AC-Unification Race: The System Solving Approach, Implementation and Benchmarks

MOHAMED ADI AND CLAUDE KIRCHNER
INRIA Lorraine & CRIN,
615 rue du jardin botanique, BP 101, 54602 Villers les Nancy Cedex, France.
E-mail: {adi, ckirchner}@loria.fr.
(Received November 16, 1989)

We present an Associative-Commutative unification algorithm and its implementation in the C language. The main point is that both the specification and the program are based on solving systems of equations. Benchmarks are proposed for evaluating the performance of the implementation of this algorithm. They demonstrate improvements not only in time and space but also in size of the complete set of unifiers generated.

1. Introduction

A number of applications and languages in computer science make intensive use of equation solving in various structures and particularly of unification in equational theories Siekmann (1990), Jouannaud & Kirchner (1991). Typical examples can be found in automated theorem proving, completion of rewrite rule systems, deduction with symbolic constraints (Kirchner et al. 1990) or in logic programming languages like LPG (Bert et al. 1989) or EQLOG Goguen & Meseguer (1986).

Associative-commutative unification (in short AC-unification) is solving equations in a term algebra where a finite set of functions symbols $(+_i)_{0 \leq i \leq n}$ are associative and commutative, that is satisfy $AC(+_i)$:

\[(x +_i y) +_i z = x +_i (y +_i z) \]
\[x +_i y = y +_i x\]

This unification problem has been the most intensively studied after the unification problem in the empty theory for at least two reasons. The first one is that the associative and commutative equational axioms are associated with many algebraic structures of interest for theorem proving or algebraic specifications. The second and more technical one is that, in term rewriting applications, one cannot dissociate associativity from commutativity because the rewriting relation associated with (left or right) associativity modulo commutativity is not terminating. Thus one should rewrite modulo associativity and commutativity, both treated as equational axioms.
But solving associative-commutative equations (i.e. solving equations that contain AC-symbols) is a difficult problem unlike for commutativity alone. Difficulties are firstly in the discovery of complete AC-unification procedures, secondly in the proof of their termination, thirdly in managing the algorithmic complexity of the problem which is known to be NP-complete Kapur & Narendran (1986).

In the mid seventies, Livesey & Siekmann (1976) and Stickel (1975), Stickel (1981), Stickel (1981) discovered independently the first AC-unification algorithms. They mainly differ in the way generalization is handled: M. Stickel's algorithm generalizes using variables and thus transforms an AC-equation into an homogeneous linear Diophantine equation, while M. Livesey and J. Siekmann consider certain variables as constants, so that an AC-equation is transformed into an inhomogeneous linear Diophantine equation. The termination of Stickel's algorithm in the presence of free symbols was proved by Fages (1984) almost ten years after its discovery. An extensive description of Stickel's algorithm, together with the use of constraints allowing to squeeze the search space is given by Hullot (1980).

More recently, Herold & Siekmann (1987) gave an improvement of Livesey-Siekmann's algorithm based on solving homogeneous and inhomogeneous linear Diophantine equations and on computing AC-unifiers from AC1-unifiers (AC1 denotes AC with an identity). A different approach to AC-unification based on solving systems of equations and on three main operations called decomposition, merging and AC-mutation has been proposed by Kirchner (1989). This allows one to unify in particular in the presence of several AC and free function symbols as an instance of a general approach to unification-algorithm combination (Kirchner (1985), Kirchner (1989)). Recently, (Boudet et al. 1990), Boudet (1990a) gives, in particular for AC-unification, a description of various controls on the transformations to be applied in order to solve a system efficiently. Using a different point of view, Lincoln & Christian (1990) gave a new AC-algorithm using a matrix with constraints which avoids solving linear Diophantine equations when the AC-problem consists only in linear equations. Let us finally mention the new algorithm proposed by Kapur & Narendran (1992).

Except Lincoln and Christian's, all these algorithms use linear (in)homogeneous Diophantine equation or system solving. Linear Diophantine equation solving has been studied in particular by Fortenbacher (1983), Clausen & Fortenbacher (1990), Huet (1978), Lambert (1987) and Lankford (1987). C. Kirchner's approach requires solving systems of linear homogeneous Diophantine equations. These systems are studied by Romeuf (1988), Pottier (1991), Contejean & Devie (1991), (Boudet et al. 1990) give a very nice generalization of Fortenbacher's algorithm to systems, and Domenjoud (1991) describes an original method, based on algebraic and geometric considerations, allowing us to test satisfiability and to directly compute the minimal solutions.

Because of its practical importance, particularly in theorem proving, a set of benchmarks was given in (Bürckert et al. 1988), in order to compare implementations and to stimulate the design and implementation of efficient AC-unification algorithms. This paper presents the implementation in the C language of the AC-unification algorithm based on solving systems of equations that has been first presented in Kirchner (1985), Kirchner (1989). In order to demonstrate the validity of our approach we give results on the benchmarks proposed in (Bürckert et al. 1988) and we give new benchmarks illuminating the behavior of the algorithm on systems of AC-equations.

As characteristic examples, let us consider the AC-symbol $f$ and the variables $x, u, v, w, y$. The unsatisfiable system:
is solved as a whole in less than 0.01 seconds by our implementation, whereas the sequential method, solving one equation after another, begins by solving the first equation which has more than a million minimal solutions and requires 424 seconds to be solved. In this case our approach demonstrates its efficiency and makes such difficult unification problem tractable.

For the system

\[ S' = \{ f(x, u) = f(y, v, r) \} \]

the situation is different. The system is quickly solved either using a sequential or system solving implementation. But the complete sets of AC-unifiers that are generated are quite different. For the sequential algorithm 45 solutions are generated, but solving the system as a whole returns a complete set of AC-unifiers consisting of only one element. The gain is now on the cardinality of the complete set of AC-unifiers generated and this is also of central importance for all the applications using AC-unification.

Our implementation, written in C, has been designed in a modular way in order to be easily reused in large software systems such as theorem provers or programming language interpreters. It is available at no cost from the authors as a part of a more general constraint solving environment called UNIF and currently encompasses also AC-matching and unification in finite algebras.

The paper is structured as follows. After this introduction, section 2 presents briefly the main operations needed for performing AC-unification. AC-mutation is presented in section 3 and section 4 is devoted to the solving of systems of Diophantine homogeneous and linear equations. The generation of complete set of AC-unifiers from the minimal solutions of Diophantine equations is recalled in section 5. The main internal features of our implementation are presented in section 6. The benchmarks are presented in section 7 and given in appendix. The interpretation of the results is given in section 8. We finally conclude in section 9.

2. AC-unification

Before presenting the AC-unification algorithm working on systems Kirchner (1989), we first present our notations: they are consistent with Jouannaud & Kirchner (1991) where the reader can find a general survey on unification. Given a set \( X \) of variables and a set \( F \) of function symbols, the algebra \( T(F, X) \) is the set of terms built over \( F \) and \( X \). A substitution is a mapping from variables to terms, which is extended to a total mapping on all terms in \( T(F, X) \). It is denoted by \((x_1 \mapsto t_1), \ldots, (x_n \mapsto t_n)\). A multiequation is a nonempty multiset of terms, a system of multiequations is a multiset of multiequations and a disjunction system is a multiset of systems of multiequations. A multiequation \( e = \{t_1, \ldots, t_m\} \) is also denoted by \( t_1 = \cdots = t_m \). We denote \( =_E \) the smallest congruence on the set of terms generated by a set of equational axioms \( E \).

For an equational theory \( E \), a substitution \( \sigma \) is an \( E \)-solution of:
a multiequation $e$ if $\sigma(t_1) =_{E} \sigma(t_2)$ for all $t_1$ and $t_2$ in $e$,
a system of multiequations $S$ if $\sigma$ is a solution of each multiequation in $S$,
a disjunction system $U$ if $\sigma$ is solution of, at least, one system in $U$.

In this paper, we are interested in associative-commutative theories consisting of the following axioms:

$$(x + y) + z = x + (y + z)$$
$$x + y = y + x$$

We consider that the set of symbols is such that $F = F_d \cup F_{AC}$ where $F_d$ is a set of free function symbols and $F_{AC}$ the set of AC-function symbols. In order to solve a system of multiequations, we introduce three main processes: decomposition which, when possible, simplifies multiequations without considering axioms, merging, which groups together the constraints on the same variable, and mutation, which transforms certain systems into disjunctions of systems using AC-axioms. This is an instance of a general schema presented in Kirchner (1985), generalized and intensively used in Jouannaud & Kirchner (1991). We now summarize the approach and develop the specific point of AC-mutation.

2.1. DECOMPOSITION

This transformation decomposes two terms having the same free head symbol into a system of multiequations using the following rule:

\[
\text{Decomposition: } \frac{r_1 = \ldots = r_m = f(t_1, \ldots, t_n) \neq f(t'_1, \ldots, t'_n)}{r_1 = \ldots = r_m = f(t_1, \ldots, t_n) \land (t_1 = t'_1) \land \ldots \land (t_n = t'_n)} \quad \text{if } f \in F_d
\]

with $r_i, t_j, t'_j$ in $T(F, X)$ and $m, n \geq 0$.

Clash of symbols brings us to failure:

\[
\text{Clash: } \frac{r_1 = \ldots = r_m = f(t_1, \ldots, t_n) = g(t'_1, \ldots, t'_n)}{f \neq g}
\]

Note that this last rule applies (undeterministically) for all symbols $f$ and $g$ regardless to their (AC) properties.

2.2. MERGING

This operation groups together constraints on the same variables:

\[
\text{Merging: } \frac{x = t_1 = \ldots = t_m \land x = t'_1 = \ldots = t'_n}{x = t_1 = \ldots = t_m = t'_1 = \ldots = t'_n} \quad \text{if } x \in X
\]

This allows us to postpone replacement until it is needed, as we will see later.

2.3. MUTATION

If there is no failure by the Clash rule (clash of symbols), decomposition and merging transformations yield three kinds of multiequations:
1. \( x_1 = \ldots = x_n \).
2. \( x_1 = \ldots = x_n = g(t_1, \ldots, t_m) \) such that \( g \in F_d \).
3. \( x_1 = \ldots = x_n = t_1 + t_2 + \ldots + t_m \) such that \( m \geq 1 \) and \( + \in F_{AC} \).

where the \( x_i \in X \) and \( t_j, t^k_i \in T(F, X) \). A system consisting of multiequations of type 1 and 2 is called a fully decomposed system. Since AC-theories are strict (or simple) Kirchner (1985), (Bürckert et al. 1989), solving such systems is straightforward. The problem is to transform a system of type 3 into an equivalent disjunction of fully decomposed systems (if it exists). This transformation is called AC-mutation and will be discussed below.

2.4. COMBINATION OF AC-THEORIES

It is important to be able to treat theories with several AC and free symbols. Based on previous works on combination of unification algorithms Kirchner (1985), Yelick (1985), Herold (1987), Schmidt-Schauß (1990), Boudet (1990a), one can build a mutation operation for a combination of AC-theories from the mutation operations of the elementary subtheories.

This is specially easy in the case of regular collapse free theories which is precisely what AC-theories are. In this case, let \( A_{+1}, \ldots, A_{+n} \) be \( n \) AC (elementary) theories and \( F_{AC} = \{ +_i | 1 \leq i \leq n \} \). The problem is to build an AC-unification algorithm for the theory \( A = A_{+1} \cup \cdots \cup A_{+n} \) when unification algorithms are given for each AC-theory \( A_{+i} \) (\( i \in [1..n] \)). This is achieved by extracting from a system \( S \) built over \( T(F_d \cup F_{AC}, X) \) all maximal (for the subset relation) pure subsystems: \( S_{+i} \) (\( 1 \leq i \leq n \)) such that \( S_{+i} \) is a submultiset of \( S \) built over \( +_i \) and variables.

Since this is fully described in Kirchner (1989), let us here give an example:

**Example 2.1.** Let \( n = 2, +_1 = + \) and \( +_2 = * \).

The system:

\[
\begin{align*}
  x \ast y &= a \ast (z + u) \\
  x + y &= v + a \\
  v &= a + b
\end{align*}
\]

is generalized in the system \( S = \{ x \ast y = x_1 \ast x_2, x_1 = a, x_2 = z + u, x + y = v + x_1, v = x_1 + x_3, x_3 = b \} \) with \( S_+ = \{ x_2 = z + u, x + y = v + x_1, v = x_1 + x_3 \} \) and \( S_- = \{ x \ast y = x_1 \ast x_2 \} \).

Given an AC-mutation algorithm for each AC-theory \( A_{+_i} \) (\( i \in [1..n] \)) (this will be the subject of the next section), the algorithm **FULL-DEC** described in Figure 1 proposes a control to transform any system in the AC-theory \( A = A_{+_1} \cup \cdots \cup A_{+_n} \) with free symbols, into a fully decomposed disjunction system. A rule-based description of the actions of this algorithm is given in Boudet (1990b) and Jouannaud & Kirchner (1991). A more general class of controls, suitable in particular for AC-unification is given by Boudet (1990b).

We will carry out the following example in the rest of this paper.
FULL-DEC \((S : \text{a system of } T(F_d \cup F_{AC}, X))\)

Transform \(S\) into a system \(S'\) by the Decomposition and Merging rules.

\begin{verbatim}
if the Clash rule applies
then returns(no solution).
else Let \(S_+\) be a nonfully decomposed subsystem of \(S\)
if such a system does not exist
then returns(S)
else if \(x_1 = t_1 \cdots = x_n = t_n \in S\)
then
- replace all occurrences of variables \(x_k (k \in [1..n])\) in \(S_+\), by \(x\)
- \([R_j]_{j \in J} \gets \text{AC-mutation}(S_+)\)
- \(R_j' \gets (S - S_+) \cup R_j\) for all \(j \in J\)
[returns(FULL-DEC \((R_j')\))]_{j \in J}.
\end{verbatim}

END FULL-DEC

Figure 1. Full decomposition in AC-theories

\textbf{Example 2.2.} Let \(+, *\) be in \(F_{AC}\), \(f\) in \(F_d\) and \(S\) the system to be solved:

\[
S = \begin{cases} 
  r & = f(x + y, z * w) \\
  r & = f(u + v, t * q) \\
  x + v & = u + u
\end{cases}
\]

Normalization using the merging and decomposition rules on the subsystem:

\[
\begin{cases} 
  r & = f(x + y, z * w) \\
  r & = f(u + v, t * q)
\end{cases}
\]
yields the equivalent system:

\[
\begin{cases} 
  r & = f(x + y, z * w) \\
  x + y & = u + v \\
  z * w & = t * q
\end{cases}
\]

\(S\) is then transformed into:

\[
S' = \begin{cases} 
  r & = f(x + y, z * w) \\
  x + y & = u + v \\
  x + v & = u + u \\
  z * w & = t * q
\end{cases}
\]

with \(S_+ = \begin{cases} 
  x + y & = u + v \\
  x + v & = u + u
\end{cases}\)
and \(S_* = \begin{cases} 
  z * w & = t * q
\end{cases}\).

In the section on AC-mutation, we will show how to solve \(S_+\) and \(S_*\).

\textbf{2.5. Detection of cycles}

The last step of our algorithm consists in checking if a fully decomposed system contains a system \(x_1 = t_1, \cdots, x_n = t_n\) such that: \(x_i \in Var(t_{i-1})\) for \(i\) such that \((2 \leq i \leq n)\) and, \(x_1 \in t_n\). In this case, the system does not have a finite AC-unifier.
EXAMPLE 2.3. *The fully decomposed system:* 

\[
\begin{align*}
  x &= f(y, z) \\
  y &= g(x) \\
  z &= h(t, u)
\end{align*}
\]

contains the cycle \( x = f(y, z), y = g(x) \).

3. **AC-mutation**

As we have seen in the previous section, it is sufficient to know how to compute the mutation operation for a pure AC-system, that is a system built over variables and only one AC-operator. We summarize in this section how this mutation is performed in our implementation.

We consider systems, called AC-systems, of the form:

\[
(t_1^k + t_2^k = r_1^k + r_2^k)_{k \in [1..q]}
\]

in \( T(\{+\}, X) \) where \(+\) is an AC-function symbol. First, an AC-system is flattened, simplified and transformed into a system of homogeneous linear Diophantine equations (in short Diophantine system) which we know how to solve.

Then, the minimal solutions of the initial AC-system are computed as certain combinations of the Diophantine system minimal solutions, as explained in Section 5.

3.1. **FLATTENING**

The flattened form of a term \( t \) in \( T(\{+\}, X) \) is \( x_1 * x_2 * \cdots * x_n \) where the \( x_i(i \in [1, n]) \) are the variables occurring in \( t \). Formally the flattened form of the term \( t \) is defined as

\[
FF(t_1 + t_2) = FF(t_1) * FF(t_2) \quad \text{and} \quad FF(x) = x \text{ if } x \text{ is a variable.}
\]

Note that \((X, *)\) is a commutative monoid whose identity is denoted \( \lambda \). For example, \( x * x * z * w \) is the flattened form of \( x + ((x + z) + w) \).

3.2. **TRANSLATION TO DIOPHANTINE SYSTEM**

After flattening all the equations of the AC-system, each equation is simplified by eliminating variables which appear in both its left and right hand side and by computing coefficients as follows. Let \( S \) be a flattened and simplified AC-system with equations of the form:

\[
x_1 * \cdots * x_m = y_1 * \cdots * y_p
\]

where the \( x_i, y_j \) are variables from \( X \).

Since \( * \) is associative-commutative, \( S \) can also be written:

\[
S = (a_1^k x_1 * \cdots * a_m^k x_m = b_1^k y_1 * \cdots * b_p^k y_p)_{k \in [1..q]}
\]

where \( a_i^k \) stands for \( x * \cdots * x \) and for all \( k \in [1..q] \), the coefficients \( (a_i^k)_{i \in [1..m]} \) and \( (b_j^k)_{j \in [1..p]} \) are natural numbers.

In Kirchner (1989), Kirchner (1985) it has been shown that solving such AC-systems...
can be reduced to solving systems of homogeneous linear Diophantine equation whose coefficients are \((a_i^j)_{i \in \{1..m\}}\) and \((-b_j^i)_{j \in \{1..p\}}\) and then combining their minimal solutions. In other words, solving linear Diophantine systems of equations is only one part of the problem. The second important one is to properly combine its solutions. Here, we just show the transformation on the following example.

**EXAMPLE 3.1. (continued)**

Let us consider the \(AC\)-system \(S_+\) given before:

\[
\begin{align*}
  x + y &= u + v \\
  x + v &= u + u
\end{align*}
\]

Then the flattened and simplified \(AC\)-system with coefficients is:

\[
\begin{align*}
  1x * 1y &= 1u * 1v \\
  1x * 1v &= 2u
\end{align*}
\]

and the Diophantine system can easily be deduced from the previous system.

\[
\begin{align*}
  1x_{nat} + 1y_{nat} - 1u_{nat} - v_{nat} &= 0 \\
  1x_{nat} - 2u_{nat} + v_{nat} &= 0
\end{align*}
\]

where the variables \(x_{nat}, y_{nat}, u_{nat}, v_{nat}\) range over the set of natural numbers \(N\).

---

4. Solving systems of homogeneous and linear Diophantine equations

We solve the Diophantine system as in real or rational vector spaces using the Gaussian elimination method. Recently, direct solving of systems of two Diophantine equations Romeuf (1989) or of an arbitrary number of Diophantine equations Contejean & Devie (1991), Domenjoud (1991), Pottier (1991), have been proposed. They are in general quite more efficient than the method described below that we give here since it is easy to understand and is the first one we implemented. It has been given first and independently in Adi (1988) and Kapur (1989).

Let \(AX = 0\) be the Diophantine system to be solved where \(A = (a_{ij})_{i \in \{1..n\}}\) is an \(m \times n\) integer matrix and \(X = (x_i)_{i \in \{1..m\}}\) are the distinct variables of the system.

The first step consists in triangulating the matrix \(A\) which depends on the values of \(m\) and \(n\). In other words, after applying the following algorithm on the matrix:

\[
\begin{align*}
  &\text{for } i = 1 \text{ to } m - 1 \\
  &\quad \text{for } k = i + 1 \text{ to } m \\
  &\quad \quad \text{for } j = i + 1 \text{ to } n \\
  &\quad \quad \quad a_{kj} = a_{ii}.a_{kj} - a_{ki}.a_{ij} \\
  &\quad \text{end for} \\
  &\text{end for} \\
  &\text{end for}
\end{align*}
\]
we get a matrix of the form:

$$M = \begin{cases} 
  b_{11}x_1 + b_{12}x_2 + b_{13}x_3 + \cdots + b_{1n}x_n = 0 \\
  0 + b_{22}x_2 + b_{23}x_3 + \cdots + b_{2n}x_n = 0 \\
  \cdots \\
  \begin{cases} 
  b_{nn}x_n = 0 \\
  b_{mn}x_n = 0 \\
  \end{cases} \\
  \begin{cases} 
  i f ~ m \geq n \\
  i f ~ m < n \\
  \end{cases} 
\end{cases}$$

(1) \quad (2) 

It is easy to check that the system associated with the matrix $M$ does not have a non-trivial positive solution for $m \geq n$ with at least $b_{in} \neq 0 \ (i \in [1..m])$. But if $b_{nn}, \ldots, b_{mn}$ are null, the system is reduced to $n-1$ equations by eliminating the $m-n+1$ last equations.

Henceforth, we suppose that $m < n$ and start to solve the last equation $(m)$, which has less variables. In order to solve this equation, the variables with negative coefficients are transferred to the other side of the equation to obtain a Diophantine equation. If there is no minimal solution to this equation, the process is stopped with no solutions to the system.

Let $s_1, \ldots, s_k \in N^{n-m+1}$ be the minimal solutions of the Diophantine equation. A solution has the general form:

$$(x_m, \ldots, x_n) = \sum_{j=1}^{k} y_j s_j$$

where $y_j \in N$.

Hence the $i^{th}$ element of the vector $(x_m, \ldots, x_n)$ can be written:

$$x_i = \left( \sum_{j=1}^{k} y_j s_j \right)_{i-m+1} = \sum_{j=1}^{k} y_j (s_j)_{i-m+1} \quad i \in [m..n]$$

Elsewhere, we have for each equation $l \ (l = 1, \ldots, m-1)$:

$$\sum_{i=l}^{m-1} b_{li} x_i = - \sum_{i=m}^{n} b_{li} x_i.$$  

$$= - \sum_{i=m}^{n} b_{li} (\sum_{j=1}^{k} y_j (s_j)_{i-m+1})$$

$$= - \sum_{i=m}^{n} b_{li} \sum_{j=1}^{k} y_j (s_j)_{i-m+1}$$

$$= - \sum_{i=m}^{n} \sum_{j=1}^{k} b_{li} (s_j)_{i-m+1}$$

$$= - \sum_{i=m}^{n} \sum_{j=1}^{k} y_j (b_{li} (s_j)_{i-m+1})$$

$$= - \sum_{j=1}^{k} \sum_{i=m}^{n} y_j (b_{li} (s_j)_{i-m+1})$$

$$= - \sum_{j=1}^{k} y_j (\sum_{i=m}^{n} b_{li} (s_j)_{i-m+1})$$

We get a system with $m-1$ equations and $m+k-1$ variables $(x_1, \ldots, x_{m-1}, y_1, \ldots, y_k)$ whose matrix is already triangulated. Then we immediately solve the last Diophantine
equation:

\[ b_{m-1,m-1}x_{m-1} + \sum_{j=1}^{k} y_j \left( \sum_{i=m}^{n} \frac{b_{mi}}{s_j} \right) x_{i-m+1} = 0. \]

The process is performed until all equations are exhausted. Then the solutions can be easily deduced from variables \( y_1, \ldots, y_k \) by the relation \((x_m, \ldots, x_n) = \sum_{j=1}^{k} y_j s_j \) and \( x_1, \ldots, x_{m-1} \) from \( x_m, \ldots, x_n \) by the triangulated matrix. Using this process, the set of solutions obtained is not minimal. Thus, at the end of the process, all non minimal solutions are eliminated to get the minimal solution set.

**EXAMPLE 4.1. (continued)**

We solve the system given in the previous example.

\[
\begin{align*}
  x_1 + x_2 - x_3 - x_4 &= 0 \\
  x_1 - 2x_3 + x_4 &= 0 \\
\end{align*}
\]

\[ \iff \]

\[
\begin{align*}
  x_1 &= -x_2 + x_3 + x_4 \\
  x_2 + x_3 &= 2x_4 \\
\end{align*}
\]

There are three minimal solutions for the second equation:

<table>
<thead>
<tr>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

The first equation can be written with only variables \( x_1, y_1, y_2, y_3 \).

\[
\begin{align*}
  x_1 &= -x_2 + x_3 + x_4 \\
  &= -(2y_1 + 1y_2 + 0y_3) + (0y_1 + 1y_2 + 2y_3) + (1y_1 + 1y_2 + 1y_3) \\
  &= -y_1 + y_2 + 3y_3 \\
\end{align*}
\]

We search for the minimal solutions of Diophantine equation \( x_1 + y_1 = y_2 + 3y_3 \) and get:

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( y_1 )</th>
<th>( y_2 )</th>
<th>( y_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

At the end, we compute the values of \( x_2, x_3, x_4 \) from those of \( y_1, y_2, y_3 \) to obtain nonminimal solutions of the system.

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Three nonminimal solutions are eliminated to get the minimal solutions set.

\[
x_1 \quad x_2 \quad x_3 \quad x_4 \\
0 \quad 3 \quad 1 \quad 2 \\
1 \quad 1 \quad 1 \quad 1 \\
3 \quad 0 \quad 2 \quad 1
\]

This process terminates, since the number of equations decreases at each step and since the set of minimal solutions of a homogeneous linear Diophantine equation is finite.

5. Combining solutions of Diophantine systems

Combining solutions is the other expensive part of the AC-mutation. This allows to compute the AC-solutions from the minimal solutions of the Diophantine system.

Let \( s_1, s_2, \ldots, s_k \) be the minimal solutions of the Diophantine system such that \( s_i = (s_{i1}, \ldots, s_{in}) \) where \( n \) corresponds to terms \( t_1, t_2, \ldots, t_n \) of the AC-system. New variables \( z_1, z_2, \ldots, z_k \) are associated to \( s_1, s_2, \ldots, s_k \).

\[
\begin{align*}
0 & \quad \ldots & \quad t_n \\
s_{11} & \quad \ldots & \quad s_{1n} & \quad z_1 \\
s_{21} & \quad \ldots & \quad s_{2n} & \quad z_2 \\
& \quad \ldots & \quad & \\
s_{k1} & \quad \ldots & \quad s_{kn} & \quad z_k
\end{align*}
\]

**Example 5.1.** (Continued)

\[
x_1 \quad x_2 \quad x_3 \quad x_4 \\
s_1 \quad 0 \quad 3 \quad 1 \quad 2 \quad z_1 \\
s_2 \quad 1 \quad 1 \quad 1 \quad 1 \quad z_2 \\
s_3 \quad 3 \quad 0 \quad 2 \quad 1 \quad z_3
\]

where \( t_i = x_i \quad (i = 1, 2, 3, 4) \). Then \( s_2 \) is a strictly positive solution and corresponds to the system:

\[
\begin{align*}
x_1 & = z_2 \\
x_2 & = z_2 \\
x_3 & = z_2 \\
x_4 & = z_2
\end{align*}
\]

But \( s_1 \) is not a strictly positive solution. Hence, \( x_1 = \lambda \), where \( \lambda \) satisfies \( \lambda + t = t + \lambda = t \), which is not allowed in AC-unification (this is a main difference with AC1-unification). This brings to the combination of such solutions with others in such a way that none of the components of a solution is empty.

There are \( 2^k \) possibilities of combination. Now for every possibility, we are testing if \( w_1.s_1 + \cdots + w_k.s_k \) is a strictly positive vector. If this is the case, we form the AC-system:

\[
(t_i = w_1.s_{i1}.z_1 + \cdots + w_k.s_{ik}.z_k)_{i \in [1..n]}
\]
where $+$ is an AC-function symbol and

$$w_j.s_j^i.z_j = \begin{cases} 
z_j + \cdots + z_j & \text{if } w_j = 1. \\
\lambda & \text{if } w_j = 0.
\end{cases}$$

The system $(t_i = w_1.s_1^i.z_1 + \cdots + w_k.s_k^i.z_k)_{i \in [1..n]}$ is decomposed, merged and AC-mutated again until we have a disjunction of fully decomposed system (which may be empty). Finally, the set of substitutions associated to all fully decomposed systems is a complete set of AC-unifiers (very often, it is the minimal set). Combining solutions of Diophantine systems is described in details for example in Adi (1991).

6. Implementation

First of all, let us outline the data structures of term, equation and system enabling us to express efficiently the transformation rules and AC-mutation given in sections 2 and 3.

A term is represented by a tree with some additional information about its type. Nodes of trees are accessed using pointers in order to have efficient operation for:

- flattening terms,
- merging and splitting terms,
- extraction and insertion of subterms,
- searching for variables and constants in terms.

A multiequation is represented as 3-tuple $e =< Ve, Me, De >$ consisting of three lists; a list $Ve$ of its variable, a list $Me$ of its terms with AC head symbols and a list $De$ of its non variable terms with free head symbols. This structure exploits the fact that AC-mutation, decomposition and merging operations are independently performed. Notice that under some constraints, we can perform decomposition and mutation in parallel on the lists of this structure. Finally, a system is simply represented as a list of multiequations. All these structures can be easily modified to incorporate additional features if needed by the user application.

Of course, data structures alone do not ensure efficient implementation. Many other ideas are used, for example the technique for computing the correct combinations of the diophantine solutions, which is one of the most expensive parts of the algorithm. Instead of working on the matrix of diophantine solutions, we work on a matrix of bits having the same dimensions. The result of a logical “and” between lines considered in this matrix is a word of $n$ bits where $n$ is the numbers of columns of the matrix. Since we search for a strictly positive diophantine solution, we have to test if this word, considered as a binary number, is equal to $2^n - 1$. In this case we construct the corresponding system of multiequations. The inspiration to use this bit representation comes from Alexandre Boudet who proposed an implementation of AC-unification Boudet (1991) based on solving diophantine systems using the approach of Contejean & Devie (1991), (Boudet et al. 1990).

Since AC-theories are strict, the detection of cycles is done on fully decomposed systems as follows: we first test if there is no variable occurring both in the left and right hand side of the system (which is often the case), otherwise we use the standard method based on topological sort as described, for example, in Kirchner (1989).
Several reasons lead us to choose the language C. The main ones are its portability, its flexibility and its efficiency. Moreover, since reusability is also one of our requirements, C is clearly the simplest and most universal choice even if currently C++ may be now a good choice too. For example, our implementation is used from the CAML environment in the software system ORME developed by P. Lescanne in order to perform AC-completion of term rewriting systems Lescanne (1990).

The choice of C has of course some drawbacks that have to be managed: since we are using dynamic structures, an appropriate garbage collector has to be implemented. However, we implement a collector mechanism allowing us to reuse the structures without recreating them when possible. The memory allocated to structures for building a first unifier is reused for a second one after keeping them in appropriate lists of available terms, equations and systems. This allows to avoid rebuilding the appropriate structure and in reusing the appropriate structure one has only to fill in the appropriate fields. Of course if one of these lists is void, we have to allocate the necessary memory and to keep it in appropriate lists for the next unifier.

Some implementations of AC-unification make the assumption that the application calling it is responsible for building the terms and substitutions from informations provided by the AC-unification program. On the contrary, in our current implementation the unifiers are indeed constructed, thus enabling the user to use directly (and maybe destructively), without further processing, the results (i.e. the unifiers) in its application.

7. Benchmarks

To test our implementation, in addition to Stickel’s classical algorithm we have chosen to implement the recent algorithm of Lincoln & Christian (1990) which obviates the need to solve homogeneous linear Diophantine equations for linear equations.

The tables presented in the appendix gives the performance of our implementation of C. Kirchner’s algorithm Kirchner (1989), Kirchner (1985) (column $CK$), M. Stickel’s algorithm Stickel (1975), Stickel (1981) (column $MS$) and Christian and Lincoln’s algorithm Lincoln & Christian (1990) (column $CL$). The first and second column give the unification problems as they appear in (Burckert et al. 1988) ($f$ and $g$ are AC-function symbols, $p, q, r, \ldots, z$ are variables and $a, b, c, d, e$ are constants, terms are written in flattened form). The $\#$ columns give the cardinality of the complete set of AC-unifiers that is obtained. For problem limited to one AC-equation there is the same number of solutions for the three approaches and same running time for $CK$ and $MS$. Times are given in seconds on a Sun Sparc2 workstation with 28 Mbytes of RAM.

8. Interpretation

Considering the result of the benchmarks and of our experiments, how does the system-solving approach compare with previously proposed equation-based approaches?

The most obvious improvement concerns the efficiency of non-trivial AC-systems solving. First on running time: for example, $CK$ requires 0.9 seconds for computing a complete set of solution for acuni-S9 while $MS$ and $CL$ systems require respectively 10.1 and 10.8 seconds. The second main efficiency improvement (and may be the most important one) is due to the fact that the system solving approach delivers complete set of
AC-unifiers that is smaller than the standard approach. The system acuni-S20:

\[
\begin{align*}
    f(x, y) &= f(u, v, q) \\
    f(x, u, z) &= f(y, v, r)
\end{align*}
\]

is typical of this situation where the sequential approach provides 6839 solutions with at least 6732 unnecessary non minimal solutions.

The second major improvement concerns feasibility. For problems like acuni-SS5, only 0.02 seconds is needed for CK but for solving only the first equation, 424 seconds are needed for MS, and the other equations are still to be solved after replacement. Now, if we consider the acuni-SS6 and acuni-SS7 examples which take at most 0.03 seconds, a sequential method will start to solve the first equation which is proved by Domenjoud (1989) to have 34.359.607.481 minimal unifiers, and thus the resulting system is clearly untractable using today's computers.

Another advantage is that the (minimal) solutions of a system are computed directly. In a sequential approach, intermediate terms have to be built and are only used to compute the solutions of a derived equation. Then they are thrown away, implying the need for more memory management and garbage collection.

Currently the most unsatisfactory part of the system solving approach concern systems where no or few variables are shared between the equations. If we take acuni-S16:

\[
\begin{align*}
    f(x, y) &= f(u, v) \\
    f(x, z, t) &= f(r, s, p)
\end{align*}
\]

a closer look to the different tasks of CK shows that it takes 3 seconds to test unnecessary combinations because of the sparse matrix solution of the Diophantine system.

Finally let us note that the order in which equations are solved is quite important in MS and CL: acuni-S11 and acuni-S12 are the same system but don't yield the same complete set of solutions and running time.

9. Conclusion

We have implemented the AC-unification algorithms proposed by Jim Christian and Patrick Lincoln, by Claude Kirchner and by Mark Stickel. We have also proposed benchmarks for testing unification of systems of AC-equations.

Our conclusion is that the system-solving approach is in all cases clearly more efficient in time, space and in the size of the complete set of unifiers generated. But this can still be improved since for systems sharing few or no variables, the matrix of minimal solutions of the associated Diophantine system is sparse. In this case a lot of combinations of minimal Diophantine solutions are unnecessary. We are currently improving the combination algorithm in such a way that it detects, a priori and as much as possible, these unnecessary combinations.

Acknowledgments: This work has been partly supported by the "GRECO de programmation du CNRS". We gratefully acknowledge Alexandre Boudet, Nachum Dershowitz, Eric Domenjoud, Hélène Kirchner, Pierre Lescanne, Denis Lugiez and René Schott for many interesting and fruitful discussions or comments about this paper. We also acknowledge interesting and constructive remarks from the anonymous referees.
10. Benchmark tables

The \#DS: num comment in the tables below means that one of the diophantine equation system needed for solving the AC-system has num minimal solutions. When num \( \geq 20 \), computing all the \( 2^{num} \) different combinations is quite expansive and can be done only on request. The time indication in this case corresponds to the elapsed time until the process has encountered this problem and has been stopped.

Let us first begin with benchmarks on AC-equations:

<table>
<thead>
<tr>
<th>Example</th>
<th>Problem</th>
<th>#</th>
<th>CK or MS</th>
<th>CL</th>
</tr>
</thead>
<tbody>
<tr>
<td>acuni-1</td>
<td>( f(x, a, b) = f(u, c, c, c) )</td>
<td>2</td>
<td>0.017</td>
<td>0.017</td>
</tr>
<tr>
<td>acuni-2</td>
<td>( f(x, a, b) = f(u, c, c, d) )</td>
<td>2</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>acuni-3</td>
<td>( f(x, a, b) = f(u, c, c, c) )</td>
<td>2</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>acuni-4</td>
<td>( f(x, a, b) = f(u, v, c, d) )</td>
<td>12</td>
<td>0.050</td>
<td>0.017</td>
</tr>
<tr>
<td>acuni-5</td>
<td>( f(x, a, b) = f(u, c, c) )</td>
<td>12</td>
<td>0.017</td>
<td>0.017</td>
</tr>
<tr>
<td>acuni-6</td>
<td>( f(x, a, b) = f(u, v, w, c) )</td>
<td>30</td>
<td>0.117</td>
<td>0.050</td>
</tr>
<tr>
<td>acuni-7</td>
<td>( f(x, a, b) = f(u, v, w, t) )</td>
<td>56</td>
<td>0.467</td>
<td>0.167</td>
</tr>
<tr>
<td>acuni-8</td>
<td>( f(x, a, b) = f(u, u, c, d) )</td>
<td>2</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-9</td>
<td>( f(x, a, b) = f(u, u, c, c) )</td>
<td>2</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-10</td>
<td>( f(x, a, b) = f(u, u, v, c) )</td>
<td>12</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-11</td>
<td>( f(x, a, b) = f(u, v, u, w) )</td>
<td>30</td>
<td>0.067</td>
<td>-</td>
</tr>
<tr>
<td>acuni-12</td>
<td>( f(x, a, b) = f(u, u, v, v) )</td>
<td>12</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-13</td>
<td>( f(x, a, b) = f(u, u, u, c) )</td>
<td>2</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-14</td>
<td>( f(x, a, b) = f(u, u, u, v) )</td>
<td>12</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-15</td>
<td>( f(x, a, b) = f(u, u, u, u) )</td>
<td>2</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-16</td>
<td>( f(x, a, a) = f(u, c, d, e) )</td>
<td>2</td>
<td>0.017</td>
<td>0.001</td>
</tr>
<tr>
<td>acuni-17</td>
<td>( f(x, a, a) = f(u, c, c, d) )</td>
<td>2</td>
<td>0.017</td>
<td>0.017</td>
</tr>
<tr>
<td>acuni-18</td>
<td>( f(x, a, a) = f(u, c, c, c) )</td>
<td>2</td>
<td>0.017</td>
<td>0.001</td>
</tr>
<tr>
<td>acuni-19</td>
<td>( f(x, a, a) = f(u, v, c, d) )</td>
<td>8</td>
<td>0.017</td>
<td>0.017</td>
</tr>
<tr>
<td>acuni-20</td>
<td>( f(x, a, a) = f(u, v, c, c) )</td>
<td>8</td>
<td>0.017</td>
<td>0.001</td>
</tr>
<tr>
<td>acuni-21</td>
<td>( f(x, a, a) = f(u, v, w, c) )</td>
<td>18</td>
<td>0.100</td>
<td>0.033</td>
</tr>
<tr>
<td>acuni-22</td>
<td>( f(x, a, a) = f(u, v, w, t) )</td>
<td>32</td>
<td>1.550</td>
<td>0.150</td>
</tr>
<tr>
<td>acuni-23</td>
<td>( f(x, a, a) = f(u, u, c, d) )</td>
<td>2</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-24</td>
<td>( f(x, a, a) = f(u, u, c, c) )</td>
<td>2</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-25</td>
<td>( f(x, a, a) = f(u, u, v, c) )</td>
<td>4</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-26</td>
<td>( f(x, a, a) = f(u, u, v, w) )</td>
<td>10</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-27</td>
<td>( f(x, a, a) = f(u, u, v, v) )</td>
<td>4</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-28</td>
<td>( f(x, a, a) = f(u, u, u, c) )</td>
<td>2</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-29</td>
<td>( f(x, a, a) = f(u, u, u, v) )</td>
<td>4</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-30</td>
<td>( f(x, a, a) = f(u, u, u, u) )</td>
<td>2</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-31</td>
<td>( f(x, y, a) = f(u, c, d, e) )</td>
<td>28</td>
<td>0.067</td>
<td>0.033</td>
</tr>
<tr>
<td>acuni-32</td>
<td>( f(x, y, a) = f(u, c, c, d) )</td>
<td>20</td>
<td>0.050</td>
<td>0.033</td>
</tr>
<tr>
<td>acuni-33</td>
<td>( f(x, y, a) = f(u, c, c, c) )</td>
<td>12</td>
<td>0.033</td>
<td>0.017</td>
</tr>
<tr>
<td>acuni-34</td>
<td>( f(x, y, a) = f(u, v, c, d) )</td>
<td>88</td>
<td>0.200</td>
<td>0.117</td>
</tr>
<tr>
<td>acuni-35</td>
<td>( f(x, y, a) = f(u, v, c, c) )</td>
<td>64</td>
<td>0.117</td>
<td>0.100</td>
</tr>
<tr>
<td>acuni-36</td>
<td>( f(x, y, a) = f(u, v, w, c) )</td>
<td>204</td>
<td>0.467</td>
<td>0.300</td>
</tr>
<tr>
<td>acuni-37</td>
<td>( f(x, y, a) = f(u, v, w, t) )</td>
<td>416</td>
<td>0.833</td>
<td>0.750</td>
</tr>
<tr>
<td>acuni-38</td>
<td>( f(x, y, a) = f(u, u, c, d) )</td>
<td>60</td>
<td>0.100</td>
<td>-</td>
</tr>
<tr>
<td>acuni-39</td>
<td>( f(x, y, a) = f(u, u, c, c) )</td>
<td>44</td>
<td>0.067</td>
<td>-</td>
</tr>
<tr>
<td>acuni-40</td>
<td>( f(x, y, a) = f(u, u, v, c) )</td>
<td>144</td>
<td>0.233</td>
<td>-</td>
</tr>
<tr>
<td>acuni-41</td>
<td>( f(x, y, a) = f(u, u, v, w) )</td>
<td>300</td>
<td>0.617</td>
<td>-</td>
</tr>
<tr>
<td>acuni-42</td>
<td>( f(x, y, a) = f(u, u, v) )</td>
<td>216</td>
<td>0.467</td>
<td>-</td>
</tr>
<tr>
<td>acuni-43</td>
<td>( f(x, y, a) = f(u, u, u, c) )</td>
<td>92</td>
<td>0.117</td>
<td>-</td>
</tr>
<tr>
<td>acuni-44</td>
<td>( f(x, y, a) = f(u, u, u, v) )</td>
<td>196</td>
<td>0.800</td>
<td>-</td>
</tr>
<tr>
<td>acuni-45</td>
<td>( f(x, y, a) = f(u, u, u, u) )</td>
<td>124</td>
<td>2.517</td>
<td>-</td>
</tr>
<tr>
<td>acuni-46</td>
<td>( f(x, y, z) = f(u, c, d, e) )</td>
<td>120</td>
<td>0.550</td>
<td>0.317</td>
</tr>
<tr>
<td>acuni-47</td>
<td>( f(x, y, z) = f(u, c, d, e) )</td>
<td>75</td>
<td>0.433</td>
<td>0.283</td>
</tr>
<tr>
<td>acuni-48</td>
<td>( f(x, y, z) = f(u, c, c, c) )</td>
<td>37</td>
<td>0.700</td>
<td>0.267</td>
</tr>
<tr>
<td>acuni-49</td>
<td>( f(x, y, z) = f(u, v, c, d) )</td>
<td>336</td>
<td>0.850</td>
<td>0.583</td>
</tr>
<tr>
<td>acuni-50</td>
<td>( f(x, y, z) = f(u, v, c, c) )</td>
<td>216</td>
<td>0.533</td>
<td>0.467</td>
</tr>
<tr>
<td>acuni-51</td>
<td>( f(x, y, z) = f(u, v, w, c) )</td>
<td>870</td>
<td>1.367</td>
<td>1.450</td>
</tr>
<tr>
<td>acuni-52</td>
<td>( f(x, y, z) = f(u, v, w, t) )</td>
<td>2161</td>
<td>1.450</td>
<td>4.167</td>
</tr>
<tr>
<td>acuni-53</td>
<td>( f(x, y, z) = f(u, v, c, d) )</td>
<td>486</td>
<td>0.917</td>
<td>-</td>
</tr>
<tr>
<td>acuni-54</td>
<td>( f(x, y, z) = f(u, v, w, t) )</td>
<td>318</td>
<td>0.567</td>
<td>-</td>
</tr>
<tr>
<td>acuni-55</td>
<td>( f(x, y, z) = f(u, v, c, c) )</td>
<td>1200</td>
<td>1.587</td>
<td>-</td>
</tr>
<tr>
<td>acuni-56</td>
<td>( f(x, y, z) = f(u, u, v, w) )</td>
<td>2901</td>
<td>1.633</td>
<td>-</td>
</tr>
<tr>
<td>acuni-57</td>
<td>( f(x, y, z) = f(u, u, v, v) )</td>
<td>3825</td>
<td>1.800</td>
<td>-</td>
</tr>
<tr>
<td>acuni-58</td>
<td>( f(x, y, z) = f(u, u, u, c) )</td>
<td>2982</td>
<td>3.867</td>
<td>-</td>
</tr>
<tr>
<td>acuni-59</td>
<td>( f(x, y, z) = f(u, u, u, u) )</td>
<td>7029</td>
<td>3.517</td>
<td>-</td>
</tr>
<tr>
<td>acuni-60</td>
<td>( f(x, y, z) = f(u, u, u, u) )</td>
<td>32677</td>
<td>15.100</td>
<td>-</td>
</tr>
<tr>
<td>acuni-61</td>
<td>( f(x, z, a) = f(u, c, c, d) )</td>
<td>2</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-62</td>
<td>( f(x, z, a) = f(u, c, c, d) )</td>
<td>2</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-63</td>
<td>( f(x, z, a) = f(u, c, c, c) )</td>
<td>2</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-64</td>
<td>( f(x, z, a) = f(u, u, v, c) )</td>
<td>60</td>
<td>0.100</td>
<td>-</td>
</tr>
<tr>
<td>acuni-65</td>
<td>( f(x, z, a) = f(u, v, c, c) )</td>
<td>12</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-66</td>
<td>( f(x, z, a) = f(u, v, w, c) )</td>
<td>486</td>
<td>0.900</td>
<td>-</td>
</tr>
<tr>
<td>acuni-67</td>
<td>( f(x, z, a) = f(u, v, w, t) )</td>
<td>3416</td>
<td>5.350</td>
<td>-</td>
</tr>
<tr>
<td>acuni-68</td>
<td>( f(x, z, a) = f(u, u, v, c) )</td>
<td>0</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-69</td>
<td>( f(x, z, a) = f(u, u, u, c) )</td>
<td>0</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-70</td>
<td>( f(x, z, a) = f(u, u, u, v) )</td>
<td>2</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-71</td>
<td>( f(x, z, a) = f(u, u, u, u) )</td>
<td>12</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-72</td>
<td>( f(x, z, a) = f(u, u, v, c) )</td>
<td>0</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-73</td>
<td>( f(x, z, a) = f(u, u, u, c) )</td>
<td>2</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-74</td>
<td>( f(x, z, a) = f(u, u, u, u) )</td>
<td>12</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-75</td>
<td>( f(x, z, a) = f(u, u, u, c) )</td>
<td>0</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-76</td>
<td>( f(x, z, y) = f(u, c, c, c) )</td>
<td>28</td>
<td>0.050</td>
<td>-</td>
</tr>
<tr>
<td>acuni-77</td>
<td>( f(x, z, y) = f(u, c, c, d) )</td>
<td>11</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-78</td>
<td>( f(x, z, y) = f(u, c, c, c) )</td>
<td>7</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-79</td>
<td>( f(x, z, y) = f(u, v, c, c) )</td>
<td>228</td>
<td>0.417</td>
<td>-</td>
</tr>
<tr>
<td>acuni-80</td>
<td>( f(x, z, y) = f(u, v, c, d) )</td>
<td>44</td>
<td>0.050</td>
<td>-</td>
</tr>
<tr>
<td>acuni-81</td>
<td>( f(x, z, y) = f(u, v, w, c) )</td>
<td>1632</td>
<td>3.283</td>
<td>-</td>
</tr>
<tr>
<td>acuni-82</td>
<td>( f(x, z, y) = f(u, v, w, t) )</td>
<td>13703</td>
<td>8.067</td>
<td>-</td>
</tr>
<tr>
<td>acuni-83</td>
<td>( f(x, z, y) = f(u, u, c, d) )</td>
<td>2</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-84</td>
<td>( f(x, z, y) = f(u, u, c, c) )</td>
<td>4</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-85</td>
<td>( f(x, z, y) = f(u, u, v, c) )</td>
<td>18</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-86</td>
<td>( f(x, z, y) = f(u, u, v, w) )</td>
<td>69</td>
<td>0.033</td>
<td>-</td>
</tr>
<tr>
<td>acuni-87</td>
<td>( f(x, z, y) = f(u, u, u, c) )</td>
<td>7</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-88</td>
<td>( f(x, z, y) = f(u, u, u, v) )</td>
<td>12</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-89</td>
<td>( f(x, z, y) = f(u, u, u, v) )</td>
<td>47</td>
<td>0.017</td>
<td>-</td>
</tr>
</tbody>
</table>
Some two equations systems: notice the size differences of the complete sets of solutions generated.

<table>
<thead>
<tr>
<th>Example</th>
<th>Problem</th>
<th>#</th>
<th>CK</th>
<th>#</th>
<th>MS</th>
<th>CL</th>
</tr>
</thead>
<tbody>
<tr>
<td>acuni-90</td>
<td>(f(x, z, y) = f(u, u, u, u))</td>
<td>5</td>
<td>0.001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-91</td>
<td>(f(x, z, z) = f(u, c, d, e))</td>
<td>2</td>
<td>0.017</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-92</td>
<td>(f(x, z, z) = f(u, c, c, d))</td>
<td>2</td>
<td>0.001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-93</td>
<td>(f(x, z, z) = f(u, c, c, c))</td>
<td>1</td>
<td>0.001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-94</td>
<td>(f(x, z, z) = f(u, v, c, d))</td>
<td>140</td>
<td>0.217</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-95</td>
<td>(f(x, z, z) = f(u, v, v, c))</td>
<td>28</td>
<td>0.033</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-96</td>
<td>(f(x, z, z) = f(u, v, w, c))</td>
<td>6006</td>
<td>80.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-97</td>
<td>(f(x, z, z) = f(u, v, w, t))</td>
<td>1044569</td>
<td>424</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-98</td>
<td>(f(x, z, z) = f(u, u, v, c))</td>
<td>2</td>
<td>0.001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-99</td>
<td>(f(x, z, z) = f(u, u, c, c))</td>
<td>2</td>
<td>0.001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-100</td>
<td>(f(x, z, z) = f(u, v, c, c))</td>
<td>12</td>
<td>0.017</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-101</td>
<td>(f(x, z, z) = f(u, u, v, w))</td>
<td>101</td>
<td>0.033</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-102</td>
<td>(f(x, z, z) = f(u, u, v, v))</td>
<td>13</td>
<td>0.001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-103</td>
<td>(f(x, z, z) = f(u, u, u, c))</td>
<td>0</td>
<td>0.001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-104</td>
<td>(f(x, z, z) = f(u, u, u, v))</td>
<td>1</td>
<td>0.017</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acuni-105</td>
<td>(f(x, z, z) = f(u, u, u, u))</td>
<td>1</td>
<td>0.001</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Some big systems:

<table>
<thead>
<tr>
<th>Example</th>
<th>Problem</th>
<th>#</th>
<th>CK</th>
<th>#</th>
<th>MS</th>
<th>CL</th>
</tr>
</thead>
<tbody>
<tr>
<td>acuni-S1</td>
<td>(f(x, x, a) = f(z, t, u, b))</td>
<td>0</td>
<td>0.017</td>
<td>0</td>
<td>6.850</td>
<td>-</td>
</tr>
<tr>
<td>acuni-S2</td>
<td>(f(x, x, x) = f(z, t, u, v))</td>
<td>0</td>
<td>0.017</td>
<td>#DS: 20</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td>acuni-S3</td>
<td>(f(x, x, x, a) = f(u, v, w, t))</td>
<td>24</td>
<td>0.050</td>
<td>#DS: 24</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-S4</td>
<td>(f(x, y, y, y, y, y) = f(z, z, z, z, z, t, t))</td>
<td>1</td>
<td>0.117</td>
<td>1023</td>
<td>11.367</td>
<td>-</td>
</tr>
<tr>
<td>acuni-S5</td>
<td>(f(x, x, x, x) = f(u, v, w, t))</td>
<td>0</td>
<td>0.017</td>
<td>#DS: 20</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-S6</td>
<td>(f(x, x, x, x) = f(u, v, w, t))</td>
<td>1</td>
<td>0.001</td>
<td>#DS: 35</td>
<td>0.017</td>
<td>-</td>
</tr>
<tr>
<td>acuni-S7</td>
<td>(f(x, y, y, y, y) = f(z, p, r, t))</td>
<td>1</td>
<td>0.033</td>
<td>#DS: 35</td>
<td>0.001</td>
<td>-</td>
</tr>
</tbody>
</table>
References


Domenjoud, E. (1989). Number of minimal unifiers of the equation \( ax_1 + \cdots + ax_p \equiv_{AC} \beta y_1 + \cdots + \beta y_q \). Research Report 89-R-2, Centre de Recherche en Informatique de Nancy, Nancy (France). To appear in Journal of Automated Reasoning.


