Average case analysis of unification algorithms

Luc Albert, Rafaël Casas, Francois Fages, Paul Zimmermann

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° 1213

Programme 1
Programmation, Calcul Symbolique
et Intelligence Artificielle

AVERAGE CASE ANALYSIS OF UNIFICATION ALGORITHMS

Luc ALBERT
Rafaël CASAS
François FAGES
Paul ZIMMERMANN

Avril 1990

RR.1213

AVERAGE CASE ANALYSIS OF UNIFICATION ALGORITHMS *

Luc Albert [1] *Rafael Casas* [2]
François Fages [3] *Paul Zimmermann* [1]

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.*

ANALYSE EN MOYENNE DES ALGORITHMES D'UNIFICATION

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.*

* This research was supported by the ESPRIT BRA Program of the EC under contract no. 3075, project ALCOM.

[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

[2] Department of Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Pau Gargallo 5, 08028-Barcelona

[3] LIENS URA 1327 CNRS, Ecole Normale Supérieure 45 rue d'Ulm 75005 Paris France.

AVERAGE CASE ANALYSIS OF UNIFICATION ALGORITHMS *

Luc ALBERT [1] *Rafael CASAS* [2]
François FAGES [3] *Paul ZIMMERMANN* [1]

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.*

Keywords: unification algorithms, average case complexity, generating functions.

I. Introduction

Solving equations on terms of a first-order language is a central operation in symbolic computation. This problem was first studied by Herbrand [Her30] in proof theory, and was called unification in Robinson's foundational work on automatic theorem proving in first-order logic [Rob65]. Today first-order unification is at the heart of various systems ranging from theorem provers [CL73], [KB70], [HO80], to logic programming language interpreters [Kow74], [Col84], functional language type checkers [Mil78], [Mai90], natural language parsers [Col78], machine learning systems, etc... All these area of application motivated the search for efficient unification algorithms, as well as extensions for unification in higher-order languages [Hue75], [Hue76], unification in equational theories [Plo72], [FH86], [BJSS88] e.g. unification in presence of associative and commutative operators [Sti75], [Fag84], etc...

Robinson's unification algorithm [Rob65] takes as input two first-order terms and produces as output a most general substitution of the variables in the input terms that makes them equal, or failure if they are not unifiable. It is well-known that Robinson's algorithm has a worst-case time complexity which is

* This research was supported by the ESPRIT BRA Program of the EC under contract no. 3075, project ALCOM.

[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

[2] Department of Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Pau Gargallo 5, 08028-Barcelona

[3] LIENS URA 1327 CNRS, Ecole Normale Supérieure 45 rue d'Ulm 75005 Paris France.

exponential in the size of the input terms, even if the output substitution is represented in a triangular form and structure sharing techniques [Rob71] are used to represent terms by directed acyclic graphs (dag). The reason for the worst-case time complexity of this algorithm is that, although the number of nodes in the dags stays constant, the dags are traversed as trees without marking. When marking techniques are used the worst-case time complexity can be reduced to a polynomial of quadratic order in the size of the input [VZ75] [CB83]. By improving these ideas Martelli and Montanari [MM82] gave an unification algorithm in $O(n + vlogv)$ where n is the size of the input terms and v is the number of distinct variables, and Paterson and Wegman gave a *linear algorithm* [PW78] (also discovered independently by Martelli and Montanari [MM76]).

By considering unification as a particular kind of closure on classes of terms, Huet [Hue76] gave a quasi-linear unification algorithm based on the well-known set-union-find algorithm [AHU74]. Some variants are given in [Fag83], [Col84], [Jaf84], [EG88]. Huet's algorithm performs unification on regular trees [Cou83], *i.e.* finite and infinite terms represented by a finite possibly cyclic graph. Unification on regular trees lead to implement a variant of the logic programming language Prolog working on infinite expressions.

Huet's algorithm can be used for unification on finite terms by adding in a final pass a check for circularity. As it is linear this test does not change the worst-case time complexity of the algorithm, *i.e.* $O(n^2)$, $O(n \ln(n))$ or $O(nG(n))$ where G is an extremely growing function (an inverse of Ackermann's function) according to the different strategies for path compression that can be used in the set-union-find algorithm.

In [DKM84] it is proved that unification is P-time complete. This result means that unification is a sequential process in nature, and no significant gain in efficiency can be obtained from a parallel implementation. The result holds also for unification on infinite regular trees, but in that case Paterson and Wegman's linear sequential algorithm cannot be used. The existence of a linear algorithm for unifying infinite expressions is an open question (the same as for testing the equivalence of deterministic finite automata).

From a practical point of view there is no consensus on which unification algorithm is the best to use. While Paterson and Wegman's algorithm has the lowest 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 due to the initialization phase for setting up counters of variable occurrences in the terms (see the next section). Huet's algorithm in the $O(n^2)$ version presents no important overhead in comparison to Robinson's algorithm which, on the other hand, reveals its exponential behaviour only on pathological examples but *not* in practice. Furthermore in applications where non-unifiable problems are preponderant the ability to detect efficiently failures may be more significant than the worst-case time complexity. These reasons tend to explain why Robinson's algorithm is still used in many implementations of theorem provers, Prolog (usually with the unsound omission of the occur check, see the next section), type checkers [CAML89], etc ...

In this paper we try to make precise the preceding arguments by performing an average case complexity analysis of unification algorithms under the assumption that all terms of same size are equiprobable. In section II, we present in some details Herbrand non-deterministic algorithm from which we derive Robinson's and Martelli-Montanari's algorithms. In section III, we show that the family of unifiable pairs of binary trees formed over l binary function symbols, c constants and v variables is exponentially negligible with respect to the family of arbitrary pairs of binary trees. For this result we analyze the different causes of failure and get asymptotical and numerical evaluations (we extend the previous results of [ACDT90]).

In section IV, we first extend the previous results of [DL89] to these families of trees, we show that a slight modification of Herbrand-Robinson's algorithm uses in the average a *constant* number of steps over random pairs of trees. Then we show that Martelli and Montanari's algorithm uses on random pairs of trees a *linear* number of steps in the average. This result holds even if decompositions are mixed with compactifications and if the initialization of counters is mixed with the computation of the first frontier (instead of in a preliminary phase). The question of whether Paterson-Wegman's algorithm needs also a linear number of steps in average over random pairs of trees remains open.

On unifiable pairs of trees, the average complexity of Robinson's algorithm has been studied in [CDS89] for binary trees having only two different types of leaves and one internal node. In that particular case, there are only three possible substitutions and *no composition*. The latter phenomenon allows to count exactly the family of unifiable pairs of trees and permits to derive that the average complexity of Robinson's algorithm is linear on unifiable pairs of trees (while being quadratic in the worst case). On unifiable pairs of trees, we

can only assert that at least the average cost of unification is linear.

Finally, we study the probability of the occur check in presence of an unbounded number of variables, which is more closely related to many applications. In section V, we consider binary trees formed over one function symbol and a countable set of variables. The number of such binary trees of size n up to variable renaming is equal to $C_n B_{n+1}$ where C_n is the n^{th} Catalan number and B_{n+1} is the $n+1^{th}$ Bell number. We show that the average number of distinct variables in a tree of size n is $n/\ln(n)(1+O(1))$. The probability that a variable occurs in a term of size n is $1-(1-\frac{1}{v})^{n+1}$ where v is the number of variables. Therefore if v is of the order of the average number of distinct variables in a term of size n, or if v is a constant as in section III, the probability that a variable occurs in a term tends to 1 when the size of the term tends to the infinity.

II. Presentation of the algorithms and tree model

2.1. Presentation of basic unification algorithms

2.1.1 Herbrand-Robinson's algorithm

In this section we present basic unification algorithms in a general framework. Let F be a finite set of function symbols given with their arity α (constants are function symbols of arity 0). Let V be an *infinite* countable set of variables. We denote by $T(F,V)$ the set of *first-order terms*, that is the free F-algebra over V. Substitutions, denoted by σ, ρ, θ, ... are the F-endomorphisms of $T(F,V)$ with finite domain on V. They are represented by a finite set of elementary substitutions. For example with $\sigma = \{x \leftarrow a, y \leftarrow g(b,b)\}$, and $M = g(x, f(y))$ we get $M\sigma = g(a, f(g(b,b)))$. In the sequel we shall be interested mainly in idempotent substitutions for which a more compact representation can be given. A substitution σ is idempotent ($\sigma o \sigma = \sigma$) if and only if $D(\sigma) \cap I(\sigma) = \emptyset$, where D denotes the domain of σ and $I(\sigma)$ denotes the set of variables introduced by σ.

Substitutions define the preorder of *pattern-matching* on $T(F,V)$. We write: $M \leq N$ iff $\exists \sigma\ M\sigma = N$. In this case we say that term M is more general than N, or that N is an instance of M. The equivalence relation associated to $\leq$ over $T(F,V)$ is *variable renaming*: $M \equiv N$ iff $M \leq N$ and$N \leq M$. E.g. $g(x, f(y)) \equiv g(u, f(v)) \equiv g(y, f(x))$. When counting families of terms containing variables we shall be interested in the number of classes of terms of a given size up to variable renaming. The pattern-matching preorder on terms can be extended to substitutions. We say a substitution σ is more general than ρ, $\sigma \leq \rho$ iff $\exists \theta\ \sigma o \theta = \rho$.

DEFINITION : Two terms M and N are *unifiable* iff $\exists \sigma\ M\sigma = N\sigma$. We denote the set of unifiers of two terms by $U(M,N) = \{\sigma \mid \sigma M = \sigma N\}$

THEOREM 1 : *(Unification theorem [Rob65]) If two terms M and N are unifiable then they possess a most general unifier (mgu) σ such that: 1) $\sigma \in U(M,N)$, 2) $\forall \rho \in U(M,N)\ \sigma \leq \rho$. Most general unifiers of two terms are equivalent modulo $\equiv$.*

For example, let $M = f(x, g(y,x))$ and $N = f(h(y), g(u, h(u)))$, the most general unifier of M and N is $\{x \leftarrow h(y), u \leftarrow y\}$.

The proof of the theorem is based on an unification algorithm:
INPUT: Two terms M and N
OUTPUT: unifiable? If yes produce a mgu.
We describe first the *non-deterministic unification algorithm of Herbrand* [Her30], which basically simplifies a system of equations Γ, *i.e.* a multiset of pairs of terms, with the following three rules :

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 $(v,T) \cup \Gamma \Longrightarrow (v,T) \cup \Gamma\sigma$ if $v \in V(\Gamma)$, $v \notin V(T)$, $\sigma = \{v \leftarrow T\}$
(the same rule applies to $(T,v) \cup \Gamma$).

A pair (v,T) is in *solved form in* Γ if v has no other occurrence in Γ. A system Γ is in solved form if all its equations are in solved form. A variable is in solved form in Γ if it has a unique occurrence in Γ in a pair in solved form. The purpose of the test $v \in V(\Gamma)$ in the third rule is to apply that rule only to equations which are not in solved form. On the other hand the test $v \notin V(T)$ is a fundamental operation called the *occur check*. It eliminates equations of the form $v = T[v]$ which have no solution on finite terms.

THEOREM 2 : **(Soundness)** *If* $\Gamma \Longrightarrow^* \Gamma' = \{(v_1,T_1),...,(v_n,T_n)\}$ *is in solved form then* $\sigma_{\Gamma'} = \{v_1 \leftarrow T_1,...,v_n \leftarrow T_n\}$ *is an idempotent mgu of* Γ.

PROOF : By induction on the length of transformation sequences, and by remarking that if $\Gamma \Longrightarrow \Gamma'$ then $U(\Gamma) = U(\Gamma')$. ∎

THEOREM 3 : **(Termination and Completeness)** *Any sequence of transformations terminates. Furthermore if* Γ *is unifiable, let* $\theta \in U(\Gamma)$, *then* $\Gamma \Longrightarrow^* \Gamma'$ *with* Γ' *in solved form and* $\sigma_{\Gamma'} \leq \theta$.

PROOF : The proof that every sequence of transformations terminates is based on a complexity measure, $c(\Gamma) = (n_v, s)$, where n_v is the number of variables which are not in solved form, and s is the sum of the size of the terms in Γ. By considering the lexicographic ordering on (n_v, s) it is easy to check that each transformation decreases strictly the complexity of the system.

Completeness is an easy consequence of the fact that the transformations are sound, and if Γ is unifiable and no rule applies to Γ' then Γ' is in solved form. ∎

Robinson's algorithm is obtained from Herbrand's non-deterministic algorithm by representing the system Γ by a stack and by transforming Γ always with the rule which applies to the equation at the top of the stack (ignoring equations in solved form). If no rule applies the algorithm stops with a failure, due either to an *occur check*, or to a *clash* if **Dec** does not apply. Under this control, terms are traversed in preorder from left to right.

Robinson's algorithm takes exponential time in the following well-known example:
$M = f(v_0, v_1, ..., v_{n-1}, v_0)$, $N = f(g(v_1, v_1), g(v_2, v_2), ..., g(v_n, v_n), v_0)$. Each variable v_i, $i < n$, is substituted by a term of size $2^{n-i+1} - 1$, the unification of the last argument v_0 takes $2^{n+1} - 1$ comparisons. On the other hand if V is *finite* with cardinality v, the worst-case time complexity of Robinson's algorithm is a polynomial of order v. On M and $N = f(g(v_1, ..., v_1), ..., g(v_{v-1}, ..., v_{v-1}), v_0)$ Robinson's algorithm needs $O(n^v)$ comparisons.

DEFINITION We say that a pair of trees is *consistent* if it is reducible by rules **Dec** and **Triv** to a system of pairs that are either in solved form or reducible by rule **Var**. Inconsistent pairs of trees fail to unify before any substitution from **Var** is applied, that is either by a clash in the first decompositions, or by a the occurrence of a variable leaf in the corresponding subterm.

In the next section we show that family of unifiable pairs of trees is exponentially negligible. For that reason, we are interested also in a variant of Herbrand-Robinson's algorithm in which substitutions in Γ (with rule **Var**) are postponed until all the initial trees have been decomposed and checked for a clash or a direct occurrence. Let us call this variant Robinson's algorithm with *substitution delaying* (RD). On inconsistent pairs of binary trees we have four exclusive cases of failure: 1) clash at top level, 2) direct occurrence of a variable, 3) decomposition at top level with failure in the first arguments, or 4) failure in the second arguments if the first arguments are consistent. Thus the cost formula are:
$U_{RD}(c,c') = U_{RD}(c,T) = U_{RD}(T,c) = U_{RD}(f(T_1,T_2), g(T_1',T_2')) = 1$ (clash)
$U_{RD}(v,T) = U_{RD}(T,v) = occ(T)$ (cost of the occur check)
$U_{RD}(f(T_1,T_2), f(T_1',T_2')) = 1 + U_{RD}(T_1,T_1')$
if (T_1,T_1') is inconsistent,
$U_{RD}(f(T_1,T_2), f(T_1',T_2')) = 1 + 2(|T_1| + |T_1'| + 1) + U_{RD}(T_2,T_2')$,
if (T_1,T_1') is consistent;
where v (resp. c, T) stands for a variable (resp. a constant, a composed term), and $|T|$ denotes the number of internal nodes of T (the total number of nodes of T, internal and external, is thus $2|T| + 1$). In section

IV, we deduce from these formula that the average cost of Robinson's algorithm with substitution delaying on arbitrary pairs of terms tends to a constant c_{RD}.

2.1.2 Linear unification algorithms

Linear unification algorithms are based on a more compact representation of substitutions in *triangular form.*

PROPOSITION : *Any idempotent substitution can be represented by a substitution in triangular form :* $[x_1 \leftarrow T_1, ..., x_n \leftarrow T_n]$ *where* $\forall i,\ 1 \leq i \leq n\ \forall j \leq i\ x_j \notin V(T_i)$.

For instance the exponential size mgu of the previous example can be represented in a linear size triangular form: $[v_o \leftarrow g(v_1, v_1), ..., v_{n-1} \leftarrow g(v_n, v_n)]$. Triangular forms correspond also to the representation of substitutions by indirections on the variable nodes of a directed acyclic graph (dag) representing the terms to unify. Sharing of common subtrees is possible though the indirections on variables after substitution. Therefore in order to obtain a linear unification algorithm we can restate the unification problem as:
INPUT: Two terms M and N (or a *dag* with 2 particular nodes M and N)
OUTPUT: Are M and N unifiable? If yes produce a mgu in triangular form (or a representation of the dag after unification)

The idea of Martelli and Montanari's algorithm as well as Paterson and Wegman's linear algorithm is to apply the rule **Var** only to a variable which has no other occurrence in the system. Such a variable must exist if the system is unifiable. In this way the occur check is built-in and substitutions are eliminated. Rules **Dec** and **Triv** are kept unchanged while **Var** is replaced by two rules:

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

Select $\{(v, T_1), ..., (v, T_n)\} \cup \Gamma \Longrightarrow \{(v, C), (T_1, T_2), ..., (T_1, T_n)\} \cup \Gamma$
if $n \geq 2$, $v \notin V(\Gamma)$, and for $1 \leq i \leq n$, $v \notin V(T_i)$, $T_i \notin V$,
where C is the common part of the terms $T_1, ..., T_n$
(the same rule applies to (T_i, v)).

We emphasize the point that in order to find solved forms of linear size it is necessary in the **Select** rule to retain for v the common part of the T_i's instead of one of them, be it the least. For instance with

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

the mgu in solved form is $[x_1 \leftarrow f(x_2, x_2), ..., x_{n-1} \leftarrow f(x_n, x_n), x_n \leftarrow f(a, a)]$, while not computing the common part for the x_i's leads to a solved form of quadratic size. Of course the actual computation of the common part in rule **Select** should be mixed with the decompositions of the T_i's by the **Dec** rule.

The usual control is ((**Dec** ∪ **Triv** ∪ **Merge**)* **Select**)*. That is the equations are fully decomposed before a set of pairs with a variable in common, called a *multiequation*, is selected. We call this non-deterministic algorithm *Herbrand's algorithm with oracle* (HO). The oracle determines when rule **Select** can be applied, that is when a variable v has no other occurrence in Γ.

The proof of correctness and completeness follows in all points the previous proof of Herbrand's algorithm. In particular termination is proved with the same complexity measure. The crux of the algorithm is the property due to [MM82] that if Γ is formed of pairs (v, T) not reducible by rule **Select**, then there is a cycle in the occur check relation in the terms of Γ, which implies that Γ in not unifiable.

On *inconsistent* pairs of terms Herbrand's algorithm with oracle has the same cost formula as U_{RD} except on pairs variable-term, since the occur check is replaced by the oracle which determines the existence or not of a multiequation to reduce. We have
$U_{HO}(v, T) = U_{HO}(T, v) = 1$ (no occur check, no selection)
In section IV, we show that the average cost of Herbrand's algorithm with oracle on inconsistent pairs of terms is a constant $c_{HO} \leq c_{RD}$.

Martelli and Montanari's algorithm and Paterson and Wegman's algorithm can be derived from Herbrand's algorithm with oracle, the difference being in the way the oracle is implemented. In Martelli and

Montanari's algorithm each variable possess a counter which indicates the number of its occurrences inside the terms of Γ. When the counter gets to 0 rule **Select** can be applied. In this way the worst-case time complexity is in $O(n + v\log v)$ where n is the size of the terms and v is the number of *distinct* variables in the terms [MM82]. The initialization of counters needs to traverse the initial terms in *linear time*. In order to keep a constant average case complexity on arbitrary pairs of terms, one can try to *mix* the initialization phase with the first decompositions, and when reaching a leaf in a term, continue to traverse the other term for initializing properly the counters. With this provision the cost formula on inconsistent pairs of terms of Martelli and Montanari's algorithm are the same as in Herbrand's algorithm with oracle except on pairs variable-term due to the initialization of counters.
$U_{MM}(v,T) = U_{MM}(T,v) = 2|T| + 1$

However we show in section IV, that even with this provision the average number of steps on arbitrary pairs of terms is still *linear* in the size of the initial terms. The mixing of decompositions (**Dec**) with compactifications (**Select**) as suggested in [MM82] to improve efficiency on non-unifying data does not change the theoretical average case complexity. The only way to restore a constant average case complexity on arbitrary pairs of trees is to add occur checks to the initialization of counters for variables after the computation of the first frontier.

In Paterson and Wegman's algorithm the oracle is implemented by a judicious bottom-up traversal of the dag. The worst-case complexity is in $O(n)$, however whether its average case complexity on arbitrary pairs of terms is a constant or not is unknown.

2.2. Binary tree model

From now on, we consider binary trees $\mathcal{B}$ built with l binary symbols, v variables and c constants.

Example : With f and g two symbols with arity 2, a constant a and x and y two variables, we can form the term $T = f(x\ ,\ g(a\ ,\ y))$

We define the size of a term as the number of its internal nodes of its usual tree representation.

Example : We have here $|T| = 2$.

To make it clearer, we shall denote the families of pairs of trees with a *double capital letter*. Thus $\mathcal{BB}$ denotes the family of pairs of arbitrary trees of $\mathcal{B}$.

Remark : All the following calculations and results are detailed in [Alb90].

III. Characteristic families of pairs of trees

3.1. Definitions

Our aim is to count the number of unifiable pairs of trees and then to analyze the different causes of failure to unification.

In order to count the family $\mathcal{UU}$ of unifiable pairs of trees, we shall *bound* it with, as a lower bound, a family of clearly unifiable pairs of trees : family $\mathcal{VB}$ recursively defined in the following way :

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

and with, as an upper bound, the family of consistent pairs of trees $\mathcal{EE}$. In $\mathcal{EE}$, there are neither direct clashes nor direct occurrences. The failure causes can come but from clashes or occurrences due to substitutions. This family can be defined as follows:

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

Example : $f\ (\ f(x,y)\ ,\ z\ )$ and $f\ (\ f(f(y,z),f(z,y))\ ,\ f(w,w)\ )$ are a non consistent pair of trees because of the direct occurrence $y \leftarrow f(z,y)$

whereas $f\ (\ f(x,y)\ ,\ z\ )$ and $f\ (\ f(f(y,z),f(z,x))\ ,\ f(w,w)\ )$ are a consistent but non-unifiable pair of trees because of the indirect occurrence due to substitution $\sigma : x \leftarrow f(y,z)$.

Thus we get:

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

We shall prove that families $\mathcal{EE}$ and $\mathcal{VB}$ have the same order of magnitude. This gives the exact asymptotical order of magnitude for $\mathcal{UU}$. Moreover, we shall see that the constants of bound are close to (the limit of their ratio tends to 1), we thus have a very correct numerical bound of the number of unifiable pairs of trees.

Now we study the causes of failure for unification. We recursively define the complementary family $\mathcal{FF}$ of inconsistent pairs of trees as:

$$\begin{aligned}
\mathcal{FF} &= (c,c') + (v,T) + (T,v) && \text{with } v \in T \text{ and } |T| \geq 1\\
&+ (c,T) + (T,c) && \text{with } T \in \mathcal{B} \text{ and } |T| \geq 1\\
&+ (f(T1,T2), g(T'1,T'2)) && \text{with } T1,T2,T'1,T'2 \in \mathcal{B}\\
&+ (f(T1,T2), f(T'1,T'2)) && \text{with } (T1,T'1) \in \mathcal{FF} \text{ and } (T2,T'2) \in \mathcal{BB}\\
&+ (f(T1,T2), f(T'1,T'2)) && \text{with } (T1,T'1) \in \mathcal{EE} \text{ and } (T2,T'2) \in \mathcal{FF}
\end{aligned}$$

The failures in $\mathcal{FF}$ are due to direct left occurrences and clashes *i.e.* the first cause of failure detected by an algorithm using a preorder traversal for trees is either an occurrence or a clash. In order to clarify the causes of failure for unification we shall distinguish the inconsistent pairs of trees because of a direct left occurrence $\mathcal{FO}$ and because of a direct left clash $\mathcal{FC}$ (note that a pair of trees may fail for both reasons, the detection of the first cause of failure depending on the algorithm that is used).

We thus define

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

We have:

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

In $\mathcal{FO}$, there is no direct clash and in $\mathcal{FC}$ there is no direct occurrence but, since there may be indirect failures in $\mathcal{EE}$ and $\mathcal{BB}$, there may be indirect clashes and occurrences in these families. Thus to get a more detailed study, we shall introduce the family of pairs of trees which *do not fail because of a direct clash*: $\mathcal{NC}$. This family remains of the same order of magnitude as $\mathcal{BB}$ and thus its complementary in $\mathcal{BB}$ is an upper bound for the set of pairs of trees that fail because of a clash. We define recursively this family as follows:

$$\begin{aligned}
\mathcal{NC} &= (v,v') + (c,c) + (c,v) + (v,c)\\
&+(v,T) + (T,v) && \text{with } |T| \geq 1\\
&+ (f(T1,T2), f(T'1,T'2)) && \text{with } (T1,T'1) \text{ and } (T2,T'2) \in \mathcal{NC}
\end{aligned}$$

3.2. Algebraic analysis

For each of the previous families, we determine the number x_n of trees T or of pairs of trees (T_1,T_2) of size n (we define $|(T_1,T_2)| = |T_1| + |T_2|$). The generating function corresponding to family $\mathcal{X}$ is given by

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

(See [FV87] for the theory and use of generating functions).

From the definitions of the previous families we get the following equations: $B(z) = (v+c) + lzB(z)^2$ thus it follows that

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

The generating function of pairs of arbitrary terms is $BB(z) = B(z)^2$.

We introduce some useful series for the following computations; thus the generating function of trees of size at least 1 is $B^+(z) := lzB(z)^2$. The generating function of trees without a given variable is $B_{v-1}(z) = v - 1 + c + lzB_{v-1}(z)$. And the generating function of trees of size at least 1 and without a given variable is $B^+_{v-1}(z) = lzB_{v-1}(z)^2$.
We also define the trees with occurrence of a given variable $C(z) = 1 + lzC(z)B(z) + lzB_{v-1}(z)C(z)$ and the trees with the occurrence of a given variable and of size at least 1 $C^+(z) := lzC(z)B(z) + lzB_{v-1}(z)C(z)$.

The generating function of the family of clearly unifiable pairs of trees verify $VB(z) = v^2 + c + 2vc + 2vB^+_{v-1}(z)$.

Finally, we can obtain the generating function for family of consistent pairs of trees $\mathcal{EE}$ that is $EE(z) = v^2 + c + 2vc + 2vB^+_{v-1}(z) + lz^2EE(z)^2$ or

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

We also have the equations for the generating functions for $\mathcal{FF}$, $\mathcal{FO}$, $\mathcal{FC}$ and $\mathcal{NC}$ (the explicit expressions are too long to be written):

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

We easily verify that

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

3.3. Asymptotic analysis

We use singularity analysis, we know that

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

and transfert lemmas (see [FV87]).

The singularity with the smallest modulus of generating functions $BB(z)$, $FF(z)$, $FO(z)$, $FC(z)$ and $NC(z)$ is $z = \frac{1}{4l(v+c)}$. On the other hand, the singularity with the smallest modulus of generating functions $EE(z)$, $VB(z)$ (and therefore $UU(z)$) is $z = \frac{1}{4l(v+c-1)}$. We thus obtain

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

($EE(z)$ is exponentially negligible with respect to $BB(z)$ and $FF(z)$).

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

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

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

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

And

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

3.4. Results

3.4.1 Unifiable pairs of trees

We have succeeded to bound on both sides $[z^n]UU(z)$:

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

We thus have the correct asymptotic order of magnitude for $[z^n]UU(z)$, and moreover, numerically, we note that $C_2/C_1 \approx 1$! We have $1 \leq C_2/C_1 \leq 1.2$ and even

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

The bound is all the better as l is large (for 3 or 4 already : for $c=2$, $v=2$, $l=3$ we have $C_2/C_1 = 1.10$, for $c=3$, $v=3$, $l=6$ we have $C_2/C_1 = 1.05$, for $c=10$, $v=4$, $l=10$ we have $C_2/C_1 = 1.01$).

Summing up, we can state the following theorem:

THEOREM 4 : *If we consider a uniform distribution on pairs of binary trees with l internal symbols, v variables and c constants, then*

- *the number of unifiable pairs of trees of total size n is:*

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

- *the ratio between the number of unifiable pairs of trees and the number of pairs of arbitrary trees is:*

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

(Θ: "of the same asymptotic order as").

which means that for sufficiently large n almost all pairs of trees are non-unifiable and that we have a good asymptotical and numerical bound for the number of unifiable pairs of trees.

3.4.2 The causes of failure

We consider the ratio

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

and represent its variation in the following three-dimensional picture:

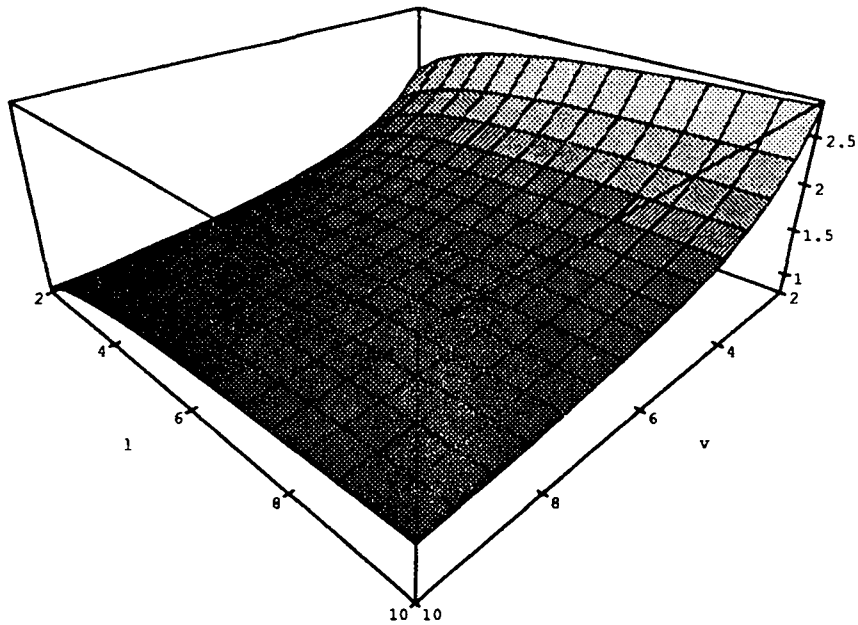

Figure 1 : Variation of $\tau_{c/o}$ for $c = 2$, $2 \leq v \leq 10$ and $2 \leq l \leq 10$

Figure 1 shows off that numerically we always have more clashes than occurrences and this proportion increases with c and l. We can note that even when v is small, the number of clashes still remains larger than the number of occurrences : this phenomenon is due to the presence of constants at the leaves of the terms because of the uniform distribution model.

Finally, in order to study with precision the clash failures, we can define the ratios:

$$\tau_c = \frac{|FC|}{|BB|} = \frac{|FC|}{|FF|}$$

and

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

This will enable us to bound on both sides the pairs of trees that fail with a direct clash. We represent on the following drawing the proportions with respect to $\mathcal{BB}$ of the families $\mathcal{FC}$ and $\mathcal{BB} \setminus \mathcal{NC}$.

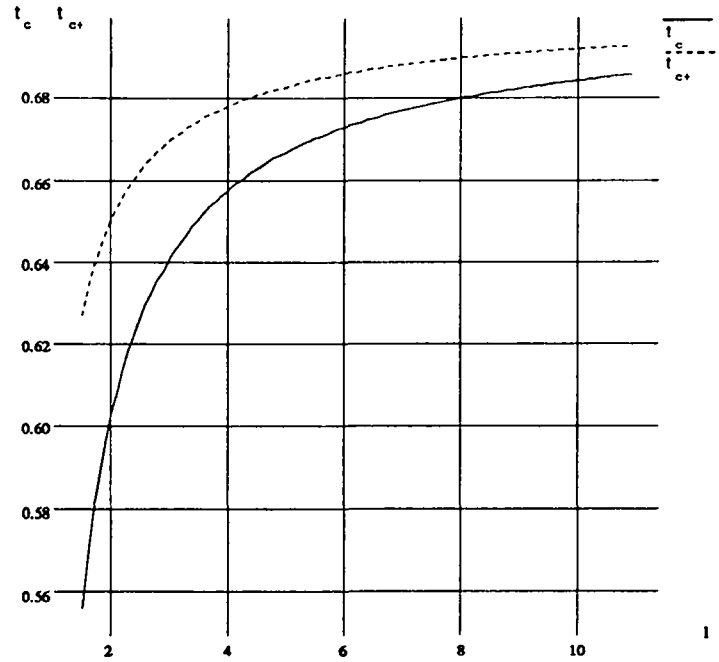

Figure 2 : Variation of τ_c and τ_{c+} for $c = 2$, $v = 3$ and $2 \leq l \leq 10$

The proportion of pairs of trees that fail with a direct clash is located between the two curves of figure 2. We notice that we get a good bound for the failures due to a clash and this bound is all the better as l is large. We can verify that

$$\lim_{l \to \infty} \tau_c = \frac{v + 2c}{2(v + c)} = \lim_{l \to \infty} \tau_{c+}$$

Summing up, we have the following theorem:

THEOREM 5 : *The families of pairs of trees which are non unifiable because of a direct left fail $\mathcal{FF}$, $\mathcal{FO}$ and $\mathcal{FC}$ and the family of pairs of trees without any direct clash $\mathcal{NC}$ have the same order of magnitude as the family of pairs of arbitrary trees $\mathcal{BB}$. In this model, where the number of variables is finite, we have a good estimation of the failures due to a clash. In this model, the occurrence is not negligible with respect to the clash but remains numerically less important.*

We can interpret these combinatorial results: the asymptotic analysis leads us to consider large binary trees thus with many leaves. This fact yields to the repetition of identical variables and therefore to a non negligible proportion of occurrences. In PROLOG (where there is no occur-check and where one unifies many small terms, see [SS86]), we can have potentially an infinity of variables (*i.e.* the number of variables can vary with the size n of the pair of terms) and that cannot of course be modelized with a finite v.

Thus it is interesting to reconsider the phenomenon of occurrence (the clash is, as we have seen, well studied in the model with bounded v). In section V, we shall modelize terms with *patterns* formed with an infinite number of variables up to variable renaming. We shall then analyze the influence of this model on the probability of occurrence. Before that, we conclude the study of the model of binary trees with a finite number of variables by giving the average cost of unification algorithms in this model.

IV. Average cost of unification

In this section, we present the average cost of the different unification algorithms detailed in section II. We first define the notion of the average cost of an algorithm on a family of input data and then, using the

results of previous section, we prove that the average cost of an unification algorithm on family of pairs of trees $\mathcal{BB}$ is the same as the average cost on family $\mathcal{FF}$ of pairs of trees that fail directly. Then we obtain that the average cost of the occur-check is constant. In chapter 4.2, we compute the precise average cost of the Robinson's algorithm with substitution delaying that is constant c_{RD}. In chapter 4.3, we compute the average cost of the Herbrand's unification algorithm with oracle on the **Select** rule, that is constant c_{HO} (that modelizes also Martelli Montanari's algorithm *after the linear initialization phase of counters*). In chapter 4.4, we compute the average cost of Martelli Montanari's algorithm with the initialization of counters mixed with first decompositions (therefore eliminating the linear initialization phase) and we obtain that the average cost on arbitrary pairs of trees still remains linear. In chapter 4.5, we draw some conclusions about all this study.

4.1. Average cost

4.1.1 Definition

The average cost spent to unify two terms with total size n belonging to a family of pairs of terms $\mathcal{XX}$ is defined with:

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

where $\tau(T_1, T_2)$ denotes the total number of steps during the unification process of T_1 and T_2 and xx_n denotes the number of pairs of terms of family $\mathcal{XX}$ of total size n.

4.1.2 The cost on $\mathcal{BB}$ is equal to the cost on $\mathcal{FF}$

Indeed, we have

LEMMA : *The average cost of an unification algorithm on arbitrary pairs of terms is the same as on non-unifiable pairs of trees because of a direct failure.*

It means that the average time of the algorithm is spent to detect direct failures.

PROOF : We know from theorem 4 that the number of pairs of trees in $\mathcal{EE}$ is exponentially negligible with respect to $\mathcal{BB}$. Moreover, the worst case complexity of any unification algorithm is polynomially bounded in a model where the number of variables v is finite (exactly $O(n^v)$). Consequently the main contribution to the average cost of unification on arbitrary pairs of trees is the contribution of direct failure. ∎

4.1.3 Average cost of the occur-check

The cost to check the occurrence of a variable v in a term T of $\mathcal{B}$ can recursively be defined as follows:
$occ(c) = occ(v) = 1$

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

If we denote $O(z) = \sum_{T \in \mathcal{B}} occ(T) z^{|T|}$ we can split this generating function $O(z) = O_1(z) + O_2(z)$ with $O_1(z) = \sum_{x \in T} occ_1(T) z^{|T|}$ and $O_2(z) = \sum_{x \notin T} occ_2(T) z^{|T|}$; occ_1 and occ_2 being recursively defined in the same way as occ.

We obtain the generating functions:

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

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

and asymptotically

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

Hence, we derive the constant average cost for the occur-check :

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

Remark : In the specific case where $l = v = c = 1$ of [DL89], we actually find again $k_o = 4 + 2\sqrt{2}$.

4.2. Average cost of Robinson's algorithm with substitution delaying

We use the recursive definition of the cost of the algorithm given in section II. Once more, we can express $U_{RD}(T,T')$ for $|T| + |T'| = n$ as the n^{th} coefficient of the generating function $U_{RD}(z)$ which verifies the following algebraic equation:

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

The singularity with smallest modulus of $U_{RD}(z)$ is again $z = \frac{1}{4l(v+c)}$. Asymptotic analysis yields to

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

whence

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

with K_1 and K_2 constants depending on l, v and c.

The coefficient $[z^n]U_{RD}(z)$ is of the same asymptotic order as $[z^n]FF(z) = [z^n]BB(z)$, we thus have:

THEOREM 6 : *The average cost of Robinson's algorithm with substitution delaying on arbitrary pairs of trees is a constant* c_{RD}.

The explicit expression of c_{RD} in its more "compacted form" obtained with the symbolic computation system Maple (see [CGG+88]) needs 54 lines ! We just note its variation with l, v and c.

- v and c remaining fixed, the leading term in c_{RD} behaves like $\sqrt{l}$,
- l and c remaining fixed, the leading term in c_{RD} behaves like $v^{3/2}$,
- l and v remaining fixed, the leading term in c_{RD} behaves like $\sqrt{c}$.

Numerically for $l = 2$, $v = 2$ and $c = 2$ we have $c_{RD} \simeq 28.38$.

4.3. Average cost of Herbrand's algorithm with oracle

The recursive expression $U_{HO}(T,T')$ of the average cost of the algorithm on a pair of trees $(T,T') \in \mathcal{FF}$ is identical to the one of $U_{RD}(T,T')$ with the only difference that:

$U_{HO}(v,T) = u_{HO}(T,v) = 1$ since the oracle determines the application or the non-application of **Select** rule.

Once more, we can express $U_{HO}(T,T')$ for $|T| + |T'| = n$ as the n^{th} coefficient of the generating function $U_{HO}(z)$ which verifies the following algebraic equation:

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

The singularity with smallest modulus of $U_{HO}(z)$ is again $z = \frac{1}{4l(v+c)}$. Asymptotic analysis yields to

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

whence

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

with K_3 and K_4 constants depending on l, v and c.

The coefficient $[z^n]U_{HO}(z)$ is of the same asymptotic order as $[z^n]FF(z) = [z^n]BB(z)$, we thus have:

THEOREM 7 : *The average cost of Herbrand's algorithm with oracle on arbitrary pairs of trees is a constant* c_{HO}.

Nota : Notice that $c_{HO} \leq c_{RD}$.

The explicit expression of constant c_{HO} is once more too long to be written. We shall just note its variation with l, v and c.

- v and c remaining fixed, the leading term in c_{HO} behaves like $\sqrt{l}$,
- l and c remaining fixed, the leading term in c_{HO} behaves like v,
- l and v remaining fixed, the leading term in c_{HO} behaves like $\sqrt{c}$.

Numerically for $l = 2$, $v = 2$ and $c = 2$ we have $c_{HO} \simeq 8.46$.

4.4. Average cost of Martelli Montanari's algorithm with initialization phase mixed with first decomposition

The recursive expression $U_{MM}(T,T')$ of the average cost of the algorithm on a pair of trees $(T,T') \in \mathcal{FF}$ is identical to the one of $U_{RD}(T,T')$ with the only difference that:

$U_{MM}(v,T) = u_{MM}(T,v) = 2|T|+1$ since the algorithm requires a complete tree traversal of T for the initialization of the counters. We shall prove that this variant of the classical Martelli Montanari's algorithm has still a linear average cost on the family of arbitrary pairs of trees.

Once more, we can express $U_{MM}(T,T')$ for $|T|+|T'| = n$ as the n^{th} coefficient of the generating function $U_{MM}(z)$ which verifies the following algebraic equation:

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

The singularity with smallest modulus of $U_{MM}(z)$ is again $z = \frac{1}{4l(v+c)}$. But this time, asymptotic analysis yields to

$$U_{MM}(z) = \frac{-16v(v+c)^2l^{3/2}}{2(v+c)l^{3/2} - (v+c)\sqrt{l} + \sqrt{X}} \frac{1}{\sqrt{1-4lz(v+c)}} + O(z - \frac{1}{4l(v+c)})$$

with $X = 4l(v+c)^2 - 3(v+c)^2 + 2c(v+c) + 4(v+c)^{3/2} - 4\sqrt{v+c}\,c - 2v$. Hence

$$[z^n]U_{MM}(z) = \frac{-16v(v+c)^2l^{3/2}}{2(v+c)l^{3/2} - (v+c)\sqrt{l} + \sqrt{X}} \frac{(4l(v+c))^n}{\sqrt{\pi}} n^{-1/2}\left(1+O\left(\frac{1}{n}\right)\right)$$

The coefficient $[z^n]U_{MM}(z)$ has not the same polynomial power of n as $[z^n]FF(z) = [z^n]BB(z)$, thus we obtain :

THEOREM 8 : *The average cost of Martelli Monatanari algorithm with initialization phase mixed with first decomposition is linear equal to :*

$$c_{MM}n\left(1+O\left(\frac{1}{n}\right)\right) = \frac{4vl^{3/2}}{2(v+c)l^{3/2} - (v+c)\sqrt{l} + \sqrt{X}} n\left(1+O\left(\frac{1}{n}\right)\right)$$

with $X = 4l(v+c)^2 - 3(v+c)^2 + 2c(v+c) + 4(v+c)^{3/2} - 4\sqrt{v+c}\,c - 2v$.

We have the variation of c_{MM} in function of l, v and c :
v and c being fixed, the leading term in c_{MM} is $2v/(v+c)$,

l and c being fixed, the leading term in c_{MM} is $4l^{3/2}/(2l^{3/2}-\sqrt{l}+\sqrt{4l-3})$,
l and v being fixed, the leading term in c_{MM} is $4l^{3/2}v/((2l^{3/2}-\sqrt{l}+\sqrt{4l-1})c)$.

Numerically for $l=v=c=2$ we get $c_{MM}=0.83$.

4.5. Conclusions

We can draw some conclusions from the previous results on the average complexity of unification algorithms. In fact, clashes are not sufficient to lead to a constant average cost : it is necessary to use an efficient occur-check (*i.e.* without a total traversal of subterms). Then, the variant of Robinson's algorithm, that we called *with substitution delaying*, appears to be the most efficient unification algorithm on arbitrary pairs of trees with an uniform distribution since its average cost is constant and it does not require specific data structures. Finally, on unifiable pairs of trees, it is easy to realize that the average cost of any unification algorithm (without oracle of course), is *at least linear* since the algorithm requires a total traversal of T for family $\mathcal{VB}$.

V. Model with an infinity of variables

Note that all the following combinatorial computations can be performed with an assistant algorithms analyzer such as Lamda-Upsilon-Omega (see [FSZ89]).

5.1. Terms with an infinity of variables up to renaming

Our aim in this section is to study the influence of a model with a non-bounded number of variables on the occurrence phenomenon. In order to simplify the following computations, we thus shall consider terms without constant and with only one internal binary symbol denoted $*$. We modelize terms with *patterns* formed with a binary tree structure and with variables at the leaves with equality up to *variable renaming*. For example, there are only two patterns of size 1 (the size of of pattern is still the number of its internal nodes):

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

it illustrates the fact that the leaves of these elementary patterns are either equal or different.

We have the following combinatorial theorem:

THEOREM 9 : *The number of patterns of size n is:*

$$C_n B_{n+1}$$

where C_n is the n^{th} Catalan number and B_n the n^{th} Bell number.

We know that $C_n = \frac{1}{n+1}\binom{2n}{n}$ and $B_n = \sum_{p=0}^{n-1}\binom{n-1}{pB_p}$.

PROOF : C_n is the number of binary trees with n internal nodes and B_n is usually defined as the number of set partitions. There is a bijective correspondence between set partitions and lists of variables up to renaming. We can associate to each variable the set of the places it takes in the list. For example, the corresponding set partition of the list (x,y,y,z,t,x,z,y) is $\{\{1,6\},\{2,3,8\},\{4,7\},\{5\}\}$. ∎

We now determine the average number of *distinct variables* in a pattern of size n. Resulting from the bijective correspondence above, this number is also the average number of classes in a set partition. Thus we easily get:

THEOREM 10 : *The average number of distinct variables in a pattern with size n is:*

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

This theorem shows off that $V_n \longrightarrow \infty$ with n, and that phenomenon cannot be modelized with a finite number of variables v. This fundamental result justifies the present study of this section.

5.2. Influence on the occurrence

Now, we reconsider the probability of occurrence in a model where v may increase with the size of the term. We consider basic pairs of trees (v, T) with T a binary tree of size n and therefore with $n+1$ variable leaves belonging to a set of v available variables. The total number of such pairs is $vC_n v^{n+1}$ and the total number of pairs for which there is no occurrence is $vC_n(v-1)^{n+1}$, the ratio is thus $(1-\frac{1}{v})^{n+1}$ and we get :

THEOREM 11 : *The probability for a pair (v, T) of total size n to be non unifiable because of an occurrence of variable v in T is:*

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

- Thus, if v remains constant, the probability of occurrence when $n \longrightarrow \infty$ tends to 1. This is the case of the previous model and this expresses again the importance of the occurr-check we obtained.
- However, the same phenomenon appears with an infinity of variables in the case where v is of the same order as $n/\ln(n)$, which corresponds to the average number of distinct variables for the previous terms with variable renaming. In that case, the probability of occurrence behaves like $P = 1 - \frac{1}{n}$.

Remark : We can derive this last result with a direct combinatorial computation considering pairs of patterns up to variable renaming (see [Alb90]).

- When the number of available variables v is of the same order as the size of the term n, we get $P(n, n) = 1 - (1 - \frac{1}{n})^{n+1} \longrightarrow 1 - \frac{1}{e}$.
- Finally, as soon as v is of the order of $n^{1+\alpha}$ with $\alpha > 0$, the probability of occurrence $P(n, v)$ tends to 0.

VI. Conclusion

Under a uniform distribution law for binary trees formed with a *finite* set of variables V, the family of unifiable pairs of trees is exponentially negligible with respect to family of arbitrary pairs of trees. The different causes of failure have been analyzed, and the analysis of the occur check has been extended to the case of an infinite set of variables. We proved that the probability that a variable occurs in a term of size n is $1 - (1 - \frac{1}{v})^{n+1}$ where v is the number of variables, and therefore if v is of the order of the average number of distinct variables in a term of size n, or if v is a constant as in section III, the probability that a variable occurs in a term tends to 1 when the size of the term tends to the infinity. On the other hand as soon as the number of variables is superlinear ($n^{1+\alpha}$ with $\alpha > 0$) that probability tends to 0.

We have shown that when V is finite, Robinson's algorithm with substitution delaying has a *constant average cost* over random pairs of trees. On the other hand over random pairs of trees various variants of Martelli and Montanari's algorithm all have a *linear average cost* in this model. 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. The average case complexity of Paterson and Wegman's algorithm on random pairs of trees as well as the average case complexity of Robinson's algorithm in a completely general framework (bounding it by a constant over non-unifiable terms, and by a polynomial over unifiable terms) remain open.

The asymptotical analysis is performed by considering arbitrary large terms. This can be criticized since in practice not very large terms are unified, but instead large *sequences* of small terms are unified. The problem of unifying an in-line sequence of terms is closer to common practice. Note that due to the composition of substitutions the use of a linear unification algorithm does not result in a linear algorithm for unifying sequences of terms. The worst-case time complexity of the algorithm proposed in [MM86] for executing a sequence of unify-deunify operations is $O(n \ \ln(\ln(n)))$. One approach for the average case complexity analysis of such sequences is to consider a forest of pairs of binary trees and perform the asymptotical analysis on the size of the forest, as in [AF88].

VII. Bibliography

[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.

[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.

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

[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

[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.

[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.

[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).

[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.

[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.

[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.

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

[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

ISSN 0249-6399