$ est identique à celle de $u_{HO}(T,T')$ à la seule différence que :

$u_{MM}(v,T) = u_{MM}(T,v) = 2|T|+1$ puisque l'algorithme parcourt le sous-arbre $|T|$ tout entier pour l'initialisation des compteurs.

A nouveau, nous pouvons exprimer $u_{MM}(T,T')$ pour $|T|+|T'| = n$ comme le $n^{ième}$ coefficient de la série génératrice $U_{MM}(z)$ qui vérifie l'équation algébrique :

$$\begin{aligned} U_{MM}(z) = {} & c^2 - c + 4vzC^+(z)' + 2vC^+(z) + 2cB^+(z) + (l^2-l)z^2B^4(z) \\ & + l(z^2B^2(z)FF(z) + z^2B^2U_{MM}(z)) \\ & + l(z^2EE(z)FF(z) + 2z^2FF(z)(z)(EE(z) + zEE'(z)) + z^2EE(z)U_{MM}(z)) \end{aligned}$$

La singularité de plus petit module de $U_{MM}(z)$ est à nouveau $z = \frac{1}{4l(v+c)}$. Nous pouvons alors effectuer l'analyse asymptotique qui fournit cette fois-ci :

$$U_{MM}(z) = \frac{16v(v+c)^2l^{1/2}}{2(v+c)l^{1/2} - \sqrt{X}} \frac{1}{\sqrt{1-4lz(v+c)}} + O(z - \frac{1}{4l(v+c)})$$

avec X la même constante que pour l'expression de c_{RR} d'où

$$[z^n]U_{MM}(z) = \frac{16v(v+c)^2l^{1/2}}{2(v+c)l^{1/2} - \sqrt{X}} \frac{(4l(v+c))^n}{\sqrt{\pi}} n^{-1/2}\left(1 + O\left(\frac{1}{n}\right)\right)$$

Le coefficent $[z^n]U_{MM}(z)$ n'a donc pas le même ordre asymptotique que $[z^n]BB(z)$ (il diffère par le coefficient polynômial en n), ceci conduit au résultat :

THÉORÈME 8 : *Le coût moyen de l'algorithme de Martelli Montanari avec initialisation retardée des compteurs sur des paires d'arbres quelconques est linéaire égal à*

$$c_{MM}n\left(1 + O\left(\frac{1}{n}\right)\right) = \frac{4vl^{1/2}}{2(v+c)l^{1/2} - \sqrt{X}} n\left(1 + O\left(\frac{1}{n}\right)\right)$$

avec $X = 4lv^2 + 8lvc + 4lc^2 - 3v^2 - 4vc - c + 4v\sqrt{v+c} - 2v$.

On peut donner les expressions explicites des variations de c_{MM} en fonction de l, v et c :

- à v et c fixés, c_{MM} se comporte comme

$$\frac{16(v+c)v}{3v^2 + 4vc + c - 4v\sqrt{v+c} + 2v} l - \frac{v}{v+c} + O(\frac{1}{l})$$

- à l et c fixés, c_{MM} se comporte comme

$$-\frac{4\sqrt{l}}{\sqrt{4l-3} - 2\sqrt{l}} + \frac{8\sqrt{l}}{(\sqrt{4l-3} - 2\sqrt{l})^2\sqrt{4l-3}} \frac{1}{\sqrt{v}} + O(\frac{1}{v})$$

- à l et v fixés, c_{MM} se comporte comme

$$\frac{16lv}{1+4v} + \frac{64lv^2}{(1+4v)^2} \frac{1}{\sqrt{c}} + O(\frac{1}{c^{3/2}})$$

4.5. Conclusion

Nous pouvons tirer quelques conclusions d'après les précédents résultats sur la complexité en moyenne des différents algorithmes d'unification. Tout d'abord, on se rend compte que les clashs ne suffisent pas pour assurer un coût en moyenne constant. Il est nécessaire pour cela d'effectuer un test d'occurrence efficace (c'est à dire qui ne nécessite pas un parcours total des sous-termes). Ensuite, il nous semble que la variante de l'algorithme de Robinson, que nous avons appelé avec retard sur les substitutions, est l'algorithme le plus efficace sur des paires d'arbres quelconques avec distribution uniforme : en effet, il n'a besoin d'aucune phase de prétraitement, d'aucune structure de données particulière et son coût est en moyenne constant. Enfin, il est facile de réaliser que la complexité moyenne de tout algorithme d'unification (sans oracle bien sûr) sur les paires d'abres unifiables est *au moins linéaire* puisque sur la famille $\mathcal{VB}$ le test d'occurrence nécessite un parcours total de T.

V. Modèle avec une infinité de variables

Comme annoncé en fin de partie III, nous reprenons notre étude sur le phénomène de l'occurrence pour des termes arborescents formés à partir d'un nombre de variables non bornés (cette présente étude se trouve combinatoirement justifiée par le théorème 10).

5.1. Motifs avec une infinité de variables au renommage pres

Notre motivation dans cette partie est d'étudier l'influence d'un modèle avec une infinité de variables sur le phénomène d'occurrence. Afin de simplifier les calculs, on va donc considérer des termes sans constante et avec un seul symbole interne noté $*$. Plus précisément, on modélise ces termes par des *motifs* constitués d'une structure d'arbre binaire avec aux feuilles des variables égales au renommage près. Ainsi il n'y a que deux motifs de taille 1 (la taille continue de désigner le nombre de nœuds internes) qui sont :

$$*(x,x) \quad \text{ou} \quad *(x,y)$$

ce qui traduit le fait que les feuilles variables pour les motifs de taille 1 sont soit égales soit différentes. Les motifs de taille 2 possibles sont :

$$*(*(x,x),x),\ *(*(x,y),x),\ *(*(y,x),x),\ *(*(x,x),y),\ *(*(x,y),z)$$

et avec la deuxième structure binaire

$$*(x,*(x,x)),\ *(x,*(x,y)),\ *(x,*(y,x)),\ *(x,*(y,y)),\ *(x,*(y,z))$$

On a le théorème de dénombrement suivant :

THÉORÈME 9 : *Le nombre d'arbres binaires avec n noeuds internes et $n+1$ feuilles contenant des variables au renommage de celles-ci près (ou motifs de taille n) est*

$$C_n B_{n+1}$$

où C_n est le $n^{ième}$ nombre de Catalan et B_{n+1} le $n+1^{ième}$ nombre de Bell.

PREUVE :.

En effet C_n est le nombre d'arbres binaires avec n nœuds internes et B_n le nombre de façons de choisir une liste de n variables au renommage près.

Avant de prouver le résultat, faisons un *rapide survol* des célèbres nombres C_n et B_n. Les C_n dénombrent, entre autres, le nombre d'arbres binaires de taille n (nœuds internes) et les B_n sont connus pour dénombrer les partitions d'ensembles (cf. [Com74]). Pour référence, les premiers entiers de Catalan sont 1, 1, 2, 5, 14, 42 ... et les premiers entiers de Bell sont 1, 1, 2, 5, 15, 52 ... On peut donner les expressions suivantes :

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \sum_{p=0}^{n-1} C_p C_{n-p-1}$$

$$B_n = e^{-1} \sum_{k\geq 0} \frac{k^n}{k!} = \sum_{p=0}^{n-1} \binom{n-1}{pB_p}$$

et les séries génératrices :

$$\sum_{n\geq 0} C_n z^n = \frac{1-\sqrt{1-4z}}{2z}$$

$$\sum_{n\geq 0} B_n \frac{z^n}{n!} = e^{e^z - 1}$$

Remarque : on peut également donner une expression sous la forme d'une fraction continue de la série génératrice ordinaire des B_n

$$\sum_{n\geq 0} B_n z^n = \cfrac{1}{1 - 1\cdot z - \cfrac{1\cdot z^2}{1 - 2\cdot z - \cfrac{2\cdot z^2}{1 - 3\cdot z - \cfrac{3\cdot z^2}{\cdots}}}}$$

Cette série est purement divergente (rayon de convergence nul) (Cf. [Fla80]).

On n'a pas de forme explicite (hormis une forme intégrale) pour la série génératrice des motifs

$$M(z) = \sum_{n\geq 0} M_n \frac{z^n}{n!} = \sum_{n\geq 0} C_n B_{n+1} \frac{z^n}{n!}$$

mais on connait le comportement asymptotique de M_n produit des developpements connus [dB58] :

$$C_n \sim \frac{4^n n^{-\frac{3}{2}}}{\sqrt{\pi}}$$

$$B_n \sim \frac{\exp(\exp(s_n))}{e\sqrt{2\pi \exp(s_n)} {s_n}^{n+1}}$$

avec s_n qui verifie $s_n \exp(s_n) = n+1$ ce qui conduit au developpement

$$s_n = \ln(n) - \ln(\ln(n)) + \frac{\ln(ln(n))}{\ln(n)} + O\left(\frac{\ln(\ln(n))^2}{\ln(n)^2}\right)$$

On peut revenir sur la preuve du théorème 9 soit

$$M_n = C_n B_{n+1}$$

En effet, B_n est le nombre de partitions de l'ensemble $\{1,\ldots,n\}$ et il existe une bijection entre les partitions de $\{1,\ldots,n\}$ et les listes de n variables au renommage près. Etant donnée une liste de variables, on associe à chaque variable l'ensemble des places qu'elle occupe dans la liste. Par exemple, à la liste :

$$(x,y,y,z,t,x,z,y)$$

est associée la partition :

$$\{\{1,6\},\{2,3,8\},\{4,7\},\{5\}\}.$$

Réciproquement, étant donnée une partition, on associe à chaque partie une variable, dont les places dans la liste sont données par les entiers de la partie. Pour l'exemple ci-dessus, on retrouve bien la suite initiale, au renommage près. ▪

A présent, nous allons calculer le nombre moyen de *variables distinctes* dans un tel motif de taille n (n nœuds internes et $n+1$ variables). Du fait de la bijection mise en évidence ci-dessus, (chaque variable correspond à une partie), c'est aussi le nombre moyen de parties dans une partition de $n+1$ éléments. On peut obtenir simplement la valeur de ce paramètre par des calculs classiques de séries génératrices mais nous allons en profiter pour illustrer l'utilisation d'un système d'analyse combinatoire automatique comme Lambda-Upsilon-Omega ($\Lambda\Upsilon\Omega$) ([FSZ89]) :

```
type
        partition = set(ens_non_vide);
        ens_non_vide = [u] set(variable,card>=1);
        variable = Latom(1);

to_analyze: moment(u,partition,1);
```

Dans la définition ci-dessus, nous avons marqué chaque partie par la lettre u, et nous avons demandé le premier moment de la marque u dans les partitions, c'est à dire le nombre moyen de parties. L'analyse de ce fichier par $\Lambda\Upsilon\Omega$:

```
Luo 1.2   Mon Dec  4 15:59:34 MET 1989  (Maple 4.2  Caml V2-6)

#analyze "partitions";;
                          No singularities found

                      The saddle point is , W(n + 1)

                        Saddle point's expansion:

                                                                  2
                                    ln(ln(n))          ln(ln(n))
(ln(n)) + (- ln(ln(n))) + (-----------) + (O(------------))
                                      ln(n)                 2
                                                         ln(n)
```

```
Moment of order 1 associated to the mark u in partition is:

                                                exp(_saddlepoint)
(exp(_saddlepoint)) + (O(--------------------))
                                                  _saddlepoint
```

ce qui signifie que résultat est $\exp(s_n)$, où le développement de s_n est donné ci-dessus. On a donc le résultat :

THÉORÈME 10 : *Le nombre moyen de variables différentes dans un motif de taille n*

$$V_n = \frac{n}{\ln n}(1 + O(1)).$$

Le fait fondamental demontre par ce théorème est que $V_n \longrightarrow \infty$ *avec* n, fait que ne modélise pas un modèle avec un nombre v fini de variables. Ce résultat justifie bien notre présente étude avec un nombre de variables non borné.

Remarque : On pourrait être tenté de reprendre, dans ce modèle de termes avec une infinité de variables au renommage près, l'étude sur la famille des paires d'arbres égalisables pour montrer là encore, par exemple, que les paires d'arbres unifiables sont négligeables mais le fait que le renommage d'un terme ne s'induise pas de façon récursive aux sous-termes, empêche une telle extension.

5.2. Influence sur l'occurrence

Nous allons reconsidérer notre étude sur la probabilité d'occurrence avec cette fois-ci un nombre de variables v qui pourra tendre vers l'infini de concert avec n.

Pour cela nous allons considérer les paires de termes élémentaires (T, x) avec x une variable et T un arbre binaire de taille n avec $n+1$ feuilles variables prises dans un ensemble de variables de taille v. Le nombre total de tels couples est donc $vC_n v^{n+1}$ et le nombre total de tels couples pour lesquels il n'y a pas d'occurrence est $vC_n(v-1)^{n+1}$. Le rapport vaut donc $(1-\frac{1}{v})^{n+1}$ et on a le résultat :

THÉORÈME 11 : *La probabilité pour qu'un couple de taille globale* n (T, x) *échoue par occurrence de la variable* x *dans le terme* T *est :*

$$P(n, v) = 1 - (1 - \frac{1}{v})^{n+1}$$

avec v *le cardinal de l'ensemble des variables* V.

- Ainsi si v reste constant la probabilité d'occurrence tend vers 1 quand $n \longrightarrow \infty$. C'est le cas du modèle précédent où le nombre de variables v est fixé, ce qui explique l'importance obtenue pour le phénomène de l'occurrence.

- C'est encore le cas lorsque $v = \frac{n}{\ln(n)}$, ce qui correspond au nombre moyen de variables distinctes, comme on vient de le montrer, pour des motifs arborescents avec égalite au renommage des variables près. La probabilité d'occurrence se comporte comme $P = 1 - \frac{1}{n}$. (exactement $P = 1 - \frac{1}{n} + \frac{\frac{1}{2}\ln(\frac{1}{n})^2 - \ln(\frac{1}{n})}{n^2} + O(\frac{1}{n^3})$). On va retrouver et préciser ce résultat par un calcul direct dans la section suivante.

- Lorsque le nombre de variables disponibles v est de l'ordre de la taille du terme n, on a $P(n,n) = 1-(1-\frac{1}{n})^{n+1} \longrightarrow 1-\frac{1}{e}$: c'est le cas limite.

- Si v est de l'ordre de $n^{1+\alpha}$ avec $\alpha > 0$ (dès que v est superlinéaire en n) alors la probabilité d'occurrence $P(n,v)$ tend vers 0.

5.3. Probabilité d'occurrence avec le modèle des motifs

On reconsidére les couples (T,x) avec égalite au renommage des variables près et on va calculer directement dans ce modèle uniforme la probabilité d'échec par occurrence entre le motif T et la variable x. Par exemple, il existe 5 tels couples (*motif*, *variables*) de taille globale 1 au renommage près qui sont :

$$(*(x,x) , x) \quad (*(x,x) , y) \quad (*(x,y) , x) \quad (*(x,y) , y) \quad (*(x,y) , z)$$

dont 2 seulement ne conduisent pas à une occurrence. Cet exemple montre qu'un terme contenant k variables distinctes induit $k+1$ paires (*terme*, *variable*), dont une seule ne produit pas d'occurrence.

On peut tout de suite faire la critique à ce modèle uniforme que, s'il emploie effectivement une infinité de variables, une variable différente de celles du motif n'aparaît (et donc n'est comptée) qu'une seule fois ! Ce fait intervenant sans doute plus souvent dans la pratique, la probabilité d'occurence que l'on va obtenir reste vraisemblablement supérieure à la probabilité "réelle".

La probabilité de non-occurrence sur les termes de taille n est donc :

$$\frac{\sum_{|T|=n} 1}{\sum_{|T|=n}(vard(T)+1)}$$

avec $vard(T)$ le nombre de variables distinctes du terme T.

Exemple : Ainsi pour nos couples de taille 1, la probabilité de non-occurrence est

$$\frac{\text{nombre de termes de taille 1}}{\sum_{|T|=1}(vard(T)+1)} = \frac{2}{2+3} = \frac{2}{5}$$

Il est plus facile de calculer l'inverse de ce nombre, qui est le nombre de parties + 1 d'une partition, à l'aide de ΛγΩ, par le programme :

```
procedure nb_parties_plus_un (p: partition);
begin
  forall e in p do count;
  count
end;
```

dont l'analyse donne :

```
tau_nb_parties_plus_un(z) = exp(z) exp(exp(z) - 1)

Average cost for nb_parties_plus_un on random inputs of size n is:

(_saddlepoint exp(_saddlepoint)) + (O(exp(_saddlepoint)))
```

Le point col étant toujours s_n, on retrouve bien le résultat

THÉORÈME 12 : *La probabilité d'occurrence dans un motif de taille n est*

$$P = 1 - \frac{1}{n} + o(\frac{1}{n}).$$

VI. Conclusion

Nous avons montré, avec une loi de distribution uniforme sur les arbres binaires formés à partir d'un ensemble *fini* de variables V, que la famille des paires d'arbres unifiables est exponentiellement négligeable devant la famille des paires d'arbres quelconques. Nous avons analysé les différentes causes d'échec dans ce modèle et précisé l'analyse de l'occurrence dans un modèle avec un nombre infini de variables. Ainsi nous avons montré que, dans le modèle où V est fini, l'occurrence est une cause d'échec non négligeable devant le clash mais d'importance numérique moindre. Plus généralement, nous obtenons que la probabilité d'occurrence d'une variable dans un terme de taille n est $1 - (1 - \frac{1}{v})^{n+1}$ avec v le cardinal de V. Cette valeur tend vers 1 quand n tend vers l'infini si le nombre de variables est fini, comme en partie III, ou si le nombre de variables est de l'ordre du nombre moyen de variables distinctes dans un terme arborescent avec égalité au renommage près, comme en partie V. Cette probabilité d'occurrence tend vers 0 dès que le nombre de variables disponibles est superlinéaire ($n^{1+\alpha}$ with $\alpha > 0$).

Nous avons montré, dans le modèle avec V fini, que le coût moyen de l'algorithme de Robinson où l'on retarde l'application des substitutions est *constant* sur la famille des paires d'arbres quelconques; ce qui en fait l'algorithme d'unification le plus efficace sur cette famille de termes (puisque ne nécessitant de plus aucune structure de donnée particulière ni prétraitement). D'autre part, le coût moyen de l'algorithme de Martelli Montanari et de ses variantes (phase d'initialisation des compteurs menée de pair avec la détermination de la première frontière ou décompositions réalisées avec les compactifications) est *linéaire*. Le fait crucial est que les échecs par clash ne suffisent pas à assurer une complexité en moyenne constante sur les paires d'arbres quelconques; il est nécessaire d'effectuer un test d'occurrence efficace c'est à dire sans parcours complet des sous-termes. La complexité en moyenne de l'algorithme de Paterson Wegman dans un cadre général (*i.e.* un résultat constant sur les paires d'arbres non-unifiables ou polynomial sur les paires d'arbres unifiables) reste un problème ouvert.

L'analyse asymptotique induit des considérations sur des termes de grande taille. Ce fait peut être critiqué puisque dans la pratique, on n'unifie peu de grands termes mais plutôt de longues séquences de termes de petite taille. Le problème de l'unification *in-line* de séquences de petits termes est plus proche de la réalité. Il faut noter que, en raison de la composition des substitutions, on ne peut pas déduire d'un algorithme d'unification linéaire pour une paire de termes un algorithme d'unification linéaire pour une séquence de termes. Ainsi la complexité dans le cas le pire de l'algorithme proposé dans [MM86] pour exécuter une séquence d'opérations unification-desunification est $O(n \ \ln(\ln(n)))$. On peut penser qu'une approche pour déterminer la complexité moyenne de tels séquences serait de considérer des *forêts d'arbres binaires* et de réaliser l'analyse asymptotique sur la taille d'une telle forêt, comme dans [AF88].

VII. Bibliographie

[AHU74] A.V. Aho, J.E. Hopcroft and J.D. Ullman. *The design and analysis of computer algorithms*. Addison Wesley Pub. Comp. 1974.

[AF88] L. Albert and F. Fages. Average case complexity analysis of the RETE pattern match algorithm. *Proceedings of the* 15^{th} *International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science* 317, Tampere, Finland, pages 18-37. Springer Verlag, July 1988.

[Alb89b] L. Albert. Average case complexity analysis of RETE pattern match algorithm and average size of join in Databases. *Proceedings of the* 9^{th} *conference on Foundations of Software Technology and Theoretical Computer Science, Lecture Notes in Computer Science* 405, Bangalore, India, pages 223-241. Springer Verlag, December 1989.

[Alb90] L. Albert. Analyse en moyenne des algorithmes d'unification. INRIA Research Report. To appear. 1990.

[ACDT90] L. Albert, R. Casas, J. Diaz and A. Torrecillas. On unification over unrestricted pairs of trees. Submitted to *Mathematical Foundations of Computer Science '90*, Bratislava, Czechoslovakia, August 1990. Report de Recerca LSI-89-26, universitat politecnica de catalunya, 1989. In *Alcom workshop on probabilistic algorithms and average case analysis*, Computer Technology Institute, Patras, Greece, March 4-7 1990.

[ACFZ90] L. Albert, R. Casas, F. Fages and P. Zimmermann. Average case analysis of unification algorithms. Submitted to *1990 North American Conference on Logic Programming*, Austin, Texas. October 1990.

[AR90] L. Albert and M. Regnier. Graph applications to recursive production rules programs. Submitted to *Third International Conference on Database Theory '90*, Paris, France, December 1990.

[BJSS88] A. Boudet, J.P. Jouannaud and M. Schmidt-Schauss. Unification in Boolean rings and Abelian groups. LICS'88. 1988.

[CAML89] Formel Project. *The CAML reference manual.* INRIA, 1989.

[CB83] A. Corbin and M. Bidoit. A rehabilitation of Robinson's unification algorithm. In *Information processing 83*, R.E.A. Mason, pages 909-914. North Holland 1983.

[CDS89] R. Casas, J. Diaz and J.M. Steyaert. Average case analysis of Robinson's unification algorithm with two different variables. *Information processing letters*, 31, pages 227-232, June 1989.

[CKS86] C.Choppy, S.Kaplan and M.Soria. Algorithmic complexity of term rewriting systems. In *Proceedings of the First Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science* 256. Dijon, France, 1986.

[CL73] C.L. Chang and R.C. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New-York. 1973.

[Com74] L. Comtet.*Advanced Combinatorics.* Reidel, Dordrecht, 1974.

[Col78] A. Colmerauer. Les grammaires de métamorphose. *Natural Language Communication with Computer, Lectures Notes in Computer Science.* Springer-Verlag. 1978.

[Col82] A. Colmerauer. Prolog II, manuel de référence et modèle théorique. Rapport interne, Groupe d'Intelligence Artificielle, Université d'Aix-Marseille II. March 1982.

[Col84] A. Colmerauer. Equations and inequations on finite and infinite trees. *Proceedings of the International Conference on Fith Generation Computer Systems.* 1984.

[Cou83] B. Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science.* 1983

[dB58] N.G. de Bruijn. *Asymptotic Methods in Analysis.* North Holland, third edition, 1958. Reprinted by Dover, 1981.

[DL89] N. Dershowitz and N. Lindenstrauss. Average time analyses related to logic programming. In 6^{th} *International Conference on Logic Programming*, pages 369-381. Lisboa 1989.

[DKM84] C. Dwork, P.C. Kanellakis and J.C. Mitchell. On the sequential nature of unification. *Journal of logic programming* 1, pages 35-50. 1984.

[Die68] J. Dieudonné. *Calcul infinitésimal.* Hermann, 1968.

[EG88] G. Escalada-Imaz and M. Ghallab. A practically efficient and almost linear unification algorithm. *Artificial Intelligence*, volume 36, number 2, pages 249-263. Sept. 1988.

[Fag83] F. Fages. *Formes canoniques dans les algèbres booléennes, et application à la démonstration automatique en logique de premier ordre.* Thèse de l'Université de Paris VI. 1983.

[Fag84] F. Fages. Associative-commutative unification. *7th CADE*, Napa Valley. 1984.

[FH86] F. Fages and G. Huet. Complete sets of unifiers and matchers in equational theories. *Theoretical Computer Science.* July 1986.

[Fla80] P. Flajolet. Combinatorial aspects of continued fractions. *Discrete Mathematics*, 32, pages 125-161, 1980.

[Fla88] P. Flajolet. Mathematical methods in the analysis of algorithms and data structures. In *Trends in theoretical computer science*, E. Borger, Computer science press, ch. 6, pages 225-304. Rockville, Maryland, 1988. (Lecture Notes for a *Graduate Course in Computer Science*, Udine, 1984).

[FSS90] P. Flajolet, P. Sipala and J.M. Steyaert. Analytic variations on the common subexpression problem. *Proceedings of the* 17^{th} *International Colloquium on Automata, Languages and Programming*, Warwick, England, July 1990.

[FSZ89a] P. Flajolet, B. Salvy and P. Zimmermann. Lambda-upsilon-omega : An assistant algorithms analyser. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 357 of *Lectures Notes in Computer Science*, pages 201-212, 1989. Proceedings AAECC6, Rome, July 1988.

[FSZ89b] P. Flajolet, B. Salvy and P. Zimmermann. Lambda-upsilon-omega : The 1989 Cookbook. Inria Research Report 1073, Institut National de Recherche en Informatique et Automatique, August 1989.

[FV87] P. Flajolet and J. Vitter. Average Case Analysis of Algorithms and Data Structures. In a *Handbook of Theoretical Computer Science*, North-Holland, 1987.

[Hen74] P. Henrici. *Applied and Computational complex Analysis*, volumes 1-3. John Wiley, New-York, 1977 .

[Her30] J. Herbrand. *Recherches sur la théorie de la démonstration.* Thèse de l'Université de Paris. 1930. In *Ecrits logiques de Jacques Herbrand.* Paris, PUF, 1968.

[HO80] G. Huet and D. Oppen. *Equations and Rewrite Rules: a Survey. In Formal Languages: Perspectives and Open Problems*, Ed. Book R., Academic Press. 1980.

[Hue75] G. Huet. A Unification Algorithm for Typed Lambda Calculus. *Theoretical Computer Science*, 1.1, pages 27-57. 1975

[Hue76] G. Huet. *Résolution d'équations dans les langages d'ordre* $1, 2, \ldots, \omega$. Thèse d'état de l'Université Paris VII. 1976.

[Jaf84] J. Jaffar. Efficient unification over infinite terms. *New Generation Computing*, 2, pages 207-219. 1984.

[KB70] D. Knuth and P. Bendix. Simple Word Problems in Universal Algebras. In *Computational Problems in Abstract Algebra*, Ed. Leech J., Pergamon Press, pages 263-297. 1970.

[Kow74] R.A. Kowalski. Predicate Logic as Programming Language. *Proceedings IFIP 74*, North Holland, pages 569-574. 1974.

[LMM86] J.L. Lassez, M.J. Maher and K. Marriott. Unification revisited. In *Foundations of deductive databases and Logic Programming*, J. Minker. 1986. IBM Research Report RC12355, 1986.

[CGG+88] B.W. Char, K.O. Geddes, G.H. Gonnet, M.B. Monagan and S.M. Watt. *MAPLE: Reference Manual.* University of Waterloo, 1988. 5th edition.

[Mai90] H.G. Mairson. Deciding ML typability is complete for deterministic exponential time. *POPL*, San Francisco. January 1990.

[Mil78] R. Milner. A Theory of Type Polymorphism in Programming. *JCSS 17*, pages 348-375. 1978.

[MM76] A. Martelli and U. Montanari. Unification in Linear Time and Space: a Structured Presentation. Internal Report B76-16, IEI, Pisa. July 1976.

[MM78] A. Meier and J.W. Moon. On the altitude of nodes in random trees. *Canadian Journal of Math.* 30, pages 997-1015. 1978.

[MM82] A. Martelli and U. Montanari. An efficient unification algorithm. In *ACM Transactions on Programming Languages and Systems*, volume 4, number 2, pages 258-282. April 1982.

[MU86] H. Mannila and E. Ukkonen. On the complexity of unification sequences. *Logic programming.* 1986.

[Plo72] G. Plotkin. Building-in Equational Theories. *Machine Intelligence 7*, pages 73-90. 1972.

[PW78] M.S. Paterson and M.N. Wegman. Linear unification. In *Journal of computer and system sciences* 16, pages 158-167. 1978.

[Rob65] J.A. Robinson. A machine-oriented logic based on the resolution principle. In *Journal of the Association for Computing Machinery*, volume 12, number 1, pages 23-41. Jan. 1965.

[Rob71] J.A. Robinson. Computational Logic: the Unification Computation. *Machine Intelligence 6*, Eds B. Meltzer and D.Michie American Elsevier, New-York. 1971.

[SF83] J.M. Steyaert, P. Flajolet. Patterns and pattern-matching in trees: an analysis. *Information and Control* 58, pages 19-58. 1983.

[SS86] L. Sterling and E. Shapiro. *The Art of Prolog: Advanced Programming Techniques.* MIT Press Series in Logic Programming, 1986.

[Ste84] J.M. Steyaert. *Complexité et structures des algorithmes.* Thèse d'Etat, Université de Paris 7, 1984.

[Sti75] M.E. Stickel. A Complete Unification Algorithm for Associative-Commutative Functions. *4th International Joint Conference on Artificial Intelligence*, Tbilisi. 1975.

[TvL84] R.E. Tarjan and J. van Leeuwen. Worst-case analysis of set union algorithms. In *Journal of the association for computing machinery*, volume 31, number 2, pages 245-281. April 1984.

[VZ75] M. Venturini-Zilli. Complexity of the Unification Algorithm for First-Order Expressions. *Calcolo XII*, Fasc. IV, pages 361-372. 1975.

Imprimé en France
par
. l'Institut National de Recherche en Informatique et en Automatique .

ISSN 0249-6399