

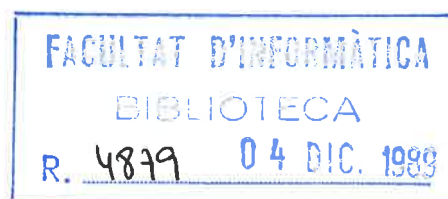
• J46000842A
còpia 1



**Average case analysis
of Robinson's unification algorithm
with two different variables**

R. Casas
J. Díaz
J.M. Steyaert

Report LSI-88-8



Abstract: This paper deals with the average complexity of Robinson's unification algorithm, for a simple case of unification; trees with all internal nodes of the same type, and two kinds of leaves.

Resum: En aquest treball es presenta l'anàlisi mitja del algorisme de unificació de Robinson, quan es consideren arbres binaris amb dues tipus de fulles.

Average case analysis of Robinson's unification algorithm with two different variables

R.Casas^{1*}, J.Diaz¹, J.M. Steyaert²

¹ Dpt. LSI, Universitat Politècnica Catalunya. Barcelona

² Ecole Polytechnique, Palaiseau

We compute the average complexity of Robinson's unification algorithm on the case of binary functions with two variables. When restricting the input to unifiable instances, the resulting complexity is linear on the size of the input.

Keywords: *Unification, Average Analysis*

1.- Introduction

Unification of first order terms is a problem which arises in several fields of computer science, for instance it is fundamental in symbolic computation and in all kind of procedures which make use of the resolution principle.

The first known algorithm was due to Herbrand [Her,30], and was rediscovered by Robinson [Ro,65]. Since then, it has received a lot of criticism due to its exponential worst-case complexity, and several other algorithms have been proposed, in particular [MM,82]. The problem is known to be in the class PTIME-complete, which implies that it will be very unlikely to find fast parallel algorithms to solve this problem. This reason adds interest to the estimation of the average case analysis of this algorithm.

In this paper, we formally study the average complexity of Robinson's algorithm for a very simple case of unification. We shall work with trees which only have two differentiated types of leaves; moreover we shall assume that the internal nodes are all of the same type, and all them have arity 2. So from now on, when writing binary tree we mean a binary tree with two differentiated types of leaves x and y .

2.- Basic definitions and algorithm

Given a pair (T_1, T_2) of binary trees, each one with two types of leaves, to unify these two trees is equivalent to find the smallest substitution σ of leaves by trees, which allows us to obtain $\sigma(T_1) = \sigma(T_2)$. For the particular case of trees we are dealing with, it is immediate that if such a substitution exists, it must be of the form $(x \leftarrow t_y)$ or $(y \leftarrow t_x)$, where t_y and t_x denote trees in which all the leaves are of one type (either y or x), and where the substitution consists in replacing the tree t_y (respectively t_x) for each leaf of type x (respectively y) in trees T_1 and T_2 .

We can carry out this unification by implementing the following algorithm, where the input is a pair of trees (T_1, T_2) ; *unifiable* is a boolean variable that takes value true if and only if the pair (T_1, T_2) is unifiable and σ is the substitution which allows the unification when possible. The algorithm starts with the initial values $\sigma := \emptyset$ and *unifiable* := true.

* Work supported by CIRIT-EE86/2-14

```

procedure UNIFY ( $T_1, T_2, \sigma, \text{unifiable}$ )
begin
  if  $T_1$  or  $T_2$  is a leaf
  then begin
    let  $l$  be this leaf and  $t$  the other tree;
    if  $t \neq l$  then if  $\sigma = \emptyset$ 
      then if  $l$  occurs in  $t$ 
        then  $\text{unifiable} := \text{false}$ 
        else  $\sigma := (l \leftarrow t)$ 
      else if  $\sigma \neq (l \leftarrow t)$  then  $\text{unifiable} := \text{false}$ 
    end
  else begin
    UNIFY (left-son ( $T_1$ ), left-son ( $T_2$ ),  $\sigma, \text{unifiable}$ );
    if  $\text{unifiable}$  then UNIFY (right-son ( $T_1$ ), right-son ( $T_2$ ),  $\sigma, \text{unifiable}$ )
  end
end.

```

This algorithm is the original from Robinson, adapted to our case of only two different leaves x and y .

Our interest is to evaluate the average complexity of this algorithm. It is easy to prove that if the computation of this average complexity is done on the set of all possible pairs of trees, the result will tend asymptotically to a constant, which is independent on the size of the trees. This is due to the fact that the ratio between the number of pairs of unifiable and non-unifiable trees tends to zero, and the average complexity of the algorithm tends to a constant over pairs of non-unifiable trees. So we shall restrict the set of possible inputs, to the set \mathcal{U} of pairs of binary trees which are unifiable, since failure is usually detected after a few comparisons. Formally, if $\mathcal{B}_{x,y}$ denotes the class of all binary trees with two different kinds of leaves x and y , then

$$\mathcal{U} = \{(T_1, T_2) \in \mathcal{B}_{x,y} \times \mathcal{B}_{x,y} \mid T_1, T_2 \text{ are unifiable}\}$$

The technique to evaluate this algorithm will be the use of generating functions as a tool to count trees with certain properties, see for example [FV,88].

Let $U(z) = \sum_{(T_1, T_2) \in \mathcal{U}} z^{|T_1|+|T_2|}$ be the generating function of \mathcal{U} , where $|T|$ denotes the size of the tree T , where the size of a tree is defined as the number of internal nodes. If $\tau(T_1, T_2)$ denotes the cost of applying the algorithm to T_1 and T_2 , let

$$A(z) = \sum_{(T_1, T_2) \in \mathcal{U}} \tau(T_1, T_2) \cdot z^{|T_1|+|T_2|} \quad (1)$$

be the generating function of the total cost of applying the algorithm to all possible couple of unifiable trees.

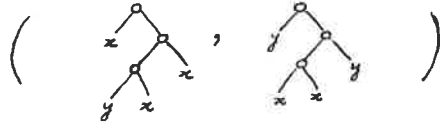
If for any generating function $f(z)$, $[z^n]f(z)$ denotes the coefficient of z^n in f , our goal is to evaluate the average complexity given by:

$$\tilde{\tau}(n) = \frac{[z^n]A(z)}{[z^n]U(z)} \quad (2)$$

In general we shall denote by capital script letters, the set to be counted, and by roman capital letters their corresponding generating functions. Let us start by evaluating the total number of unifiable binary trees with n nodes.

3.- Counting the number of unifiable trees

Let \mathcal{B}_x and \mathcal{B}_y denote respectively the binary trees with all leaves x and all leaves y . Then the possible substitutions ($l \leftarrow t$) can be one of the following types: ($x \leftarrow t$), with $t \in \mathcal{B}_y$, or ($y \leftarrow t$), with $t \in \mathcal{B}_x$. We could classify the pairs in \mathcal{U} by classes of substitutions, but we should take care that the substitutions ($x \leftarrow y$) and ($y \leftarrow x$) are equivalent. Therefore we have to dissociate this class from the others. The pairs of trees in this class have identical internal structure and they differ (if they do) in the leaves. Each one of these pairs is the result of applying to the same tree $T \in \mathcal{B}_l$ (trees with only one type l of leaves) two mappings μ_1 and μ_2 which are defined as $(|T| + 1)$ -tuples from $\{x, y\}$. For example the pair:



is the result of mapping $\mu_1 = (x, y, x, x)$ and $\mu_2 = (y, x, x, y)$ on the tree T :



Therefore if we denote by \mathcal{M}_T the set of all these mappings μ on tree T , we could decompose the family \mathcal{U} in the following way:

$$\mathcal{U} = \left(\bigcup_{t \in (\mathcal{B}_x - \{x\})} \mathcal{U}_t \right) + \left(\bigcup_{t \in (\mathcal{B}_y - \{y\})} \mathcal{U}_t \right) + \left(\bigcup_{T \in \mathcal{B}_t} \{(\mu_1(T), \mu_2(T)) | \mu_1, \mu_2 \in \mathcal{M}_T\} \right), \quad (3)$$

where \mathcal{U}_t denotes the family of pairs of unifiable binary trees, which must be unified by the substitution $(l \leftarrow t)$, where l is either x or y , and t is a binary tree which does not contain x or y respectively. Equation (3) can be translated in terms of generating functions as follows,

$$U(z) = 2 \sum_{|t| > 0} U_t(z) + \sum_{T \in \mathcal{B}_t} \left(\sum_{\mu_1, \mu_2} z^{2|T|} \right), \quad (4)$$

where $U_t(z) = \sum_{(T_1, T_2) \in \mathcal{U}_t} z^{|T_1| + |T_2|}$. It is classical to derive

$$\sum_{T \in \mathcal{B}_t} \left(\sum_{\mu_1, \mu_2} z^{2|T|} \right) = 4B(4z^2) = -\frac{(1 - 16z^2)^{1/2}}{2z^2} + \frac{1}{2z^2}$$

$B(z)$ being the well known generating function of binary trees, [FV,88]:

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

where C_n are the Catalan numbers $\frac{1}{n+1} \binom{2n}{n}$.

The family \mathcal{U}_t can be expressed by the following recursive disjoint union

$$\begin{aligned} \mathcal{U}_t = (l, t) + (t, l) + & \left\{ (T_1 \wedge T, T_2 \wedge T) \mid (T_1, T_2) \in \mathcal{U}_t \text{ and } T \in \mathcal{B}_{x,y} \right\} + \\ & \left\{ (T \wedge T_1, T \wedge T_2) \mid (T_1, T_2) \in \mathcal{U}_t \text{ and } T \in \mathcal{B}_{x,y} \right\} + \\ & \left\{ (T_1 \wedge T'_1, T_2 \wedge T'_2) \mid (T_1, T_2), (T'_1, T'_2) \in \mathcal{U}_t \right\} \end{aligned} \quad (5)$$

where l stands for x or y .

This equation can be translated in terms of generating functions. We obtain, for $|t| > 0$,

$$U_t(z) = \frac{\sqrt{1 - 8z^2} - \sqrt{1 - 8z^2 - 8z^{|t|+2}}}{2z^2}$$

so each one of the C_n trees of size $|t| = n$ gives the same generating function.

To do the calculation of $\sum_{|t| > 0} U_t(z)$ it is convenient to express $U_t(z)$ in the following way:

$$U_t(z) = \frac{2z^{|t|}}{\sqrt{1 - 8z^2}} + \sqrt{1 - 8z^2} \cdot f_{|t|}(z)$$

where it is easy to prove that $f_n(z)$ can be bounded as follows:

$$\forall n > 0, \forall z, |z| < 0.3 \Rightarrow |f_n(z)| < 20 \times 0.1^n$$

By the Weierstrass's double-series theorem [Ti,79] we can state that under these conditions the function

$$F(z) = \sqrt{1-8z^2} \times \sum_{n>0} C_n f_n(z)$$

is analytic at $|z| < 0.3$. Therefore we have

$$\sum_{|t|>0} U_t(z) = \frac{2}{\sqrt{1-8z^2}} \cdot \sum_{n>0} C_n z^n + F(z)$$

and we can express $U(z)$ in the form

$$U(z) = -\frac{1}{2z^2} \cdot (1-16z^2)^{1/2} - \frac{2}{z\sqrt{1-8z^2}} \cdot (1-4z)^{1/2} + \xi(z)$$

with $\xi(z)$ analytic in $|z| < 0.3$. Now we can use Darboux theorem [Hen,77] to obtain an asymptotical approximation to the n -th Taylor coefficient of $U(z)$.

Proposition 1. *The number of pairs of unifiable binary trees of global size n is given by the following expression:*

$$[z^n]U(z) \approx \begin{cases} \frac{4\sqrt{2}}{\sqrt{\pi}} \cdot 4^n \cdot n^{-\frac{3}{2}} \cdot (1 + O(\frac{1}{n})) & \text{if } n \text{ is odd;} \\ \frac{12\sqrt{2}}{\sqrt{\pi}} \cdot 4^n \cdot n^{-\frac{3}{2}} \cdot (1 + O(\frac{1}{n})) & \text{if } n \text{ is even.} \end{cases}$$

The difference between the two cases is due to the fact that in the even case 2/3 of the instances are pairs of trees with identical structure.

4.- Analysis of the algorithm

To get the average analysis of the algorithm, we should get some estimations on the numerator $[z^n]A(z)$ in (2).

Looking at the way the algorithm explores two trees T_1, T_2 , we can see that the algorithm will compare both trees node by node in preorder, until one of the following two situations arises:

- 1- one tree has a leaf where the other tree has a subtree,
- 2- both trees have a leaf.

In the first situation the algorithm saves the subtree, and each time it finds again the same situation it must compare the subtrees.

Moreover, the algorithm completely explores both trees, but doing it at the same time for the common part. So the time the algorithm spends to explore both trees can be expressed by

$$\tau(T_1, T_2) = \begin{cases} 2|T| + 1 & \text{if one is a leaf and the other is tree } T; \\ 1 + \tau(T_1^l, T_2^l) + \tau(T_1^r, T_2^r) & \text{otherwise} \end{cases}$$

where T^l and T^r respectively denote the left and right subtrees of T .

From this definition together with (3), we get

$$A(z) = 2 \sum_{|t|>0} A_t(z) + \sum_{T \in \mathcal{B}} (2|T| + 1) z^{2|T|} 2^{2|T|+2} \quad (6)$$

where

$$A_t(z) = \sum_{(T_1, T_2) \in \mathcal{U}_t} \tau(T_1, T_2) z^{|T_1|+|T_2|}$$

From the second term of (6), we immediately obtain

$$\sum_{T \in \mathcal{B}} (2|T| + 1) z^{2|T|} 2^{2|T|+2} = \frac{1}{2z^2} \cdot (1 - 16z^2)^{-1/2} - \frac{1}{2z^2} \quad (7)$$

On the other hand to compute $A_t(z)$ we start from (5) to obtain

$$A_t(z) = \frac{1}{2z^2} \left[\frac{8|t|z^{|t|+2} + 1}{\sqrt{1 - 8z^2 - 8z^{|t|+2}}} - \frac{1}{\sqrt{1 - 8z^2}} \right].$$

In a similar way to the previous section the main terms of this expression can be computed to get

$$A_t(z) = \frac{4|t|z^{|t|}}{\sqrt{1 - 8z^2}} + \frac{2z^{|t|}}{(1 - 8z^2)^{3/2}} + g_{|t|}(z) \quad (8)$$

where $g_n(z)$ can be bounded as follows:

$$\forall n > 0, \forall z, |z| < 0.3 \Rightarrow |g_n(z)| < 74 \times 0.14^n$$

and using again the Weierstrass's double-series theorem we can state that the function

$$G(z) = \sum_{n>0} C_n \cdot g_n(z)$$

is analytic in $|z| < 0.3$.

Finally from (7) and (8) we get

$$A(z) = \frac{1}{2z^2} \cdot (1 - 16z^2)^{-1/2} + \frac{64z^3 - 32z^2 + 2}{z(1 - 8z^2)^{3/2}} \cdot (1 - 4z)^{-1/2} + \varphi(z)$$

and we can realize that the first terms have their smallest singularities at $|z| = \frac{1}{4}$ whereas $\varphi(z)$ is analytic in $|z| < 0.3$.

Again we have the right conditions to apply Darboux theorem and get

$$[z^n]A(z) \approx \begin{cases} \frac{8\sqrt{2}}{\sqrt{\pi}} \cdot 4^n \cdot n^{-1/2} \cdot (1 + O(\frac{1}{n})) & \text{if } n \text{ is odd;} \\ \frac{16\sqrt{2}}{\sqrt{\pi}} \cdot 4^n \cdot n^{-1/2} \cdot (1 + O(\frac{1}{n})) & \text{if } n \text{ is even.} \end{cases}$$

This result together with the one obtained at the end of section 3, allows us to compute the average complexity of the algorithm as indicated in (2).

Theorem 1. *The average time of Robinson's algorithm on unifiable pairs of binary trees with global size n , is given by the following expression:*

$$\tilde{\tau}(n) \approx \begin{cases} 2n(1 + O(\frac{1}{n})) & \text{if } n \text{ is odd;} \\ \frac{4}{3}n(1 + O(\frac{1}{n})) & \text{if } n \text{ is even.} \end{cases}$$

Again the different behaviour is explained by the case of two trees with the same internal structure and trivially unifying with $(x \leftarrow y)$. In this case, the number of steps is obviously n , while for the other cases is $2n$ on the average.

The above results have been obtained for a very specific type of binary trees, as we already said at the beginning. The reader should not have any problem generalizing these results to trees where internal nodes represent general functions. So far, it has been very difficult to generalize the techniques used in this work to trees with m different types of leaves.

Acknowledgment. The authors thank an anonymous referee for his comments which improved the final version of the manuscript.

References

- [FV,88] P.Flajolet, J.S.Vitter: "Average-case analysis of algorithms and data structures". In *Handbook of Theoretical Computer Science*. North-Holland (1988). (To appear)
- [Hen,77] P.Henrici: "Applied and Computational Complex Analysis", Vol. 2. J. Wiley (1977).
- [Her,30] J. Herbrand: "Recherches sur la théorie de la démonstration" (Thèse, Université de Paris). In: *Ecrits logiques de Jacques Herbrand*, PUF (1968).
- [MM,82] A.Martelli, U.Montanari: "An efficient unification algorithm". *Transactions on Programming Languages and Systems*, 4, 2, (1982), 258–282.
- [Ro,65] J.A.Robinson: "A machine-oriented logic based on the resolution principle". *Journal of the ACM*, 12, 1, (1965), 23–41.
- [Ti,79] E.C.Titchmarsh: "The theory of functions". Second Edition. Oxford University Press (1979).