Monotone AC-Tree Automata

Hitoshi Ohsaki[1], Jean-Marc Talbot[2], Sophie Tison[2] and Yves Roos[2]

[1]National Institute of Advanced Industrial Science and Technology
and
PRESTO, Japan Science and Technology Agency
ohsaki@ni.aist.go.jp

[2]Laboratoire d'Informatique Fondamentale de Lille
Université des Sciences et Technologies de Lille, France
{talbot,tison,yroos}@lifl.fr

Abstract. We consider several questions about *monotone AC-tree automata*, a class of equational tree automata [21] whose transition rules correspond to rules in Kuroda normal form of context-sensitive grammars. Whereas it is known that this class has a decision procedure to determine if a given monotone AC-tree automaton accepts no term [23], other decidability and complexity results have not been well-investigated yet. In the paper, we prove that the membership problem for monotone AC-tree automata is *PSPACE-complete*. We then study the expressiveness of monotone AC-tree automata: precisely, we prove that the family of AC-*regular* tree languages is strictly subsumed in that of AC-monotone tree languages. This result immediately yields the answers to two open problems, specially that the family of monotone AC-tree languages is *not* closed under complementation, and that the inclusion problem for monotone AC-tree automata is undecidable.
Keywords: equational tree languages, complementation, decidability.

1 Introduction

Tree automata [5] have been applied successfully in many area of computer science, such as protocol verification [1, 12], type inference [7, 11], checking the sufficient completeness of algebraic specifications [3], and checking the consistency of semi-structured documents [17]. This widespread use is due to good closure properties of tree automata, such as the (effective) closedness under Boolean operations and rewrite descendant computation, as well as efficient decision procedures. However, the standard framework of tree automata is not powerful when some algebraic laws such as associativity and commutativity have to be taken into account. In particular, it is known that the regularity of tree languages is not preserved for the congruence closure with respect to an equational theory. To overcome this problem, Ohsaki [21] in 2001 and Goubault-Larrecq and Verma [14] in 2002 independently proposed extensions of tree automata. Their ideas in new frameworks are to combine tree automata with equational theories, and each of their studies considers by coincidence the case in particular where some of the function symbols have associative (A), commutative (C), and/or some other equational properties like the identity (I) and nilpotent (U) axioms. The notion of accepted

languages may differ for these two approaches, however, they coincide in the regular case for any combination of the axioms A, C, I and U.

The AC case is of particular interest since this kind of automata which are able to deal with AC symbols are closely related to tree automata with arithmetical constraints, such as multitree automata [20] and Presburger tree automata [27]. Further discussion on this relationship can be found in our recent paper [2]. It has been shown that for AC-tree automata good properties of "classical" tree automata remain: the membership and emptiness are decidable and the closure of automata by Boolean operations can be computed [21, 28, 29].

Motivated by cryptographic protocol verification, Goubault-Larrecq and Verma proposed to extend AC-tree automata by considering two-way and/or alternating computations [14]. They proved on one hand that two-way AC-tree automata are not more powerful than (one-way) AC-tree automata. On the other hand, the alternation strictly increases the expressiveness of AC-tree automata while the emptiness problem is undecidable.

Inspired by multiset grammars (alternatively, called commutative grammars) [13, 18], Ohsaki proposed another extension of AC-tree automata [21], called monotone AC-tree automata; he proved that both emptiness and membership remains decidable for monotone AC-tree automata and that the languages defined by these automata are closed under union and intersection. Furthermore, Ohsaki and Takai develop the automated system, called ACTAS, manipulating AC-tree automata computation by using the exact and approximation algorithms [24].

In this paper, we further investigate monotone AC-tree automata. First, we prove that the membership problem of deciding, "given a term t and an automaton $\mathcal{A}/\mathsf{AC}$, whether t belongs to the language defined by $\mathcal{A}/\mathsf{AC}$" is PSPACE-complete: we give a nondeterministic algorithm running in polynomial space with respect to the size of the input tree and automaton. For the lower bound, we reduce the validity problem of quantified Boolean formulas to the membership problem. Then we show that the class of monotone AC-tree automata is strictly wider than the class of regular AC-tree automata by exhibiting a tree language accepted by a monotone AC-tree automaton but that cannot be defined by any regular AC-tree automaton. Following the same ideas, we prove that the family of AC-monotone tree languages is *not* closed under complement while this class is closed under union and intersection. Finally, using similar techniques, we show that the inclusion problem for monotone AC-tree automata is not decidable.

The paper is organized as follows. Definitions and terminologies concerning equational tree language theory are introduced in Section 2. The closure properties and the decidability of equational tree automata are also summarized. In Section 3, we discuss the complexity of the membership problem for monotone AC-tree automata, proving that the problem is PSPACE-complete. Section 4 is devoted to the study of the relative expressiveness of AC-tree automata. Using the proof technique introduced in the previous section, we show in Section 5 that AC-monotone tree languages are not closed under complementation. Section 6 contains the proof for the undecidability of the inclusion problem. Finally, we conclude by summarizing the results obtained in the paper that give us the solutions to open questions in [6].

2 Preliminaries

A *signature* is a finite set $\mathcal{F}$ of function symbols together with natural numbers n. A natural number n associated with f, denoted by $\mathsf{arity}(f) = n$, is the *arity* of f. Function symbols of arity 0 are called *constants*. We assume the existence of countably infinite set $\mathcal{V}$ of variables. The set $\mathcal{T}(\mathcal{F}, \mathcal{V})$ of terms over $\mathcal{F}$ with $\mathcal{V}$ is inductively defined as follows: $\mathcal{V} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V})$; $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ if $\mathsf{arity}(f) = n$ and $t_i \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ for all $1 \leqslant i \leqslant n$. Elements in the set $\mathcal{T}(\mathcal{F}, \varnothing)$ are called *ground* terms. In the paper, we write $\mathcal{T}(\mathcal{F})$ for $\mathcal{T}(\mathcal{F}, \varnothing)$.
Let $\Box$ be a fresh constant, named a *hole*. Elements in the set $\mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{V})$ of terms, denoted by $\mathcal{C}(\mathcal{F}, \mathcal{V})$, are *contexts*. The *empty* context is the hole $\Box$. If C is a context with n holes and $t_1, \ldots, t_n$ are terms, then $C[t_1, \ldots, t_n]$ represents the term obtained from C by replacing the holes from left to right by $t_1, \ldots, t_n$ in $\mathcal{F}(\mathcal{F}, \mathcal{V})$. Terms $t_1, \ldots, t_n$ are *subterms* of $C[t_1, \ldots, t_n]$.
A *tree automaton* (TA for short) $\mathcal{A}$ is a 4-tuple $(\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$, whose components are the signature $\mathcal{F}$, a finite set $\mathcal{Q}$ of states such that $\mathcal{F} \cap \mathcal{Q} = \varnothing$, a subset $\mathcal{Q}_{fin}$ of $\mathcal{Q}$ consisting of the so-called *final states*, and a finite set Δ of transition rules in one of the following forms:

$$\text{(TYPE 1)} \quad f(p_1, \ldots, p_n) \to q \qquad \text{(TYPE 2)} \quad f(p_1, \ldots, p_n) \to f(q_1, \ldots, q_n)$$

for some $f \in \mathcal{F}$ with $\mathsf{arity}(f) = n$ and $p_1, \ldots, p_n, q, q_1, \ldots, q_n \in \mathcal{Q}$.
An equational system (ES for short) $\mathcal{E}$ is a set of equations $s = t$, where s, t are terms over the signature $\mathcal{F}$ with the set $\mathcal{V}$ of variables. For two terms s, t, we write $s =_{\mathcal{E}} t$ whenever s, t are equivalent modulo the equational system $\mathcal{E}$, i.e. s, t are the elements in the same equivalence class of the quotient term model $\mathcal{T}(\mathcal{F}, \mathcal{V})/=_{\mathcal{E}}$. The *associativity* and *commutativity* axioms for a binary function symbol f in $\mathcal{F}$ are the equations

$$f(f(x, y), z) = f(x, f(y, z)) \qquad\qquad f(x, y) = f(y, x),$$

respectively, where x, y, z are variables in $\mathcal{V}$. In the paper, we write $\mathcal{F}_{\mathrm{A}}$ for the set of binary function symbols with associativity laws *only*, and $\mathcal{F}_{\mathrm{AC}}$ for the set of binary symbols equipped with *both* associativity and commutativity. The ES A consists of the associativity axioms for each $f \in \mathcal{F}_{\mathrm{A}}$, and AC is the ES consisting of the associativity and commutativity axioms for each $f \in \mathcal{F}_{\mathrm{AC}}$.
An *equational tree automaton* (ETA for short) $\mathcal{A}/\mathcal{E}$ is a pair of a TA $\mathcal{A}$ and an ES $\mathcal{E}$ over the same signature $\mathcal{F}$. An ETA $\mathcal{A}/\mathcal{E}$ is called

- *regular* if it has only rules of TYPE 1,
- *monotone* if it is allowed to have rules of TYPE 1 and TYPE 2.

We say $\mathcal{A}/\mathcal{E}$ is a AC-TA (A-TA) if $\mathcal{E} = \mathsf{AC}$ (resp. $\mathcal{E} = \mathsf{A}$). Besides, in the following discussion, we suppose $\mathcal{F}_{\mathrm{A}} = \varnothing$ when considering $\mathcal{A}/\mathsf{AC}$; likewise, $\mathcal{F}_{\mathrm{AC}} = \varnothing$ for $\mathcal{A}/\mathsf{A}$. The readers are recommended to consult [21] for a more detailed presentation.
We write $s \to_{\mathcal{A}/\mathcal{E}} t$ if there exist s', t' such that $s =_{\mathcal{E}} s'$, $s' = C[l]$, $t =_{\mathcal{E}} t'$ and $t' = C[r]$ for some transition rule $l \to r \in \Delta$ and context $C \in \mathcal{C}(\mathcal{F} \cup \mathcal{Q})$. This relation $\to_{\mathcal{A}/\mathcal{E}}$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is called a *move relation* of $\mathcal{A}/\mathcal{E}$. The transitive closure and reflexive-transitive closure of $\to_{\mathcal{A}/\mathcal{E}}$ are denoted by $\to^+_{\mathcal{A}/\mathcal{E}}$ and $\to^*_{\mathcal{A}/\mathcal{E}}$, respectively. For an ETA $\mathcal{A}/\mathcal{E}$ with $\mathcal{E} = \varnothing$, we simply write $\to_{\mathcal{A}}$, $\to^+_{\mathcal{A}}$ and $\to^*_{\mathcal{A}}$, instead.

	regular AC-TA	monotone A-TA	monotone AC-TA
closure under union, intersection	Yes [4]	Yes	Yes
closure under complement	Yes [4]	Yes	?
decidability of emptiness	Linear	No	Yes
decidability of membership	NP-complete	PSPACE-complete	?
decidability of inclusion	Yes	No	?

Fig. 1. Some closure properties and decidability results

A term t is *accepted* by $\mathcal{A}/\mathcal{E}$ if $t \in \mathcal{T}(\mathcal{F})$ and $t \rightarrow^*_{\mathcal{A}/\mathcal{E}} q$ for some $q \in \mathcal{Q}_{fin}$. Elements in the set $\mathcal{L}(\mathcal{A}/\mathcal{E})$ are ground terms accepted by $\mathcal{A}/\mathcal{E}$. A *tree language* L over $\mathcal{F}$ is a subset of $\mathcal{T}(\mathcal{F})$. A tree language L is *$\mathcal{E}$-regular* (*$\mathcal{E}$-monotone*) if there exists some regular (resp. monotone) $\mathcal{E}$-tree automaton $\mathcal{A}/\mathcal{E}$ such that $L = \mathcal{L}(\mathcal{A}/\mathcal{E})$. If L is $\mathcal{E}$-regular with $\mathcal{E} = \varnothing$, we say L is regular. Likewise, we say L is monotone if L is $\varnothing$-monotone.

Let op be an n-ary mapping from $\wp(\mathcal{T}(\mathcal{F}))^n \mapsto \wp(\mathcal{T}(\mathcal{F}))$. The family of $\mathcal{E}$-regular (resp. $\mathcal{E}$-monotone) languages is *closed under* op if whenever $L_1, \ldots, L_n$ are $\mathcal{E}$-regular (resp. $\mathcal{E}$-monotone) languages then so is $\mathsf{op}(L_1, \ldots, L_n)$. We say that the family of $\mathcal{E}$-regular (resp. $\mathcal{E}$-monotone) languages is *effectively closed under* op if there exists an algorithm which, given regular (resp. monotone) ETA $\mathcal{A}_1/\mathcal{E}, \ldots, \mathcal{A}_n/\mathcal{E}$, computes a regular (resp. monotone) ETA $\mathcal{A}/\mathcal{E}$ such that $\mathcal{L}(\mathcal{A}/\mathcal{E}) = \mathsf{op}(\mathcal{L}(\mathcal{A}_1/\mathcal{E}), \ldots, \mathcal{L}(\mathcal{A}_n/\mathcal{E}))$. One should note that non-regular and equational tree automata defined in [21] are in the above monotone case. It is folklore that whenever $\mathcal{E} = \varnothing$ then $\varnothing$-regular and $\varnothing$-monotone languages coincide. Things are different when some equational theory is taken into account. For instance, it has been shown in [22] that monotone A-TA are strictly more expressive than regular A-TA. But the question remained open in the case of AC.

We sum up in the table of Fig. 1 some known results concerning respectively regular AC-TA, monotone A-TA and monotone AC-TA. The positive results are marked with "Yes", and the negative cases are marked with "No". In case the results are proved in our previous work, the references are omitted. The complexity of the emptiness for regular AC-TA is a direct consequence of Lemma 2 in [21] and the result of regular TA [5]. Question marks "?" in the three columns denote open problems registered in [6].

For most of the results described in the present paper, we will consider a rather simple signature consisting in a finite number of constant symbols and a single AC symbol f. In this case, the *regular* transition rules $\mathsf{f}(p_1, p_2) \rightarrow q$ and $\mathsf{a} \rightarrow q$ correspond to the production rules $q \rightarrow p_1\, p_2$ and $q \rightarrow \mathsf{a}$ of context-free grammars in Chomsky normal form. In case of monotone TA, the additional form $\mathsf{f}(p_1, p_2) \rightarrow \mathsf{f}(q_1, q_2)$ together with the previous two forms corresponds to context-sensitive grammar in Kuroda normal form [19]. Following the same approach for monotone AC-TA, the transition rules

correspond to the production rules of some commutative context-sensitive grammar. The commutative context-sensitive grammars are known to be close to Petri nets [10]. Therefore, most of our developments are related to Petri nets. For this reason, on the other hand, the complexity of the emptiness problem for monotone AC-TA is unclear, that may correspond to the reachability problem for Petri nets [9].

3 The Complexity of the Membership Problem

In this section, we investigate the complexity of the membership problem for monotone AC-tree automata. To show in particular the PSPACE-hardness, we use a proof technique proposed by Esparza [8] where he shows that the reachability problem for one-safe Petri nets is PSPACE-hard.

Theorem 1. *Given a monotone AC-tree automaton $\mathcal{A}/\mathsf{AC}$ and a term t, the problem whether $t \in \mathcal{L}(\mathcal{A}/\mathsf{AC})$ is PSPACE-complete.* □

To show that the membership problem for monotone AC-TA is in PSPACE, it suffices to prove that the size of any ground term t reachable from an initial term t_0 by the move relation of $\mathcal{A}/\mathsf{AC}$ is polynomial relative to the size of t_0 and $\mathcal{A}/\mathsf{AC}$. This allows us to prove that the existence of a successful run for t_0 implies that there exists a "short" successful run at most exponential with respect to the size of t_0 and $\mathcal{A}/\mathsf{AC}$. We use this property to devise a non-deterministic polynomial space algorithm for the membership problem using that the execution of the move relation can be done in polynomial time. Then we appeal to Savitch's theorem [26] stating that NPSPACE = PSPACE to conclude.
Let us define the special notation of terms. We assume that a term t in this section is represented by the following grammar:

$$t \ ::= \ f\langle t_1, \ldots, t_n \rangle \ \mid \ a$$

where f is a function symbol in $\mathcal{F}$ with $\mathsf{arity}(f) > 0$, and a is a constant. Moreover, $\langle t_1, \ldots, t_n \rangle$ is a non-empty sequence of terms $t_1, \ldots, t_n$ such that:

1. if f is a non-AC symbol, then n is the arity of f,
2. if f is an AC symbol, then $n \geqslant 2$ and the root symbol of t_i is not f.

Given a subterm position and a rule to be applied at the subterm position, the corresponding transition step by $\mathcal{A}/\mathsf{AC}$ can be performed on the above term representation in linear time with respect to the size of a term. In the transition steps, there are two non-standard cases, that are done by the transitions rules of the form $f(p_1, p_2) \to q$ and $f(p_1, p_2) \to f(q_1, q_2)$ with f an AC symbol. In both of the two cases, instead of the standard pattern matching, we find p_1, p_2 among subterms $t_1, \ldots, t_n$ of $f\langle t_1, \ldots, t_n \rangle$.
By definition of monotone AC-TA, if a term s is reachable from t by $\to_{\mathcal{A}/\mathsf{AC}}$, the size $|s|$ is less than or equal to $|t| + |\mathcal{A}/\mathsf{AC}|$, where $|\mathcal{A}/\mathsf{AC}|$ is the number of state symbols of $\mathcal{A}/\mathsf{AC}$. Then we can show that for any tree t admitting a successful run $r\colon t \to^*_{\mathcal{A}/\mathsf{AC}} q$ with q a final state of $\mathcal{A}/\mathsf{AC}$, there exists a successful run $r'\colon t \to^*_{\mathcal{A}/\mathsf{AC}} q$ reaching the same state q of the length at most $2^{|t|+|\mathcal{A}/\mathsf{AC}|}$. In fact, we suppose to the contradiction that $t \to^*_{\mathcal{A}/\mathsf{AC}} q$ is the shortest successful run whose length is strictly greater than

$2^{|t|+|\mathcal{A}/\mathsf{AC}|}$. Then terms reachable from t by $\to_{\mathcal{A}/\mathsf{AC}}$ can be described using a space relative to the size at most $|t| + |\mathcal{A}/\mathsf{AC}|$. This implies that the previous shortest run $t \to^*_{\mathcal{A}/\mathsf{AC}} q$ can be represented as $t \to^*_{\mathcal{A}/\mathsf{AC}} u \to^+_{\mathcal{A}/\mathsf{AC}} u \to^*_{\mathcal{A}/\mathsf{AC}} q$. By shrinking this run by chopping off the loop of u, one can obtain a successful run strictly shorter than the original, leading to the contradiction to the minimality assumption.

Based on the above observation, let us define (non-deterministic) algorithm to solve the question if $t \in \mathcal{L}(\mathcal{A}/\mathsf{AC})$. We write in the algorithm $\mathsf{apply}(u, u', r)$ for denoting to "apply the transition rule r at the position of a subterm u' of u." This algorithm needs for the computation a polynomially bounded space with respect to the size $|t| + |\mathcal{A}/\mathsf{AC}|$: Let t be a term over the signature $\mathcal{F}$ and $\mathcal{A}/\mathsf{AC}$ a monotone AC-TA with $\mathcal{A} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$.

```
membership(t, A/AC) {
  c := 1 ; u := t ;
  while (c ⩽ 2^{|t|+|A/AC|}) {
    if (u ∈ Q_fin) then {
      return true }
    else {
      guess r: transition rule in Δ, u': subterm of u to which r is applied at the root ;
      nu := apply(u, u', r) ;
      u := nu }
    c := c + 1
  }
  return false }
```

Let us estimate the space complexity of this algorithm. One can see that apply runs in polynomial time, and thus, in polynomial space. For membership we observe that this procedure requires the space for the counter c and the terms u, u' and nu. Obviously this space can be bounded linearly in $|t| + |\mathcal{A}/\mathsf{AC}|$. So, membership can be executed by a non-deterministic machine using polynomial space.

Next, to show that the membership problem is PSPACE-hard, we consider the validity problem for closed *quantified Boolean formulas* (QBF). This problem is known to be PSPACE-complete. Every formula φ can be represented by the following grammar:

$$\varphi \ ::= \ x \ \mid \ \neg\varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \exists x.\, \varphi \qquad (x\colon \text{ a proposition variable})$$

This assumption is justified by the fact that any quantified Boolean formula can be translated into a formula of the above form in linear time. We assume also that each variable x in the formula occurs in the scope of some quantifier $\exists x$ or $\forall x$ and that each variable is bounded exactly once in the formula.

We suppose that $x_1, \ldots, x_k$ are variables bounded in φ. We show in the following that we can build from a closed formula φ a monotone AC-tree automaton $\mathcal{A}_\varphi/\mathsf{AC}$ and a term t_φ in polynomial time relative to the size of φ such that t_φ is accepted by $\mathcal{A}_\varphi/\mathsf{AC}$ if and only if φ is valid. For this construction, we take the signature $\{\oplus, \mathsf{i}, \mathsf{v}, \mathsf{e}\}$, where $\oplus$ is an AC symbol and $\mathsf{i}, \mathsf{v}, \mathsf{e}$ are constants. We denote by t_φ a term consisting of exactly k constants of v, a constant of i, and a constant of e. For each subformula ψ of φ, we define the state symbols $q_{(\psi,?)}$, $q_{(\psi,T)}$, and $q_{(\psi,F)}$. In case of $\psi \ \equiv \ \exists x.\, x \wedge x$,

the two subformulas x's are distinguished in this construction. For each variable x_i $(1 \leqslant i \leqslant k)$, we take the two states q_{true/x_i} and q_{false/x_i}. The state q_{fin} is the final state. Let us describe the intended meaning of each state symbol. The truth value of the formula φ is computed recursively in our encoding. Along this idea, the state $q_{(\psi,?)}$ means that the subformula ψ can be taken into consideration. When the computation for ψ is performed, the state $q_{(\psi,?)}$ is "transformed" to either $q_{(\psi,T)}$ or $q_{(\psi,F)}$, depending on the truth value of ψ. The state $q_{(\psi,T)}$ means that ψ is true, and $q_{(\psi,F)}$ means that ψ is false. The two states q_{true/x_i} and q_{false/x_i} are the environment to store the information for the valuation to x_i.
Using the above state symbols, next we define the transition rules. For the constants $\mathsf{i}, \mathsf{v}, \mathsf{e}$, we take the following transition rules:

$$\mathsf{i} \to q_{(\varphi,?)} \qquad \mathsf{v} \to q_{\mathsf{v}} \qquad \mathsf{e} \to q_{\mathsf{e}}$$

The first rule is used to initiate the computation. We define the transition rules for instantiating a variable x_i $(1 \leq i \leq k)$ to true or false:

$$q_{(x_i,?)} \oplus q_{\mathsf{true}/x_i} \to q_{(x_i,T)} \oplus q_{\mathsf{true}/x_i} \qquad q_{(x_i,?)} \oplus q_{\mathsf{false}/x_i} \to q_{(x_i,F)} \oplus q_{\mathsf{false}/x_i}$$

The rules for negation are defined as follows: for a subformula $\neg\psi$ of the formula φ,

$$q_{(\neg\psi,?)} \oplus q_{\mathsf{e}} \to q_{(\psi,?)} \oplus q_{\mathsf{e}}$$

and

$$q_{(\psi,T)} \oplus q_{\mathsf{e}} \to q_{(\neg\psi,F)} \oplus q_{\mathsf{e}} \qquad q_{(\psi,F)} \oplus q_{\mathsf{e}} \to q_{(\neg\psi,T)} \oplus q_{\mathsf{e}}.$$

The first rule decomposes $\neg\,\psi$ and the last two rules re-construct $\neg\psi$ with the truth value by using ψ with the truth value. Similarly, the rules for the conjunction can be defined. For any subformula $\psi \wedge \psi'$ of the formula φ,

$$q_{(\psi\wedge\psi',?)} \oplus q_{\mathsf{e}} \to q_{(\psi,?)} \oplus q_{\mathsf{e}}$$

$$\begin{array}{llll} q_{(\psi,F)} \oplus q_{\mathsf{e}} & \to q_{(\psi\wedge\psi',F)} \oplus q_{\mathsf{e}} & q_{(\psi,T)} \oplus q_{\mathsf{e}} & \to q_{(\psi',?)} \oplus q_{\mathsf{e}} \\ q_{(\psi',F)} \oplus q_{\mathsf{e}} & \to q_{(\psi\wedge\psi',F)} \oplus q_{\mathsf{e}} & q_{(\psi',T)} \oplus q_{\mathsf{e}} & \to q_{(\psi\wedge\psi',T)} \oplus q_{\mathsf{e}}. \end{array}$$

In the above definition, $\psi \wedge \psi'$ is evaluated in a sequential manner: first we consider the subformula ψ and evaluate it, and then we take the remaining subformula ψ'. For the existential quantification $\exists x_i.\psi$, we need to consider both valuations for the bound variable x_i and the computation for ψ:

$$q_{(\exists x_i.\psi,?)} \oplus q_{\mathsf{v}} \to q_{\mathsf{true}/x_i} \oplus q_{(\psi,?)}$$

$$\begin{array}{llll} q_{\mathsf{true}/x_i} \oplus q_{(\psi,T)} & \to q_{(\exists x_i.\psi,T)} \oplus q_{\mathsf{true}/x_i} & q_{\mathsf{true}/x_i} \oplus q_{(\psi,F)} & \to q_{\mathsf{false}/x_i} \oplus q_{(\psi,?)} \\ q_{\mathsf{false}/x_i} \oplus q_{(\psi,T)} & \to q_{(\exists x_i.\psi,T)} \oplus q_{\mathsf{false}/x_i} & q_{\mathsf{false}/x_i} \oplus q_{(\psi,F)} & \to q_{(\exists x_i.\psi,F)} \oplus q_{\mathsf{false}/x_i} \end{array}$$

In the above definition, we start with the valuation associating the Boolean value true with x_i. If ψ turns out to be true under this valuation, $\exists x.\psi$ is also true; otherwise, the valuation associating the Boolean value false with x_i is tried. The following rules are used to finalize the computation:

$$\oplus(q_{(\varphi,T)}, q_{\mathsf{e}}) \to q_{fin} \qquad \oplus\,(q_{\mathsf{true}/x_i}, q_f) \to q_{fin} \qquad \oplus\,(q_{\mathsf{false}/x_i}, q_f) \to q_{fin}$$

We can show that the previous encoding is correct, by using the induction over the structure of the formula ψ. The remainder of the proof is obtained from the following observation: Let $t_{(\psi,?)}$ be a term that contains exactly a $q_{(\psi,?)}$, a q_{e}, and n_v occurrences of c_v (c_v being either the constant v or the state q_{v}, and n_v being the number of variables that do not freely occur in ψ) and for each free variable $x_{i_1}, \ldots, x_{i_\ell}$, either $q_{\mathsf{true}/x_{i_j}}$ or $q_{\mathsf{false}/x_{i_j}}$. Suppose δ is the Boolean valuation defined for $x_{i_1}, \ldots, x_{i_\ell}$ such that δ associates with x_{i_j} the value true if $q_{\mathsf{true}/x_{i_j}}$ appears in $t_{(\psi,?)}$, and false otherwise. Then we have:

- $t_{(\psi,?)} \rightarrow^*_{\mathcal{A}/\varphi/\mathsf{AC}} t_{(\psi,T)}$ if and only if ψ is valid under δ, where $t_{(\psi,T)}$ is the same as $t_{(\psi,?)}$ except
  1. $q_{(\psi,?)}$ in $t_{(\psi,?)}$ is replaced by $q_{(\psi,T)}$,
  2. if $x_{i_{l+1}}, \ldots, x_{i_{l+m}}$ are bound variables in ψ, then m occurrences of v and q_{v} in $t_{(\psi,?)}$ are replaced by $q_{b_1/x_{i_{\ell+1}}}, \ldots, q_{b_m/x_{i_{\ell+m}}}$ with $b_1, \ldots, b_m \in \{\mathsf{true}, \mathsf{false}\}$.
- $t_{(\psi,?)} \rightarrow^*_{\mathcal{A}/\varphi/\mathsf{AC}} t_{(\psi,F)}$ if and only if ψ is not valid under δ, where $t_{(\psi,F)}$ is the same as $t_{(\psi,?)}$ except
  1. $q_{(\psi,?)}$ in $t_{(\psi,?)}$ is replaced by $q_{(\psi,F)}$,
  2. if $x_{i_{l+1}}, \ldots, x_{i_{l+m}}$ are bound variables in ψ, then m occurrences of v and q_{v} in $t_{(\psi,?)}$ are replaced by $q_{b_1/x_{i_{\ell+1}}}, \ldots, q_{b_m/x_{i_{\ell+m}}}$ with $b_1, \ldots, b_m \in \{\mathsf{true}, \mathsf{false}\}$.

4 Expressiveness: Regular vs. Monotone AC-Tree Automata

Obviously, by definition, monotone AC-tree automata are at least as expressive as regular AC-tree automata. We show in this section that monotone AC-tree automata are strictly more expressive than the regular AC-tree automata. In other words, we are going to present a monotone AC-tree automaton whose accepted language can not be defined by any regular AC-tree automaton.

To construct such a tree language, we consider in particular the signature $\mathcal{F}_\oplus = \{\oplus\} \cup \mathcal{F}_0$ consisting of a single AC symbol $\oplus$ and constant symbols $a_1, \ldots, a_n$ ($n \geqslant 1$). We then define the Parikh mapping π ([25]) associated with the signature $\mathcal{F}_\oplus$ as follows. For a term t in $\mathcal{T}(\mathcal{F}_\oplus)$, $\pi(t)$ is a vector v in $\mathbb{N}^n$ such that the i-th component $v(i)$ is the number of occurrences of a_i in t. For instance, $\pi(\oplus(a_1, \oplus(a_3, a_1))) = (2, 0, 1, 0, \ldots, 0)$. The Parikh mapping π is homomorphically extended to tree languages: for a tree language L over $\mathcal{F}_\oplus$, $\pi(L)$ is the set of vectors in $\mathbb{N}^n$ defined as $\pi(L) = \{\pi(t) \mid t \in L\}$.

Proposition 1 ([4]). *Given an AC-regular tree language L over $\mathcal{F}_\oplus$, the set $\pi(L)$ is a semi-linear set over $\mathbb{N}^n$.* □

The reverse of the above property also holds; for a semi-linear set S, there effectively exists an AC-regular tree language L with $\pi(L) = S$. We recall that a subset S of $\mathbb{N}^n$ is called a *linear set* if $S = \mathsf{Lin}(b, p_1, \ldots, p_k)$, where b is a vector, called *base*, in $\mathbb{N}^n$ and $p_1, \ldots, p_k$ are a finite number k of vectors, called *periods*, such that

$$\mathsf{Lin}(b, p_1, \ldots, p_k) = \{\, b + \sum_{i=1}^{k} (\lambda_i \times p_i) \mid \lambda_1, \ldots, \lambda_k \in \mathbb{N} \,\}.$$

A finite union of such linear sets is called a *semi-linear set*.

Lemma 1. *Suppose $\mathcal{F}_\oplus$ is defined with 5 constants. There exists a monotone AC-tree automaton $\mathcal{A}_\leqslant/\mathsf{AC}$ over $\mathcal{F}_\oplus$ defining a tree language $L_\leqslant$ such that*

$$\pi(L_\leqslant) = \{\, (k_1, k_2, k_3, 1, 2) \mid\ k_3 \leqslant k_1 \times k_2 \ \textit{for}\ \ k_1, k_2, k_3 \in \mathbb{N} \,\}.$$

Proof. We take $\mathsf{a}, \mathsf{b}, \mathsf{c}, \#, \mathsf{s}$ for the constants of $\mathcal{F}_\oplus$. The corresponding Parikh images are the numbers of these constants in the above order. We define the tree automaton $\mathcal{A}_\leqslant = (\mathcal{F}_\oplus, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta_\leqslant)$ over $\mathcal{F}_\oplus$, where

$$\begin{array}{lll}
\mathcal{Q}: & p_\mathsf{a} \quad p_\mathsf{b} \quad p_\mathsf{c} \quad p_\# \quad p_\mathsf{s} \quad p_{fin} \quad q_\mathsf{a} \quad q_\# \quad q_\mathsf{s} \quad r_\# & \\
\mathcal{Q}_{fin}: & p_{fin} & \\
\Delta_\leqslant: & \mathsf{a} \to p_\mathsf{a} \quad \mathsf{b} \to p_\mathsf{b} \quad \mathsf{c} \to p_\mathsf{c} \quad \# \to p_\# \quad \mathsf{s} \to q_\mathsf{s} \quad q_\mathsf{s} \oplus q_\mathsf{s} \to p_\mathsf{s} & \\
& p_\# \oplus p_\mathsf{a} \to p_\# \qquad p_\# \oplus p_\mathsf{a} \to q_\# \oplus q_\mathsf{a} & \\
& q_\# \oplus p_\mathsf{c} \to p_\# \qquad p_\# \oplus p_\mathsf{b} \to r_\# & \\
& r_\# \oplus q_\mathsf{a} \to r_\# \oplus p_\mathsf{a} \qquad r_\# \oplus p_\mathsf{s} \to p_\# \oplus p_\mathsf{s} \qquad p_\# \oplus p_\mathsf{s} \to p_{fin} &
\end{array}$$

We denote by $|t|_\alpha$ the number of occurrences of a constant α ($\in \mathcal{F}_0 \cup \mathcal{Q}$) in a term t over $\mathcal{F}_\oplus \cup \mathcal{Q}$. We observe that for any term t over $\mathcal{F}_\oplus$ such that $|t|_\# = 1$ and $|t|_\mathsf{s} = 2$ and $|t|_\mathsf{c} \leqslant |t|_\mathsf{a} \times |t|_\mathsf{b}$, there exists a derivation $t \to^*_{\mathcal{A}_\leqslant/\mathsf{AC}} p_{fin}$ from t to p_{fin}. In order to prove this observation, let us define the assertions and the algorithm in Fig. 2. The function apply in the algorithm corresponds to a single application of its argument to a term in consideration. The derivation of t is the sequence of terms obtained during the computation. Proofs of correctness and termination easily follow from the annotations. Conversely, for any term t_0 over $\mathcal{F}_\oplus$ and t over $\mathcal{F}_\oplus \cup \mathcal{Q}$, if $t_0 \to^*_{\mathcal{A}_\leqslant/\mathsf{AC}} t$, it holds that:

$$|t_0|_\mathsf{s} = (|t|_\mathsf{s} + |t|_{p_\mathsf{s}}) + 2 \times (|t|_{p_{fin}} + |t|_{p_\mathsf{s}}) \qquad \text{(INV 1)}$$

$$|t_0|_\# = |t|_\# + |t|_{p_\#} + |t|_{q_\#} + |t|_{r_\#} + |t|_{p_{fin}} \qquad \text{(INV 2)}$$

$$|t_0|_\mathsf{a} \geqslant |t|_\mathsf{a} + |t|_{p_\mathsf{a}} + |t|_{q_\mathsf{a}} \qquad \text{(INV 3)}$$

Moreover, if $t_0 \to^*_{\mathcal{A}_\leqslant/\mathsf{AC}} p_{fin}$, then by (INV 1), $|t_0|_\mathsf{s} = 2$ and by (INV 2), $|t_0|_\# = 1$. Now we suppose $|t_0|_\# = 1$. Due to (INV 3), we have

$$|t_0|_\mathsf{c} - (|t|_\mathsf{c} + |t|_{p_\mathsf{c}}) \leqslant |t_0|_\mathsf{a} \times (|t_0|_\mathsf{b} - (|t|_\mathsf{b} + |t|_{p_\mathsf{b}})) + |t|_{q_\mathsf{a}} \times (1 - |t|_{r_\#}) - |t|_{q_\#}.$$

Accordingly, if $t_0 \to^*_{\mathcal{A}_\leqslant/\mathsf{AC}} p_{fin}$, then $|t_0|_\mathsf{c} \leqslant |t_0|_\mathsf{a} \times |t_0|_\mathsf{b}$. Therefore, $t_0 \in L_\leqslant$ if and only if $\pi(t_0) = (k_1, k_2, k_3, 1, 2)$ with $k_3 \leqslant k_1 \times k_2$. □

Theorem 2. *The family of AC-regular tree languages is properly included in the family of AC-monotone tree languages.*

Proof. Straightforward from Proposition 1 and Lemma 1, because the Parikh image of $L_\leqslant$ is not semi-linear. □

5 Complementation of AC-Monotone Tree Languages

As explained in the introduction, *monotone* rules in tree case correspond to context-sensitive grammars in word case. In fact, based on this observation, we proved in a

previous paper [22] that A-monotone tree languages are closed under Boolean operations by the reduction from the fact that context-sensitive languages are closed under complementation. In this section, however, we show that AC-monotone tree languages are *not* closed under complementation.

Theorem 3. *There exists an AC-monotone tree language whose complement is not an AC-monotone tree language.*

In the remaining part of this section, we devote to show the proof of Theorem 3. Our proof proceeds in the way of proof by contradiction.

Lemma 2. *Suppose $\mathcal{F}_\oplus$ is defined with 5 constants. There exists an AC-tree automaton $\mathcal{A}_</\mathsf{AC}$ over $\mathcal{F}_\oplus$ defining a tree language $L_<$ such that*

$$L_< = \{\, (k_1, k_2, k_3, 1, 2) \mid k_3 < k_1 \times k_2 \ \textit{for}\ \ k_1, k_2, k_3 \in \mathbb{N} \,\}.$$

Proof. We define the automaton $\mathcal{A}_</\mathsf{AC}$ exactly as is the monotone AC-tree automaton $\mathcal{A}_\leqslant/\mathsf{AC}$ in Lemma 1 *except* where the rule $q_\mathsf{s} \oplus q_\mathsf{s} \to p_\mathsf{s}$ replaced by the rule $q_\mathsf{s} \oplus q_\mathsf{s} \to p_\mathsf{s} \oplus p_\mathsf{c}$. One can show as we have done for $\mathcal{A}_\leqslant/\mathsf{AC}$, that for any term t_0 in $\mathcal{T}(\mathcal{F}_\oplus)$, $t_0 \to^*_{\mathcal{A}_</\mathsf{AC}} p_{\mathit{fin}}$ if and only if $\pi(t_0) = (k_1, k_2, k_3, 1, 2)$ with $k_3 < k_1 \times k_2$. □

Let us consider the tree language $L_\geqslant$ defined below over the above signature $\mathcal{F}_\oplus = \{\oplus\} \cup \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \#, \mathsf{s}\}$:

$$L_\geqslant = \{\, (k_1, k_2, k_3, 1, 2) \mid k_3 \geqslant k_1 \times k_2 \ \text{ for }\ k_1, k_2, k_3 \in \mathbb{N} \,\},$$

and we take the hypothesis

$\mathcal{H}$: $L_\geqslant$ is an AC-monotone tree language.

We then state the following property associated to $\mathcal{H}$.

Lemma 3. *If $\mathcal{H}$ holds, there exists a monotone AC-tree automaton that accepts $L_=$ over $\mathcal{F}_\oplus$ such that $\pi(L_=) = \{(k_1, k_2, k_3, 1, 2) \mid k_3 = k_1 \times k_2 \ \textit{for}\ \ k_1, k_2, k_3 \in \mathbb{N}\}$.*

Proof. Due to $\mathcal{H}$, there exists a monotone AC-tree automaton $\mathcal{A}_\geqslant/\mathsf{AC}$ with $\mathcal{L}(\mathcal{A}_\geqslant/\mathsf{AC}) = L_\geqslant$. It is known that the class of monotone AC-tree automata is effectively closed under intersection (Theorem 3, [21]). Then we let $\mathcal{B}/\mathsf{AC}$ be the intersection of $\mathcal{A}_\leqslant/\mathsf{AC}$ in the previous section and $\mathcal{A}_\geqslant/\mathsf{AC}$. According to the trivial fact that $(n_1 \geqslant n_2) \wedge (n_1 \leqslant n_2)$ if and only if $n_1 = n_2$, $\mathcal{B}/\mathsf{AC}$ accepts $L_\geqslant \cap L_\leqslant$, and therefore, $\mathcal{B}/\mathsf{AC}$ accepts $L_=$. □

Lemma 4. *If $\mathcal{H}$ holds, there exists an algorithm that takes as an input $\mathcal{D}$ a diophantine equation and returns as an output "yes" if $\mathcal{D}$ admits a non-negative solution; otherwise, "no".*

Proof. Let us assume a finite set of variables $x_1, \ldots, x_n$ ranging over the natural numbers $\mathbb{N}$. We consider a *system of numerical equations* $\mathcal{S} = \{\mathsf{Eq}_1, \ldots, \mathsf{Eq}_m\}$, where each Eq_ℓ ($1 \leqslant \ell \leqslant m$) in $\mathcal{S}$ is in one of the following forms:

$$x_i = c \quad (c\text{: a fixed natural number}) \qquad x_i = x_j + x_k \qquad x_i = x_j \times x_k$$

Here i must be different from j and k, i.e. x_i does not occur in the right-hand side of the same equation. But a variable x_i may occur in the left hand-sides of different equations. A solution σ for an equation Eq_ℓ is a mapping from $\{ x_1, \ldots, x_n \}$ to $\mathbb{N}$, such that the structure $(\mathbb{N}, +, *, =)$ is a model of Eq_ℓ under the valuation σ. A solution σ for a system $\mathcal{S}$ is a solution for every equation in $\mathcal{S}$.

It is well-known that from any diophantine equation $\mathcal{D}$, one can compute a system of numerical equations $\mathcal{S}$ such that $\mathcal{D}$ admits a solution if and only if $\mathcal{S}$ admits a solution. Now, for each equation Eq_ℓ in $\mathcal{S}$, we define a monotone AC-TA $\mathcal{A}_{\mathsf{Eq}_\ell/\mathsf{AC}}$ over the signature $\mathcal{F}_\oplus = \{ \oplus \} \cup \{ \mathsf{a}_1, \ldots, \mathsf{a}_n, \#, \mathsf{s} \}$, such that for any term t in $\mathcal{T}(\mathcal{F}_\oplus)$, $t \in \mathcal{L}(\mathcal{A}_{\mathsf{Eq}_\ell}/\mathsf{AC})$ if and only if $|t|_\# = 1$, $|t|_s = 2$ and the valuation σ defined as $\sigma(x_i) = |t|_{\mathsf{a}_i}$ (for $1 \leqslant i \leqslant n$) is a solution for Eq_ℓ. For each kind of numerical equations, we define the transition rules of the automaton assuming that p_{fin} is the unique final state:

- For the constraint equation $x_i = 0$ we define the tree automaton $\mathcal{A}_{x_i=0}$ equipped with the transition rules
  $\{ p_\mathsf{s} \oplus p_\mathsf{s} \to q_\mathsf{s},\ q_\mathsf{s} \oplus p_\# \to p_{fin} \} \cup$
  $\{ p_{\mathsf{a}_j} \oplus p_{fin} \to p_{fin} \mid j \neq i \text{ and } 1 \leqslant j \leqslant n \}$
  with the rules for constants $\{ \mathsf{a}_j \to p_{\mathsf{a}_j} \mid 1 \leqslant j \leqslant n \} \cup \{ \# \to p_\#,\ \mathsf{s} \to p_\mathsf{s} \}$. For $x_i = c\ (c > 0)$ we additionally take the transition rules
  $\{ p_{\mathsf{a}_i} \oplus p_{fin} \to p_1 \} \cup \{ p_{\mathsf{a}_i} \oplus p_j \to p_{j+1} \mid 1 \leqslant j \leqslant c-2 \} \cup$
  $\{ p_{\mathsf{a}_i} \oplus p_{c-1} \to p_{fin} \}$.
- For the linear equation $x_i + x_j = x_k$ we define the tree automaton $\mathcal{A}_{x_i+x_j=x_k}$ equipped with the transition rules
  $\{ p_\mathsf{s} \oplus p_\mathsf{s} \to q_\mathsf{s},\ q_\mathsf{s} \oplus p_\# \to p_{fin} \} \cup \{ p_{\mathsf{a}_i} \oplus p_{\mathsf{a}_k} \to p,\ p_{\mathsf{a}_j} \oplus p_{\mathsf{a}_k} \to p \} \cup$
  $\{ p_{\mathsf{a}_\ell} \oplus p_{fin} \to p_{fin} \mid \ell \neq i \text{ and } \ell \neq j \text{ and } \ell \neq k \} \cup$
  $\{ p \oplus p \to p,\ p \oplus p_{fin} \to p_{fin} \}$
  with the rules for constants $\{ \mathsf{a}_\ell \to p_{\mathsf{a}_\ell} \mid 1 \leqslant \ell \leqslant n \} \cup \{ \# \to p_\#,\ \mathsf{s} \to p_\mathsf{s} \}$.
- Finally, for a numerical equation $x_i = x_j \times x_k$, we build the automaton $\mathcal{A}_{x_i=x_j\times x_k}$; let $\mathcal{B}/\mathsf{AC}$ the automaton defined in the proof of Lemma 3. We assume without loss of generality that p_{fin} is the unique final state of $\mathcal{B}/\mathsf{AC}$. We then define $\mathcal{A}_{x_i=x_j\times x_k}$ by relabeling c by a_i, a by a_j and b by a_k and by adding the transition rules
  $\{ p_{\mathsf{a}_\ell} \oplus p_{fin} \to p_{fin} \mid \ell \neq i \text{ and } \ell \neq j \text{ and } \ell \neq k \} \cup$
  $\{ \mathsf{a}_\ell \to p_{\mathsf{a}_\ell} \mid 1 \leqslant \ell \leqslant n \}$.

One should note that for the first two cases, transition rules for $\#$ and s are not essential, but they must be included under our construction if a system $\mathcal{S}$ contains an equation Eq_k of the multiplication $x_i = x_j \times x_k$.

Accordingly, for the system $\mathcal{S} = \{ \mathsf{Eq}_1, \ldots, \mathsf{Eq}_m \}$ of numerical equations, we can construct a monotone AC-TA $\mathcal{A}_\mathcal{S}/\mathsf{AC}$ such that

$$\mathcal{L}(\mathcal{A}_\mathcal{S}/\mathsf{AC}) = \bigcap_{1 \leqslant \ell \leqslant m} \mathcal{L}(\mathcal{A}_{\mathsf{Eq}_\ell}/\mathsf{AC})$$

whose accepted language is non-empty if and only if $\mathcal{S}$ admits a solution. Since the emptiness problem for monotone AC-TA is decidable, there exists an algorithm under the hypothesis $\mathcal{H}$ that takes as an input a diophantine equation $\mathcal{D}$ and returns "yes" if there is a non-negative solution; otherwise, "no". □

It is well-known that Hilbert's 10th problem [16] is undecidable, even only in the case of non-negative solutions to be considered. Thus we obtain the following property.

Theorem 4. *There is no monotone AC-tree automaton that accepts $L_{\geqslant}$ over the signature $\mathcal{F}_{\oplus}$.*

Corollary 1. *The class of AC-monotone tree languages is not closed under complementation.*

Proof. Straightforward from Theorem 4, as AC-monotone tree languages are closed under intersection and $L_{\geqslant} = (L_{<})^{\mathsf{c}} \cap \{\, t \in \mathcal{T}(\mathcal{F}_{\oplus}) \mid |t|_{\#} = 1 \;\&\; |t|_{\mathsf{s}} = 2 \,\}$, where $L_{<}$ and $\{\, t \in \mathcal{T}(\mathcal{F}_{\oplus}) \mid |t|_{\#} = 1 \;\&\; |t|_{\mathsf{s}} = 2 \,\}$ are AC-monotone tree languages. □

6 The Inclusion Problem for Monotone AC-Tree Automata

Using the previous tree automata construction, we show in this section that the inclusion problem for AC-monotone tree languages is undecidable. The remainder of this section is devoted to the proof of the following undecidability result.

Theorem 5. *Given two monotone AC-tree automata $\mathcal{A}_1/\mathsf{AC}$ and $\mathcal{A}_2/\mathsf{AC}$ over the same signature, the problem whether $\mathcal{L}(\mathcal{A}_1/\mathsf{AC}) \subseteq \mathcal{L}(\mathcal{A}_2/\mathsf{AC})$ is not decidable.*

As we did in the previous section, we consider a system $\mathcal{S} = \{\, \mathsf{Eq}_1, \ldots, \mathsf{Eq}_m \,\}$ of numerical equations defined over a finite set of variables $\{\, x_1, \ldots, x_n \,\}$. One should note that according to the syntax, Eq_i is an equation in the form of $x_j = e$, where e is either a fixed natural number c, the addition $x_k + x_\ell$, or the multiplication $x_k \times x_\ell$, such that $x_j \neq x_k$ and $x_j \neq x_\ell$.
We then define the system $\mathcal{S}_{\leqslant}$ of *inequations* obtained by replacing each equation $x_i = e$ by the inequation $x_i \leqslant e$. Namely, $\mathcal{S}_{\leqslant} = \{\, x_i \leqslant e \mid \; x_i = e \; \in \mathcal{S} \,\}$.
Finally we define, for each k with $1 \leqslant k \leqslant m$, $\mathcal{S}_k$ a system of inequations obtained from $\mathcal{S}_{\leqslant}$ by replacing only the k-th inequation $x_i \leqslant e_k$ by the strict inequation $x_i < e_k$.
From previous sections, we know that one can effectively associate with each inequation Ineq_k (being either $x_j \leqslant e_k$ or $x_j < e_k$) a monotone AC-tree automaton $\mathcal{A}_{\mathsf{Ineq}_k}$ such that a term t from $\mathcal{T}(\mathcal{F}_{\oplus})$ is accepted by an automaton $\mathcal{A}_{\mathsf{Ineq}_k}/\mathsf{AC}$ if and only if $|t|_{\#} = 1$, $|t|_{\mathsf{s}} = 2$ and either Ineq_k is of the form

- $x_i \leqslant c$ and $|t|_{\mathsf{a}_i} \leqslant c$ (resp. $x_i < c$ and $|t|_{\mathsf{a}_i} < c$),
- $x_i \leqslant x_v + x_w$ and $|t|_{\mathsf{a}_i} \leqslant |t|_{\mathsf{a}_v} + |t|_{\mathsf{a}_w}$ (resp. $x_i < x_v + x_w$ and $|t|_{\mathsf{a}_i} < |t|_{\mathsf{a}_v} + |t|_{\mathsf{a}_w}$), or
- $x_i \leqslant x_v * x_w$ and $|t|_{\mathsf{a}_i} \leqslant |t|_{\mathsf{a}_v} * |t|_{\mathsf{a}_w}$ (resp. $x_i < x_v * x_w$ and $|t|_{\mathsf{a}_i} < |t|_{\mathsf{a}_v} * |t|_{\mathsf{a}_w}$).

Moreover, we let

$$\mathcal{A}_{\mathcal{S}_{\leqslant}}/\mathsf{AC} = \bigcap_{\mathsf{Ineq} \in \mathcal{S}_{\leqslant}} \mathcal{A}_{\mathsf{Ineq}}/\mathsf{AC},$$

$$\mathcal{A}_{\mathcal{S}_k}/\mathsf{AC} = \bigcap_{\mathsf{Ineq} \in \mathcal{S}_k} \mathcal{A}_{\mathsf{Ineq}}/\mathsf{AC}$$

for all $1 \leqslant k \leqslant m$. In the above definition, $\bigcap_{\mathsf{Ineq} \in \mathcal{S}_{\leqslant}} \mathcal{A}_{\mathsf{Ineq}}/\mathsf{AC}$ represents an AC-TA that accepts the tree language accepted by $\mathcal{A}_{\mathsf{Ineq}}/\mathsf{AC}$ for all $\mathsf{Ineq} \in \mathcal{S}$.

Lemma 5. $\mathcal{L}(\mathcal{A}_{\mathcal{S}_\leqslant}/\mathsf{AC}) \not\subseteq \mathcal{L}(\bigcup_{1\leqslant i\leqslant m} \mathcal{A}_{\mathcal{S}_i}/\mathsf{AC})$ *if and only if* $\mathcal{S}$ *admits a solution.* □

Theorem 5 follows easily from Lemma 5 and the effective closedness under union and intersection of monotone AC-tree automata.

7 Concluding Remarks

In this paper, we have shown the 4 new results (Theorems 1, 2, 5 and Corollary 1) for the class of monotone AC-tree automata. Our proof technique used for showing the expressiveness of AC-monotone tree languages explains also a new idea of how to interpret by AC-tree automata the *arithmetic constraints* over the natural numbers, while an observation obtained from this tree automata construction gives rise to the negative closure property of the complementation and the undecidability of the inclusion problem.
For further research along monotone AC-tree automata, it might be interesting to consider the question about decision problems concerning regularity, called the *regularity* problem; it is not clear how to determine, given a monotone AC-tree automaton, whether the accepted tree language can also be accepted by some regular AC-tree automaton. Useful ideas to solve this decision problem are found in the study about Petri nets. In fact, it is known that the *semi-linearity* problem for Petri nets is decidable [15]. The regularity problem for AC-monotone tree languages can be regarded in some sense as an generalization of the above semi-linearity problem.
Another interesting question about monotone AC-tree automata is the universality problem [6]; this problem is known to be decidable for regular AC-tree automata and it is undecidable for monotone A-tree automata.

References

1. A. Armando, D. Basin, M. Bouallagui, Y. Chevalier, L. Compagna, S. Mödersheim, M. Rusinowitch, M. Turuani, L. Viganò, and L. Vigneron. The AVISS Security Protocol Analysis Tool. In *Proc. of 14th CAV, Copenhagen (Denmark)*, volume 2404 of *LNCS*, pages 349–353. Springer, 2002.
2. I. Boneva and J.-M. Talbot. Automata and Logics for Unranked and Unordered Trees. In *Proc. of 16th RTA, Nara (Japan)*, volume 3467 of *LNCS*, pages 500–515. Springer, 2005.
3. A. Bouhoula, J.P. Jouannaud, and J. Meseguer. Specification and Proof in Membership Equational Logic. *Theoretical Computer Science*, 236:35–132, 2000.
4. T. Colcombet. Rewriting in the Partial Algebra of Typed Terms Modulo AC. In *Proc. of 4th INFINITY, Brno (Czech Republic)*, volume 68(6) of *ENTCS*. Elsevier, 2002.
5. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree Automata Techniques and Applications, 2002. (http://www.grappa.univ-lille3.fr/tata).
6. N. Dershowitz and R. Treinen. *Problem #101*. The RTA List of Open Problems. Available at http://www.lsv.ens-cachan.fr/rtaloop/.
7. P. Devienne, J.-M. Talbot, and S. Tison. Set-Based Analysis for Logic Programming and Tree Automata. In *Proc. of 4th SAS, Paris (France)*, volume 1302 of *LNCS*, pages 127–140. Springer, 1997.
8. J. Esparza. Decidability of Model-Checking for Infinite-State Concurrent Systems. *Acta Informatica*, 34:85–107, 1997.

9. J. Esparza. Decidability and Complexity of Petri Net Problems – An Introduction. In *Petri Nets*, volume 1491 of *LNCS*, pages 374–428. Springer, 1998.
10. J. Esparza. Grammars as Processes. In *Formal and Natural Computing – Essays Dedicated to Grzegorz Rozenberg*, volume 2300 of *LNCS*, pages 277–297. Springer, 2002.
11. J.P. Gallagher and G. Puebla. Abstract Interpretation over Non-Deterministic Finite Tree Automata for Set-Based Analysis of Logic Programs. In *Proc. of 4th PADL, Portland (USA)*, volume 2257 of *LNCS*, pages 243–261. Springer, 2002.
12. T. Genet and F. Klay. Rewriting for Cryptographic Protocol Verification. In *Proc. of 17th CADE, Pittsburgh (USA)*, volume 1831 of *LNCS*, pages 271–290. Springer, 2000.
13. S. Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, 1966.
14. J. Goubault-Larrecq and K.N. Verma. Alternating Two-way AC-Tree Automata. Research Report LSV-02-11, Laboratoire Spécification et Vérification, November 2002. Draft available at http://www.lsv.ens-cachan.fr/Publis/.
15. D. Hauschildt. Semilinearity of the Reachability Set is Decidable for Petri Nets. Technical Report FBI-HH-B-146/90, Universität Hamburg, 1990.
16. D. Hilbert. Mathematical Problem. In *Proc. of the Symposia in Pure Mathematics, Providence (USA), Matiematical Developments Arising from Hilbert's Problems*, volume 28, pages 1–34. Americal Mathematical Society, 1976.
17. H. Hosoya, J. Vouillon, and B.C. Pierce. Regular Expression Types for XML. In *Proc. of 5th ICFP, Montreal (Canada)*, volume 35(9) of *SIGPLAN Notices*, pages 11–22. ACM, 2000.
18. M. Kudlek and V. Mitrana. Normal Forms of Grammars, Finite Automata, Abstract Families, and Closure Properties of Multiset Languages. In *Multiset Processing*, volume 2235 of *LNCS*, pages 135–146. Springer, 2001.
19. S.Y. Kuroda. Classes of Languages and Linear Bounded Automata. *Information and Control*, 7(2):207–223, 1964.
20. D. Lugiez. Counting and Equality Corstraints for Multitree Automata. In *Proc. of 6th FOSSACS, Warsaw (Poland)*, volume 2620 of *LNCS*, pages 328–342. Springer, 2003.
21. H. Ohsaki. Beyond Regularity: Equational Tree Automata for Associative and Commutative Theories. In *Proc. of 15th CSL, Paris (France)*, volume 2142 of *LNCS*, pages 539–553. Springer, 2001.
22. H. Ohsaki, H. Seki, and T. Takai. Recognizing Boolean Closed A-Tree Languages with Membership Conditional Rewriting Mechanism. In *Proc. of 14th RTA, Valencia (Spain)*, volume 2706 of *LNCS*, pages 483–498. Springer, 2003.
23. H. Ohsaki and T. Takai. Decidability and Closure Properties of Equational Tree Languages. In *Proc. of 13th RTA, Copenhagen (Denmark)*, volume 2378 of *LNCS*, pages 114–128. Springer, 2002.
24. H. Ohsaki and T. Takai. ACTAS: A System Design for Associative and Commutative Tree Automata Theory. In *Proc. of 5th RULE, Aachen (Germany)*, volume 124(1) of *ENTCS*, pages 97–111. Elsevier, 2005.
25. R.J. Parikh. On Context-Free Languages. *Journal of the ACM*, 13(4):570–581, 1966.
26. W. Savitch. Relationships between Nondeterministic and Deterministic Tape Complexities. *Journal of Computer and Systems Sciences*, 4(2):177–192, 1970.
27. H. Seidl, T. Schwentick, and A. Muscholl. Numerical Document Queries. In *Proc. of 22nd PODS, San Diego (USA)*, pages 155–166. ACM, 2003.
28. K.N. Verma. On Closure under Complementation of Equational Tree Automata for Theories Extending AC. In *Proc. of 10th LPAR, Almaty (Kazakhstan)*, volume 2850 of *LNCS*, pages 183–197. Springer, 2003.
29. K.N. Verma. Two-Way Equational Tree Automata for AC-like Theories: Decidability and Closure Properties. In *Proc. of 14th RTA, Valencia (Spain)*, volume 2706 of *LNCS*, pages 180–197. Springer, 2003.

/* INVARIANT:

$$|t|_{p_{\mathsf{c}}} + |t|_{q_{\mathsf{a}}} + (|t|_{p_{\mathsf{a}}} \times |t|_{r_\#}) \;\leqslant\; (|t|_{p_{\mathsf{a}}} + |t|_{q_{\mathsf{a}}}) \times (|t|_{p_{\mathsf{b}}} + |t|_{r_\#}) + |t|_{q_\#}$$

$$|t|_{p_\#} + |t|_{q_\#} + |t|_{r_\#} = \; |t|_{p_{\mathsf{s}}} \; = 1$$

$$|t|_{\mathsf{a}} \; + |t|_{\mathsf{b}} \; + |t|_{\mathsf{c}} \;\; = \;\; 0$$

*/

/* Given t in $\mathcal{T}(\mathcal{F}_\oplus)$ such that $|t|_\# = 1$ and $|t|_{\mathsf{s}} = 2$ and $|t|_{\mathsf{c}} \leqslant |t|_{\mathsf{a}} \times |t|_{\mathsf{b}}$ */

while ($|t|_{\mathsf{a}} + |t|_{\mathsf{b}} + |t|_{\mathsf{c}} + |t|_\# + |t|_{\mathsf{s}} > 0$) {

  apply $\mathsf{a} \to p_{\mathsf{a}},\ \mathsf{b} \to p_{\mathsf{b}},\ \mathsf{c} \to p_{\mathsf{c}},\ \# \to p_\#,\ \mathsf{s} \to q_{\mathsf{s}}$

}

apply $q_{\mathsf{s}} \oplus q_{\mathsf{s}} \to p_{\mathsf{s}}$; /* INVARIANT & $|t|_{p_\#} = 1$ & $|t|_{q_a} = 0$ */

while ($|t|_{p_{\mathsf{b}}} > 0$) { /* INVARIANT & $|t|_{p_\#} = 1$ & $|t|_{q_a} = 0$ */

  while ($|t|_{p_{\mathsf{a}}} > 0 \;\&\; |t|_{p_{\mathsf{c}}} > 0$) {

    apply $p_\# \oplus p_{\mathsf{a}} \to q_\# \oplus q_{\mathsf{a}}$; apply $q_\# \oplus p_{\mathsf{c}} \to p_\#$

  }

  /* INVARIANT & $|t|_{p_\#} = 1$ */

  apply $p_\# \oplus p_{\mathsf{b}} \to r_\#$;

  /* INVARIANT & $|t|_{r_\#} = 1$ */

  while ($|t|_{q_a} > 0$) {

    apply $r_\# \oplus q_{\mathsf{a}} \to r_\# \oplus p_{\mathsf{a}}$

  }

  /* INVARIANT & $|t|_{r_\#} = 1$ & $|t|_{q_a} = 0$ */

  apply $r_\# \oplus p_{\mathsf{s}} \to p_\# \oplus p_{\mathsf{s}}$

  /* INVARIANT & $|t|_{p_\#} = 1$ & $|t|_{q_a} = 0$ */

}

/* $|t|_{p_\#} = 1$ & $|t|_{q_a} = |t|_{p_b} = |t|_{p_c} = 0$ */

while ($|t|_{p_a} > 0$) {

  apply $p_\# \oplus p_{\mathsf{a}} \to p_\#$

  }

/* $t = p_\# \oplus p_s$ */

apply $p_\# \oplus p_{\mathsf{s}} \to p_{fin}$

/* $t = p_{fin}$ */

Fig. 2. Reduction strategy and the assertions