

Recognizing Boolean Closed A-Tree Languages with Membership Conditional Rewriting Mechanism

Hitoshi Ohsaki^{1,2}, Hiroyuki Seki³, and Toshinori Takai²

¹PRESTO “Information and Systems”
Japan Science and Technology Corporation

²National Institute of Advanced Industrial Science and Technology
Nakoji 3-11-46, Amagasaki 661-0974, Japan
{ohsaki, takai}@ni.aist.go.jp

³Nara Institute of Science and Technology
Takayama 8916-5, Ikoma 630-0192, Japan
seki@is.aist-nara.ac.jp

Abstract. This paper provides an algorithm to compute the complement of tree languages recognizable with A-TA (tree automata with associativity axioms [16]). Due to this closure property together with the previously obtained results, we know that the class is boolean closed, while keeping recognizability of A-closures of regular tree languages. In the proof of the main result, a new framework of tree automata, called *sequence-tree automata*, is introduced as a generalization of Lugiez and Dal Zilio’s multi-tree automata [14] of an associativity case. It is also shown that recognizable A-tree languages are closed under a one-step rewrite relation in case of ground A-term rewriting. This result allows us to compute an under-approximation of A-rewrite descendants of recognizable A-tree languages with arbitrary accuracy.

1 Introduction

In the tree automata theory, the following question has been asked frequently: What are natural definitions of *equationally* and *boolean* closed tree languages? The class of *regular* tree languages, which is the counterpart of regular word languages, is known to be well-behaved, such as to be closed under boolean operations together with many positive decidability properties [4, 8]. However, under consideration of several equational axioms, the equational closures of regular tree languages are no longer regular [6]. Due to this problem, recently there have been several attempts in which the tree automata framework is extended. Alternating two-way AC-tree automata of Goubault-Larrecq and Verma [9] succeeded in the sense that AC-closed tree languages can be recognized, while keeping decidability of the intersection-emptiness problem. Lugiez and Dal Zilio coped with the same question by inventing multi-tree automata [14]. Their extended framework is useful for manipulating *flattened*-tree languages. Actually, the framework provides the starting basis of this research, because the authors have shown that there exists *some* subclass of regular tree languages closed under the AC-congruence relation and under boolean operations. However, multi-tree automata for associativity axioms are not powerful in practice. For instance, the multi-trees like

$f(a, \dots, a, b, \dots, b)$, assuming the numbers of a and b are the same and f has a flexible arity, are not recognizable with multi-tree automata. In the usual term model the previous example can be represented as the A-closure of the language recognized by a regular tree automaton with the following transition rules $f(q_a, q_b) \rightarrow q_f$, $f(q_a, q_f) \rightarrow q$, $f(q, q_b) \rightarrow q_f$.

The past couple of years we have investigated in [16, 17] the recognizability and closure properties of associative and/or commutative tree languages by introducing a new framework, called *equational tree automata*. This framework allows us to handle equational tree languages, such as recognizable A-tree languages, which are effectively closed under union and intersection. The same closure property also holds in the AC-case. Our extension is useful for infinite-state model checking based on algebraic description languages of concurrent systems [18]. For instance, in protocol verification, network messages consisting of sequential data components are modeled with multiple lists, which are represented as terms with A-symbols. Besides, it is known that the emptiness problem for A-regular tree languages (i.e. tree languages recognizable with regular A-tree automata) is decidable. Unfortunately, there is also a negative result on A-tree languages: A-regular tree languages are *not* boolean closed. However, in the equational tree automata setting, the class of recognizable A-tree languages is properly wider than that of A-regular tree languages. So the remaining question arises again, as to whether or not recognizable A-tree languages are closed under complementation.

In the paper we obtain the solution to the unsolved question, in the way that A-tree languages are translated into sequence-tree languages. The translation is performed by a *flattening* operation, which is the standard technique in term rewriting, e.g. found in [2]. Intuitively, every non-empty and maximal context consisting of an A-symbol is replaced by a special function symbol $\langle \rangle$ which has a flexible arity. For instance, the term $f(a, f(b, c), d)$ is interpreted as $\langle a, b, c, d \rangle$. The resulting term is called a *sequence-term* (or sequence-tree). We thus need a new mechanism that manipulates sequence-terms. In Section 2 we introduce *sequence-tree automata* that allow a Galois connection to A-tree automata, even that have a bijective correspondence in recognizability. The new tree automata definition is inspired from Toyama's membership conditional term rewriting systems [20]. Furthermore, our formalization of sequence-tree automata generalizes Lugiez and Dal Zilio's multi-tree automata of the associativity case. In Section 3 we discuss the determinization of sequence-tree automata, which leads to the desired answer, that is, the complement closedness of recognizable A-tree languages. In Section 4 we demonstrate the usefulness of A-tree languages by showing that recognizable A-tree languages are closed under one-step ground A-rewriting. More precisely, since the tree language $\{s \mid t \rightarrow_{\mathcal{R}/A}^* s\}$ can not be handled with a decidable tree automata theory [6], we show in the paper that $\{s \mid \exists t \in L \text{ such that } t \rightarrow_{\mathcal{R}/A} s\}$ is recognizable with A-tree automata, provided L is recognizable with A-tree automata and \mathcal{R}/A is a ground A-rewrite system (i.e. rewrite rules in \mathcal{R} are ground and equations in A are associativity axioms for some binary symbols). This result is helpful in figuring out decidable subclasses for the first-order theories of one-step rewriting [5].

In the paper we assume the readers are familiar with the basics of term rewriting (explained, e.g., in [1, 3]). A subset \mathcal{F}_A of the signature \mathcal{F} consists of some binary function symbols. An equational system (ES for short) denoted by A is a set of associativity axioms $f(f(x, y), z) \approx f(x, f(y, z))$ for all f in \mathcal{F}_A . An *associative term rewriting system* (A-TRS for short) \mathcal{R}/A over \mathcal{F} is the combination of a TRS \mathcal{R} and an ES A . We write $s \rightarrow_{\mathcal{R}/A} t$ if there exist terms s' and t' such that $s \sim_A s' \rightarrow_{\mathcal{R}} t' \sim_A t$. The binary relation \sim_A is the equivalence relation induced by A . The reflexive-transitive closure and the transitive closure of $\rightarrow_{\mathcal{R}/A}$ are denoted by $\rightarrow_{\mathcal{R}/A}^*$ and $\rightarrow_{\mathcal{R}/A}^+$, respectively. A term t reachable from a term s with respect to \mathcal{R}/A , i.e. $s \rightarrow_{\mathcal{R}/A}^* t$, is called an \mathcal{R}/A -*descendant* of s . A term t with $s \rightarrow_{\mathcal{R}/A} t$ is called a *one-step \mathcal{R}/A -descendant* of s . The sets $\{t \mid \exists s \in L. s \rightarrow_{\mathcal{R}/A}^* t\}$ and $\{t \mid \exists s \in L. s \rightarrow_{\mathcal{R}/A} t\}$ for some set L of terms are denoted by $(\rightarrow_{\mathcal{R}/A}^*)[L]$ and $(\rightarrow_{\mathcal{R}/A})[L]$, respectively.

Next we fix our terminologies on tree automata. A *tree automaton* (TA for short) \mathcal{A} is the 4-tuple $(\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$ consisting of the signature \mathcal{F} , a finite set \mathcal{Q} of state symbols (special constants with $\mathcal{F} \cap \mathcal{Q} = \emptyset$), a set $\mathcal{Q}_{fin} (\subseteq \mathcal{Q})$ of final state symbols, and a finite set Δ of transition rules in one of the following forms:

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

for some $f \in \mathcal{F}$ with $\text{arity}(f) = n$ and $p_1, \dots, p_n, q, q_1, \dots, q_n \in \mathcal{Q}$. In the latter form, the root function symbols of the left- and right-hand sides must be the same. An *associative tree automaton* (A-TA for short) \mathcal{A}/A is the combination of a TA \mathcal{A} and an ES A over the same signature \mathcal{F} with \mathcal{F}_A . An A-TA \mathcal{A}/A is called *regular* if Δ consists only of rules in the former shape $f(p_1, \dots, p_n) \rightarrow q$.

An A-TA \mathcal{A}/A is a special A-TRS. In fact, Δ/A defines an A-TRS over the signature $\mathcal{F} \cup \mathcal{Q}$. We write $s \rightarrow_{\mathcal{A}/A} t$ if $s \rightarrow_{\Delta/A} t$. The binary relation $\rightarrow_{\mathcal{A}/A}$ over $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is called the *move relation* of \mathcal{A}/A . A term t in $\mathcal{T}(\mathcal{F})$ is *accepted* by \mathcal{A}/A if $t \rightarrow_{\mathcal{A}/A}^* q$ for some $q \in \mathcal{Q}_{fin}$. The set of terms accepted by \mathcal{A}/A is denoted by $\mathcal{L}(\mathcal{A}/A)$. A *tree language* L over \mathcal{F} is some subset of $\mathcal{T}(\mathcal{F})$. A tree language L is *recognizable* with A-TA if there exists \mathcal{A}/A such that $L = \mathcal{L}(\mathcal{A}/A)$. If L is recognizable with regular A-TA, it is called A-regular. A tree language closed under A-congruence relation is called an A-closed tree language, or simply called an A-tree language.

2 Sequence-Tree Automata

We begin this section by introducing a new concept of tree automata, which is called *sequence-tree automata*. The new framework enables us to accept (ground) *sequence-terms*: Given the signature \mathcal{F} , we say t is a sequence-term (or sequence-tree) in $\text{s-}\mathcal{T}(\mathcal{F})$

- if $t = f(t_1, \dots, t_n)$ such that $f \in \mathcal{F}_n$ and $t_i \in \text{s-}\mathcal{T}(\mathcal{F})$ for all $1 \leq i \leq n$, or
- if $t = \langle t_1, \dots, t_m \rangle$ such that $m \geq 2$ and $t_i \in \text{s-}\mathcal{T}(\mathcal{F})$ and $\text{root}(t_i) \in \mathcal{F}$ for all $1 \leq i \leq m$.

We assume $\langle \rangle$ to be a special symbol with a flexible arity. Each function symbol f in \mathcal{F} has a fixed arity, which is represented by the mapping $\text{arity} : \mathcal{F} \rightarrow \mathbb{N}$. A subset \mathcal{F}_n of \mathcal{F} consists of function symbols f with $\text{arity}(f) = n$.

A sequence-tree automaton (s-TA for short) $\mathcal{M} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$ is defined as an easy extension of the standard TA, by allowing transition rules for the special symbol $\langle \rangle$ with the following shape:

$$\langle X \rangle \rightarrow q \Leftarrow \text{concat}(X) \text{ in } \mathcal{L}(\mathcal{G})$$

Here the capital letter X is called a *sequence-variable* [10,13]. In the above transition rule, X is expected to be instantiated to some sequence q_1, \dots, q_n ($n \geq 2$) of states. So in the *conditional part* of the rule, the concatenation $q_1 \cdots q_n$ of state symbols, denoted by $\text{concat}(X)$, is examined whether it is accepted by a *word grammar* \mathcal{G} over \mathcal{Q} . Each grammar in conditional rules is not necessarily the same as each other grammar in different rules. If the membership condition $\text{concat}(X) \text{ in } \mathcal{L}(\mathcal{G})$ is satisfied for some instance of X , the left pattern can be replaced by a single state q .

A word grammar $\mathcal{G} = (\Sigma, \mathcal{S}, s_0, A)$ over the alphabet Σ with \mathcal{S} nonterminal symbols and s_0 the starting symbol is called (1) *context-sensitive* (CSG for short) if $|l| \leq |r|$, (2) *context-free* if $l \in \mathcal{S}$, (3) *regular* if $l \in \mathcal{S}$ and $r \in \{as \mid a \in \Sigma \text{ and } s \in \mathcal{S}\} \cup \{a \mid a \in \Sigma\}$ for all production rules $l \rightarrow r$ in A . We say \mathcal{G} is a *monotone* grammar if every rule in A has one of the following forms:

$$p \rightarrow a \quad \text{or} \quad p \rightarrow q_1 q_2 \quad \text{or} \quad p_1 p_2 \rightarrow q_1 q_2$$

with $p, p_1, p_2, q_1, q_2 \in \mathcal{S}$ and $a \in \Sigma$.

An s-TA is called a monotone (resp. context-sensitive, context-free, regular) s-TA if word grammars in the conditional part are monotone (resp. context-sensitive, context-free, regular). In the paper we say an s-TA instead of a monotone s-TA. A sequence-tree language recognizable with a monotone (resp. context-sensitive, context-free, regular) s-TA is called monotone (resp. context-sensitive, context-free, regular). In the literature, e.g. in [15], monotone grammars are called *Kuroda normal forms* of CSG. Moreover, it is known that for every CSG, we can compute an equivalent grammar in Kuroda normal form [12]. The expressive power of sequence-tree automata is determined by the generative power of \mathcal{G} . In fact, we have the strict language hierarchy between the classes of monotone, context-free, and regular sequence-tree languages.

Next we discuss the relationship between A-TA and s-TA. Hereafter we assume $\mathcal{F}_A = \{f\}$. Moreover, we say a context of a term is an *f-block* if it is a non-empty maximal context consisting of f only. For notational convenience, we write $C'_f[C_f[t_1, \dots, t_n]]$ for $C'[C[t_1, \dots, t_n]]$ if C is an f -block. For instance, $g(C_f[a, b, c])$ represents the terms $g(f(a, f(b, c)))$ and $g(f(f(a, b), c))$.

We define the two mappings *flat* and *unflat* as follows: Let t be a term in $\mathcal{T}(\mathcal{F})$,

$$\text{flat}(t) = \begin{cases} \langle \text{flat}(t_1), \dots, \text{flat}(t_m) \rangle & \text{if } t = C_f[t_1, \dots, t_m], \\ f(\text{flat}(t_1), \dots, \text{flat}(t_n)) & \text{if } t = f(t_1, \dots, t_n) \text{ and } f \neq f. \end{cases}$$

Let $\mathcal{F}' = \mathcal{F} - \{f\}$ and let t be a sequence-term in $s\text{-}\mathcal{T}(\mathcal{F}')$,

$$\text{unflat}(t) = \begin{cases} f(\dots f(\text{unflat}(t_1), \text{unflat}(t_2)) \dots, \text{unflat}(t_m)) & \text{if } t = \langle t_1, t_2, \dots, t_m \rangle, \\ f(\text{unflat}(t_1), \dots, \text{unflat}(t_n)) & \text{if } t = f(t_1, \dots, t_n) \text{ and } f \neq f. \end{cases}$$

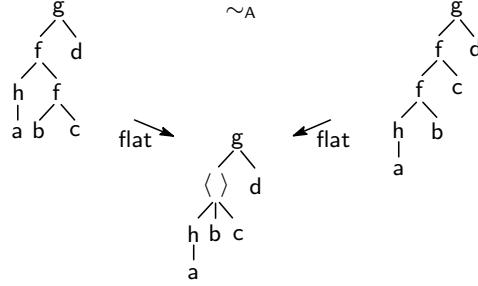


Fig. 1. Flattening by flat

An example of a flat-transformation is illustrated in Fig. 1. One should notice that $\text{flat}(\text{unflat}(t)) = t$ for any sequence-term t in $s\mathcal{T}(\mathcal{F}', \mathcal{V})$. However, $\text{unflat}(\text{flat}(t))$ and t are not always the same, but $\text{unflat}(\text{flat}(t)) \sim_A t$. For instance,

$$\text{unflat}(\text{flat}(g(f(h(a), f(b, c)), d))) = g(f(f(h(a), b), c), d) \neq g(f(h(a), f(b, c)), d)$$

although $g(f(f(h(a), b), c), d) \sim_A g(f(h(a), f(b, c)), d)$.

Using the mappings flat and unflat , we show that the classes of A-TA and s-TA have a bijective correspondence in recognizability. More precisely, in the remaining part of this section, we prove the next statement.

Theorem 1. *Given an A-tree language L over the signature \mathcal{F} with $\mathcal{F}_A = \{f\}$, there exists an A-TA \mathcal{A}/A such that $\mathcal{L}(\mathcal{A}/A) = L$ if and only if there exists an s-TA \mathcal{M} such that $\mathcal{L}(\mathcal{M}) = \text{flat}(L)$. \square*

In the above theorem A-closedness of L is essential. For instance, the tree language $\{f(f(a, a), a)\}$ is not recognizable with A-TA, because an A-TA which accepts $f(f(a, a), a)$ also accepts $f(a, f(a, a))$. On the other hand, we can easily define an s-TA that accepts $\langle a, a, a \rangle$ only.

Let us explain the idea for the “only if” proof of the above theorem. We consider the A-TA \mathcal{A}/A with $\mathcal{A} = (\{f, a, b\}, \{p, q_1, q_2\}, \{p\}, \Delta)$ where Δ :

$$\begin{array}{lll} a \rightarrow q_1 & f(q_1, q_2) \rightarrow p & f(q_1, q_2) \rightarrow f(q_2, q_1) \\ b \rightarrow q_2 & f(p, p) \rightarrow p & f(q_2, q_1) \rightarrow f(q_1, q_2) \end{array}$$

The A-TA \mathcal{A}/A recognizes the tree language $\{t \in \mathcal{T}(\{f, a, b\}) \mid |t|_a = |t|_b\}$, i.e. the set of ground terms t which have the same numbers of a and b in t . Now we define the associated s-TA $\mathcal{M}_{\mathcal{A}/A} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta_1 \cup \Delta_2)$ as follows:

$$\begin{array}{l} \mathcal{F} : a, b \\ \mathcal{Q} : p, q_1, q_2 \\ \mathcal{Q}_{fin} : p \\ \Delta_1 : a \rightarrow q_1, b \rightarrow q_2 \\ \Delta_2 : \langle X \rangle \rightarrow p \Leftarrow \text{concat}(X) \text{ in } \mathcal{L}(\mathcal{G}_p) \end{array}$$

where $\mathcal{G}_p = (\{p, q_1, q_2\}, \{\alpha_p, \alpha_{q_1}, \alpha_{q_2}\}, \alpha_p, \Lambda)$ and

$$\begin{aligned} \Lambda = & \{ \alpha_p \rightarrow \alpha_{q_1} \alpha_{q_2}, \alpha_p \rightarrow \alpha_p \alpha_p, \alpha_{q_2} \alpha_{q_1} \rightarrow \alpha_{q_1} \alpha_{q_2}, \alpha_{q_1} \alpha_{q_2} \rightarrow \alpha_{q_2} \alpha_{q_1} \} \\ & \cup \{ \alpha_p \rightarrow p, \alpha_{q_1} \rightarrow q_1, \alpha_{q_2} \rightarrow q_2 \}. \end{aligned}$$

In the construction, a monotone grammar \mathcal{G}_p has a production rule of the form $\alpha_p \rightarrow \alpha_q \alpha_r$ if Δ contains a transition rule $f(q, r) \rightarrow p \in \Delta$. Likewise, \mathcal{G}_p has a rule $\alpha_p \alpha_q \rightarrow \alpha_r \alpha_s$ if $f(r, s) \rightarrow f(p, q)$ in Δ . As a consequence, we obtain the s-TA $\mathcal{M}_{\mathcal{A}/\mathcal{A}}$ satisfying that: $\mathcal{M}_{\mathcal{A}/\mathcal{A}}$ allows a one-step move $\langle p_1, \dots, p_n \rangle \rightarrow_{\mathcal{M}_{\mathcal{A}/\mathcal{A}}} p$ if and only if $\alpha_p \xrightarrow{\mathcal{G}_p}^* \alpha_{p_1} \cdots \alpha_{p_n}$. In fact, the above s-TA construction satisfies the following property.

Lemma 1. *Given an A-TA \mathcal{A}/\mathcal{A} with $\mathcal{A} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$. Suppose $\mathcal{M}_{\mathcal{A}/\mathcal{A}} = (\mathcal{F}', \mathcal{Q}', \mathcal{Q}'_{fin}, \Delta')$ is the s-TA obtained by the above construction. Then, for all $p_1, \dots, p_n, q \in \mathcal{Q}$, $C_f \llbracket p_1, \dots, p_n \rrbracket \xrightarrow{\mathcal{A}/\mathcal{A}}^* q$ if and only if $\alpha_q \xrightarrow{\mathcal{G}_q}^* \alpha_{p_1} \cdots \alpha_{p_n}$. \square*

Using this lemma, we can prove the “only if” direction of Theorem 1.

Lemma 2. *Given an A-TA \mathcal{A}/\mathcal{A} with $\mathcal{A} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$. Suppose $\mathcal{M}_{\mathcal{A}/\mathcal{A}} = (\mathcal{F}', \mathcal{Q}', \mathcal{Q}'_{fin}, \Delta')$ is the s-TA obtained as in the previous lemma. Then, $t \in \mathcal{L}(\mathcal{A}/\mathcal{A})$ if and only if $\text{flat}(t) \in \mathcal{L}(\mathcal{M}_{\mathcal{A}/\mathcal{A}})$.*

Proof. We show the “only if” part. The reverse can be proved in a similar way. We assume $t \xrightarrow{\mathcal{A}/\mathcal{A}}^* q$ for some $q \in \mathcal{Q}$. Then we show by the structural induction that $\text{flat}(t) \xrightarrow{\mathcal{M}_{\mathcal{A}/\mathcal{A}}}^* q$. If t is a constant c , there exists a transition rule $c \rightarrow q \in \Delta$. Then, by construction, $c \rightarrow q \in \Delta'$. If $t = f(t_1, \dots, t_n)$ with $f \neq \text{f}$, then $t_i \xrightarrow{\mathcal{A}/\mathcal{A}}^* p_i$ for some $p_i \in \mathcal{Q}$ ($1 \leq i \leq n$) such that $f(p_1, \dots, p_n) \rightarrow q \in \Delta$. By induction hypothesis, $\text{flat}(t_i) \xrightarrow{\mathcal{M}_{\mathcal{A}/\mathcal{A}}}^* p_i$ ($1 \leq i \leq n$). Moreover, $f(p_1, \dots, p_n) \rightarrow q \in \Delta'$. Thus, $\text{flat}(t) \xrightarrow{\mathcal{M}_{\mathcal{A}/\mathcal{A}}}^* f(p_1, \dots, p_n) \rightarrow_{\mathcal{M}_{\mathcal{A}/\mathcal{A}}} q$. If $t = C_f \llbracket t_1, \dots, t_m \rrbracket$, then $t_i \xrightarrow{\mathcal{A}/\mathcal{A}}^* p_i$ for all $1 \leq i \leq m$ and $C_f \llbracket p_1, \dots, p_m \rrbracket \xrightarrow{\mathcal{A}/\mathcal{A}}^* q$. By induction hypothesis, $\text{flat}(t_i) \xrightarrow{\mathcal{M}_{\mathcal{A}/\mathcal{A}}}^* p_i$ ($1 \leq i \leq m$). Thus, $\text{flat}(t) = \langle \text{flat}(t_1), \dots, \text{flat}(t_m) \rangle \xrightarrow{\mathcal{M}_{\mathcal{A}/\mathcal{A}}}^* \langle p_1, \dots, p_m \rangle \rightarrow_{\mathcal{M}_{\mathcal{A}/\mathcal{A}}} q$, because there exists a transition rule $\langle X \rangle \rightarrow q \Leftarrow \text{concat}(X)$ in $\mathcal{L}(\mathcal{G}_q)$. By Lemma 1, $p_1 \cdots p_m \in \mathcal{L}(\mathcal{G}_q)$ is guaranteed. \square

Next we show the reverse (the “if” part of Theorem 1). The proof is achieved as in the previous lemma. Suppose $\mathcal{M} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$. Without loss of generality, we assume that for every rule of the form $\langle X \rangle \rightarrow q \Leftarrow \text{concat}(X)$ in $\mathcal{L}(\mathcal{G})$, nonterminal symbols of \mathcal{G} are pairwise distinct from the other word grammars. We define the associated A-TA $\mathcal{A}_{\mathcal{M}}/\mathcal{A}$ as follows: $\mathcal{A}_{\mathcal{M}} = (\mathcal{F} \cup \{f\}, \mathcal{Q}', \mathcal{Q}'_{fin}, \Delta')$ and $\mathcal{F}_{\mathcal{A}} = \{f\}$ where

$$\begin{aligned} \Delta' = & \{ l \rightarrow r \in \Delta \mid \text{root}(l) \in \mathcal{F} \} \\ & \cup \{ f(\alpha, \beta) \rightarrow f(\gamma, \delta) \mid \langle X \rangle \rightarrow q \Leftarrow \text{concat}(X) \text{ in } \mathcal{L}(\mathcal{G}) \in \Delta, \gamma \delta \rightarrow \alpha \beta \in \mathcal{G} \} \\ & \cup \{ f(\alpha, \beta) \rightarrow \gamma \mid \langle X \rangle \rightarrow q \Leftarrow \text{concat}(X) \text{ in } \mathcal{L}(\mathcal{G}) \in \Delta, \gamma \rightarrow \alpha \beta \in \mathcal{G} \} \\ & \cup \{ f(\alpha, \beta) \rightarrow q \mid \langle X \rangle \rightarrow q \Leftarrow \text{concat}(X) \text{ in } \mathcal{L}(\mathcal{G}) \in \Delta, \gamma \rightarrow \alpha \beta \in \mathcal{G}, \\ & \hspace{15em} \text{and } \gamma \text{ is a starting symbol of } \mathcal{G} \} \\ & \cup \{ f(p_1, \dots, p_n) \rightarrow \alpha \mid f(p_1, \dots, p_n) \rightarrow p \in \Delta \text{ with } f \neq \text{f}, \\ & \hspace{10em} \langle X \rangle \rightarrow q \Leftarrow \text{concat}(X) \text{ in } \mathcal{L}(\mathcal{G}) \in \Delta, \alpha \rightarrow p \in \mathcal{G} \} \end{aligned}$$

The set \mathcal{Q}' of new nonterminal symbols is \mathcal{Q} together with nonterminal symbols of all word grammars of conditional rules in Δ . Then we can prove the next lemma.

Lemma 3. *Given an s-TA \mathcal{M} over the signature \mathcal{F} such that $f \notin \mathcal{F}$. Suppose $\mathcal{F}' = \mathcal{F} \cup \mathcal{F}_A$ with $\mathcal{F}_A = \{f\}$ and $\mathcal{A}_{\mathcal{M}}/A$ is the A-TA over \mathcal{F}' obtained from \mathcal{M} by the above construction. Then, $t \in \mathcal{L}(\mathcal{M})$ if and only if $\text{unflat}(t) \in \mathcal{L}(\mathcal{A}_{\mathcal{M}}/A)$. \square*

Bijjective correspondence between s-TA and A-TA is beneficial: For instance, the proof of the emptiness problem for A-TA (Corollary 1 in [16]) can be simplified. We prove the same undecidability for s-TA below: Given a monotone grammar $\mathcal{G} = (\Sigma, \mathcal{S}, s_0, A)$, we define the s-TA $\mathcal{M}_{\mathcal{G}} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$ associated with \mathcal{G} as follows. $\mathcal{F} = \{c_a \mid a \in \Sigma\}$, $\mathcal{Q} = \Sigma \cup \{q\}$ with $q \notin \Sigma$, $\mathcal{Q}_{fin} = \{q\}$, $\Delta = \{c_a \rightarrow a \mid a \in \Sigma\} \cup \{\langle X \rangle \rightarrow q \Leftarrow \text{concat}(X) \text{ in } \mathcal{L}(\mathcal{G}_q)\}$ Then $\mathcal{L}(\mathcal{M}_{\mathcal{G}}) = \emptyset$ if and only if $\mathcal{L}(\mathcal{G}_q) = \emptyset$. It is known that the emptiness problem for monotone grammars is undecidable, and thus, the problem is also undecidable for s-TA.

Using our transformation (from s-TA to A-TA, and the reverse), it can also be proved that the class of monotone sequence-tree languages is effectively closed under union and intersection, because the class of A-tree languages is also effectively closed under union and intersection (Theorems 2,3 in [16]). The additional benefit from using Theorem 1 is that a complexity result in one framework yields the same result in the other framework.

Theorem 2. *The membership problem for s-TA is PSPACE-complete. \square*

Proof (Outline). The nontrivial part, showing the PSPACE-hardness, of this theorem can be achieved by reducing from the membership problem for CSG, which is known to be PSPACE-complete. \square

As an immediate consequence, we obtain the complexity result for A-TA.

Corollary 1. *The membership problem for A-TA is PSPACE-complete. \square*

3 Determinization of Sequence-Tree Automata

We show in this section that context-sensitive sequence-tree automata can be determinized using the algorithm in Fig. 2. The algorithm is obtained by generalizing the standard *subset construction* technique. This implies that recognizable sequence-tree languages and hence A-tree languages are closed under complementation, which provides a positive answer to an important remaining question in [16, 17].

Lemma 4. *Given some subsets A_i ($1 \leq i \leq n$) of a set S , for every subset I of indices $\{1, \dots, n\}$, we define $C_I = \bigcap_{i \in I} A_i - \bigcup_{i \in (\{1, \dots, n\} - I)} A_i$. Then $C_I \cap C_J = \emptyset$ if $I \neq J$. \square*

Lemma 5. *\mathcal{M}_d is a complete and deterministic s-TA.*

Let $\mathcal{M} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$ be an s-TA. We assume without loss of generality that Δ does not contain different conditional rules for the same right-hand side q . In case Δ contains $\langle X \rangle \rightarrow q \Leftarrow \text{concat}(X)$ in $\mathcal{L}(\mathcal{G}_1)$ and $\langle X \rangle \rightarrow q \Leftarrow \text{concat}(X)$ in $\mathcal{L}(\mathcal{G}_2)$ with $\mathcal{G}_1 \neq \mathcal{G}_2$, they can be merged to a single rule $\langle X \rangle \rightarrow q \Leftarrow \text{concat}(X)$ in $\mathcal{L}(\mathcal{G}_3)$ such that $\mathcal{L}(\mathcal{G}_3) = \mathcal{L}(\mathcal{G}_1) \cup \mathcal{L}(\mathcal{G}_2)$.

Next, for every conditional rule $\langle X \rangle \rightarrow q \Leftarrow \text{concat}(X)$ in $\mathcal{L}(\mathcal{G}_q)$ in Δ , we define a monotone (but not necessarily deterministic) grammar \mathcal{G}_q^d as follows: Let $\mathcal{G}_q = (\Sigma, \mathcal{S}, s_0, A)$, then $\mathcal{G}_q^d = (2^\Sigma, \mathcal{S}, s_0, A')$ where

$$A' = (A - \{\alpha \rightarrow a \mid a \in \Sigma\}) \cup \{\alpha \rightarrow A \mid \alpha \rightarrow a \in A, A \in 2^\Sigma, a \in A\}.$$

Finally, we define an s-TA $\mathcal{M}_d = (\mathcal{F}, \mathcal{Q}_d, \mathcal{Q}_{dfin}, \Delta_d)$ as follows:

$$\begin{aligned} \mathcal{Q}_d &= 2^\mathcal{Q} \\ \mathcal{Q}_{dfin} &= \{A \in \mathcal{Q}_d \mid A \cap \mathcal{Q}_{fin} \neq \emptyset\} \\ \Delta_d &= \Delta_{d1} \cup \Delta_{d2} \cup \Delta_{d3} \text{ where} \\ \Delta_{d1} &: f(A_1, \dots, A_n) \rightarrow A \\ &\quad \text{for } A_i \in \mathcal{Q}_d \ (1 \leq i \leq n) \text{ and} \\ &\quad A = \{q \mid f(q_1, \dots, q_n) \rightarrow q \in \Delta \text{ such that } q_i \in A_i \ (1 \leq i \leq n)\}, \\ \Delta_{d2} &: \langle X \rangle \rightarrow A \Leftarrow \text{concat}(X) \text{ in } \mathcal{L}(\mathcal{G}_A) \\ &\quad \text{for } A \in (\mathcal{Q}_d - \{\emptyset\}) \text{ and} \\ &\quad \mathcal{L}(\mathcal{G}_A) = \bigcap_{q \in A} \mathcal{L}(\mathcal{G}_q^d) - \bigcup_{q \in \mathcal{Q} - A} \mathcal{L}(\mathcal{G}_q^d), \\ \Delta_{d3} &: \langle X \rangle \rightarrow \emptyset \Leftarrow \text{concat}(X) \text{ in } \mathcal{L}(\mathcal{G}_0), \\ &\quad \text{where } \mathcal{L}(\mathcal{G}_0) = \{w \in \mathcal{Q}_d^* \mid |w| \geq 2\} - \bigcup_{q \in \mathcal{Q}} \mathcal{L}(\mathcal{G}_q^d). \end{aligned}$$

Fig. 2. Determinization of sequence-tree automata

Proof. First we show that \mathcal{M}_d is a deterministic s-TA. By Kuroda's Lemma ([12]), given a context-sensitive grammar, there effectively exists a monotone grammar with the same generative power. Moreover, monotone languages are effectively closed under boolean operations [11, 12]. Thus a language $\mathcal{L}(\mathcal{G}_A)$ of a conditional rule in Δ_{d2} can be defined by a monotone grammar. The same holds for $\mathcal{L}(\mathcal{G}_0)$ of Δ_{d3} . So we can compute an s-TA \mathcal{M}_d . Next we show the determinism of Δ_d . By definition, there is no overlapping at the root position in the left-hand sides of two different rules in Δ_{d1} . For transition rules in Δ_{d2} , we take two rules $\langle X \rangle \rightarrow A_1 \Leftarrow \text{concat}(X)$ in $\mathcal{L}(\mathcal{G}_{A1})$ and $\langle X \rangle \rightarrow A_2 \Leftarrow \text{concat}(X)$ in $\mathcal{L}(\mathcal{G}_{A2})$ in Δ_{d2} . By Lemma 4, we obtain $\mathcal{L}(\mathcal{G}_{A1}) \cap \mathcal{L}(\mathcal{G}_{A2}) = \emptyset$. Then there is no instance t of $\langle X \rangle$ such that $t \in \mathcal{L}(\mathcal{G}_{A1}) \cap \mathcal{L}(\mathcal{G}_{A2})$. For Δ_{d3} , the intersection of $\{w \in \mathcal{Q}_d^* \mid |w| \geq 2\} - \bigcup_{q \in \mathcal{Q}} \mathcal{L}(\mathcal{G}_q^d)$ and $\bigcap_{q \in \mathcal{Q}} \mathcal{L}(\mathcal{G}_q^d)$ is empty. Hence, there is no overlapping between Δ_{d3} and Δ_{d2} .

Next we show that \mathcal{M}_d is complete. By definition, for every $f \in \mathcal{F}$ and $A_1, \dots, A_n \in \mathcal{Q}_d$, there exists a transition rule of the form $f(A_1, \dots, A_n) \rightarrow A$ in Δ_{d1} . For another pattern $\langle A_1, \dots, A_m \rangle$ with $A_1, \dots, A_m \in \mathcal{Q}_d$ and $m \geq 2$, we take $L = \bigcup_{A \in (\mathcal{Q}_d - \{\emptyset\})} \mathcal{L}(\mathcal{G}_A)$. Then we can show that $\bigcup_{q \in \mathcal{Q}} \mathcal{L}(\mathcal{G}_q^d)$ is the same

as L . This implies $L \cup \mathcal{L}(\mathcal{G}_0) = \{w \in \mathcal{Q}_d^* \mid |w| \geq 2\}$. Hence, if $A_1 \cdots A_m \in L$, there exists a transition rule $\langle X \rangle \rightarrow A \Leftarrow \text{concat}X$ in $\mathcal{L}(\mathcal{G}_A)$ in Δ_{d2} that is applicable to $\langle A_1, \dots, A_m \rangle$; otherwise, Δ_{d3} is applicable. \square

Lemma 6. *Suppose $\langle X \rangle \rightarrow A \Leftarrow \text{concat}(X)$ in $\mathcal{L}(\mathcal{G}_A)$ is a transition rule in Δ_{d2} . Let $L_A = \{A_1 \cdots A_k \in \mathcal{Q}_d^* \mid q \in A \text{ if and only if } \exists q_i \in A_i (1 \leq i \leq k) \text{ and } q_1 \cdots q_k \in \mathcal{L}(\mathcal{G}_q)\}$. Then $\mathcal{L}(\mathcal{G}_A) = L_A$. \square*

Lemma 7. $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}_d)$.

Proof. We show that for every sequence-term $t \in \text{s-T}(\mathcal{F})$ and state symbol $A \in \mathcal{Q}_d$, if $t \rightarrow_{\mathcal{M}_d}^* A$ then A is the same as the set $\{q \in \mathcal{Q} \mid t \rightarrow_{\mathcal{M}}^* q\}$. We use the structural induction on t . If t is a constant, \mathcal{M}_d has a transition rule $t \rightarrow \{q \in \mathcal{Q} \mid t \rightarrow q \in \Delta\}$, and \mathcal{M}_d has no other rule for t . We suppose $t = f(t_1, \dots, t_n)$ with $f \in \mathcal{F}$. Then, $t_i \rightarrow_{\mathcal{M}_d}^* A_i$ ($1 \leq i \leq n$) and $f(A_1, \dots, A_n) \rightarrow A \in \Delta_d$. By Lemma 5, \mathcal{M}_d does not allow the move relation for t except $t \rightarrow_{\mathcal{M}_d}^* f(A_1, \dots, A_n) \rightarrow_{\mathcal{M}_d} A$. By definition, $A = \{q \in \mathcal{Q} \mid \exists f(q_1, \dots, q_n) \rightarrow q \in \Delta \text{ such that } q_i \in A_i (1 \leq i \leq n)\}$. On the other hand, by induction hypothesis, we obtain $A_i = \{q_i \in \mathcal{Q} \mid t_i \rightarrow_{\mathcal{M}}^* q_i\}$. Then, $A = \{q \in \mathcal{Q} \mid \exists f(q_1, \dots, q_n) \rightarrow q \in \Delta \text{ such that } t_i \rightarrow_{\mathcal{M}}^* q_i (1 \leq i \leq n)\} = \{q \in \mathcal{Q} \mid f(t_1, \dots, t_n) \rightarrow_{\mathcal{M}}^* q\}$. Next we suppose $t = \langle t_1, \dots, t_m \rangle$ such that $m \geq 2$ and $t_i \rightarrow_{\mathcal{M}_d}^* A_i$ ($1 \leq i \leq m$). By Lemma 5, \mathcal{M}_d does not allow the move relation for t except $t \rightarrow_{\mathcal{M}_d}^* \langle A_1, \dots, A_m \rangle \rightarrow_{\mathcal{M}_d} A$. By construction, the last move relation $\langle A_1, \dots, A_m \rangle \rightarrow_{\mathcal{M}_d} A$ is made by the rule $\langle X \rangle \rightarrow A \Leftarrow \text{concat}(X)$ in $\mathcal{L}(\mathcal{G}_A)$ in Δ_d . By Lemma 6, $q \in A$ if and only if $\exists q_i \in A_i (1 \leq i \leq m)$ and $q_1 \cdots q_m \in \mathcal{L}(\mathcal{G}_q)$. Hence, by induction hypothesis, A is the same as the set $\{q \in \mathcal{Q} \mid \exists q_i \in \mathcal{Q} \text{ such that } t_i \rightarrow_{\mathcal{M}}^* q_i (1 \leq i \leq m) \text{ and } \langle q_1, \dots, q_m \rangle \rightarrow_{\mathcal{M}} q\}$.

The rest of the proof is easy. By Lemma 5 for every $t \in \text{s-T}(\mathcal{F})$ there exists a (and only one) state $A \in \mathcal{Q}_d$ such that $t \rightarrow_{\mathcal{M}_d}^* A$. From the above property, $A \in \mathcal{Q}_{d\text{fin}}$ if and only if $t \rightarrow_{\mathcal{M}}^* q \in \mathcal{Q}_{\text{fin}}$ for some $q \in A$. \square

Theorem 3. *Given an s-TA \mathcal{M} , we can compute a complete and deterministic s-TA \mathcal{M}_d such that $\mathcal{L}(\mathcal{M}_d) = \mathcal{L}(\mathcal{M})$. Thus monotone sequence-tree languages are effectively closed under complementation. \square*

Corollary 2. *Tree languages recognizable with A-TA are effectively closed under complementation.*

Proof. Given an A-TA \mathcal{A}/A over the signature \mathcal{F} with $\mathcal{F}_A = \{f\}$, by Lemmata 2 and 3 and Theorem 3, we can compute \mathcal{B}/A such that $t \in \text{flat}(\mathcal{T}(\mathcal{F}) - \mathcal{L}(\mathcal{A}/A))$ if and only if $\text{unflat}(t) \in \mathcal{L}(\mathcal{B}/A)$. Note that $\text{s-T}(\mathcal{F} - \mathcal{F}_A) - \text{flat}(\mathcal{L}(\mathcal{A}/A)) = \text{flat}(\mathcal{T}(\mathcal{F}) - \mathcal{L}(\mathcal{A}/A))$. Thus, by showing that \mathcal{B}/A recognizes the A-closure of $\text{unflat}(\text{flat}(\mathcal{T}(\mathcal{F}) - \mathcal{L}(\mathcal{A}/A)))$, we know that the complement of \mathcal{A}/A is recognizable with \mathcal{B}/A . \square

4 Recognizing One-step A-Rewrite Descendants

In order to validate a safety property with model checking approach, *reachability* is a fundamental problem to be handled. Nevertheless, it is proved in equational

term rewriting that, there is no algorithm capable of deciding $s \rightarrow_{\mathcal{R}/A}^* t$ even if \mathcal{R} is ground. Due to this fact, we know that the tree language $(\rightarrow_{\mathcal{R}/A}^*)[\{s\}]$ is no longer *effectively* recognizable. However, these negative results do not imply incomputability of $(\rightarrow_{\mathcal{R}/A}^n)[\{s\}]$ for an arbitrary but fixed n . In fact, we show in this section that, given a ground A-TRS \mathcal{R}/A and an A-TA \mathcal{A}/A , the tree language $(\rightarrow_{\mathcal{R}/A})[\mathcal{L}(\mathcal{A}/A)]$ is effectively recognizable with A-TA.

In case that \mathcal{R} is ground, every rule $l \rightarrow r$ in \mathcal{R} can be simulated by decomposed rules $f(c_1, \dots, c_m) \rightarrow c_0$ and $d_0 \rightarrow g(d_1, \dots, d_n)$, where c_i ($0 \leq i \leq m$) and d_j ($0 \leq j \leq n$) are fresh constants. The nontrivial case to be considered in the proofs occurs in the rewrite relation made by a rule $f(c_1, c_2) \rightarrow c_0$ with f an associativity symbol. First we consider in the following the string rewriting case. We then generalize this result to the A-term rewriting case.

Lemma 8. *Given a monotone grammar $\mathcal{G} = (\Sigma, \mathcal{S}, s_0, \Lambda)$ and a string rewrite system $\mathcal{R} = \{a b \rightarrow c\}$ with $a, b, c \in \Sigma$, we can compute a monotone grammar that recognizes $(\rightarrow_{\mathcal{R}})[\mathcal{L}(\mathcal{G})]$.*

Proof. We assume without loss of generality that in Λ , (1) there is only one transition rule whose left-hand side is s_0 and (2) s_0 does not appear in the right-hand sides. We define a monotone grammar $\mathcal{G}' = (\Sigma, \mathcal{S}', s_0, \Lambda')$ as follows: $\mathcal{S}' = \mathcal{S} \cup \{[pq] \mid p, q \in \mathcal{S}\}$ and

$$\Lambda' = \left\{ \begin{array}{l|l} s_0 \rightarrow c & s_0 \rightarrow_{\mathcal{G}}^* a b \end{array} \right\} \dots (1)$$

$$\cup \left\{ \begin{array}{l|l} s_0 \rightarrow [pq] r & p, q, r \in \mathcal{S} \text{ such that } s_0 \rightarrow_{\mathcal{G}}^* p q r \end{array} \right\} \dots (2)$$

$$\cup \left\{ \begin{array}{l|l} [pq] \rightarrow c & p \rightarrow a, q \rightarrow b \in \Lambda \end{array} \right\} \dots (3)$$

$$\cup \left\{ \begin{array}{l|l} p \rightarrow a & p \rightarrow a \in \Lambda, p \neq s_0, a \in \Sigma \end{array} \right\} \dots (4)$$

$$\cup \left\{ \begin{array}{l|l} [pq] \rightarrow [p_1 p_2] q \\ [qp] \rightarrow [q p_1] p_2 \\ p \rightarrow p_1 p_2 & p \rightarrow p_1 p_2 \in \Lambda, p \neq s_0, q \in \mathcal{S} \end{array} \right\} \dots (5)$$

$$\cup \left\{ \begin{array}{l|l} [p_1 p_2] q \rightarrow [q_1 q_2] q \\ q [p_1 p_2] \rightarrow q [q_1 q_2] \\ p_1 p_2 \rightarrow q_1 q_2 & p_1 p_2 \rightarrow q_1 q_2 \in \Lambda, q \in \mathcal{S} \end{array} \right\} \dots (6)$$

$$\cup \left\{ \begin{array}{l|l} [pq] r \rightarrow p [q r] \\ p [q r] \rightarrow [p q] r & p, q, r \in \mathcal{S} \end{array} \right\} \dots (7)$$

Rules in (1) and (2) are computable, because the membership problem is decidable for monotone grammar. The rest of the proof is done by case analysis. \square

In Lemma 8, each of the letters a, b, c is not necessarily different from each other. Moreover, we can take a string rewrite rule, such that the length of the left-hand side is more than 2. In fact, if $\mathcal{R} = \{w \rightarrow c\}$ such that $w \in \Sigma^+$ and $|w| = n$ (≥ 2), we define Λ' as follows: (1)–(3) are replaced by

$$\left\{ \begin{array}{l|l} s_0 \rightarrow c & s_0 \rightarrow_{\mathcal{G}}^* w \end{array} \right\} \dots (1')$$

$$\left\{ \begin{array}{l|l} s_0 \rightarrow [p_1 \dots p_n] q & s_0 \rightarrow_{\mathcal{G}}^* p_1 \dots p_n q \in \mathcal{S}^+ \end{array} \right\} \dots (2')$$

$$\left\{ [p_1 \dots p_n] \rightarrow c \quad \left| \quad p_i \rightarrow a_i \in \Lambda, w = a_1 \dots a_n \right. \right\} \dots (3')$$

Rules in (5) are replaced by, e.g.

$$[p_1 \cdots p_i \cdots p_{n-1} p_n] \rightarrow [p_1 \cdots q_1 q_2 \cdots p_{n-1}] p_n$$

if $1 \leq i \leq n-1$ and $p_i \rightarrow q_1 q_2$ in Λ . For $p_n \rightarrow q_1 q_2$ in Λ ,

$$[p_1 \cdots p_{n-1} p_n] \rightarrow [p_1 \cdots p_{n-1} q_1] q_2.$$

Likewise, rules in (6) and (7) are modified. Using the general version of \mathcal{G}' , we can show that $(\rightarrow_{\mathcal{R}})[\mathcal{L}(\mathcal{G})]$ is a monotone language for $\mathcal{R} = \{w \rightarrow c\}$.

Lemma 9. *Given an A-TA \mathcal{A}/\mathcal{A} and a ground TRS $\mathcal{R} = \{f(a, b) \rightarrow c\}$ over the same signature \mathcal{F} with $\mathcal{F}_A = \{f\}$, we can compute an A-TA that recognizes $(\rightarrow_{\mathcal{R}/A})[\mathcal{L}(\mathcal{A}/\mathcal{A})]$.*

Proof. Suppose $\mathcal{M}_A = (\mathcal{F} - \{f\}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$ is an s-TA associated with \mathcal{A}/\mathcal{A} . Without loss of generality, we assume that for all $a \rightarrow p, b \rightarrow q \in \Delta$ with $a, b \in \mathcal{F}_0$, $a \neq b$ implies $p \neq q$. Let \diamond be a fresh symbol with $\diamond \notin \mathcal{F} \cup \mathcal{Q}$ and $W = \{pq \mid a \rightarrow p, b \rightarrow q \in \Delta\}$. We write \rightarrow_c for the binary relation induced by the string rewrite system (SRS for short) $\{w \rightarrow \diamond \mid w \in W\}$ over the language $(\mathcal{Q} \cup \{\diamond\})^+$. Similarly, \rightarrow_\diamond for the induced binary relation of the SRS $\{p \rightarrow p_\diamond \mid p \in \mathcal{Q}\}$. Define $\mathcal{M}'_A = (\mathcal{F} - \{f\}, \mathcal{Q}', \mathcal{Q}'_{fin}, \Delta')$ as follows: $\mathcal{Q}' = \mathcal{Q} \cup \{\diamond\} \cup \{q_\diamond \mid q \in \mathcal{Q}\}$, $\mathcal{Q}'_{fin} = \{q_\diamond \mid q \in \mathcal{Q}_{fin}\}$ and

$$\begin{aligned} \Delta' = & \Delta \cup \{c \rightarrow \diamond\} \\ & \cup \{c \rightarrow q_\diamond \mid \langle X \rangle \rightarrow q \Leftarrow \text{concat}(X) \text{ in } L \in \Delta, L \cap W \neq \emptyset\} \\ & \cup \{f(p_1, \dots, p_{i_\diamond}, \dots, p_n) \rightarrow q_\diamond \mid f(p_1, \dots, p_i, \dots, p_n) \rightarrow q \in \Delta\} \\ & \cup \{\langle X \rangle \rightarrow q_\diamond \Leftarrow \text{concat}(X) \text{ in } (\rightarrow_\diamond)[L] \mid \langle X \rangle \rightarrow q \Leftarrow \text{concat}(X) \text{ in } L \in \Delta\} \\ & \cup \left\{ \langle X \rangle \rightarrow q_\diamond \Leftarrow \text{concat}(X) \text{ in } (\rightarrow_c)[L'] \mid \left. \begin{array}{l} \langle X \rangle \rightarrow q \Leftarrow \text{concat}(X) \text{ in } L \in \Delta \\ L' = L - W \end{array} \right\} \end{aligned}$$

By Lemma 8, a monotone grammar generating $(\rightarrow_c)[L']$ is computable. Furthermore, for every $\langle X \rangle \rightarrow q \Leftarrow \text{concat}(X)$ in L in Δ , a monotone grammar generating $(\rightarrow_\diamond)[L]$ is computable. So it can be proved by the structural induction that the s-TA \mathcal{M}'_A recognizes the sequence-term representation of $(\rightarrow_{\mathcal{R}/A})[\mathcal{L}(\mathcal{A}/\mathcal{A})]$. \square

Theorem 4. *Given a ground A-TRS \mathcal{R}/A over the signature \mathcal{F} with $\mathcal{F}_A = \{f\}$: (1) For every A-TA \mathcal{A}/\mathcal{A} over the same signature \mathcal{F} , we can compute an A-TA that recognizes $(\rightarrow_{\mathcal{R}/A})[\mathcal{L}(\mathcal{A}/\mathcal{A})]$. (2) In case that \mathcal{A} is regular, we can compute a regular A-TA that recognizes $(\rightarrow_{\mathcal{R}/A})[\mathcal{L}(\mathcal{A}/\mathcal{A})]$.*

Proof. For (1), let $l \rightarrow r$ be a rewrite rule in \mathcal{R} . We define the ground TRS

$$\begin{aligned} \mathcal{R}_{l \rightarrow r} = & \{f(c_{t_1}^{p_1}, \dots, c_{t_n}^{p_n}) \rightarrow c_{f(t_1, \dots, t_n)}^p \mid p \in \text{pos}(l), l|_p = f(t_1, \dots, t_n)\} \\ & \cup \{d_{f(t_1, \dots, t_n)}^p \rightarrow f(d_{t_1}^{p_1}, \dots, d_{t_n}^{p_n}) \mid p \in \text{pos}(r), r|_p = f(t_1, \dots, t_n)\} \end{aligned}$$

It can be proved that for every s, t in $\mathcal{T}(\mathcal{F})$, $s = C[l]$ and $t = C[r]$ if and only if $s \xrightarrow{+}_{\mathcal{R}_{l \rightarrow r}} C[c_l^\epsilon]$ and $C[d_r^\epsilon] \xrightarrow{+}_{\mathcal{R}_{l \rightarrow r}} t$. Moreover, this statement is generalized as follows: $s \sim_A C[l]$ and $t \sim_A C[r]$ if and only if $s \xrightarrow{+}_{\mathcal{R}_{l \rightarrow r}/A} C[c_l^\epsilon]$

```

 $\mathcal{A}_0 := \mathcal{A}; i := 0; j := 0;$ 
 $S := \text{pos}(l);$ 
 $T := \text{pos}(r);$ 
while  $S \neq \emptyset$  do
  select  $p \in S$  such that  $\nexists p' \in S. p' \succ p$ 
  let  $l|_p = f(t_1, \dots, t_n)$  and
  compute  $\mathcal{A}_{i+1}/A$  such that
     $\mathcal{L}(\mathcal{A}_{i+1}/A) = (\rightarrow_{\{f(c_{t_1}^{p-1}, \dots, c_{t_n}^{p-n}) \rightarrow c_{l|_p}^p\}/A})[\mathcal{L}(\mathcal{A}_i/A)]$ 
   $i := i + 1;$ 
   $S := S - \{p\};$ 
od
compute  $\mathcal{B}_0/A$  such that
   $\mathcal{L}(\mathcal{B}_0/A) = (\rightarrow_{\{c_l^\epsilon \rightarrow d_r^\epsilon\}/A})[\mathcal{L}(\mathcal{A}_i/A)]$ 
while  $T \neq \emptyset$  do
  select  $q \in T$  such that  $\nexists q' \in T. q \succ q'$ 
  let  $r|_q = f(t_1, \dots, t_n)$  and
  compute  $\mathcal{B}_{j+1}/A$  such that
     $\mathcal{L}(\mathcal{B}_{j+1}/A) = (\rightarrow_{\{d_{r|_q}^q \rightarrow f(d_{t_1}^{q-1}, \dots, d_{t_n}^{q-n})\}/A})[\mathcal{L}(\mathcal{B}_j/A)]$ 
   $j := j + 1;$ 
   $T := T - \{q\};$ 
od
 $\mathcal{B}_{l \rightarrow r} := \mathcal{B}_j;$ 
return  $\mathcal{B}_{l \rightarrow r}/A$ 

```

Fig. 3. One-Step A-Rewrite Descendants for One-Rule Case

and $C[d_r^\epsilon] \xrightarrow{+}_{\mathcal{R}_{l \rightarrow r}/A} t$. Using this fact, we define the procedure in Fig. 3. If the root symbol f of a decomposed rule $f(c_1, \dots, c_n) \rightarrow c$ is an A-symbol, we know that the one-step A-rewrite descendants are effectively recognizable by Lemma 9; otherwise, the one-step A-rewrite descendants $(\rightarrow_{\mathcal{R}_{l \rightarrow r}/A})[L]$ are the same as $\mathbf{A}((\rightarrow_{\mathcal{R}_{l \rightarrow r}})[L])$, which has been noted by Dauchet and Tison [5]. Then the procedure is capable of computing an A-TA $\mathcal{B}_{l \rightarrow r}/A$ that recognizes the tree language

$$L_{l \rightarrow r} = \left\{ t \in \mathcal{T}(\mathcal{F}) \mid s \in \mathcal{L}(\mathcal{A}/A), s \xrightarrow{+}_{\mathcal{R}_{l \rightarrow r}/A} C[c_l^\epsilon] \text{ and } C[d_r^\epsilon] \xrightarrow{+}_{\mathcal{R}_{l \rightarrow r}/A} t \right\}.$$

Since $(\rightarrow_{\mathcal{R}/A})[\mathcal{L}(\mathcal{A}/A)] = \bigcup_{l \rightarrow r \in \mathcal{R}} L_{l \rightarrow r}$ and tree languages recognizable with A-TA are effectively closed under union, we can compute an A-TA that recognizes $(\rightarrow_{\mathcal{R}/A})[\mathcal{L}(\mathcal{A}/A)]$.

The second statement can be shown in a similar way. First we prove that context-free languages are closed under one-step string rewriting of $\{\mathbf{a b} \rightarrow \mathbf{c}\}$, that corresponds to Lemma 8. Then we apply the result to the A-tree case. In

the construction of context-free grammars, we use different transition rules. The set of transition rules \mathcal{A}' of \mathcal{G}' consists of (1), (3), (4) and the following rules:

$$\left\{ \begin{array}{l} s_0 \rightarrow [p q] r \\ s_0 \rightarrow p [q r] \end{array} \middle| p, q, r \in \mathcal{S} \text{ such that } s_0 \rightarrow_{\mathcal{G}}^* p q r \right\} \dots (2')$$

$$\left\{ \begin{array}{l} [p q] \rightarrow [p_1 p_2] q \\ [p q] \rightarrow p_1 [p_2 q] \\ [q p] \rightarrow [q p_1] p_2 \\ [q p] \rightarrow q [p_1 p_2] \\ p \rightarrow p_1 p_2 \end{array} \middle| p \rightarrow p_1 p_2 \in \Lambda, p \neq s_0, q \in \mathcal{S} \right\} \dots (5')$$

In this case, transition rules (6) and (7) in Lemma 8 are unnecessary. \square

5 Concluding Remarks

We have shown in this paper that A-tree languages recognizable with A-tree automata are closed under boolean operations. The newly obtained closure property is a direct consequence of (1) complement closedness of monotone sequence-tree languages (Theorem 3), and (2) bijective correspondence between A-tree automata and sequence-tree automata (Theorem 1). The theorem is also helpful for simplifying the proof of undecidability of the emptiness problem for A-tree automata. The new framework introduced in Section 2, called sequence-tree automata, enables us to have an easy proof of the complexity of the membership problem for A-tree languages (Corollary 1).

In the previous section, we also showed that recognizable A-tree languages are closed under one-step ground A-rewrite descendants (Theorem 4). This allows us to provide an *under-approximation* algorithm for computing A-rewrite descendants of A-tree languages with arbitrary accuracy, which is useful in practice, e.g. for infinite-state model checking [7].

In this paper, we introduced a special symbol $\langle \rangle$ for translating A-tree languages of a singleton \mathcal{F}_A case. But, for many arbitrary associative symbols, the sequence-term model has to be allowed to contain more special symbols, e.g. $\langle \rangle_1, \dots, \langle \rangle_n$. More precisely, by modifying the definition of sequence-terms and indexing the special symbol $\langle \rangle$ as above, Theorems 1–4 and Corollaries 1–2 can be extended to the general case.

By restricting grammars in the conditional part of transition rules, as mentioned already, we obtain the language hierarchy, which is illustrated in Fig. 4. The outermost square represents the class of sequence-tree languages recognizable with the context-sensitive s-TA. The second largest square is the class of context-free s-TA. Since it has the bijective correspondence to regular A-TA, the class in the usual term model is the same as the A-congruence closure of regular tree languages. The third largest square, i.e. the class of regular s-TA, is identical to multi-tree automata for associativity axioms only [14]. This class and the class of CS s-TA are closed under boolean operations and the A-congruence relation. Furthermore, the third class is important in the following sense:

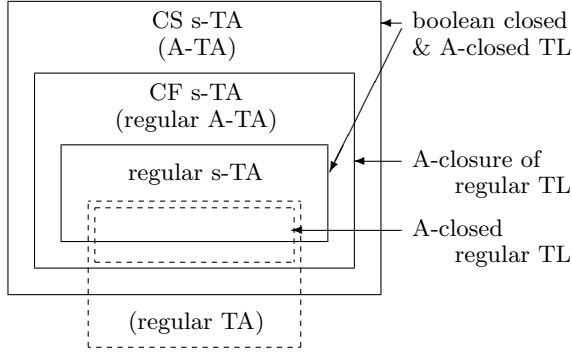


Fig. 4. Relationships in sequence-tree automata

Theorem 5. *Given a ground A-TRS \mathcal{R}/A over the signature \mathcal{F} with $\mathcal{F}_A = \{f\}$, the set of normal forms with respect to \mathcal{R}/A is effectively recognizable with regular A-TA. In the sequence-term model, the language is effectively recognizable with regular s-TA.*

Proof. Let $l \rightarrow r$ be a rewrite rule in \mathcal{R} . We define an s-TA $\mathcal{M}_{l \rightarrow r}$ that recognizes the sequence-term representation of the A-closure of terms $C[l]$, where C is an arbitrary ground context: $\mathcal{M}_{l \rightarrow r} = (\mathcal{F} - \{f\}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$ such that $\mathcal{Q} = \{q\} \cup \{q_t \mid p \in \text{pos}(l), t = l|_p\}$, $\mathcal{Q}_{fin} = \{q_l\}$ and Δ consists of the transition rules

$$\langle X \rangle \rightarrow q_t \Leftarrow \text{concat}(X) \text{ in } \{q_{t_1} \cdots q_{t_n}\} \quad \text{if } t = C_f[[t_1, \dots, t_n]] \text{ is an } f\text{-block of } l,$$

$$f(q_{t_1}, \dots, q_{t_m}) \rightarrow q_t \quad \text{if } t = f(t_1, \dots, t_m) \trianglelefteq l \text{ and } f \neq f$$

together with $\langle X \rangle \rightarrow q_l \Leftarrow \text{concat}(X) \text{ in } \{w \mid w = q^* q_l q^* \text{ and } |w| \geq 2\}$, $\langle X \rangle \rightarrow q \Leftarrow \text{concat}(X) \text{ in } \{w \mid w = q^* \text{ and } |w| \geq 2\}$, $f(q, \dots, q_l, \dots, q) \rightarrow q_l$ and $f(q, \dots, q) \rightarrow q$ for all $f \in \mathcal{F} - \{f\}$. Observe that the membership test in each conditional rule can be represented by a regular grammar. Let $\mathcal{M}_{\mathcal{R}}$ be an s-TA that recognizes $\bigcup_{l \rightarrow r \in \mathcal{R}} \mathcal{L}(\mathcal{M}_{l \rightarrow r})$. The (sequence-term representation of) set of normal forms of \mathcal{R}/A is the complement of $\mathcal{L}(\mathcal{M}_{\mathcal{R}})$. Determinizing $\mathcal{M}_{\mathcal{R}}$ with subset construction results in a regular s-TA $\mathcal{M}'_{\mathcal{R}}$, and it is computable, because the conditional part of a transition rule of $\mathcal{M}'_{\mathcal{R}}$ is represented as follows: $\text{concat}(X) \text{ in } \bigcap_{i \in I} \mathcal{L}(\mathcal{G}_i) - \bigcup_{i \in (K-I)} \mathcal{L}(\mathcal{G}_i)$, where \mathcal{G}_i is a regular grammar for every $i \in (K \cup I)$. Therefore, the set of normal forms of \mathcal{R}/A is effectively recognizable with regular A-TA. \square

The larger dotted square in Fig. 4 denotes the class of regular tree languages. This square contains the subclass allowed to be A-closed (and thus, it is both regular and A-closed). Since it can be shown that every regular and A-closed tree language is recognizable with regular A-TA, the innermost dotted square has no overlapping with the class of CS s-TA, i.e., there is no example that is regular, A-closed, and recognizable with A-TA, but not recognizable with regular A-TA.

Acknowledgments. The authors are grateful to Aart Middeldorp for his comments on the early draft. We also thank anonymous referees for their detailed comments.

References

1. F. Baader and T. Nipkow: *Term Rewriting and All That*, Cambridge University Press, 1998.
2. L. Bachmair and D.A. Plaisted: *Associative Path Orderings*, Proc. of 1st RTA, Dijon (France), LNCS 202, pp. 241–254, 1985.
3. M. Bezem, R. de Vrijer, and J.W. Klop (eds.): *Term Rewriting Systems*, Cambridge Tracts in Theoretical Computer Science 55, Cambridge University Press, 2003.
4. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi: *Tree Automata Techniques and Applications*, 2002. Draft available on <http://www.grappa.univ-lille3.fr/tata/>
5. M. Dauchet and S. Tison: *The Theory of Ground Rewrite Systems is Decidable*, Proc. of 5th LICS, Philadelphia (Pennsylvania), pp. 242–248, 1990.
6. 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.
7. A. Finkel and Ph. Schnoebelen: *Well-Structured Transition Systems Everywhere!*, Theoretical Computer Science 256, pp. 63–92, 2001.
8. F. Gécseg and M. Steinby: *Tree Languages*, Handbook of Formal Languages 3, pp. 1–68 (Chapter 1), Springer-Verlag, 1996.
9. J. Goubault-Larrecq and K.N. Verma: *Alternating Two-Way AC-Tree Automata*, Research Report LSV-02-11, ENS de Cachan (France), 2002.
10. M. Hamana: *Term Rewriting with Sequences*, Proc. of 1st Theorema Workshop, Hagenberg (Austria), 1997. Draft included in technical report 97–20, RISC - Linz.
11. N. Immerman: *Nondeterministic Space is Closed Under Complementation*, SIAM Journal on Computing 17(5), pp. 935–938, 1988.
12. S.Y. Kuroda: *Classes of Languages and Linear Bounded Automata*, Information and Control 7, pp. 207–223, 1964.
13. T. Kutsia: *Unification in the Empty and Flat Theories with Sequence Variables and Flexible Arity Symbols*, technical report 01–13, RISC - Linz, 2001. Available on <http://www.sfb013.uni-linz.ac.at/~sfb/reports/2001/ps-files/>
14. D. Lugiez and S. Dal Zilio: *Multitrees Automata, Presburger’s Constraints and Tree Logics*, technical report 08–2002, LIF - CNRS - Université Provence, 2002. Available on <http://www.lim.univ-mrs.fr/Rapports/>
15. A. Mateescu and A. Salomaa: *Aspects of Classical Language Theory*, Handbook of Formal Languages 1, pp. 175–251 (Chapter 4), Springer-Verlag, 1996.
16. H. Ohsaki: *Beyond Regularity: Equational Tree Automata for Associative and Commutative Theories*, Proc. of 15th CSL, Paris (France), LNCS 2142, pp. 539–553, 2001.
17. H. Ohsaki and T. Takai: *Decidability and Closure Properties of Equational Tree Languages*, Proc. of 13th RTA, Copenhagen (Denmark), LNCS 2378, pp. 114–128, 2002.
18. H. Ohsaki and T. Takai: *A Tree Automata Theory for Unification Modulo Equational Rewriting*, 16th UNIF, Copenhagen (Denmark), 2002. Draft available from <http://staff.aist.go.jp/hitoshi.ohsaki/unif2002.ps.gz>
19. H. Ohsaki, H. Seki, and T. Takai: *Recognizable A-Tree Languages are Boolean Closed*, technical report AIST-PS-2003-001, Programming Science Group, AIST, 2003. The revised version with the new title (*Recognizing Boolean Closed A-Tree Languages with Membership Conditional Rewriting Mechanism*) is available from <http://staff.aist.go.jp/hitoshi.ohsaki/rta2003.ps.gz>
20. Y. Toyama: *Membership Conditional Term Rewriting Systems*, Trans. of IEICE E72(11), pp. 1224–1229, 1989.