PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1984,
edited by ICOT. © ICOT, 1984

EQUATIONS AND INEQUATIONS ON FINITE AND INFINITE TREES

Alain Colmerauer
Université Aix Marseille II
Groupe Intelligence Artificielle
ERA CNRS 363

ABSTRACT

This article is devoted to the description and justification of an algorithm for solving systems of equations and "inequations" in the domain of finite and infinite trees. By "inequations" the author designates equations in which the sign = is replaced by the sign ≠.

FOREWORD

Before plunging into an arid lecture on systems of equations and inequations, let us say a word about Prolog.

The basic mechanism of a Prolog machine can be resumed in three lines:

(1) $(\, q_0\ q_1 \ldots q_n \,)$

(2) $p_0 \rightarrow p_1 \ldots p_m$

(3) $(\, \mu p_1 \ldots \mu p_m\ \mu q_1 \ldots \mu q_n \,)$

Line (1) represents the state of the machine at time **t**, line (3) the state of the machine at time **t+1**, and line (2) the rule used to operate this change of state. The **pi**'s and **qi**'s are terms and μ represents the most general substitution of terms for variables which unifies **po** and **qo**, i.e. which renders the terms **μpo** and **μqo** identical.

This very simple formulation is nothing more than a particular case of Alan Robinson's principle of resolution (Robinson 1965). Here is another formulation inspired by Prolog implementations:

(1) $(\, q_0\ q_1 \ldots q_n\, ,\ S\,)$

(2) $p_0 \rightarrow p_1 \ldots p_m$

(3) $(\, p_1 \ldots p_m\ q_1 \ldots q_n\, ,\ SU\{p_0=q_0\}\,)$

where **S** and **SU{po=qo}** are systems of equations admitting at least one solution in the field of finite trees (without variables). We have replaced the notion of unification by that of resolution of a system of equations in a given field. This perspective allows us two types of generalisation.

First, we can operate in a field other than that of finite trees without worrying whether the notion of most general unifier still makes sense. All we need is to be able to decide quickly whether a given system admits or does not admit a solution in the new field. Secondly, we can introduce inequations (equations with sign ≠) by adjoining to each rule a set of constraints limiting the values which can be taken by the variables of the rule. The program which allows us to verify that the element **x** is not included in the list **l** can be written at last in a clean way:

This work was supported by a grant from Digital Equipment within the External Research Program: Extensions to the Prolog Language, Research Agreement no 91 (July 83 – June 84).

```
out(x,nil) -> ;
out(x,list(y,l)) -> out(x,l),  {x≠y}.
```

We must then redefine the basic mechanism of the Prolog machine, by the three lines:

(1) $(q_0\ q_1 \ldots q_n ,\ S)$

(2) $p_0 \rightarrow p_1 \ldots p_m,\ T$

(3) $(p_1 \ldots p_m\ q_1 \ldots q_n ,\ SUTU\{p_0{=}q_0\})$

where S, T and $SUTU\{p_0{=}q_0\}$ are systems of equations and inequations admitting at least one solution.

This latter formulation is the one we have adopted in Prolog II (Colmerauer 1982b) and (Colmerauer, Kanoui and Van Caneghem 1983). Prolog II accepts inequations and operates in the fields of finite and infinite trees. The treatment of inequations considerably increases the capacity of the language and reduces recourses to cuts (/ or !) in the search space. Infinite trees allow us to represent such unusual data as flowcharts and the reader interested in this aspect can consult (Colmerauer 1982a) and (Pique 1984). But the fundamental reason for their introduction is to facilitate the resolution of equations on trees and to avoid the equivalent of the notorious "occur check" in traditional unification. An analogy can be suggested between finite trees and integers on the one hand, and finite/infinite trees and fractions on the other hand: to determine whether an equation, like for example $6x-3y=100$, admits at least one solution is much more difficult in the field of integers than in that of fractions.

The aim of this foreword was to explain why we are now going to devote 15 pages to systems of equations and inequations, to finite and infinite trees, and to the construction of a general algorithm of resolution of equations and inequations on trees. This algorithm, as opposed to that proposed in (Colmerauer 1982) is actually used in the implementations of Prolog II, such as that on VAX under VMS system.

TABLE OF CONTENTS

1 EQUATIONS AND INEQUATIONS IN AN ALGEBRA

1.1 Basic Elements

We have created the word "inequation" for an equation in which the sign = is replaced by the sign ≠. Our aim now is to solve equations and inequations involving trees. We will first consider the general case of equations and inequations on arbitrary mathematical entities constituting an "algebra".

Throughout this paper we will consider an infinite, enumerable set **V** of symbols called "variables". Each time we will need a subset of variables, we will take it from the set **V**. We will also need other symbols called "functional symbols" to each of which a positive integer, called "arity", is associated. When the arity is zero the symbols are called "constants".

Definition: An "algebra" **A** is defined by a set **D**, its "domain", and a set of operations on **D** whose names constitute a set **F** of functional symbols. More precisely:

- to each constant **k** is associated an element **a** of **D**, written $a=\underline{k}$;
- to each functional symbol **f** having arity $n>0$ is associated a function written $\underline{f}$ which maps each sequence $a_1 \ldots a_n$ of elements in **D** into an element **a** of **D**, written $a=\underline{f}a_1 \ldots a_n$.

From now on we will be in the realm of such an algebra which implies that we have taken, among other things, a domain **D** and a set **F** of functional symbols. The well-known notion of "term" lends us formulas to represent elements of the domain **D**. We propose the following definition of "terms":

Definition: A "term" **t** of depth $n \geq 0$ is a finite sequence of juxtaposed elements of **FUV** such that:

- if $n=0$, **t** is reduced to a constant or a variable,
- if $n \geq 1$, **t** is of the form $ft_1 \ldots t_m$, **f** being a functional symbol of arity **n**, and the t_i's being terms of depth less than **n**, with at least one of them of depth $n-1$.

To define the element of the domain represented by a term we must introduce the notion of assignment of variables.

Definition: Let $W=\{x_1,x_2,\ldots\}$ be a subset of variables. An "assignment" **X** of **W** is a set of pairs of the form $X = \{x_1:=a_1,\ x_2:=a_2,\ \ldots\}$ in which the a_i's are elements of **D**. If **t** is a term which does not involve variables other than those of **W**, then the value t/X of **t** in the assignment **X** is the element **a** of **D** obtained by replacing in **t** the x_i's by the corresponding a_i's. More precisely:

- if **t** is a simple variable **x**, then **a** is the only element such that the pair $x:=a$ occurs in **X**,
- if **t** is a simple constant **k**, then $a=\underline{k}$,
- if **t** is of the form $ft_1 \ldots t_m$ and $a_i=t_i/X$, then $a=\underline{f}a_1 \ldots a_m$.

We are now ready to define both syntactically and semantically the notions of equation and inequation.

Definition: If t_1 and t_2 are terms, then the ordered triplets $(t_1,=,t_2)$ and $(t_1,\neq,t_2)$, written as $t_1=t_2$ and $t_1 \neq t_2$, are respectively an "equation" and an "inequation". A "system" **S** is a possibly infinite set of equations and inequations. When **S** contains no inequations, it is referred to as a "system of equations". The assignment **X** of the set **V** of all variables is said to be a solution of **S** if for each equation $s=s'$ and inequation $t \neq t'$ of **S** one has $s/X=s'/X$ and $t/X \neq t'/X$. We use the notation $\underline{\mathbf{Sol}}[S]$ to denote the set of all solutions of **S**. If **S** admits at least one solution, **S** is "solvable", otherwise **S** is "unsolvable". If **W** is an arbitrary subset of variables, we call "solution of" **S** "on" **W**, each assignment of **W** which is a subset of a solution of **S**. We use the notation $\underline{\mathbf{Sol}_W}[S]$ to denote the set of all these assignments. Finally two systems are considered "equivalent" if they have the same set of solutions.

Note that to show that two systems have the same set of solutions it suffices to consider the solution on the subset of variables ocurring in the union of the two systems. These definitions are illustrated by two examples.

Example 1.1.1

Consider the algebra **A**:

- its domain **D** is the set of rational numbers, i.e. the numbers which are representable by fractions;
- its set **F** of functional symbols is $\{0,1,+\}$ where **0** and **1** are constants and the symbol **+** is binary (i.e. of arity 2);
- the elements $\underline{0}$ and $\underline{1}$ are respectively the rationals 0 and 1 and the function $\underline{+}$ is addition.

Let $\{x,y,z,x_1,y_1,z_1,\ldots\}$ be the set **V** of all variables. The system

$$\{+xx=+1+xz,\ \ y=+xz,\ \ +xy \neq +1z\}$$

which is written in infix notation as

$$\{(x+x)=(1+(x+z)),\ \ y=(x+z),(x+y) \neq (1+z)\}$$

is solvable and has this solution among others:

$$\{x:=\underline{1},\ y:=\underline{1},\ z:=\underline{0},\ x_1:=\underline{0},\ y_1:=\underline{0},\ \ldots\}.$$

Example 1.1.2

Consider the classical boolean algebra:

- its domain **D** is the set of logical values "true" and "false";
- its set F of functional symbols is {**1**,**0**,¬,**&**,**v**}, the symbols **1**,**0** being unary (i.e., of arity 1) and the symbol **&**,**v** being binary.
- its elements $\underline{1}$ and $\underline{0}$ are respectively the values "true" and "false", the function $\underline{\neg}$ is the logical negation, the function **&̲** is the logical "and" and the function **v̲** is the logical "or".

The system **{vxy=&xz, x=¬y}** is solvable and has as solution on **{x,y}** the assignment **{x:=$\underline{1}$, y:=$\underline{0}$}**. In contrast the system **{x=y, x≠0, y≠1}** is unsolvable.

1.2 Eliminable variables

A classical way of solving a system of equations is to proceed by elimination of variables. An equation is transformed so that a variable occurs as its left hand side without occurring in its right hand side. By substitution one can then eliminate all other occurrences of this variable in the system. If the system is solvable, repeated applications of this procedure will allow one to express the value of a subset **W** of the variables of the system as a function of the value of the subset **V−W** of the remaining variables. The system is then solved since to enumerate its solutions it suffices to assign any value to the variables in **V−W** and to compute the corresponding values of the variables in **W**.

For example, in the algebra of example 1.1.1 the system **{(x+x)=(1+(x+z), y=(x+z)}** is equivalent to **{x=(1+z), y=(x+z)}** and therefore to **{x=(1+z), y=(1+(z+z))}**. In this latter form the system is solved: its solutions on **{x,y,z}** are obtained by considering any assignment of **{z}** and by computing the corresponding assignments of **{x,y}**. However, this elimination of variables is not always possible as shown in the two systems **{x+y=1}** and **{x+x=y+y+y}**. To render elimination always possible one has to move to classical linear equations by adding the operations of subtraction and multiplication by a given rational number. However, these additions do not change the power of expressiveness of the equations. For, on the one hand, subtractions can be eliminated by moving subterms from one side to the other of an equation and, on the other hand, multiplictions by given rationals can also be eliminated: a linear equation with rational coefficients can be replaced by an equation with integer coefficients, and **kx** can by replaced by **x+...+x** (**k** times) and **k** itself by **0** or **k+...+k** (**k** times). One can therefore say that, in the algebra of example 1.1.1, there exists a potentiality for eliminating a subset of the variables of a solvable system. This leads us to introduce the following notion:

Definition: Given a system **S**, a set of "eliminable variables" is a subset **W** of **V** such that, for any assignment **X** of **V−W**, there exists a unique assignment **Y** of **W** such that **XUY** is a solution of **S**.

In the algebra of example 1.1.1 the solvable system **{(x+y)=1, (y+y)=(z+(z+z))}** yields indifferently **{x,y}**, **{y,z}** or **{x,z}** as sets of eliminable variables. In contrast, in the algebra of example 1.1.2 the solvable system **{&xy=1}** does not have a set of eliminable variables.

We will see later that in the algebra of trees, each solvable system of equations admits at least one set of eliminable variables; hence the importance of the following property.

Property I of eliminable variables. In an algebra whose domain contains at least two elements, let S_1 and S_2 be two systems admitting respectively W_1 and W_2 for sets of eliminable variables. Neither of the following situations is possible:

(1) $W_1=W_2$ and $\underline{Sol}[S_1]<\underline{Sol}[S_2]$,

(2) $W_1<W_2$ and $\underline{Sol}[S_1]=\underline{Sol}[S_2]$.

(The sign < denotes strict set inclusion.)

The proof of this property is easy and is given in (Colmerauer 1984).

1.3 Independent Inequations

Let us now examine the problem of deciding whether or not a system with inequations is solvable. This problem may be divided into several simpler subproblems if the inequations are independent, in the following sense:

Definition: Inequations are said to be "independent" in an algebra if, for any finite system **S** of equations and for any **n** inequations $s_1 \neq t_1$, ..., $s_n \neq t_n$, one has the following property:

- the global system $S \cup \{s_1 \neq t_1, \dots, s_n \neq t_n\}$ is solvable if and only if each of the **n** subsystems $S \cup \{s_1 \neq t_1\}$, ..., $S \cup \{s_n \neq t_n\}$ is separately solvable.

In the boolean algebra of example 1.1.2, the global system $\{x=y,\ x\neq 0,\ y\neq 1\}$ is unsolvable, whereas the subsystems $\{x=y, x\neq 0\}$ and $\{x=y, y\neq 1\}$ are solvable. Therefore inequations are not independent in this algebra. As to the algebra of example 1.1.1 the next property allows us to prove that the inequations are indeed independent. We will use this same property to show the independence of inequations in the algebra of trees.

Property II of eliminable variables: The following two conditions are sufficient to establish the independence of inequations in a given algebra:

(1) the domain of the algebra is infinite;

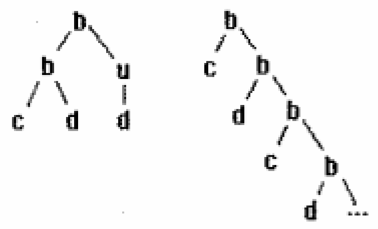
(2) each finite solvable system of equations S_1 admits a set W_1 of eliminable variables; in addition if S_2 is another system of this type then $Sol[S_2] < Sol[S_1]$ implies that S_2 admits a set W_2 of eliminable variables such that $W_1 < W_2$. (We remind the reader that $<$ denotes strict set inclusion.)

The proof of this property is given in (Colmerauer 1984) and is based on an idea of Pierre Siegel: to compare the "sizes" of different infinite sets, we render them finite by intersection with a single and judiciously chosen finite set.

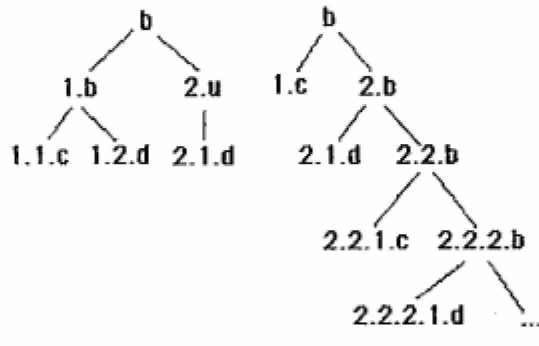
2 ALGEBRA OF TREES

2.1 Definition

The trees that interest us are formed by nodes labeled by functional symbols taken from a set F. The branches which stem from a node are ordered and their number must be equal to the arity of the symbol **f**, the node label. We present two examples of trees on **F={c,d,u,b}**, the symbols **c** and **d** being constants, **u** a unary functional symbol and **b** a binary functional symbol.



Note that a tree can be infinite. To achieve a formal definition of trees we must introduce a scheme for naming the nodes which express their hierarchy. We use the common notation of numbering paragraphs, then subparagraphs and so on:



The numbering allows us to represent the above trees by the sets:

{b, 1.b, 2.u, 1.1.c, 1.2.d, 2.1.d}

{b, 1.c, 2.b, 2.1.d, 2.2.b, 2.2.1.c, 2.2.2.b, 2.2.2.1, ...}

This technique suggests the two following definitions:

Definition: A "node" **u** of depth **n≥0** is a sequence of **n** positive integers $i_1,\dots,i_n$ ending by a functional symbol **f**, and the elements of which are separated by periods: $u=i_1.---.i_n.f$. If **j** is a positive integer, then **j.u** denotes the node $j.i_1.---.i_n.f$.

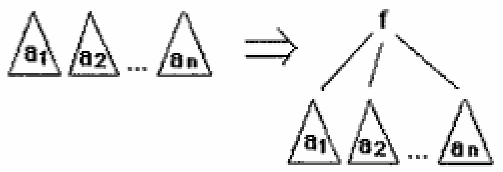
Definition: A "tree" **a** is a set of nodes satisfying the following conditions:

(1) If $i_1.---.i_n.f$ is a node of **a** and if **g** is a functional symbol other than **f**, then $i_1.---.i_n.g$ is not a node of **a**. (Intuitively, two nodes cannot occupy the same position.)

(2) For any integer **i** and node **u**, if **i.u** is a node of **a** then **u** is also a node of **a**. (Intuitively, each node other than the root has a father.)

(3) For all **n≥0** if $i_1.---.i_n.f$ is a node of **a** and if **f** is a functional symbol of arity **m**, then the set of integers **j** such that **a** contains a node of the form $i_1.---.i_n.j.g$, where **g** is any functional symbol, is the set **{1,...,m}**. (Intuitively, the number of sons of a node is the arity of its label.)

The simplest operation on trees consists of taking **n** trees $a_1,\dots,a_n$, a functional symbol **f** of arity **n** and constructing a new tree $\underline{f}a_1\dots a_n$ as follows:



Equipped with this operation, the trees constitute the algebra which interests us and which we now define more formally.

Definition: Let F be a set of functional symbols containing at least one constant and one binary functional symbol **b**. The algebra of trees on F has :

- as domain, the set of trees on F,
- as set of functionnal symbols, F,
- as operation $\underline{f}$, associated to each functional symbol **f** of arity **n≥0**, the operation which to each sequence of **n** trees $a_1,\dots,a_n$ associates the tree $a=\underline{f}a_1\dots a_n$ defined by:

a = {f} U

{1.u | u $\underline{el}$ a_1} U ... U {m.u | u $\underline{el}$ a_m},

where $\underline{el}$ indicates set membership.

The existence of at least a constant and a binary functional symbol ensures that the domain of the algebra is infinite and also allows the coding of any sequence of trees by a single tree.

2.2. Caracteristic properties

The algebra of trees has two basic properties referred to as "unique decomposition" and "unique solution" of a "generating" system. We will qualify these two properties as characteristic, since, as we will see in paragraph 2.4, they are sufficient to prove all other properties of this algebra.

Definition: A "generating" system is a system of equations of the form

$\{x_1=t_1, x_2=t2, \dots\}$,

in which the x_i's are distinct variables, the t_i's are not variables and do not contain variables other than the x_i's.

Characteristic properties: The algebra of trees, of domain **D** and of set of functional symbols **F**, has the following two properties:

- "unique decomposition": for any element **a** of **D**, there exists one and only one sequence (eventually empty) $a_1\dots a_n$ of elements of **D** and one and only one element **f** of **F** with arity **n** such that $a=\underline{f}a_1\dots a_n$; the sequence $a_1\dots a_n$ is called "sequence of sons" of **a**;
- "unique solution": each generating system, involving a subset **W** of variables, admits one and only one solution on **W**.

These properties are proved in (Colmerauer 1984). The first is obvious. The second is proved by exhibiting formulas which recursively define the set of nodes of depth **n** which make up the trees of the solution.

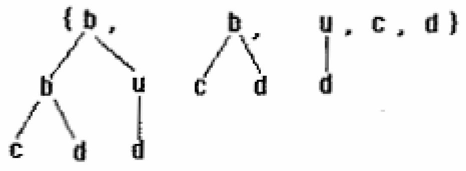
2.3 Subtrees

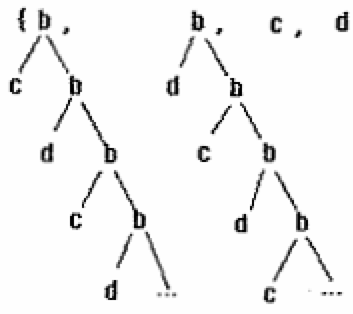
The formulation of the property of unique decomposition leads to the notion of sequence of sons which in turn allows us to introduce the notion of "subtrees".

Definition: The set E of "subtrees" of a given tree **a** is the subset of trees obtained by the infinite union **E** = $E_0 \cup E_1 \cup \ldots$ where E_i, the set of "subtrees of level" **i** of **a**, is defined by:

- $E_0 = \{a\}$,
- E_{i+1} is the set of **b**'s such that there exists an element of E_i whose sequence of sons has the tree **b** as element.

The two preceding examples of trees have for set of subtrees:

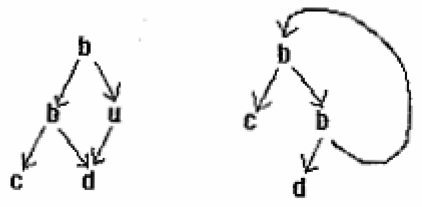




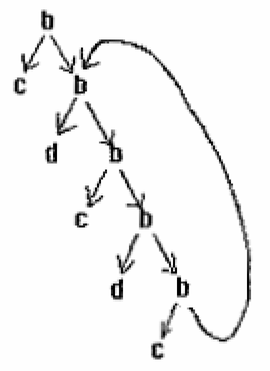
It should be noted that even the infinite tree has a finite set of subtrees. It is therefore a "rational" tree:

Definition: A "rational" tree is a tree whose set of subtrees is finite.

Rational trees can always be represented by a finite diagram: it suffices to merge all the nodes from which the same subtrees stem. In the case of our examples we get:

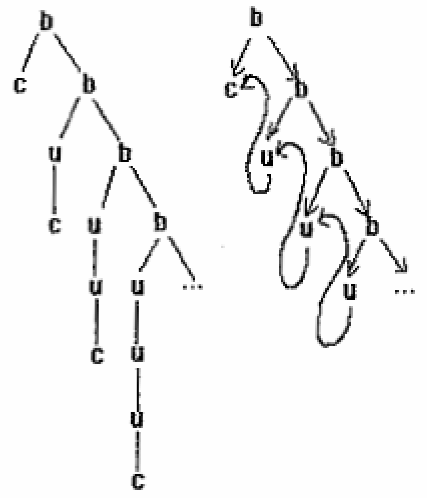


By not merging all the nodes of the second tree one can also obtain:



Therefore one has to be aware that the same tree may be represented by different diagrams.

Finally, it is interesting to present a non-rational tree and its corresponding infinite diagram obtained after all possible merges of nodes.



Note that the set **E** of subtrees of a given tree **a**, is always enumerable since it is obtained by enumerable unions of finite sets. One can therefore express **E** as **E={a1,a2,...}**.

Since the total set **V** of variables is infinite but enumerable, one can associate a distinct variable x_i to each subtree a_i. Let **W** be the set of these x_i's. The decomposition property implies that to each a_i corresponds one and only one equality of the form $a_i=\underline{f_i}a_1\ldots a_{n_i}$, to which corresponds the equation $x_i=\underline{f_i}x_1\ldots x_{n_i}$. The set of these equations forms therefore a generating system which, according to the property of unique solution, admits on **W** the unique solution $\{x_1:=a_1, x_2:=a_2, \ldots\}$. We thus conclude that:

Property of associated system: For any tree a_1 there exists, modulo the name of the variables, one and only one generating system **S**:

- which has the form $S = \{x_1=t_1, x_2=t_2, \ldots\}$, the t_i's containing one and only one occurrence of a functional symbol,
- and which is such that, on $\{x_1, x_2, \ldots\}$, the only solution of **S** is the assignment $\{x_1:=a_1,\ x_2:=a_2,\ \ldots\}$, where the a_i's are all distinct and form the set of subtrees of a_1.

The three trees which we considered for example yield the following systems:

$$\{x_1=bx_2x_3,\ x_2=bx_4x_5,\ x_3=ux_5,\ x_4=c,\ x_5=d\}$$

$$\{x_1=bx_3x_2,\ x_2=bx_4x_1,\ x_3=c,\ x_4=d\}$$

$$\{x_1=bx_2x_3,\ x_2=c,\ x_3=bx_4x_5,\ x_4=ux_2,\ x_5=bx_6x_7,\ x_6=ux_4,\ x_7=bx_8x_9,\ \ldots\}$$

2.4. Isomorphic Algebras

We defined the notion of subtree by means of the property of unique decomposition, and we proved the property of associated system using only the notion of subtree and the property of unique decomposition and solution. This property of associated system is thus valid in all algebras having the other two properties. This founds the following isomorphisms, the proof of which is given in (Colmerauer 1984):

Property of isomorphism: Let **A** and **B** be two algebras having domain D_A and D_B and having the same set **F** of functional symbols. If these two algebras have the characteristic property of the algebra of trees, then these algebras are isomorphic, i.e. there exists a mapping **g** from D_A into D_B and a mapping **h** from D_B into D_A which satisfy the two conditions:

(1) $a = h[g[a]]$ and $b = g[h[b]]$, for each element **a** of D_A and for each element **b** of D_B;

(2) $g[\underline{f}a_1\ldots a_n] = \underline{f}(g[a_1]\ldots g[a_n])$ and $h[\underline{f}b_1\ldots b_n] = f(\underline{h}[b_1]\ldots gh[b_n])$, for each **f** belonging to **F** and with arity $n\geq 1$, for each sequence $a_1,\ldots,a_n$ of elements of D_A and for each sequence $b,\ldots,b_n$ of elements of D_B.

Given this isomorphism we could have defined the algebra of trees as "the" algebra having the properties of unique decomposition and unique solution. In the rest of this paper we can therefore ignore that trees are made from nodes and base our reasoning on these two properties of unique decomposition and unique solution.

3 SOLVING EQUATIONS ON TREES

3.1 Reduced system

Let us first introduce three specific types of systems of equations and the concept of "representative":

Definition: An "endless" system is a system of equations in which every term which occurs as the right hand side of an equation also occurs as the left hand side of an equation.

Definition: A "reduced" system is a finite system of equations having the following two properties :

(1) the left hand sides of its equations are distinct variables,

(2) it does not contain an endless subsystem.

Definition: The notion of "representative" is defined exclusively for a system **S** in which left hand sides of equations are distinct and which does not contain an endless subsystem.

To each term **t** is associated a term written **<u>rep</u>[t,S]** called the "representative" of **t** in **S** and defined by:

- **<u>rep</u>[t,S]** = **<u>rep</u>[t',S]**, if **S** contains an equation of the form **t=t'**,
- **<u>rep</u>[t,S]** = **t**, otherwise.

Definition: A "decomposed" system is a finite system of equations **S** having the following five properties :

(1) the left hand sides of its equations are distinct;

(2) it does not contain an endless subsystem;

(3) it does not contain any equation of the form **t=x** in which **x** is a variable and **t** is not a variable;

(4) it does not contain any equation of the form $fs_1...s_m=gt_1...t_n$ where **f** and **g** are distinct functional symbols and where **m** and **n** may be zero;

(5) the presence in **S** of any equation of the form $fs_1...s_n=ft_1...t_n$, with $n \geq 1$, implies that for each $i=1,...,n$, **<u>rep</u>[s$_i$,S]=<u>rep</u>[t$_i$,S]**.

Let us reconsider the set of variables **V={x,y,z,...}** and the set of functional symbols **F={c,d,u,b}** where **c** and **d** are constants, **u** is a unary functional symbol and **b** a binary functional symbol. The systems **{x=x}**, **{x=y, y=uz, uz=x}** and **{c=uc, uc=uuc, uuc=uuuc, ...}** are endless systems. The system **{x=y, y=bxz}** is a reduced system and the system **{x=y, y=bxz, ux=ubxz}** is a decomposed system.

Reduced systems are the most interesting ones. They have the following fundamental property, the proof of which is given in (Colmerauer 1984):

Property I of reduced systems: Each reduced system admits as a set of eliminable variables the set of the left hand sides of its equations.

Therefore, let **S** be a reduced system involving the subset of variables **W** and let **W'** be the set of variables occurring as left hand sides of its equations. Such a system is already solved: according to the definition of a set of eliminable variables, the set of its solutions on **W** is obtained by considering each assignment **X** of **W-W'** and completing it by the only assignment **Y** of **W'** such that **XUY** be a solution of **S** on **W**.

Let us now go on to decomposable systems. These systems always contain one largest reduced subsystem constituted by the equations whose left hand sides are variables. This reduced system has the following advantageous property:

Property of decomposable systems: Each decomposable system is equivalent to the largest reduced subsystem it contains.

Although it initially may appear easy, the proof of this property is far from being trivial. It is given in (Colmerauer 1984).

3.2 Reduction algorithm

To solve a system of equations **S** in the algebra of trees, we will subject it to a series of transformations. These will produce a final system, equivalent to the initial system, and whose form will allow, either to conclude that the initial system is unsolvable, or to render all its solutions visible. In this last case the form of the final system will be that of a reduced system.

Each transformation will be justified by two properties which are direct consequences of the property of unique decomposition of trees:

Properties of an equation: If **f** and **g** are distinct functional symbols of arity **m** and **n** respectively and if the s_i's and t_i's are arbitrary terms then :

- the systems $\{fs_1...s_m=ft_1...t_m\}$ and $\{s_1=t_1,...,s_n=t_m\}$ are equivalent,
- the system $\{fs_1...s_m=gt_1...t_n\}$ is unsolvable.

Note that neither of these properties is true in the majority of classical algebras: in arithmetic (example 1.1.1) the systems

{(x+1)=(0+y)} and **{x=0,1=y}** are not equivalent and the system **{(x+1)=0}** is solvable.

We are now ready to propose and justify an algorithm which allows, whenever possible, the transformation of a finite system into an equivalent reduced system.

Reduction algorithm: It is proposed to "reduce" a finite system of the form **SUT**, where **S** is already reduced and where **T** is the set of equations occuring in a finite sequence **Z** of equations. The pair **<S,Z>** is repeatedly modified using the "basic operation" which is defined below. If in the way, a basic operation is not executed normally then the initial system **SUT** is unsolvable. Otherwise the final result is a pair of the form **<S',()>**. The subset of **S'**, constituted from the equations whose left hand sides are variables, is then a reduced system equivalent to the initial system **SUT** and contains the original system **S**.

Basic operation: Choose in **Z** any occurrence of an equation **s'=t'** and remove it. Let **s=<u>rep</u>[s',S]** and **t=<u>rep</u>[t',S]**. If **s=t** the operation terminates, otherwise there are 3 cases :

- at least one of the terms **s** and **t** is a variable; in this case add to **S** one of the equations **s=t** or **t=s**, provided that the left hand side of the added equation is a variable;
- the terms **s** and **t** are respectively of the form **fs₁...sₙ** and **ft₁...tₙ**, with **n≥1**; in this case add to **S** one of the equations **s=t** or **t=s** and insert, anywhere in **Z**, the sequence of equations **s₁=t₁,...,sₙ=tₙ**;
- the terms **s** and **t** are respectively of the form **fs₁...sₘ** and **gt₁...tₙ**, where **f** and **g** are distinct functional symbols, with **m≥1** and **n≥1**; in this case alone, we consider the execution of the basic operation to be abnormal.

The following three arguments provide a proof of the correctness of the algorithm.

First, we must prove that the algorithm is well defined, i.e that the notion of representative has always a meaning in **S**:

The initial system is reduced. The left hand sides of its equations are therefore distinct and the system does not contain an endless subsystem. The changes in **S** are made by successive additions of new equations both sides of which are different from the left hand sides of the already existing equations. The left hand sides of the equations of **S** remain therefore distinct and an endless subsystem cannot be generated.

Secondly, we must prove that the algorithm always terminates:

Each basic operation modifies the pair **<S,Z>**, either by increasing **S** or by maintaining the previous **S** but by decreasing the length of the sequence **Z**. The system **S** cannot be increased indefinitely because the left hand sides of its equations are distinct and are terms occurring somewhere in the initial pair **<S,Z>**. If the algorithm does not terminate abnormally, there will always be a phase after which **S** will not increase and **Z** will decrease until it becomes empty.

Finally, we must prove that the algorithm produces the result announced:

Let **T** be the set of equations which are present in the current sequence **Z**. According to the properties of an equation, each basic operation transforms **SUT** into an equivalent system and if the operation executes abnormally, the initial system is unsolvable. Moreover, due to its initial form and to the nature of the basic operations, the current pair **<S,Z>** always has the following property: the presence in **S** of a an equation of the form **fs₁...sₙ=ft₁...tₙ**, with **n≥1**, implies that for each **i=1,...,n**, if one sets **sᵢ'=<u>rep</u>[sᵢ,S]** and **tᵢ'=<u>rep</u>[tᵢ,S]**, either **sᵢ'=tᵢ'**, or one of the two equations **sᵢ'=tᵢ'** or **tᵢ'=sᵢ'** occurs in **Z**. It follows that in the final pair **<S',()>** the system **S'** is decomposed and equivalent to the initial system **SUT**. Therefore, according to the property of decomposed systems (paragraph 3.1), **S'** has the properties announced.

The general idea of this algorithm, and in particular the notion of "representative", is due to Gérard Huet (Huet 1976). The same algorithm, formulated in the context of the unification of extended terms (which may be infinite) can be found in (Fages 1983).

Example 3.2.1

Let us terminate the reduction of the system represented by the pair:

<{x=uux,y=uuuy}, (x=y)>,

where **x,y** are variables and **u** is a unary functional symbol. We obtain successively:

<{x=uux,y=uuuy}, (x=y)>,

<{x=uux,y=uuuy}U{uux=uuuy}, (ux=uuy)>,

<{x=uux,y=uuuy}U{uux=uuuy,ux=uuy}, (x=uy)>,

<{x=uux,y=uuuy}U {uux=uuuy,ux=uuy,uuuy=uy}, (uuy=y)>,

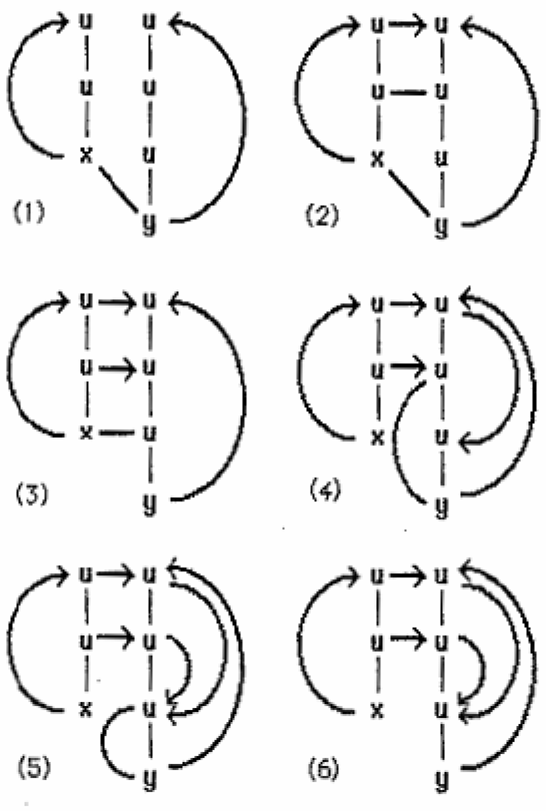
<{x=uux,y=uuuy}U {uux=uuuy,ux=uuy,uuuy=uy,uuy=uy}, (uy=y)>,

<{x=uux,y=uuuy}U {uux=uuuy,ux=uuy,uuuy=uy,uuy=uy}, ()>,

and thus finally:

{x=uux,y=uuuy}.

It is interesting to depict in graphic form these different steps. We associate to each term **t**, occuring anywhere (even inside a term) in the initial pair **<S,Z>**, a unique point **<u>pt</u>[t]**. On these points we trace a structure representing these terms. Each equation **s=t** of the system **S** is materialised by the arrow **<u>pt</u>[s]->**<u>**pt**</u>**[t]**. Similarly each equation **s=t** of the sequence **Z** is materialized by the line **<u>pt</u>[s]--<u>pt</u>[t]**. The six steps of the reduction of our previous example are thus represented by:

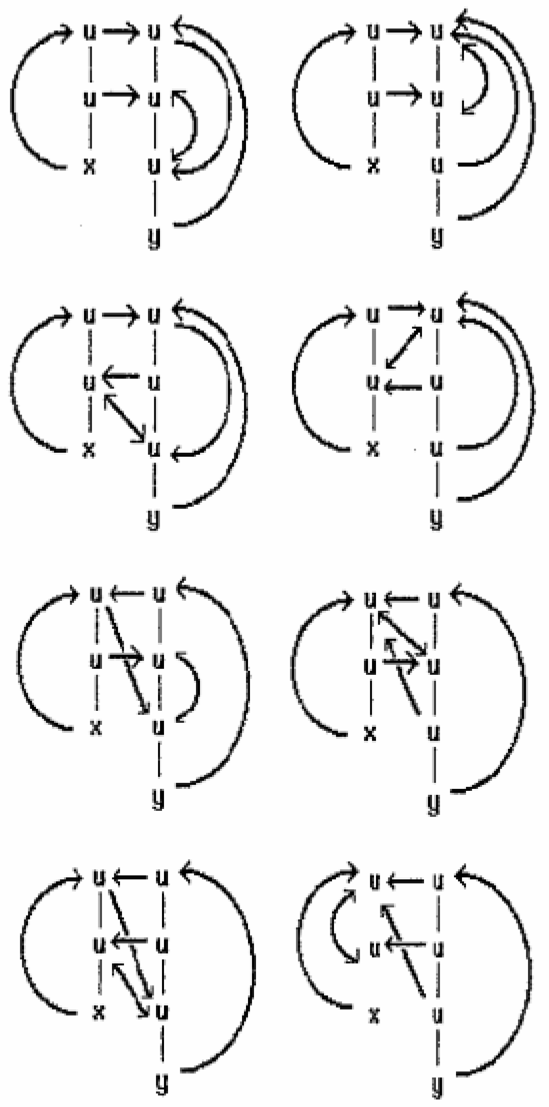


In the process of reduction of the pair **<S,Z>** there exist two degrees of freedom: the first degree is the possibility of adding to the system either the equation **s=t** or the equation **t=s**; the second degree is the possibility of selecting arbitrarily the next equation in **Z** to be processed.

The use of the first degree of freedom leads to the computation of 16 final different pairs when starting from the pair:

<{x=uux,y=uuuy}, (x=y)>.

If we agree that the bidirectionnal arrow **<->** replaces indifferently the arrow **->** or the arrow **<-**, these 16 pairs can be represented by the 8 following drawings:



By exploiting the second degree of freedom one can generate indifferently one of the 3 reduced systems:

{x=uux, y=uuuy},
{x=y, y=uuuy},
{x=uux, y=x},

when starting from the pair:

<{}, (x=y,x=uux,y=uuuy)>.

It is interesting to estimate the speed of our reduction algorithm. We get the following result:

Property of linearity: Consider the reduction of a system of equations represented by the pair **<S,Z>**, and let **n** be the total number of occurences of symbols in **<S,Z>**. The reduction algorithm terminates after performing at most **n** basic operations.

The proof of this property is as follows. Let **m** be the total number of executed basic operations. We have:

(1) $m \le p + \underline{length}[Z]$

where **p** is the total number of equations which will be inserted in the system **S** and **<u>length</u>[Z]** is the length of **Z**. According to the definition of the basic operations, **p** is also the number of equations which will be added to **S** and whose left hand sides have non-zero depth. These left hand sides are all distinct and are terms already occurring somewhere (even in a term) in the initial pair **<S,Z>**. The total number of occurences of terms in **<S,Z>** is **n** and, since each equation contains at least two occurrences of terms of depth zero, we have :

(2) $p \le n - 2(\underline{card}[S]+\underline{length}[Z])$.

From (1) and (2) it follows that :

(3) $m \le n - 2\underline{card}[S] - \underline{length}[Z]$,

and thus:

(4) $m \le n$.

Note however that this property does not prove that our algorithm is linear with respect to the size of data it processes. Actually the processing time of a basic operation increases gradually as execution progresses.

Finally the very existence of this algorithm allows us to conclude this section with an important result:

Property II of reduced systems: A finite system of equations S_1 is solvable if and only if there exists a reduced system T_1 which is equivalent to S_1; moreover, if S_1 is solvable, and if S_2 is another finite and solvable system of equations, then $Sol[S_2]<Sol[S_1]$ if and only if, from all reduced systems T_2 equivalent to S_2, there is one such that $T_1<T_2$.

We remind the reader that we use the sign **<** for strict set inclusion. The first "if" and the first "only if" part of property II are obvious. The second "if" part is proven by combinir the property I of reduced systems (paragraph 3.1) and the property I of eliminable variables (paragraph 1.2). The second "only if" part is proven by applying the reduction algorithm to the pair **$\langle T_1, Z_2\rangle$**, where **Z_2** is a sequence of equations representing **S_2**.

4 SOLVING EQUATIONS AND INEQUATIONS ON TREES

4.1 Simplified System

We are now interested in a decision procedure for establishing weither a system which contains inequations is solvable. This problem can be subdivided into several subproblems because of what follows:

Independance property of inequations: In the algebra of trees the inequations are independent, i.e. for every finite system of equations and for all **n** inequations **$s_1 \neq t_1$**, ..., **$s_n \neq t_n$**:

- the global system SU**$\{s_1 \neq t_1, \ldots, s_n \neq t_n\}$** is solvable if and only if each of the **n** subsystems SU**$\{s_1 \neq t_1\}$**, ..., SU**$\{s_n \neq t_n\}$** is separately solvable.

This is a direct consequence of the property II of eliminable variables (paragraph 3.3), of the fact that the set of trees is infinite, and of the properties I and II of reduced systems (paragraphs 3.1 and 3.2). We can therefore limit ourselves to a system containing only one inequation.

In formulating the next result, the binary functional symbol **b** is used to code each non-empty sequence of terms **$t_s, \ldots, t_n$** by the term **$b..bt_n...t_1$** which contains **n** occurrences of **b**.

Properties of an inequation: Let **S** be a reduced system and **$s \neq t$** an inequation. The system SU**$\{s \neq t\}$** is solvable if and only if one of the two following cases occurs:

(1) the system SU**$\{s = t\}$** is unsolvable: the system SU**$\{s \neq t\}$** is then equivalent to the system **S**;

(2) the system **SU$\{s = t\}$** is solvable and admits a reduced system of the form **SUT**, with **T** non empty and disjoint from **S**: if one sets **T = $\{y_1 = t_1, \ldots, y_n = t_n\}$**, the system **SU$\{s \neq t\}$** is then equivalent to **SU$\{b..by_n...y_1 \neq b..bt_n...t_1\}$**.

Let us now introduce the notion of a "simplified" system, which is an extension of that of a reduced system

Definition: A "simplified" system is a finite system having the two following properties :

(1) the set of its equations forms a reduced system **S**;

(2) each of its inequations is of the form **$b..bs_n...s_1 \neq b..bt_n...t_1$**, with **$n \geq 1$**, the **$s_i$**'s and the **$t_i$**'s being any term, but for **s_1** which must be a variable not occurring as left hand side of an equation in **S**, and but for **t_1** which must be such that the system **SU$\{s_1 = t_1\}$** does not contain an endless system.

By noticing that the system **SU$\{b..bs_n...s_1 \neq b..bt_n...t_1\}$** is solvable when the system **SU$\{s_1 \neq t_1\}$** is solvable, we are led to the following foreseeable result:

Property of simplified systems: All simplified systems are solvable.

4.2 Simplification Algorithm

To solve a system with inequations it suffices to put it in a simplified form. For this purpose we propose the following algorithm:

Simplification algorithm: We wish to simplify a system of the form **$(S_e US_i)U(T_e UT_i)$**, where the system **$S_e US_i$** is already simplified and where **$T_e UT_i$** is the complete set of equations and inequations occurring in a finite sequence **Z_e** of equations and a finite sequence **Z_i** of inequations. The indices **e** and **i** are systematically used to specify respectively equation parts and inequation parts. The algorithm consists of transforming the pair **$\langle S_e US_i, (Z_e, Z_i)\rangle$** in three steps:

(1) We process first the equations by applying the reduction algorithm to the pair $\langle S_e, Z_e \rangle$. We obtain a reduced system S_e'. If this reduced system S_e' does not exist, the initial system $(S_e \cup S_i) \cup (T_e \cup T_i)$ is unsolvable and the algorithm terminates. Otherwise the pair $\langle S_e' \cup S_i, Z_i \rangle$ remains to be processed.

(2) We remove from the set S_i all inequations $s \neq t$ such that the system $S_e' \cup \{s \neq t\}$ is not in a simplified form, and we insert them anywhere in the sequence Z_i. Let S_i' and Z_i' be the new resulting set and the new resulting sequence. The pair $\langle S_e' \cup S_i', Z_i' \rangle$ remains to be processed.

(3) We consider the pair $\langle S_e' \cup S_i', Z_i' \rangle$ and repeatedly modify it by the "basic operation" which is defined below. If a basic operation executes abnormally the initial system $(S_e \cup S_i) \cup (T_e \cup T_i)$ is unsolvable. Otherwise we obtain a final pair of the form $\langle S_e' \cup S_i'', () \rangle$. The system $S_e' \cup S_i''$ then constitutes a simplified system equivalent to the initial system $(S_e \cup S_i) \cup (T_e \cup T_i)$.

Basic operation: Remove an occurrence of an inequation $s \neq t$ from the sequence Z_i' and apply the reduction algorithm on the pair $\langle S_e', (s = t) \rangle$. Three cases are possible :

(1) the reduction algorithm fails to produce a reduced system: in this case the operation terminates;

(2) the reduced system generated is of the form $S_e' \cup T$ with T disjoint from S_e' and of the form $T = \{y_1 = t_1, \ldots, y_n = t_n\}$ with $n \geq 1$: in this case we add to the set S_i' the inequation $b..by_n \ldots y_1 \neq b..bt_n \ldots t_1$;

(3) the reduced system generated is S_e': in this case alone we consider the execution of the basic operation to be abnormal.

The algorithm always terminates, since it consists of applying at most $1+m+n$ times the reduction algorithm, m and n being respectively the cardinality of the set S_i and the length of the sequence Z_i. In addition, the various manipulations involved are justified by the properties of an inequation (paragraph 4.1). Thus the correctness of the algorithm is proven.

Example 4.2.1

Let us finish the simplification of the system represented by the pair:

$$\langle \{x=uy\} \cup \{z \neq d, bzy \neq bcc\},\ (y=bzy, y \neq z) \rangle,$$

where x, y, z are variables, u is a unary functional symbol and b is the standard binary functional symbol.

First step: We process the equations. To that end we apply the reduction algorithm to:

$\langle \{x=uy\},\ (y=bzy) \rangle$

obtaining the reduced system:

$\{x=uy,\ y=bzy\}$.

The pair

$\langle \{x=uy, y=bzy\} \cup \{z \neq d, bzy \neq bcc\},\ (y \neq z) \rangle$

now remains to be processed.

Second step: We single out the inequations which need reprocessing. The inequation $bzy \neq bcc$ is moved to the right. The pair

$\langle \{x=uy, y=bzy\} \cup \{z \neq d\},\ (bzy \neq bcc, y \neq z) \rangle$

remains to be processed.

Third step: We process consecutively the two inequations occurring on the right. We start by applying the reduction algorithm to:

$\langle \{x=uy, y=bzy\},\ (bzy=bcc) \rangle$,

which yields successively:

$\langle \{x=uy, y=bzy\} \cup \{bzy=bcc\},\ (z=c, y=c) \rangle$,

$\langle \{x=uy, y=bzy\} \cup \{bzy=bcc, z=c\},\ (y=c) \rangle$,

and terminates abnormally. The first inequation is thus deleted. The following pair remains to be simplified:

$\langle \{x=uy, y=bzy\} \cup \{z \neq d\},\ (y \neq z) \rangle$.

The reduction algorithm is applied to:

$\langle \{x=uy, y=bzy\},\ (y=z) \rangle$,

yielding the reduced system:

$\{x=uy, y=bzy, z=bzy\}$.

The final pair is therefore:

$\langle \{x=uy, y=bzy\} \cup \{z \neq d, z \neq bzy\},\ () \rangle$,

and the final result is thus the simplified system:

$$\{x=uy, y=bzy\} \cup \{z \neq d, z \neq bzy\}.$$

Example 4.2.2

Assume we wish to continue the simplification of the system represented by the pair:

$$\langle\{x=ux, y=uuy\}, (x\neq y)\rangle.$$

We can proceed directly to the third step and apply the reduction algorithm to:

$$\langle\{x=ux, y=uuy\}, (x=y)\rangle.$$

We successively obtain:

$$\langle\{x=ux, y=uuy\}\cup\{ux=uuy\}, (x=uy)\rangle,$$
$$\langle\{x=ux, y=uuy\}\cup\{ux=uuy, uuy=uy\}, (uy=y)\rangle,$$
$$\langle\{x=ux, y=uuy\}\cup\{ux=uuy, uuy=uy\}, ()\rangle,$$

yielding the reduced system:

$$\{x=ux, y=uuy\}.$$

Therefore the system we proposed to simplify was unsolvable.

Let us mention a technical point which is of importance when implementing the simplification algorithm. In the second step, consisting of singling out the inequations $b..bx_n...x_1\neq b..bt_n...t_1$ to be reprocessed, there is a method for ensuring that the system $S_\bullet'\cup\{x_1=t_1\}$ never contains an endless system. It suffices to order the variables once and for all; then each time it is possible in the reduction algorithm to add the equation $x=y$ or $y=x$ to the system S of the pair $\langle S,Z\rangle$, we choose $x=y$ if $x<y$ and $y=x$ if $y<x$. Under these conditions, the only inequations $b..bx_n...x_1\neq b..bt_n...t_1$ to be reprocessed are those in which the variable x_1 occurs as the left hand side of an equation of the system $S_\bullet'$.

We will end this presentation by mentioning that as early as 1971, Philippe Roussel did some work on the treatment of the inequality $\neq$ in the context of automatic theorem proving (Roussel 1972).

REFERENCES

Colmerauer A. (1982a). Prolog and Infinite Trees, *Logic Programming*, edited by K.L. Clark and S.A. Tarnlund, APIC Studies in Data Processing no 16, Academic Press.

Colmerauer A. (1982b). *Prolog II: Reference Manual and Theoretical Model*, Internal report, Groupe Intelligence Artificielle, Université Aix-Marseille II.

Colmerauer A. (1984). *Equations et inéquations sur les arbres finis et infinis*, Internal report, Groupe Intelligence Artificielle, Université Aix-Marseille II.

Colmerauer A., H. Kanoui and M. Van Caneghem (1983). Prolog, theoretical principle and current trends, *Technology and Science of Informatics, Volume 2, no 4*, North Oxford Academic.

Fages F. (1983). *Formes canoniques dans les algèbres booléennes, et application à la démonstration automatique*, Thèse de 3ème cycle, Université Paris VI.

Huet G. (1976). *Résolution d'équations dans les langages d'ordre 1,2,..., Ω*, Thèse de doctorat d'état, Université Paris VI.

Pique J.F. (1984). Drawing Trees and Their Equations in Prolog, *Proceedings of the Second International Logic programming Conference*, Uppsala University, Sweden.

Roussel P. (1972). *Définition et traitement de l'égalité formelle en démonstration automatique*, Thèse de 3ème cycle, Groupe Intelligence Artificielle, Université Aix-Marseille II.

Robinson J.A. (1965). A Machine-Oriented Logic based on the Resolution Principle, *JACM 12, no 1*, pp. 227-234, decembre.