Decidability and Closure Properties of Equational Tree Languages

Hitoshi Ohsaki and Toshinori Takai

National Institute of Advanced Industrial Science and Technology
Nakoji 3–11–46, Amagasaki 661–0974, Japan
{ohsaki,takai}@ni.aist.go.jp

Abstract. Equational tree automata provide a powerful tree language framework that facilitates to recognize congruence closures of tree languages. In the paper we show the emptiness problem for AC-tree automata and the intersection-emptiness problem for regular AC-tree automata, each of which was open in our previous work [20], are decidable, by a straightforward reduction to the reachability problem for ground AC-term rewriting. The newly obtained results generalize decidability of so-called *reachable property problem* of Mayr and Rusinowitch [17]. We then discuss complexity issue of AC-tree automata. Moreover, in order to solve some other questions about regular A- and AC-tree automata, we recall the basic connection between word languages and tree languages.

1 Introduction

Tree automata theory has been studied with much attention as the basis of computational model dealing with terms. Recently tree automata techniques are applied in a particular domain for automatically solving security problems of *cryptographic protocols* (network protocols with some cryptographic primitives) [7, 10, 19]. This is a natural phenomenon resulting from the background of tree automata, in which complexity issues, especially on decidability, have been investigated with considerable efforts. In fact, applications based on tree automata theory, e.g. for program and system verification [9, 22], depend deeply upon decidability results and closure properties.

Regular tree languages, which is the counterpart of regular word languages, are known to be well-behaved, in the sense that they are closed under boolean operations and their decision problems are computable in many cases. Moreover, in term rewriting there are some useful subclasses in which rewriting closures preserve regularity [24, 25]. But, on the other hand, it causes the situation where questions of non-regular tree language are considered to be troublesome. For instance, because term algebra modulo equational theory is not handled with regular tree automata (e.g., AC-congruence closure of a regular tree language is not regular [3]), the standard tree automata technique can not be applied for modeling cryptographic primitives like Diffie-Hellman exponentiation and one-time pads [23]. The difficulty of system verification allowing AC-operators is also found in another algebraic approach [18].

Over the last decade tree automata framework has been generalized. So far we can find several extensions, such as tree automata with constraints (e.g. [1, 13]), tree set automata [14] and timed tree automata [12]. Each of the extensions covers a wider class of regular tree languages, while keeping the benefit of some decidability results and closure properties. *Equational tree automata* proposed in our previous paper [20] is the recent extension, with which congruence closures of recognizable tree languages are recognizable. We also proved that in certain useful cases, recognizable equational tree languages are closed under union and intersection operations. However, there are several important questions remaining open in the previous work. Our goal of this paper is to find the answers to the open questions. More precisely, we show the following results:

(1) decidability of emptiness problem for AC-tree automata,
(2) decidability of intersection-emptiness, subset and universality problems for regular A- and regular AC-tree automata, and
(3) closure properties, intersection and complement, of A-regular and AC-regular tree languages.

Moreover, we show the computability of equational rewrite descendents. This result gives rise to the decidability of reachable property problem for ground AC-term rewriting.

The rest of the paper is organized as follows. In the remaining part of this section we fix our terminologies and notations on term rewriting. In Section 2 we recall equational tree automata. And we show a positive answer to the problem (1), by generalizing the decidability result of Mayr and Rusinowitch [16]. Using this result we show the computability of ground AC-rewrite descendents. In Section 3 we discuss the complexity issue on some AC-tree automata problems. The problems (2) and (3) for A-regular tree languages are (negatively) solved in Section 4. We then partially solve the same questions for AC-regular case in Section 5. Finally we conclude this paper by summarizing decidability results and closure properties of A-, C- and AC-tree languages.

We assume familiarity with the basics of term rewriting. An *equation* over the signature $\mathcal{F}$ (a finite set of function symbols f whose arity is denoted by $\mathsf{arity}(f)$) and a set $\mathcal{V}$ of variables is a pair (s, t) of terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. An equation (s, t) is denoted by $s \approx t$. An equation $l \approx r$ is called *linear* if neither l nor r contains multiple positions of the same variable. We say $l \approx r$ is *variable-preserving* if a multiset occurrence of each variable x in l is the same as a multiset occurrence of x in r. A *ground* equation is an equation whose left- and right-hand sides do not contain any variable. An *equational system* (ES for short) $\mathcal{E}$ is a set of equations. Given two sets $\mathcal{F}_{\mathrm{A}}$, $\mathcal{F}_{\mathrm{C}}$ of some binary function symbols in $\mathcal{F}$. The intersection of $\mathcal{F}_{\mathrm{A}}$ and $\mathcal{F}_{\mathrm{C}}$ is denoted by $\mathcal{F}_{\mathrm{AC}}$. An ES consisting of associativity axioms $f(f(x, y), z) \approx f(x, f(y, z))$ for all $f \in \mathcal{F}_{\mathrm{A}}$ is denoted by $\mathsf{A}(\mathcal{F}_{\mathrm{A}})$. An ES of commutativity axioms $f(x, y) \approx f(y, x)$ for all $f \in \mathcal{F}_{\mathrm{C}}$ is $\mathsf{C}(\mathcal{F}_{\mathrm{C}})$. We write $\mathsf{AC}(\mathcal{F}_{\mathrm{AC}})$ for the union of $\mathsf{A}(\mathcal{F}_{\mathrm{AC}})$ and $\mathsf{C}(\mathcal{F}_{\mathrm{AC}})$. If unnecessary to be explicit, we simply write A, C and AC. A binary relation $\to_{\mathcal{E}}$ induced by an ES $\mathcal{E}$ is defined as follows: $s \to_{\mathcal{E}} t$ if $s = C[l\sigma]$ and $t = C[r\sigma]$ for some equation $l \approx r \in \mathcal{E}$, context $C \in \mathcal{C}(\mathcal{F}, \mathcal{V})$ and substitution σ. The symmetric closure of $\to_{\mathcal{E}}$ is denoted by

$\vdash_{\mathcal{E}}$. In the paper we do not assume $r \approx l \in \mathcal{E}$ if $l \approx r \in \mathcal{E}$, so $\rightarrow_{\mathcal{E}} \neq \vdash_{\mathcal{E}}$. The equivalence relation of $\rightarrow_{\mathcal{E}}$ (i.e., the reflexive-transitive closure of $\vdash_{\mathcal{E}}$) is written by $\sim_{\mathcal{E}}$.

A term rewriting system is an ES whose equations $l \approx r$ are called *rewrite rules* and are denoted by $l \rightarrow r$. An equational TRS (ETRS for short) $\mathcal{R}/\mathcal{E}$ is a combination of a TRS $\mathcal{R}$ and an ES $\mathcal{E}$ over the same signature $\mathcal{F}$. To emphasize the signature of an ETRS, it can be denoted by the pair $(\mathcal{F}, \mathcal{R}/\mathcal{E})$ of two components. An ETRS is called *ground* if the TRS consists of ground rewrite rules. Note that $\mathcal{E}$ is not necessarily ground. We write $s \rightarrow_{\mathcal{R}/\mathcal{E}} t$ if there exist terms s' and t' such that $s \sim_{\mathcal{E}} s' \rightarrow_{\mathcal{R}} t' \sim_{\mathcal{E}} t$. As a special case, if $\mathcal{E} = \mathsf{A}$ ($\mathcal{E} = \mathsf{C}$, $\mathcal{E} = \mathsf{AC}$, respectively), $\mathcal{R}/\mathcal{E}$ is called an A-TRS (C-TRS, AC-TRS).

Given an ETRS $\mathcal{R}/\mathcal{E}$. A term s reachable from a term t with respect to $\mathcal{R}/\mathcal{E}$, i.e. $t \rightarrow_{\mathcal{R}/\mathcal{E}}^* s$ is called a *descendent* of t. For a set L of terms, descendents of terms in L, i.e. $\{\, t \mid \exists s \in L.\ s \rightarrow_{\mathcal{R}/\mathcal{E}}^* t \,\}$, are denoted by $(\rightarrow_{\mathcal{R}/\mathcal{E}}^*)[L]$.

2 AC-Tree Automata and Emptiness Problem

A *tree automaton* (TA for short) $\mathcal{A} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$ consists of a signature $\mathcal{F}$, a finite set $\mathcal{Q}$ of states (special constants with $\mathcal{F} \cap \mathcal{Q} = \varnothing$), a set $\mathcal{Q}_{fin}$ ($\subseteq \mathcal{Q}$) of final states and a finite set Δ of transition rules in one of the following forms:

$$f(p_1, \ldots, p_n) \rightarrow q \qquad \text{or} \qquad f(p_1, \ldots, p_n) \rightarrow f(q_1, \ldots, q_n)$$

for some $f \in \mathcal{F}$ with $\mathsf{arity}(f) = n$ and $p_1, \ldots, p_n, q, q_1, \ldots, q_n \in \mathcal{Q}$. In the latter form, the root function symbols of the left- and right-hand sides must be the same. An *equational tree automaton* (ETA for short) $\mathcal{A}/\mathcal{E}$ is the combination of a TA $\mathcal{A}$ and an ES $\mathcal{E}$ over the same signature $\mathcal{F}$. We often denote the 5-tuple $(\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta, \mathcal{E})$ for $\mathcal{A}/\mathcal{E}$. An ETA $\mathcal{A}/\mathcal{E}$ is called *regular* if Δ consists of transition rules in the shape of $f(p_1, \ldots, p_n) \rightarrow q$. We say $\mathcal{A}/\mathcal{E}$ is an A-TA (associative-tree automaton) if $\mathcal{E} = \mathsf{A}$. An ETA $\mathcal{R}/\mathcal{E}$ with $\mathcal{E} = \mathsf{C}$ is called a C-TA (commutative-tree automaton). Likewise, if $\mathcal{E} = \mathsf{AC}$, it is called an AC-TA.

We write $s \rightarrow_{\mathcal{A}/\mathcal{E}} t$ if there exist a transition rule $l \rightarrow r \in \Delta$ and a context $C \in \mathcal{C}(\mathcal{F} \cup \mathcal{Q})$ such that $s \sim_{\mathcal{E}} C[l]$ and $t \sim_{\mathcal{E}} C[r]$. The relation $\rightarrow_{\mathcal{A}/\mathcal{E}}$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is called *move relation* of $\mathcal{A}/\mathcal{E}$. The transitive closure and reflexive-transitive closure of $\rightarrow_{\mathcal{A}/\mathcal{E}}$ are denoted by $\rightarrow_{\mathcal{A}/\mathcal{E}}^+$ and $\rightarrow_{\mathcal{A}/\mathcal{E}}^*$. For a TA $\mathcal{A}$, we simply write $\rightarrow_{\mathcal{A}}$, $\rightarrow_{\mathcal{A}}^+$ and $\rightarrow_{\mathcal{A}}^*$, instead. A state q is called *accessible* with $\mathcal{A}/\mathcal{E}$ if there exists a term t in $\mathcal{T}(\mathcal{F})$ such that $t \rightarrow_{\mathcal{A}/\mathcal{E}}^* q$. 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 some subset of $\mathcal{T}(\mathcal{F})$. We can easily observe that: membership problem for an ETA $\mathcal{A}/\mathcal{E}$ is decidable if $\mathcal{E}$ is length-preserving, i.e., if $\mathcal{E}$ is variable-preserving and for every $l \approx r \in \mathcal{E}$, the number of function symbols occurring in l is the same as the number of function symbols in r.

We write $\mathcal{E}(L)$ for $\{t \mid \exists s \in L.\ s \sim_{\mathcal{E}} t\}$. The set $\mathcal{E}(L)$ is called $\mathcal{E}$-*congruence closure* of L. A tree language L is $\mathcal{E}$-*recognizable* if there exists $\mathcal{A}/\mathcal{E}$ such that

$L = \mathcal{L}(\mathcal{A}/\mathcal{E})$. In certain useful cases, $\mathcal{E}$-recognizable tree languages are closed under union and intersection:

Lemma 1. *If $\mathcal{E}$ is a variable-preserving ES, the union of $\mathcal{E}$-recognizable tree languages L_1 and L_2 is $\mathcal{E}$-recognizable. Moreover, if $\mathcal{E} = \mathsf{A}$ (or $\mathcal{E} = \mathsf{AC}$), the intersection of L_1 and L_2 is $\mathcal{E}$-recognizable.* □

A tree language L is called $\mathcal{E}$-*regular* if L is $\mathcal{E}$-recognizable with a regular ETA $\mathcal{A}/\mathcal{E}$. If $\mathcal{E} = \varnothing$, the tree language L is called regular. Every recognizable tree language is regular, i.e. every TA is transformed to a regular TA with the same expressive power. However, $\mathcal{E}$-recognizable tree languages are not $\mathcal{E}$-regular in general.

In the previous paper [20] we studied decidability on equational tree automata. Especially we focused on emptiness problems, i.e. a question of whether $\mathcal{L}(\mathcal{A}/\mathcal{E}) = \varnothing$. In case $\mathcal{E}$ is linear, this question for regular $\mathcal{E}$-tree automata can be reduced to the same question for regular tree automata by using the commutation lemma:

Lemma 2. *Every regular ETA $\mathcal{A}/\mathcal{E}$ with $\mathcal{E}$ linear satisfies $\mathcal{E}(\mathcal{L}(\mathcal{A})) = \mathcal{L}(\mathcal{A}/\mathcal{E})$.* □

Since $\mathcal{E}(\mathcal{L}(\mathcal{A})) = \varnothing$ if and only if $\mathcal{L}(\mathcal{A}) = \varnothing$, the following property is a direct consequence of Lemma 2.

Corollary 1 ([20]). *Emptiness problem for regular $\mathcal{E}$-tree automata with $\mathcal{E}$ linear is decidable.* □

Thus the same problem for regular A-, C- and AC-tree automata is decidable. In contrast, the non-regular A-case is undecidable (Corollary 1, [20]). The non-regular AC-case remains open in the previous paper.

In the following part, we show the positive answer to this open question. First of all, we introduce a recent work on *reachability problem* for ground AC-TRS of Mayr and Rusinowitch [16].

Proposition 1. *Reachability problem for ground AC-TRS, i.e. a question of whether $s \to_{\mathcal{R}/\mathsf{AC}}^* t$ for an arbitrary ground AC-TRS $\mathcal{R}/\mathsf{AC}$ and terms s, t, is decidable.* □

Next we state the following property.

Lemma 3. *Given two ground AC-TRSs $(\mathcal{F}, \mathcal{R}/\mathsf{AC})$ and $(\mathcal{G}, \mathcal{S}/\mathsf{AC})$, where $\mathcal{F}_{\mathsf{AC}} = \mathcal{G}_{\mathsf{AC}}$. If $\mathcal{F}_0 \cap \mathcal{G}_0 = \varnothing$, i.e. $\mathcal{F}$ and $\mathcal{G}$ do not share any constant symbol, then for all terms $s, t \in \mathcal{T}(\mathcal{F} \cup \mathcal{G})$, $s \to_{(\mathcal{R} \cup \mathcal{S})/\mathsf{AC}}^* t$ if and only if $s \to_{\mathcal{R}/\mathsf{AC}}^* s'$ and $s' \to_{\mathcal{S}/\mathsf{AC}}^* t$ for some s'.*

Proof. Since the "if" part is trivial, it suffices to show the "only if" part. Suppose $u \to_{\mathcal{S}/\mathsf{AC}} v \to_{\mathcal{R}/\mathsf{AC}} w$. Then $u \sim_{\mathsf{AC}} C_1[l_1]$ and $v \sim_{\mathsf{AC}} C_1[r_1]$ for some $l_1 \to r_1 \in \mathcal{S}$, and $v \sim_{\mathsf{AC}} C_2[l_2]$ and $w \sim_{\mathsf{AC}} C_2[r_2]$ for some $l_2 \to r_2 \in \mathcal{R}$. By assumption, r_1 is a term in $\mathcal{T}((\mathcal{F} \cup \mathcal{G}) - \mathcal{F}_0)$ and l_2 is a term in $\mathcal{T}((\mathcal{F} \cup \mathcal{G}) - \mathcal{G}_0)$. This yields

a context D such that $D[r_1, l_2] \sim_{\mathsf{AC}} C_1[r_1]$ and $D[r_1, l_2] \sim_{\mathsf{AC}} C_2[l_2]$, and thus, $u \sim_{\mathsf{AC}} D[l_1, l_2] \to_{\mathcal{S}} D[r_1, l_2] \to_{\mathcal{R}} D[r_1, r_2] \sim_{\mathsf{AC}} w$. Hence $u \sim_{\mathsf{AC}} D[l_1, l_2] \to_{\mathcal{R}} D[l_1, r_2] \to_{\mathcal{S}} D[r_1, r_2] \sim_{\mathsf{AC}} w$. This implies $\to^*_{(\mathcal{R} \cup \mathcal{S})/\mathsf{AC}} \subseteq \to^*_{\mathcal{R}/\mathsf{AC}} \cdot \to^*_{\mathcal{S}/\mathsf{AC}}$. $\qquad\square$

In the previous lemma, by taking an AC-TA $(\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta, \mathsf{AC})$ as a ground AC-TRS $(\mathcal{F} \cup \mathcal{Q}, \Delta/\mathsf{AC})$, we can extend Proposition 1 as follows.

Lemma 4. *Given a ground AC-TRS $(\mathcal{F}, \mathcal{R}/\mathsf{AC})$ and tree languages L_1, L_2 over $\mathcal{F}$. If L_1 and L_2 are AC-recognizable tree languages, it is decidable whether there exist some s in L_1 and t in L_2 such that $s \to^*_{\mathcal{R}/\mathsf{AC}} t$.*

Proof. Suppose $L_1 = \mathcal{L}(\mathcal{A}/\mathsf{AC})$ and $L_2 = \mathcal{L}(\mathcal{B}/\mathsf{AC})$ where $\mathcal{A} = (\mathcal{F}, \mathcal{P}, \mathcal{P}_{fin}, \Delta_1)$ and $\mathcal{B} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta_2)$, such that $\mathcal{P} \cap \mathcal{Q} = \varnothing$. Here we assume without loss of generality that p, q are state symbols such that $\mathcal{P}_{fin} = \{\mathsf{p}\}$ and $\mathcal{Q}_{fin} = \{\mathsf{q}\}$. Then we obtain

$$\exists s \in L_1, \ \exists t \in L_2. \ \ s \to^*_{\mathcal{R}/\mathsf{AC}} t \qquad\qquad\qquad \cdots \text{(A)}$$

if and only if

$$\exists u, v \in \mathcal{T}(\mathcal{F}). \ \ u \to^*_{\mathcal{A}/\mathsf{AC}} \mathsf{p} \ \wedge \ v \to^*_{\mathcal{B}/\mathsf{AC}} \mathsf{q} \ \wedge \ u \to^*_{\mathcal{R}/\mathsf{AC}} v. \qquad \cdots \text{(B)}$$

Moreover, using Lemma 3 twice, (B) holds if and only if

$$\mathsf{p} \to^*_{(\mathcal{R} \cup \mathcal{A}^{-1} \cup \mathcal{B})/\mathsf{AC}} \mathsf{q}. \qquad\qquad\qquad\qquad \cdots \text{(C)}$$

Due to Proposition 1, we know (C) is decidable, and so is (A). $\qquad\square$

Given a term t, a predicate P and a binary relation $\to$ on terms, a question of whether there exists some s in $(\to^*)[\{t\}]$ with $P(s)$ is called *reachable property problem*. Here $P(s)$ can be replaced by the membership test $s \in [\![P]\!]$. Mayr and Rusinowitch showed in [17] that if P is a so-called state formula, this problem is decidable within their framework called *process rewrite systems*. In term rewriting setting, it corresponds to the following result:

Proposition 2. *Reachable property problem is decidable for ground AC-TRSs if $[\![P]\!]$ is a regular tree language.* $\qquad\square$

Due to Lemma 4 we know that $(\to^*_{\mathcal{R}/\mathsf{AC}})[L_1] \cap L_2 = \varnothing$ is a computable question, provided $\mathcal{R}/\mathsf{AC}$ is a ground AC-TRS and L_1, L_2 are AC-recognizable. Therefore the above proposition is generalized by our equational tree automata framework.

One should notice that even if L_1 and/or L_2 are regular, the above decidability holds. Note that regular tree languages are not AC-recognizable: Suppose $\mathcal{F}_{\mathsf{AC}} = \{\mathsf{f}\}$ and $L = \{\mathsf{f}(\mathsf{a}, \mathsf{b})\}$. The tree language L is regular, but it is not AC-recognizable (and not AC-regular), because every AC-TA $\mathcal{A}/\mathsf{AC}$ accepting $\mathsf{f}(\mathsf{a}, \mathsf{b})$ also accepts $\mathsf{f}(\mathsf{b}, \mathsf{a})$.

On the other hand, as easy instances of Lemma 4, we can show the positive answers to our open questions:

Corollary 2. *Emptiness problem for AC-TA is decidable.*

Proof. Let $\mathcal{A}/\mathsf{AC}$ be an AC-TA over the signature $\mathcal{F}$. In Lemma 4, by taking $\mathcal{R} = \varnothing$, $L_1 = \mathcal{L}(\mathcal{A}/\mathsf{AC})$ and $L_2 = \mathcal{T}(\mathcal{F})$, we know that $L_1 \neq \varnothing$ if and only if there exist $s \in L_1$ and $t \in L_2$ such that $s \to^*_{\mathcal{R}/\mathsf{AC}} t$. □

Corollary 3. *Intersection-emptiness problem for regular AC-TA is decidable.*

Proof. Along the same lines of the previous case, we take $\mathcal{R} = \varnothing$, $L_1 = \mathcal{L}(\mathcal{A}/\mathsf{AC})$ and $L_2 = \mathcal{L}(\mathcal{B}/\mathsf{AC})$ such that $\mathcal{A}/\mathsf{AC}$ and $\mathcal{B}/\mathsf{AC}$ are regular AC-TA. □

But an important remark for the above results is that Mayr and Rusinowitch did not actually show in [16] the proof of the general case of Proposition 1, i.e. there is no proof of decidability of the reachability problem for ground AC-TRS with *arbitrary many* AC-symbols. In other words, as far as the above proofs depend upon Proposition 1, the newly obtained result (e.g. Lemma 4) works only for the single-AC case.

So in the next section, by showing the original procedure, we generalize our results (Corollaries 2 and 3) together with doing estimation of the complexity of emptiness questions.

3 Complexity of AC-Tree Language Problems

Let us introduce a special notation for AC-tree languages: Suppose f is a function symbol in $\mathcal{F}_{\mathsf{AC}}$. An *$f$-block of a term t* is a non-empty maximal context in t consisting only of f. For notational convenience, if $t = C'[C[t_1, \ldots, t_n]]$ such that C is an f-block, we write $C'[C_f[\![t_1, \ldots, t_n]\!]]$. For instance, $g(f(a, f(b, c)))$ is represented as $g(C_f[\![a, b, c]\!])$.

First we state the basic property of the procedure in Fig. 2. This procedure is defined as in the way of Garey and Johnson [5].

Lemma 5. *At the step 2 in Fig. 2, for every $q \in \mathcal{Q}_2$, if $c \to^*_{\mathcal{S}/\mathsf{AC}(f)} q$, there exists a term $t \in \mathcal{T}(\mathcal{Q}_1 \cup \{f\})$ such that $c \to^*_{\mathcal{S}/\mathsf{AC}(f)} t \to^*_{\mathcal{R}/\mathsf{AC}(f)} q$.*

Proof. Let $\mathcal{R}_1 = \{c \to f(c, c)\}$ and $\mathcal{R}_2 = \{c \to q \mid q \in \mathcal{Q}_1\}$. Since c does not appear in the right-hand sides of rewrite rules of a ground TRS $\mathcal{R} \cup \mathcal{R}_2$, we can prove that $\to_{(\mathcal{R} \cup \mathcal{R}_2)/\mathsf{AC}(f)} \cdot \to_{\mathcal{R}_1/\mathsf{AC}(f)} \subseteq \to_{\mathcal{R}_1/\mathsf{AC}(f)} \cdot \to_{(\mathcal{R} \cup \mathcal{R}_2)/\mathsf{AC}(f)}$. By the same reason, $\to_{\mathcal{R}/\mathsf{AC}(f)} \cdot \to_{\mathcal{R}_2/\mathsf{AC}(f)} \subseteq \to_{\mathcal{R}_2/\mathsf{AC}(f)} \cdot \to_{\mathcal{R}/\mathsf{AC}(f)}$. Thus we obtain a rewrite sequence $c \to^*_{\mathcal{R}_1/\mathsf{AC}(f)} t_1 \to^*_{\mathcal{R}_2/\mathsf{AC}(f)} t_2 \to^*_{\mathcal{R}/\mathsf{AC}(f)} q$. Since rewrite rules in $\mathcal{R}$ are ground and do not contain c, the number of occurrences of c does not change in $t_2 \to^*_{\mathcal{R}/\mathsf{AC}(f)} q$. This implies $|t_2|_c = 0$, because $q \neq c$. Similarly, since $\mathcal{R}_1 \cup \mathcal{R}_2$ is ground and consists of function symbols in $\mathcal{Q}_1 \cup \{f\}$, we have $t_2 \in \mathcal{T}(\mathcal{Q}_1 \cup \{f\} \cup \{c\})$. Hence $t_2 \in \mathcal{T}(\mathcal{Q}_1 \cup \{f\})$. □

Next we show soundness of the main procedure.

Lemma 6. *For every AC-TA $\mathcal{A}/\mathsf{AC}$, a state q is accessible with $\mathcal{A}/\mathsf{AC}$ if and only if q is in $\mathcal{Q}_{acc}$.*
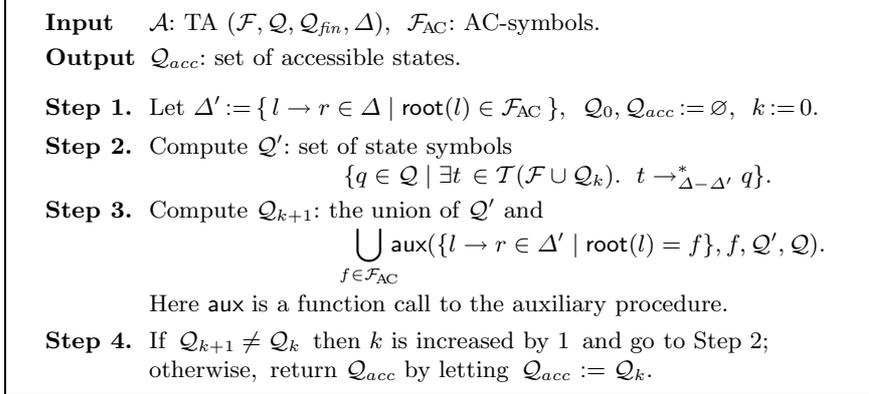
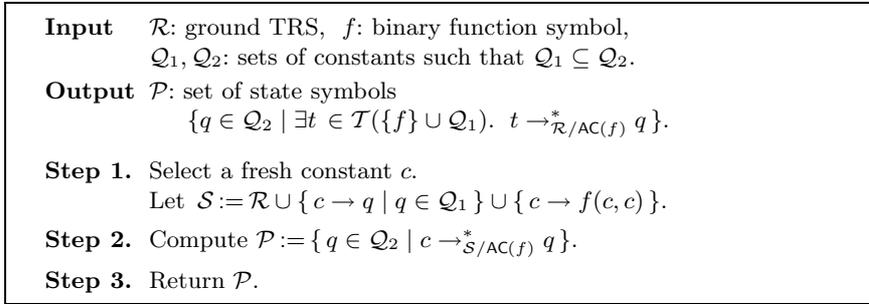**Fig. 1.** Procedure: STATE SYMBOLS ACCESSIBLE WITH AC-TA

Fig. 2. Auxiliary Procedure

Proof. Let $\mathcal{A} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$. Using the induction on the loop-counter k of the procedure in Fig. 1, we show that $q \in \mathcal{Q}_k$ for some k if and only if q is accessible with $\mathcal{A}/\mathsf{AC}$. First we prove the "if" part. The base case is trivial, because $\mathcal{Q}_0 = \varnothing$. For induction step, we suppose $q \in \mathcal{Q}_k$. We divide into the three cases as follows: (1) If $q \in \mathcal{Q}_{k'}$ with $k' < k$, the lemma holds by induction hypothesis. (2) If $q \notin \mathcal{Q}_{k'}$ for any $k' < k$ but $q \in \mathcal{Q}'$ at Step 2, the lemma also holds by induction hypothesis and the definition of Step 2. (3) If $q \notin \mathcal{Q}_{k'}$ for any $k' < k$ but $q \in \mathcal{Q}_k$ at Step 3 then $c \to^*_{\mathcal{S}/\mathsf{AC}(f)} q$ for some $f \in \mathcal{F}_{\mathsf{AC}}$. By Lemma 5, we have a term $t \in \mathcal{T}(\mathcal{Q}' \cup \{f\})$ such that $t \to^*_{\mathcal{R}/\mathsf{AC}(f)} q$ where $\mathcal{R} = \{l \to r \in \Delta \mid \mathsf{root}(l) = f\}$. From the previous cases, since $\mathcal{Q}'$ at Step 2 is a set of accessible states, so is q (as $t' \to^*_{\mathcal{A}/\mathsf{AC}} t$ for some $t' \in \mathcal{T}(\mathcal{F})$). For the "only if" part, we take a state $q \in \mathcal{Q}$ such that $t \to^*_{\mathcal{A}/\mathsf{AC}} q$ for some $t \in \mathcal{T}(\mathcal{F})$. By induction on the number of blocks in t, we show that q is in $\mathcal{Q}_k$ for some k. If there is no block in t, by letting $k = 0$ the lemma holds obviously. For induction step, we assume without loss of generality that $t = C'[s_1, \ldots, s_{i-1}, C_f[\![t_1, \ldots, t_n]\!], s_{i+1}, \ldots, s_m]$, where $C' \in \mathcal{C}(\mathcal{F} - \mathcal{F}_{\mathsf{AC}})$. C'

is possibly the empty context. By assumption, $t \to^*_{\mathcal{A}/\mathsf{AC}} C'[p_1, \ldots, p_m] \to^*_{\mathcal{A}} q$ and $C_f[\![t_1, \ldots, t_n]\!] \to^*_{\mathcal{A}/\mathsf{AC}} C_f[\![q_1, \ldots, q_n]\!] \to^*_{\mathcal{A}/\mathsf{AC}} p_i$ for some $p_1, \ldots, p_m, q_1, \ldots, q_n \in \mathcal{Q}$. By induction hypothesis, $q_j \in \mathcal{Q}_{k_j}$ for some k_j $(1 \leqslant j \leqslant n)$. Moreover, $p_j \in \mathcal{Q}_{k'_j}$ for some k'_j $(1 \leqslant j \leqslant m)$ if $j \neq i$. By definition of the procedure in Fig. 1, we obtain $p_i \in \mathcal{Q}_{k_{\max}+1}$ where $k_{\max} = \max\{k_j \mid 1 \leqslant j \leqslant n\}$. Hence, by letting $k'_{\max} = \max(\{k'_j \mid 1 \leqslant j \leqslant m \text{ and } j \neq i\} \cup \{k_{\max}+1\})$, $q \in \mathcal{Q}_{k'_{\max}}$ or $q \in \mathcal{Q}'$ at the $(k'_{\max}+1)$-th loop. □

Theorem 1. *Emptiness problem for AC-TA with arbitrary many AC-symbols is decidable.* □

The procedure in Fig. 1 needs at most $(|\mathcal{Q}|^2 \times |\mathcal{F}_{\mathrm{AC}}|)$-auxiliary function calls, and each computation is reachability test of a Petri net instance. Reachability problem for Petri nets is EXPSPACE-hard [15].

In contrast, the emptiness problem for *regular* AC-TA is solved in linear time, due to Lemma 2. From the similar observation, the complexity of membership problem can also be obtained.

Corollary 4. *Membership problem for regular AC-TA is NP-complete.*

Proof. We show that this problem is in NP. Let $\mathcal{A}/\mathsf{AC}$ be a regular AC-TA. From Lemma 2, a term t is accepted by $\mathcal{A}/\mathsf{AC}$ if and only if there exists a term t' accepted by $\mathcal{A}$ and $t' \sim_{\mathsf{AC}} t$. Thus the following procedure is sound:

(1) Guess a term t' such that $t' \sim_{\mathsf{AC}} t$.
(2) Check whether or not $t' \in \mathcal{L}(\mathcal{A})$.

Since the membership problem for regular TA is solved in linear time, the above algorithm is in NP. On the other hand, membership problem for commutative context-free grammar, which is known as an NP-hard problem [4], is reduced to this problem by assuming $\mathcal{F} = \{f\} \cup \Sigma$ with $\mathcal{F}_{\mathrm{AC}} = \{f\}$. Here Σ is the alphabet of a grammar. Hence the membership problem for regular AC-TA is NP-complete. □

On the other hand, because AC-recognizable tree languages are closed under intersection (Lemma 1), we can extend another decidability result.

Theorem 2. *Intersection-emptiness problem for regular AC-TA with arbitrary many AC-symbols is decidable.* □

4 Yet Negative Results on A-Tree Languages

In this section we discuss decidability of A-tree languages. In the previous paper we showed that emptiness problem for A-TA is *undecidable* in general. Using this result with the same proof argument of Lemma 4, we can prove the following undecidability.

Proposition 3 ([3]). *Reachability problem for ground A-TRS is undecidable.*

Proof. We use the following fact (Corollary 1, [20]): Emptiness problem is undecidable for A-TA. We take an A-TA $\mathcal{A}/\mathsf{A}$. Here we assume without loss of generality that $\mathcal{A} = (\mathcal{F}, \mathcal{Q}, \{\mathsf{q}\}, \Delta_1)$ like in the proof of Lemma 4. Define the TA $\mathcal{B} = (\mathcal{F}, \{\mathsf{p}\}, \{\mathsf{p}\}, \Delta_2)$ where $\Delta_2 = \{f(\mathsf{p}, \ldots, \mathsf{p}) \rightarrow \mathsf{p} \mid f \in \mathcal{F}\}$. Obviously, $\mathcal{L}(\mathcal{B}/\mathsf{A}) = \mathcal{T}(\mathcal{F})$. Similar to the proof of Lemma 3, we obtain $\rightarrow_{\mathcal{A}/\mathsf{A}} \cdot \rightarrow_{\mathcal{B}^{-1}/\mathsf{A}} \subseteq \rightarrow_{\mathcal{B}^{-1}/\mathsf{A}} \cdot \rightarrow_{\mathcal{A}/\mathsf{A}}$. This implies that $\mathsf{p} \rightarrow^*_{(\mathcal{A} \cup \mathcal{B}^{-1})/\mathsf{A}} \mathsf{q}$ if and only if $\mathsf{p} \rightarrow^*_{\mathcal{B}^{-1}/\mathsf{A}} t \rightarrow^*_{\mathcal{A}/\mathsf{A}} \mathsf{q}$ for some t. By definitions of $\mathcal{A}$ and $\mathcal{B}$, $t \in \mathcal{T}(\mathcal{F} \cup \{\mathsf{p}\})$, and moreover, $t \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q})$. Thus t is in $\mathcal{T}(\mathcal{F})$ if there exists. Since it is undecidable whether $t \rightarrow^*_{\mathcal{A}/\mathsf{A}} \mathsf{q}$ for some $t \in \mathcal{T}(\mathcal{F})$, so is $\mathsf{p} \rightarrow^*_{(\mathcal{A} \cup \mathcal{B}^{-1})/\mathsf{A}} \mathsf{q}$. $\qquad\square$

Other problems, such as subset and universality problems, are also undecidable for A-TA. However, decidability of the two problems for regular A-TA remains open in [20]. In the following part, we show these decidability results by reducing to the same *word* problems.

A *grammar* $\mathcal{G}$ is the 4-tuple $(\Sigma, \mathcal{Q}, \mathsf{q}_0, \Delta)$, whose components are the alphabet Σ, a finite set $\mathcal{Q}$ of state symbols such that $\Sigma \cap \mathcal{Q} = \varnothing$, an initial state $\mathsf{q}_0 (\in \mathcal{Q})$ and a finite set Δ of string rewrite rules (called *production rules*) in the form of $l \rightarrow r$ for some $l, r \in (\Sigma \cup \mathcal{Q})^+$. A grammar is called *context-free* if $l \in \mathcal{Q}$ for all rules $l \rightarrow r \in \Delta$. A regular grammar is a context-free grammar whose right-hand sides of rules are in $(\Sigma \cup \mathcal{Q})\Sigma^*$. A word generated by $\mathcal{G}$ is an element w in Σ^* such that $\mathsf{q}_0 \rightarrow^*_{\mathcal{G}} w$. Note that $\mathcal{G}$ does not generate the empty word ϵ. The language generated by $\mathcal{G}$, denoted by $\mathcal{L}(\mathcal{G})$, is a set of generated words.

A relationship between word languages and tree languages can be found in the literature, e.g. in [2, 6]. A study of leaf languages is one of the examples:

Proposition 4 ([6]). *The following two statements hold true:*
1. *For every CFG $\mathcal{G}$ there exists a regular TA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{G}) = \mathsf{leaf}(\mathcal{L}(\mathcal{A}))$.*
2. *For every regular TA $\mathcal{A}$ there exists a CFG $\mathcal{G}$ such that $\mathsf{leaf}(\mathcal{L}(\mathcal{A})) = \mathcal{L}(\mathcal{G})$.*
$\qquad\square$

A *leaf-word* $\mathsf{leaf}(t)$ of a term t over the signature $\mathcal{F}$ is a word over the alphabet $\mathcal{F}_0$ (a set of constants in $\mathcal{F}$), that is inductively defined as follows: $\mathsf{leaf}(t) = t$ if t is a constant; $\mathsf{leaf}(t) = \mathsf{leaf}(t_1) \ldots \mathsf{leaf}(t_n)$ if $t = f(t_1, \ldots, t_n)$ and $\mathsf{arity}(f) \geqslant 1$. We use the same notation leaf for the extension to tree languages, i.e. let L be a tree language over $\mathcal{F}$, then $\mathsf{leaf}(L) = \{\mathsf{leaf}(t) \mid t \in L\}$. We say $\mathsf{leaf}(L)$ is a *leaf-language* of L.

One should notice that the following statement is *not* true: If $\mathsf{leaf}(L)$ is CFG then L is a regular tree language. For instance, let us consider the following example. We take $\mathcal{G}_1 = (\{\mathsf{a}, \mathsf{b}\}, \{\alpha, \beta, \gamma\}, \gamma, \Delta_1)$ where Δ_1:

$$\gamma \rightarrow \mathsf{a}\beta \qquad \alpha \rightarrow \mathsf{a} \qquad \alpha \rightarrow \mathsf{b}\alpha\alpha \qquad \beta \rightarrow \mathsf{b} \qquad \beta \rightarrow \mathsf{a}\beta\beta$$
$$\gamma \rightarrow \mathsf{b}\alpha \qquad \alpha \rightarrow \mathsf{a}\gamma \qquad \qquad\qquad \beta \rightarrow \mathsf{b}\gamma$$

By induction on the length of words, we can prove that for every non-empty word w over the alphabet $\{\mathsf{a}, \mathsf{b}\}$,

$\alpha \rightarrow^*_{\mathcal{G}_1} w$ if and only if $|w|_{\mathsf{a}} = |w|_{\mathsf{b}} + 1$,
$\beta \rightarrow^*_{\mathcal{G}_1} w$ if and only if $|w|_{\mathsf{b}} = |w|_{\mathsf{a}} + 1$,
$\gamma \rightarrow^*_{\mathcal{G}_1} w$ if and only if $|w|_{\mathsf{a}} = |w|_{\mathsf{b}}$.

This implies $\mathcal{L}(\mathcal{G}_1) = \{w \mid |w|_\mathsf{a} = |w|_\mathsf{b}\}$, i.e. $\mathcal{G}_1$ generates the set of words w such that the number of occurrences of a in w is the same as the number of occurrences of b in w. On the other hand, we consider a tree language over a signature $\mathcal{F}$ such that $\mathcal{F}_0 = \{\mathsf{a}, \mathsf{b}\}$. If $\mathcal{F}$ contains at least a function symbol f with $\mathsf{arity}(f) \geqslant 2$, a tree language $L = \{t \mid |t|_\mathsf{a} = |t|_\mathsf{b}\}$ is no longer a regular tree language, due to PUMPING LEMMA.

Now we discuss the relationship between word languages and equational tree languages. The following properties are obtained as an easy observation of leaf-operation.

Lemma 7. *Given an arbitrary signature $\mathcal{F}$. For all terms $s, t \in \mathcal{T}(\mathcal{F})$ and tree languages L over $\mathcal{F}$,*

1. *$\mathsf{leaf}(s) = \mathsf{leaf}(t)$ if $s \sim_\mathsf{A} t$. The reverse holds if $\mathcal{F} = \mathcal{F}_0 \cup \{f\}$ and $\mathcal{F}_{\mathrm{AC}} = \{f\}$.*
2. *$\mathsf{leaf}(L) = \mathsf{leaf}(\mathsf{A}(L))$.*

Suppose $\mathcal{F} = \mathcal{F}_0 \cup \{f\}$ and $\mathcal{F}_{\mathrm{AC}} = \{f\}$. If L_1, L_2 are tree languages over $\mathcal{F}$, the following statements hold true:

3. *$\mathsf{leaf}(L_1 \cup L_2) = \mathsf{leaf}(L_1) \cup \mathsf{leaf}(L_2)$,*
4. *$\mathsf{leaf}(\mathsf{A}(L_1) \cap \mathsf{A}(L_2)) = \mathsf{leaf}(L_1) \cap \mathsf{leaf}(L_2)$,*
5. *$\mathsf{A}(L_1) \subseteq \mathsf{A}(L_2)$ if and only if $\mathsf{leaf}(L_1) \subseteq \mathsf{leaf}(L_2)$.*

Proof. To prove the property 1, we observe that

$$\mathsf{leaf}(C[\mathsf{f}(\mathsf{f}(t_1, t_2), t_3)]) = w_1 \,\mathsf{leaf}(t_1)\,\mathsf{leaf}(t_2)\,\mathsf{leaf}(t_3)\, w_2 = \mathsf{leaf}(C[\mathsf{f}(t_1, \mathsf{f}(t_2, t_3))])$$

for some $w_1, w_2 \in \mathcal{F}_0^*$. It can be proved by the structural induction of C. This implies that if $s \sim_\mathsf{A} t$ then $\mathsf{leaf}(s) = \mathsf{leaf}(t)$. For the reverse, we define the mapping tree as follows: $\mathsf{tree}(w) = a$ if $w \in \mathcal{F}_0$; $\mathsf{tree}(w) = \mathsf{f}(a, \mathsf{tree}(w'))$ if $w = a\,w'$ for some $a \in \mathcal{F}_0$ and $w' \in \mathcal{F}_0^+$. Then it suffices to show that for all u in $\mathcal{T}(\mathcal{F})$, $\mathsf{tree}(\mathsf{leaf}(u)) \sim_\mathcal{A} u$. It can be proved by the structural induction of u.

The properties 2 and 3 are easy. Note that union ($\cup$) is distributive over set comprehension, i.e. $\{\mathsf{leaf}(t) \mid t \in L_1 \cup L_2\} = \{\mathsf{leaf}(t) \mid t \in L_1\} \cup \{\mathsf{leaf}(t) \mid t \in L_2\}$.

Next we show the property 4. We suppose $w \in \mathsf{leaf}(\mathsf{A}(L_1) \cap \mathsf{A}(L_2))$. By definition, there exists $t \in \mathsf{A}(L_1) \cap \mathsf{A}(L_2)$ such that $w = \mathsf{leaf}(t)$. Moreover, $\mathsf{leaf}(t) \in \mathsf{leaf}(\mathsf{A}(L_1))$ and $\mathsf{leaf}(t) \in \mathsf{leaf}(\mathsf{A}(L_2))$. Thus $w \in \mathsf{leaf}(\mathsf{A}(L_1)) \cap \mathsf{leaf}(\mathsf{A}(L_2))$. Due to the property 2, $w \in \mathsf{leaf}(L_1) \cap \mathsf{leaf}(L_2)$. For the reverse inclusion, we suppose $w \in \mathsf{leaf}(\mathsf{A}(L_1)) \cap \mathsf{leaf}(\mathsf{A}(L_2))$. Note that $\mathsf{leaf}(L_1) = \mathsf{leaf}(\mathsf{A}(L_1))$ and $\mathsf{leaf}(L_2) = \mathsf{leaf}(\mathsf{A}(L_2))$. Then there exist $t_1 \in \mathsf{A}(L_1)$ and $t_2 \in \mathsf{A}(L_2)$ such that $w = \mathsf{leaf}(t_1)$ and $w = \mathsf{leaf}(t_2)$. Due to the property 1, $t_1 \sim_\mathsf{A} t_2$, and thus, $t_1 \in \mathsf{A}(L_2)$ (and $t_2 \in \mathsf{A}(L_1)$ as well). Hence $t_1 \in \mathsf{A}(L_1) \cap \mathsf{A}(L_2)$. This implies $w \in \mathsf{leaf}(\mathsf{A}(L_1) \cap \mathsf{A}(L_2))$.

Finally we show the property 5. The "only if" part is trivial. To show the reverse, we suppose $t \in \mathsf{A}(L_1)$. Then $\mathsf{leaf}(t) \in \mathsf{leaf}(L_1)$ (as $\mathsf{leaf}(t) \in \mathsf{leaf}(\mathsf{A}(L_1))$), and thus, $\mathsf{leaf}(t) \in \mathsf{leaf}(L_2)$. This implies that, due to the property 1, there exists $s \in L_2$ such that $s \sim_\mathsf{A} t$. Hence $t \in \mathsf{A}(L_2)$. $\qquad\square$

One should note that $\mathsf{leaf}(L_1 \cap L_2) \neq \mathsf{leaf}(L_1) \cap \mathsf{leaf}(L_2)$. For example, suppose $L_1 = \{\mathsf{f}(\mathsf{f}(\mathsf{a},\mathsf{b}),\mathsf{c})\}$ and $L_2 = \{\mathsf{f}(\mathsf{a},\mathsf{f}(\mathsf{b},\mathsf{c}))\}$ such that $\mathcal{F} = \{\mathsf{f},\mathsf{a},\mathsf{b},\mathsf{c}\}$ and $\mathcal{F}_\mathsf{A} = \{\mathsf{f}\}$. Then $\mathsf{leaf}(L_1 \cap L_2) = \varnothing$, but $\mathsf{leaf}(L_1) \cap \mathsf{leaf}(L_2) = \{\mathsf{abc}\}$. Furthermore, we see that $\mathsf{leaf}(L_1) \subseteq \mathsf{leaf}(L_2)$ does not imply $L_1 \subseteq L_2$.

We show an extension of Proposition 4 (1). In the extension, *maximality* of tree languages refines the relationship to word languages.

Definition 1. *A tree languages L over $\mathcal{F}$ is* maximal *for a word language W if for all terms t in $\mathcal{T}(\mathcal{F})$, $\mathsf{leaf}(t) \in W$ if and only if $t \in L$. An ETA $\mathcal{A}/\mathcal{E}$ is* maximal *for W if $\mathcal{L}(\mathcal{A}/\mathcal{E})$ is maximal for W.*

Lemma 8. *Given a CFG $\mathcal{G}$ over the alphabet Σ. Then there exists the associated regular A-TA $\mathcal{A}_\mathcal{G}/\mathsf{A}$ that is maximal for $\mathcal{L}(\mathcal{G})$.*

Proof. Let $\mathcal{G} = (\Sigma, \mathcal{Q}, \mathsf{q}_0, \Delta)$. Suppose Δ is in Chomsky Normal Form. We take $\mathcal{F} = \Sigma \cup \{\mathsf{f}\}$ and $\mathcal{F}_\mathsf{A} = \{\mathsf{f}\}$. Define the regular A-TA $\mathcal{A}_\mathcal{G}/\mathsf{A}$ as follows: $\mathcal{A}_\mathcal{G} = (\mathcal{F}, \mathcal{Q}, \{\mathsf{q}_0\}, \Delta')$ where $\Delta' = \{\mathsf{f}(q_1, q_2) \to q \mid q \to q_1\,q_2 \in \Delta\} \cup \{a \to q \mid q \to a \in \Delta\}$. It is easy to show that for every (non-empty) word $w \in \Sigma^+$, $\mathsf{q}_0 \to_\mathcal{G}^* w$ if and only if $\mathsf{tree}(w) \to_{\mathcal{A}_\mathcal{G}/\mathsf{A}}^* \mathsf{q}_0$. Here tree is the mapping defined in the previous proof. For maximality we use Lemma 7 (1) and the following property: $\mathsf{leaf}(\mathsf{tree}(w)) = w$ for every $w \in \Sigma$. Then we can prove that for every term $t \in \mathcal{T}(\Sigma \cup \{\mathsf{f}\})$ and word $w \in \Sigma^*$, $\mathsf{leaf}(t) = w$ if and only if $t \sim_\mathsf{A} \mathsf{tree}(w)$. Thus, if $\mathsf{leaf}(t) \in \mathcal{L}(\mathcal{G})$ then $t \sim_\mathsf{A} \mathsf{tree}(\mathsf{leaf}(t))$ and $\mathsf{tree}(\mathsf{leaf}(t)) \to_{\mathcal{A}_\mathcal{G}/\mathsf{A}}^* \mathsf{q}_0$. Hence $t \in \mathcal{L}(\mathcal{A}_\mathcal{G}/\mathsf{A})$. $\square$

We consider the CFG $\mathcal{G}_2 = (\{\mathsf{a},\mathsf{b}\}, \{\alpha, \alpha', \beta, \beta', \gamma, \delta, \xi\}, \gamma, \Delta_2)$ where Δ_2:

$$\begin{array}{lllllll}
\gamma \to \delta\,\beta & \alpha \to \mathsf{a} & \alpha \to \xi\,\alpha' & \beta \to \mathsf{b} & \beta \to \delta\,\beta' & \delta \to \mathsf{a} \\
\gamma \to \xi\,\alpha & \alpha \to \delta\,\gamma & \alpha' \to \alpha\,\alpha & \beta \to \xi\,\gamma & \beta' \to \beta\,\beta & \xi \to \mathsf{b}
\end{array}$$

Note that $\mathcal{G}_2$ is in Chomsky Normal Form, and $\mathcal{L}(\mathcal{G}_2) = \{w \mid |w|_\mathsf{a} = |w|_\mathsf{b}\}$ because $\mathcal{G}_2$ is essentially the same as $\mathcal{G}_1$. Then the associated A-TA $\mathcal{A}_{\mathcal{G}_2}/\mathsf{A}$ over the signature $\mathcal{F} = \{\mathsf{f},\mathsf{a},\mathsf{b}\}$ and $\mathcal{F}_\mathsf{A} = \{\mathsf{f}\}$ recognizes a tree language maximal for $\mathcal{L}(\mathcal{G}_2)$ and for $\mathcal{L}(\mathcal{G}_1)$.

Next we extend Proposition 4 (2) as in the previous way.

Lemma 9. *Given an regular A-TA $\mathcal{A}/\mathsf{A}$ over the signature $\mathcal{F}$. Then there exists the associated CFG $\mathcal{G}_{\mathcal{A}/\mathsf{A}}$ such that $\mathcal{L}(\mathcal{G}_{\mathcal{A}/\mathsf{A}}) = \mathsf{leaf}(\mathcal{L}(\mathcal{A}/\mathsf{A}))$.*

Proof. From Proposition 4 (2), for the language $\mathsf{leaf}(\mathcal{L}(\mathcal{A}))$ there exists a CFG $\mathcal{G}$ such that $\mathcal{L}(\mathcal{G}) = \mathsf{leaf}(\mathcal{L}(\mathcal{A}))$. On the other hand, from Lemma 7 (2), $\mathsf{leaf}(\mathcal{L}(\mathcal{A})) = \mathsf{leaf}(\mathsf{A}(\mathcal{L}(\mathcal{A})))$. Since $\mathcal{L}(\mathcal{A}/\mathsf{A}) = \mathsf{A}(\mathcal{L}(\mathcal{A}))$ by Lemma 2, we obtain $\mathsf{leaf}(\mathcal{L}(\mathcal{A})) = \mathsf{leaf}(\mathcal{L}(\mathcal{A}/\mathsf{A}))$. By letting $\mathcal{G}_{\mathcal{A}/\mathsf{A}} = \mathcal{G}$, it satisfies that $\mathcal{L}(\mathcal{G}_{\mathcal{A}/\mathsf{A}}) = \mathsf{leaf}(\mathcal{L}(\mathcal{A}/\mathsf{A}))$. $\square$

In this case, $\mathcal{A}/\mathsf{A}$ is not always maximal. For instance, we take an A-TA $\mathcal{A}/\mathsf{A}$ such that $\mathcal{L}(\mathcal{A}/\mathsf{A}) = \{\mathsf{a}\}$ over the signature $\{\mathsf{f},\mathsf{a}\}$. Obviously there exists a CFG $\mathcal{G}_{\mathcal{A}/\mathsf{A}}$ such that $\mathcal{L}(\mathcal{G}_{\mathcal{A}/\mathsf{A}}) = \{\mathsf{a}\}$. But then, $\mathcal{A}/\mathsf{A}$ is not maximal for $\mathcal{L}(\mathcal{G}_{\mathcal{A}/\mathsf{A}})$, because $\mathsf{f}(\mathsf{a})$ is not in $\mathcal{L}(\mathcal{A}/\mathsf{A})$.

Corollary 5. *The following two statements hold true:*

1. *For every CFG $\mathcal{G}$ there exists a regular A-TA $\mathcal{A}/\mathsf{A}$ such that $\mathcal{L}(\mathcal{A}/\mathsf{A})$ is a maximal tree language for $\mathcal{L}(\mathcal{G})$.*
2. *For every regular A-TA $\mathcal{A}/\mathsf{A}$ there exists a CFG $\mathcal{G}$ such that $\mathsf{leaf}(\mathcal{L}(\mathcal{A}/\mathsf{A}))$ $= \mathcal{L}(\mathcal{G})$. In case the signature $\mathcal{F} = \mathcal{F}_0 \cup \{\mathsf{f}\}$ and $\mathcal{F}_\mathsf{A} = \{\mathsf{f}\}$, $\mathcal{L}(\mathcal{A}/\mathsf{A})$ is a maximal tree language for $\mathcal{L}(\mathcal{G})$.*

Proof. An immediate consequence of Lemmata 8 and 9, except the case that $\mathcal{F} = \mathcal{F}_0 \cup \{\mathsf{f}\}$ and $\mathcal{F}_\mathsf{A} = \{\mathsf{f}\}$ in the latter statement. In the particular case, it holds that $\mathsf{tree}(\mathsf{leaf}(s)) \sim_\mathsf{A} s$ for any s in $\mathcal{T}(\mathcal{F})$. Thus, if $\mathsf{leaf}(t) \in \mathcal{L}(\mathcal{G})$, there exists $s \in \mathcal{L}(\mathcal{A}/\mathsf{A})$ such that $\mathsf{leaf}(s) = \mathsf{leaf}(t)$. Moreover, $s \sim_\mathsf{A} t$ by Lemma 7 (1). Hence $t \in \mathcal{L}(\mathcal{A}/\mathsf{A})$. □

Thus A-regular tree languages inherits the negative results of context-free languages.

Theorem 3. *A-regular tree languages are not closed under intersection or complement.*

Proof. Given CFGs $\mathcal{G}_1$ and $\mathcal{G}_2$ over the alphabet Σ. Due to Lemma 8, there are regular A-TA $\mathcal{A}/\mathsf{A}$ and $\mathcal{B}/\mathsf{A}$ over $\mathcal{F}$, where $\mathcal{F} = \Sigma \cup \{\mathsf{f}\}$ and $\mathcal{F}_\mathsf{A} = \{\mathsf{f}\}$, such that $\mathcal{L}(\mathcal{G}_1) = \mathsf{leaf}(\mathcal{L}(\mathcal{A}/\mathsf{A}))$ and $\mathcal{L}(\mathcal{G}_2) = \mathsf{leaf}(\mathcal{L}(\mathcal{B}/\mathsf{A}))$, respectively. Suppose to the contradiction that A-regular tree languages are closed under intersection. Then there exists a regular A-TA $\mathcal{C}/\mathsf{A}$ over $\mathcal{F}$ such that $\mathcal{L}(\mathcal{C}/\mathsf{A}) = \mathcal{L}(\mathcal{A}/\mathsf{A}) \cap \mathcal{L}(\mathcal{B}/\mathsf{A})$. Moreover, due to Lemma 9, there exists a CFG $\mathcal{G}_3$ such that $\mathcal{L}(\mathcal{G}_3) = \mathsf{leaf}(\mathcal{L}(\mathcal{C}/\mathsf{A}))$. From Lemma 7 (2) and (4), we obtain $\mathcal{L}(\mathcal{G}_3) = \mathsf{leaf}(\mathcal{L}(\mathcal{A}/\mathsf{A})) \cap \mathsf{leaf}(\mathcal{L}(\mathcal{B}/\mathsf{A})) = \mathcal{L}(\mathcal{G}_1) \cap \mathcal{L}(\mathcal{G}_2)$. However, it contradicts to the fact that context-free languages are not closed under intersection. Hence A-regular tree languages are not closed under intersection.

To show not being closed under complement, we use the previous fact together with De Morgan Law. □

Theorem 4. *The following questions are undecidable in A-regular tree languages: Given $\mathcal{A}/\mathsf{A}$ and $\mathcal{B}/\mathsf{A}$ are regular A-TA over the signature $\mathcal{F}$, then*

- $\mathcal{L}(\mathcal{A}/\mathsf{A}) \subseteq \mathcal{L}(\mathcal{B}/\mathsf{A})$?   *(subset)*
- $\mathcal{L}(\mathcal{A}/\mathsf{A}) = \mathcal{T}(\mathcal{F})$?   *(universarity)*
- $\mathcal{L}(\mathcal{A}/\mathsf{A}) \cap \mathcal{L}(\mathcal{B}/\mathsf{A}) = \varnothing$?   *(intersection-emptiness)*

Proof. Given CFGs $\mathcal{G}_1$ and $\mathcal{G}_2$ over the alphabet Σ. Due to Lemma 8, there exist regular A-TA $\mathcal{A}/\mathsf{A}$ and $\mathcal{B}/\mathsf{A}$ over $\mathcal{F}$ such that $\mathcal{L}(\mathcal{G}_1) = \mathsf{leaf}(\mathcal{L}(\mathcal{A}/\mathsf{A}))$ and $\mathcal{L}(\mathcal{G}_2) = \mathsf{leaf}(\mathcal{L}(\mathcal{B}/\mathsf{A}))$. Here the signature $\mathcal{F} = \Sigma \cup \{\mathsf{f}\}$ and $\mathcal{F}_\mathsf{A} = \{\mathsf{f}\}$. Due to Lemma 7 (2) and (5), we know that $\mathcal{L}(\mathcal{A}/\mathsf{A}) \subseteq \mathcal{L}(\mathcal{B}/\mathsf{A})$ if and only if $\mathsf{leaf}(\mathcal{L}(\mathcal{A}/\mathsf{A})) \subseteq \mathsf{leaf}(\mathcal{L}(\mathcal{B}/\mathsf{A}))$, because $\mathcal{L}(\mathcal{A}/\mathsf{A}) = \mathsf{A}(\mathcal{L}(\mathcal{A}))$ and $\mathcal{L}(\mathcal{B}/\mathsf{A}) = \mathsf{A}(\mathcal{L}(\mathcal{B}))$, respectively. This implies the question if $\mathcal{L}(\mathcal{A}/\mathsf{A}) \subseteq \mathcal{L}(\mathcal{B}/\mathsf{A})$ is undecidable, because the question if $\mathcal{L}(\mathcal{G}_1) \subseteq \mathcal{L}(\mathcal{G}_2)$ is undecidable [8]. Similarly, using Lemma 7 we can prove that

- $\mathcal{L}(\mathcal{A}/\mathsf{A}) = \mathcal{T}(\mathcal{F})$ if and only if $\mathsf{leaf}(\mathcal{L}(\mathcal{A}/\mathsf{A})) = \mathsf{leaf}(\mathcal{T}(\mathcal{F}))$,

		C	A	AC
$\mathcal{L}(\mathcal{A}/\mathcal{E}) = \varnothing$?	regular	✓	✓	✓
	non-regular		×	
$\mathcal{L}(\mathcal{A}/\mathcal{E}) \subseteq \mathcal{L}(\mathcal{B}/\mathcal{E})$?	regular	✓	×	(✓)
	non-regular			?
$\mathcal{L}(\mathcal{A}/\mathcal{E}) = \mathcal{T}(\mathcal{F})$?	regular	✓	×	(✓)
	non-regular			?
$\mathcal{L}(\mathcal{A}/\mathcal{E}) \cap \mathcal{L}(\mathcal{B}/\mathcal{E}) = \varnothing$?	regular	✓	×	✓
	non-regular			

		C	A	AC
closed under $\cup$	regular	✓	✓	✓
	non-regular			
closed under $\cap$	regular	✓	×	(✓)
	non-regular		✓	✓
closed under $\overline{(\ )}$	regular	✓	×	(✓)
	non-regular		?	?

Fig. 3. Some decidability results and closure properties

- $\mathcal{L}(\mathcal{A}/\mathsf{A}) \cap \mathcal{L}(\mathcal{B}/\mathsf{A}) = \varnothing$ if and only if $\mathsf{leaf}(\mathcal{L}(\mathcal{A}/\mathsf{A})) \cap \mathsf{leaf}(\mathcal{L}(\mathcal{B}/\mathsf{A})) = \varnothing$.

Hence the other two questions are also undecidable, because universality and intersection-emptiness problems for context-free languages are undecidable. ☐

Additionally, we can prove that equivalence problem for regular A-TA is undecidable: We take two regular A-TA $\mathcal{A}/\mathsf{A}, \mathcal{B}/\mathsf{A}$. The subset problem $\mathcal{L}(\mathcal{A}/\mathsf{A}) \subseteq \mathcal{L}(\mathcal{B}/\mathsf{A})$ is represented as $\mathcal{L}(\mathcal{A}/\mathsf{A}) \cup \mathcal{L}(\mathcal{B}/\mathsf{A}) = \mathcal{L}(\mathcal{B}/\mathsf{A})$. By Lemma 2, we have $\mathcal{L}(\mathcal{A}/\mathsf{A}) \cup \mathcal{L}(\mathcal{B}/\mathsf{A}) = \mathsf{A}(\mathcal{L}(\mathcal{A})) \cup \mathsf{A}(\mathcal{L}(\mathcal{B}))$, and then, $\mathsf{A}(\mathcal{L}(\mathcal{A})) \cup \mathsf{A}(\mathcal{L}(\mathcal{B})) = \mathsf{A}(\mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{B}))$. The A-congruence closure of a regular tree language $\mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{B})$ is A-regular, i.e. there is a regular A-TA $\mathcal{C}/\mathsf{A}$ such that $\mathcal{L}(\mathcal{C}/\mathsf{A}) = \mathsf{A}(\mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{B}))$. This implies that $\mathcal{L}(\mathcal{A}/\mathsf{A}) \subseteq \mathcal{L}(\mathcal{B}/\mathsf{A})$ if and only if $\mathcal{L}(\mathcal{C}/\mathsf{A}) = \mathcal{L}(\mathcal{B}/\mathsf{A})$. Since the former question is undecidable, so is the equivalence $\mathcal{L}(\mathcal{C}/\mathsf{A}) = \mathcal{L}(\mathcal{B}/\mathsf{A})$.

5 Concluding Remarks

In the paper we have shown decidability results and closure properties of A- and AC-tree language. New results on the decidability (Theorems 1, 2 and 4) and closure properties (Theorem 3) are the solutions to the questions remaining

open in our previous paper [20]. Using the following Parikh's result (Theorem 2 in [21]; the same result also in [4]), we can show a partial solution to the question of whether AC-regular tree languages are closed under intersection and complement.

Lemma 10. *Permutation closures of context-free languages are closed under boolean operations (union, intersection and complement).* □

Corollary 6. *Every AC-regular tree languages over the signature $\mathcal{F} = \{f\} \cup \mathcal{F}_0$ and $\mathcal{F}_{\mathrm{AC}} = \{f\}$ is closed under boolean operations.*

Proof. We show closedness under union and intersection below. Let $\mathcal{A}/\mathsf{AC}$, $\mathcal{B}/\mathsf{AC}$ be regular AC-TA over $\mathcal{F}$. By Proposition 4 (2), there exists a context-free grammar $\mathcal{G}_1$ and $\mathcal{G}_2$ such that $\mathsf{leaf}(\mathcal{L}(\mathcal{A})) = \mathcal{L}(\mathcal{G}_1)$ and $\mathsf{leaf}(\mathcal{L}(\mathcal{B})) = \mathcal{L}(\mathcal{G}_2)$. Due to Lemma 2, we have $\mathsf{leaf}(\mathcal{L}(\mathcal{A}/\mathsf{AC})) = \mathsf{leaf}(\mathsf{AC}(\mathcal{L}(\mathcal{A})))$. Moreover, by assumption of $\mathcal{F}$, $\mathsf{leaf}(\mathsf{AC}(\mathcal{L}(\mathcal{A}))) = \mathsf{perm}(\mathsf{leaf}(\mathcal{L}(\mathcal{A})))$. Thus $\mathsf{leaf}(\mathcal{L}(\mathcal{A}/\mathsf{AC})) = \mathsf{perm}(\mathcal{L}(\mathcal{G}_1))$. The same property holds for $\mathcal{B}/\mathsf{AC}$. By Lemma 10, there exists a context-free grammar $\mathcal{G}_R$ such that $\mathsf{perm}(\mathcal{L}(\mathcal{G}_R)) = \mathsf{perm}(\mathcal{L}(\mathcal{G}_1)) \; R \; \mathsf{perm}(\mathcal{L}(\mathcal{G}_2))$ for each operation $R \in \{\cup, \cap\}$. By Proposition 4 (1), we obtain a TA $\mathcal{C}_R$ such that $\mathsf{leaf}(\mathcal{L}(\mathcal{C}_R)) = \mathcal{L}(\mathcal{G}_R)$, and thus, $\mathsf{leaf}(\mathcal{L}(\mathcal{C}_R/\mathsf{AC})) = \mathsf{perm}(\mathcal{L}(\mathcal{G}_R))$. Therefore $\mathcal{L}(\mathcal{C}_R/\mathsf{AC}) = \mathcal{L}(\mathcal{A}/\mathsf{AC}) \; R \; \mathcal{L}(\mathcal{B}/\mathsf{AC})$ over $\mathcal{F} = \{f\} \cup \mathcal{F}_0$ with $\mathcal{F}_{\mathrm{AC}} = \{f\}$. □

Due to the above corollary together with decidability of emptiness problem for AC-TA (Corollary 2), the subset and universality problems are decidable in this particular case.

On the other hand, from the recent study about *multiset grammars* [11], we know that regular and non-regular AC-tree automata correspond to multiset context-free grammars and multiset monotone grammars, respectively. More precisely, the relationships like in Corollary 5 hold for them. This implies the expressive power of non-regular AC-tree automata strictly subsumes the regular case, which is also the answer to an open question of our previous paper.

We summarize the decidability results and closure properties of C-, A- and AC-tree languages in Fig. 3. New results are indicated by dotted squares. All the results of C-tree languages are easily obtained, because C-TA are essentially the same as regular TA (Lemma 3, [20]). In the figure, the check mark ✓ means "positive" and the cross × is "negative". The question mark ? means "open". If the same result holds in both regular and non-regular cases, it is represented by a single mark in a large column. A positive result proved only in a special case, e.g. of $\mathcal{F} = \{f\} \cup \mathcal{F}_0$ with $\mathcal{F}_{\mathrm{AC}} = \{f\}$, is denoted by the check mark with round brackets.

References

1. B. Bogaert and S. Tison: *Equality and Disequality constraints on Direct Subterms in Tree Automata*, Proc. of 9th STACS, Cachan (France), LNCS 577, pp. 161–171 1992.

2. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison and M. Tommasi: *Tree Automata Techniques and Applications*, draft, 1999. Available on `http://www.grappa.univ-lille3.fr/tata/`

3. A. Deruyver and R. Gilleron: *The Reachability Problem for Ground TRS and Some Extensions*, Proc. of 14th CAAP, Barcelona (Spain), LNCS 351, pp. 227–243, 1989.

4. J. Esparza: *Petri Nets, Commutative Context-Free Grammars, and Basic Parallel Processes*, Fundamenta Informaticae 31(1), pp. 13–25, 1997.

5. M.R. Garey and D.S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman & Company, New York, 1979.

6. F. Gécseg and M. Steinby: *Tree Automata*, Akadémiai Kiadó, Budapest, 1984.

7. T. Genet and F. Klay: *Rewriting for Cryptographic Protocol Verification*, Proc. of 17th CADE, Pittsburgh (PA), LNCS 1831, pp. 271–290, 2000.

8. J.E. Hopcroft and J.D. Ullman: *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing Company, 1979.

9. H. Hosoya, J. Vouillon and B.C. Pierce: *Regular Expression Types for XML*, Proc. of 5th ICFP, Montreal (Canada), SIGPLAN Notices 35(9), pp. 11–22, 2000.

10. Y. Kaji, T. Fujiwara and T. Kasami: *Solving a Unification Problem under Constrained Substitutions Using Tree Automata*, JSC 23(1), pp. 79–117, 1997.

11. M. Kudlek and V. Mitrana: *Normal Forms of Grammars, Finite Automata, Abstract Families, and Closure Properties of Macrosets*, Multiset Processing, LNCS 2235, pp. 135–146, 2001.

12. S. La Torre and M. Napoli: *Timed Tree Automata with an Application to Temporal Logic*, Acta Informatica 38(2), pp. 89–116, 2001.

13. D. Lugiez: *A Good Class of Tree Automata and Application to Inductive Theorem Proving*, Proc. of 25th ICALP, Aalborg (Denmark), LNCS 1443, pp. 409–420, 1998.

14. D. Lugiez and J.L. Moysset: *Tree Automata Help One to Solve Equational Formulae in AC-Theories*, JSC 18(4), pp. 297–318, 1994.

15. E.W. Mayr: *An Algorithm for the General Petri Net Reachability Problem*, SIAM J. Comput. 13(3), pp. 441–460, 1984.

16. R. Mayr and M. Rusinowitch: *Reachability is Decidable for Ground AC Rewrite Systems*, Proc. of 3rd INFINITY, Aalborg (Denmark), 1998. Draft available from `http://www.informatik.uni-freiburg.de/~mayrri/ac.ps`

17. R. Mayr and M. Rusinowitch: *Process Rewrite Systems*, Information and Computation 156, pp. 264–286, 1999.

18. J. Millen and V. Shmatikov: *Constraint Solving for Bounded-Process Cryptographic Protocol Analysis*, Proc. of 8th CCS, Philadelphia (PA), pp. 166–175, 2001.

19. D. Monniaux: *Abstracting Cryptographic Protocols with Tree Automata*, Proc. of 6th SAS, Venice (Italy), LNCS 1694, pp. 149–163, 1999.

20. H. Ohsaki: *Beyond Regularity: Equational Tree Automata for Associative and Commutative Theories*, Proc. of 15th CSL, Paris (France), LNCS 2142, pp. 539–553, 2001.

21. R.J. Parikh: *On Context-Free Languages*, JACM 13(4), pp. 570–581, 1966.

22. X. Rival and J. Goubault-Larrecq: *Experiments with Finite Tree Automata in Coq*, Proc. of 14th TPHOLs, Edinburgh (Scotland), LNCS 2152, pp. 362–377, 2001.

23. B. Schneier: *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Second Edition, John Wiley & Sons, 1996.

24. H. Seki, T. Takai, Y. Fujinaka and Y. Kaji: *Layered Transducing Term Rewriting System and Its Recognizability Preserving Property*, Proc. of 13th RTA, Copenhagen (Denmark), 2002. To appear in LNCS.

25. T. Takai, Y. Kaji and H. Seki: *Right-Linear Finite-Path Overlapping Term Rewriting Systems Effectively Preserve Recognizability*, Proc. of 11th RTA, Norwich (UK), LNCS 1833, pp. 246–260, 2000.