

Forward and Backward Application of Symbolic Tree Transducers

Zoltán Fülöp^{a*} and Heiko Vogler^b

^a Department of Foundations of Computer Science, University of Szeged
Árpád tér 2., H-6720 Szeged, Hungary. fulop@inf.u-szeged.hu

^b Faculty of Computer Science, Technische Universität Dresden
Mommsenstr. 13, D-01062 Dresden, Germany. Heiko.Vogler@tu-dresden.de

November 21, 2018

Abstract: We consider symbolic tree automata (sta) and symbolic tree transducers (stt). We characterize s-recognizable tree languages (which are the tree languages recognizable by sta) in terms of (classical) recognizable tree languages and relabelings. We prove that sta and the recently introduced variable tree automata are incomparable with respect to their recognition power. We define symbolic regular tree grammars and characterize s-regular tree languages in terms of regular tree languages and relabelings. As a consequence, we obtain that s-recognizable tree languages are the same as s-regular tree languages.

We show that the syntactic composition of two stt computes the composition of the tree transformations computed by each stt, provided that (1) the first one is deterministic or the second one is linear and (2) the first one is total or the second is nondeleting. We consider forward application and backward application of stt and prove that the backward application of an stt to any s-recognizable tree language yields an s-recognizable tree language. We give a linear stt of which the range is not an s-recognizable tree language. We show that the forward application of simple and linear stt preserves s-recognizability. As a corollary, we obtain that the type checking problem of simple and linear stt and the inverse type checking problem of arbitrary stt is decidable.

ACM classification: F.1.1, F.4.2, F.4.3

*Research of this author was supported by the program TÁMOP-4.2.1/B-09/1/KONV-2010-0005 of the Hungarian National Development Agency.

Key words and phrases: Tree automata, tree transducers, composition of tree transducers

1 Introduction

Symbolic tree automata (sta) and symbolic tree transducers (stt) were introduced in [VB11a] and [VB11b]. They differ from classical finite-state tree automata and tree transducers [GS84, GS97] in that they work with trees over an infinite, unranked set of symbols. According to [GKS10], examples of systems with finite control and infinite source of data are software with integer parameters [BHM03], datalog systems with infinite data domain [BHJS07], and XML documents of which the leaves are associated with data values from some infinite domain [BCC⁺03]. It was mentioned in [VB11a] that lifting the finite alphabet restriction is useful to enable efficient symbolic analysis. Symbolic transducers are useful for exploring symbolic solvers when performing basic automata-theoretic transformations [VHL⁺12].

In this paper we provide new formal definitions of sta and stt which slightly differs from those given in [VB11a, VB11b]. At the end of Sections 3.1 and 5.1 we will compare our definitions with the original ones.

Roughly speaking, an sta is a finite-state tree automaton [Don70] except that the input trees are built up over an infinite set of labels. In order to ensure a finite description of the potentially infinite set of transitions we bind the maximal number of the successors of a any node occurring in an input tree by an integer $k \in \mathbb{N}$, and we employ finitely many unary Boolean-valued predicates over the set of labels. Then every transition of a symbolic k -bounded tree automaton (sk -ta) has the form

$$(q_1 \dots q_l, \varphi, q)$$

where $0 \leq l \leq k$, q, q_1, \dots, q_l are states, and φ is a unary Boolean-valued predicate. Such a transition is applicable to a node if φ holds for the label of that node. The tree language $L(\mathcal{A})$ recognized by an sta \mathcal{A} is defined as the union of all tree languages $L(\mathcal{A}, q)$, where q is a final state, and the family $(L(\mathcal{A}, q) \mid q \in Q)$ is defined inductively in the same way as for finite-state tree automata. A tree language is sk -recognizable if there is an sk -ta which recognizes this language, and it is s -recognizable if it is sk -recognizable for some $k \in \mathbb{N}$. An example of an $s2$ -recognizable tree language is the set of all binary trees with labels taken from \mathbb{N} such that every label is divisible by 2 or every label is divisible by 3 as, e.g., 2(4, 6) or 3(15, 18) (cf. Example 3.2).

By restricting the set of labels to a ranked alphabet Σ and just allowing, for every $\sigma \in \Sigma$, the characteristic mapping on $\{\sigma\}$ as predicate, we reobtain the classical finite-state tree automata. In [VB11a] it was proved that bottom-up sta are determinizable, that the class of s -recognizable tree languages is closed under the Boolean operations, and that the emptiness problem for s -recognizable tree languages is decidable provided the emptiness problem in the Boolean algebra of predicates is decidable.

Similarly, an stt is a top-down tree transducer [Tha70, Rou70, Eng75] except that

its input and output trees are built up over potentially infinite sets of (resp., input and output) labels. In the same way as for *sta*, we ensure finiteness by an a priori bound k on the maximal number of the successors of a node and by using a finite set of unary predicates. The right-hand side of each rule of a symbolic k -bounded tree transducer (*sk-tt*) contains unary functions, rather than explicit output symbols as in top-down tree transducers. These functions are then applied to the current input label and thereby produce the output labels. More formally, a rule has the form

$$q(\varphi(x_1, \dots, x_l)) \rightarrow u$$

where $0 \leq l \leq k$, q is a state, φ is a unary Boolean-valued predicate over the set of input labels, x_1, \dots, x_l are the usual variables that represent input subtrees, and u is a tree in which each internal node has at most k successors and is labeled by a unary function symbol; the leaves of u can be labeled alternatively by objects $q'(x_i)$ with state q' and $x_i \in \{x_1, \dots, x_l\}$. Clearly, the leaf labels of the form $q'(x_i)$ organize the recursive descent on the input tree as usual in a top-down tree transducer. The tree transformation computed by an *stt* is defined in the obvious way by means of a binary derivation relation. For instance, there is a (nondeterministic) *s2-tt* which transforms each binary tree over \mathbb{N} into a set of binary trees over \mathbb{N} such that a subtree $n(\xi_1, \xi_2)$ of the input tree is transformed into $m(\xi'_1, \xi'_2)$ where

- $m = n$, and ξ'_1 and ξ'_2 are transformations of ξ_1 and ξ_2 , respectively, or
- $m = \frac{n}{6}$ if n is divisible by 6, and both ξ'_1 and ξ'_2 are transformations of ξ_1 (cf. Example 5.3).

By restricting the predicates on the input labels to some ranked alphabet (as for *sta* above) and by only allowing unary functions such that each one produces a constant symbols from some ranked (output) alphabet, we reobtain top-down tree transducers.

Since *sta* and *stt* can check and manipulate data from an infinite set, they can be considered as tools for analyzing and transforming trees as they occur, e.g., in XML documents. Thus, the theoretical investigation of *sta* and *stt* is motivated by practical problems as e.g. type checking and inverse type checking.

In this paper we further develop the theory of *sta* and *stt*. We prove a characterization of *s*-recognizable tree languages in terms of (classical) recognizable tree languages and relabelings (Thm. 3.5). We compare the recognition power of *sta* with that of variable tree automata [MR11] (also cf. [GKS10]). More specifically, we characterize the tree language recognized by a variable tree automaton by the union of infinitely many *s*-recognizable tree languages (Prop. 3.8) and we show that *sta* and variable tree automata are incomparable with respect to recognition power (cf. Thm. 3.9). Moreover, as a generalization of (classical) regular tree grammars [Bra69] we introduce symbolic regular tree grammars and characterize *s*-regular tree languages in terms of regular tree languages and relabelings (Thm. 4.3). As a corollary, we obtain that *s*-recognizable tree languages are the same as *s*-regular tree languages (Thm. 4.4).

For *stt* we recall the concept of the syntactic composition from [VB11b]. We show that syntactic composition of two *stt* \mathcal{M} and \mathcal{N} computes the composition of the tree transformations computed by \mathcal{M} and \mathcal{N} , provided that (1) \mathcal{M} is deterministic or \mathcal{N} is

linear or (2) \mathcal{M} is total or \mathcal{N} is nondeleting (Thm. 5.8). Hereby, we generalize Baker's classical result [Bak79, Thm. 1].

Finally, we consider forward application and backward application of stt; these investigations are motivated by the (inverse) type checking problem (see among others [MSV03, AMN⁺03, EM03, MBPS05]). We show that the backward application of an *sk*-tt (which is the application of its inverse) to any *sk*-recognizable tree language yields an *sk*-recognizable tree language (Thm. 6.3). It is well-known that the forward application of linear top-down tree transducers preserves recognizability of tree languages (see e.g. [Tha69] or [GS84, Ch. IV, Cor. 6.6]). It is surprising that for stt the corresponding result does not hold, in fact there is a linear *sk*-tt of which the range is not an *sk*-recognizable tree language (Lm. 6.4). However, the application of simple and linear stt preserve s-recognizability (Thm. 6.5). As a corollary, we obtain that the type checking problem of simple and linear stt, as well as, the inverse type checking problem of arbitrary stt is decidable (Thm. 6.7).

Since the theory of sta and stt is based on concepts which are slightly different from the foundations of classical finite-state tree automata and tree transducers, we list them in detail in Section 2.

2 Preliminaries

2.1 General

The set of nonnegative integers is denoted by \mathbb{N} .

For a set A , we denote by $|A|$ and $\mathcal{P}(A)$ the cardinality and the set of all subsets of A . Moreover, we denote by ι_A the identical mapping over A . For a set I , an *I-indexed family over A* is a mapping $f : I \rightarrow A$. We denote the family f also by $(f_i \mid i \in I)$.

Let $\rho \subseteq A \times B$ be a relation. For every $A' \subseteq A$, we define $\rho(A') = \{b \in B \mid (a, b) \in \rho \text{ for some } a \in A'\}$. For another relation $\sigma \subseteq B \times C$, the composition of ρ and σ is the relation $\rho \circ \sigma = \{(a, c) \mid \exists (b \in B) : (a, b) \in \rho \text{ and } (b, c) \in \sigma\}$. The reflexive and transitive closure of a relation $\rho \subseteq A \times A$ is denoted by ρ^* .

2.2 Trees

In this paper we mainly consider trees over a nonempty and unranked set. We note that our concept of a tree differs from that of [VB11a, VB11b] in that we do not consider the empty tree as the base of the inductive definition.

Let U be a (possibly infinite) nonempty set, called the set of *labels*, and Y a further set. The *set of trees over U* (or: *U-trees*) *indexed by Y*, denoted by $T_U(Y)$, is the smallest subset T of $(U \cup Y \cup \{(\cdot, \cdot)\} \cup \{, \})^*$ such that (i) $(U \cup Y) \subseteq T$, and (ii) if $a \in U$ and $\xi_1, \dots, \xi_l \in T$ with $l \geq 1$, then $a(\xi_1, \dots, \xi_l) \in T$. If $Y = \emptyset$, then we write T_U for $T_U(Y)$. A *tree language over U* (or: *U-tree language*) is any subset of T_U .

Let Q be a set with $Q \cap U = \emptyset$. Then we denote by $Q(T_U(Y))$ the subset $\{q(\xi) \mid q \in Q, \xi \in T_U(Y)\}$ of $T_{Q \cup U}(Y)$.

We define the set of *positions in a U -tree* by means of the mapping $\text{pos} : T_U(Y) \rightarrow \mathcal{P}(\mathbb{N}^*)$ inductively on the argument $\xi \in T_U(Y)$ as follows: (i) if $\xi \in (U \cup Y)$, then $\text{pos}(\xi) = \{\varepsilon\}$, and (ii) if $\xi = a(\xi_1, \dots, \xi_l)$ for some $a \in U$, $l \geq 1$ and $\xi_1, \dots, \xi_l \in T_U(Y)$, then $\text{pos}(\xi) = \{\varepsilon\} \cup \{iv \mid 1 \leq i \leq l, v \in \text{pos}(\xi_i)\}$.

For every $\xi \in T_U(Y)$ and $w \in \text{pos}(\xi)$, the *label of ξ at w* , denoted by $\xi(w) \in (U \cup Y)$, the *subtree of ξ at w* , denoted by $\xi|_w \in T_U(Y)$, and the *rank at w* , denoted by $\text{rk}_\xi(w) \in \mathbb{N}$, are defined inductively as follows: (i) if $\xi \in (U \cup Y)$, then $\xi(\varepsilon) = \xi|_\varepsilon = \xi$, and $\text{rk}_\xi(\varepsilon) = 0$, and (ii) if $\xi = a(\xi_1, \dots, \xi_l)$ for some $a \in U$, $l \geq 1$ and $\xi_1, \dots, \xi_l \in T_U(Y)$, then $\xi(\varepsilon) = a$, $\xi|_\varepsilon = \xi$, and $\text{rk}_\xi(\varepsilon) = l$, and if $1 \leq i \leq l$ and $w = iv$, then $\xi(w) = \xi_i(v)$, $\xi|_w = \xi_i|_v$, and $\text{rk}_\xi(w) = \text{rk}_{\xi_i}(v)$.

Let $\xi \in T_U(Y)$ be a tree. For any $V \subseteq U$, we define $\text{pos}_V(\xi) = \{w \in \text{pos}(\xi) \mid \xi(w) \in V\}$. If $V = \{a\}$, then we write just $\text{pos}_a(\xi)$ for $\text{pos}_V(\xi)$. Moreover, for every $\zeta \in T_U(Y)$ and $w \in \text{pos}(\xi)$, we denote by $\xi[\zeta]_w$ the tree which is obtained by replacing the subtree $\xi|_w$ by ζ .

We will consider trees with variables and the substitution of trees for variables. For this, let $X = \{x_1, x_2, \dots\}$ be an infinite set of variables, disjoint with U , and let $X_l = \{x_1, \dots, x_l\}$ for every $l \in \mathbb{N}$. For trees $\xi \in T_U(X_l)$ and $\zeta_1, \dots, \zeta_l \in T_U(Y)$, we denote by $\xi[\zeta_1, \dots, \zeta_l]$ the tree which we obtain by replacing every occurrence of x_i by ζ_i for every $1 \leq i \leq l$. We note that $\xi[\zeta_1, \dots, \zeta_l] \in T_U(Y)$. Moreover, we denote by $C_U(X_l)$ the set of trees in $T_U(X_l)$ in which each variable x_i occurs exactly once and the order of variables from left to right is x_1, \dots, x_l . We call the elements of $C_U(X_l)$ *l -contexts*.

Finally, let $\xi \in T_U(Y)$ and $k \in \mathbb{N}$. We define the *rank* $\text{rk}(\xi)$ of ξ to be $\text{rk}(\xi) = \max\{\text{rk}_\xi(w) \mid w \in \text{pos}(\xi)\}$ and we say that ξ is *k -bounded* if $\text{rk}(\xi) \leq k$. We denote the set of all k -bounded U -trees indexed by Y by $T_U^{(k)}(Y)$. Clearly, $T_U^{(k)}(Y) \subset T_U^{(k+1)}(Y)$. A *k -bounded U -tree language* (or: *(U, k) -tree language*) is a subset of $T_U^{(k)}$. A U -tree language L is *bounded* if there is a $k \in \mathbb{N}$ such that L is k -bounded. Moreover, we define the set of *k -bounded l -contexts* to be $C_U^{(k)}(X_l) = C_U(X_l) \cap T_U^{(k)}(X_l)$.

In this paper U , V , and W will always denote arbitrary nonempty sets unless specified otherwise.

2.3 Tree transformations

Let $k \in \mathbb{N}$. A *k -bounded tree transformation* (or: *k -tree transformation*) is a mapping $\tau : T_U^{(k)} \rightarrow \mathcal{P}(T_V^{(k)})$ (or: alternatively, a relation $\tau \subseteq T_U^{(k)} \times T_V^{(k)}$). A *tree transformation* is a k -tree transformation for some $k \in \mathbb{N}$. If for every $\xi \in T_U^{(k)}$, there is exactly one $\zeta \in T_V^{(k)}$ such that $(\xi, \zeta) \in \tau$ (i.e., τ is a mapping), then we also write $\tau : T_U^{(k)} \rightarrow T_V^{(k)}$. The *inverse* τ^{-1} , the *domain* $\text{dom}(\tau)$, and the *range* $\text{range}(\tau)$ of a tree transformation τ are defined in the standard way.

Let $\tau \subseteq T_U^{(k)} \times T_V^{(k)}$ be a tree transformation, $L \subseteq T_U^{(k)}$ and $L' \subseteq T_V^{(k)}$ tree languages. The *forward application* (or just: *application*) of τ to L is the tree language $\tau(L) = \{\zeta \in T_V^{(k)} \mid \exists(\xi \in L) : (\xi, \zeta) \in \tau\}$. The *backward application* of τ to L' is the tree language $\tau^{-1}(L')$ (which is the forward application of τ^{-1} to L').

We extend the above concepts and the composition of tree transformations to classes of tree transformations and classes of tree languages in a natural way. For instance, if \mathcal{C} and \mathcal{C}' are classes of k -tree transformations, and \mathcal{L} is a class of k -tree languages, then we define $\mathcal{C} \circ \mathcal{C}' = \{\tau \circ \sigma \mid \tau \in \mathcal{C} \text{ and } \sigma \in \mathcal{C}'\}$ and $\mathcal{C}(\mathcal{L}) = \{\tau(L) \mid \tau \in \mathcal{C} \text{ and } L \in \mathcal{L}\}$.

A *relabeling* is a mapping $\tau : U \rightarrow \mathcal{P}(V)$ such that $\tau(a)$ is recursive and it is decidable if $\tau(a) = \emptyset$ for every $a \in U$; it is called *deterministic* if $\tau(a)$ is a singleton for every $a \in U$. Let $k \in \mathbb{N}$. The *k -tree relabeling (induced by τ)* is the mapping $\tau' : T_U^{(k)} \rightarrow \mathcal{P}(T_V^{(k)})$, defined by

$$\tau'(a(\xi_1, \dots, \xi_l)) = \{b(\zeta_1, \dots, \zeta_l) \mid b \in \tau(a) \text{ and } \zeta_i \in \tau'(\xi_i) \text{ for } 1 \leq i \leq l\}.$$

Then the mapping τ' is extended to $\tau'' : \mathcal{P}(T_U^{(k)}) \rightarrow \mathcal{P}(T_V^{(k)})$ by $\tau''(L) = \bigcup_{\xi \in L} \tau'(\xi)$ for every $L \in \mathcal{P}(T_U^{(k)})$.

We note that the composition of two k -tree relabelings τ'_1 and τ'_2 is again a k -tree relabeling. In fact, if $\tau_1 : U \rightarrow \mathcal{P}(V)$ and $\tau_2 : V \rightarrow \mathcal{P}(W)$, then $\tau_1 \circ \tau_2$ induces $\tau'_1 \circ \tau'_2$. In the sequel, we drop the primes from τ' and τ'' and identify both mappings with τ .

2.4 Predicates and label structures

A (unary) *predicate over U* is a mapping $\varphi : U \rightarrow \{0, 1\}$. We denote by $\text{Pred}(U)$ the set of all predicates over U . Let $\varphi \in \text{Pred}(U)$ be a predicate. We introduce the notation $\llbracket \varphi \rrbracket$ for $\{a \in U \mid \varphi(a) = 1\}$.

We define the operations \neg , \wedge , and \vee over $\text{Pred}(U)$ in the obvious way and extend \wedge and \vee to finite families $(\varphi_i \mid i \in I)$ of predicates in $\text{Pred}(U)$. In particular, $\llbracket \bigwedge_{i \in \emptyset} \varphi_i \rrbracket = U$ and $\llbracket \bigvee_{i \in \emptyset} \varphi_i \rrbracket = \emptyset$.

Let $\Phi \subseteq \text{Pred}(U)$ be a finite set of recursive predicates such that $\llbracket \varphi \rrbracket = \emptyset$ is decidable for every $\varphi \in \Phi$. We call the pair (U, Φ) a *label structure*. The *Boolean closure of Φ* , denoted by $\text{BC}(\Phi)$, is the smallest set $B \subseteq \text{Pred}(U)$ such that

- (i) $\Phi \subseteq B$,
- (ii) $\perp, \top \in B$ where $\top(a) = 1$ and $\perp(a) = 0$ for every $a \in U$, and
- (iii) for every $\varphi, \psi \in B$, the predicates $\neg\varphi$, $\varphi \wedge \psi$, and $\varphi \vee \psi$ are in B .

It is clear that $(\text{BC}(\Phi), \wedge, \vee, \neg, \perp, \top)$ is a Boolean algebra for every $\Phi \subseteq \text{Pred}(U)$.

2.5 Tree automata, tree grammars, tree transducers

We assume that the reader is familiar with the basic concepts of the theory of (classical) tree automata and tree transducers which can be found among others in [GS84, GS97]

and [CDG⁺97]. In particular, we freely use the concept of a ranked alphabet, a tree language over a ranked alphabet, a finite-state tree automaton, a recognizable tree language, a regular tree grammar, a regular tree language, a top-down tree transducer, and of a tree transformation. Here we recall only some notations.

A *ranked alphabet* is a finite set Σ equipped with a rank mapping $\text{rk}_\Sigma : \Sigma \rightarrow \mathbb{N}$. We define $\Sigma_l = \{\sigma \in \Sigma \mid \text{rk}_\Sigma(\sigma) = l\}$ ($l \geq 0$) and $\text{maxrk}(\Sigma) = \max\{\text{rk}_\Sigma(\sigma) \mid \sigma \in \Sigma\}$. It is clear that every tree $\xi \in T_\Sigma$ is $\text{maxrk}(\Sigma)$ -bounded.

A *finite-state tree automaton* is a system $\mathcal{A} = (Q, \Sigma, \delta, F)$, where Q is a finite, nonempty set (states), Σ is a ranked alphabet, $\delta = (\delta_\sigma \mid \sigma \in \Sigma)$ is the family of sets of transitions, i.e., $\delta_\sigma \subseteq Q^l \times Q$ for every $l \in \mathbb{N}$ and $\sigma \in \Sigma$ with $\text{rk}_\Sigma(\sigma) = l$, and $F \subseteq Q$ is the set of final states. The set of trees recognized by \mathcal{A} is denoted by $L(\mathcal{A})$. A tree language $L \subseteq T_\Sigma$ is *recognizable* if there is a finite-state tree automaton \mathcal{A} such that $L = L(\mathcal{A})$.

A *regular tree grammar* is a tuple $\mathcal{G} = (Q, \Sigma, q_0, R)$ where Q is a finite set of states¹, Σ is a ranked alphabet, $q_0 \in Q$ (initial state), and R is a finite set of rules of the form $q \rightarrow u$ with $q \in Q$ and $u \in T_\Sigma(Q)$. The derivation relation induced by \mathcal{G} and the tree language generated by \mathcal{G} are denoted by $\Rightarrow_{\mathcal{G}}$ and $L(\mathcal{G})$, respectively. We will also consider *reduced* regular tree grammars and regular tree grammars in *normal form* in the sense of [CDG⁺97].

3 Symbolic tree automata

In this section we formalize our adaptation of the concept of a symbolic tree automaton from [VB11a] and compare our model with the original one. Then we prove basic properties of sta. Finally, we compare the recognition capacity of sta with that of variable tree automata.

3.1 Definition of sta

Definition 3.1 Let $k \in \mathbb{N}$. A *symbolic k -bounded tree automaton* (sk -ta) is a tuple $\mathcal{A} = (Q, U, \Phi, F, R)$ where

- Q is a finite, nonempty set (states),
- (U, Φ) is a label structure,
- $F \subseteq Q$ (set of final states), and
- R is a finite set of rules of the form $(q_1 \dots q_l, \varphi, q)$ where $0 \leq l \leq k$, $q_1, \dots, q_l, q \in Q$, and $\varphi \in \text{BC}(\Phi)$.

Let $\rho = (q_1 \dots q_l, \varphi, q) \in R$. We call $(q_1 \dots q_l)$ the *left-hand side*, φ the *guard*, and q the *right-hand side* of the rule ρ , and denote them by $\text{lhs}(\rho)$, $\text{grd}(\rho)$, and $\text{rhs}(\rho)$

¹Usually these symbols are called nonterminals; but since this notion leads to misunderstandings in the application area of natural language processing, we prefer to call these symbols states.

respectively. Clearly, every sk -ta is an $s(k+1)$ -ta. By a *symbolic tree automaton* (sta) we mean an sk -ta for some $k \in \mathbb{N}$.

For every $q \in Q$, we define the tree language $L(\mathcal{A}, q) \subseteq T_U^{(k)}$ recognized by \mathcal{A} in state q , as follows. The family $(L(\mathcal{A}, q) \mid q \in Q)$ is the smallest Q -family $(L_q \mid q \in Q)$ of tree languages such that

- (i) if $a \in U$, $(\varepsilon, \varphi, q) \in R$, and $a \in \llbracket \varphi \rrbracket$, then $a \in L_q$, and
- (ii) if $a \in U$, $(q_1 \dots q_l, \varphi, q) \in R$ with $1 \leq l \leq k$ and $a \in \llbracket \varphi \rrbracket$, and $\xi_1 \in L(\mathcal{A}, q_1), \dots, \xi_l \in L(\mathcal{A}, q_l)$, then $a(\xi_1, \dots, \xi_l) \in L_q$.

The condition that all predicates in Φ (and hence in $\text{BC}(\Phi)$) are recursive ensure that we can decide whether $\xi \in L(\mathcal{A}, q)$ for every $q \in Q$ and $\xi \in T_U^{(k)}$.

The *tree language recognized by \mathcal{A}* , denoted by $L(\mathcal{A})$, is the set

$$L(\mathcal{A}) = \bigcup_{q \in F} L(\mathcal{A}, q) .$$

A tree language $L \subseteq T_U^{(k)}$ is *symbolically k -recognizable* (sk -recognizable) if there is an sk -ta \mathcal{A} such that $L(\mathcal{A}) = L$. We denote the class of all sk -recognizable U -tree languages by $\text{REC}^{(k)}(U)$. Moreover, we call a tree language *s -recognizable* if it is sk -recognizable for some $k \in \mathbb{N}$.

Two sk -ta \mathcal{A} and \mathcal{B} are *equivalent* if $L(\mathcal{A}) = L(\mathcal{B})$.

Example 3.2 We give an example of an sta. For this we consider the set $U = \mathbb{N}$ and the 2-bounded tree language

$$L = \{ \xi \in T_{\mathbb{N}}^{(2)} \mid \xi \text{ is binary and } (\forall w \in \text{pos}(\xi) : \xi(w) \text{ is divisible by } 2) \vee (\forall w \in \text{pos}(\xi) : \xi(w) \text{ is divisible by } 3) \}$$

where a tree ξ is binary if $\text{rk}_{\xi}(w) \in \{0, 2\}$ for every $w \in \text{pos}(\xi)$. For instance, the trees $2(4, 6)$ and $3(15, 18)$ are in L .

The following s2-ta $\mathcal{A} = (Q, \mathbb{N}, \Phi, F, R)$ recognizes L :

- $Q = F = \{2, 3\}$,
- $\Phi = \{\text{div}(2), \text{div}(3)\}$ with $\llbracket \text{div}(i) \rrbracket = \{n \in \mathbb{N} \mid n \text{ is divisible by } i\}$,
- for every $i \in \{2, 3\}$ the transitions $(\varepsilon, \text{div}(i), i)$ and $(i i, \text{div}(i), i)$ are in R .

For instance, $6(12, 18) \in L(\mathcal{A}, 2) \cap L(\mathcal{A}, 3)$.

Our definition of sta slightly differs from the one in [VB11a] in the following two points.

1. They fix a Boolean algebra B of predicates in advance, and then they make a theory of sta only over B . We are free to choose predicates whenever we need them.
2. In [VB11a] no bound on the number of successors of nodes is mentioned. In our definition we put an explicit bound on this number in order to guarantee closure of sk -recognizable tree languages under complement.

We note that this closure under complement is not discussed clearly in [VB11a]. The root of the ambiguity is that the complement of a tree language, appearing in Prop. 3 of that paper, is not defined. If the complement of a tree language L is meant to be $\mathcal{U}^{T(\sigma)} \setminus L$ (as maybe is suggested by the definition of the complement of a predicate [VB11a, p.146]), which corresponds to $T_U \setminus L$ in our notation, then the class of s-recognizable tree languages is not closed under complement as stated in [VB11a, Prop. 3, Thm. 1]. This can be seen easily as follows. Let L be an s-recognizable tree language (in the sense of [VB11a] or of the present paper). Then obviously L is bounded, while the tree language $\mathcal{U}^{T(\sigma)} \setminus L$ is not bounded. Hence the latter cannot be s-recognizable.

However, if we define the complement of a k -bounded tree language L with respect to $T_U^{(k)}$, i.e., to be $T_U^{(k)} \setminus L$, then the class of sk -recognizable tree languages is closed under complement (by using the appropriate adaptations of [VB11a, Prop. 3, Thm. 1]).

3.2 Basic properties

Here we give a characterization of s-recognizable tree languages in terms of (classical) recognizable tree languages and tree relabelings. Moreover, we introduce uniform tree languages and show that any uniform tree language is not s-recognizable.

We will need the following obvious fact.

Observation 3.3 Both \emptyset and the set $T_U^{(k)}$ are sk -recognizable for every set U and $k \in \mathbb{N}$.

In the following we give a characterization of s-recognizable tree languages in terms of recognizable tree languages and relabelings. First we prove the next lemma.

Lemma 3.4

1. For every sk -recognizable tree language L we can effectively construct a k -bounded recognizable tree language L' and a k -tree relabeling τ such that $L = \tau(L')$.
2. For every k -bounded recognizable tree language L' and k -tree relabeling τ we can effectively construct an sk -recognizable tree language L such that $L = \tau(L')$.

PROOF First assume that $L = L(\mathcal{A})$ for some sk -ta $\mathcal{A} = (Q, U, \Phi, F, R)$. We construct the finite-state tree automaton $\mathcal{A}' = (Q, \Sigma, \delta, F)$, where

- $\Sigma_l = \{[\varphi, l] \mid (q_1 \dots q_l, \varphi, q) \in R \text{ for some } q_1, \dots, q_l, q \in Q\}$, $0 \leq l \leq k$ and
- $\delta_{[\varphi, l]} = \{(q_1 \dots q_l, q) \mid (q_1 \dots q_l, \varphi, q) \in R\}$.

It should be clear that $L(\mathcal{A}')$ is k -bounded. Moreover, we define the relabeling $\tau : \Sigma \rightarrow \mathcal{P}(U)$ by $\tau([\varphi, l]) = \llbracket \varphi \rrbracket$ for every $[\varphi, l] \in \Sigma$.

We can easily prove the following statement by induction on trees: for every $\xi \in T_U^{(k)}$ and $q \in Q$ we have

$$\xi \in L(\mathcal{A}, q) \iff \exists(\zeta \in L(\mathcal{A}', q)) \text{ such that } \xi \in \tau(\zeta),$$

which proves that $L(\mathcal{A}) = \tau(L(\mathcal{A}'))$.

For the proof of the other implication, let us consider a finite-state tree automaton $\mathcal{A}' = (Q, \Sigma, \delta, F)$ such that $L(\mathcal{A}')$ is k -bounded. We may assume without loss of generality that $\text{maxrk}(\Sigma) \leq k$. Moreover, let $\tau : \Sigma \rightarrow \mathcal{P}(U)$ be a relabeling. We construct the sk -ta $\mathcal{A} = (Q, U, \Phi, \delta', F)$, where Φ and R are defined as follows:

- $\Phi = \{\varphi_\sigma \mid \sigma \in \Sigma\}$, where $\llbracket \varphi_\sigma \rrbracket = \tau(\sigma)$ for every $\sigma \in \Sigma$,
- $\delta' = \{(q_1 \dots q_l, \varphi_\sigma, q) \mid (q_1 \dots q_l, q) \in \delta_\sigma \text{ for some } l \geq 0, \sigma \in \Sigma_l\}$.

It should be clear that $L(\mathcal{A}) = \tau(L(\mathcal{A}'))$. ■

By letting τ be the identity mapping in Lemma 3.4(2), we obtain that each recognizable tree language is also s -recognizable. A further consequence of Lemma 3.4 is the mentioned characterization.

Theorem 3.5 *A tree language L is sk -recognizable if and only if it is the image of a k -bounded recognizable tree language under a k -tree relabeling.*

Using the above characterization result, we can easily give examples of bounded tree languages that are not s -recognizable. For an infinite U , we call a tree language $L \subseteq T_U$ *uniform* if it satisfies the following conditions:

- (a) L is infinite,
- (b) all trees in L have the same shape, i.e., for every $\xi, \zeta \in L$, we have $\text{pos}(\xi) = \text{pos}(\zeta)$, and
- (c) for every $\xi \in L$, there is an $a \in U$ such that $\xi(w) = a$ for every $w \in \text{pos}(\xi)$.

For instance, the tree language $L_2 = \{a(a) \mid a \in U\}$ is uniform provided U is infinite. In particular, $\text{pos}(\xi) = \{\varepsilon, 1\}$ for every $\xi \in L_2$. Now we can prove the following.

Lemma 3.6 Let $L \subseteq T_U^{(k)}$ be a uniform tree language such that $|\text{pos}(\xi)| > 1$ for every $\xi \in L$. Then L is not sk -recognizable.

PROOF We prove by contradiction, i.e., we assume that L is sk -recognizable. By Lemma 3.4(1), there is a ranked alphabet Σ , a k -bounded recognizable tree language $L' \subseteq T_\Sigma$, and a relabeling $\tau : \Sigma \rightarrow \mathcal{P}(U)$ such that $L = \tau(L')$. Since τ , being a k -tree relabeling, preserves the shape of trees, the shape of all trees in L' is the same as that of all trees in L . Then, since Σ is a finite set, L' is also finite. Finally, since L is infinite, there are a tree $\zeta \in L'$, different positions v and w of ζ , and different labels $a, b \in U$ such that $a \in \tau(\zeta(v))$ and $b \in \tau(\zeta(w))$. Then there is a tree $\xi \in \tau(\zeta)$ such that $\xi(v) = a$ and $\xi(w) = b$, which contradicts to condition (c) for uniform tree languages. ■

By the above lemma, for an infinite U , the 1-bounded tree language L_2 is not s -recognizable.

3.3 Comparison with variable tree automata

In [GKS10] another automaton model with infinite input alphabet was introduced. It is called variable (string) automaton. In [MR11] this concept has been extended to variable tree automata over infinite alphabets (vta). The theory of vta is different from that of sta, e.g., the class of s-recognizable tree languages is closed under complement (cf. [VB11a, Prop. 3]) which does not hold for the class of v-recognizable tree languages (cf. [MR11, Cor. 2], and [GKS10, Thm. 2]). Moreover, every sta is determinizable (cf. [VB11a, Thm. 1]), whereas not every variable (string) automata over infinite alphabets is determinizable (cf. [GKS10, Sec. 4.1]).

In this section we will compare the recognition power of sta and of vta. In order to be able to do so, (1) we modify our sta model a bit and then (2) we recall the concepts of vta from [MR11] in a slightly adapted form.

By a *ranked set* we mean a nonempty set U of symbols such that with each symbol $a \in U$ an element in \mathbb{N} , the *rank of a* , is associated. For every $l \geq 0$, we denote by U_l the set of all symbols of U with rank l .

The set of trees over a ranked set U is defined in the obvious way.

An *sk-ta* $\mathcal{A} = (Q, U, \Phi, F, R)$ is a *ranked sk-ta (rsk-ta)* if

- U is a ranked set, and
- Φ is a finite set of predicates (we do not require that predicates in Φ are recursive and that the emptiness problem in Φ is decidable).

The concepts of an rsk-recognizable tree language and an rs-recognizable tree language are defined in the obvious way.

Now we prepare the definition of a variable tree automaton. Let U and V be ranked sets. A *rank preserving relabeling (r-relabeling)* from U to V is a mapping $\tau : U \rightarrow \mathcal{P}(V)$ such that $\tau(U_l) \subseteq V_l$ ($l \geq 0$). Then τ extends to trees in the same way as in case of k -tree relabelings (cf. Section 2.3). We note that $\tau(a)$ need not be recursive and $\tau(a) = \emptyset$ need not be decidable for $a \in U$.

Let Σ be a ranked alphabet, V an infinite ranked set, A , Z , and Y ranked alphabets. We say that the collection (A, Z, Y) is a *valid partitioning of Σ for V* if

- $A = \Sigma \cap V$, and $\Sigma_l = A_l \cup Z_l \cup Y_l$ for every $l \geq 0$,
- A , Z , and Y are pairwise disjoint, and
- $|Y_l| \leq 1$ for every $0 \leq l \leq \maxrk(\Sigma)$.

The elements of Z and Y are called *bounded variable symbols* and *free variable symbols*.

Let (A, Z, Y) be a valid partitioning of Σ for V and $\tau : \Sigma \rightarrow \mathcal{P}(V)$ an r-relabeling. We say that τ is *(A, Z, Y) -valid* if

- (i) τ is the identity on A ,
- (ii) $|\tau(z)| = 1$ for every $z \in Z$,
- (iii) τ is injective on Z and $A_l \cap \tau(Z_l) = \emptyset$ for every $l \geq 0$, and

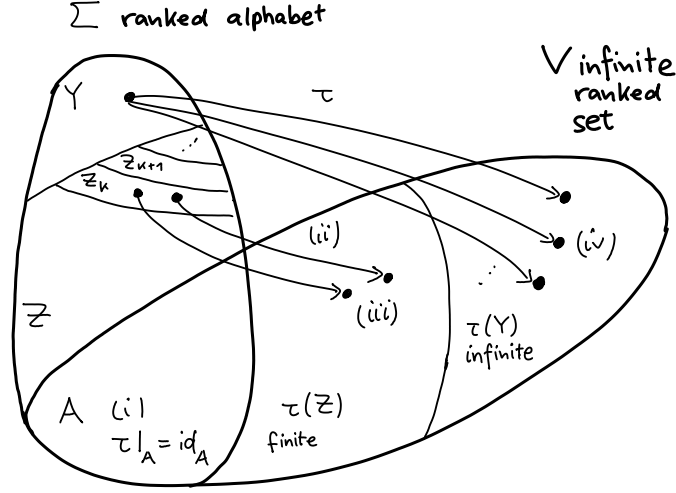


Figure 1: An (A, Z, Y) -valid r-relabeling $\tau : \Sigma \rightarrow \mathcal{P}(V)$.

(iv) $\tau(y) = V_l \setminus (A_l \cup \tau(Z_l))$ for every $l \geq 0$ and $y \in Y_l$.

We denote the set of all (A, Z, Y) -valid r-relabelings by $\text{VR}(A, Z, Y)$. In Fig. 1 we illustrate the conditions for a valid r-relabeling.

A *variable tree automaton (vta)* is a tuple $\mathcal{B} = (\mathcal{A}, V, A, Z, Y)$ where

- $\mathcal{A} = (Q, \Sigma, \delta, F)$ is a finite-state tree automaton,
- V is an infinite ranked set,
- (A, Z, Y) is a valid partitioning of Σ for V .

The tree language *recognized* by \mathcal{B} is the set

$$L(\mathcal{B}) = \bigcup (\tau(L(\mathcal{A})) \mid \tau \in \text{VR}(A, Z, Y)) .$$

We call a tree language *v-recognizable* if it can be recognized by a vta.

Proposition 3.7 *For every ranked alphabet Σ , every recognizable tree language L over Σ is also v-recognizable.*

PROOF Let \mathcal{A} be a finite-state tree automaton (with input ranked alphabet Σ) such that $L = L(\mathcal{A})$. Moreover, let V be an arbitrary infinite ranked set such that $\Sigma \subseteq V$. We observe that $(\Sigma, \emptyset, \emptyset)$ is a valid partitioning of Σ for V , hence $\mathcal{B} = (\mathcal{A}, V, \Sigma, \emptyset, \emptyset)$ is a vta over V . Moreover, the only $(\Sigma, \emptyset, \emptyset)$ -valid r-relabeling is the identity mapping over Σ . Hence we obtain that $L(\mathcal{B}) = L(\mathcal{A})$. ■

Next we relate v-recognizable tree languages and s-recognizable tree languages.

Proposition 3.8 *Let $\mathcal{B} = (\mathcal{A}, V, A, Z, Y)$ be a vta and \mathcal{A} have input ranked alphabet Σ . Then there is a family $(L_\tau \mid \tau \in \text{VR}(A, Z, Y))$ of rsk-recognizable tree languages over V such that*

$$L(\mathcal{B}) = \bigcup (L_\tau \mid \tau \in \text{VR}(A, Z, Y)),$$

where $k = \text{maxrk}(\Sigma)$.

PROOF We note that every $\tau \in \text{VR}(A, Z, Y)$ is a k -tree relabeling. Hence, by an easy adaptation of Lemma 3.4(2), we have that the tree language $\tau(L(\mathcal{A}))$ is recognizable by an rsk-ta \mathcal{A}_τ . Hence the statement holds with $L_\tau = L(\mathcal{A}_\tau)$. ■

In spite of the above fact, we can prove the following statement.

Theorem 3.9 *The class of v-recognizable tree languages and the class of s-recognizable tree languages are incomparable with respect to inclusion.*

PROOF a) We give a v-recognizable tree language and show that it is not rs-recognizable. For this, let $V = V_0 \cup V_1$ be an infinite ranked set such that $V_0 = \{c\}$, and consider the ranked alphabet $\Sigma = \Sigma_0 \cup \Sigma_1$ with $\Sigma_0 = \{c\}$ and $\Sigma_1 = \{z\}$ with $z \notin V$. Let \mathcal{A} be a finite-state tree automaton with input ranked alphabet Σ such that $L(\mathcal{A}) = \{zzc\}$ (where parentheses are omitted). Now $(\{c\}, \{z\}, \emptyset)$ is a valid partitioning of Σ for V , hence $\mathcal{B} = (\mathcal{A}, V, \{c\}, \{z\}, \emptyset)$ is a vta over V . Since every $(\{c\}, \{z\}, \emptyset)$ -valid r-relabeling takes z to an element $a \in V_1$, we have $L(\mathcal{B}) = \{aac \mid a \in V_1\}$. By an easy adaptation of Lemma 3.6 we obtain that $L(\mathcal{B})$ is not rs-recognizable.

b) We give an rs1-recognizable tree language and show that it is not v-recognizable.

For this, let $\Sigma = \Sigma_0 \cup \Sigma_1$ be a ranked alphabet with $\Sigma_0 = \{a\}$ and $\Sigma_1 = \{e, o\}$, and let $V = V_0 \cup V_1$ be an infinite ranked set with $V_0 = \{a\}$ and $V_1 = \mathbb{N}$. Consider the recognizable tree language

$$L = \{a, oa, eoa, oeo, eoeo, \dots\}$$

over Σ (where parentheses are omitted), and the r-relabeling $\bar{\tau}$ defined by

$$\bar{\tau}(a) = a, \bar{\tau}(e) = \text{set of all even numbers}, \text{ and } \bar{\tau}(o) = \text{set of all odd numbers}.$$

By the adaptation of Lemma 3.4(2) to rsk-ta, we obtain that the tree language $\bar{\tau}(L)$ can be recognized by an rs1-ta. Roughly speaking, $\bar{\tau}(L)$ consists of all sequences of the form $n_k \dots n_2 n_1 a$, where $k \geq 0$, n_i is an odd number if i is odd and an even number otherwise. We show by contradiction that $\bar{\tau}(L)$ cannot be recognized by any vta.

For this, assume that there is a vta $\mathcal{B} = (\mathcal{A}, V, A, Z, Y)$, where the input alphabet of \mathcal{A} is $\Sigma = A \cup Z \cup Y$ such that $L(\mathcal{B}) = \bar{\tau}(L)$. We may assume without loss of generality that $Y = \emptyset$, which can be seen as follows. Assume that $Y = \{y\}$, and that there is a tree $\xi \in L(\mathcal{A})$ such that y occurs in ξ at the position 1^i (see Section 2.2 for the definition of a position). Moreover, let $\tau : \Sigma \rightarrow \mathcal{P}(V)$ be an (A, Z, Y) -valid r-relabeling. Since $\tau(y)$ contains both even and odd numbers, there are trees ζ and ζ' in the set $\tau(\xi)$ such that at the position 1^i of ζ and ζ' there is an odd number and an even number, respectively. On the other hand, $\tau(\xi) \subseteq L(\mathcal{B})$, which is a contradiction

because at the position 1^i of every tree in $L(\mathcal{B})$ there is either an odd number or an even number (depending on whether i is odd or even).

Hence $Y = \emptyset$. Now assume that $|A \cup Z| = m$. Then every tree in $\xi \in L(\mathcal{A})$ consists of at most m different symbols. Moreover, by the definition of the (A, Z, Y) -valid relabeling, for every $\tau \in \text{VR}(A, Z, Y)$, each tree in $\tau(\xi)$ consists of m different symbols. It means, each tree in $L(\mathcal{B})$ consists of m different symbols, which contradicts to the much more flexible form of trees in $\bar{\tau}(L)$. \blacksquare

4 Symbolic regular tree grammars

In this section we introduce symbolic regular tree grammars and show that they are semantically equivalent to sta.

Definition 4.1 A *symbolic k -bounded regular tree grammar* (*sk-rtg*) is a tuple $\mathcal{G} = (Q, U, \Phi, q_0, R)$, where

- Q is a finite set (states²),
- (U, Φ) is a label structure,
- $q_0 \in Q$ (initial state), and
- R is a finite set of rules of the form $q \rightarrow u$ where $q \in Q$ and $u \in T_{\text{BC}(\Phi)}^{(k)}(Q)$.

By a *symbolic regular tree grammar* (srtg) we mean an *sk-rtg* for some $k \in \mathbb{N}$.

The *sk-rtg* $\mathcal{G} = (Q, U, \Phi, q_0, R)$ induces the derivation relation $\Rightarrow_{\mathcal{G}} \subseteq T_U^{(k)}(Q) \times T_U^{(k)}(Q)$ defined by $\xi_1 \Rightarrow_{\mathcal{G}} \xi_2$ iff there is a position $w \in \text{pos}_q(\xi_1)$ and a rule $q \rightarrow u$ in R , such that $\xi_2 = \xi_1[u']_w$, where u' is obtained from u by replacing every occurrence of $\varphi \in \text{BC}(\Phi)$ by some $a \in \llbracket \varphi \rrbracket$. (The condition that all predicates in Φ are recursive makes the relation $\Rightarrow_{\mathcal{G}}$ recursive.)

The k -bounded tree language $L(\mathcal{G}, q)$ generated by \mathcal{G} from a state $q \in Q$ is the set

$$L(\mathcal{G}, q) = \{\xi \in T_U^{(k)} \mid q \Rightarrow_{\mathcal{G}}^* \xi\} .$$

The *tree language generated by \mathcal{G}* , denoted by $L(\mathcal{G})$, is the set $L(\mathcal{G}, q_0)$. A tree language $L \subseteq T_U^{(k)}$ is called *symbolically k -regular* (for short: *sk-regular*) if there is an *sk-rtg* \mathcal{G} such that $L = L(\mathcal{G})$. Moreover, a tree language is *s-regular* if it is *sk-regular* for some $k \in \mathbb{N}$.

Two *sk-rtg* \mathcal{G}_1 and \mathcal{G}_2 are *equivalent* if $L(\mathcal{G}_1) = L(\mathcal{G}_2)$.

In the following we give a characterization of *s-regular* tree languages in terms of regular tree languages and relabelings.

²In classical regular tree grammars these elements are called nonterminals.

Lemma 4.2

1. For every sk -regular tree language L we can effectively construct a k -bounded regular tree language L' and a k -tree relabeling τ such that $L = \tau(L')$.
2. For every k -bounded regular tree language L' and k -tree relabeling τ we can effectively construct an sk -regular tree language L such that $L = \tau(L')$.

PROOF First let $L = L(\mathcal{G})$ for some sk -rtg $\mathcal{G} = (Q, U, \Phi, q_0, R)$. We construct the regular tree grammar $\mathcal{G}' = (Q, \Sigma, q_0, R')$ as follows.

- For every $l \leq k$, let

$$\Sigma_l = \{[\varphi, l] \mid \exists (q \rightarrow u) \in R, w \in \text{pos}_{\text{BC}(\Phi)}(u) : u(w) = \varphi \text{ and } \text{rk}_w(u) = l\},$$

- and let R' be the set of all rules $q \rightarrow u'$ such that there is a rule $q \rightarrow u$ in R and u' is obtained from u as follows: for every $w \in \text{pos}(u)_{\text{BC}(\Phi)}$, we replace $u(w)$ by $[u(w), \text{rk}_w(u)]$.

It is obvious that $L(\mathcal{G}')$ is k -bounded. Moreover, we let the relabeling $\tau : \Sigma \rightarrow \mathcal{P}(U)$ be defined by $\tau([\varphi, l]) = \llbracket \varphi \rrbracket$ for every $0 \leq l \leq k$ and $[\varphi, l] \in \Sigma_l$.

We can prove the following statement by tree induction:

for every $\zeta \in T_U^{(k)}$ and $q \in Q$:
 $q \Rightarrow_{\mathcal{G}}^* \zeta$ iff there is a $\xi \in T_{\Sigma}$: $q \Rightarrow_{\mathcal{G}'}^* \xi$ and $\zeta \in \tau(\xi)$.

Then $L(\mathcal{G}) = \tau(L(\mathcal{G}'))$.

For the proof of Statement 2, let us consider a regular tree grammar $\mathcal{G}' = (Q, \Sigma, q_0, R)$ such that $L(\mathcal{G}')$ is k -bounded and a relabeling $\tau : \Sigma \rightarrow \mathcal{P}(U)$. We may assume without loss of generality that $\text{maxrk}(\Sigma) \leq k$. We construct the sk -rtg $\mathcal{G} = (Q, U, \Phi, q_0, R')$, where Φ and R' are defined as follows:

- $\Phi = \{\varphi_{\sigma} \mid \sigma \in \Sigma\}$, where $\llbracket \varphi_{\sigma} \rrbracket = \tau(\sigma)$ for every $\sigma \in \Sigma$,
- R' : if $q \rightarrow u$ is in R , then $q \rightarrow u'$ is in R' where u' is obtained from u by replacing every σ by φ_{σ} .

It should be that $L(\mathcal{G}) = \tau(L(\mathcal{G}'))$. ■

It follows from Lemma 4.2(2) that each regular tree language is also s -regular. We obtain this by letting τ be the identity mapping. As another consequence of Lemma 4.2, we obtain the following characterization result.

Theorem 4.3 *A tree language L is sk -regular if and only if it is the image of a k -bounded regular tree language under a k -tree relabeling.*

We can also show that s -recognizable tree languages are the same as s -regular tree languages.

Theorem 4.4 *A tree language is s -recognizable if and only if it is s -regular.*

PROOF It follows directly from Lemmas 3.4 and 4.2 and the fact that a tree language is recognizable if and only if it can be generated by a regular tree grammar (cf. e.g. Theorem 3.6 in Chapter II of [GS84]). ■

In the rest of this section we show some useful transformations on *sk*-rtg which preserve the generated tree language. For this, we need some preparation.

Let $\mathcal{G} = (Q, U, \Phi, q_0, R)$ be an *sk*-rtg. A rule $q \rightarrow u$ in R is *feasible* if, for every predicate φ which occurs in u , we have $\llbracket \varphi \rrbracket \neq \emptyset$, and we call \mathcal{G} *clean* if all its rules are feasible. It is obvious that rules which are non-feasible cannot be used in any valuable derivations. Hence, they can be dropped from R without any effect on the generated tree language $L(\mathcal{G})$. Moreover, it is decidable whether a rule is feasible or not due to the fact that the emptiness of predicates in Φ is decidable. Summarizing up, for every *sk*-rtg we can construct an equivalent one, which is clean.

The *sk*-rtg \mathcal{G} is in *normal form* if every rule has the form $q \rightarrow \varphi(q_1, \dots, q_l)$ for some $l \leq k$, $\varphi \in \text{BC}(\Phi)$, and $q_1, \dots, q_l \in Q$. A state $q \in Q$ is *reachable* if there is a tree $\xi \in T_U^{(k)}(Q)$ such that $q_0 \Rightarrow_{\mathcal{G}}^* \xi$ and q occurs in ξ . Moreover, the state q is *productive*, if $L(\mathcal{G}, q) \neq \emptyset$. Finally, \mathcal{G} is *reduced* if all its states are reachable and productive. We can prove the following result.

Lemma 4.5 For every *sk*-rtg there is an equivalent reduced *sk*-rtg which is in normal form.

PROOF Let $\mathcal{G} = (Q, U, \Phi, q_0, R)$ be an *sk*-rtg. We may assume that \mathcal{G} is clean. By Lemma 4.2(1), there is a regular tree grammar \mathcal{G}' over some ranked alphabet Σ such that $L(\mathcal{G})$ is k -bounded, and there is a relabeling $\tau : \Sigma \rightarrow \mathcal{P}(U)$ such that $L(\mathcal{G}) = \tau(L(\mathcal{G}'))$. Since \mathcal{G} is clean, $\tau(\sigma) \neq \emptyset$ for every $\sigma \in \Sigma$ (see the proof of that lemma).

Then we transform \mathcal{G}' into an equivalent regular tree grammar \mathcal{G}'' which is reduced and is in normal form using the transformations in [CDG⁺97, Prop. 2.1.3, 2.1.4]. Note that $L(\mathcal{G}'')$ is k -bounded and $L(\mathcal{G}) = \tau(L(\mathcal{G}''))$.

Finally, we follow the proof of Lemma 4.2(2) to construct an *sk*-rtg $\bar{\mathcal{G}}$ from \mathcal{G}'' and τ such that $L(\bar{\mathcal{G}}) = \tau(L(\mathcal{G}''))$. Then $\bar{\mathcal{G}}$ is clean due to the above condition on τ . Moreover, a direct inspection of that construction shows that $\bar{\mathcal{G}}$ is reduced and is in normal form. ■

5 Symbolic tree transducers

In this section we formalize our adaptation of the concept of a symbolic tree transducer from [VB11a, VB11b]. Then we show some basic properties, relate symbolic tree transducers to classical top-down tree transducers [Tha70, Rou70, Eng75], and compare our model with the original one. Finally, we prove a composition result for symbolic tree transducers.

5.1 Definition of stt

For every finite set Q and $l \in \mathbb{N}$, we let $Q(X_l) = \{q(x_i) \mid q \in Q, x_i \in X_l\}$.

We denote by $\mathcal{F}(U \rightarrow V)$ the set of all unary computable functions from U to V . Moreover, for every tree $u \in T_{\mathcal{F}(U \rightarrow V)}(Y)$ and $a \in U$, we denote by $u(a)$ the tree which is obtained by replacing every function f in u by the value $f(a) \in V$. Hence we have that $u(a) \in T_V(Y)$.

Definition 5.1 Let $k \in \mathbb{N}$. A *symbolic k -bounded tree transducer* (sk -tt) is a tuple $\mathcal{M} = (Q, U, \Phi, V, q_0, R)$, where

- Q is a finite set (states),
- (U, Φ) is a label structure (input label structure) and V is a set (output labels),
- $q_0 \in Q$ (initial state), and
- R is a finite set of rules of the form $q(\varphi(x_1, \dots, x_l)) \rightarrow u$ where $q \in Q$, $\varphi \in \text{BC}(\Phi)$, $0 \leq l \leq k$, and $u \in T_{\mathcal{F}(U \rightarrow V)}^{(k)}(Q(X_l))$.

Clearly, every sk -tt is an $s(k+1)$ -tt. By an stt we mean an sk -tt for some $k \in \mathbb{N}$.

For a rule $\rho = q(\varphi(x_1, \dots, x_l)) \rightarrow u$, we call the pair (q, l) the *left-hand side state-rank pair*, φ the *guard*, and u the *right-hand side* of ρ , and denote them by $\text{lhs}(\rho)$, $\text{grd}(\rho)$, and $\text{rhs}(\rho)$, respectively.

We say that the stt \mathcal{M} is *linear* (resp. *nondeleting*) if, for each rule ρ as above, its right-hand side contains at most (resp. at least) one occurrence of x_i for every $1 \leq i \leq l$.

Moreover, \mathcal{M} is *deterministic* if, for any two different rules ρ_1 and ρ_2 in R , the condition $\text{lhs}(\rho_1) = \text{lhs}(\rho_2)$ entails that $\llbracket \text{grd}(\rho_1) \rrbracket \cap \llbracket \text{grd}(\rho_2) \rrbracket = \emptyset$. Finally, \mathcal{M} is *total* if for every $q \in Q$ and $0 \leq l \leq k$, we have

$$\llbracket \bigvee_{\substack{\rho \in R \\ \text{lhs}(\rho) = (q, l)}} \text{grd}(\rho) \rrbracket = U.$$

We note that, as for sta, no sk -tt is a total $s(k+1)$ -tt.

Next we define the semantics of an sk -tt $\mathcal{M} = (Q, U, \Phi, V, q_0, R)$. We define the *derivation relation* of \mathcal{M} , denoted by $\Rightarrow_{\mathcal{M}}$, to be the smallest binary relation $\Rightarrow_{\mathcal{M}} \subseteq T_V(Q(T_U)) \times T_V(Q(T_U))$ such that for every $\xi_1, \xi_2 \in T_V(Q(T_U))$:

$\xi_1 \Rightarrow_{\mathcal{M}} \xi_2$ iff there is a position $w \in \text{pos}(\xi_1)$ and a rule $q(\varphi(x_1, \dots, x_l)) \rightarrow u$ in R , such that

- $\xi_1|_w = q(a(\zeta_1, \dots, \zeta_l))$ for some $a \in \llbracket \varphi \rrbracket$ and $\zeta_1, \dots, \zeta_l \in T_U^{(k)}$, and
- $\xi_2 = \xi_1[u']_w$, where u' is obtained from $u(a)$ by replacing every index $p(x_i) \in Q(X_l)$ by $p(\zeta_i)$.

The conditions that all predicates in Φ are recursive and that all functions in the right-hand side of the rules are computable make the relation $\Rightarrow_{\mathcal{M}}$ recursive. Sometimes,

we drop \mathcal{M} from $\Rightarrow_{\mathcal{M}}$.

Let $q \in Q$, $\xi \in T_U$, and $\zeta \in T_V$. We can show by induction on ξ that if $\xi \in T_U^{(k)}$ and $q(\xi) \Rightarrow_{\mathcal{M}}^* \zeta$ holds, then also $\zeta \in T_V^{(k)}$. The q -tree transformation computed by \mathcal{M} , denoted by \mathcal{M}_q , is the relation

$$\mathcal{M}_q = \{(\xi, \zeta) \in T_U^{(k)} \times T_V^{(k)} \mid q(\xi) \Rightarrow_{\mathcal{M}}^* \zeta\} .$$

The *tree transformation* computed by \mathcal{M} , also denoted by \mathcal{M} , is defined by $\mathcal{M} = \mathcal{M}_{q_0}$. The class of tree transformations computed by *sk-tt* (resp. linear, nondeleting, deterministic, and total, *sk-tt*) is denoted by $\text{STT}^{(k)}$ (resp. $1\text{-STT}^{(k)}$, $n\text{-STT}^{(k)}$, $d\text{-STT}^{(k)}$, and $t\text{-STT}^{(k)}$). These restrictions can be combined in the usual way, for instance, we will denote by $\text{ln-STT}^{(k)}$ the class of tree transformations computed by linear and nondeleting *sk-tt*.

A deterministic *sk-tt* (total *sk-tt*) transforms every input tree into at most one (at least one) output tree.

Lemma 5.2 If \mathcal{M} is a deterministic (resp. total) *sk-tt*, then we have $|\mathcal{M}_q(\xi)| \leq 1$ (resp. $|\mathcal{M}_q(\xi)| \geq 1$) for every $q \in Q$ and $\xi \in T_U^{(k)}$.

Example 5.3 We consider the s2-tt $\mathcal{M} = (Q, U, \Phi, U, q, R)$ with $Q = \{q\}$, $U = \mathbb{N}$, and $\Phi = \{\text{div}(2), \text{div}(3)\}$ with $\llbracket \text{div}(i) \rrbracket$ is the set of all non-negative integers which are divisible by i . Moreover, R has the following rules:

$$\begin{array}{lll} \rho_1 : & q(\llbracket \text{div}(2) \wedge \text{div}(3) \rrbracket(x_1, x_2)) & \rightarrow \quad \llbracket :6 \rrbracket(q(x_1), q(x_1)) \\ \rho_2 : & q(\top(x_1, x_2)) & \rightarrow \quad \llbracket \text{id} \rrbracket(q(x_1), q(x_2)) \\ \rho_3 : & q(\top) & \rightarrow \quad \llbracket \text{id} \rrbracket \end{array}$$

where the unary functions $\llbracket :6 \rrbracket$ and id perform division by 6 and the identity, respectively. Note that \mathcal{M} is not deterministic, because $\text{lhs}(\rho_1) = \text{lhs}(\rho_2) = (q, 2)$ and

$$\llbracket \text{grd}(\rho_1) \rrbracket \cap \llbracket \text{grd}(\rho_1) \rrbracket = \llbracket \text{div}(2) \wedge \text{div}(3) \rrbracket \cap \llbracket \top \rrbracket = \llbracket \text{div}(2) \wedge \text{div}(3) \rrbracket \neq \emptyset .$$

Also note that \mathcal{M} is not total, because for $l = 1$ we have:

$$\llbracket \bigvee_{\substack{\rho \in R \\ \text{lhs}(\rho) = (q, 1)}} \text{grd}(\rho) \rrbracket = \llbracket \bigvee_{\rho \in \emptyset} \text{grd}(\rho) \rrbracket = \llbracket \perp \rrbracket = \emptyset \neq U .$$

Also \mathcal{M} is neither linear nor nondeleting, because of rule ρ_1 .

On the input tree $\xi = 6(12(4, 6), 7)$ the s2-tt \mathcal{M} can perform the following derivation:

$$\begin{array}{l} q(6(12(4, 6), 7)) \\ \Rightarrow \quad 1(q(12(4, 6)), q(12(4, 6))) \\ \Rightarrow^2 \quad 1(2(q(4), q(4)), 12(q(4), q(6))) \\ \Rightarrow^4 \quad 1(2(4, 4), 12(4, 6)) \end{array}$$

The s2-tt \mathcal{M} transforms a binary tree ξ in the following way. At each position w , \mathcal{M} can reproduce the label $\xi(w)$ of this position and recursively transforms the subtrees (using rules ρ_2 and ρ_3). If $\xi(w)$ is divisible by 6, then, additionally (using rule (ρ_1)), \mathcal{M} can divide it by 6, delete the second subtree, and process two copies of the first subtree independently.

Next we show that stt generalize (classical) top-down tree transducers. For every $b \in V$, we denote by c_b the constant function in $\mathcal{F}(U \rightarrow V)$ defined by $c_b(a) = b$ for every $a \in U$. An sk -tt $\mathcal{M} = (Q, U, \Phi, V, q_0, R)$ is *alphabetic* if

- U and V are ranked alphabets such that $\text{maxrk}(U), \text{maxrk}(V) \leq k$,
- $\Phi = \{\varphi^\sigma \mid \sigma \in U\}$ where $\llbracket \varphi^\sigma \rrbracket = \{\sigma\}$,
- each rule in R has the form $q(\varphi^\sigma(x_1, \dots, x_l)) \rightarrow u$, where
 - $l = \text{rk}_U(\sigma)$, and
 - for every $w \in (\text{pos}(u) \setminus \text{pos}_{Q(X_l)}(u))$ we have $u(w) = c_b$ and $\text{rk}_u(w) = \text{rk}_V(b)$ for some $b \in V$.

We call predicates of the form φ^σ *alphabetic*.

Let $\mathcal{M} = (Q, \Sigma, \Phi, \Delta, q_0, R)$ be an alphabetic sk -tt with rank mappings rk_Σ and rk_Δ . Let $\mathcal{N} = (Q, \Sigma, \Delta, q_0, R')$ be a top-down tree transducer with the same rank mappings. Then we say that \mathcal{M} and \mathcal{N} are *related* if

$$q(\varphi^\sigma(x_1, \dots, x_l)) \rightarrow u \in R \text{ iff } q(\sigma(x_1, \dots, x_l)) \rightarrow u' \in R' ,$$

where we obtain u' from u by replacing c_δ by δ for every $\delta \in \Delta$.

For every alphabetic sk -tt \mathcal{M} we can construct a related top-down tree transducer \mathcal{N} and vice versa. Moreover, it is easy to see that if \mathcal{M} and \mathcal{N} are related, then the tree transformations computed by \mathcal{M} and by \mathcal{N} are the same. Hence we obtain the following result.

Observation 5.4 The class of tree transformations computed by alphabetic stt is the same as the class of top-down tree transformations.

Recall that ι_U is the identity mapping on U . Let $\mathcal{A} = (Q, U, \Phi, F, R)$ be an sk -ta. We introduce the sk -tt $\mathcal{A}_= = (Q, U, \Phi, U, F, R_=)$, where

$$R_= = \{q(\varphi(x_1, \dots, x_l)) \rightarrow \iota_U(q_1(x_1), \dots, q_l(x_l)) \mid (q_1 \dots q_l, \varphi, q) \in R\}.$$

We will need the following fact.

Lemma 5.5 For every sk -recognizable tree language L , there is a linear and nondeleting sk -tt \mathcal{N} such that $\mathcal{N} = \iota_L$.

PROOF Let $L = L(\mathcal{A})$ for some sk -ta \mathcal{A} . Then $\mathcal{N} = \mathcal{A}_=$ is appropriate. ■

Finally, we want to compare our model with the original one from [VB11b]. Each rule of their symbolic tree transducer has either of the following two forms:

(a) $q(\varepsilon) \rightarrow e$ or

(b) $q(f(x, y_1, \dots, y_k)) \xrightarrow{\varphi[x]} u[x, q_1(y_1), \dots, q_k(y_k)]$

where ε is the only nullary constructor for trees (more precisely, for the empty tree) and f is the only non-nullary constructor for trees. Since in our approach we have neither the empty tree nor the constructor ε , there are no rules in our definition of symbolic tree transducers which correspond to rules of type (a). Also the constructor ε does not occur in the right-hand side of rules of type (b). Then, in our approach, a rule of type (b) looks as follows:

$$q(\varphi(y_1, \dots, y_k)) \rightarrow \psi(u)$$

where the transformation ψ is defined inductively on its argument as follows:

- $\psi(f(p, u_1, \dots, u_l)) = (\lambda x.p)(\psi(u_1), \dots, \psi(u_l))$, and
- $\psi(q_i(y_i)) = q_i(y_i)$.

That is, ψ applies the constructor f , replaces an expression p (in which the variable x occurs) by the unary function $\lambda x.p$, and recursively calls itself on the subterms u_1, \dots, u_l .

5.2 Composition results concerning stt

In [VB11b], among others, composition properties of symbolic tree transformations are considered. Their main result is Theorem 1 which, in its first statement, says that tree transformations computed by stt are closed under composition. For this they give the following proof: “The first statement can be shown along the lines of the proof of compositionality of TOP [15, Theorem 3.39].” where “[15]” is [FV98] in the current paper. Unfortunately, the mentioned proof of [FV98] is not applicable, because there the authors only consider total and deterministic top-down tree transducers.

Moreover, also on the semantics level there is a deficiency. In Section 4.1 they claim the following:

- (†) For two arbitrary stt \mathcal{M} and \mathcal{N} , the composition algorithm delivers an stt which computes the composition $\mathcal{M} \circ \mathcal{N}$.

However, this is not true, which can be seen as follows. Let us apply their composition algorithm to two alphabetic *sk*-tt \mathcal{M} and \mathcal{N} , then the resulting *sk*-tt is also alphabetic (by Observation 5.7) and, due to their claim, it computes $\mathcal{M} \circ \mathcal{N}$. Since alphabetic *sk*-tt correspond to top-down tree transducers it means that the class of all top-down tree transformations is closed under composition. However, it is not, due to the counter examples given in [Rou70, p 267.] (cf. also [Tha70, Eng75]).

So, the proof of the first statement is insufficient. We even conjecture that this statement is wrong, i.e., $\text{STT}^{(k)}$ is not closed under composition.

In this section we prove a weaker version of claim (†) which only holds for particular stt \mathcal{M} and \mathcal{N} , cf. Theorem 5.8. In fact, we generalize the composition theorem [Bak79, Thm. 1] for top-down tree transducers to symbolic tree transducers.

For this, we use the composition algorithm of [VB11b] which results in the syntactic composition $\mathcal{M};\mathcal{N}$, and we show that in certain cases the stt $\mathcal{M};\mathcal{N}$ computes the relation $\mathcal{M} \circ \mathcal{N}$. In the following, we recall the composition algorithm of [VB11b] in our formal setting. We note that this composition algorithm generalizes the (syntactic) composition of top-down tree transducers as presented in the definition before Theorem 1 of [Bak79].

Let $f \in \mathcal{F}(U \rightarrow V)$ and $v \in T_{\mathcal{F}(V \rightarrow W)}(Y)$. We denote by $f \circ v$ the tree obtained from v by replacing every occurrence of a function $g \in \mathcal{F}(V \rightarrow W)$ by the function $f \circ g \in \mathcal{F}(U \rightarrow W)$. Of course $f \circ v \in T_{\mathcal{F}(U \rightarrow W)}(Y)$.

In the following it will be useful to show the occurrences of objects of the form $q(x_i)$ in the right-hand side of rules of an stt explicitly. Therefore sometimes we write an arbitrary element of $T_{\mathcal{F}(U \rightarrow V)}^{(k)}(Q(X_l))$ in the form $u[q_1(x_{i_1}), \dots, q_m(x_{i_m})]$, where $m \geq 0$, $u \in C_{\mathcal{F}(U \rightarrow V)}^{(k)}(X_m)$, $q_1, \dots, q_m \in Q$, and $1 \leq i_1, \dots, i_m \leq l$.

We define the syntactic composition $\mathcal{M};\mathcal{N}$ of two *sk-tt* \mathcal{M} and \mathcal{N} by applying \mathcal{N} to the right-hand side of rules of \mathcal{M} . However, we can do it only symbolically because such a right-hand side is built up from functions and not from labels. In fact, we define a symbolic version of the derivation relation $\Rightarrow_{\mathcal{N}}$, denoted by $\Rightarrow_{\mathcal{N}}^s$ which processes trees over functions. Besides, the rewrite relation $\Rightarrow_{\mathcal{N}}^s$ also deals with objects of the form $q(x_i)$ in its input trees. Moreover, we have to collect the Boolean combinations which are encountered during the transformation of a right-hand side.

Formally, let $\mathcal{M} = (Q, U, \Phi_1, V, q_0, R_1)$ and $\mathcal{N} = (P, V, \Phi_2, W, p_0, R_2)$ be two *sk-tt* and $\Phi = \Phi_1 \cup \{f \circ \psi \mid f \in \mathcal{F}(U \rightarrow V), \psi \in \Phi_2\}$. First, we define the binary relation $\Rightarrow_{\mathcal{N}}^s$ over the set

$$\text{BC}(\Phi) \times T_{\Sigma} \left(P(T_{\Delta}(Q(X_l))) \cup (P \times Q)(X_l) \right)$$

where $\Sigma = \mathcal{F}(U \rightarrow W)$ and $\Delta = \mathcal{F}(U \rightarrow V)$ (cf. Fig. 2). For every

$$(\theta, t), (\theta', t') \in \text{BC}(\Phi) \times T_{\Sigma} \left(P(T_{\Delta}(Q(X_l))) \cup (P \times Q)(X_l) \right)$$

we have

$(\theta, t) \Rightarrow_{\mathcal{N}}^s (\theta', t')$ iff one of the following two conditions hold:

- (i) there is a position $w \in \text{pos}(t)$ such that
 - $t|_w = p(q(x_i))$ for some $p \in P$, $q \in Q$, and $x_i \in X_l$,
 - $t' = t[\langle p, q \rangle(x_i)]_w$, and
 - $\theta' = \theta$, or
- (ii) there is a position $w \in \text{pos}(t)$ and a rule $p(\psi(x_1, \dots, x_l)) \rightarrow v$ in R_2 such that
 - $t|_w = p(f(t_1, \dots, t_l))$ for some $p \in P$, $f \in \mathcal{F}(U \rightarrow V)$, and $t_1, \dots, t_l \in T_{\mathcal{F}(U \rightarrow V)}(Q(X_l))$,
 - $t' = t[v']_w$ where v' is obtained from $f \circ v$ by replacing every $\bar{p}(x_i) \in Q(X_l)$ by $\bar{p}(t_i)$, and

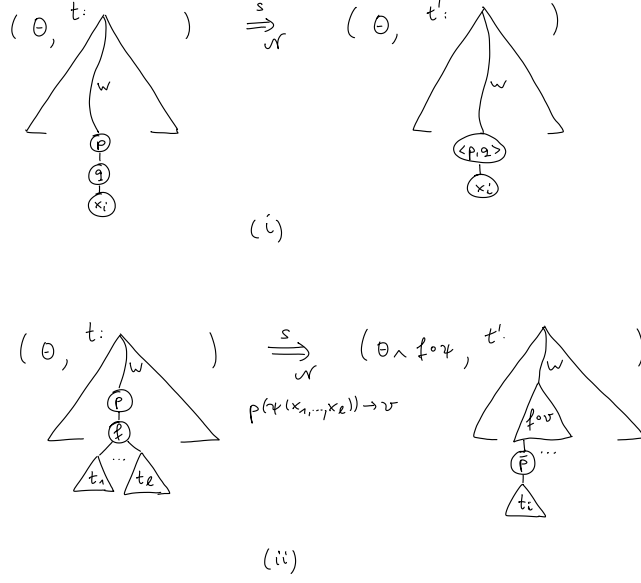


Figure 2: The derivation relation $\Rightarrow_{\mathcal{N}}^s$.

- $\theta' = \theta \wedge f \circ \psi$.

The following statement follows from the definition of the relation $\Rightarrow_{\mathcal{N}}^s$.

Lemma 5.6 (Lift lemma.) If $(\varphi, t) (\Rightarrow_{\mathcal{N}}^s)^* (\theta, t')$, then, for every $a \in \llbracket \theta \rrbracket$ we have $a \in \llbracket \varphi \rrbracket$ and $t(a) \Rightarrow_{\mathcal{N}'}^* t'(a)$, where $\Rightarrow_{\mathcal{N}'}$ is the extension of $\Rightarrow_{\mathcal{N}}$ to the set

$$T_W \left(P(T_V(Q(X_l))) \cup (P \times Q)(X_l) \right),$$

which we obtain by adding the rules $p(q(x_i)) \rightarrow \langle p, q \rangle(x_i)$ to R_2 for every $1 \leq i \leq k$ (cf. p.195 of [Bak79]).

Second, we construct the sk -tt $\mathcal{M}; \mathcal{N} = (P \times Q, U, \Phi, W, \langle p_0, q_0 \rangle, R)$, called the *syntactic composition of \mathcal{M} and \mathcal{N}* , where the set R of rules is defined as follows. If

$$q(\varphi(x_1, \dots, x_l)) \rightarrow u[q_1(x_{i_1}), \dots, q_m(x_{i_m})] \quad (1)$$

is a rule in R_1 , and for some $p \in P$ and $v \in C_{\mathcal{F}(U \rightarrow W)}^{(k)}(X_n)$ we have

$$(\varphi, p(u[q_1(x_{i_1}), \dots, q_m(x_{i_m})])) (\Rightarrow_{\mathcal{N}}^s)^* (\theta, v[\langle p_1, q_{j_1} \rangle(x_{i_{j_1}}), \dots, \langle p_n, q_{j_n} \rangle(x_{i_{j_n}})]) \quad (2)$$

and $\llbracket \theta \rrbracket \neq \emptyset$

then let the rule

$$\langle p, q \rangle(\theta(x_1, \dots, x_l)) \rightarrow v[\langle p_1, q_{j_1} \rangle(x_{i_{j_1}}), \dots, \langle p_n, q_{j_n} \rangle(x_{i_{j_n}})] \quad (3)$$

be in R . Note that

$$\{i_{j_1}, \dots, i_{j_n}\} \subseteq \{i_1, \dots, i_m\} \subseteq \{1, \dots, l\}.$$

We also note that syntactic composition preserves the properties linear, nondeleting, total, and deterministic. For instance, if both \mathcal{M} and \mathcal{N} are linear, then $\mathcal{M};\mathcal{N}$ is also linear.

Observation 5.7 The syntactic composition of two alphabetic sk -tt is an alphabetic sk -tt.

PROOF Let us assume that \mathcal{M} and \mathcal{N} are alphabetic. Then φ in (2) is an alphabetic predicate. Moreover, we observe that if in (i) of the definition of $\Rightarrow_{\mathcal{N}}^s$, the predicate θ is alphabetic, then also θ' is alphabetic; moreover, if in (ii) of this definition φ and ψ are alphabetic and f is a constant function, then either $\llbracket \theta' \rrbracket = \emptyset$ or $\theta' = \theta$. Therefore, θ in (3) is alphabetic. Moreover, by direct inspection of (ii) of the definition of $\Rightarrow_{\mathcal{N}}^s$ we can see that v in (2) consists of constant functions over W . ■

Now we are able to prove our main composition result, which is in fact the generalization of [Bak79, Thm. 1].

Theorem 5.8 *Let \mathcal{M} and \mathcal{N} be sk -tt for which the following two conditions hold:*

- (a) *\mathcal{M} is deterministic or \mathcal{N} is linear, and*
- (b) *\mathcal{M} is total or \mathcal{N} is nondeleting.*

Then the sk -tt $\mathcal{M};\mathcal{N}$ induces $\mathcal{M} \circ \mathcal{N}$.

PROOF We prove that, for every $\xi \in T_U^{(k)}$, $p \in P$, $q \in Q$, and $\zeta \in T_W^{(k)}$, we have

$$\langle p, q \rangle(\xi) \Rightarrow_{\mathcal{M};\mathcal{N}}^* \zeta \quad (4)$$

if and only if

$$\text{there exists an } \eta \in T_V^{(k)} \text{ such that } q(\xi) \Rightarrow_{\mathcal{M}}^* \eta \text{ and } p(\eta) \Rightarrow_{\mathcal{N}}^* \zeta. \quad (5)$$

The proof can be performed by induction on ξ . The proof of the implication (5) \Rightarrow (4) is straightforward, hence we leave it. We note that (as in [Bak79, Thm.1]) we need neither condition (a) nor (b) for the proof of this direction.

To prove that (4) \Rightarrow (5), let us assume that (4) holds and that $\xi = a(\xi_1, \dots, \xi_l)$ for some $a \in U$, $0 \leq l \leq k$, and $\xi_1, \dots, \xi_l \in T_U^{(k)}$.

Let us assume that we applied the rule (3) in the first step of (4). Then (4) can be written as

$$\begin{aligned} \langle p, q \rangle(a(\xi_1, \dots, \xi_l)) &\Rightarrow_{\mathcal{M};\mathcal{N}} v(a)[\langle p_1, q_{j_1} \rangle(\xi_{i_{j_1}}), \dots, \langle p_n, q_{j_n} \rangle(\xi_{i_{j_n}})] \\ &\Rightarrow_{\mathcal{M};\mathcal{N}}^* v(a)[\zeta_1, \dots, \zeta_n] \end{aligned}$$

for some $\zeta_1, \dots, \zeta_n \in T_W^{(k)}$, where $a \in \llbracket \theta \rrbracket$ and $\zeta = v(a)[\zeta_1, \dots, \zeta_n]$. Hence,

$$\langle p_1, q_{j_1} \rangle(\xi_{i_{j_1}}) \Rightarrow_{\mathcal{M}; \mathcal{N}}^* \zeta_1, \dots, \langle p_n, q_{j_n} \rangle(\xi_{i_{j_n}}) \Rightarrow_{\mathcal{M}; \mathcal{N}}^* \zeta_n,$$

and thus, by the induction hypothesis, there are $\eta_1, \dots, \eta_n \in T_V^{(k)}$ such that

$$q_{j_1}(\xi_{i_{j_1}}) \Rightarrow_{\mathcal{M}}^* \eta_1 \text{ and } p_1(\eta_1) \Rightarrow_{\mathcal{N}}^* \zeta_1, \dots, q_{j_n}(\xi_{i_{j_n}}) \Rightarrow_{\mathcal{M}}^* \eta_n \text{ and } p_n(\eta_n) \Rightarrow_{\mathcal{N}}^* \zeta_n. \quad (6)$$

Since rule (3) is in R , there is a rule of then form (1) in R_1 such that the derivation (2) holds. Hence, by Lemma 5.6, $a \in \llbracket \varphi \rrbracket$ and

$$p(u(a)[q_1(x_{i_1}), \dots, q_m(x_{i_m})]) (\Rightarrow_{\mathcal{N}}^s)^* v(a)[\langle p_1, q_{j_1} \rangle(x_{i_{j_1}}), \dots, \langle p_n, q_{j_n} \rangle(x_{i_{j_n}})]]. \quad (7)$$

Now we define the tree η . For this, let $1 \leq \lambda \leq m$. If $\lambda = j_\alpha$ for some $1 \leq \alpha \leq n$, then we define $\bar{\eta}_\lambda = \eta_\alpha$. This $\bar{\eta}_\lambda$ is well-defined, which can be seen as follows. Assume that $j_\alpha = j_\beta$ for some $1 \leq \beta \neq \alpha \leq n$. Then \mathcal{N} is not linear, and thus by condition (a) \mathcal{M} is deterministic, which implies $\eta_\alpha = \eta_\beta$. Note that by (6)

$$q_\lambda(\xi_{i_\lambda}) = q_{j_\alpha}(\xi_{i_{j_\alpha}}) \Rightarrow_{\mathcal{M}}^* \eta_\alpha = \bar{\eta}_\lambda.$$

If there is no α with $\lambda = j_\alpha$, then \mathcal{N} is deleting and thus by condition (b) \mathcal{M} is total. Hence, there is a tree $\bar{\eta}_\lambda \in T_V^{(k)}$ such that $q_\lambda(\xi_{i_\lambda}) \Rightarrow_{\mathcal{M}}^* \bar{\eta}_\lambda$.

Let $\eta = u(a)[\bar{\eta}_1, \dots, \bar{\eta}_m]$. Since the rule (1) is in R_1 and $a \in \llbracket \varphi \rrbracket$, we have

$$q(a(\xi_1, \dots, \xi_l)) \Rightarrow_{\mathcal{M}}^* u(a)[q_1(\xi_{i_1}), \dots, q_m(\xi_{i_m})] \Rightarrow_{\mathcal{M}}^* u(a)[\bar{\eta}_1, \dots, \bar{\eta}_m].$$

Moreover by an obvious modification of (7) and by (6)

$$\begin{aligned} p(u(a)[\bar{\eta}_1, \dots, \bar{\eta}_m]) &\Rightarrow_{\mathcal{N}}^* v(a)[p_1(\bar{\eta}_{j_1}), \dots, p_1(\bar{\eta}_{j_n})] = \\ v(a)[p_1(\eta_1), \dots, p_1(\eta_n)] &\Rightarrow_{\mathcal{N}}^* v(a)[\zeta_1, \dots, \zeta_n]. \end{aligned} \quad \blacksquare$$

Due to Observation 5.7 this theorem generalizes [Bak79, Thm.1].

As an application of the above theorem, we can show that both the class of tree transformations computed by total and deterministic sk -tt and the one computed by linear and nondeleting sk -tt are closed under composition.

Corollary 5.9 (a) $\text{td-STT}^{(k)} \circ \text{td-STT}^{(k)} = \text{td-STT}^{(k)}$
(b) $\text{ln-STT}^{(k)} \circ \text{ln-STT}^{(k)} = \text{ln-STT}^{(k)}.$

PROOF We prove only (a) because the proof of (b) is similar. The inclusion from left to right can be seen as follows. Let \mathcal{M} and \mathcal{N} be total and deterministic sk -tt. The sk -tt $\mathcal{M}; \mathcal{N}$ is also total and deterministic and, by Theorem 5.8, for the computed tree transformations $\mathcal{M}; \mathcal{N} = \mathcal{M} \circ \mathcal{N}$ holds. The other inclusion follows from the facts that (i) any tree transformation $\tau \subseteq T_U^{(k)} \times T_V^{(k)}$ can be decomposed as $\tau \circ \iota_{T_V^{(k)}}$ and (ii) $\iota_{T_V^{(k)}}$ can be computed by a total and deterministic sk -tt. \blacksquare

6 Forward and backward application of stt

In this section we consider forward and backward application of stt to s-recognizable tree languages. In particular, we consider the domain and the range of tree transformations computed by stt. Finally, we apply these results to the problem of (inverse) type checking.

6.1 Application of stt

We begin with the following result.

Theorem 6.1 $\text{dom}(\text{STT}^{(k)}) = \text{REC}^{(k)}$.

PROOF First we prove the inclusion from left to right. For this, let $\mathcal{M} = (Q, U, \Phi, V, q_0, R)$ be an sk -tt. We construct the sk -rtg $\mathcal{G} = (\mathcal{P}(Q), U, \Phi, \{q_0\}, R')$ such that $\text{dom}(\mathcal{M}) = L(\mathcal{G})$, where the set R' of rules is defined as follows.

For every $0 \leq l \leq k$ and $P \subseteq Q$ with $P = \{p_1, \dots, p_m\}$ for some $m \geq 1$, and rules

$$p_1(\varphi_1(x_1, \dots, x_l)) \rightarrow u_1, \dots, p_m(\varphi_m(x_1, \dots, x_l)) \rightarrow u_m \quad (8)$$

in R , let R' contain the rule

$$P \rightarrow (\varphi_1 \wedge \dots \wedge \varphi_m)(P_1, \dots, P_l)$$

where $P_i = \{q \in Q \mid q(x_i) \text{ occurs in } u_j \text{ for some } 1 \leq j \leq m\}$. Thus, in particular, for every $0 \leq l \leq k$, the rule

$$\emptyset \rightarrow \top(\emptyset, \dots, \emptyset)$$

with l occurrences of \emptyset in its right-hand side is in R' . Hence $\emptyset \Rightarrow_{\mathcal{G}}^* \xi$ for every $\xi \in T_U^{(k)}$.

We claim that for every $P \subseteq Q$ and $\xi \in T_U$ we have:

$$P \Rightarrow_{\mathcal{G}}^* \xi \text{ iff } \left(\text{for every } p \in P \text{ there is a } \zeta \in T_V \text{ such that } p(\xi) \Rightarrow_{\mathcal{M}}^* \zeta \right). \quad (9)$$

The statement is clear for $P = \emptyset$, therefore we assume that $P = \{p_1, \dots, p_m\}$ for some $m \geq 1$. We prove (9) by induction on ξ .

Let $\xi = a(\xi_1, \dots, \xi_l) \in T_U^{(k)}$. Then

- $P \Rightarrow_{\mathcal{G}} a(P_1, \dots, P_l) \Rightarrow_{\mathcal{G}}^* a(\xi_1, \dots, \xi_l)$
- iff there is a rule $P \rightarrow (\varphi_1 \wedge \dots \wedge \varphi_m)(P_1, \dots, P_l)$ in R' with $a \in (\bigcap_{j=1}^m \llbracket \varphi_j \rrbracket)$ and for every $1 \leq i \leq l$, $P_i \Rightarrow_{\mathcal{G}}^* \xi_i$
 - iff there are rules (8) in R with $a \in (\bigcap_{j=1}^m \llbracket \varphi_j \rrbracket)$ and for every $1 \leq i \leq l$ and $q \in P_i$ there is a tree $\zeta_{i,q} \in T_V$ s.t. $q(\xi_i) \Rightarrow_{\mathcal{M}}^* \zeta_{i,q}$
 - iff for every $1 \leq j \leq m$ there is a rule $p_j(\varphi_j(x_1, \dots, x_l)) \rightarrow u_j$ s.t. $a \in \llbracket \varphi_j \rrbracket$ and for every occurrence of $q(x_i)$ in $u_j \exists$ a tree $\zeta_{i,q} \in T_V$ s.t. $q(\xi_i) \Rightarrow_{\mathcal{M}}^* \zeta_{i,q}$
 - iff for every $1 \leq j \leq m$ there is a tree $\zeta_j \in T_V$ such that $p_j(a(\xi_1, \dots, \xi_l)) \Rightarrow_{\mathcal{M}}^* \zeta_j$.

Statement (9) with $P = \{q_0\}$ implies $L(\mathcal{G}) = \text{dom}(\mathcal{M})$. Hence, by Theorem 4.4 we obtain that $\text{dom}(\mathcal{M})$ is *sk*-recognizable. The other inclusion follows from Lemma 5.5. ■

Example 6.2 We illustrate the construction of the *sk*-rtg \mathcal{G} in the proof of Theorem 6.1 by an example.

Let the s2-tt \mathcal{M} contain the rules

$$\begin{array}{lll}
 q_0(\varphi(x_1, x_2)) & \rightarrow & f(p(x_1), q(x_1)) & \bar{p}(\theta_1) & \rightarrow & h_1 \\
 p(\psi(x_1)) & \rightarrow & g(\bar{p}(x_1), p'(x_1)) & p'(\theta_2) & \rightarrow & h_2 \\
 q(\psi'(x_1)) & \rightarrow & \hat{p}(x_1) & \hat{p}(\theta_3) & \rightarrow & h_3
 \end{array}$$

Then s2-rtg \mathcal{G} contains (among others) the following rules:

$$\begin{array}{lll}
 \{q_0\} & \rightarrow & \varphi(\{p, q\}, \emptyset) & \emptyset & \rightarrow & \top \\
 \{p, q\} & \rightarrow & (\psi \wedge \psi')(\{\bar{p}, p', \hat{p}\}) & \emptyset & \rightarrow & \top(\emptyset) \\
 \{\bar{p}, p', \hat{p}\} & \rightarrow & (\theta_1 \wedge \theta_2 \wedge \theta_3) & \emptyset & \rightarrow & \top(\emptyset, \emptyset).
 \end{array}$$

Now we can prove that backward application of stt preserve recognizability of tree languages.

Theorem 6.3 $(\text{STT}^{(k)})^{-1}(\text{REC}^{(k)}) = \text{REC}^{(k)}$.

PROOF First we prove the inclusion from left to right. For this, let $\mathcal{M} = (Q, U, \Phi, V, q_0, R)$ be an *sk*-tt, and $L \subseteq T_V^{(k)}$ an *sk*-recognizable tree language. It is an elementary fact that $\mathcal{M}^{-1}(L) = \text{dom}(\mathcal{M} \circ \iota_L)$. By Lemma 5.5, there is a linear and nondeleting *sk*-tt \mathcal{N} with $\mathcal{N} = \iota_L$. Moreover, by Theorem 5.8, the *sk*-tt $\mathcal{M}; \mathcal{N}$ induces $\mathcal{M} \circ \mathcal{N}$. Hence $\mathcal{M}^{-1}(L) = \text{dom}(\mathcal{M}; \mathcal{N})$, which is *sk*-recognizable by Theorem 6.1.

The other inclusion follows from Lemma 5.5. ■

It is well-known from the theory of classical tree automata and tree transducers that the forward application of linear top-down tree transformations preserve recognizability of tree languages (see e.g. [Tha69] or [GS84, Ch. IV, Cor. 6.6]). In particular,

the range of every linear top-down tree transformation is a recognizable tree language. We can show easily that a linear *sk*-tt does not have the analogous property.

Lemma 6.4 There is a linear s1-tt \mathcal{M} such that $\text{range}(\mathcal{M})$ is not 1-recognizable.

PROOF Let us assume that U is infinite and define the s1-tt $\mathcal{M} = (\{q\}, U, \{\top\}, U, q, R)$, where R consists of the only rule

$$q(\top()) \rightarrow \iota_U(\iota_U).$$

It is clear that \mathcal{M} induces the 1-tree transformation $\{(a, a(a)) \mid a \in U\}$. Thus $\text{range}(\mathcal{M}) = \{a(a) \mid a \in U\}$, which is not 1-recognizable by the remark after Lemma 3.6. \blacksquare

The non-recognizability of $\text{range}(\mathcal{M})$ above is due to the fact that \mathcal{M} is able to “duplicate” a node of the input tree by having two occurrences of an appropriate function symbol on the right-hand side of its rule. We would like to identify a restricted version of an stt which does not have this capability in the hope of that such an stt preserves recognizability. Therefore we define simple stt as follows. An *sk*-tt $\mathcal{M} = (Q, U, \Phi, V, q_0, R)$ is *simple* if $\text{rhs}(\rho)$ contains exactly one function symbol for every rule $\rho \in R$. We denote the class of tree transformations computed by simple and linear stt by sl-STT. Then we can prove the desired result using the following notation. If $\varphi \in \text{Pred}(U)$ and $f : U \rightarrow V$ is a mapping, then $f(\varphi)$ denotes the predicate defined by $\llbracket f(\varphi) \rrbracket = f(\llbracket \varphi \rrbracket)$.

Theorem 6.5 $\text{sl-STT}^{(k)}(\text{REC}^{(k)}) = \text{REC}^{(k)}$.

PROOF First we prove the inclusion from left to right. Let $\mathcal{M} = (Q, U, \Phi, V, q_0, R)$ be a simple and linear *sk*-tt and L be an s-recognizable tree language such that $L = L(\mathcal{G})$ for some reduced *sk*-rtg $\mathcal{G} = (P, U, \Psi, p_0, R_{\mathcal{G}})$ which is in normal form (cf. Theorem 4.4 and Lemma 4.5).

We construct the *sk*-rtg $\mathcal{G}' = (Q \times P, V, \Psi', \langle q_0, p_0 \rangle, R')$, where

- $\Psi' = \{f(\varphi \wedge \psi) \mid \varphi \text{ and } f \text{ occur in a rule of } R, \text{ and } \psi \text{ in a rule of } R_{\mathcal{G}}\}$, and
- R' is the smallest set of rules satisfying that if $p \rightarrow \psi(p_1, \dots, p_l)$ is in $R_{\mathcal{G}}$ and $q(\varphi(x_1, \dots, x_l)) \rightarrow f(q_1(x_{i_1}), \dots, q_m(x_{i_m}))$ is in R , then the rule

$$\langle q, p \rangle \rightarrow f(\varphi \wedge \psi)(\langle q_1, p_{i_1} \rangle, \dots, \langle q_m, p_{i_m} \rangle) \quad (10)$$

is in R' .

We show that $L(\mathcal{G}') = \mathcal{M}(L)$. For this it suffices to prove the following statement. For every $q \in Q$, $p \in P$, and $\zeta \in T_V$ we have

$$\langle q, p \rangle \Rightarrow_{\mathcal{G}'}^* \zeta \iff \exists(\xi \in L(\mathcal{G}, p)) \text{ such that } q(\xi) \Rightarrow_{\mathcal{M}}^* \zeta.$$

We prove only the direction \Rightarrow by induction on the number n of steps of the corresponding derivation and we show only the induction step n to $n + 1$. The other direction can be proved in a similar way.

Direction \Rightarrow , step $n \rightarrow n+1$: We assume that in the first step of the derivation we applied the rule (10) obtained from the rules $p \rightarrow \psi(p_1, \dots, p_l)$ in R_G and $q(\varphi(x_1, \dots, x_l)) \rightarrow f(q_1(x_{i_1}), \dots, q_m(x_{i_m}))$ in R . (Note that $\{i_1, \dots, i_m\} \subseteq \{1, \dots, l\}$.) Then we have

$$\langle q, p \rangle \Rightarrow_{G'} b(\langle q_1, p_{i_1} \rangle, \dots, \langle q_m, p_{i_m} \rangle) \Rightarrow_{G'}^n b(\zeta_1, \dots, \zeta_m)$$

for some $b \in \llbracket f(\varphi \wedge \psi) \rrbracket$ and $\zeta_1, \dots, \zeta_m \in T_V^{(k)}$. By the I.H., there are trees $\xi_{i_j} \in L(\mathcal{G}, p_{i_j})$ such that $q_{i_j}(\xi_{i_j}) \Rightarrow_{\mathcal{M}}^* \zeta_j$ for every $1 \leq j \leq m$. Moreover, there is a $a \in (\llbracket \varphi \rrbracket \cap \llbracket \psi \rrbracket)$ such that $b = f(a)$. Now define the tree $\xi = a(\bar{\xi}_1, \dots, \bar{\xi}_l) \in T_U^{(k)}$, where $\bar{\xi}_j = \xi_{i_l}$ if $j = i_l$ for some $1 \leq l \leq m$; and let $\bar{\xi}_j$ be an arbitrary tree in $L(\mathcal{G}, p_j)$ otherwise (note that \mathcal{G} is reduced). Then

$$p \Rightarrow_G a(p_1, \dots, p_l) \Rightarrow_G^* a(\bar{\xi}_1, \dots, \bar{\xi}_l),$$

hence $\xi \in L(\mathcal{G}, p)$. Moreover

$$q(a(\bar{\xi}_1, \dots, \bar{\xi}_l)) \Rightarrow_{\mathcal{M}} b(\langle q_1, p_{i_1} \rangle(\xi_{i_1}), \dots, \langle q_m, p_{i_m} \rangle(\xi_{i_m})) \Rightarrow_{\mathcal{M}}^* b(\zeta_1, \dots, \zeta_m).$$

The inclusion from right to left follows from Lemma 5.5 and the fact that $\mathcal{A}_=$ is a simple and linear stt. \blacksquare

Corollary 6.6 $\text{range}(\text{sl-STT}^{(k)}) = \text{REC}^{(k)}$.

PROOF Let $\mathcal{M} = (Q, U, \Phi, V, q_0, R)$ be a simple and linear sk -tt. Obviously, $\text{range}(\mathcal{M}) = \mathcal{M}(T_U^{(k)})$. Moreover, by Observation 3.3, $T_U^{(k)}$ is sk -recognizable. Hence the statement follows from Theorem 6.5. \blacksquare

6.2 Type checking with stt

Intuitively, type checking means to verify whether or not all documents in a view have a certain type. According to [EM03], a typical scenario of type checking is that τ translates XML documents into HTML documents. Thus, for a set L of XML documents $\tau(L)$ is an HTML-view of the documents in L . In practice, we are interested in particular XML documents, which turn to be a recognizable tree language of unranked trees over some alphabet. Also, certain desired properties of the so-obtained HTML documents can be described in terms of recognizability of tree languages. Thus, the type checking problem of τ in fact means to check whether $\tau(L) \subseteq L'$ for recognizable tree languages L and L' . The inverse type checking problem can be described in a similar way. The type checking and the inverse type checking problem for different kinds of transducers was considered in several works, see among others [MSV03, AMN⁺03, EM03, MBPS05]. For stt we obtain the following results.

Theorem 6.7

(a) *The inverse type checking problem for stt is decidable.*

(b) *The type checking problem for simple and linear stt is decidable.*

PROOF Both statements follow from the fact that the inclusion problem of s-recognizable tree languages is decidable. This latter fact can be seen as follows. By [VB11a, Thm. 3], s-recognizable tree languages are effectively closed under Boolean operations, for closure under complement, see our correction at the end of Section 3.1. Moreover, by [VB11a, Thm. 4], the emptiness problem is decidable for s-recognizable tree languages provided that the emptiness problem in the underlying label structure is decidable. Since, by our definition, the label structure underlying an *sk*-ta has a decidable emptiness problem, we obtain that the inclusion problem of s-recognizable tree languages is decidable.

Then the proof of (a) is as follows. Let $\mathcal{M} : T_U^{(k)} \rightarrow T_V^{(k)}$ be an *sk*-tt and $L' \subseteq T_U^{(k)}$ and $L \subseteq T_V^{(k)}$ s-recognizable tree languages. By Theorem 6.3, the tree language $\mathcal{M}^{-1}(L)$ is effectively *sk*-recognizable, thus we can decide if $\mathcal{M}^{-1}(L) \subseteq L'$ holds or not. Statement (b) can be proved in a similar way, using Theorem 6.5. ■

7 Conclusion and an open problem

In this paper we have further elaborated the theory of sta and stt. Our main contributions are: the characterization of s-recognizable tree languages in terms of relabelings of recognizable tree languages, the introduction of symbolic regular tree grammars and the proof of their equivalence to sta, the comparison of sta and variable tree automata, the composition of stt, and the forward and backward application of stt to s-recognizable tree languages.

Finally, we mention an open problem. In the definition of simple *sk*-tt we required that the right-hand side of each rule contains exactly one function symbol. We conjecture that, for the closure result in Theorem 6.5, it is sufficient to require that right-hand sides of rules contain at most one function

References

- [AMN⁺03] N. Alon, T. Milo, F. Neven, D. Suciu, and V. Vianu. XML with data values: typechecking revisited. *J. Comput. Syst. Sci.*, 66(4):688–727, 2003.
- [Bak79] B.S. Baker. Composition of top-down and bottom-up tree transductions. *Inform. and Control*, 41(2):186–213, 1979.
- [BCC⁺03] M. Brambilla, S. Ceri, S. Comai, P. Fraternali, and I. Manolescu. Specification and design of workflow-driven hypertexts. *J. Web Eng.*, 2:163–182, 2003.
- [BHJS07] A. Bouajjani, P. Habermehl, Y. Jurski, and M. Sighireanu. Rewriting systems with data. In *Proc. of FCT 2007*, volume 4639 of *Lecture Notes in Comput. Sci.*, pages 1–22. Springer-Verlag, 2007.

- [BHM03] A. Bouajjani, P. Habermehl, and R. Mayr. Automatic verification of recursive procedures with one integer parameter. *Theoret. Comput. Sci.*, 295:85–106, 2003.
- [Bra69] W. S. Brainerd. Tree generating regular systems. *Inform. and Control*, 14:217–231, 1969.
- [CDG⁺97] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: <http://www.grappa.univ-lille3.fr/tata>, 1997.
- [Don70] J. Doner. Tree acceptors and some of their applications. *J. Comput. System Sci.*, 4:406–451, 1970.
- [EM03] J. Engelfriet and S. Maneth. A comparison of pebble tree transducers with macro tree transducers. *Acta Inform.*, 39(9):613–698, 2003.
- [Eng75] J. Engelfriet. Bottom-up and top-down tree transformations - a comparison. *Math. Systems Theory*, 9(3):198–231, 1975.
- [FV98] Z. Fülöp and H. Vogler. *Syntax-directed semantics — Formal Models Based on Tree Transducers*. Monogr. Theoret. Comput. Sci. EATCS Ser. Springer-Verlag, 1998.
- [GKS10] O. Grumberg, O. Kupferman, and S. Sheinvald. Variable automata over infinite alphabets. In C. Martin-Vide A.-H. Dediu, H. Fernau, editor, *LATA 2010*, volume 6031 of *Lecture Notes in Computer Science*, pages 561–572. Springer-Verlag, 2010.
- [GS84] F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.
- [GS97] F. Gécseg and M. Steinby. Tree languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer-Verlag, 1997.
- [MBPS05] S. Maneth, A. Berlea, T. Perst, and H. Seidl. XML type checking with macro tree transducers. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, PODS 2005, Baltimore, Maryland*, pages 283–294. ACM Press, 2005.
- [MR11] I.-E. Mens and G. Rahonis. Variable tree automata over infinite ranked alphabets. In F. Winkler, editor, *CAI 2011*, volume 6742 of *Lecture Notes in Computer Science*, pages 247–260. Springer-Verlag, 2011.
- [MSV03] T. Milo, D. Suciu, and V. Vianu. Typechecking for XML transformers. *J. Comput. System Sci.*, 66:688–727, 2003.
- [Rou70] W.C. Rounds. Mappings and grammars on trees. *Math. Systems Theory*, 4(3):257–287, 1970.
- [Tha69] J.W. Thatcher. Generalized² sequential machine maps. IBM Res. Report RC 2466, 1969.
- [Tha70] J.W. Thatcher. Generalized² sequential machine maps. *J. Comput. System*

- Sci.*, 4(4):339–367, 1970.
- [VB11a] M. Veanes and N. Bjorner. Foundations of Finite Symbolic Tree Transducers. *Bulletin of EATCS*, 105:141–173, 2011.
 - [VB11b] M. Veanes and N. Bjorner. Symbolic tree transducers. In M. Clarke, I. Virbitskaite, and A. Voronkov, editors, *Proc. of Perspectives of System Informatics (PSI' 11)*, volume 7162 of *LNCS*, pages 371–387. Springer-Verlag, 2011.
 - [VHL⁺12] M. Veanes, P. Hooimeijer, B. Livshits, D. Molnar, and N. Bjorner. Symbolic Finite Transducers: Algorithms and Applications. In M. Hicks, editor, *Proc. of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'12)*, pages 137–150. ACM SIGPLAN, 2012.