A Unified View of Tree Automata and Term Schematisations

Nicolas Peltier

LIG, CNRS
46, avenue Félix Viallet
38031 Grenoble Cedex, France
Nicolas.Peltier@imag.fr

Abstract. We propose an extension of tree automata, called N-automata, which captures some of the features of term schematisation languages, for instance the use of counter variables and parameters. We show that the satisfiability problem is decidable for positive, purely existential, membership formulae which permits to include the proposed formalism into most existing symbolic computation procedures (such as SLD-resolution).

1 Introduction

Formalisms able to handle infinite sets of terms (and manipulate them) are useful in various domains of computer science, for instance for preventing divergence of symbolic computation procedures (such as resolution, superposition, etc.). Among these formalisms, *tree automata* (TA) play a central rôle, mainly due to their nice computational properties [4]: the set of regular term languages (i.e. the languages representable by a tree automaton) is closed under all boolean operations (intersection, union and complement) and the emptiness problem (i.e. the problem of deciding whether a given automaton denotes an empty set of terms) is decidable. TA have many applications, for instance in rewriting [7, 13] or constraint solving [12]. As for word automata, a TA can be defined by a set of states and by a transition function, and the set of recognized terms is specified by a final state. Alternatively, it can be seen as a set of (Horn) monadic clauses satisfying some additional properties, where each predicate corresponds to a state. The recognized language is simply the interpretation of the final predicate in the minimal model of this set of clauses. Using this view, TA can be easily extended by considering non-monadic predicate symbols [1, 9, 10], representing (synchronized) *term tuple languages*.

Other formalisms, called *term schematisations* (TS), have been proposed during the 90's to denote infinite sequences of structurally similar terms. The idea is to denote infinite sequences of terms obtained by starting from a given base term s and by iterating from s a particular "context" $C[\diamond]$, where $\diamond$ is a distinguished subterm in C (denoting a "hole"). If $C[x]$ denotes the term obtained from C by replacing $\diamond$ in C by x, we get the sequence s, $C[s]$, $C[C[s]]$, $\dots C^n[s]$. For instance, the set of terms $x, g(g(x)), g(g(g(g(x)))), \dots, g^{2m}(x)$ is obtained by iterating m times the context $C = g(g(\diamond))$ on the base term x (m denotes an arithmetic variable).

There exist several classes of term schematisation languages corresponding to different classes of contexts: the *recurrent terms* [2] (unique context with only one hole), the *terms with integer exponents* [3] (arbitrary contexts with one hole), the *R-terms* [14] (contexts containing several holes) and the most expressive language of *primal grammars* [8], in which the contexts can depend on the rank in the iteration. Unification is decidable for all these languages.

There are important differences between TA and TS and the representable languages are not comparable. TS allow one to denote terms containing several occurrences of the same (non ground) term, which is not possible using TA[1]. For instance, one can denote using a TS a list of the form $[x, x, x, x, \ldots, x]$, where x is an arbitrary term. The list is obtained by iterating the context $cons(x, \diamond)$ on the base term *nil*. This set of terms cannot be denoted by a TA because this would require an arbitrary number of equality tests. Moreover, TS use arithmetic variables to count the number of iterations in the sequences. This feature can be used for instance to denote the sequence $(f^n(g^n(a)))_{n\in\mathbb{N}}$ which is well known to be non regular, i.e. not representable be a (tree) automaton.

On the other hand, TA can denote many sets of terms that are not representable by a TS. Indeed, a TS cannot denote a term containing an arbitrary number of variables: for instance it is not possible to denote the sequence $a, f(x_1, a), f(x_2, f(x_1, a)), f(x_3, f(x_2, f(x_1, a))), \ldots$, because the variables cannot depend on the rank of the iteration[2]. Moreover, more flexible iterations can be denoted using TA, with non-unique contexts, for instance one can denote the term $f(t_1, f(t_2, f(t_3, \ldots (f(t_n, x)) \ldots)))$ where for all $i \in [1..n]$, $t_i \in \{b, c\}$. Such a term cannot be described using existing TS. More generally, one can denote iterations combining different contexts.

A very natural question arises: is it possible to *unify* these two approaches? The goal is to define a formalism that *combines* all the above features: use of *counter variables*, *indexed* and *non-indexed* variables and *non-unique contexts*. Ideally, it should be *strictly more expressive* than both approaches, and hopefully also more expressive than the union of the two languages, because some "hybrid" terms representable neither by TA nor by TS could be denoted by combining both approaches. Of course, a basic requirement is that both emptiness *and* unification problems should remain decidable.

The present paper is a first answer to this problem. More precisely, we propose a (strict) extension of tree automata, called N**-automata**. We shall prove that this formalism strictly subsumes the terms with integer exponents of [3]. Other, more expressive term schematisation languages are non comparable with N-automata.

[1] Some limited equality tests can be safely considered [4].

[2] The formalism of *non flat* primal grammar does offer the possibility of considering "indexed" (or *marked*) variables but unification is decidable only for flat primal grammars, i.e. for primal grammars without indexed variables.

As in [1, 9], we extend TA by adding additional parameters to the states. Some of these parameters are arithmetic variables allowing one to count the number of times the automaton enters some specific states. The other ones denote standard terms, that are allowed to occur several times into the terms recognized by the automaton. They play the same rôle as (non-indexed) variables in TS. The language recognized by the N-automaton depends on the value of the above parameters.

We shall show that the emptiness problem is decidable for N-automata. Moreover, the set of recognized languages is stable under intersection. More generally, we define a notion of N^+-*formulae*, which are positive and purely existential logical formulae combining arithmetic (linear) equality with atoms of the form $p(t_1, \ldots, t_n, s)$, meaning that s occurs in the language recognized by the N-automaton at state p, using $t_1, \ldots, t_n$ as parameters (one can view N^+-formulae as existential, positive, "membership" formulae [5]). With these semantics, we show that the satisfiability problem is decidable by providing an algorithm transforming any (closed) N^+-formula into a purely arithmetic formula. N^+-formulae subsume both emptiness and unification problems. To the best of our knowledge there is no formalism sharing these features[3]. Our results do not follow from the ones in [15] since the iterations we consider cannot be expressed using positive formulae built on equality and subterm ordering, nor from the ones in [11], because the (ground) rewrite rules in [11] are not comparable with the iterations we use in the present paper, and also because the considered problems are different.

Due to space restriction the proofs are not included.

2 Preliminaries

We denote by T_Σ the set of terms constructed as usual on a set of *function symbols* Σ and on a set of *ordinary variables* $\mathcal{X}$ and by T_N the set of arithmetic terms built on the function symbols $0, succ, +$ and on a set of *arithmetic variables* $\mathcal{X}_N$ disjoint from $\mathcal{X}, \Sigma$. As usual the term $succ^n(0)$ is simply denoted by n. A term (arithmetic or standard) is said to be *ground* iff it contains no variable.

We shall consider predicate symbols whose arguments will be either natural numbers or standard terms. Thus, we assume a set of *predicate symbols* Ω is given with a function pr mapping each symbol $p \in \Omega$ to a *profile* $pr(p)$, which is a finite sequence $\tau_1 \times \ldots \times \tau_n$ where n denotes the arity of p and where for every $i \in [1..n]$, τ_i is either `int` (natural numbers) or `t` (standard terms). A predicate is said to be *monadic* if its arity is 1. If O is a subset of Ω then $Atom(O)$ denotes the set of *atoms* of the form $p(t_1, \ldots, t_n)$ where p is a predicate symbol of profile

[3] For instance the languages in [1] or [9] are very expressive, but lack the same decidability results.

$(\tau_1, \ldots, \tau_n)$ and for all $i \in [1..n]$, if $\tau_i = \mathtt{t}$ then $t_i \in T_\Sigma$, and if $\tau_i = \mathtt{int}$ then $t_i \in T_N$.

A *substitution* is a function mapping each ordinary variable in $\mathcal{X}$ to a term in T_Σ and each arithmetic variable in $\mathcal{X}_N$ to an arithmetic term in T_N. As usual a substitution can be extended to a homomorphism of T_Σ, T_N and $Atom(\Omega)$. The image of a term t by a substitution σ is denoted by $t\sigma$. Two terms t, s are said to be *unifiable* iff there exists a substitution σ s.t. $t\sigma = s\sigma$. As usual two unifiable terms have a most general unifier. A substitution is *ground* if for all variables x, $x\sigma$ is ground.

A *rule* is a formula of the form $H_1 \wedge \ldots \wedge H_n \Rightarrow C$, where $H_1, \ldots, H_n, C$ are atoms such that all the variables occurring in $H_1, \ldots, H_n$ also occur in C. C is called the *head* of the rule and $H_1, \ldots, H_n$ are the *premises*. We may have $n = 0$, in this case $H_1 \wedge \ldots \wedge H_n \Rightarrow C$ is to be read as C.

The notions of interpretations, models etc. are defined as usual. It is well known that any set of rules S has a minimal model, denoted by $Mod(S)$.

3 *N*-Automata

For technical convenience we use a clausal view of tree automata. A tree automaton (in the usual sense) can be seen as a set of Horn monadic clauses. In this section we extend the definition to handle (some classes of) non-monadic predicate symbols.

3.1 Rules and Automata

We assume that the profile of every predicate symbol p (corresponding to a *state*) is of the form $\tau_1 \times \ldots \times \tau_n \times \mathtt{t}$. The last argument can be seen as the term to be recognized by the automaton, and the first ones correspond to parameters. We denote by $A_{\mathtt{int}}(p)$ the set of indices $i \in [1..n]$ s.t. $\tau_i = \mathtt{int}$ and by $A_{\mathtt{t}}(p)$ the set of indices s.t. $\tau_i = \mathtt{t}$.

We associate to every predicate p:

– a unique natural number *level*(p) used to control the "dependencies" between the predicates (recursive calls): if a predicate symbol p depends on another predicate symbol q, then the level of q must be lower or equal to the one of p.
– two disjoint sets $A_c(p) \subseteq A_{\mathtt{int}}(p)$ and $A_=(p) \subseteq A_{\mathtt{t}}(p)$. The elements of $A_c(p)$ are called the *counters* of p. Intuitively, $A_=(p)$ denotes the set of non arithmetic parameters that must be equal to the terms accepted by p (this corresponds to a kind of equality test: at any state p one can test that the

consider term is equal to a non arithmetic parameter, see Condition 1 below) and $A_c(p)$ denotes the arithmetic parameters used by p.

Definition 1. An *N-rule* is a rule $H \Rightarrow p(t_1, \ldots, t_n, s)$ satisfying the following conditions.

1. For all $i \in A_{\mathtt{t}}(p)$, if $i \in A_{=}(p)$ then $t_i = s$, otherwise t_i is a variable occurring only once in the head.
2. For all $i \in A_{\mathtt{int}}(p)$, t_i is either n_i or 0 or $succ(n_i)$, where n_i is a variable occurring only once in the head, and $t_i \neq n_i$ then $i \in A_c(p)$.
3. s is either a term of the form $f(x_1, \ldots, x_k)$ where $x_1, \ldots, x_k$ are distinct variables, or a variable and in this case H is empty.
4. If s is of the form $f(x_1, \ldots, x_k)$ then $H = \bigwedge_{i=1}^{k} q_i(s^i_1, \ldots, s^i_n, x_i)$ where:

   a. For all $i \in [1..k]$, *level*$(q_i) \leq$ *level*(p) and q_i has the same profile as p.
   b. For any $i \in [1..k]$, $j \in A_{\mathtt{int}}(p)$, if $t_j = succ(n_j)$, then $s^i_j = n_j$ or $s^i_j = 0$. Otherwise we have either $s^i_j = t_j$ or $s^i_j = 0$. Moreover, if $s^i_j \neq t_j$ then $j \in A_c(p)$.
   c. For any $i \in [1..k]$, $j \in A_{\mathtt{t}}(p)$, $s^i_j = t_j$.
   d. There exists *at most* one $i \in [1..k]$ s.t. the two following conditions hold: *level*$(q_i) =$ *level*(p) **and** there exists $j \in A_c(q_i)$ s.t. $s^i_j \neq 0$.
   Moreover, for all $j \in A_c(p)$ and for all $l \neq i$, we must have $s^l_j = 0$. A rule containing such a literal is called *inductive* and in this case the literal $q_i(s^i_1, \ldots, s^i_n, x_i)$ satisfying the above conditions is called the *principal* literal[4].
   e. If *level*$(q_i) =$ *level*(p) then $A_c(q_i) = A_c(p)$ and $A_{=}(q_i) = A_{=}(p) = \emptyset$.

In the particular case where $n = 0$, our definition coincides with the standard rules of tree automata (all the predicate symbols have the same level and there is no inductive rule).

Example 1. Here are examples of N-rules:

$$\begin{array}{ll}
p(x,n,m,y_1) \wedge r(x,0,m,y_2) \wedge p(x,0,m,y_3) & \Rightarrow p(x, succ(n), m, f(y_1,y_2,y_3)) \\
q(x,0,m,y) & \Rightarrow p(x,0,m,g(y)) \\
r(h(y),n,m,y) & \Rightarrow q(h(y),n,m,h(y)) \\
r(x,n,m,y) & \Rightarrow r(x,n,succ(m),i(y)) \\
 & r(x,n,0,a)
\end{array}$$

We have
level$(p) = 2$, *level*$(q) =$ *level*$(r) = 1$. $A_{\mathtt{int}}(p) = A_{\mathtt{int}}(q) = A_{\mathtt{int}}(r) = \{2,3\}$, $A_{\mathtt{t}}(p) = A_{\mathtt{t}}(q) = A_{\mathtt{t}}(r) = \{1\}$. $A_c(p) = \{2\}$, $A_c(q) = \emptyset$, $A_c(r) = \{3\}$, $A_{=}(p) = A_{=}(r) = \emptyset$, $A_{=}(q) = \{1\}$. The first and fourth clauses are inductive and the first literal is principal in these clauses.

[4] This condition is the most complex and non-intuitive one. Roughly speaking, it states that the counter variables can only be used along **one** position in the term. The remaining subterms should not depend on the variables in $A_c(p)$.

The reader can check that the minimal model of the above set of rules is the set of terms of the form $p(h(v), n, m, u), q(h(v), n, m, h(v)), r(t, n, m, v)$ where $t \in T_\Sigma$, $n, m \in \mathbb{N}$, $v = i^m(a)$ and $u = f(f(\ldots(f(f(g(h(v)), v, g(h(v)), v, g(h(v))), \ldots, v, g(h(v)) \ldots), v, g(h(v)))))$.

On the other hand the following rules are *not* N-rules:

$q(x, n, m, y) \Rightarrow p(g(x), n, m, f(y))$	Condition 1 is violated
$p(x, n, n, y) \Rightarrow p(x, n, n, g(y))$	Condition 2
$p(x, n, m, y) \Rightarrow p(x, n, m, f(y, y))$	Condition 3, $f(y, y)$ non linear
$p(x, n, m, y) \wedge q(x, n, m, y) \Rightarrow p(x, n, m, g(y))$	Condition 4, y occurs twice
$p(a, n, m, y) \Rightarrow p(x, n, m, g(y))$	Condition 4.c, a should be x
$p(x, n, m, y_1) \wedge r(x, n, m, y_2) \Rightarrow p(x, s(n), m, j(y_1, y_2))$	Cond. 4.d, arg. 2 of r is not 0

A rule is called a *p-rule* if its head is of the form $p(\mathbf{t})$ for some vector of terms $\mathbf{t}$.

Definition 2. An *N-automaton* $\mathcal{A}$ is a pair $(S_\mathcal{A}, \rho_\mathcal{A})$, where $S_\mathcal{A}$ is a set of predicate symbols (of arity > 0) and $\rho_\mathcal{A}$ is a set of N-rules built on the set of predicates $S_\mathcal{A}$ s.t. for every $p \in S_\mathcal{A}$:

- $\rho_\mathcal{A}$ contains at most one inductive p-rule.
- There exists no pair of distinct rules with unifiable heads (in the usual setting this means that the automaton is deterministic).

For any $n+1$-ary predicate symbol $p \in S_\mathcal{A}$, $(t_1, \ldots, t_m)$ is said to be a *p-vector* iff $m = n$ and for every $i \in [1..n]$, $t_i \in T_\Sigma$ if $i \in A_{\mathtt{t}}(p)$ and $t_i \in T_N$ if $i \in A_{\mathtt{int}}(p)$. This implies that $p(t_1, \ldots, t_m, s)$ is an atom (where s denotes an arbitrary term in T_Σ).

Definition 3. (Accepted Language) Let $\mathcal{A}$ be an automaton and $p \in S_\mathcal{A}$. For any p-vector $\mathbf{t}$, we denote by $p_\mathcal{A}(\mathbf{t})$ the set of terms s s.t. $\mathit{Mod}(\rho_\mathcal{A}) \models p(\mathbf{t}, s)$. $p_\mathcal{A}(\mathbf{t})$ is *the language recognized by $\mathcal{A}$ at state p with parameters* $\mathbf{t}$.

Note that by definition, if $s \in p_\mathcal{A}(t_1, \ldots, t_n)$ then for every $i \in A_=(p)$, we have $t_i = s$.

We need to introduce some additional notations. Let $\mathcal{A}$ be an N-automaton. We write $p \geq_\mathcal{A} q$ iff there exists a p-rule $H \Rightarrow p(\mathbf{t})$ s.t. q occurs in H. $\geq_\mathcal{A}^*$ denotes the reflexive and transitive closure of $\geq_\mathcal{A}$. An index i is said to be an *inductive counter* for p if there exists a predicate symbol q s.t. $p \geq_\mathcal{A}^* q$ and i is a counter for q. The set of inductive counters of a predicate p is denoted by $IC_\mathcal{A}(p)$.

A natural number i is said to be *active* for a predicate symbol p if $i \in A_c(p) \cup A_=(p)$. It is said to be *inductively active* if there exists a predicate symbol q s.t. $p \geq_\mathcal{A}^* q$ and i is active for q. The set of inductively active arguments of a predicate p is denoted by $IA_\mathcal{A}(p)$. An essential property of $IA_\mathcal{A}(p)$ is that if $i \notin IA_\mathcal{A}(p)$ (i.e. if i is not inductively active for p) then the language $p_\mathcal{A}(\mathbf{t})$ does not depend on the i-th component of the vector $\mathbf{t}$.

Lemma 1. *Let $\mathcal{A}$ be an N-automaton. Let p be a $n+1$-ary predicate symbol in $S_{\mathcal{A}}$ and let $(s_1, \dots, s_n), (s'_1, \dots, s'_n)$ be two p-vectors s.t. for all $i \in [1..n]$, if $s_i \neq s'_i$ then $i \notin IA_{\mathcal{A}}(p)$.*

We have $p_{\mathcal{A}}(s_1, \dots, s_n) = p_{\mathcal{A}}(s'_1, \dots, s'_n)$.

A N-automaton is said to be *normal* iff all its rules are of the form $H \Rightarrow p(\mathbf{t}, f(x_1, \dots, x_k))$ for some function symbol f (with possibly $k = 0$). It is easy to see that any N-automaton can be transformed into an equivalent normal automaton.

Lemma 2. *For any N-automata $\mathcal{A}$ one can construct a normal $\mathcal{A}$-automaton $\mathcal{A}'$ s.t. for all $p \in S_{\mathcal{A}}$ and for all ground p-vectors $\mathbf{t}$: $p_{\mathcal{A}}(\mathbf{t}) = p_{\mathcal{A}'}(\mathbf{t})$.*

3.2 N^+-Formulae

Sometimes N-automata alone are not expressive enough and one has to add conditions on the parameters, in particular arithmetic conditions. Rather than including them into the rules, it is more convenient to put them outside the automaton, yielding the following definition:

Definition 4. The set of N^+-*formulae* for an N-automaton $\mathcal{A}$ is the smallest set of formulae satisfying the following properties:

- *true*, *false* are N^+-formulae.
- Any atom $p(t_1, \dots, t_n)$ in $Atom(S_{\mathcal{A}})$ is an N^+-formula.
- If $t, s \in T_{\Sigma}$ or $t, s \in T_N$ then $t = s$ is an N^+-formula.
- If ϕ, ψ are N^+-formulae, then $\phi \vee \psi$ and $\phi \wedge \psi$ are N^+-formulae.
- If ϕ is an N^+-formula and x is a variable (occurring either in $\mathcal{X}$ or in $\mathcal{X}_N$) then $(\exists x)\phi$ is an N^+-formula.

Definition 5. A ground substitution σ is said to be a *solution* of an N^+-formula ϕ w.r.t. an N-automaton $\mathcal{A}$ iff one of the following condition holds:

- ϕ is $t = s$, $t, s \in T_{\Sigma}$ and $t\sigma = s\sigma$.
- ϕ is $t = s, t, s \in T_N$ and $t\sigma$ and $s\sigma$ can be reduced to the same natural number by the usual rules of Presburger arithmetic: $0 + x \rightarrow x$ and $succ(x) + y \rightarrow succ(x + y)$.
- ϕ is $p(t_1, \dots, t_n, s)$ and $s\sigma \in p_{\mathcal{A}}(t_1\sigma, \dots, t_n\sigma)$.
- ϕ is $\phi_1 \vee \phi_2$ (resp. $\phi_1 \wedge \phi_2$) and σ is a solution of ϕ_1 or ϕ_2 (resp. ϕ_1 and ϕ_2).
- ϕ is $(\exists x)\phi$ and there exists a term t s.t. σ is a solution of $\phi\{x \rightarrow t\}$.

We denote by $sol_{\mathcal{A}}(\phi)$ the set of solutions of ϕ w.r.t. $\mathcal{A}$ and we write $\phi \equiv_{\mathcal{A}} \psi$ iff $sol_{\mathcal{A}}(\phi) = sol_{\mathcal{A}}(\psi)$.

3.3 Examples and Comparisons

Example 2. Let $\Sigma = \{a, f, g\}$. Let $\mathcal{A}$ be the N-automaton defined as follows: $S_{\mathcal{A}} \overset{\text{def}}{=} \{p, q, r, s\}$, $A_{\texttt{int}}(u) = \{1\}$, for all $u \in S_{\mathcal{A}}$.

$$\rho_{\mathcal{A}} \overset{\text{def}}{=} \begin{cases} q(0,x,y) \wedge p(n,x,l) & \Rightarrow p(succ(n), x, cons(y,l)) \\ & \Rightarrow p(0,x,nil) \\ r(0,x,y_1) \wedge s(0,x,y_2) & \Rightarrow q(0,x,f(y_1,y_2)) \\ & \Rightarrow r(0,x,x) \\ & \Rightarrow s(0,x,y) \end{cases}$$

We have $A_c(p) = \{1\}$, $A_c(q) = A_c(r) = A_c(s) = \emptyset$, $A_=(p) = A_=(q) = A_=(s) = \emptyset$, $A_=(r) = 2$. $p_{\mathcal{A}}(n,x)$ denotes the set of terms of the form $\{cons(f(x,y_1), cons(f(x,y_2), \ldots, cons(f(x,y_n), nil)\ldots))\}$ where $x, y_1, \ldots, y_n$ are arbitrary terms, i.e. the lists of the form $[f(x,y_1), \ldots, f(x,y_n)]$. Notice that this set of terms cannot be denoted by a standard tree automaton (due to the several occurrences of x), nor by any known term schematisation for which unification is decidable (due to the indexed variables $y_1, \ldots, y_n$).

The N^+-formula $(\exists m, x)[n = m + m \wedge r(0,x,a) \wedge p(n,x,y)]$ has the following set of solutions: $\{x \rightarrow a, n \rightarrow 2m, y \rightarrow [f(a,y_1), \ldots, f(a,y_{2m})]\}$, where $m \in \mathbb{N}$.

As already seen, N-automata are strict extensions of usual TA. Some of the existing extensions of TA could be included into N-automata, for instance we could add equality or disequality tests between brothers (i.e. between the variables $x_1, \ldots, x_k$ in Definition 1). We did not consider these additional possibilities in the present paper for the sake of simplicity and conciseness. We now compare N-automata and I-terms.

***N*-automata and *I*-Terms** The *terms with integer exponents* (or I-terms [3]) are a particular class of term schematisations. Formally speaking, the set of *I-terms* T_I and the set of *contexts* (terms with one hole) $T_\diamond$ are the least sets that satisfies the following conditions:

– $\mathcal{X} \subseteq T_I$ and $\diamond \in T_\diamond$.
– If $t_1, \ldots, t_n \in T_I^n$, and f is a function of arity n in Σ, then $f(t_1, \ldots, t_n) \in T_I$.
– If $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n \in T_I^{n-1}$, f is a function of arity n in Σ and $t_i \in T_\diamond$, then $f(t_1, \ldots, t_n) \in T_\diamond$.
– If $t \in T_\diamond$, $t \neq \diamond$, $n \in T_N$ and $s \in T_I$, then $t^n.s \in T_I$.

If t is a term in $T_\diamond$ and $s \in T_I$ then $t[s]$ denotes the term of T_I obtained by replacing $\diamond$ with s, formally defined as follows: $\diamond[s] \overset{\text{def}}{=} s$, $f(t_1, \ldots, t_n)[s] \overset{\text{def}}{=} f(t_1[s], \ldots, t_n[s])$ and $(t^n.u)[s] \overset{\text{def}}{=} t^n.u$. Then the semantics of (ground) I-terms is given by the following rewriting rules: $t^0.s \rightarrow s$ and $t^{n+1}.s \rightarrow t[t^n.s]$. Using these two rules (and the usual arithmetic rules), any ground I-term t can be transformed into a standard term $t\downarrow$. For instance, the I-term $f(x,\diamond)^n.a$ denotes the term $f(x, f(x, \ldots, f(x,a)\ldots))$. The next lemma shows that I-terms can be denoted by N^+-formulae.

Lemma 3. *Let t be an I-term of free variables $x_1, \dots, x_n$. There exist an N-automaton $\mathcal{A}_t$ and an N^+-formula $\phi_t(y)$ of free variables $x_1, \dots, x_n, y$ (where y does not occur in $x_1, \dots, x_n$) s.t. for every ground substitution σ, we have: $\sigma \in sol_{\mathcal{A}_t}(\phi_t(y))$ iff $y\sigma = t\sigma\downarrow$. Moreover, $\phi_t(y)$ contains no equation between non-arithmetic terms.*

Remark 1. Consequently, any unification problem $t = s$ between I-terms can be associated to an N^+-formula $\psi = (\exists x)[\phi_t(x) \wedge \phi_s(x)]$ s.t. $(t\sigma)\downarrow = (s\sigma)\downarrow$ iff σ is a solution of ψ w.r.t. the union of the automata $\mathcal{A}_t$ and $\mathcal{A}_s$.

Example 3. The equation $y = f(g(\diamond), x)^n.a$ is equivalent to the N^+-formula: $p(x, n, y)$ where p is defined by the rules:

$$\begin{array}{lll} q(x,n,y_1) \wedge r(x,0,y_2) & \Rightarrow p(x, succ(n), f(y_1,y_2)) & \quad p(x,0,a) \\ p(x,n,y) & \Rightarrow q(x,n,g(y)) & \quad r(x,0,x) \end{array}$$

We have $A_{\mathtt{int}}(u) = \{2\}$ and $A_{\mathtt{t}}(u) = \{1\}$, for every $u \in \{p,q,r\}$, $A_c(p) = A_c(q) = \{2\}$, $A_c(r) = A_=(p) = A_=(q) = \emptyset$ and $A_=(r) = \{1\}$.

Unfortunately, the previous result does not extend to other more expressive term schematisation languages such as primal grammars. This is mainly due to the possibility of "diagonalisation" i.e. inductive contexts depending on the rank of the iteration, as in the term $f(g^n(x), f(g^{n-1}(\dots, f(g(x), x))))$. Such a term can be expressed easily by a primal grammar, but it cannot be denoted by an N-automaton. Thus N-automata do not subsume primal grammars and the two formalisms are not comparable.

4 Intersection

In this section, we show how to compute the intersection of two languages denoted by N-automata, which is the first step toward solving N^+-formulae. The obtained language can itself be denoted by an N-automaton. This is more complicated than in the case of standard tree automata, because one has to handle the additional parameters, but the procedure is similar.

Two predicate symbols p, q are said to be *disjoint* in an N-automaton $\mathcal{A}$ if $pr(p) = pr(q)$ and $IC_{\mathcal{A}}(p) \cap IC_{\mathcal{A}}(q) = \emptyset$. We first show how to handle this particular case.

Let $\mathcal{A}$ be an N-automaton. We denote by $S^\star_{\mathcal{A}}$ the set of predicates $[p,q]^I$ where p, q are disjoint symbols of arity $n+1$ in $S_{\mathcal{A}}$ and $I \subseteq [1..n]$. Intuitively, we will have $[p,q]^I{}_{\mathcal{A}}(\mathbf{t}) = p_{\mathcal{A}}(\mathbf{t}) \cap p_{\mathcal{A}}(\mathbf{t})$, if the I-components of $\mathbf{t}$ are 0 (I is useful mainly to ensure that the level decreases). We construct an automaton $\hat{\mathcal{A}}$ defined on the set of predicate symbols $S_{\mathcal{A}} \cup S^\star_{\mathcal{A}}$ as follows.

We first define: $level([p,q]^I) \stackrel{\text{def}}{=} level(p) + level(q) + arity(p) - |I|$, $pr([p,q]^I) \stackrel{\text{def}}{=} pr(p) = pr(q)$, $A_c([p,q]^I) \stackrel{\text{def}}{=} A_c(p) \cup A_c(q)$ and $A_=([p,q]^I) \stackrel{\text{def}}{=} A_=(p) \cup A_=(q)$.

A substitution θ is said to be a *I-unifier* of two vectors $(t_1, \ldots, t_n)$ and $(s_1, \ldots, s_n)$ iff for every $i \in [1..n]$ we have $t_i\theta = s_i\theta$ and if $i \in I$ then $t_i\theta = s_i\theta = 0$.

We denote by $R_{\mathcal{A}}$ the set of rules of the form $H\theta \wedge H'\theta \rightarrow [p,q]^I(\mathbf{t}\theta)$ s.t. p, q are two $n+1$-ary predicate symbols in $S_{\mathcal{A}}$, I is a subset of $[1..n]$, $H \Rightarrow p(\mathbf{t}), H' \rightarrow q(\mathbf{s})$ are two rules in $\rho_{\mathcal{A}}$ and θ is the most general I-unifier of $\mathbf{t}$ and $\mathbf{s}$.

Lemma 4. *Let $\mathcal{A}$ be an N-automaton. Let $\mathcal{I} = Mod(\rho_{\mathcal{A}})$ and $\mathcal{J} = Mod(R_{\mathcal{A}} \cup \rho_{\mathcal{A}})$. For any pair of disjoint predicate symbols (p,q) of arity n, for every $I \subseteq [1..n]$ for every term s and for every ground p-vector $(t_1, \ldots, t_n)$ we have $\mathcal{J} \models [p,q]^I(t_1, \ldots, t_n, s)$ iff $\forall i \in I, t_i = 0$ and $\mathcal{I} \models p(t_1, \ldots, t_n, s) \wedge q(t_1, \ldots, t_n, s)$.*

In particular, Lemma 4 shows that the language denoted by the predicate $[p,q]^{\emptyset}$ is the intersection of the languages denoted by p and q, which is the desired result, but of course the rules in $R_{\mathcal{A}}$ are not N-rules. In order to transform them into N-rules with the same minimal model, we introduce the following rewrite rules (operating on rules):

Merging: $[p(t_1, \ldots, t_n, x) \wedge q(t_1, \ldots, t_n, x) \wedge H \Rightarrow C] \longrightarrow$
$[p,q]^I(t_1, \ldots, t_n, x) \wedge H \Rightarrow C$
if p, q are disjoint and I is the set of indices $i \in [1..n]$ s.t. $t_i = 0$.
Agreement: $[p(t_1, \ldots, t_n, x) \wedge q(s_1, \ldots, s_n, x) \wedge H \Rightarrow C] \longrightarrow$
$[p(t_1, \ldots, t_{i-1}, s_i, t_{i+1}, \ldots, t_n, x) \wedge q(s_1, \ldots, s_n, x) \wedge H \Rightarrow C]$
if $p \in S_{\mathcal{A}}$ and $i \notin IA_{\mathcal{A}}(p)$.

It is clear that these rules terminate on any set of rules. We denote by $R_{\mathcal{A}}\downarrow$ an arbitrarily chosen normal form of $R_{\mathcal{A}}$ w.r.t. the two rules above. The two following lemmata show in some sense the soundness and completeness of the above rules.

Lemma 5. *Let $\mathcal{A}$ be an N-automaton. $Mod(R_{\mathcal{A}} \cup \rho_{\mathcal{A}}) = Mod(R_{\mathcal{A}}\downarrow \cup \rho_{\mathcal{A}})$.*

Lemma 6. *Let $\mathcal{A}$ be an N-automaton. $(S^{\star}_{\mathcal{A}} \cup S_{\mathcal{A}}, R_{\mathcal{A}}\downarrow \cup \rho_{\mathcal{A}})$ is an N-automaton.*

We take $\hat{\mathcal{A}} \overset{\text{def}}{=} (S^{\star}_{\mathcal{A}} \cup S_{\mathcal{A}}, R_{\mathcal{A}}\downarrow \cup \rho_{\mathcal{A}})$. By the above lemmata, for every pair of disjoint predicates p, q and for every ground p-vector $\mathbf{s}$, we have $t \in [p,q]^{\emptyset}{}_{\hat{\mathcal{A}}}(\mathbf{s})$ iff $t \in p_{\mathcal{A}}(\mathbf{s}) \cap q_{\mathcal{A}}(\mathbf{s})$.

We need the following:

Lemma 7. *Let $\mathcal{A}$ be an automaton. Let $p \in S_{\mathcal{A}}$ and let $l \in A_{\mathtt{t}}(p)$. For all ground terms $t_1, \ldots, t_n, t$, $Mod(\rho_A) \models p(t_1, \ldots, t_{l-1}, \bot, t_{l+1}, \ldots, t_n, t)$ iff $Mod(\rho_A) \models p(t_1, \ldots, t_n, t)$ and t_l does not occur in t.*

The next lemma handles the more general case of non-disjoint intersection.

Lemma 8. *Let $\mathcal{A}$ be an N-automaton. For any N^+-formula $\phi = p(\mathbf{t}, x) \wedge q(\mathbf{t'}, x)$, one can compute an extension $\mathcal{A}'$ of $\mathcal{A}$ and an N^+-formula $\Lambda(\phi)$ of the form $r(\mathbf{s}, x)$ s.t. all the components of $\mathbf{s}$ are components of $\mathbf{t}$ or $\mathbf{t'}$ and $sol_{\mathcal{A}}(\phi) = sol_{\mathcal{A}'}(\Lambda(\phi))$.*

5 Solving N^+-Formulae

In this section, we show (constructively) that there exists an algorithm checking whether a given (closed) N^+-formula has solutions or not. This entails in particular that emptiness problems or unification problems are decidable since they can be easily encoded into N^+-formulae. According to Lemma 3, any equation $t = s$ between terms in T_I (hence also between terms in T_Σ) can be eliminated and replaced by an equivalent N^+-formula ϕ not containing any such equations (see Remark 1). Moreover, we assume, w.l.o.g., that for all non-arithmetic atoms $p(t_1, \dots, t_n)$ occurring in the formula, $t_1, \dots, t_n$ are either variables, or 0 or $\perp$, where $\perp$ is a special constant symbol not occurring in the considered automaton.

5.1 Emptiness Problems

We first consider a particular case. An N^+-formula ϕ of the form $(\exists x)p(t_1, \dots, t_n, x)$ where x does not occur in $t_1, \dots, t_n$ is called an *emptiness problem.* ϕ is said to be *simple* if $A_=(p) = \emptyset$, and for all $i \in A_c(p)$, $t_i = 0$.

If S is a set of rules, and $p(\mathbf{t})$ an atom (where $\mathbf{t}$ denotes a vector of terms), we denote by $S_{[p(\mathbf{t})]}$ the set of rules $(H \Rightarrow p(\mathbf{s}))\theta$ s.t. $H \Rightarrow p(\mathbf{s}) \in S$ and θ is a most general unifier of $\mathbf{t}$ and $\mathbf{s}$. Note that the heads of the rules in $S_{[p(\mathbf{t})]}$ are instances of $p(\mathbf{t})$.

Let $\phi = (\exists x)p(\mathbf{t}, x)$ be an emptiness problem (e.p. for short) and let $\mathcal{A}$ be an N-automaton. We denote by $D_\mathcal{A}(\phi)$ the set of formulae of the form $(\exists \mathbf{z})[\mathbf{t} = \mathbf{s} \wedge \bigwedge_{i=1}^k (\exists x_i) q_i(\mathbf{v_i}, x_i)]$, where $\bigwedge_{i=1}^k q_i(\mathbf{u_i}, x_i) \Rightarrow p(\mathbf{s}, f(x_1, \dots, x_k))$ is a rule occurring in $\rho_{\mathcal{A}[p(\mathbf{t},x)]}$, $\mathbf{z}$ denotes the variables in $\mathbf{s}$ and $\mathbf{v_i}$ is obtained from $\mathbf{u_i}$ by replacing any occurrence of $f(x_1, \dots, x_k)$ by $\perp$. We denote by $U_\mathcal{A}(\phi)$ the disjunction of all the formulae occurring in $D_\mathcal{A}(\phi)$.

Proposition 1. *For any e.p. ϕ and for every N-automaton $\mathcal{A}$, $\phi \equiv_\mathcal{A} U_\mathcal{A}(\phi)$.*

For any e.p. ϕ, we shall define an equivalent N^+-formula $\Gamma_\mathcal{A}(\phi)$ containing no existential non-arithmetic variable. To this purpose, we need to introduce some additional definitions. If p is a predicate symbol, then we denote by $n(p)$ the (necessarily unique) predicate q s.t. the principal atom of the inductive p-rule is of the form $q(\dots)$ (if there is no inductive p-rule then $n(p)$ is defined arbitrarily), and by $m(p)$ the smallest integer k s.t. there exists l s.t. $n^l(n^k(p)) = n^k(p)$ (k, l exist since the number of predicate symbols is finite).

Let $\mathcal{A}$ be an N-automaton and let $\phi = (\exists x)p(\mathbf{t}, x)$ and $\psi = (\exists y)q(\mathbf{s}, y)$ be two e.p.'s. We write $\phi > \psi$ iff:

- Either *level*$(p) >$ *level*(q), or *level*$(p) =$ *level*(q) and $|var(\mathbf{s})| < |var(\mathbf{t})|$.
- Or *level*$(p) =$ *level*(q), $|var(\mathbf{s})| = |var(\mathbf{t})|$, ϕ is simple and ψ is not.

– Or $level(p) = level(q)$, $|var(\mathbf{s})| = |var(\mathbf{t})|$, neither ϕ nor ψ is simple and $m(p) > m(q)$.

$\Gamma_{\mathcal{A}}(\phi)$ **is constructed by induction on the ordering** $>$**.** If ξ is a complex formula then we shall denote by $\Gamma_{\mathcal{A}}(\xi)$ the formula obtained by replacing each e.p. ψ occurring in ξ by $\Gamma_{\mathcal{A}}(\psi)$. Of course this definition makes sense only if $\Gamma_{\mathcal{A}}(\psi)$ has been defined, i.e. if $\phi > \psi$ for every e.p. ψ occurring in ξ.

Let $\phi = (\exists x)p(\mathbf{t}, x)$ where $\mathbf{t} = (t_1, \ldots, t_n)$. $\Gamma_{\mathcal{A}}(\phi)$ is defined as follows:

1. **If** $A_{=}(p)$ **is non empty**, then $\Gamma_{\mathcal{A}}(\phi) \overset{\text{def}}{=} p(t_1, \ldots, t_n, t_i)$ where i is an arbitrarily chosen index in $A_{=}(p)$.
2. **If there exists** $j \in A_c(p)$ **s.t.** $t_j \neq 0$ **and if either** $\rho_{\mathcal{A}}$ **contains no inductive** p**-rule or** $m(p) > 0$, then $\Gamma_{\mathcal{A}}(\phi) \overset{\text{def}}{=} \Gamma_{\mathcal{A}}(U_{\mathcal{A}}(\phi))$.
3. **If for all** $j \in A_c(p)$**,** $t_j = 0$**:**
   We denote by E the smallest set of conjunctions of e.p. s.t. $\phi \in E$ and if $\phi \wedge \psi \in E$, ϕ is simple, and $(\exists \mathbf{u})[\mathbf{t} = \mathbf{s} \wedge \gamma]$ is in $D_{\mathcal{A}}(\phi)$ then $\gamma \wedge \psi \in E$.
   E must be finite. Indeed, since the head of the rules contains all the variables, all the variables in E must occur in ϕ, hence the number of possible e.p. is finite (up to equivalence). Thus the number of distinct disjunctions is finite. Let ξ be the disjunction of conjunctions $\psi \in E$ that contain no simple e.p. We define $\Gamma_{\mathcal{A}}(\phi) \overset{\text{def}}{=} \Gamma_{\mathcal{A}}(\xi)$.
4. **If** $\rho_{\mathcal{A}}$ **contains an inductive** p**-rule and** $m(p) = 0$**:**
   Let $\{i_1, \ldots, i_m\}$ be the elements in $A_c(p)$ s.t. t_{i_j} is a variable. Starting from the formula ϕ we repeatedly replace any e.p. of the form $(\exists x)n^i(p)(\mathbf{v}_i, x)$ (initially we have $i = 0$) by $U_{\mathcal{A}}((\exists x)n^i(p)(\mathbf{v}_i, x_i))$, until we obtain another e.p. of head p (which is possible since $m(p) = 0$). The obtained formula can be reduced (by miniscoping and distributivity) to a formula of the form $\psi \vee [(\exists \mathbf{z}) \bigwedge_{j=1}^{m} t_{i_j} = succ^{k_j}(z_j) \wedge \gamma \wedge (\exists x)p(\mathbf{v_k}, x)]$, where $k_1, \ldots, k_m \in \mathbb{N}$, $z_1, \ldots, z_m$ are either 0 or variables not occurring in γ, $\mathbf{z}$ denotes the vector of variables in $z_1, \ldots, z_m$ and $\mathbf{v_k}$ is obtained from $\mathbf{t}$ by replacing each component t_{i_j} by z_j.
   We define: $\Gamma_{\mathcal{A}}(\phi) \overset{\text{def}}{=} (\exists l, \mathbf{z})[\bigwedge_{j=1}^{m}(t_{i_j} = l \times k_i + z_j) \wedge \Gamma_{\mathcal{A}}(\gamma) \wedge \Gamma_{\mathcal{A}}(\psi')]$, where ψ' is obtained from ψ by replacing t_{i_j} by z_j ($l \times k_i$ denotes the term $l + l + \ldots + l$ (k_i times)).

Lemma 9. *$\Gamma_{\mathcal{A}}(\phi)$ is well defined, for every emptiness problem ϕ and for every automaton $\mathcal{A}$. Moreover, $\phi \equiv_{\mathcal{A}} \Gamma_{\mathcal{A}}(\phi)$ and the quantified variables in $\Gamma_{\mathcal{A}}(\phi)$ are arithmetic.*

5.2 Reduction to Presburger Arithmetic

By distributivity and miniscoping, any N^+-formula ϕ can be reduced into a formula of the form $\bigvee_{i=1}^{n}(\exists \mathbf{x_i})\psi_i$ where $\psi_i = \bigwedge_{j=1}^{k_i} \gamma_{ij}$ and where the γ_{ij}'s are atoms. The algorithm is defined by the following rules, applied in the specified order, on the disjuncts ψ_i (and not on the formulae occurring in them). The formulae are normalized after each rule application.

(r_0) $(\exists \mathbf{x})$ $[\psi \wedge p(\mathbf{t}, \bot)]$ $\rightarrow$ *false*

(r_1) $(\exists \mathbf{x})$ $[\psi \wedge p(\mathbf{t}, x) \wedge q(\mathbf{t'}, x)]$ $\rightarrow (\exists \mathbf{x})[\psi \wedge \Lambda(p(\mathbf{t}, x) \wedge q(\mathbf{t'}, x))]$

(r_2) $(\exists x_1, \ldots, x_k)$ $(\exists \mathbf{n})$ ϕ $\rightarrow \bigvee_{i=1}^{k} (\exists x_1, \ldots, x_k)\ \exists \mathbf{n}\ rm_{x_i}(\phi)$
If for all $i \in [1..k]$ there exists an atom $p(\mathbf{t}, x_j)$ in ϕ s.t. $j \neq i$
and x_i occurs in $\mathbf{t}$, $x_1, \ldots, x_k \in \mathcal{X}$ and $\mathbf{n}$ is a vector of variables in $\mathcal{X}_N$.
$rm_x(\phi)$ is defined below.

(r_3) $(\exists x_1, \ldots, x_k)$ $(\psi \wedge p(\mathbf{t}, x_k)) \rightarrow (\exists x_1, \ldots, x_{k-1})$ $[\psi \wedge p(\mathbf{t}, x_k)\{x_k \rightarrow t_i\}]$
If x_k does not occur in ψ and i is an index in $A_=(p)$ s.t. $t_i \neq x_k$.

(r_4) $(\exists x_1, \ldots, x_k)$ $[\psi \wedge p(\mathbf{t}, x_k)]$ $\rightarrow (\exists x_1, \ldots, x_{k-1})$ $[\psi \wedge U_{\mathcal{A}}((\exists x_k)\ p(\mathbf{t}, x_k))]$
If x_k occurs in $\mathbf{t}$ but not in ψ.

(r_5) $(\exists x_1, \ldots, x_k)$ $[\psi \wedge p(\mathbf{t}, x_k)]$ $\rightarrow (\exists x_1, \ldots, x_{k-1})$ $[\psi \wedge \Gamma_{\mathcal{A}}((\exists x_k) p(\mathbf{t}, x_k))]$
If $A_=(p) = \emptyset$, x_k does not occur in $\mathbf{t}$ nor in ψ.

$rm_x(\phi)$ is defined by the following (auxiliary) rule:

(rm_x) $(\exists x_1, \ldots, x_n, y)$ $(p(\mathbf{t}, y) \wedge \psi) \rightarrow (\exists x_1, \ldots, x_n)$ $(p(\mathbf{t}, y) \wedge \psi)\{y \rightarrow x\}$
$\vee (\exists x_1, \ldots, x_n, y)$ $(p(\mathbf{t}\{x \rightarrow \bot\}, y) \wedge \psi)$
If x occurs in $\mathbf{t}$ and $y \neq x$.

Lemma 10. *Let ϕ be a N^+-formula. The rules $r_0, \ldots, r_5$ (with the above strategy) terminate on ϕ and preserve equivalence. Moreover, any irreducible formula is purely arithmetic.*

Since Presburger arithmetic is known to be decidable, Lemma 10 provides an algorithm for checking whether a given closed N^+-formula is satisfiable or not.

6 A Simple Application

We show a simple example of application in the context of Logic Programming. If the satisfiability problem is decidable for N^+-formulae (as shown by Lemma 10), then N-automata can be integrated in Logic Programs. The corresponding unification problems can be solved by our constraint solving algorithm.

Assume that one wants to define a predicate $\mathit{last}(l, x)$ which is true iff x is the last element of the list l. Using standard Horn clauses, $\mathit{last}(l, x)$ is defined as follows: $\{\mathit{last}(\mathit{cons}(x, \mathit{nil}), x), \mathit{last}(\mathit{cons}(y, l), x) \Leftarrow \mathit{last}(l, x)\}$.

Using N-automata we obtain: $\mathit{last}(l, x) \Leftarrow p(x, l)$, where p is defined by the following N-rules:

$$\{q(x, y) \Rightarrow p(x, \mathit{cons}(y, \mathit{nil})), p(x, l) \Rightarrow p(x, \mathit{cons}(y, l)), q(x, x)\}.$$

Both techniques yield exactly the same result (from a semantic point of view). But the use of N-automata allows one to compute the solution of a request $\mathit{last}(l, x)$ in a symbolic way, rather than enumerating all possible lists.

Thus a request of the form $\mathit{last}(l, 1) \wedge \mathit{last}(l, 2)$ *diverges* with the first approach and simply *terminates* and *fails* using our technique. Of course, the programmer does not need to write the N-automaton explicitly: it could be compiled automatically from the set of Horn clauses (in case they are of the required form).

7 Conclusion

We have presented a framework unifying tree automata [4] with some term schematisation languages [3]. By combining the features of both approaches, we obtained a formalism which is strictly more expressive than the original ones. We provided an algorithm to check the satisfiability of positive, purely existential membership formulae, which allows one to include N-automaton into most existing symbolic computation procedures (such as SLD-resolution in Logic Programming). Our work extends the power of tree automata by showing how to include integer counters and parameters. It also strictly enhances the expressive power of term schematisations by using more general contexts.

Future works include the extension of the presented approach in order to capture more expressive term schematisation languages such as R-terms or primal grammars, and the extension of the class of considered formulae in order to handle formulae with negations and universal quantifiers. In the context of tree automata this corresponds to the complement problem (i.e. compute the complement of the set of terms recognized by a tree automaton) and in the context of term schematisations, this corresponds to disunification problems [6].

References

1. J. Chabin, J. Chen, and P. Réty. Synchronized-context free tree-tuple languages. Technical Report RR-2006-13, LIFO, 4 rue Léonard de Vinci, BP 6759, F-45067 Orléans Cedex 2 FRANCE, 2006.
2. H. Chen and J. Hsiang. Logic programming with recurrence domains. In *Automata, Languages and Programming (ICALP'91)*, pages 20–34. Springer, LNCS 510, 1991.
3. H. Comon. On unification of terms with integer exponents. *Mathematical System Theory*, 28:67–88, 1995.
4. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 1997.
5. H. Comon and C. Delor. Equational formulae with membership constraints. *Information and Computation*, 112(2):167–216, August 1994.
6. H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–475, 1989.

7. G. Feuillade, T. Genet, and V. Viet Triem Tong. Reachability Analysis over Term Rewriting Systems. *J. Automated Reasoning*, 33 (3-4):341–383, 2004.
8. M. Hermann and R. Galbavý. Unification of Infinite Sets of Terms schematized by Primal Grammars. *Theoretical Computer Science*, 176(1–2):111–158, 1997.
9. S. Limet and G. Salzer. Manipulating tree tuple languages by transforming logic programs. *Electr. Notes Theor. Comput. Sci.*, 86(1), 2003.
10. S. Limet and G. Salzer. Proving properties of term rewrite systems via logic programs. In *RTA*, volume 3091 of *Lecture Notes in Computer Science*, pages 170–184. Springer, 2004.
11. C. Löding. Model-checking infinite systems generated by ground tree rewriting. In M. Nielsen and U. Engberg, editors, *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002.*, volume 2303 of *Lecture Notes in Computer Science*, pages 280–294. Springer, 2002.
12. D. Lugiez and J. L. Moysset. Tree automata help one to solve equational formulae in AC-theories. *Journal of Symbolic Computation*, 18(4):297–318, 1994.
13. P. Réty and J. Vuotto. Tree automata for rewrite strategies. *J. Symb. Comput.*, 40(1):749–794, 2005.
14. G. Salzer. The unification of infinite sets of terms and its applications. In *Logic Programming and Automated Reasoning (LPAR'92)*, pages 409–429. Springer, LNAI 624, July 1992.
15. K. N. Venkataraman. Decidability of the purely existential fragment of the theory of term algebras. *J. ACM*, 34(2):492–510, 1987.