J. Symbolic Computation (1994) **18**, 297–318

Tree Automata Help One To Solve Equational Formulae In AC-Theories

D.LUGIEZ and J.L. MOYSSET

CRIN-INRIA, BP239 54506 Vandoeuvre-lès-Nancy, FRANCE

(Received 16th March 1993)

In this paper we consider particular equational formulae where equality $=_{AC}$ is the congruence induced by a set of associative-commutative axioms. The formulae we are interested in have the form $t \neq_{AC} t_1 \wedge \ldots t \neq_{AC} t_n$ and are usually known as complement problems. To solve a complement problem is to find an instance of t which is not an instance of any t_i modulo associativity-commutativity. We give a decision procedure based on tree automata which solves these formulae when all the t_i 's are linear. We show that this solution also gives a decision of inductive reducibility modulo associativity and commutativity in the linear case and we give several extensions of this approach. Then, we define a new class of tree automata, called conditional tree automata which recognize sets of generalized terms, i.e terms written as multisets, and where the application of a rule depends on the satisfiability of a formula of Presburger's arithmetic. This class of tree automata allows one to solve non-linear complement problems when all occurrences of a non-linear variable occur under the same node (in flattened terms). This solution also provides a procedure to decide inductive reducibility modulo associativity-commutativity in the same case.

Introduction

Many problems arising in Computer Science involve formulae built on the equality predicate and syntactic objects, usually terms. But terms are merely denotations for complex entities and a purely syntactical approach lacks any insight into the semantics of these entities. To overcome this limitation, a classical solution is to add axioms which model the semantic behaviour of the real objects. Amongst the most useful axioms one finds the *commutativity* axiom (C) $f(x, y) = f(y, x)$, the *associativity* axiom (A) $f(x, f(y, z)) = f(f(x, y), z)$, the *idempotency* axiom (I) $f(x, x) = x$ and the *unit element* axiom (1) $f(x, e) = x$. Many operators studied in Computer Science satisfy one or more of these axioms, for example the parallel operator in parallelism is associative-commutative (AC for short), the boolean connectives are associative-commutative and idempotent (ACI for short), and many algebraic functions are associative-commutative and have a unit element (AC1 for short). Therefore we are led to tackle the problem of dealing with formulae on equality modulo a set of axioms. Decision procedures exist when no axiom is included (Malc'ev (1971), Colmerauer (1984), Lassez and Marriot (1986), Maher (1988), Comon and Lescanne (1989)) but a decision procedure for these formulae usually does not exist in the presence of axioms (indeed, the theory of Σ_3 formulae modulo AC is undecidable (Treinen (1992)). Actually, we are asking for too much: in practice, the formulae that we must solve belong to very restricted classes. After the well-known *unification problem*, the most interesting one is the *complement problem*, i.e to solve $t \neq_E t_1 \wedge \ldots \wedge t \neq_E t_n$ where E is the theory we are interested in. These formulae occur in algebraic specifications and functional programming when dealing with sufficient completeness issues, in logic programming and constraint logic programming, especially concerning constructive negation and in the paradigm of learning by examples and counter-examples in the field of machine learning (see Lassez, Maher and Marriott (1991) for example).

This paper describes a new approach to the AC-complement problem which relies on tree automata, as proposed in Lugiez and Moysset (1993) . First, we show how tree automata solve linear AC-complement problems and we give some extensions of this result. Then we describe a new class of tree automata, called conditional tree automata which deal with (some kind of) flattened trees and involve conditions which are formulas of Presburger's arithmetic. Using this new class, we can solve a non-linear case of the AC-complement problem, more precisely when all the occurrences of a non-linear variable are under the same AC-symbol. Moreover our approach can be extended to the decision of the inductive reducibility property modulo AC which is a key notion in inductionless induction. This property is undecidable (Kapur, Narendran, Rosenkrantz and Zhang *(1991))* , *but we are able to decide it in the aforementioned cases.*

The paper is organized as follows: section 1 gives the preliminary notions required in the following. Section 2 gives our first results for linear problems and these results are extended in section 3. We introduce generalized and normalized terms in section 4 and conditional tree automata are discussed in section 5. Finally, we give our most general results on non-linear problems in section 6.

1. Definitions and notations

We need some definitions and notations, let us start with terms. Missing definitions can be found in Dershowitz and Jouannaud (1988).

1.1. Terms

Terms denoted by $s, t, \ldots$ are constructed from a finite set of function symbols Σ and a denumerable set of variables X denoted by $x, y, z \ldots$ Each function symbol has a fixed arity and constants are function symbols of arity 0. From now on we suppose that Σ contains at least one constant such that T_Σ, the set of *ground* terms i.e. without variables is not empty. The set of terms is denoted by $T_\Sigma(X)$. A variable of t is *linear* if it occurs at most once in t otherwise it is non-linear, and a term is *linear* if it contains only linear variables. A *position* is a sequence of integers, the empty sequence is denoted by ϵ, and a non-empty sequence is written as $p.i$ with i an integer and p a sequence. The subterm of a term s at position p, denoted by $s_{|p}$, is defined by $s_{|\epsilon} = s$ and $s_{|p.i} = s_i$ if $s_{|p} = f(s_1, \ldots, s_n)$ and $1 \leq i \leq n$.

An equation (i.e. an axiom) is a pair of terms written $s = t$. A finite set of equations E defines an equational theory and induces a congruence relation on terms denoted by $=_E$. The class of a term t is the set of terms equal to t modulo E. We are mainly interested in the following equations: commutativity (C) $f(x,y) = f(y,x)$, associativity (A) $f(x, f(y,z)) = f(f(x,y),z)$, idempotency (I) $f(x,x) = x$ and unit element (1) $f(x,e) = x$. More precisely, we are interested in the $=_{AC}$ congruence (resp. $=_{ACI}, \ldots$) obtained when $\Sigma = F \cup G$ where the functions of G are free and the functions of F are commutative and associative (resp. commutative and associative and idempotent...).

Substitutions, denoted by $\sigma, \theta, \rho \ldots$, are the morphisms on terms and the application of σ to t is denoted by $t\sigma$. A term s is an *E-instance* of a term t iff $s =_E t\sigma$ for some substitution σ. It is a *ground E-instance* if s is ground. When E is the empty theory, we say *instances* and *ground instances*, dropping the E prefix.

A term rewrite system R is a finite set of rewrite rules $l \to r$ which defines a rewrite relation $\to_R$ on terms. The system is left-linear if for each rule $l \to r$ of R, the left-hand side l is linear. A term t is *reducible* by R modulo E iff there are some term t', some position p in t', some substitution θ and some rule $l \to r \in R$ s.t. $t' =_E t$ and $t'_{|p} = l\theta$.

To solve the **E-complement problem** $t \neq_E t_1 \wedge \ldots \wedge t \neq_E t_n$ with $\vec{x} \cap \vec{y_i} = \emptyset$ and $\vec{y_i} \cap \vec{y_j} = \emptyset$ if $\vec{y_i} = Var(t_i)$ and $\vec{x} = Var(t)$, is to find a ground instance of t which is not an E-instance of any t_i. In other words, it amounts to deciding the validity of the formula $\exists \vec{x} \forall \vec{y_1} \ldots \vec{y_n} : t \neq_E t_1 \wedge \ldots \wedge t \neq_E t_n$ in the algebra of ground terms. In this paper, we are mainly concerned with the AC-complement problem and related issues.

The cardinality of a set S is denoted by $|S|$ and $S_1 \uplus S_2$ denotes the disjoint union of S_1 and S_2.

1.2. Regular languages and tree automata

Regular sets of trees and tree automata are similar to regular sets of words and finite automata. We recall some definitions and results, and we refer the reader to Gécseg and Steinby (1984) for details.

Definition 1. *Given a signature Σ, a bottom-up tree automaton $\mathcal{A}$ is a triple (Q, Q_{Final}, R) where Q is a finite set of states, $Q_{Final} \subseteq Q$ is a set of final states, and R is a set of transition rules of the form $f(q_1, \ldots q_n) \to q_{n+1}$ with $arity(f) = n$ and $q_i \in Q$.*
The transition relation $\to$ is the smallest relation on $T_\Sigma(X) \times Q$ s.t.:

$$\frac{t_1 \to q_1 \quad \ldots \quad t_n \to q_n \quad f(q_1, \ldots, q_n) \to q_{n+1} \in R}{f(t_1, \ldots, t_n) \to q_{n+1}}$$

The language accepted by $\mathcal{A}$ is the set of ground terms t such that $t \to q$ for some $q \in Q_{Final}$.

A set of ground terms is a *regular tree language* iff it is accepted by a tree automaton. An automaton is *completely specified* if for each ground term t, there is some q s.t. $t \to q$. The state q is said to be *accessible*. The automaton is *deterministic* if for each t, the state q is unique. Regular languages are closed under the boolean operations and it is decidable if the language accepted by a tree automaton is empty or not. Regular languages can also be characterized as least fixed-point solution of systems of equations:

$L_1 = \sum_{i \in I_1} h_i^1(L_{i_1}^1, \ldots, L_{i_p}^1)$

...

$L_n = \sum_{i \in I_n} h_i^n(L_{i_1}^n, \ldots, L_{i_p}^n)$

where the h_i^j are symbols of Σ and the sum means set union.

1.3. Flattened terms

From now we suppose that $\Sigma = F \uplus G$. The elements f of F are binary AC-function symbols and the elements g of G are free function symbols. A term $f(t_1, t_2)$ has sort f and a term x or $g(t_1, \dots, t_n)$ have sort *nac*. To make easier the reasoning involving AC-axioms, one usually introduces the notion of *flattened* terms:

Definition 2. *A* flattened *term is either a term $f(t_1, f(t_2, \dots, f(t_{m-1}, t_m)))$ where the t_i's are flattened terms not of sort f, or $g(s_1, \dots, s_n)$ where the s_i's are flattened terms, or x.*

The notation $f(|t_1, \dots, t_m|)$ denotes the flattened term $f(t_1, f(t_2, \dots, t_m))$ and $f(|t|)$ denotes t if t is a flattened term not of sort f. Given some f and a flattened term t, there exists a unique representation $t = f(|t_1, \dots, t_n|)$. The operation $\sqcup_f$ is defined on flattened terms by: $f(|s_1, \dots, s_m|) \sqcup_f f(|t_1, \dots, t_n|) = f(|s_1, \dots, s_m, t_1, \dots, t_n|)$. By definition $\sqcup_f$ is associative and returns a flattened term.

Example Let $F = \{f\}$ and $G = \{0, g\}$, then $g(0) \sqcup_f f(0, g(0)) = f(g(0), f(0, g(0)))$

Terms and flattened terms are strongly related:

Proposition 1. *For each term s there exists a flattened term t s.t. $s =_{AC} t$*

Proof Let R be the rewrite system defined by $f(f(x, y), z) \to f(x, f(y, z))$ for all $f \in F$. It is terminating and confluent. Let $Flat(s)$ be the unique result of rewriting s with R, then $Flat(s)$ is a flattened term. $\square$

The permutative congruence $=_P$ is the smallest congruence on terms s.t. $f(|\dots, t_i, \dots, t_j, \dots|) = f(|\dots, t_j, \dots, t_i, \dots|)$. It is related to $=_{AC}$ congruence in the following way:

Proposition 2. *Let s, t be terms, then $s =_{AC} t$ iff $Flat(s) =_P Flat(t)$.*

Proof By structural induction on s and t. $\square$

In the following, we write $[t]_{AC} = \{s \mid s =_{AC} t\}$ and $[t]_P = \{s \mid s =_P t\}$. Given a set of terms L, we introduce the sets:

$Gr(L) = \{s \mid s = t\theta$ for some $t \in L$ and some ground substitution $\theta\}$

$AC(L) = \{s \mid s =_{AC} t$ for some $t \in L\}$

Proposition 3. *Let s, t be two terms s.t. $s =_{AC} t$ then $AC(Gr(s)) = AC(Gr(t))$.*

Proof By structural induction on s and t. $\square$

Proposition 4. *Let t be a linear term s.t. $Flat(t) =_P f(|t_1, \dots, t_n, x_1, \dots, x_m|)$ with $t_i \notin X$, then $s \in AC(Gr(t))$ iff $s = f(s_1, s_2)$ with*

- $s_1 \in AC(Gr(f(|t_{i_1}, \dots, t_{i_k}, x_{j_1}, \dots, x_{j_l}|)))$

- $s_2 \in AC(Gr(f(\lvert t_{i'_1}, \ldots, t_{i'_{k'}}, x_{j'_1}, \ldots, x_{j'_{l'}} \rvert)))$
  where $\{t_1, \ldots, t_n\} = \{t_{i_1}, \ldots, t_{i_k}\} \uplus \{t_{i'_1}, \ldots, t_{i'_{k'}}\}$ *and* $\{x_1, \ldots, x_m\} = \{x_{j_1}, \ldots, x_{j_l}\} \cup \{x_{j'_1}, \ldots, x_{j'_{l'}}\}$

Proof According to the previous propositions,

$$\begin{array}{lll}
s \in AC(Gr(t)) \text{ iff} & & \\
s =_{AC} f(\lvert t_1, \ldots, t_n, x_1, \ldots, x_m \rvert)\theta \text{ iff} & & \\
s =_{AC} Flat(t_1\theta) \sqcup_f \ldots \sqcup_f Flat(t_n\theta) \sqcup_f Flat(x_1\theta) \ldots \sqcup_f Flat(x_m\theta) \text{ iff} & & \\
Flat(s) = r_1 \sqcup_f \ldots \sqcup_f r_p \text{ where} & Flat(x_i\theta) = u_i^1 \sqcup_f \ldots \sqcup_f u_i^{p_i} & \text{iff} \\
 & \{r_1, \ldots, r_p\} = \{t_1\theta, \ldots, t_n\theta, u_1^1, \ldots, u_m^{p_m}\} & \\
s = f(s_1, s_2) \text{ with } Flat(s_1) = r_1 \sqcup_f \ldots \sqcup_f r_k \text{ and } Flat(s_2) = r_{k+1} \sqcup_f \ldots \sqcup_f r_p \text{ iff} & & \\
s = f(s_1, s_2) \text{ with} & s_1 \in AC(Gr(f(\lvert t_{i_1}, \ldots, t_{i_k}, x_{j_1}, \ldots, x_{j_l} \rvert)) & \\
 & s_2 \in AC(Gr(f(\lvert t_{i'_1}, \ldots, t_{i'_{k'}}, x_{j'_1}, \ldots, x_{j'_{l'}} \rvert))) & \\
 & \{t_1, \ldots, t_n\} = \{t_{i_1}, \ldots, t_{i_k}\} \uplus \{t_{i'_1}, \ldots, t_{i'_{k'}}\} & \\
 & \{x_1, \ldots, x_m\} = \{x_{j_1}, \ldots, x_{j_l}\} \cup \{x_{j'_1}, \ldots, x_{j'_{l'}}\} &
\end{array}$$

since t is linear (hence a variable x_i does not occur in any t'_js). □

2. A tree automata solution for linear complement problems

In this section, we relate regular languages and the set of ground AC-instances of a linear term.

Example Let $t = f(0, x)$ where $F = \{f\}$ and $G = \{0, g\}$. Let $L_{\{0,x\}} = AC(Gr(t))$, let $L_{\{0\}} = AC(Gr(0))$, let $L_{\{x\}} = AC(Gr(x))$. These languages are the least fixed-point solutions of the system of equations:

$$\begin{array}{l}
L_{\{0\}} = \{0\} \\
L_{\{x\}} = L_{\{0\}} + g(L_{\{x\}}) + f(L_{\{x\}}, L_{\{x\}}) \\
L_{\{0,x\}} = f(L_{\{0\}}, L_{\{x\}}) + f(L_{\{x\}}, L_{\{0\}}) + f(L_{\{0,x\}}, L_{\{x\}}) + f(L_{\{x\}}, L_{\{0,x\}})
\end{array}$$

therefore $L_{\{0\}}, L_{\{x\}}$ and $L_{\{0,x\}}$ are regular tree languages, and an automaton recognizing $L_{\{0,x\}}$ is $\mathcal{A} = (\{q_x, q_0, q_L\}, \{q_L\}, R)$ with R consisting of: $0 \to q_0$, (the rule of an automaton recognizing $L_{\{0\}}$), $0 \to q_x$, $g(q_x) \to q_x$, $f(q_x, q_x) \to q_x$ (the rules of an automaton recognizing the instances of x) and $f(q_0, q_x) \to q_L, f(q_x, q_0) \to q_L, f(q_x, q_L) \to q_L, f(q_L, q_x) \to q_L$.

An algebraic solution of the next proposition exists but we give a less elegant combinatorial proof since we are interested in the constructive aspects of the problem.

PROPOSITION 5. *For each linear term t, $AC(Gr(t))$ is regular.*

Proof We show that the set of ground AC-instances of t satisfies a least fixed-point equation defining regular languages. At the same time, we sketch the construction of an automaton accepting this language. The proof is by structural induction on t.

- $t \in X$: the proof is obvious.
- $t = g(t_1, \ldots, t_n)$: then

$$AC(Gr(t)) = g(AC(Gr(t_1)), \ldots, AC(Gr(t_n)))$$

where the $AC(Gr(t_i))$ are regular by induction hypothesis, therefore $AC(Gr(t))$ is regular. An automaton accepting L is $\mathcal{A} = (F, Q, Q_F, R)$ where $Q = \cup_i Q_i \cup \{q_F\}$, $Q_F = \{q_F\}$, $R = \cup_i R_i \cup \{g(q_1, \ldots, q_n) \rightarrow q_F \textit{ where } q_i \in Q_F^i\}$.

- $t = f(\ldots)$ and $Flat(t) = f(|t_1, \ldots, t_n|)$. For simplicity, we write $L_{\{i_1,\ldots,i_k\}} = AC(Gr(f(|t_{i_1}, \ldots, t_{i_k}|)))$. According to proposition 3, one has :

$$L_{\{1,\ldots,n\}} = \sum_{\substack{I \cup J = \{1 \ldots n\} \\ I, J \neq \emptyset \\ i \in I \cap J \Rightarrow t_i \in X}} f(L_I, L_J)$$

  where each L_I for $I \neq \{1, \ldots, n\}$ is regular by induction hypothesis. Therefore $AC(Gr(t)) = L_{\{1,\ldots,n\}}$ is regular.
  Moreover, if $\mathcal{A}_I = (F, Q^I, Q_F^I, R_I)$ is an automaton recognizing L_I for $I \neq \{1, \ldots, n\}$, an automaton accepting L is $\mathcal{A} = (F, Q, Q_F, R)$ where:
  - $Q = \cup_I Q_I \cup \{q_F\}$
  - $Q_F = \{q_F\}$
  - $R = \cup_I R_I \cup \{f(q_i, q_j) \rightarrow q_F \textit{ for all } q_i \in Q_F^I, q_j \in Q_F^J s.t.\ I \cup J = \{1 \ldots n\}, i \in I \cap J \Rightarrow t_i \in X\}$

In each case, the automaton accepting $AC(Gr(t))$ satisfies $t \rightarrow q$. Moreover $s =_{AC} t$ implies $s \rightarrow q$. □

THEOREM 1. *The linear complement problem $t \neq_{AC} t_1 \wedge \ldots \wedge t \neq_{AC} t_n$ is decidable.*

Proof Let $t \neq_{AC} t_1 \wedge \ldots \wedge t \neq_{AC} t_n$ be a complement problem s.t the t_i's are linear (but t may be non-linear). Let L be the union of the ground AC-instances of the t_i's, then L is a regular language since the t_i's are linear. Let L_C be the complement of L and $\mathcal{A}_C$ be an automaton recognizing L_C. If there is a ground instance $t\theta$ of t which is a solution of the complement problem, then this instance is accepted by $\mathcal{A}_C$. For each variable x_i of t, $x_i\theta$ is some ground term and $x_i\theta \rightarrow q_i$ for some accessible state q_i of $\mathcal{A}_C$. Therefore there is a solution of the complement problem iff there is some assignment of accessible states to the variables of t s.t. $t\{x_1 \leftarrow q_1, \ldots, x_n \leftarrow q_n\} \rightarrow q_f$ where q_f is a final state of $\mathcal{A}_C$ and where $t\{x_1 \leftarrow q_1, \ldots, x_n \leftarrow q_n\}$ denotes t where each occurrence of x_i is replaced by q_i. If there is no such assignment, then there is no solution to the complement problem. Since there is a finite number of possible assignments, the complement problem modulo AC is decidable. □

3. Extensions

3.1. To inductive reducibility modulo AC

A key property in inductionless induction is that of *inductive reducibility*: given a set E of equations and a set R of rewrite rules, a term t is inductively reducible modulo E iff each ground instance s of t is reducible by R modulo E. In the AC case, this property is undecidable but our proof of complement problems can be easily extended to decide inductive reducibility modulo AC for left-linear rewrite systems [†]. We prove that *Red*(l) the set of ground terms reducible modulo AC by $l \to r$ is regular when l is linear.

PROPOSITION 6. *Let l be a linear term, then Red(l) is a regular tree language.*

Proof The proof is similar to the proof of proposition 5. □

From this result, we get the decision of the inductive-reducibility modulo AC.

THEOREM 2. *Inductive reducibility modulo AC of a term t for a left-linear system R is decidable.*

Proof Let $l_1, \ldots, l_n$ be the set of left-hand sides of R, and let $\mathcal{A}$ be a deterministic automaton recognizing the union of the *Red*(l_i)'s. The term t is inductively reducible iff for each ground θ, $t\theta$ belongs to this set. We proceed as in the proof of the complement modulo AC: to each variable x_i of t assign some accessible state q_i, and compute the state q such that $t\{x_1 \leftarrow q_1, \ldots, x_n \leftarrow q_n\} \to q$. If there is some assignment such that the state q is not a final state of $\mathcal{A}$, then t is not inductively reducible, otherwise t is inductively reducible. Since there is only a finite number of possible assignments, the inductive reducibility property is decidable. □

3.2. To other theories

The previous results can be easily extended to other theories. The first one is the AC1 theory, i.e AC with unity: for each AC-symbol f, there exists a constant e such that $f(x, e) = x$. To handle this identity element, we add the terms $f(L_e, L)$ and $f(L, L_e)$ to the equations defining the ground AC-instances of a term $t = f(\ldots)$ where $L_e = \{e\} + f(L_e, L_e)$. From the automaton viewpoint, this amounts to adding rules $f(q, q_e) \to q$, $f(q_e, q) \to q$ where q is a final state, and $e \to q_e$, $f(q_e, q_e) \to q_e$.

The second one is the associativity theory. In this case, the flattened version of a term is unique since the commutativity axiom does not hold. The proof that the A-instances of a linear term is a regular language works as in the AC case for the first two cases, and in the proof of the third case, sets are replaced by lists: $\{1 \ldots n\}$ becomes $[1 \ldots n]$, subsets I and J become sublists and union is replaced by concatenation.

THEOREM 3. *The complement problem $t \neq_{AC1} t_1 \wedge \ldots \wedge t \neq_{AC1} t_n$ (resp. $\neq_A$) where the t_i 's are linear terms is decidable. The inductive reducibility modulo AC1 (resp. modulo A) of a term t for a left-linear term rewrite system is decidable.*

3.3. To some non-linear cases

Tree automata with syntactical equality tests between brothers are introduced in Bogaert and Tison (1992) where it is shown that this class is closed under boolean operations and that the emptiness of the accepted language is decidable. We extend this class by allowing equality tests modulo AC in order to get the decidability of the AC-complement problem for *strictly restricted* non-linear terms, as defined by:

[†] the decidability of inductive reducibility modulo AC for left-linear rewrite systems has been stated first in Jouannaud and Kounalis (1989)

DEFINITION 3. *A term is strictly restricted iff for each non-linear variable x, there exists a position p such that all the occurrences of x occur at positions $p.i$ with i an integer, and the symbol of t at position p is not AC.*

For example, if $F = \{f\}$ and $G = \{0, g\}$ then $f(g(x,x),0)$ is strictly restricted but $f(x,x)$ and $f(x,g(0,x))$ are not. The approach of Bogaert and Tison generalizes smoothly to the AC case, except that the decision of emptiness requires that equivalent terms reach the same states, which is true in our applications. The reader is referred to the original work of Bogaert and Tison (1992) for details since the algorithms are identical except that $=$ is replaced by $=_{AC}$. Firstly, we define our new class of tree automata.

DEFINITION 4. *Given a signature Σ, an automaton with equality tests between brothers $\mathcal{A}$ is a triple (Q, Q_F, R) where Q is a finite set of states, $Q_F \subseteq Q$ is a set of final states and R is a set of rules $\varphi : h(q_1, \ldots, q_n) \to q_{n+1}$ with $h \in \Sigma$, $\varphi \in Form_n$ where $Form_n$ is inductively defined by:*

- *$\#i =_{AC} \#j \in Form_n$ (which means that the i^{th} son is equal to the j^{th} son modulo AC), $\top \in Form_n$ (meaning that no condition is required)*
- *if $\varphi \in Form_n$ and $\psi \in Form_n$ then $\neg\varphi \in Form_n, \psi \vee \varphi \in Form_n, \psi \wedge \varphi \in Form_n$.*

***Moreover if** $h \in F$ **and** $\varphi : h(q_1, q_2) \to q_3 \in R$ **we demand that** φ **is** $\top$ (i.e there is no condition for rules with AC symbols). The transition relation $\to$ is the smallest relation on $T_\Sigma(X) \times Q$ s.t.:*

$$\frac{t_1 \to q_1 \quad \ldots \quad t_n \to q_n \quad \varphi : h(q_1, \ldots, q_n) \to q_{n+1} \in R \quad h(t_1, \ldots, t_n) \models \varphi}{h(t_1, \ldots, t_n) \to q_{n+1}}$$

where $h \in \Sigma$ and $h(t_1, \ldots, t_n) \models \varphi$ means that $h(t_1, \ldots, t_n)$ satisfies φ. The language accepted by $\mathcal{A}$ is the set of ground terms t such that $t \to q$ for some $q \in Q_{Final}$.

One should notice that equality tests are allowed under non-AC symbols only. The notions of deterministic and completely specified automata are the same as for usual tree automata. An incompletely specified tree automaton with equality tests can be easily extended into a completely specified one. Now we sketch the determinization process which is similar to the determinization process for tree automata with syntactical equality tests.

1 Separate the conditions such that the conditions becomes mutually exclusive (for example use the subset $\bigwedge_{i,j \in I} \#i =_{AC} \#j \bigwedge_{i,j \in \{1,\ldots,n\}-I} \#i \neq_{AC} \#j$ of $Form_n$)

2 Use a classical determinization algorithm to compute the equivalent deterministic automaton (a state of the deterministic automaton is a set of states of the non-deterministic one)

Remark. : let t be a term and let $q_1, \ldots, q_n$ be the states such that $t \to q_i$ in the non-deterministic automaton then $t \to \{q_1, \ldots, q_n\}$ in the deterministic one.

A direct consequence of the determinization process is that the class of accepted languages is closed under complement (exchange final and non-final states) and union (straightforward) hence under intersection. What remains for us to give is a way to decide the emptiness of the language accepted by an automaton with equality tests. This works as in Bogaert and Tison (1992) , *provided that equivalent terms reach the same states, i.e $t =_{AC} t'$ and $t \to q$ implies that $t' \to q$.* Fortunately, this property is preserved under determinization (see previous remark) union, intersection and complement. From now we suppose that this property is satisfied by the automata that we consider. The following algorithm is used for deciding emptiness ($N_{AC}(L)$ denotes the number of distinct equivalence classes of a finite set of ground terms L and max-arity is the maximal arity of the symbols of Σ):

```
For each state $q$ do set $\mathcal{L}_q^0 = \emptyset$.
i=1.
Repeat
For each state $q$ do    $\mathcal{L}_q^i \leftarrow \mathcal{L}_q^{i-1}$
                         For each rule $r$ of the form $\varphi : h(q_1, \dots, q_n) \to q$ do
                         if $N_{AC}(\mathcal{L}_q^{i-1}) <$ max-arity
                         then $\mathcal{L}_q^i = \mathcal{L}_q^{i-1} \cup \{h(t_1, \dots, t_n)$ satisfying $\varphi$ where $t_j \in \mathcal{L}_{q_j}^{i-1}\}$
i=i+1
until $\exists q \in Q_F$ s.t. $\mathcal{L}_q^i \neq \emptyset$ or $\forall q, \mathcal{L}_q^i = \mathcal{L}_q^{i-1}$
if $\exists q \in Q_F$ s.t. $\mathcal{L}_q^i \neq \emptyset$ then return not empty
else return empty
```

PROPOSITION 7. *The emptiness decision algorithm terminates and is correct.*

Proof The proof is as in Bogaert and Tison (1992) and similar to the proof given for more complicated automata described in section 5.1. □

Now we prove that the ground AC-instances of a strictly restricted term are recognized by a tree automaton with equality tests:

PROPOSITION 8. *Let t be a strictly restricted term, then $AC(Gr(t))$ is accepted by a tree automaton with equality tests s.t. if $t \to q$ and $t' =_{AC} t$ then $t' \to q$.*

Proof The proof is as in the linear case with one difference: if $t = g(\dots)$ with $g \in G$, for each non-linear variable x occurring as the $i_1^{th}, i_2^{th}, \dots, i_n^{th}$ sons of t, add the condition $\#i_1 =_{AC} \#i_2 =_{AC} \dots =_{AC} \#i_n$ to the related rule of the automaton. □

The theorem on the decidability of the complement problem and inductive reducibility is an immediate consequence of the previous results.

THEOREM 4. *The complement problem $t \neq_{AC} t_1 \wedge \dots \wedge t \neq_{AC} t_n$ (resp. $\neq_A$) where the t_i's are strictly restricted, is decidable. Inductive reducibility modulo AC (resp. modulo A) of a term t for a term rewrite system with strictly restricted left-hand sides is decidable.*

Proof The proof is the same as for the linear case. □

The next step is to allow occurrences of non-linear variables under a node labeled by some AC-symbol. First, we introduce generalized terms which are needed to consider terms with AC-symbols as multisets. Then we introduce a new class of tree automata acceptings sets of generalized terms instead of sets of terms, *conditional tree automata*, and we study its properties. Finally, we show how this class can be used to solve AC-complement problems and to decide the inductive reducibility property in this restricted non-linear case.

4. Generalized and normalized terms

Generalized terms are introduced to make explicit the multiset structure induced by AC functions.

DEFINITION 5. *Generalized terms over $\Sigma \cup X$ are defined by the following grammar:*
$n, m ::= \ 1 \mid 2 \mid 3 \ldots$
$S, T ::= x \mid g(T_1, \ldots, T_n) \mid f[n_1.T_1, \ldots, n_p.T_p]$
where $f \in F$, $g \in G$, p is an integer greater than or equal to 1 and n is the arity of g. The generalized term $f[n_1.T_1, \ldots, n_p.T_p]$ has sort f and the other have sort nac (meaning not AC*). A generalized term is ground if it contains no variable.*

A generalized term can be seen as a flattened term using the mapping I defined by:
$I(f[1.T]) = I(T)$
$I(f[n.T]) = f(I(T), I(f[(n-1).T]))$ if $n \geq 2$
$I(f[n_1.T_1, \ldots, n_p.T_p]) = I(f[n_1.T_1]) \sqcup_f I(f[n_2.T_2, \ldots, n_p.T_p])$
$I(g(T_1, \ldots, T_n)) = g(I(T_1), \ldots, I(T_n))$

We note that the notion of sorts on generalized terms and the notion of sorts on terms do not coincide, and that the mapping I is onto but not one-to-one.

PROPOSITION 9. *For each flattened term t and each $f \in F$ there exists a generalized term T of sort f s.t. $I(T) = t$.*

Proof By structural induction on t. □

The $\sqcup_f$ operation is defined on generalized terms of sort f by:
$f[n_1.T_1, \ldots, n_p.T_p] \sqcup_f f[m_1.S_1, \ldots, m_q.S_q] = f[n_1.T_1, \ldots, n_p.T_p, m_1.S_1, \ldots, m_q.S_q]$
and the $=_P$ congruence is the smallest congruence s.t. $f[\ldots, n_i.T_i, \ldots, n_j.T_j, \ldots] =_P f[\ldots, n_j.T_j, \ldots, n_i.T_i, \ldots]$. The congruence class of T for $=_P$ is denoted by $[T]_P$. To deal with non-linearity in the AC-complement problem, we must concentrate on some particular generalized terms where equal parts are grouped together, i.e. normalized terms.

DEFINITION 6. *Normalized terms are defined by:*

- *x is normalized,*
- *$g(T_1, \ldots, T_n)$ is normalized iff $T_1, \ldots, T_n$ are normalized,*

- *$f[n_1.T_1, \ldots, n_p.T_p]$ is normalized iff $n_1 + \ldots + n_p \geq 2$, for $i = 1, \ldots, p$ the sort of T_i is not f and T_i is normalized, $T_i \neq_P T_j$ for $i \neq j$*

Example The generalized term $f[2.0, g(f[2.0])]$ is normalized, but $f[1.g(f[2.0])]$ and $f[2.0, g(f[2.0]), 2.0]$ are not normalized.

The next propositions state that congruence classes of normalized terms for $=_P$ and congruence classes of flattened terms for $=_{AC}$ are related.

PROPOSITION 10. *Let S, T be normalized terms, then $S =_P T$ iff $I(S) =_P I(T)$.*

Proof By structural induction on S, T. □

PROPOSITION 11. *Let s, t be two flattened terms s.t. $s =_P t$ and let S, T be two normalized terms s.t. $I(S) = s$ and $I(T) = t$ then $S =_P T$.*

Proof By structural induction on S, T. □

Finally the $\sqcup_f$ operation is compatible with $=_P$.

PROPOSITION 12. *Let $S_1, S2, T_1, T_2$ be normalized terms of sort f, then $S_1 =_P S_2$ and $T_1 =_P T_2$ implies $S_1 \sqcup_f T_1 =_P S_2 \sqcup_f T_2$.*

The set $Norm(AC(Gr(t))) = \{T \mid T \textit{ normalized term s.t. } I(T) =_{AC} t\theta\}$ is called the set of *normalized ground AC-instances* of t.

PROPOSITION 13. *Let S, T be normalized terms s.t. $I(S)$ and $I(T)$ are in $Norm(AC(Gr(t)))$, then $S =_P T$.*

Proof $I(S) =_{AC} I(T)$ then $Flat(I(S)) =_P Flat(I(T))$ then $I(S) =_P I(T)$ since $I(T)$ and $I(S)$ are flattened terms, then $S =_P T$. □

5. Conditional tree automata

To deal with generalized terms, we introduce conditional tree automata.

5.1. DEFINITION

DEFINITION 7. *A conditional tree automaton is a four-tuple $((Q_f)_{f \in F}, Q_{nac}, Q_{Final}, R)$ s.t.*

- *for each f, Q_f is a finite set of states of sort f,*
- *Q_{nac} is a finite set of states of sort nac,*
- *$Q_{Final} \subseteq \cup_{f \in F} Q_f \cup Q_{nac}$ is the set of final states,*
- *R is a finite set of transition rules*

Each transition rule has one of the following forms:

- $\varphi(N) : f[N.q_1] \to q$ *where* q *has sort* f, *and* φ *is a formula of Presburger arithmetic with the unique free variable* N,
- $q_1 \sqcup_f q_2 \to q_3$ *where* q_1, q_2, q_3 *have sort* f,
- $g(q_1, \ldots, q_n) \to q$ *where* q *has sort nac.*

Moreover we require that $\sqcup_f$ *satisfies the following properties:*

- $q_1 \sqcup_f q_2 \to q$ *iff* $q_2 \sqcup_f q_1 \to q$,
- $(q_1 \sqcup_f q_2) \sqcup_f q_3 \to q$ *iff* $q_1 \sqcup_f (q_2 \sqcup_f q_3) \to q$.

Given a conditional tree automaton $\mathcal{A}$, the transition relation on generalized terms related to $\mathcal{A}$ is defined by:

$$\frac{\models \varphi(n) \quad T \to q_1 \quad (\varphi(N) : f[N.q_1] \to q) \in R}{f[n.T] \to q}$$

$$\frac{S \to q_1 \quad T \to q_2 \quad (q_1 \sqcup_f q_2 \to q) \in R}{S \sqcup_f T \to q}$$

$$\frac{T_1 \to q_1 \quad \ldots \quad T_n \to q_n \qquad (g(q_1, \ldots, q_n) \to q) \in R}{g(T_1, \ldots, T_n) \to q}$$

The language $L(\mathcal{A})$ accepted by a conditional tree automaton is the set of ground generalized terms T s.t. $T \to q$ for some $q \in Q_{Final}$. Two automata are *equivalent* iff they accept the same language. From the definition, conditional tree automata are compatible with the $=_P$ congruence:

PROPOSITION 14. *If* $T \to q$ *then* T *and* q *have the same sort. If* $T =_P S$ *then* $S \to q$.

5.2. DETERMINIZATION OF CONDITIONAL TREE AUTOMATA

5.2.1. COMPLETELY SPECIFIED AUTOMATA

A conditional tree automaton is *completely specified* if for each generalized term T there exists some state q s.t. $T \to q$.

PROPOSITION 15. *For each conditional tree automaton, there exists an equivalent completely specified conditional tree automaton.*

Proof If $\mathcal{A}$ is not completely specified, one adds new error states $(q^e_f)_{f \in F}$, q^e_{nac} and new rules: $g(\ldots, q^e_h, \ldots) \to q^e_g$, $\varphi(N) : f[N.q^e_h] \to q^e_f$ for $h \in F \cup G$ where $\varphi(N)$ is a formula equivalent to true, $q^e_f \sqcup_f q \to q^e_f$ and $q \sqcup_f q^e_f \to q^e_f$ for every q of sort f. By construction the automaton is a conditional tree automaton and it is completely specified. $\square$

5.2.2. Separation of conditions

Proposition 16. *Let $\mathcal{A}$ be a completely specified tree automaton, let $\varphi_1(N), \ldots, \varphi_n(N)$ be the conditions occurring in the rules of $\mathcal{A}$, then there exists an equivalent completely specified tree automaton $\mathcal{B}$ s.t. the conditions $\psi_1(N), \ldots, \psi_m(N)$ of the rules of $\mathcal{B}$ satisfy the two requirements:*

- *$\psi_i(N) \wedge \psi_j(N)$ is unsatisfiable if $i \neq j$,*
- *for each $i \in \{1, \ldots, n\}$, there exist $j_1, \ldots, j_i$ s.t. $\varphi_i(N) \Leftrightarrow \psi_{j_1}(N) \vee \ldots \vee \psi_{j_i}(N)$*

Proof The ψ_j are boolean combinations of the φ_i's using $\neg, \wedge, \vee$ and a rule $\varphi_i(N) : f[N.q_1] \to q$ is replaced by the rules $\psi_{j_l}(N) : f[N.q_1] \to q$. □

The automaton $\mathcal{B}$ is said to have separated conditions.

5.2.3. The determinization algorithm

An automaton is *deterministic*, iff for each generalized term T there exists a unique state q s.t. $T \to q$.

Proposition 17. *For each conditional automaton $\mathcal{A}$ there exists a equivalent deterministic conditional automaton $\mathcal{A}_D$.*

Proof From the two previous propositions, we can suppose that $\mathcal{A} = ((q_f)_{f\in F}, Q_{nac}, Q_{Final}, R)$ is completely specified and has separated conditions. The deterministic automaton $\mathcal{A}_D = ((Q_f)_{f\in F}, \mathcal{Q}_{nac}, \mathcal{Q}_{Final}, \mathcal{R}_D)$ is constructed as follows:

- a state Q of $\mathcal{A}_D$ is a set $\{q_1, \ldots, q_m\}$ of states of $\mathcal{A}$ of the same sort,
- a state Q of $\mathcal{A}_D$ is a final state if it contains a final state of $\mathcal{A}$,
- $g(Q_1, \ldots, Q_n) \to Q$ is in $\mathcal{R}_D$ if Q is the set of states q s.t. there exists $q_1 \in Q_1, \ldots, q_n \in Q_n$ and a rule $g(q_1, \ldots, q_n) \to q$ in R,
- $\varphi(N) : f[N.Q_1] \to Q$ is in $\mathcal{R}_D$ if Q is the set of states q s.t. there exists $q_1 \in Q_1$ and a rule $\varphi(N) : f[N.q_1] \to q$ in R,
- $Q_1 \sqcup_f Q_2 \to Q$ is in $\mathcal{R}_D$ if Q is the set of states q s.t. there exists $q_1 \in Q_1, q_2 \in Q_2$ and a rule $q_1 \sqcup_f q_2 \to q$ in R.

By construction $\mathcal{A}_D$ is a conditional tree automaton since $Q_1 \sqcup_f Q_2 \to Q$ iff $Q_2 \sqcup_f Q_1 \to Q$, and $(Q_1 \sqcup_f Q_2) \sqcup_f Q_3) \to Q$ iff $Q_1 \sqcup_f (Q_2 \sqcup_f Q_3) \to Q$. Moreover $\mathcal{A}_D$ is deterministic since $\mathcal{A}$ has separated conditions. By structural induction, one proves that if T is a generalized term and if $q_1, \ldots, q_n$ are the states s.t. $T \to q_i$ in $\mathcal{A}$, then $T \to \{q_1, \ldots, q_n\}$ in $\mathcal{A}_D$. The equivalence of $\mathcal{A}$ and $\mathcal{A}_D$ is a direct consequence of this remark. □

Since Presburger arithmetic is decidable, one can decide if a condition $\varphi(N)$ of a rule is satisfiable or not. From now on, the rules with unsatisfiable conditions are discarded.

5.3. Boolean Properties

PROPOSITION 18. *The class of languages accepted by conditional tree automata is closed under union, complement and intersection.*

Proof The complement of a language accepted by a (deterministic) conditional automaton is accepted by the automaton obtained by exchanging the final states and the non-final states. The union is accepted by an automaton which is the union of $\mathcal{A}_1$ accepting $\mathcal{L}_1$ and of $\mathcal{A}_2$ accepting $\mathcal{L}_2$. The closure under intersection is a consequence of the closure under complement and union (a direct proof also exists). □

5.4. Decision of emptiness

In this section, $\mathcal{A} = ((Q_f)_{f \in F}, Q_{nac}, Q_{Final}, R)$ is a conditional tree automaton and N_0 is a fixed integer greater than the number of states of $\mathcal{A}$. Our first result states the decision of emptiness for generalized terms.

PROPOSITION 19. *It is decidable if there is a generalized term accepted by $\mathcal{A}$*

Proof For each condition $\varphi(N)$ one can compute some n s.t. $\models \varphi(n)$. Moreover if there is some T s.t.$T \rightarrow q_1$ and some $\varphi(N)$: $f[N.q_1] \rightarrow q \in R$, then $f[n.T] \rightarrow q$, and if $T_1 \rightarrow q_1, T_2 \rightarrow q_2, q_1 \sqcup_f q_2 \rightarrow q$ then $T_1 \sqcup_f T_2 \rightarrow q$. Therefore a straightforward modification of the classical algorithm to decide emptiness of tree automata also works for conditional tree automata. □

Unfortunately, generalized terms are not suitable for modelling non-linearity, and we must restrict ourselves to normalized terms for solving the AC-complement problem. Therefore the decision of emptiness becomes more involved and some preliminary results are needed.

PROPOSITION 20. *Let $T = f[n_1.T_1, \ldots, n_p.T_p]$ a normalized term s.t. $T \rightarrow q$ and p is greater than or equal to N_0, then there exists a normalized term $S = f[m_1.S_1, \ldots, m_k.S_k]$ s.t. $k < p$ and $S \rightarrow q$.*

Proof Let $q_1, \ldots, q_p$ be the states s.t. $f[n_p.T_p] \rightarrow q_p, \ldots, f[n_i.q_i, \ldots, n_p.T_p] \rightarrow q_i, \ldots,$ $f[n_1.T_1, \ldots, n_p.T_p] \rightarrow q_1 = q$. The sequence $q_1, \ldots, q_p$ has more elements than the number of states of $\mathcal{A}$, therefore there exist some $i > j$ s.t $q_i = q_j$.
Let $S = f[n_1.T_1, \ldots, n_{i-1}.T_{i-1}, n_j.T_j, \ldots, n_p.T_p]$ which has the required form $f[m_1.S_1, \ldots,$ $m_k.S_k]$ with $k < p$. Since $f[n_i.T_i, \ldots, n_p.T_p] \rightarrow q_i$ and $f[n_j.T_j, \ldots, n_p.T_p] \rightarrow q_j$ with $q_i = q_j$, then $S \rightarrow q_1 = q$. □

PROPOSITION 21. *One can compute B s.t. for each condition $\varphi(N)$ either each integer n validating φ is less than B or there are at least N_0 integers less than B validating φ.*

Proof One tests if $\varphi(N)$ has a finite number of solutions by checking if $\exists P : \varphi(N) \Rightarrow N < P$ is true. If $\varphi(N)$ has a finite number of solutions, it is easy to find a bound B' on these solutions by testing $\exists n : n > k \wedge \varphi(n)$ for $k = 1, 2, \ldots$, and if $\varphi(N)$ has an infinite number of solutions, one computes the first N_0 ones by checking $\models \varphi(k)$ for $k = 1, 2, \ldots$.

The bound B is the maximum of B' and of the maximal values needed to get N_0 solutions for conditions having an infinite set of solutions. $\square$

In what follows, given a set $\mathcal{S}$ of normalized terms, $N_P(\mathcal{S})$ denotes the number of distinct congruence classes for $=_P$ in $\mathcal{S}$.

Combining the previous results, one can design the following algorithm to decide the emptiness of the set of normalized terms accepted by a conditional tree automaton:

```
For each state $q$ do set $\mathcal{L}_q^0 = \emptyset$.
i=1.
Repeat   For each state $q$ do
             $\mathcal{L}_q^i \leftarrow \mathcal{L}_q^{i-1}$
             if $N_P(\mathcal{L}_q^{i-1}) < N_0$
             then     repeat   add to $\mathcal{L}_q^i$ all normalized terms
                               $T = g(T_1, \ldots, T_n)$ or $f[n_1.T_1, \ldots, n_p.T_p]$ s.t.
                               $T_j \in \mathcal{L}_{q_{i_j}}^{i-1}$, $n_j \leq B$ and $T \rightarrow q$
                      until    no new term can be added or $N_P(\mathcal{L}_q^i) \geq N_0$
         i=i+1
until $\exists q \in Q_F$ s.t. $\mathcal{L}_q^i \neq \emptyset$ or $\forall q, \mathcal{L}_q^i = \mathcal{L}_q^{i-1}$
if $\exists q \in Q_{Final}$ s.t. $\mathcal{L}_q^i \neq \emptyset$ then return not empty
else return empty
```

This algorithm allows us to state the following theorem:

THEOREM 5. *It is decidable if there is a normalized term accepted by $\mathcal{A}$*

Proof The algorithm terminates since the congruence class $[T]_P$ of any normalized term is finite. The proof of correctness requires more work. The $\xrightarrow{N}$ relation is defined by:

- $g(T_1, \ldots, T_n) \xrightarrow{N} q$ if $T_i \xrightarrow{N_i} q_i$ with $N_i < N$ and $g(q_1, \ldots, q_n) \rightarrow q \in R$,
- $f[n_1.T_1, \ldots, n_p.T_p] \xrightarrow{N} q$ if $T_i \xrightarrow{N_i} q_i$ with $N_i < N$ and $f[n_1.T_1, \ldots, n_p.T_p] \rightarrow q$.

Then we define $\mathcal{M}_q^N = \{T \mid T \xrightarrow{R} q \textit{ for } R \leq N\}$. By definition one has $\mathcal{L}_q^N \subseteq \mathcal{M}_q^N$. The proof of the theorem is based upon the following lemma:

LEMMA 1. *For each q, N, either $\mathcal{L}_q^N = \mathcal{M}_q^N$ or $N_P(\mathcal{M}_q^N)$ and $N_P(\mathcal{L}_q^N)$ are greater than or equal to N_0*

The proof is by induction on N.

- $N = 1$: obvious.

- Let the property be true for any $R < N$. Since the result is obvious when $\mathcal{M}_q^N = \emptyset$ we suppose that $\mathcal{M}_q^N \neq \emptyset$.
  - let q be of sort *nac*. Then,
    - either there is some $T = g(T_1, \ldots, T_n)$ s.t. $T \xrightarrow{N} q$ and $N_P(\mathcal{M}_{q_i}^R) \geq N_0$ where $T_i \xrightarrow{R} q_i$ and $R < N$,
    - or for each $T = g(T_1, \ldots, T_n)$ s.t. $T \xrightarrow{N} q$ with $T_i \xrightarrow{R} q_i$, $N_P(\mathcal{M}_{q_i}^R) < N_0$.

    In the last case, the induction hypothesis yields that $\mathcal{M}_q^N = \{T = g(T_1, \ldots, T_n) \mid T_i \in \mathcal{L}_{q_i}^R\}$ and we are done. In the first case one gets that $N_P(\mathcal{M}_q^N) \geq N_0$ and $N_P(\mathcal{L}_q^N) \geq N_0$.
  - let q be of sort f. Then,
    - either there is some $T = f[n_1.T_1, \ldots, n_p.T_p]$ s.t. $T \xrightarrow{N} q$ where $T_{i_0} \xrightarrow{R} q_{i_0}$ and $N_P(\mathcal{M}_{q_{i_0}}^R) \geq N_0$ for some i_0,
    - or for each $T = f[n_1.T_1, \ldots, n_p.T_p]$ s.t. $T \xrightarrow{N} q$, $N_P(\mathcal{M}_{q_i}^R) < N_0$ where $T_i \xrightarrow{R} q_i$.

    The last case works as above since $\mathcal{L}_q^N = \{T = f[n_1.T_1, \ldots, n_p.T_p] \mid T_i \in \mathcal{L}_q^R = \mathcal{M}_q^R,\ n_i < B\}$ and $\mathcal{M}_q^N = \{T = f[n_1.T_1, \ldots, n_p.T_p] \mid T_i \in \mathcal{M}_q^R = \mathcal{L}_q^R\}$ and we are done.
    In the first case, we can suppose that $p < N_0$ (use proposition 20). We show that we can construct at least N_0 normalized terms in $\mathcal{L}_q^N$.
    Let $T_{j_1}, \ldots, T_{j_k}$ be the T_i's s.t. $T_{j_l} \rightarrow q_{i_0}$ (hence $k \leq p < N_0$). Let $S = f[n_1.S_1, \ldots, n_p.S_p]$ be as follows:
    $S_i = T_i$ if i is not one of the $j_1, \ldots, j_k$, otherwise S_{j_1} is some term of $\mathcal{L}_{q_{i_0}}^P$, S_{j_2} is some term of $\mathcal{L}_{q_{i_0}}^P$ s.t $S_{j_2} \neq_P S_{j_1}$, ..., S_{j_k} is some term of $\mathcal{L}_{q_{i_0}}^P$ s.t. $S_{j_k} \neq_P S_{j_1}$ and ...and $S_{j_k} \neq_P S_{j_k}$.
    Therefore one can construct at least $N_0.(N_0-1)\ldots(N_0-(k-1))/1.2\ldots k \geq N_0$ [†] non-equivalent normalized terms S s.t $S \in \mathcal{M}_q^N$ and we are done.

$\square$

6. Applications to complement problems and inductive reducibility

6.1. Restricted terms

Normalized terms and conditional tree automata have been introduced to deal to restricted terms, as defined by:

DEFINITION 8. *A term t is restricted iff*

- *either $Flat(t) = x$,*

[†] this is the number of possible choices of k objects among N_0 with $k < N_0$

- *or $Flat(t) = g(t_1, \ldots, t_n)$ where each t_i is restricted , t_i and t_j do not have a variable in common for $i \neq j$,*

- *or $Flat(t) = f(|t_1, \ldots, t_n|)$ where each t_i is restricted for $i = 1, \ldots, n$, t_i and t_j have a variable x in common for $i \neq j$ iff $t_i =_{AC} t_j$.*

Example The term $g(f(f(x,y), f(x,y)), g(z))$ is restricted but $f(g(f(x,y)), g(f(x,z)))$ is not restricted.

As in the linear case, we are interested in $AC(Gr(t))$. In the next proposition t is a term s.t.:

- t is restricted and

- $Flat(t) = f(|\underbrace{x_1, \ldots, x_1}_{m_1}, \ldots, \underbrace{x_q, \ldots, x_q}_{m_q}, \underbrace{t_1, \ldots, t_1}_{n_1}, \ldots, \underbrace{t_p, \ldots, t_p}_{n_p}|)$ where the t_i's are not variables and $t_i \neq_{AC} t_j$ for $i \neq j$.

By definition, $I(f[m_1.x_1, \ldots, m_p.x_p, n_1.T_1, \ldots, m_p.T_p]) = t$ iff $I(T_i) = t_i$ for $i = 1, \ldots, p$. The set of normalized ground AC-instances of t is characterized by:

Proposition 22.
$S \in Norm(AC(Gr(t)))$ iff $S = f[M_1.S_1, \ldots, M_r.S_r]$ with

- $M_k = \Sigma_{i \in I_k} n_i + \Sigma_{j \in J_k} k_j.m_j$

- *$I(S_k) \in AC(Gr(t_i))$ for each $i \in I_k$*

- *the sets I_k satisfy $\cup_k I_k = \{1, \ldots, p\}$ and $I_k \cap I_{k'} = \emptyset$ if $k \neq k'$,*

- *the sets J_k satisfy $\cup_k J_k = \{1, \ldots, q\}$*

Proof Let S be as above, then

$$Flat(I(S)) = I(S) = f(|\underbrace{S_1, \ldots, S_1}_{M_1}, \ldots, \underbrace{S_p, \ldots, S_p}_{M_p}|)$$

$$I(S) =_P f(|\underbrace{S_{j_1}, \ldots, S_{j_1}}_{(\Sigma_l k_l).m_1}, \ldots, \underbrace{S_{j_q}, \ldots, S_{j_q}}_{(\Sigma_l k_l).m_q}, \underbrace{S_{i_1} \ldots, S_{i_1}}_{n_1}, \ldots, \underbrace{S_{i_p} \ldots, S_{i_p}}_{n_p}|)$$

where $S_{i_l} \in AC(Gr(t_l))$

$$I(S) =_P Flat(t\theta)$$

for some θ since the x_i's do not occur in the t_i's and t_i and t_j have no variables in common for $i \neq j$.

Conversely, let $S \in Norm(AC(Gr(t)))$, then

$I(S) =_{AC} t\theta$ for some θ,

$I(S) =_P Flat(t\theta)$

$$I(S) =_P \underbrace{Flat(x_1\theta) \sqcup_f \ldots \sqcup_f Flat(x_1\theta)}_{m_1} \sqcup_f \ldots \sqcup_f \underbrace{Flat(x_q\theta) \sqcup_f \ldots \sqcup_f Flat(x_q)\theta}_{m_q} \sqcup_f$$
$$\underbrace{Flat(t_1\theta) \sqcup_f \ldots \sqcup_f Flat(t_1\theta)}_{n_1} \sqcup_f \ldots \sqcup_f \underbrace{Flat(t_p\theta) \sqcup_f \ldots \sqcup_f Flat(t_p\theta)}_{n_p}$$

$$I(S) = \underbrace{I(S_1) \sqcup_f \ldots \sqcup_f I(S_1)}_{M_1} \sqcup_f \ldots \sqcup_f \underbrace{I(S_r) \sqcup_f \ldots \sqcup_f I(S_r)}_{M_r}$$

where S_i and M_i are as above, then S has the required form. □

6.2. Solution of restricted complement problems

Now we prove the decidability of AC-complement problems for restricted terms. The key proposition is the next one and it also has a more elegant non-combinatorial proof which unfortunately lacks the constructive aspects we are interested in.

PROPOSITION 23. *Let t be a restricted term then $Norm(AC(Gr(t)))$ is accepted by a conditional tree automaton.*

Proof The proof is by structural induction on $Flat(t)$.

- $Flat(t) = x$ the result is obvious.
- $Flat(t) = g(t_1, \ldots, t_n)$. Let $\mathcal{A}_i$ be an automaton accepting $Norm(AC(Gr(t_i)))$, then an automaton accepting $Norm(AC(Gr(t)))$ is obtained by taking all the states of the $\mathcal{A}_i$ plus a new final state q_F, all the rules of the $\mathcal{A}_i$ plus the rules $g(q_1, \ldots, q_n) \to q_F$ where q_i is a final state of $\mathcal{A}_i$ for $i = 1, \ldots, n$.
- $Flat(t) =_P f(\| \underbrace{x_1, \ldots, x_1}_{m_1}, \ldots, \underbrace{x_q, \ldots, x_q}_{m_q}, \underbrace{t_1, \ldots, t_1}_{n_1}, \ldots, \underbrace{t_p, \ldots, t_p}_{n_p} \|)$. Let $\mathcal{A}_i$ be an automaton accepting $Norm(AC(Gr(t_i)))$ for $i = 1, \ldots, p$. From these automata, one can construct automata $\mathcal{A}_I$ accepting $\cap_{i\in I} Norm(AC(Gr(t_i)))$ for any $I \subseteq \{1, \ldots, p\}$ with $I \neq \emptyset$. An automaton accepting the set of all ground normalized terms is denoted by $\mathcal{A}_T$. An automaton accepting $Norm(AC(Gr(t)))$ can be constructed by taking all the states of the rules of the previous automata and by adding new states $q_{I,J}$ of sort f for each $I \subseteq \{1, \ldots, p\}, J \subseteq \{1, \ldots, q\}$ and new rules:
  - $\exists k_j > 0 : N = \Sigma_{i\in I} n_i + \Sigma_{j\in J} k_j.m_j : f[N.q] \to q_{I,J}$ for each final state $q$of $\mathcal{A}_I$,
  - $\exists k_j > 0 : N = \Sigma_{j\in J} k_j.m_j : f[N.q] \to q_{\emptyset,J}$ for each final state q of $\mathcal{A}_T$,
  - $q_{I_1,J_1} \sqcup_f q_{I_2,J_2} \to q_{I_1 \uplus I_2, J_1 \cup J_2}$.

The unique final state of the automaton is $q_{\{1,\dots,p\},\{1,\dots,q\}}$. By construction, $\mathcal{A}$ is a conditional tree automaton since set union is commutative and associative, and the language accepted by $\mathcal{A}$ is $Norm(AC(Gr(t)))$ (use the previous proposition). $\square$

Example Let $t = f(0, x)$, we sketch the construction of the automaton accepting $Norm(AC(Gr(t)))$.

An automaton accepting $Norm(AC(Gr(x)))$ has the rules:

$$\begin{array}{l} 0 \to q \\ g(q \textit{ or } q_f) \to q \\ N \geq 1 : f[N.q] \to q_f \\ q_f \sqcup_f q_f \to q_f \end{array}$$

where q and q_f are final states.

An automaton accepting $Norm(AC(Gr(0)))$ has the rules:

$$0 \to q_0$$

where q_0 is the final state.

To get the automaton to accept $Norm(AC(Gr(t)))$, we add new states $q_{I,J}$. For clarity, we use subsets I (resp. J) of $\{0\}$ (resp. of $\{x\}$) instead of subsets of $\{1\}$ (resp. $\{1\}$). The new rules are:

$$\begin{array}{l} N = 1 : f(q_0) \to q_{\{0\},\emptyset} \\ \exists k > 0 : N = 1 + k.1 : f(N.q_0) \to q_{\{0\},\{x\}} \\ \exists k > 0 : N = k.1 : f(N.(q \textit{ or } q_f)) \to q_{\emptyset,\{x\}} \\ q_{\emptyset,\{x\}} \sqcup_f q_{\{0\},\emptyset} \to q_{\{0\},\{x\}} \text{ and } q_{\{0\},\emptyset} \sqcup_f q_{\emptyset,\{x\}} \to q_{\{0\},\{x\}} \\ q_{\{0\},\{x\}} \sqcup_f q_{\emptyset,\{x\}} \to q_{\{0\},\{x\}} \text{ and } q_{\emptyset,\{x\}} \sqcup_f q_{\{0\},\{x\}} \to q_{\{0\},\{x\}} \\ q_{\emptyset,\{x\}} \sqcup_f q_{\emptyset,\{x\}} \to q_{\emptyset,\{x\}} \text{ and } q_{\emptyset,\{x\}} \sqcup_f q_{\emptyset,\{x\}} \to q_{\emptyset,\{x\}} \end{array}$$

where $q_{\{0\},\{x\}}$ is the final state.

THEOREM 6. *The complement problem $t \neq_{AC} t_1 \wedge \dots \wedge t \neq_{AC} t_n$ where t and the t_i's are restricted terms is decidable.*

Proof The result holds since

- there is a solution iff there is a normalized term in $Norm(AC(Gr(t)))$ which does not belong to $Norm(AC(Gr(t_i)))$ for $i = 1, \dots, n$,
- $Norm(AC(Gr(s)))$ is accepted by a conditional automaton for each s,
- the class of languages accepted by conditional automata is closed under boolean operations and the emptiness of a language accepted by a conditional tree automaton is decidable.

□

6.3. Decision of inductive reducibility

The proof that the normalized ground AC-instances of a restricted term are accepted by a conditional tree automaton can be generalized to get a proof that $Norm(Red(t))$ is accepted by a conditional tree automaton if t is a restricted term ($Red(t)$ denotes the ground terms reducible by t). The proof is by induction on $Flat(t)$. The cases $Flat(t) = x$ or $Flat(t) = g(t_1, \dots, t_n)$ are easy and the case $Flat(t) =_P f(|x_1, \dots, x_q, t_1, \dots, t_p|)$ is dealt with as follows:

- construct an automaton accepting $Norm(AC(Gr(t))) \cup Norm(AC(Gr(x \sqcup_f t)))$,
- to this automaton, add a new final state q_{nac} and the rules $g(\dots, q_{nac}, \dots) \to q_{nac}, g(\dots, q_F, \dots) \to q_{nac}, N \geq 1 : f[N.q_{nac}] \to q_F, q_F \sqcup_f q \to q_F, q \sqcup_f q_F \to q_F$ where q_F is any final state of sort f, and q is any state of sort f.

Therefore one gets the theorem:

Theorem 7. *The inductive reducibility modulo AC of a restricted term for a term rewrite system with restricted left-hand side is decidable.*

6.4. Extensions

6.4.1. Generalized conditional tree automata

In section 3.3, we have introduced automata with equality tests between brothers and in section 5, we have discussed conditional tree automata. These classes can be merged into a single one, *generalized conditional tree automata* which are defined like conditional tree automata except that the rules $g(q_1, \dots, q_n) \to q$ are replaced by rules $\varphi : g(q_1, \dots, q_n) \to q$ where φ is a formula of $Form_n$ where $Form_n$ is defined by:

- $\#i \neq_P \#j \in Form_n$, $\top \in Form_n$,
- if $\varphi \in Form_n$ and $\psi \in Form_n$ then $\neg\varphi \in Form_n$, $\varphi \vee \psi \in Form_n$, $\varphi \wedge \psi \in Form_n$.

These generalized conditional automata accept sets of generalized terms, and have the same properties as conditional tree automata. They are used to extend the non-linear cases that we can deal with, i.e. we are able to handle *loosely restricted* terms. A term t is loosely restricted if $Flat(t) = x$ or $Flat(t) = g(t_1, \dots, t_n)$ where the t_i's are restricted and have no variable in common if they are not variables, or else $Flat(t) = f(|t_1, \dots, t_p|)$ where the t_i are restricted and either $t_i =_{AC} t_j$ or t_i and t_j have no variable in common. The proofs designed for conditional automata can be easily modified to work for generalized conditional automata, and we get our next result:

Theorem 8. *The complement problem $t \neq_{AC} t_1 \wedge \dots \wedge t \neq_{AC} t_n$ where t and the t_i's are loosely restricted terms is decidable. The inductive reducibility modulo AC of a loosely restricted term for a term rewrite system with loosely restricted left-hand side is decidable.*

6.4.2. Working on multisets

To solve the AC-complement problem, we focussed on normalized terms. But our solutions works also for similar problems on multisets, if we change the definition of normalized terms in order to allow terms like $f[n_1.T_1, \ldots, n_p.T_p]$ with $\sum_{i=1}^{i=p} = 1$. The same results hold and we can decide multiset inclusion, intersection, union and complement.

6.4.3. Adding idempotency

Another useful axiom is the idempotency axiom $f(x,x) = x$ called I. Our solution relying on conditional tree automata can be modified in order to handle this axiom together with associativity and commutativity, i.e $f \in F$ satisfies ACI. The normalized terms required for this solution will have the form $f[1.T_1, \ldots, 1.T_p]$ where T_i and T_j are not equivalent. Therefore one gets the result:

Theorem 9. *The complement problem $t \neq_{ACI} t_1 \wedge \ldots \wedge t \neq_{ACI} t_n$ where t and the t_i's are linear terms is decidable. The inductive reducibility modulo ACI of a linear term for a left-linear term rewrite system is decidable.*

Conclusion

We have shown that language theory and tree automata are useful for solving formuale in equational theories involving associativity and commutativity. The new class of tree automata that we have introduced can be used in other applications (sets and multisets for instance) and is interesting by itself. For example, similar tree automata are used in Niehren and Podelski (1993) to deal with feature terms a.k.a. multisets. Moreover, our solution can be easily extended to deal with sorted terms where the relations on sorts are described by a tree automaton. The decision procedures that we have given have a high complexity, but the intrinsic complexity of the problem is high (AC-matching is NP-complete and an AC unification problem can have an exponential number of solutions) and we prefered to give simple proofs rather than the most efficient algorithms. Many improvements can be devised in order to get better algorithms. For instance, dealing with flattened terms only and choosing an efficient representation of sets will decrease the complexity by one exponential.

Aknowledgements: the authors thanks one referee for his fruitful remarks and suggestions.

References

Bogaert, B.,Tison, S. (1992) Equality and disequality constraints on direct subterms in tree automata. In *Proceedings of the 9th Symposium on Theoretical Computer Science*, volume 577 of *Lecture Notes in Computer Science*, pages 161–172, 1992.

Comon, H., Lescanne, P. (1989). Equational problems and disunification. *Journal of Symbolic Computation*, 7(3 & 4):371–426, 1989. Special issue on unification. Part one.

Colmerauer, A. (1984) Equations and inequations on finite and infinite trees. In *Proceedings of FGCS'84*, pages 85–99, November 1984.

Dershowitz, N., Jouannaud, J-P. (1988) Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. Elsevier Science Publishers B. V. (North-Holland), 1990.

Gecseg, F., Steinby, M. (1984) *Tree automata*. Akadémiai Kiadó, Budapest, Hungary, 1984.

Jouannaud, J-P., Kounalis, E. (1989) Automatic proofs by induction in theories without constructors. *Information and Computation*, 82:1–33, 1989.

Kapur, D.,Narendran, P., Rosenkrantz, D.J., Zhang, H. (1991) Sufficient completeness, ground-reducibility and their complexity. *Acta Informatica*, 28:311–350, 1991.

Lassez, J-L., Marriot, K. (1986) Explicit representation of terms defined by counter examples. *Journal of Automated Reasoning*, 3(3):301–318, 1986.

Lassez, J-L., Maher, M., Marriott, K. (1991) Elimination of negation in term algebra. In A. Tarlecki, editor, *Proceedings 16th International Symposium on Mathematical Foundations of Computer Science, Kazimierz Dolny (Poland)*, volume 520 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 1991.

Lugiez, D., Moysset, J-L., (1993) Complement problems and tree automata in AC-like theories. In P. Enjalbert, A. Finkel, and K. W. Wagner, editors, *Proceedings STACS 93*, volume 665 of *Lecture Notes in Computer Science*, pages 515–524. Springer-Verlag, February 1993.

Maher, M. (1988) Complete axiomatization of the algebra of finite, rational trees and infinite trees. In *Proceedings 3rd IEEE Symposium on Logic in Computer Science, Edinburgh (UK)*. COMPUTER SOCIETY PRESS, 1988.

Malc'ev, A. (1971) Axiomatizable classes of locally free algebra of various type. In Benjamin Franklin Wells, editor, *The Metamathematics of Algebraic Systems: Collected Papers 1936-1967*, chapter 23, pages 262–281. North Holland, 1971.

Niehren, J., Podelski, A., (1993) Feature automata and recognizable sets of feature trees. In M.C. Gaudel and J.P. Jouannaud, editors, *Proceedings TAPSOFT'93*, volume 668 of *Lecture Notes in Computer Science*, pages 356–375. Springer-Verlag, 1993.

Treinen, R., (1992) A new method for undecidability proofs of first order theories. *Journal of Symbolic Computation*, 14(5):437–458, 1992.