Recognizable A-Tree Languages are Boolean Closed

Hitoshi Ohsaki[1,2], Hiroyuki Seki[3], and Toshinori Takai[2]

[1]PRESTO "Information and Systems"
Japan Science and Technology Corporation

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

[3]Nara 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 [15]). 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 [13] 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 with deep interest: 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, through 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 [13]. 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 $\mathsf{f}(\mathsf{a},\ldots,\mathsf{a},\mathsf{b},\ldots,\mathsf{b})$, 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 $\mathsf{f}(\mathsf{q}_\mathsf{a},\mathsf{q}_\mathsf{b}) \to \mathsf{q}_\mathsf{f}$, $\mathsf{f}(\mathsf{q}_\mathsf{a},\mathsf{q}_\mathsf{f}) \to \mathsf{q}$, $\mathsf{f}(\mathsf{q},\mathsf{q}_\mathsf{b}) \to \mathsf{q}_\mathsf{f}$.

The past couple of years we have investigated in [15, 16] 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. Unfortunately, there is also a negative result on A-tree languages: regular A-tree languages (i.e, A-closure of 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 regular A-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 $\mathsf{f}(\mathsf{a},\mathsf{f}(\mathsf{f}(\mathsf{b},\mathsf{c}),\mathsf{d}))$ is interpreted as $\langle\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d}\rangle$. 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 [17]. 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 $\{\, s \mid t \to^*_{\mathcal{R}/\mathsf{A}} s \,\}$ can not be handled with a decidable tree automata theory [6], we show in the paper that $\{\, s \mid \exists t \in L$ such that $t \to_{\mathcal{R}/\mathsf{A}} s \,\}$ is recognizable with A-tree automata, provided L is recognizable with A-tree automata and $\mathcal{R}/\mathsf{A}$ 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).

In the rest of this section, we fix our terminologies on term rewriting and tree automata. In the paper, we assume the readers are familiar with the basics of term rewriting (explained, e.g., in [1, 3]). A subset $\mathcal{F}_\mathrm{A}$ 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

$\mathcal{F}_{\mathrm{A}}$. An *associative term rewriting system* (A-TRS for short) $\mathcal{R}/\mathsf{A}$ over $\mathcal{F}$ is the combination of a TRS $\mathcal{R}$ and an ES A. We write $s \to_{\mathcal{R}/\mathsf{A}} t$ if there exist terms s' and t' such that $s \sim_{\mathsf{A}} s' \to_{\mathcal{R}} t' \sim_{\mathsf{A}} t$. The binary relation $\sim_{\mathsf{A}}$ is the equivalence relation induced by A. The reflexive-transitive closure and the transitive closure of $\to_{\mathcal{R}/\mathsf{A}}$ are denoted by $\to^*_{\mathcal{R}/A}$ and $\to^+_{\mathcal{R}/\mathsf{A}}$, respectively. A term s reachable from a term t with respect to $\mathcal{R}/\mathsf{A}$, i.e. $t \to^*_{\mathcal{R}/\mathsf{A}} s$, is called an $\mathcal{R}/\mathsf{A}$-*descendant* of t. A term s with $s \to_{\mathcal{R}/\mathsf{A}} t$ is called a *one-step* $\mathcal{R}/\mathsf{A}$-*descendant* of t. The sets $\{\, s \mid \exists t \in L.\ t \to^*_{\mathcal{R}/\mathsf{A}} s \,\}$ and $\{\, s \mid \exists t \in L.\ t \to_{\mathcal{R}/\mathsf{A}} s \,\}$ for some set L of terms are denoted by $(\to^*_{\mathcal{R}/\mathsf{A}})[L]$ and $(\to_{\mathcal{R}/\mathsf{A}})[L]$, respectively.

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 $\mathcal{F} \cap \mathcal{Q} = \varnothing$), a set $\mathcal{Q}_{fin}$ $(\subseteq \mathcal{Q})$ of so-called final state symbols, and a finite set Δ of transition rules in one of the following forms:

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

for some $f \in \mathcal{F}$ with $\mathsf{arity}(f) = n$ and $p_1, \ldots, p_n, q, q_1, \ldots, q_n \in \mathcal{Q}$. 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) $\mathcal{A}/\mathsf{A}$ is the combination of a TA $\mathcal{A}$ and an ES A over the same signature $\mathcal{F}$ with $\mathcal{F}_{\mathrm{A}}$. An A-TA $\mathcal{A}/\mathsf{A}$ is called *regular* if Δ consists only of rules in the former shape $f(p_1, \ldots, p_n) \to q$.

An A-TA $\mathcal{A}/\mathsf{A}$ is a special A-TRS. In fact, Δ/A defines an A-TRS over the signature $\mathcal{F} \cup \mathcal{Q}$. We write $s \to_{\mathcal{A}/\mathsf{A}} t$ if $s \to_{\Delta/\mathsf{A}} t$. The binary relation $\to_{\mathcal{A}/\mathsf{A}}$ over $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ is called the *move relation* of $\mathcal{A}/\mathsf{A}$. A term t in $\mathcal{T}(\mathcal{F})$ is *accepted* by $\mathcal{A}/\mathsf{A}$ if $t \to^*_{\mathcal{A}/\mathsf{A}} q$ for some $q \in \mathcal{Q}_{fin}$. The set of terms accepted by $\mathcal{A}/\mathsf{A}$ is denoted by $\mathcal{L}(\mathcal{A}/\mathsf{A})$. 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 $\mathcal{A}/\mathsf{A}$ such that $L = \mathcal{L}(\mathcal{A}/\mathsf{A})$.

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 *sequence-terms*: Given the signature $\mathcal{F}$ and the set $\mathcal{V}$ of variables, we say t is a sequence-term (or sequence-tree) in s-$\mathcal{T}(\mathcal{F}, \mathcal{V})$

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

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 $\mathsf{arity} : \mathcal{F} \to \mathbb{N}$. A subset $\mathcal{F}_n$ of $\mathcal{F}$ denotes the set of function symbols f with $\mathsf{arity}(f) = n$.

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:

$$\langle X \rangle \to q \Leftarrow \mathsf{leaf}(X) \text{ in } \mathcal{L}(\mathcal{G})$$

Here the capital letter X is called a *sequence-variable* [10, 12], whose domain is an arbitrary sequence of sequence-terms $t_1, \ldots, t_n$ with $n \geqslant 2$. In other words, the left-hand side represents $\langle x_1, \ldots, x_n \rangle$ for some $x_1, \ldots, x_n \in \mathcal{V}$, but n is not fixed except $n \geqslant 2$. If x_i is instantiated to a state q_i $(1 \leqslant i \leqslant n)$, the sequence of state symbols $q_1, \ldots, q_n$, denoted by $\mathsf{leaf}(X)$, is examined in the *conditional part* whether it is accepted by a *word grammar* $\mathcal{G}$ over $\mathcal{Q}$. Grammars in conditional rules are not necessarily the same for different two rules. If the membership condition $\mathsf{leaf}(X)$ in $\mathcal{L}(\mathcal{G})$ is satisfied, the left pattern is replaced by a state q.

A word grammar $\mathcal{G} = (\Sigma, \mathcal{S}, \mathsf{s}_0, \Lambda)$ over the alphabet Σ with $\mathcal{S}$ nonterminal symbols and s_0 the starting symbol is called (1) *context-sensitive* (CSG for short) if $|l| \leqslant |r|$, (2) *context-free* if $l \in \mathcal{S}$, (3) *regular* if $l \in \mathcal{S}$ and $r \in \{\, a\, s \mid a \in \Sigma \text{ and } s \in \mathcal{S} \,\} \cup \{\, a \mid a \in \Sigma \,\}$ for all production rules $l \to r$ in Λ. We say $\mathcal{G}$ is a *monotone* grammar if every rule in Λ has one of the following forms:

$$p \to a \qquad \text{or} \qquad p \to q_1\, q_2 \qquad \text{or} \qquad p_1\, p_2 \to q_1\, q_2$$

with $p, p_1, p_2, q_1, q_2 \in \mathcal{S}$ and $a \in \Sigma$. In the literature, e.g. in [14], 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 [11].

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). 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 $\mathcal{F}_{\mathrm{A}} = \{\, \mathsf{f} \,\}$. 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 $C'[C_{\mathsf{f}}[\![t_1, \ldots, t_n]\!]]$ for $C'[C[t_1, \ldots, t_n]]$ if C is an f-block. Then, $\mathsf{g}(C_{\mathsf{f}}[\![\mathsf{a}, \mathsf{b}, \mathsf{c}]\!])$ represents the terms $\mathsf{g}(\mathsf{f}(\mathsf{a}, \mathsf{f}(\mathsf{b}, \mathsf{c})))$ and $\mathsf{g}(\mathsf{f}(\mathsf{f}(\mathsf{a}, \mathsf{b}), \mathsf{c}))$.

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

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

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

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

An example of a flat-transformation is illustrated in Fig. 1. One should no-

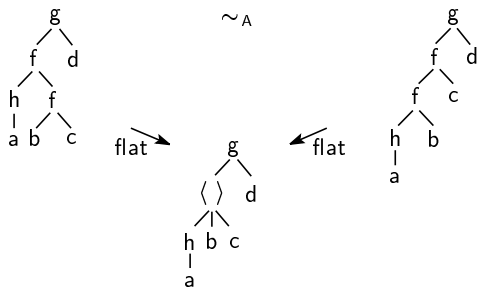

Fig. 1. Flattening by flat

tice that $\mathsf{flat}(\mathsf{unflat}(t)) = t$ for any sequence-term t in s-$\mathcal{T}(\mathcal{F}', \mathcal{V})$. However, $\mathsf{unflat}(\mathsf{flat}(t))$ and t are not the same, but $\mathsf{unflat}(\mathsf{flat}(t)) \sim_{\mathsf{A}} t$. For instance,

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

although $\mathsf{g}(\mathsf{f}(\mathsf{f}(\mathsf{h}(\mathsf{a}), \mathsf{b}), \mathsf{c}), \mathsf{d}) \sim_{\mathsf{A}} \mathsf{g}(\mathsf{f}(\mathsf{h}(\mathsf{a}), \mathsf{f}(\mathsf{b}, \mathsf{c})), \mathsf{d})$.

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 $\mathcal{F}_{\mathrm{A}} = \{\mathsf{f}\}$. There exists an A-TA $\mathcal{A}/\mathsf{A}$ such that $\mathcal{L}(\mathcal{A}/\mathsf{A}) = L$ if and only if there exists an s-TA $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M}) = \mathsf{flat}(L)$.* □

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

Let us explain the idea for the proof of the "only if" part of the above theorem. We consider the A-TA $\mathcal{A}/\mathsf{A}$ with $\mathcal{A} = (\{\mathsf{f}, \mathsf{a}, \mathsf{b}\}, \{\mathsf{p}, \mathsf{q}_1, \mathsf{q}_2\}, \{\mathsf{p}\}, \Delta)$ where Δ :

$$\begin{array}{lll} \mathsf{a} \to \mathsf{q}_1 & \mathsf{f}(\mathsf{q}_1, \mathsf{q}_2) \to \mathsf{p} & \mathsf{f}(\mathsf{q}_1, \mathsf{q}_2) \to \mathsf{f}(\mathsf{q}_2, \mathsf{q}_1) \\ \mathsf{b} \to \mathsf{q}_2 & \mathsf{f}(\mathsf{p}, \mathsf{p}) \to \mathsf{p} & \mathsf{f}(\mathsf{q}_2, \mathsf{q}_1) \to \mathsf{f}(\mathsf{q}_1, \mathsf{q}_2) \end{array}$$

The A-TA $\mathcal{A}/\mathsf{A}$ recognizes the tree language $\{t \in \mathcal{T}(\{\mathsf{f}, \mathsf{a}, \mathsf{b}\}) \mid |t|_{\mathsf{a}} = |t|_{\mathsf{b}}\}$, 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 $\mathcal{M}_{\mathcal{A}/\mathsf{A}} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta_1 \cup \Delta_2)$ as follows:

$$\begin{array}{rl} \mathcal{F}: & \mathsf{a}, \mathsf{b} \\ \mathcal{Q}: & \mathsf{p}, \mathsf{q}_1, \mathsf{q}_2 \\ \mathcal{Q}_{fin}: & \mathsf{p} \\ \Delta_1: & \mathsf{a} \to \mathsf{q}_1,\ \mathsf{b} \to \mathsf{q}_2 \\ \Delta_2: & \langle X \rangle \to \mathsf{p} \Leftarrow \mathsf{leaf}(X) \text{ in } \mathcal{L}(\mathcal{G}_{\mathsf{p}}) \end{array}$$

where $\mathcal{G}_{\mathsf{p}} = (\{\mathsf{p}, \mathsf{q}_1, \mathsf{q}_2\}, \{\alpha_{\mathsf{p}}, \alpha_{\mathsf{q}1}, \alpha_{\mathsf{q}2}\}, \alpha_{\mathsf{p}}, \Lambda)$ and

$$\Lambda = \{\alpha_{\mathsf{p}} \to \alpha_{\mathsf{q}1}\,\alpha_{\mathsf{q}2},\ \alpha_{\mathsf{p}} \to \alpha_{\mathsf{p}}\,\alpha_{\mathsf{p}},\ \alpha_{\mathsf{q}2}\,\alpha_{\mathsf{q}1} \to \alpha_{\mathsf{q}1}\,\alpha_{\mathsf{q}2},\ \alpha_{\mathsf{q}1}\,\alpha_{\mathsf{q}2} \to \alpha_{\mathsf{q}2}\,\alpha_{\mathsf{q}1}\} \cup \{\alpha_{\mathsf{p}} \to \mathsf{p},\ \alpha_{\mathsf{q}_1} \to \mathsf{q}_1,\ \alpha_{\mathsf{q}2} \to \mathsf{q}_2\}.$$

In the construction, a monotone grammar $\mathcal{G}_{\mathsf{p}}$ has a production rule of the form $\alpha_p \to \alpha_q\,\alpha_r$ if Δ contains a transition rule $\mathsf{f}(q,r) \to p \in \Delta$. Likewise, $\mathcal{G}_{\mathsf{p}}$ has a rule $\alpha_p\,\alpha_q \to \alpha_r\,\alpha_s$ if there exists $\mathsf{f}(r,s) \to \mathsf{f}(p,q)$ in Δ. As a consequence, we obtain $\mathcal{M}_{\mathcal{A}/\mathsf{A}}$ such that it allows a one-step move $\langle p_1,\ldots,p_n\rangle \to_{\mathcal{M}_{\mathcal{A}/\mathsf{A}}} \mathsf{p}$ if and only if $\alpha_{\mathsf{p}} \to^*_{\mathcal{G}_{\mathsf{p}}} \alpha_{p1}\cdots\alpha_{pn}$. In fact, the above s-TA construction satisfies the following property:

Lemma 1. *Given an A-TA $\mathcal{A}/\mathsf{A}$ with $\mathcal{A} = (\mathcal{F},\mathcal{Q},\mathcal{Q}_{fin},\Delta)$. Suppose $\mathcal{M}_{\mathcal{A}/\mathsf{A}} = (\mathcal{F}',\mathcal{Q}',\mathcal{Q}'_{fin},\Delta')$ is the s-TA obtained by the above construction. Then, for all $p_1,\ldots,p_n,q \in \mathcal{Q}$, $C_{\mathsf{f}}[\![p_1,\ldots,p_n]\!] \to^*_{\mathcal{A}/\mathsf{A}} q$ if and only if $\alpha_q \to^*_{\mathcal{G}_q} \alpha_{p1}\cdots\alpha_{pn}$.* □

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

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

Proof. We show the "only if" part. The reverse can be proved in a similar way. We assume $t \to^*_{\mathcal{A}/\mathsf{A}} q$ for some $q \in \mathcal{Q}$. Then we show by the structural induction that $t \to^*_{\mathcal{M}_{\mathcal{A}/\mathsf{A}}} q$. If t is a constant c, there exists a transition rule $c \to q \in \Delta$. Then, by construction, $c \to q \in \Delta'$. If $t = f(t_1,\ldots,t_n)$ with $f \neq \mathsf{f}$, then $t_i \to^*_{\mathcal{A}/\mathsf{A}} p_i$ for some $p_i \in \mathcal{Q}$ $(1 \leqslant i \leqslant n)$ such that $f(p_1,\ldots,p_n) \to q \in \Delta$. By induction hypothesis, $\mathsf{flat}(t_i) \to^*_{\mathcal{M}_{\mathcal{A}/\mathsf{A}}} p_i$ $(1 \leqslant i \leqslant n)$. Moreover, $f(p_1,\ldots,p_n) \to q \in \Delta'$. Thus, $\mathsf{flat}(t) \to^*_{\mathcal{M}_{\mathcal{A}/\mathsf{A}}} f(p_1,\ldots,p_n) \to_{\mathcal{M}_{\mathcal{A}/\mathsf{A}}} q$. If $t = C_{\mathsf{f}}[\![t_1,\cdots,t_m]\!]$, then $t_i \to^*_{\mathcal{A}/\mathsf{A}} p_i$ for all $1 \leqslant i \leqslant m$ and $C_{\mathsf{f}}[\![p_1,\ldots,p_m]\!] \to^*_{\mathcal{A}/\mathsf{A}} q$. By induction hypothesis, $\mathsf{flat}(t_i) \to^*_{\mathcal{M}_{\mathcal{A}/\mathsf{A}}} p_i$ $(1 \leqslant i \leqslant n)$. Thus,

$$\mathsf{flat}(t) = \langle \mathsf{flat}(t_1),\ldots,\mathsf{flat}(t_m)\rangle \to^*_{\mathcal{M}_{\mathcal{A}/\mathsf{A}}} \langle p_1,\ldots,p_m\rangle \to_{\mathcal{M}_{\mathcal{A}/\mathsf{A}}} q,$$

because there exists a transition rule $\langle X\rangle \to q \Leftarrow \mathsf{leaf}(X)\ \mathsf{in}\ \mathcal{L}(\mathcal{G}_q)$. By Lemma 1, $p_1\cdots p_m \in \mathcal{L}(\mathcal{G}_q)$ is guaranteed. □

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 $\langle X\rangle \to q \Leftarrow \mathsf{leaf}(X)\ \mathsf{in}\ \mathcal{L}(\mathcal{G})$, nonterminal symbols of $\mathcal{G}$ are pairwise disjoint from the other word grammars. We can now define the associated A-TA $\mathcal{A}_{\mathcal{M}}/\mathsf{A}$ as follows: $\mathcal{A}_{\mathcal{M}} = (\mathcal{F}\cup\{\mathsf{f}\},\mathcal{Q}',\mathcal{Q}_{fin},\Delta')$ and $\mathcal{F}_{\mathrm{A}} = \{\mathsf{f}\}$ where

$$\begin{aligned}
\Delta' = {} & \{\, l \to r \in \Delta \mid \mathsf{root}(l) \in \mathcal{F} \,\} \\
& \cup \{\, \mathsf{f}(\alpha,\beta) \to \mathsf{f}(\gamma,\delta) \mid \langle X\rangle \to q \Leftarrow \mathsf{leaf}(X)\ \mathsf{in}\ \mathcal{L}(\mathcal{G}) \in \Delta,\ \gamma\,\delta \to \alpha\,\beta \in \mathcal{G} \,\} \\
& \cup \{\, \mathsf{f}(\alpha,\beta) \to \gamma \mid \langle X\rangle \to q \Leftarrow \mathsf{leaf}(X)\ \mathsf{in}\ \mathcal{L}(\mathcal{G}) \in \Delta,\quad \gamma \to \alpha\,\beta \in \mathcal{G} \,\} \\
& \cup \{\, \mathsf{f}(\alpha,\beta) \to q \mid \langle X\rangle \to q \Leftarrow \mathsf{leaf}(X)\ \mathsf{in}\ \mathcal{L}(\mathcal{G}) \in \Delta,\quad \gamma \to \alpha\,\beta \in \mathcal{G}, \\
& \qquad\qquad \text{and } \gamma \text{ is a starting symbol of } \mathcal{G} \,\} \\
& \cup \{\, f(p_1,\ldots,p_n) \to \alpha \mid f(p_1,\ldots,p_n) \to p \in \Delta \text{ with } f \neq \mathsf{f}, \\
& \qquad\qquad \langle X\rangle \to q \Leftarrow \mathsf{leaf}(X)\ \mathsf{in}\ \mathcal{L}(\mathcal{G}) \in \Delta,\quad \alpha \to p \in \mathcal{G} \quad \,\}
\end{aligned}$$

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 $\mathsf{f} \notin \mathcal{F}$. Suppose $\mathcal{F}' = \mathcal{F} \cup \mathcal{F}_{\mathrm{A}}$ with $\mathcal{F}_{\mathrm{A}} = \{\mathsf{f}\}$ and $\mathcal{A}_{\mathcal{M}}/\mathsf{A}$ 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 $\mathsf{unflat}(t) \in \mathcal{L}(\mathcal{A}_{\mathcal{M}}/\mathsf{A})$.*

Proof. Similar to the previous lemma, but we prove for the "only if" part that $t \to^*_{\mathcal{M}} q$ implies $\mathsf{unflat}(t) \to^*_{\mathcal{A}_{\mathcal{M}}/\mathsf{A}} q$. We proceed by the structural induction on t. If t is a constant c, there exists a transition rule $c \to q \in \Delta'$. Then, $c \to_{\mathcal{A}_{\mathcal{M}}/\mathsf{A}} q$ by construction. If $t = f(t_1, \ldots, t_n)$ with $f \neq \mathsf{f}$, then $t_i \to^*_{\mathcal{M}} p_i$ for some $p_i \in \mathcal{Q}$ $(1 \leqslant i \leqslant n)$ such that $f(p_1, \ldots, p_n) \to q \in \Delta$. By induction hypothesis, $\mathsf{unflat}(t) \to^*_{\mathcal{A}_{\mathcal{M}}/\mathsf{A}} f(p_1, \ldots, p_n)$. Moreover, $f(p_1, \ldots, p_n) \to q \in \Delta'$. Thus, $\mathsf{unflat}(t) \to^*_{\mathcal{A}_{\mathcal{M}}/\mathsf{A}} f(p_1, \ldots, p_n) \to_{\mathcal{A}_{\mathcal{M}}/\mathsf{A}} q$. If $t = \langle t_1, \ldots, t_m \rangle$, then $t_i \to^*_{\mathcal{M}} p_i$ $(1 \leqslant i \leqslant m)$ and there exists a rule $\langle X \rangle \to q \Leftarrow \mathsf{leaf}(X)$ in $\mathcal{L}(\mathcal{G})$ in Δ such that $p_1 \cdots p_m \in \mathcal{L}(\mathcal{G})$. By assumption, $\mathsf{root}(t_i) \in \mathcal{F}$. Thus, by induction hypothesis, we obtain $\mathsf{unflat}(t) \to^*_{\mathcal{A}_{\mathcal{M}}/\mathsf{A}} C_{\mathsf{f}}[\![p_1, \ldots, p_m]\!]$. On the other hand, along the similar lines of the proof of Lemma 1, we can show that $C_{\mathsf{f}}[\![p_1, \ldots, p_m]\!] \to^*_{\mathcal{A}_{\mathcal{M}}/\mathsf{A}} q$ if and only if $\alpha \to^*_{\mathcal{G}} p_1 \cdots p_m$, provided α is the starting symbol of $\mathcal{G}$. Hence, $\mathsf{unflat}(t) \to^*_{\mathcal{A}_{\mathcal{M}}/\mathsf{A}} q$. □

Bijective correspondence between s-TA and A-TA gives rise to several benefits. For instance, it simplifies the proof of undecidability of the emptiness problem for A-TA (Corollary 1 in [15]). We prove the same undecidability for s-TA below: Given a monotone grammar $\mathcal{G} = (\Sigma, \mathcal{S}, \mathsf{s}_0, \Lambda)$. Then we define the s-TA $\mathcal{M}_{\mathcal{G}} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$ associated with $\mathcal{G}$ as follows. $\mathcal{F} = \{\mathsf{c}_a \mid a \in \Sigma\}$, $\mathcal{Q} = \Sigma \cup \{\mathsf{q}\}$ with $\mathsf{q} \notin \Sigma$, $\mathcal{Q}_{fin} = \{\mathsf{q}\}$, $\Delta = \{\mathsf{c}_a \to a \mid a \in \Sigma\} \cup \{\langle X \rangle \to \mathsf{q} \Leftarrow \mathsf{leaf}(X) \text{ in } \mathcal{L}(\mathcal{G}_{\mathsf{q}})\}$. Then $\mathcal{L}(\mathcal{M}_{\mathcal{G}}) = \varnothing$ if and only if $\mathcal{L}(\mathcal{G}_{\mathsf{q}}) = \varnothing$. 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 [15]). The additional benefit from using Theorem 1 is that a complexity result in one framework implies the same result in the other framework.

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

Proof. It is not difficult to show that the membership problem for s-TA is in PSPACE. Let $\mathcal{M}$ be an s-TA and t be a sequence-term over the same signature $\mathcal{F}$. Then, for every transition rule $\langle X \rangle \to q \Leftarrow \mathsf{leaf}(X) \text{ in } \mathcal{L}(\mathcal{G}_q)$, we can provide a non-deterministic algorithm deciding whether $\mathsf{leaf}(X) \text{ in } \mathcal{L}(\mathcal{G}_q)$. The decision problem is solved in polynomial space, because $\mathcal{G}$ is a monotone grammar. Using the algorithms, we can simulate the behavior of $\mathcal{M}$ in polynomial space of the sizes of $\mathcal{M}$ and t.

PSPACE-hardness is proved by the reduction of the membership problem for CSG. We take an arbitrary CSG $\mathcal{G}$. Due to Kuroda [11], we can compute an equivalent monotone grammar $\mathcal{G}' = (\Sigma, \mathcal{S}, \mathsf{s}_0, \Lambda)$ within a polynomial time of

the size of $\mathcal{G}$. On the other hand, we define $\mathcal{M}_{\mathcal{G}'} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$ as follows: $\mathcal{F} = \{ \mathsf{c}_a \mid a \in \Sigma \}$, $\mathcal{Q} = \Sigma \cup \{ \mathsf{q} \}$, $\mathcal{Q}_{fin} = \{ \mathsf{q} \}$, and $\Delta = \Delta_1 \cup \Delta_2$ where

$\Delta_1 = \{ \mathsf{c}_a \to \mathsf{q} \mid \exists a \in \Sigma$ such that $\mathsf{s}_0 \to a \in \Lambda \}$,

$\Delta_2 = \{ \mathsf{c}_a \to a \mid a \in \Sigma \} \cup \{ \langle X \rangle \to \mathsf{q} \Leftarrow \mathsf{leaf}(X)$ in $\mathcal{L}(\mathcal{G}') \}$.

The s-TA $\mathcal{M}_{\mathcal{G}'}$ satisfies that for every word $w = a_1 \cdots a_n$ over Σ, (1) if $n = 1$ then $w \in \mathcal{L}(\mathcal{G}')$ if and only if $\mathsf{c}_w \in \mathcal{L}(\mathcal{M}_{\mathcal{G}'})$, (2) if $n \geqslant 2$ then $w \in \mathcal{L}(\mathcal{G}')$ if and only if $\langle \mathsf{c}_{a1}, \ldots, \mathsf{c}_{an} \rangle \in \mathcal{L}(\mathcal{M}_{\mathcal{G}'})$. Furthermore, the above construction works in a polynomial time of the size of $\mathcal{G}'$. □

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

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

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 are closed under complementation. Furthermore, due to the bijective correspondence between A-tree automata and sequence-tree automata, recognizable A-tree languages also have the same closure property, which provides a positive answer to an important remaining question in [15, 16].

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

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

Proof. By Kuroda's Lemma ([11]), monotone languages are effectively closed under boolean operations. Then 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}. Thus $\mathcal{M}_d$ is an s-TA. Next we show the determinisity 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 $\langle X \rangle \to A_1 \Leftarrow \mathsf{leaf}(X)$ in $\mathcal{L}(\mathcal{G}_{A1})$ and $\langle X \rangle \to A_2 \Leftarrow \mathsf{leaf}(X)$ in $\mathcal{L}(\mathcal{G}_{A2})$ in Δ_{d2}. By Lemma 4, we obtain $\mathcal{L}(\mathcal{G}_{A1}) \cap \mathcal{L}(\mathcal{G}_{A2}) = \varnothing$. 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 $\{w \in \mathcal{Q}_d^* \mid |w| \geqslant 2\} - \bigcup_{q \in \mathcal{Q}} \mathcal{L}(\mathcal{G}_q^d)$ 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}. □

Lemma 6. *$\mathcal{M}_d$ is complete.*

Proof. By definition, for every $f \in \mathcal{F}$ and $A_1, \ldots, A_n \in \mathcal{Q}_d$, there exists a transition rule of the form $f(A_1, \ldots, A_n) \to A$ in Δ_{d1}. For another pattern $\langle A_1, \ldots, A_m \rangle$ with $A_1, \ldots, A_m \in \mathcal{Q}_d$ and $m \geqslant 2$, we take $L = \bigcup_{A \in (\mathcal{Q}_d - \{\varnothing\})} \mathcal{L}(\mathcal{G}_A)$. Then we can show that $\bigcup_{q \in \mathcal{Q}} \mathcal{L}(\mathcal{G}_q^d)$ is the same as L. This implies $L \cup \mathcal{L}(\mathcal{G}_0) = \{w \in \mathcal{Q}_d^* \mid |w| \geqslant 2\}$. Hence, if $A_1 \cdots A_m \in L$, there exists a transition rule $\langle X \rangle \to A \Leftarrow \mathsf{leaf} X$ in $\mathcal{L}(\mathcal{G}_A)$ in Δ_{d2} that is applicable to $\langle A_1, \ldots, A_m \rangle$; otherwise, Δ_{d3} is applicable. □

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 $\langle X \rangle \to q \Leftarrow \mathsf{leaf}(X)$ in $\mathcal{L}(\mathcal{G}_1)$ and $\langle X \rangle \to q \Leftarrow \mathsf{leaf}(X)$ in $\mathcal{L}(\mathcal{G}_2)$ with $\mathcal{G}_1 \neq \mathcal{G}_2$, they can be merged to a single rule $\langle X \rangle \to q \Leftarrow \mathsf{leaf}(X)$ 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 $\langle X \rangle \to q \Leftarrow \mathsf{leaf}(X)$ in $\mathcal{L}(\mathcal{G}_q)$ in Δ, we define a monotone (but not necessarily deterministic) grammar $\mathcal{G}_q^d$ as follows: Let $\mathcal{G}_q = (\Sigma, \mathcal{S}, \mathsf{s}_0, \Lambda)$, then $\mathcal{G}_q^d = (2^\Sigma, \mathcal{S}, \mathsf{s}_0, \Lambda')$ where

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

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

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

Fig. 2. Determinization of sequence-tree automata

Lemma 7. *Suppose* $\langle X \rangle \to A \Leftarrow \mathsf{leaf}(X)$ *in* $\mathcal{L}(\mathcal{G}_A)$ *is a transition rule in* Δ_{d2}. *Let* $L_A = \{\, \boldsymbol{A_k} \in \mathcal{Q}_d{}^* \mid q \in A$ *if and only if* $\exists q_i \in A_i (1 \leqslant i \leqslant k)$ *and* $\boldsymbol{q_k} \in \mathcal{L}(\mathcal{G}_q) \,\}$. *Here* $\boldsymbol{A_k}$ *and* $\boldsymbol{q_k}$ *denote words* $A_1 \cdots A_k$ *and* $q_1 \cdots q_k$, *respectively. Then* $\mathcal{L}(\mathcal{G}_A) = L_A$.

Proof. First we observe that $\mathcal{L}(\mathcal{G}_A)$ can be represented as $\mathcal{L}(\mathcal{G}_A) = B_A - C_A$ where $B_A = \bigcap_{q \in A} \mathcal{L}(\mathcal{G}_q^d)$ and $C_A = \bigcup_{q \in \mathcal{Q} - A} \mathcal{L}(\mathcal{G}_q^d)$. Then B_A consists of words $\boldsymbol{A_k}$ such that ($k \geqslant 2$ and) for any $q \in A$ there exists $\boldsymbol{q_k} \in \mathcal{L}(\mathcal{G}_q^d)$ such that $q_i \in A_i$ $(1 \leqslant i \leqslant k)$, i.e.

$$B_A = \{\, \boldsymbol{A_k} \mid \forall q \in A, \qquad \exists \boldsymbol{q_k} \in \mathcal{L}(\mathcal{G}_q) \text{ such that } q_i \in A_i\ (1 \leqslant i \leqslant k) \,\}.$$

Likewise, we have

$$C_A = \{\, \boldsymbol{A_k} \mid \exists q \in \mathcal{Q} - A,\ \exists \boldsymbol{q_k} \in \mathcal{L}(\mathcal{G}_q) \text{ such that } q_i \in A_i\ (1 \leqslant i \leqslant k) \,\}.$$

We show $\mathcal{L}(\mathcal{G}_A) \supseteq L_A$. Suppose $\boldsymbol{A_k}$ is an element in L_A. By definition, $\boldsymbol{A_k} \in B_A$. Moreover, for any $q \in \mathcal{Q} - A$, there is no word $\boldsymbol{q_k} \in \mathcal{L}(\mathcal{G}_q^d)$ such that $q_i \in A_i$ $(1 \leqslant i \leqslant k)$. Then $\boldsymbol{A_k} \notin C_A$. Similarly, we suppose for the reverse that $\boldsymbol{B_k}$ is an

element in $B_A - C_A$. This implies that $q \in A$ if and only if $\exists q_i \in A_i$ $(1 \leqslant i \leqslant k)$ and $\boldsymbol{q_k} \in \mathcal{L}(\mathcal{G}_q)$. Hence, $\boldsymbol{B_k} \in L_A$. □

Lemma 8. $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}_d)$.

Proof. We show that for every sequence-term $t \in \text{s-}\mathcal{T}(\mathcal{F})$ and state symbol $A \in \mathcal{Q}_d$, if $t \to^*_{\mathcal{M}_d} A$ then A is the same as the set $\{\, q \in \mathcal{Q} \mid t \to^*_{\mathcal{M}} q \,\}$. We use the structural induction on t. If t is a constant, $\mathcal{M}_d$ has a transition rule $t \to \{\, q \in \mathcal{Q} \mid t \to q \in \Delta \,\}$, and $\mathcal{M}_d$ has no other rule for t. We suppose $t = f(t_1, \ldots, t_n)$ with $f \in \mathcal{F}$. Then, $t_i \to^*_{\mathcal{M}_d} A_i$ $(1 \leqslant i \leqslant n)$ and $f(A_1, \ldots, A_n) \to A \in \Delta_d$. By Lemma 5, $\mathcal{M}_d$ does not allow the move relation for t except $t \to^*_{\mathcal{M}_d} f(A_1, \ldots, A_n) \to_{\mathcal{M}_d} A$. By definition, $A = \{\, q \in \mathcal{Q} \mid \exists f(q_1, \ldots, q_n) \to q \in \Delta$ such that $q_i \in A_i$ $(1 \leqslant i \leqslant n) \,\}$. On the other hand, by induction hypothesis, we obtain $A_i = \{\, q_i \in \mathcal{Q} \mid t_i \to^*_{\mathcal{M}} q_i \,\}$. Then, $A = \{\, q \in \mathcal{Q} \mid \exists f(q_1, \ldots, q_n) \to q \in \Delta$ such that $t_i \to^*_{\mathcal{M}} q_i$ $(1 \leqslant i \leqslant n) \,\} = \{\, q \in \mathcal{Q} \mid f(t_1, \ldots, t_n) \to^*_{\mathcal{M}} q \,\}$. Next we suppose $t = \langle t_1, \ldots, t_m \rangle$ such that $m \geqslant 2$ and $t_i \to^*_{\mathcal{M}_d} A_i$ $(1 \leqslant i \leqslant m)$. By Lemma 5, $\mathcal{M}_d$ does not allow the move relation for t except $t \to^*_{\mathcal{M}_d} \langle A_1, \ldots, A_m \rangle \to_{\mathcal{M}_d} A$. By construction, the last move relation $\langle A_1, \ldots, A_m \rangle \to_{\mathcal{M}_d} A$ is made by the rule $\langle X \rangle \to A \Leftarrow \mathsf{leaf}(X)$ in $\mathcal{L}(\mathcal{G}_A)$ in Δ_d. By Lemma 7, $q \in A$ if and only if $\exists q_i \in A_i$ $(1 \leqslant i \leqslant m)$ and $\boldsymbol{q_m} \in \mathcal{L}(\mathcal{G}_q)$. Hence, by induction hypothesis, A is the same as the set $\{\, q \in \mathcal{Q} \mid \exists q_i \in \mathcal{Q}$ such that $t_i \to^*_{\mathcal{M}} q_i$ $(1 \leqslant i \leqslant m)$ and $\langle q_1, \ldots, q_m \rangle \to_{\mathcal{M}} q \,\}$.

The rest of the proof is easy. By Lemmata 5 and 6 for every $t \in \text{s-}\mathcal{T}(\mathcal{F})$ there exists a (and only one) state $A \in \mathcal{Q}_d$ such that $t \to^*_{\mathcal{M}_d} A$. From the above property, $A \in \mathcal{Q}_{dfin}$ if and only if $t \to^*_{\mathcal{M}} q \in \mathcal{Q}_{fin}$ for some $q \in A$. □

Theorem 3. *Given an s-TA $\mathcal{M}$, we can compute a complete and deterministic s-TA $\mathcal{M}_d$ such that $\mathcal{L}(\mathcal{M}_d) = \mathcal{L}(\mathcal{M})$.* □

Corollary 2. *Monotone sequence-tree languages are effectively closed under complementation.* □

Therefore, the desired closure property is obtained.

Corollary 3. *Recognizable A-tree languages are effectively closed under complementation.*

Proof. For every A-closed tree language L over the signature $\mathcal{F}$ with $\mathcal{F}_{\mathrm{A}} = \{\mathsf{f}\}$, $\mathcal{T}(\mathcal{F}) - L$ is A-closed, and s-$\mathcal{T}(\mathcal{F}) - \mathsf{flat}(L) = \mathsf{flat}(\mathcal{T}(\mathcal{F}) - L)$. Then, by Lemma 2 and the above corollary, there effectively exists an s-TA that recognizes s-$\mathcal{T}(\mathcal{F}) - \mathsf{flat}(L)$. By Lemma 3, we can compute an A-TA $\mathcal{A}/\mathsf{A}$ such that $t \in \mathsf{flat}(\mathcal{T}(\mathcal{F}) - L)$ if and only if $\mathsf{unflat}(t) \in \mathcal{L}(\mathcal{A}/\mathsf{A})$. Thus, by showing that $\mathcal{A}/\mathsf{A}$ recognizes the A-closure of $\mathsf{unflat}(\mathsf{flat}(\mathcal{T}(\mathcal{F}) - L))$, we know the complement of L is recognizable with $\mathcal{A}/\mathsf{A}$. □

4 Recognizing One-step A-Rewrite Descendents

This section shows that, given an A-TA $\mathcal{A}/\mathsf{A}$ and a ground A-TRS $\mathcal{R}/\mathsf{A}$, the tree language $(\to_{\mathcal{R}/\mathsf{A}})[\mathcal{L}(\mathcal{A}/\mathsf{A})]$ is *effectively* recognizable with A-TA. Because $\mathcal{R}$ is ground, every rule $l \to r$ in $\mathcal{R}$ can be simulated by decomposed rules of the following shape: $f(c_1, \ldots, c_m) \to c_0$ or $d_0 \to g(d_1, \ldots, d_n)$, where c_i $(0 \le i \le m)$ and d_j $(0 \le j \le n)$ are fresh constants. The nontrivial case to be considered in the proofs occurs in the rewrite relation made by a rule $\mathsf{f}(c_1, c_2) \to c_0$ with f an associativity symbol. First we consider the string rewriting case below, and then, we generalize the result to A-tree rewriting case.

Lemma 9. *Given a monotone grammar $\mathcal{G} = (\Sigma, \mathcal{S}, \mathsf{s}_0, \Lambda)$ and a string rewrite system $\mathcal{R} = \{\mathsf{a}\,\mathsf{b} \to \mathsf{c}\}$ with $\mathsf{a}, \mathsf{b}, \mathsf{c} \in \Sigma$. Then we can compute a monotone grammar that recognizes $(\to_{\mathcal{R}})[\,\mathcal{L}(\mathcal{G})\,]$.*

Proof. We assume without loss of generality that Λ does not contain a transition rule whose left-hand side is s_0 and s_0 appears in the right-hand side. We define a monotone grammar $\mathcal{G}' = (\Sigma, \mathcal{S}', \mathsf{s}_0, \Lambda')$ as follows: $\mathcal{S}' = \mathcal{S} \cup \{\,[p\,q] \mid p, q \in \mathcal{S}\,\}$ and

$$
\begin{array}{lrcl|l}
\Lambda' = & \{ & \mathsf{s}_0 \to \mathsf{c} & \mid \mathsf{s}_0 \to^*_{\mathcal{G}} \mathsf{a}\,\mathsf{b} \} & \cdots (1)\\
\cup & \{ & \mathsf{s}_0 \to [p\,q]\,r & \mid p, q, r \in \mathcal{S} \text{ such that } \mathsf{s}_0 \to^*_{\mathcal{G}} p\,q\,r \} & \cdots (2)\\
\cup & \{ & [p\,q] \to \mathsf{c} & \mid p \to \mathsf{a}, q \to \mathsf{b} \in \Lambda \} & \cdots (3)\\
\cup & \{ & p \to a & \mid p \to a \in \Lambda, p \neq s_0, a \in \Sigma \} & \cdots (4)\\
\cup & \left\{ \begin{array}{rcl} [p\,q] & \to & [p_1\,p_2]\,q \\ [q\,p] & \to & [q\,p_1]\,p_2 \\ p & \to & p_1\,p_2 \end{array} \right. & \mid p \to p_1\,p_2 \in \Lambda, p \neq s_0, q \in \mathcal{S} \left. \right\} & \cdots (5)\\
\cup & \left\{ \begin{array}{rcl} [p_1\,p_2]\,q & \to & [q_1\,q_2]\,q \\ p_1\,[p_2\,q] & \to & q_1\,[q_2\,q] \\ q\,[p_1\,p_2] & \to & q\,[q_1\,q_2] \\ [q\,p_1]\,p_2 & \to & [q\,q_1]\,q_2 \\ p_1\,p_2 & \to & q_1\,q_2 \end{array} \right. & \mid p_1\,p_2 \to q_1\,q_2 \in \Lambda, q \in \mathcal{S} \left. \right\} & \cdots (6)\\
\cup & \left\{ \begin{array}{rcl} [p\,q]\,r & \to & p\,[q\,r] \\ p\,[q\,r] & \to & [p\,q]\,r \end{array} \right. & \mid p, q, r \in \mathcal{S} \left. \right\} & \cdots (7)
\end{array}
$$

Rules in (1) and (2) are computable, because the membership problem is decidable for monotone grammar. In addition, (2) is finite, because possible combinations are finite. The second rules in (5) can be replaced by $[q\,p] \to q\,[p_1 p_2]$. For the proof convenience, the second and fourth shapes of rules For the proof convenience, the second and fourth shapes of rules in (6) are included. In fact, they can be omitted, because their move relation is simulated by using other rules in (6) and (7).

First we prove that for all $p, p', q, q' \in \mathcal{S}$ and $\alpha, \alpha', \beta, \beta' \in \mathcal{S}^*$, $\alpha\,p\,q\,\beta \to^{=}_{\mathcal{G}} \alpha' p' q' \beta'$ if and only if $\alpha\,[p\,q]\,\beta \to^{=}_{\mathcal{G}'} \alpha'[p'q']\beta'$.

$\Rightarrow$ Suppose $\alpha\,p\,q\,\beta \to_{\mathcal{G}} \alpha' p' q' \beta'$. An applied rule in this step has the shape of either $p_1\,p_2 \to q_1\,q_2$ or $p_1 \to q_1\,q_2$ with $p_1, p_2, q_1, q_2 \in \mathcal{S}$. It implies that there

exists a rule in (5) or (6) applicable to $\alpha\,[p\,q]\,\beta$. By case analysis of the shape and position of such a rule, we obtain $\alpha\,[p\,q]\,\beta \to_{\mathcal{G}'} \alpha'[p'q']\beta'$.

$\Leftarrow$ Suppose $\alpha\,[p\,q]\,\beta \to_{\mathcal{G}'} \alpha'[p'q']\beta'$. By construction of $\mathcal{G}'$, the applied rule is in (5), (6) or (7). By case analysis, we obtain $\alpha\,p\,q\,\beta \to_{\mathcal{G}} \alpha'p'q'\beta'$ if $\alpha\,[p\,q]\,\beta \to_{\mathcal{G}'} \alpha'[p'q']\beta'$ is done by a rule in (5) or (6); otherwise, $\alpha\,p\,q\,\beta = \alpha'p'q'\beta'$.

We denote below $\to_{\Lambda_1}$ to be the move relation made by rules in $\Lambda_1 = \{\, p \to a \mid p \in \mathcal{S},\, a \in \Sigma \text{ such that } p \to a \in \Lambda \,\}$, and $\to_{\Lambda_2}$ to be the move relation by $\Lambda - \Lambda_1$. Then we can prove that $\to_{\Lambda_1} \cdot \to_{\Lambda_2} \subseteq \to_{\Lambda_2} \cdot \to_{\Lambda_1}$. This implies that if $\mathsf{s}_0 \to^*_{\mathcal{G}} u\,\mathsf{a}\,\mathsf{b}\,v \in \Sigma^+$, there exists some $\gamma \in \mathcal{S}^+$ such that $\mathsf{s}_0 \to^*_{\Lambda_2} \gamma \to^*_{\Lambda_1} u\,\mathsf{a}\,\mathsf{b}\,v$. One should notice that $|\gamma| = |u\,\mathsf{a}\,\mathsf{b}\,v|$, because of the shape of transition rules in Λ_1. Moreover, if $u\,\mathsf{a}\,\mathsf{b}\,v \in \mathcal{L}(\mathcal{G})$, we can assume

$$\mathsf{s}_0 \to^*_{\Lambda_2} \alpha\,p\,q\,\beta \to^*_{\Lambda_1} u\,\mathsf{a}\,\mathsf{b}\,v \in \Sigma^+$$

such that $|\alpha| = |u|$ and $|\beta| = |v|$. Since $u\,\mathsf{a}\,\mathsf{b}\,v$ is obtained by application of rules in Λ_1, there must be transition rules $p \to \mathsf{a}$, $q \to \mathsf{b}$ in Λ in this case. If $|u| + |v| = 0$, we have $\mathsf{c} \in \mathcal{L}(\mathcal{G}')$ by (1). If $|u| + |v| > 0$, the first Λ_2-steps can be represented as $\mathsf{s}_0 \to^*_{\Lambda_2} p_1\,p_2\,p_3 \to^*_{\Lambda_2} \alpha\,p\,q\,\beta$ for some $p_1, p_2, p_3 \in \mathcal{S}$. By (2), we have $\mathsf{s}_0 \to_{\mathcal{G}'} [p_1\,p_2]\,p_3$ if and only if $\mathsf{s}_0 \to^*_{\Lambda_2} p_1\,p_2\,p_3$. From bijective correspondence between $\to_{\overline{\mathcal{G}}}$ and $\to_{\overline{\mathcal{G}'}}$, this implies $[p_1\,p_2]\,p_3 \to^*_{\mathcal{G}'} \alpha\,[p\,q]\,\beta$ if and only if $p_1\,p_2\,p_3 \to^*_{\mathcal{G}} \alpha\,p\,q\,\beta$. Hence, by rules in (3) and (4), we obtain $u\,\mathsf{c}\,v$.

For the reverse, we observe that $\mathsf{s}_0 \to^*_{\mathcal{G}'} u\,\mathsf{c}\,v \in \Sigma^+$ if and only if

$$\mathsf{s}_0 \to_{\Lambda'_1} \mathsf{c} \quad \text{or} \quad \mathsf{s}_0 \to^*_{\Lambda'_2} \alpha\,[p\,q]\,\beta \to^*_{\Lambda'_1} u\,\mathsf{c}\,v.$$

with $p, q \in \mathcal{S}$ and $\alpha, \beta \in \mathcal{S}^*$. Here $\Lambda'_1 = \{p \to a \in \Lambda' \mid a \in \Sigma\}$ and $\Lambda'_2 = \Lambda' - \Lambda'_1$. For the latter case, we can prove that $\mathsf{s}_0 \to^*_{\Lambda_2} \alpha\,p\,q\,\beta \to^*_{\Lambda_1} u\,\mathsf{a}\,\mathsf{b}\,v$ as in the previous way. □

In Lemma 9, the letters $\mathsf{a}, \mathsf{b}, \mathsf{c}$ are not necessarily different from each other. Moreover, we can take a string rewrite rule where the length of the left-hand side is more than 2. In fact, if $\mathcal{R} = \{w \to \mathsf{c}\}$ with $w \in \Sigma^+$ and $|w| = n$ ($\geqslant 2$), we define Λ' as follows: (1)–(3) are replaced by

$$\begin{array}{lll}
\{ & \mathsf{s}_0 \to \mathsf{c} & \mid \mathsf{s}_0 \to^*_{\mathcal{G}} w \} \quad \cdots (1') \\
\{ & \mathsf{s}_0 \to [p_1 \cdots p_n]\,q & \mid \mathsf{s}_0 \to^*_{\mathcal{G}} p_1 \cdots p_n\,q \in \mathcal{S}^+ \} \quad \cdots (2') \\
\{ & [p_1 \cdots p_n] \to \mathsf{c} & \mid p_i \to a_i \in \Lambda,\ w = a_1 \cdots a_n \} \quad \cdots (3')
\end{array}$$

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

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

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

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

Likewise, rules in (6) and (7) are modified. Therefore the general version of $\mathcal{G}'$ recognizes $(\to_{\mathcal{R}})[\,\mathcal{L}(\mathcal{G})\,]$ with $\mathcal{R} = \{\, w \to \mathsf{c}\,\}$.

Lemma 10. *Given an A-TA $\mathcal{A}/\mathsf{A}$ and a ground TRS $\mathcal{R} = \{\mathsf{f}(\mathsf{a},\mathsf{b}) \to \mathsf{c}\}$ over the same signature $\mathcal{F}$ with $\mathcal{F}_{\mathrm{A}} = \{\mathsf{f}\}$. Then we can compute an A-TA that recognizes $(\to_{\mathcal{R}/\mathsf{A}})[\,\mathcal{L}(\mathcal{A}/\mathsf{A})\,]$.*

Proof. Suppose $\mathcal{M}_{\mathcal{A}} = (\mathcal{F} - \{\mathsf{f}\}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$ is an s-TA associated with $\mathcal{A}/\mathsf{A}$. Without loss of generality, we assume that for all $a \to p, b \to q \in \Delta$ 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 $W = \{\, p\,q \mid \mathsf{a} \to p, \mathsf{b} \to q \in \Delta \,\}$. We write $\to_{\mathsf{c}}$ for the binary relation induced by the string rewrite system (SRS for short) $\{w \to \diamond \mid w \in W\}$ over the language $(\mathcal{Q} \cup \{\diamond\})^+$. Similarly, $\to_{\diamond}$ for the induced binary relation of the SRS $\{p \to p_{\diamond} \mid p \in \mathcal{Q}\}$. Define $\mathcal{M}'_{\mathcal{A}} = (\mathcal{F} - \{\mathsf{f}\}, \mathcal{Q}', \mathcal{Q}'_{fin}, \Delta')$ 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

$$
\begin{aligned}
\Delta' = \Delta \cup \{\, & \mathsf{c} \to \diamond \,\} \\
\cup \{\, & \mathsf{c} \to q_{\diamond} \mid \langle X \rangle \to q \Leftarrow \mathsf{leaf}(X) \text{ in } L \in \Delta,\ L \cap W \neq \varnothing \,\} \\
\cup \{\, & f(p_1, \ldots, p_{i\diamond}, \ldots, p_n) \to q_{\diamond} \mid f(p_1, \ldots, p_i, \ldots, p_n) \to q \in \Delta \,\} \\
\cup \{\, & \langle X \rangle \to q_{\diamond} \Leftarrow \mathsf{leaf}(X) \text{ in } (\to_{\diamond})[L\,] \mid \langle X \rangle \to q \Leftarrow \mathsf{leaf}(X) \text{ in } L \in \Delta \,\} \\
\cup \Big\{ & \langle X \rangle \to q_{\diamond} \Leftarrow \mathsf{leaf}(X) \text{ in } (\to_{\mathsf{c}})[L'\,] \;\Big|\; \begin{array}{l} \langle X \rangle \to q \Leftarrow \mathsf{leaf}(X) \text{ in } L \in \Delta \\ L' = L - W \end{array} \Big\}
\end{aligned}
$$

By Lemma 9, a monotone grammar recognizing $(\to_{\mathsf{c}})[\,L'\,]$ is computable. Furthermore, for every $\langle X \rangle \to q \Leftarrow \mathsf{leaf}(X)$ in L in Δ, a monotone grammar recognizing $(\to_{\diamond})[\,L\,]$ is computable. Then it can be proved by the structural induction that the s-TA $\mathcal{M}'_{\mathcal{A}}$ recognizes the sequence-term representation of $(\to_{\mathcal{R}/\mathsf{A}})[\,\mathcal{L}(\mathcal{A}/\mathsf{A})\,]$. □

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

Proof. For the first part, let $l \to r$ be a rewrite rule in $\mathcal{R}$. We define the ground TRS

$$
\begin{aligned}
\mathcal{R}_{l \to r} = \{\, & f(\mathsf{c}_{t_1}^{p \cdot 1}, \ldots, \mathsf{c}_{t_n}^{p \cdot n}) \to \mathsf{c}_{f(t_1, \ldots, t_n)}^{p} \mid p \in \mathsf{pos}(l),\ \ l_{|p} = f(t_1, \ldots, t_n) \,\} \\
& \cup \{\, \mathsf{d}_{f(t_1, \ldots, t_n)}^{p} \to f(\mathsf{d}_{t_1}^{p \cdot 1}, \ldots, \mathsf{d}_{t_n}^{p \cdot n}) \mid p \in \mathsf{pos}(r),\ r_{|p} = f(t_1, \ldots, t_n) \,\}
\end{aligned}
$$

Then, 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 $s \to^+_{\mathcal{R}_{l \to r}} C[\mathsf{c}_l^{\epsilon}]$ and $C[\mathsf{d}_r^{\epsilon}] \to^+_{\mathcal{R}_{l \to r}} t$. Moreover, this statement is generalized as follows: $s \sim_{\mathsf{A}} C[l]$ and $t \sim_{\mathsf{A}} C[r]$ if and only if $s \to^+_{\mathcal{R}_{l \to r}/\mathsf{A}} C[\mathsf{c}_l^{\epsilon}]$ and $C[\mathsf{d}_r^{\epsilon}] \to^+_{\mathcal{R}_{l \to r}/\mathsf{A}} t$.

Using this fact, we define the procedure in Fig. 3. If the root symbol f of a decomposed rule $f(c_1, \ldots, c_n) \to c$ is an A-symbol, we know that the one-step A-rewrite descendents are effectively recognizable by Lemma 10; otherwise, the one-step A-rewrite descendents $(\to_{\mathcal{R}_{l \to r}/\mathsf{A}})[L]$ are the same as $\mathsf{A}((\to_{\mathcal{R}_{l \to r}})[L])$, which has been noted by Dauchet and Tison [5]. Then the procedure computes an A-TA $\mathcal{B}_{l \to r}/\mathsf{A}$ that recognizes the tree language

$$
L_{l \to r} = \{\, t \in \mathcal{T}(\mathcal{F}) \mid s \in \mathcal{L}(\mathcal{A}/\mathsf{A}),\ s \to^+_{\mathcal{R}_{l \to r}/\mathsf{A}} C[\mathsf{c}_l^{\epsilon}] \text{ and } C[\mathsf{d}_r^{\epsilon}] \to^+_{\mathcal{R}_{l \to r}/\mathsf{A}} t \,\}.
$$

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

$\mathcal{A}_0 := \mathcal{A};\ \ i := 0;\ \ j := 0;$
$S\ := \mathsf{pos}(l);$
$T\ := \mathsf{pos}(r);$
while $S \neq \varnothing$ **do**
  select $p \in S$ such that $\nexists p' \in S.\ p' \succ p$
  let $l_{|p} = f(t_1, \ldots, t_n)$ and
  compute $\mathcal{A}_{i+1}/\mathsf{A}$ such that
    $\mathcal{L}(\mathcal{A}_{i+1}/\mathsf{A}) = (\rightarrow_{\{f(\mathsf{c}^{p\cdot 1}_{t_1}, \ldots, \mathsf{c}^{p\cdot n}_{t_n}) \rightarrow \mathsf{c}^{p}_{l_{|p}}\}/\mathsf{A}})[\,\mathcal{L}(\mathcal{A}_i/\mathsf{A})\,]$
  $i\ := i + 1;$
  $S := S - \{p\};$
od
compute $\mathcal{B}_0/\mathsf{A}$ such that
  $\mathcal{L}(\mathcal{B}_0/\mathsf{A}) = (\rightarrow_{\{\mathsf{c}^{\epsilon}_{l} \rightarrow \mathsf{d}^{\epsilon}_{r}\}/\mathsf{A}})[\,\mathcal{L}(\mathcal{A}_i/\mathsf{A})\,]$
while $T \neq \varnothing$ **do**
  select $q \in T$ such that $\nexists q' \in T.\ q \succ q'$
  let $r_{|q} = f(t_1, \ldots, t_n)$ and
  compute $\mathcal{B}_{j+1}/\mathsf{A}$ such that
    $\mathcal{L}(\mathcal{B}_{j+1}/\mathsf{A}) = (\rightarrow_{\{\mathsf{d}^{q}_{r_{|q}} \rightarrow f(\mathsf{d}^{q\cdot 1}_{t_1}, \ldots, \mathsf{d}^{q\cdot n}_{t_n})\}/\mathsf{A}})[\,\mathcal{L}(\mathcal{B}_j/\mathsf{A})\,]$
  $j\ := j + 1;$
  $T := T - \{q\};$
od
$\mathcal{B}_{l\rightarrow r}\ := \mathcal{B}_j;$
return $\mathcal{B}_{l\rightarrow r}/\mathsf{A}$

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

The second part can be shown in a similar way. First we prove that context-free languages are closed under one-step string rewriting of $\{\,\mathsf{a}\,\mathsf{b} \rightarrow \mathsf{c}\,\}$, that corresponds to Lemma 9. 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 Λ' of $\mathcal{G}'$ consists of (1), (3), (4) and the following rules:

$$\left\{ \begin{array}{rcl|l} s_0 & \rightarrow & [p\,q]\,r & \\ s_0 & \rightarrow & p\,[q\,r] & p, q, r \in \mathcal{S} \text{ such that } \mathsf{s}_0 \rightarrow^*_{\mathcal{G}} p\,q\,r \end{array} \right\} \quad \cdots\ (2')$$

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

Transition rules (6) and (7) in Lemma 9 are unnecessary. □

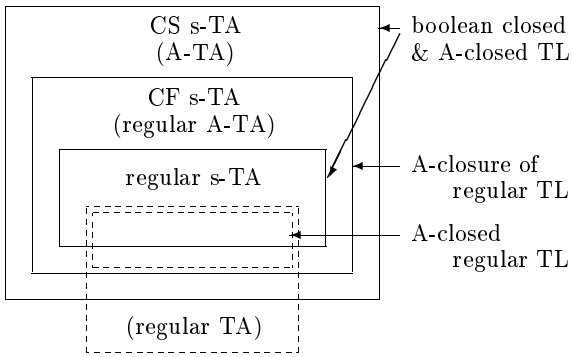

Fig. 4. Relationships in sequence-tree languages

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 (Corollary 2), 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 A-rewrite descendents (Theorem 4). This allows us to provide an *under-approximation* algorithm for computing A-rewrite descendents 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 $\mathcal{F}_{\mathrm{A}}$ case. But, for many arbitrary associative symbols, the sequence-term model has to be allowed to contain more special symbols, e.g. $\langle\ \rangle_1, \ldots, \langle\ \rangle_n$. Actually, this simple idea works, so that all the above results can be generalized to an arbitrary $\mathcal{F}_{\mathrm{A}}$. More precisely, by modifying the definition of sequence-terms and indexing the special symbol $\langle\ \rangle$ as above, Theorems 1–4 and Corollaries 1–3 can be extended to the general case.

By restricting the grammatical condition of conditional 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 [13]. 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:

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

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

$$\langle X\rangle \to \mathsf{q}_t \Leftarrow \mathsf{leaf}(X) \ \mathsf{in}\ \{\mathsf{q}_{t_1}\cdots\mathsf{q}_{t_n}\} \quad \text{if}\ \ t = C_{\mathsf{f}}[\![t_1,\ldots,t_n]\!]\ \text{ is an f-block of } l$$

and

$$g(\mathsf{q}_{t_1},\ldots,\mathsf{q}_{t_m}) \to \mathsf{q}_t \qquad\qquad \text{if}\ \ t = g(t_1,\ldots,t_m) \trianglelefteq l\ \text{ with }\ q \neq \mathsf{f}$$

together with $\langle X\rangle \to \mathsf{q}_l \Leftarrow \mathsf{leaf}(X)\ \mathsf{in}\ \{w \mid w = \mathsf{q}^*\mathsf{q}_l\,\mathsf{q}^* \text{ and } |w| \geqslant 2\}$, $\langle X\rangle \to \mathsf{q} \Leftarrow \mathsf{leaf}(X)\ \mathsf{in}\ \{w \mid w = \mathsf{q}^* \text{ and } |w| \geqslant 2\}$, $g(\mathsf{q},\ldots,\mathsf{q}_l,\ldots,\mathsf{q}) \to \mathsf{q}_l$ and $g(\mathsf{q},\ldots,\mathsf{q}) \to \mathsf{q}$ for all $g \in \mathcal{F} - \{\mathsf{f}\}$. Observe that the membership test in the conditional part of each transition rule can be represented by a regular grammar. Let $\mathcal{M}_{\mathcal{R}}$ be an s-TA that recognizes $\bigcup_{l\to r\in\mathcal{R}} \mathcal{L}(\mathcal{M}_{l\to r})$. The (sequence-term representation of) set of normal forms of $\mathcal{R}/\mathsf{A}$ 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: $\mathsf{leaf}(X)\ \mathsf{in}\ \bigcap_{i\in I}\mathcal{L}(\mathcal{G}_i) - \bigcup_{i\in(K-I)}\mathcal{L}(\mathcal{G}_i)$ with $\mathcal{G}_i$ a regular grammar for every $i \in (K \cup I)$. Therefore, the set of normal forms of $\mathcal{R}/\mathsf{A}$ is effectively recognizable with regular A-TA. □

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.

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.C. de Vrijer, and J.W. Klop (eds.): *Term Rewriting Systems*, Cambridge Tracts in Theoretical Computer Science 25, Cambridge University Press, 2003. To appear.

4. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi: *Tree Automata Techniques and Applications*, draft, 2002. Available on `http://www.grappa.univ-lille3.fr/tata/`
5. M. Dauchet and S. Tison: *The Theory of Ground Rewrite Systems is Decidable*, Proc. 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. S.Y. Kuroda: *Classes of Languages and Linear Bounded Automata*, Information and Control 7, pp. 207–223, 1964.
12. 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/`
13. 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/LIF/Rapports/`
14. A. Mateescu and A. Salomaa: *Aspects of Classical Language Theory*, Handbook of Formal Languages 1, pp. 175–251 (Chapter 4), Springer-Verlag, 1996.
15. H. Ohsaki: *Beyond Regularity: Equational Tree Automata for Associative and Commutative Theories*, Proc. of 15th CSL, Paris (France), LNCS 2142, pp. 539–553, 2001.
16. 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.
17. Y. Toyama: *Membership Conditional Term Rewriting Systems*, Trans. of IEICE E72(11), pp. 1224–1229, 1989. Information of the journal is found at `http://www.ieice.org/`