Decidability of Equivalence for Deterministic Synchronized Tree Automata

Kai Salomaa

Department of Mathematics, University of Turku
FIN-20500 Turku, Finland
E-mail: ksalomaa@sara.cc.utu.fi

Abstract. Synchronized tree automata allow limited communication between computations in independent subtrees of the input. This enables them to verify, for instance, the equality of two unary subtrees of unlimited size. The class of tree languages recognized by synchronized tree automata is strictly included in the context-free tree languages. As our main result we show that equivalence of tree languages recognized by deterministic synchronized tree automata can be effectively decided. This contrasts the earlier undecidability result for the equivalence problem for nondeterministic synchronized tree automata.

1 Introduction

The recognition capability of finite tree automata is restricted by the fact that computations in independent subtrees of the input are not allowed to communicate with each other. The restriction is especially severe for deterministic top-down tree automata which recognize a proper subfamily of the regular tree languages. Because of this handicap, various extensions of the finite tree automaton model have been proposed. Top-down tree automata augmented with different types of look-ahead capabilities are considered in [4,5,6,16,17]. Other natural extensions of the finite tree automaton model are the automata with constraints and encompassment automata studied in [1,2,3].

A top-down synchronized tree automaton allows a simple limited form of communication between independent computations. Some states of the automaton contain synchronizing symbols, these are called synchronizing states. The synchronization condition requires that the sequences of synchronization symbols produced along any two paths of the input are in the prefix relation, i.e, one is a prefix of the other. In the equality-synchronized computation mode we require that the synchronization sequences corresponding to all paths of the input are the same. Intuitively, the synchronization condition can be interpreted by saying that when the automaton enters a synchronizing state, it must stop and wait until all other computations in independent subtrees either terminate successfully or enter a synchronizing state containing the same synchronizing symbol.

The above notion of synchronization was originally introduced for alternating machines by Hromkovič [9]. Synchronization has turned out to be a very useful

notion in the study of parallel computations, see [10,11,12] and the references listed there. Synchronized tree automata were first considered in [14]. In spite of the similarity of the definitions, the notion of synchronization is essentially different, respectively, for alternating machines and for tree automata. In the former case one synchronizes parallel computations on the same input but in the case of tree automata synchronization represents communication between different parts of the input. Alternating tree automata [13,15] combine these two notions of parallelism. It is not clear what would be the right way to define synchronization for alternating tree automata.

The inclusion relations between the families of tree languages defined by deterministic and nondeterministic prefix- and equality-synchronized tree automata were established in [14]. Furthermore, it was shown that all synchronized tree language families are properly included in the context-free tree languages. This implies that emptiness is decidable for synchronized tree languages. On the other hand, equivalence turns out to be undecidable for nondeterministic synchronized tree automata. Contrasting this result it was established in [14] that equivalence of deterministic equality-synchronized automata can be decided effectively. This question was reduced to the equivalence problem for deterministic multitape finite automata which is known to be decidable [8]. An essential part of the proof was the so called normalization property for equality-synchronized automata. A normalized automaton recognizes as its extended tree language exactly the set of prefix-trees of the tree language defined by the automaton. A similar property could not be established for prefix-synchronized computations.

As our main result here we show that equivalence of deterministic prefix-synchronized automata is decidable. As a tool for our proof we consider so called globally deterministic synchronized tree automata. Globally deterministic synchronized alternating machines are considered in [11]. We use slightly weaker conditions to define the notion of global determinism than the conditions of the definition of [11].

Globally deterministic automata can recognize the set of so called two-pruned prefix-trees of a deterministic synchronized tree automaton. The (nontrivial part of the) equivalence problem for deterministic synchronized automata can be reduced to the question of equivalence of the corresponding sets of two-pruned prefix-trees. Although we do not know whether equivalence of globally deterministic tree automata in general is decidable, we can effectively decide the equivalence of the specific automata used in our proof.

The tree language families defined, respectively, by the deterministic prefix-synchronized and equality-synchronized automata are incomparable. However, a prefix-synchronized automaton can simulate arbitrary equality-synchronized computations provided that we allow the use of end-markers at the leaves of the input. This means that the decidability of equivalence for deterministic prefix-synchronized automata gives as an immediate corollary the corresponding decidability result for equality-synchronized automata.

The synchronized tree automata recognize only a small subfamily of the context-free tree languages. However, even a deterministic synchronized automa-

ton can, for instance, determine the equality of unary subtrees of unlimited height. In view of this fact the above decidability results seem to be of some interest. Note that the automata with constraints considered in [1,2,3] can verify much more general properties of the inputs, but for them equivalence is undecidable. Without certain restrictions even the question of emptiness is undecidable for these automata.

2 Preliminaries

Here we fix some notations and briefly recall definitions concerning trees that will be used in the later sections.

The set of positive integers is denoted by $\mathbb{N}$ and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Let A be a set. The power set of A is $\mathcal{P}(A)$. If A is finite we denote its cardinality by $\#A$. The set of finite words over A is denoted by A^* and λ is the empty word. Also, $A^+ = A^* - \{\lambda\}$. For $w \in A^*$ and $L \subseteq A^*$, the quotient of L by w is $w^{-1}L = \{v \in A^* \mid wv \in L\}$. The length of a word w is denoted by $|w|$.

If $w_2 = w_1 u$, $w_1, w_2, u \in A^*$, we say that w_1 is a prefix of w_2 and denote $w_1 \preceq w_2$. The *prefix relation* $\simeq_{\mathrm{pr}} \subseteq A^* \times A^*$ is defined by setting $w_1 \simeq_{\mathrm{pr}} w_2$ if and only if $w_1 \preceq w_2$ or $w_2 \preceq w_1$. Let W be a finite subset of A^* such that $w_1 \simeq_{\mathrm{pr}} w_2$ for all $w_1, w_2 \in W$. Then $\max_{\simeq}(W)$ denotes the unique word $w \in W$ such that $w' \preceq w$ for all $w' \in W$.

A *tree domain* D is a nonempty finite subset of $\mathbb{N}^*$ that satisfies the following two conditions: (i) If $u \preceq v \in D$, then $u \in D$. (ii) For every $u \in D$ there exists $\mathrm{rank}_D(u) \in \mathbb{N}_0$ such that $ui \in D$ for $i = 1, \ldots, \mathrm{rank}_D(u)$ and $ui \notin D$ for $i > \mathrm{rank}_D(u)$.

Let A be a set. An A-labeled tree is a mapping $t : \mathrm{dom}(t) \to A$, where $\mathrm{dom}(t)$ is a tree domain. A node $u \in \mathrm{dom}(t)$ is said to be labeled by $t(u) \in A$. A node v is a *child* of a node u ($u, v \in \mathrm{dom}(t)$) if $v = uj$, $j \in \mathbb{N}$.

We use symbols Σ and Ω to denote finite ranked alphabets. The set of symbols of Σ of rank m, $m \geq 0$, is denoted by Σ_m. Let Y be a set of auxiliary symbols. The set of ΣY-trees (or ΣY-terms), $F_\Sigma(Y)$, is the smallest set such that $\Sigma_0 \cup Y \subseteq F_\Sigma(Y)$ and $\sigma(t_1, \ldots, t_m) \in F_\Sigma(Y)$ for all $m \geq 1$, $\sigma \in \Sigma_m$, $t_1, \ldots, t_m \in F_\Sigma(Y)$. By choosing above $Y = \emptyset$ we obtain the definition of the set of Σ-trees F_Σ.

In the natural way, a given ΣY-tree can be viewed as a $(\Sigma \cup Y)$-labeled tree $t : \mathrm{dom}(t) \to \Sigma \cup Y$ that satisfies the condition that every node of rank m is labeled by an m-ary symbol. In the following, we use interchangeably the above algebraic definition of a tree and the notion of a labeled tree defined using a tree domain.

We denote by $X = \{x_1, x_2, \ldots\}$ a fixed countably infinite set of variables. We assume that notions such as the height, the root, a leaf and a subtree of a ΣX-tree t are known. The set of leaves of t is denoted $\mathrm{leaf}(t) \subseteq \mathrm{dom}(t)$. The set of leaves of t labeled by elements of Σ_0 is $\mathrm{leaf}_\Sigma(t)$ and $\mathrm{leaf}_X(t)$ is the set of leaves labeled by variables of X. The set of variables appearing in a tree $t \in F_\Sigma(X)$ is $\mathrm{var}(t)$ and we say that t is *linear* (in variables X) if t has only one occurrence

of any variable of X. The set of linear ΣX-trees is denoted $\mathrm{lin}(\Sigma, X)$. For our purposes the names of the variables of a ΣX-tree are irrelevant and we identify trees that are obtained from each other by renaming variables.

Let $t, t_1, \ldots, t_m \in F_\Sigma(X)$ and $x_1, \ldots, x_m \in \mathrm{var}(t)$. Then $t(x_1 \leftarrow t_1, \ldots, x_m \leftarrow t_m)$ denotes the ΣX-tree obtained from t by replacing each occurrence of the variable x_i with t_i, $i = 1, \ldots, m$.

Let $t \in F_\Sigma$. We define the set of *prefix-trees* of t, $\mathrm{pref}(t)$, to consist of all $r \in \mathrm{lin}(\Sigma, X)$ with $\mathrm{var}(r) = \{x_1, \ldots, x_m\}$, $m \geq 0$, such that there exist $r_1, \ldots, r_m \in F_\Sigma$ such that $r(x_1 \leftarrow r_1, \ldots, x_m \leftarrow r_m) = t$. The set $\mathrm{pref}(t)$ consists of all linear ΣX-trees that are obtained from t by replacing a set of independent subtrees by distinct variables.

To conclude this section we define notations concerning paths in trees that will be used when considering synchronization conditions for tree automata. Let t be an A-labeled tree for some set A. A *path* of t from the root to a node $u_m \in \mathrm{dom}(t)$ is a word

$$t(u_1) \cdots t(u_m) \in A^+, \tag{1}$$

where $u_1 = \lambda$, and u_{i+1} is a child of u_i, $i = 1, \ldots, m-1$. A path (1) is denoted $\mathrm{path}(t, u_m)$. The set of paths of the tree t consists of all paths from the root of t to a leaf and we denote

$$\mathrm{path}(t) = \{\mathrm{path}(t, u) \mid u \in \mathrm{leaf}(t)\}.$$

The *domain* of $\mathrm{path}(t, u_m)$ as in (1) is defined by $\mathrm{PATH}(t, u_m) = \{u_1, \ldots, u_m\}$.

Let $t \in F_\Sigma(X)$ and $u \in \mathrm{leaf}(t)$. The set of variable nodes corresponding to the path $\mathrm{path}(t, u)$, $\mathrm{varnd\text{-}path}(t, u)$, consists of all nodes $v \in \mathrm{leaf}_X(t)$ such that v is a child of a node of $\mathrm{PATH}(t, u)$ and $v \notin \mathrm{PATH}(t, u)$. Thus $\mathrm{varnd\text{-}path}(t, u)$ consists of all nodes "branching out" from the path that are labeled by variables.

Let $t \in F_\Sigma$ and u_1, u_2 be two distinct leaves of t. Denote $P(u_1, u_2) = \mathrm{PATH}(t, u_1) \cup \mathrm{PATH}(t, u_2)$. The *two-pruned tree* $2\mathrm{pr}(t, u_1, u_2)$ is the linear ΣX-tree r determined by the below conditions (i) and (ii).

(i) $\mathrm{dom}(r)$ is the subset of $\mathrm{dom}(t)$ containing the set $P(u_1, u_2)$ and all children of the nodes of $P(u_1, u_2)$.

(ii) If $u \in P(u_1, u_2)$, then $r(u) = t(u)$. If u is a child of a node of $P(u_1, u_2)$ not belonging to $P(u_1, u_2)$, then u is labeled by a variable.

Intuitively, $2\mathrm{pr}(t, u_1, u_2)$ is obtained from t by cutting off all subtrees branching out from the paths leading to u_1 and u_2 and replacing each subtree by a distinct variable. The tree $2\mathrm{pr}(t, u_1, u_2)$ is defined uniquely by the above conditions because we consider trees obtained from each other by a renaming of variables to be identical. Note that, in general, the paths from the root to the leaves u_1 and u_2 may contain a common prefix.

The set of two-pruned trees corresponding to $t \in F_\Sigma$ is $2\mathrm{pr}(t) = \{2\mathrm{pr}(t, u_1, u_2) \mid u_1, u_2 \in \mathrm{leaf}(t),\ u_1 \neq u_2\}$, and,

$$2\mathrm{pr}(\Sigma) = \bigcup_{t \in F_\Sigma} 2\mathrm{pr}(t). \tag{2}$$

3 Synchronized Tree Automata

We recall the definition of synchronized tree automata from [14]. Roughly speaking, they are top-down finite tree automata where the computations corresponding to independent subtrees of the input can communicate by way of so called synchronizing symbols. Here we give only the definitions that are needed for the decidability results in the last section. For more details and examples of synchronized tree automata see [14].

A *nondeterministic top-down tree automaton, nta,* is a four-tuple $\mathcal{A} = (\Sigma, Q, Q_0, g)$, where Σ is a ranked alphabet of input symbols, Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, and g determines the state transitions by associating to each $\sigma \in \Sigma_m$, $m \geq 0$, a mapping $\sigma_g : Q \to \mathcal{P}(Q^m)$. The automaton $\mathcal{A}$ is *deterministic, dta,* if $Q_0 = \{q_0\}$ is a singleton set and for all $q \in Q$, $\sigma \in \Sigma_m$, $m \geq 1$, $\#\sigma_g(q) \leq 1$. The class of nondeterministic (respectively, deterministic) top-down tree automata is denoted *nt* (respectively, *dt.*)

We make the notational convention that m-tuples $q_1, \ldots, q_m$ belonging to $\sigma_g(q)$, $\sigma \in \Sigma_m$, $m \geq 1$, $q \in Q$, are denoted using square brackets: $[q_1, \ldots, q_m] \in \sigma_g(q)$. This is done for easier readability because synchronized automata will have states with several components. By a computation step we mean a pair (q, σ), $q \in Q$, $\sigma \in \Sigma$. A computation step (q, σ) is said to be *deterministic* if $\#\sigma_g(q) \leq 1$, and otherwise it is a *nondeterministic computation step.*

Definition 3.1 *Let $\mathcal{A} = (\Sigma, Q, Q_0, g) \in nt$ and $t \in F_\Sigma$. A* (successful) computation *of $\mathcal{A}$ on the input t is a Q-labeled tree $r : dom(t) \to Q$ satisfying the following three conditions:*

(i) $r(\lambda) \in Q_0$.
(ii) *Let $u \in dom(t)$, $t(u) = \sigma \in \Sigma_m$, $m \geq 1$, and let $u_1, \ldots, u_m$ be the children of the node u. Then $[r(u_1), \ldots, r(u_m)] \in \sigma_g(r(u))$.*
(iii) *If $u \in leaf(t)$ and $t(u) = \sigma \in \Sigma_0$, then $r(u) \in \sigma_g$.*

The set of computations of $\mathcal{A}$ on a tree t is denoted $\mathrm{com}_{\mathcal{A}}(t)$. If $\mathcal{A}$ is deterministic, then $\#\mathrm{com}_{\mathcal{A}}(t) \leq 1$ for every $t \in F_\Sigma$. The tree language *recognized* by $\mathcal{A}$ is $L(\mathcal{A}) = \{t \in F_\Sigma \mid \mathrm{com}_{\mathcal{A}}(t) \neq \emptyset\}$. We denote the tree language families recognized by nondeterministic and deterministic (top-down) tree automata, respectively, NT and DT. It is well known that DT is strictly included in NT [7].

Definition 3.2 *[14] A* nondeterministic synchronized tree automaton, nsta, *is a top-down tree automaton $\mathcal{A} = (\Sigma, Q, Q_0, g)$ where the state set is of the form*

$$Q = Q_1 \cup (Q_2 \times S). \tag{3}$$

The set S is the synchronization alphabet *and elements of S are called* synchronizing symbols, *(sync-symbols for short). States belonging to $Q_2 \times S$ are said to be the* synchronizing states *of the automaton. When referring to an nsta $\mathcal{A}$, unless otherwise mentioned, we always assume that the state set of $\mathcal{A}$ is as in (3) and S denotes the synchronization alphabet.*

The above automaton $\mathcal{A}$ is a deterministic synchronized tree automaton, dsta, *if it is a deterministic top-down tree automaton. We denote the class of nondeterministic (respectively, deterministic) synchronized tree automata by* nst *(respectively,* dst*).*

We define a morphism $h_{\mathcal{A}} : Q^* \to S^*$ by setting $h_{\mathcal{A}}(q_1) = \lambda$ and $h_{\mathcal{A}}((q_2, s)) = s$ for all $q_i \in Q_i$, $i = 1, 2$, $s \in S$. Let $t \in F_\Sigma$. The set of synchronized computations *(or* prefix-synchronized computations*) of $\mathcal{A}$ on $t \in F_\Sigma$ is*

$$scom_{\mathcal{A}}(t) = \{r \in com_{\mathcal{A}}(t) \mid (\forall u, v \in path(r))\ h_{\mathcal{A}}(u) \simeq_{pr} h_{\mathcal{A}}(v)\}. \qquad (4)$$

The set of equality-synchronized *computations of $\mathcal{A}$ on the tree t is defined as*

$$s_e com_{\mathcal{A}}(t) = \{r \in com_{\mathcal{A}}(t) \mid (\forall u, v \in path(r))\ h_{\mathcal{A}}(u) = h_{\mathcal{A}}(v)\}.$$

The tree language (prefix-)synchronized recognized *by $\mathcal{A}$ is defined as*

$$L_s(\mathcal{A}) = \{t \in F_\Sigma \mid scom_{\mathcal{A}}(t) \neq \emptyset\},$$

and the tree language equality-synchronized recognized *by $\mathcal{A}$ is*

$$L_e(\mathcal{A}) = \{t \in F_\Sigma \mid s_e com_{\mathcal{A}}(t) \neq \emptyset\}.$$

The families of tree languages synchronized recognized by nondeterministic and deterministic automata are denoted, respectively, NST and DST. The corresponding families defined by the equality-synchronized computation mode are N_eST and D_eST. Here we are mainly concerned with the prefix-synchronized computation mode and we call prefix-synchronized computations simply synchronized computations.

We will need the following result concerning string languages defined as synchronizing sequences of computations of tree automata. Let $\mathcal{A} = (\Sigma, Q, Q_0, g) \in nst$, $t \in F_\Sigma$ and $r \in \text{scom}_{\mathcal{A}}(t)$ be a synchronized computation on t. The (maximal) *synchronizing sequence* of the computation r is $\text{seq}(r) = \max_{\simeq}(h_{\mathcal{A}}(\text{path}(r))$. The right side exists by the definition of synchronized computations. The set of synchronizing sequences corresponding to the input t is $\text{seq}(\mathcal{A}, t) = \{\text{seq}(r) \mid r \in \text{scom}_{\mathcal{A}}(t)\}$ and the *synchronization language* of $\mathcal{A}$ is $\text{sync}(\mathcal{A}) = \bigcup_{t \in F_\Sigma} \text{seq}(\mathcal{A}, t)$.

Theorem 3.1 *[14] The language sync($\mathcal{A}$) is a regular word language for every $\mathcal{A} \in nst$.*

For our decidability results it turns out to be useful to consider automata that locally allow nondeterministic choices in the computation, but where the global computation on any given input is deterministic via the synchronization condition. We define these automata similarly as the globally deterministic synchronized alternating machines of [11].

Definition 3.3 *Let $\mathcal{A} = (\Sigma, Q, Q_0, g) \in nst$, $t \in F_\Sigma$ and let $r \in scom_{\mathcal{A}}(t)$ be a synchronized computation of $\mathcal{A}$ on t. For each $i \in \{1, \ldots, |seq(r)|\}$, the ith* synchronization cut *of r is the set $scut(r, i)$ consisting of all nodes $u \in dom(r)$ such that u is labeled by a synchronizing state of $\mathcal{A}$ and $|h_{\mathcal{A}}(path(r, u))| = i$.*

The set $\mathrm{scut}(r,i)$ consists of exactly all nodes where the computation reaches the ith synchronizing state in different branches of the input. From the definition of synchronized computations it follows that every node of a given synchronization cut $\mathrm{scut}(r,i)$ contains the same sync-symbol.

Definition 3.4 *An automaton $\mathcal{A} = (\Sigma, Q, Q_0, g) \in nst$, $Q = Q_1 \cup (Q_2 \times S)$, is* globally deterministic, gdsta, *if it has a unique initial state $Q_0 = \{q_0\}$ and the following conditions hold.*

(i) *For every nondeterministic computation step (q,σ), $q \in Q$, $\sigma \in \Sigma_m$, $m \geq 1$, the set $\sigma_g(q)$ consists of m-tuples of synchronizing states, $\sigma_g(q) = \{[(q_1^i, s_i), \ldots, (q_m^i, s_i)] \mid i = 1, \ldots, k,\ k \geq 2\}$, and $s_i \neq s_j$ when $i \neq j$, $1 \leq i, j \leq k$.*

(ii) *Let $t \in F_\Sigma$ and $r \in scom_{\mathcal{A}}(t)$. Let $scut(r,i)$ be a synchronization cut of r and let s be the sync-symbol appearing in the nodes of $scut(r,i)$. Then s is the only sync-symbol that can be produced by* all *the nondeterministic computation steps at the parent nodes of the synchronization cut $scut(r,i)$.*

The condition (i) requires that in all nondeterministic computation steps of $\mathcal{A}$ the different nondeterministic decisions must be connected with a choice of different sync-symbols. Thus all computation steps producing nonsynchronizing states are deterministic. Condition (ii) guarantees that, for every synchronization cut among all the nondeterministic choices, there is only one that does not immediately violate the synchronization condition. This means that the global computation is in fact deterministic. Note that (ii) above is weaker than the corresponding condition in the definition of globally deterministic synchronized computations in [11]. There it is required that every synchronization cut of a successful computation contains a node where the corresponding sync-symbol is enforced deterministically. Our condition (ii) requires only that the sync-symbol is enforced deterministically as the only common choice for all the nondeterministic steps in question. The automaton we use in the construction for the decidability proof will have only this weaker property.

Combining the above observations it is easy to prove the following lemma. An analogous result for synchronized alternating machines appears in [11].

Lemma 3.1 *Let $\mathcal{A} = (\Sigma, Q, q_0, g)$ be a gdsta and $t \in F_\Sigma$. Then $scom_{\mathcal{A}}(t)$ consists of a unique computation or is empty.*

Condition (ii) of Definition 3.4 requires that a certain property has to be satisfied for all synchronization cuts of all successful computations of $\mathcal{A}$. Given an automaton $\mathcal{A}$, it is not immediately obvious whether $\mathcal{A}$ satisfies the condition (ii). It can be shown that we can effectively decide whether a given nst $\mathcal{A}$ is globally deterministic. In any case, for the globally deterministic automata that we use in the constructions of the next section the conditions of Definition 3.4 can be immediately verified.

The class of globally deterministic automata as given in Definition 3.4 is denoted *gdst* and the corresponding tree language family is denoted GDST. The

above definition of a gdsta $\mathcal{A}$ uses explicitly the prefix-synchronized computation mode. Similarly, by considering equality-synchronized computations of the automaton we can define *globally deterministic equality-synchronized tree automata, gd$_e$sta's.* Again, the corresponding family of tree languages is denoted GD$_e$ST.

Note that if a given nsta $\mathcal{A}$ is globally deterministic in the sense of Definition 3.4 (a gdsta), then $\mathcal{A}$ is always a globally deterministic equality-synchronized automaton, gd$_e$sta. This follows from the observation that for every tree t, $s_e com_{\mathcal{A}}(t) \subseteq scom_{\mathcal{A}}(t)$. However, it is easy to see (using similar examples as in [14]) that GDST is not a subfamily of GD$_e$ST. This is because of course usually $L_e(\mathcal{A}) \neq L_s(\mathcal{A})$.

The inclusions between the tree language families defined by the various classes of synchronized tree automata are depicted in Figure 1. In the figure a line indicates strict inclusion and unconnected classes are incomparable. CFT denotes the family of context-free tree languages and REC is the family of recognizable tree languages. All relations of the figure not involving the families defined by globally deterministic automata are proved in [14]. We leave the remaining relations as an exercise.

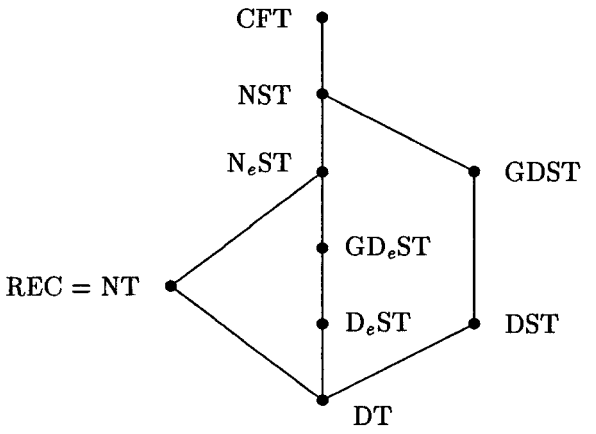

Fig. 1

4 Decidability of Equivalence

The equivalence problem for nondeterministic prefix- and equality-synchronized tree automata is undecidable [14]. Here we will prove that equivalence is decidable for DST. It was established in [14] that equivalence is decidable for D$_e$ST. This result follows now as a corollary of the decidability result for DST since a prefix-synchronized automaton can simulate arbitrary equality-synchronized computations provided that the input trees are augmented with end-markers on each path before the leaf node [14].

A central tool for proving the decidability of equivalence for D_eST was the so called normalization condition. A synchronized tree automaton $\mathcal{A}$ is said to be normalized if any synchronized partial computation ending at leaves labeled by variables can be completed successfully, assuming that one substitutes suitably chosen Σ-trees for the variables. Intuitively, we can say that a normalized automaton does not produce synchronizing sequences that do not appear in any successful computation. Given a dst $\mathcal{A}$ we can effectively construct a normalized dst $\mathcal{B}$ such that $L_e(\mathcal{B}) = L_e(\mathcal{A})$ [14], that is, equality-synchronized automata can be assumed to be normalized. A normalized automaton $\mathcal{A}$ recognizes as its extended tree language exactly the set of prefix-trees of the tree language defined by $\mathcal{A}$. Using this observation we can reduce the equivalence problem for deterministic normalized automata to deciding equivalence of deterministic multitape finite automata. However, an analogous normalization result does not hold for prefix-synchronized tree automata, see [14].

Here we will show that the set of two-pruned prefix-trees of a tree language defined by a prefix-synchronized deterministic automaton can be recognized using a *globally deterministic* synchronized tree automaton. This "quasi-normalization" condition turns out to be sufficient for carrying out the rest of the decidability proof. The crucial observation is that the automaton operating on prefix-trees does not need to be deterministic, the weaker requirement of global determinism suffices.

We view a computation of $\mathcal{A} = (\Sigma, Q, Q_0, g) \in nst$ on a tree having variables as an intermediate stage of a computation that is to be continued from the nodes labeled by variables. The set of synchronized computations of $\mathcal{A} \in$ nst on $t \in F_\Sigma(X)$, $\mathrm{scom}_\mathcal{A}(t)$, is defined to consist of all labeled trees $r : \mathrm{dom}(t) \to Q$ that satisfy the three conditions of Definition 3.1 and the synchronization condition given by (4). Thus, at leaves of t labeled by variables, the computations of $\mathrm{scom}_\mathcal{A}(t)$ can end in an arbitrary state and it is only required that the synchronization condition has not been violated up to that point.

The *extended tree language synchronized recognized* by $\mathcal{A}$ is denoted

$$M_s(\mathcal{A}) = \{t \in F_\Sigma(X) \cap \mathrm{lin}(\Sigma, X) \mid \mathrm{scom}_\mathcal{A}(t) \neq \emptyset\}.$$

We restrict the ΣX-trees that $\mathcal{A}$ receives as inputs to be linear and we do not distinguish between trees that differ only in the names of the variables. The automaton $\mathcal{A}$ treats every leaf labeled by a variable identically. Clearly, $L_s(\mathcal{A}) = M_s(\mathcal{A}) \cap F_\Sigma$ and $\mathrm{pref}(L_s(\mathcal{A})) \subseteq M_s(\mathcal{A})$. Note that this inclusion is, in general, strict since there is no guarantee that the computations from variables of $t \in M_s(\mathcal{A})$ can be continued without violating the synchronization condition.

Let Σ be a ranked alphabet. We denote by $\mathrm{un}(\Sigma)$ the set of all unary Σ-trees, that is, trees containing only symbols of rank at most one. The decision algorithm for the equivalence problem will be based on the following lemma.

Lemma 4.1 *Let $\mathcal{A}_i = (\Sigma, Q_i, q_{i0}, g_i) \in dst$, $i = 1, 2$. Denote $Y_i = pref(L_s(\mathcal{A}_i)) \cap 2pr(\Sigma)$ (see (2)) and $Z_i = L_s(\mathcal{A}_i) \cap un(\Sigma)$, $i = 1, 2$.*

Then $L_s(\mathcal{A}_1) = L_s(\mathcal{A}_2)$ if and only if $Y_1 = Y_2$ and $Z_1 = Z_2$.

Proof. The proof is similar to the proof of Lemma 5.5 of [14]. □

We want to show that for arbitrary dst's $\mathcal{A}_i$, $i = 1, 2$, we can construct globally deterministic automata that are equivalent if and only if $Y_1 = Y_2$, where $Y_i = \mathrm{pref}(L_s(\mathcal{A}_i)) \cap 2\mathrm{pr}(\Sigma)$. We still need some technical lemmas.

We say that $\mathcal{A} = (\Sigma, Q, q_0, g) \in \mathrm{dst}$ has the *end-marker property* if there exists $\$ \in \Sigma_1$ such that for every $t \in L_s(\mathcal{A})$ the following holds.

(EM1) Every subtree of t of height one is of the form $\$(\sigma)$, $\sigma \in \Sigma_0$, and the symbol $\$ does not appear in t except in subtrees of height one.

(EM2) The unique computation of $\mathrm{scom}_{\mathcal{A}}(t)$ enters each of the leaves of t in a synchronizing state.

If $\mathcal{A}$ has the end-marker property, then $\mathcal{A}$ ends the computation in each branch of the input by producing a synchronizing symbol. The following lemma states that, when considering the equivalence problem for DST, without restriction we can assume that the given automata have the end-marker property.

Lemma 4.2 *Let $\mathcal{A}_i = (\Sigma, Q, q_{i0}, g_i) \in dst$, $i = 1, 2$. We can effectively construct automata $\mathcal{B}_i = (\Omega, P, p_{i0}, h_i) \in dst$ having the end-marker property, $i = 1, 2$, such that $L_s(\mathcal{B}_1) = L_s(\mathcal{B}_2)$ if and only if $L_s(\mathcal{A}_1) = L_s(\mathcal{A}_2)$.*

Proof. Define Ω by setting $\Omega_1 = \Sigma_1 \cup \{\$\}$ and $\Omega_m = \Sigma_m$ when $m \neq 1$. Here $\$ \notin \Sigma$. Let $\alpha : F_\Sigma \to F_\Omega$ be the tree homomorphism that replaces every leaf labeled by $\sigma \in \Sigma_0$ with the subtree $\$(\sigma)$. We construct $\mathcal{B}_i$ such that $L_s(\mathcal{B}_i) = \alpha(L_s(\mathcal{A}_i))$, $i = 1, 2$. The tree language $\alpha(L_s(\mathcal{A}_i))$ clearly satisfies the condition (EM1) so it is sufficient to show that the automaton $\mathcal{B}_i$, $1 \leq i \leq 2$, can be made to satisfy (EM2).

Without restriction we can assume that the initial state q_{i0} of $\mathcal{A}_i$ is a synchronizing state. Thus we can write $q_{i0} = (r_0, s_0)$ where s_0 is a sync-symbol. On an input $t \in F_\Omega$ the automaton $\mathcal{B}_i$ exactly simulates the computation of $\mathcal{A}_i$ except that along each path of t it delays the production of the synchronizing sequence by one step. Thus the initial state of $\mathcal{B}_i$ is not synchronizing and $\mathcal{B}_i$ produces the sync-symbol s_0 when $\mathcal{A}_i$ would produce the next sync-symbol s_1 on a given path. $\mathcal{B}_i$ then stores s_1 in its finite-state memory (the nonsynchronizing component of the states) and produces s_1 when $\mathcal{A}_i$ would produce the following sync-symbol. When $\mathcal{B}_i$ reaches a unary symbol $\$ it produces the previous sync-symbol of the computation of $\mathcal{A}_i$ that it is simulating. Finally, $\mathcal{B}_i$ simulates the computation step of $\mathcal{A}_i$ on the leaf below. The operation of $\mathcal{B}_i$ is completely deterministic. For every $t \in F_\Sigma$, the unique computation of $\mathcal{B}_i$ on $\alpha(t)$ is synchronized if and only if the computation of $\mathcal{A}_i$ on t is synchronized. □

We want to develop a criterion for deciding whether a given two-pruned tree is a prefix-tree of the tree language recognized by a dsta. In the following let $\mathcal{A} = (\Sigma, Q, q_0, g) \in \mathrm{dst}$. For $q \in Q$ we denote

$$\mathcal{A}(q) = (\Sigma, Q, q, g).$$

$\mathcal{A}(q)$ is the dsta obtained from $\mathcal{A}$ by changing the initial state to be q.

Let $t \in M_s(\mathcal{A}) \cap 2\mathrm{pr}(\Sigma)$ and let $r \in \mathrm{scom}_{\mathcal{A}}(t)$ be the unique synchronized computation of $\mathcal{A}$ on t. Let u_1 and u_2 be the two nodes of t labeled by elements of Σ_0 and denote $w_i = h_{\mathcal{A}}(\mathrm{path}(r, u_i))$, $i = 1, 2$. Thus w_i is the synchronization sequence corresponding to the node u_i. Let $v \in \mathrm{varnd\text{-}path}(t, u_i)$, $1 \leq i \leq 2$, that is, v is a node labeled by a variable "corresponding" to the path determined by u_i. Denote $v = v'j$, $j \in \mathbb{N}$. Assume that, in the computation r, $\mathcal{A}$ reaches the node v in state q_v, that is, $r(v) = q_v$. Denote $w_{v'} = h_{\mathcal{A}}(\mathrm{path}(r, v'))$. Then clearly $w_{v'} \preceq w_i$. We define the *residue* of v with respect to u_i in the computation r by setting

$$\mathrm{res}(v, u_i) = w_{v'}^{-1} w_i.$$

Let $w_M = \max_{\simeq}\{w_1, w_2\}$. The word w_M is just the longer one of the two synchronization sequences corresponding to the nodes u_1 and u_2. Clearly, the synchronization sequence corresponding to v' is a prefix of w_M, $w_{v'} \preceq w_M$. We define the *residue* of v (with respect to both u_1 and u_2) as

$$\mathrm{res}(v) = w_{v'}^{-1} w_M.$$

The computation continuing from the variable at node v can be completed successfully (with a suitable choice of the subtree to be substituted there) if and only if $\mathrm{res}(v)$ is in the prefix-relation with some word $w_v \in \mathrm{sync}(\mathcal{A}(q_v))$. The tree t is a prefix-tree of $L_s(\mathcal{A})$ if and only if the words w_v can be chosen so that, additionally, they are all pairwise in the prefix-relation. We note that if we choose $w_v \in \mathrm{sync}(\mathcal{A}(q_v))$ to be a prefix of the word $\mathrm{res}(v)$ then w_v does not affect the possible choices for the computations beginning from the remaining variables of t. We say that a node $v \in \mathrm{leaf}_X(t)$ is *relevant* with respect to the computation r if $\mathrm{sync}(\mathcal{A}(q_v))$ does not contain any prefix of $\mathrm{res}(v)$. Recall that q_v is the state where $\mathcal{A}$ reaches the node v. Combining the above observations, we have proved the following lemma.

Lemma 4.3 *Let $\mathcal{A} = (\Sigma, Q, q_0, g) \in dst$ and $t \in M_s(\mathcal{A}) \cap 2pr(\Sigma)$. Let r be the unique computation in $scom_{\mathcal{A}}(t)$. Denote by REL the set of all relevant variable nodes of t in the computation r. Then $t \in pref(L_s(\mathcal{A}))$ if and only if there exist words*

$$w_v \in res(v)^{-1} sync(\mathcal{A}(r(v))), \quad v \in REL, \tag{5}$$

such that the words w_v are pairwise in the prefix-relation.

Note that if for some relevant variable node v, $\mathrm{res}(v)^{-1}\mathrm{sync}(\mathcal{A}(r(v))) = \emptyset$, then $t \notin \mathrm{pref}(L_s(\mathcal{A}))$, i.e., the computation cannot be continued successfully from t. By Theorem 3.1 the languages $\mathrm{sync}(\mathcal{A}(q))$, $q \in Q$, are regular. Denote

$$\Xi(\mathcal{A}) = \{w^{-1}\mathrm{sync}(\mathcal{A}(q)) \mid w \in S^*,\ q \in Q\}.$$

Here S is the synchronization alphabet of the automaton $\mathcal{A}$. Then all the languages belonging to $\Xi(\mathcal{A})$ are regular and $\Xi(\mathcal{A})$ contains only finitely many different languages.

Let Σ be a ranked alphabet. The *look-ahead alphabet* corresponding to Σ is the ranked alphabet Σ^{LA} defined by setting

$$(\Sigma^{LA})_m = \{\sigma[z_1, \ldots, z_m] \mid \sigma \in \Sigma_m,\ z_1, \ldots, z_m \in \Sigma \cup \{*\}\},$$

$m \geq 0$. We define the mapping $\beta : \Sigma^{LA} \to \Sigma$ by setting $\beta(\sigma[z_1, \ldots, z_m]) = \sigma$, $\sigma \in \Sigma_m$, $m \geq 0$. The relabeling $F_{\Sigma^{LA}}(X) \to F_\Sigma(X)$ induced by β is denoted also simply by β.

A tree $t \in F_{\Sigma^{LA}}(X)$ is said to be *well-formed* if it satisfies the following condition. Let $u \in \mathrm{dom}(t)$ be a node of rank $m \geq 1$. Then $t(u) = \sigma[z_1, \ldots, z_m]$, where for $j = 1, \ldots, m$,

$$z_j = \begin{cases} \beta(t(uj)) & \text{if } t(uj) \in \Sigma^{LA}, \\ * & \text{if } t(uj) \in X. \end{cases}$$

The set of well-formed $\Sigma^{LA}X$-trees is denoted by $F^{\mathrm{wf}}_{\Sigma^{LA}}(X)$. Intuitively, in a well-formed $\Sigma^{LA}X$-tree the label of each node contains the "look-ahead one" information about the labels of the children. Clearly for every $t \in F_\Sigma(X)$ there exists a unique tree $r \in F^{\mathrm{wf}}_{\Sigma^{LA}}(X)$ such that $\beta(r) = t$. We denote by β_{wf} the restriction of β to the set of well-formed trees. Then β_{wf} is a bijection $F^{\mathrm{wf}}_{\Sigma^{LA}}(X) \to F_\Sigma(X)$.

The set of well-formed two-pruned trees $F^{\mathrm{wf}}_{\Sigma^{LA}}(X) \cap 2\mathrm{pr}(\Sigma^{LA})$ can obviously be recognized by a deterministic top-down tree automaton $\mathcal{A}$. Also, the tree language families DT and GDST are closed with respect to intersection with a tree language belonging to DT. Thus in the below Lemmas 4.4 and 4.5, without further mention, we assume that the input belongs to $F^{\mathrm{wf}}_{\Sigma^{LA}}(X) \cap 2\mathrm{pr}(\Sigma^{LA})$ and the automata need not verify this property.

Lemma 4.4 *Let $\mathcal{A} = (\Sigma, Q, q_0, g)$ be a dsta. Then there exists a deterministic top-down tree automaton $\mathcal{B} = (\Sigma^{LA}, P, p_0, h)$ such that for every $t \in F^{\mathrm{wf}}_{\Sigma^{LA}}(X) \cap 2pr(\Sigma^{LA})$ the automaton $\mathcal{B}$ reaches the two nonvariable leaves u of t in a state that contains (as its second component) a set $\Lambda(u) \subseteq \Xi(\mathcal{A})$ defined as follows. Denote $\beta_{\mathrm{wf}}(t) = t'$, $t' \in F_\Sigma(X)$. Since β_{wf} is a relabeling we can identify the nodes of t and t'. Let r be the unique computation of $\mathcal{A}$ on t'.*

Then $\Lambda(u)$ contains all elements

$$res(v, u)^{-1} sync(\mathcal{A}(r(v))),$$

where $v \in varnd\text{-}path(t, u)$ and

$$sync(\mathcal{A}(r(v))) \text{ does not contain any prefix of } res(v, u). \tag{6}$$

Proof. The states of $\mathcal{B}$ contain two components. When reading an input symbol $\sigma \in \Sigma^{LA}$, the first component simulates directly the computation step of $\mathcal{A}$ on $\beta(\sigma)$. The second component is a subset of $\Xi(\mathcal{A})$. When the simulated computation of $\mathcal{A}$ enters a variable node v in a state q, the automaton $\mathcal{B}$ ads the element $\mathrm{sync}(\mathcal{A}(q))$ to the second component of the state in the brother node of v that belongs to the path leading to a nonvariable leaf. Note that since t is well-formed, the look-ahead capability enables $\mathcal{B}$ to know which child nodes are

labeled by variables. On the paths labeled by symbols of Σ^{LA}, always when the simulated computation of $\mathcal{A}$ produces a sync-symbol s, $\mathcal{B}$ updates the elements of the second component by replacing a language L with $s^{-1}L$. If some language L contains the empty word, it means that the negation of the condition (6) holds for the variable nodes whose synchronization language is represented by L. Such languages will be discarded from the subset of $\Xi(\mathcal{A})$ appearing in the second component. It is clear that at the leaves u corresponding to the nonvariable paths of t the second component of $\mathcal{B}$ will contain exactly the set $\Lambda(u)$. □

In the above proof, note that $\Xi(\mathcal{A})$ consists of a finite number of regular languages, and thus each subset of $\Xi(\mathcal{A})$ can be presented using, for instance, a constant number of finite automata. These can be stored in the finite-state memory of $\mathcal{B}$. Also, for each $L \in \Xi(\mathcal{A})$ the operations $s^{-1}L$, where s is a sync-symbol, can be finitely specified.

Lemma 4.5 *Let $\mathcal{A} = (\Sigma, Q, q_0, g) \in dst$. Then there exists a gdsta $\mathcal{C} = (\Sigma^{LA}, P, p_0, h)$ such that*

$$M_s(\mathcal{C}) = \beta_{\mathrm{wf}}^{-1}(pref(L_s(\mathcal{A}))) \cap 2pr(\Sigma^{LA}).$$

Proof. The states of $\mathcal{C}$ have two components. On an input tree t the first component simulates the computation of $\mathcal{A}$ on $\beta_{\mathrm{wf}}(t)$ and verifies that $\beta_{\mathrm{wf}}(t) \in M_s(\mathcal{A})$. Let $\mathcal{B}$ be the dta constructed corresponding to $\mathcal{A}$ as in Lemma 4.4. The second component of $\mathcal{C}$ simulates the computation of $\mathcal{B}$. Since $\mathcal{B}$ is just a finite tree automaton, the two components of $\mathcal{C}$ can operate independently in parallel.

The second component reaches the two nonvariable nodes u_1 and u_2 of t in states containing the subsets of $\Xi(\mathcal{A})$, $\Lambda(u_1)$ and $\Lambda(u_2)$, as defined in Lemma 4.4. Assume that in the computation r of $\mathcal{A}$ on $\beta_{\mathrm{wf}}(t)$ the synchronization sequence corresponding to u_1 is a prefix of the synchronization sequence corresponding to u_2. If we assume that the automaton can transfer the state $\Lambda(u_1)$ to the corresponding position in the computation on the path to u_2, then it can reach the leaf u_2 with a subset of $\Xi(\mathcal{A})$ consisting of all the languages $\mathrm{res}(v)^{-1}\mathrm{sync}(\mathcal{A}(r(v)))$, where v is a relevant node of t in the computation r. This means that the automaton can decide whether the condition (5) of Lemma 4.3 holds and, thus, it can decide whether $t \in \beta_{\mathrm{wf}}^{-1}(\mathrm{pref}(L_s(\mathcal{A})))$. Since the languages of $\Xi(\mathcal{A})$ are regular, given an arbitrary subset $\{\mu_1, \ldots, \mu_k\} \subseteq \Xi(\mathcal{A})$, we can decide effectively whether there exist words $w_i \in \mu_i$, $i = 1, \ldots, k$, that are all pairwise in the prefix relation. The answer for each of the finitely many subsets of $\Xi(\mathcal{A})$ can then be stored in the states of $\mathcal{C}$.

Thus it is sufficient to show that when simulating the synchronized computation of $\mathcal{A}$, the gdsta $\mathcal{C}$ can transfer a finite amount of information from the node u_1 to u_2, where u_1 is the leaf corresponding to the shorter synchronization sequence. By Lemma 4.2, we can assume that $\mathcal{A}$ has the end-marker property. Thus the computation of $\mathcal{A}$ produces a sync-symbol when entering the node u_1.

The automaton $\mathcal{C}$ can take care of the transfer of information by allowing nondeterministic guesses always when $\mathcal{A}$ produces a sync-symbol s. In the left path of t the guesses are defined as follows. If the input symbol is not the end-marker, the possibilities are: (i) s (representing the guess that also the right path

is not yet at the end-marker), (ii) (s, χ, right) for all $\chi \subseteq \Xi(\mathcal{A})$ (representing the guess that the right path is at the end-marker with $\Lambda(u) = \chi$). If the input is the end-marker, the sync-symbol produced will be (s, χ, left) where χ is the corresponding state of the automaton $\mathcal{B}$. In the right path, if the input is not the end-marker the possibilities are (i) s and (ii) (s, χ, left) for all $\chi \subseteq \Xi(\mathcal{A})$. At the end-marker of the right path one has also two possibilities: (i) (s, χ, right) where χ is the corresponding state of $\mathcal{B}$, and (ii) (s, χ, left) for all $\chi \subseteq \Xi(\mathcal{A})$. The last choice corresponds to the guess that the left path is simultaneously at the end-marker. It is easy to verify that in all cases there is exactly one global choice for the two paths that does not violate the synchronization condition. Thus $\mathcal{C}$ is globally deterministic. □

Now we can prove our main result.

Theorem 4.1 *Given $\mathcal{A}_1$, $\mathcal{A}_2 \in dst$ we can effectively decide whether $L_s(\mathcal{A}_1) = L_s(\mathcal{A}_2)$.*

Proof. Without loss of generality we can assume that the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same input alphabet Σ. By Lemma 4.1, to prove the claim it is suffient to show that the condition

$$\text{pref}(L_s(\mathcal{A}_1)) \cap 2\text{pr}(\Sigma) = \text{pref}(L_s(\mathcal{A}_2)) \cap 2\text{pr}(\Sigma) \tag{7}$$

can be decided effectively. The other condition appearing in Lemma 4.1 concerning unary trees is clearly decidable. By Lemma 4.5, there exist globally deterministic automata $\mathcal{C}_i$, $i = 1, 2$, such that

$$M_s(\mathcal{C}_i) = \beta_{\text{wf}}^{-1}(\text{pref}(L_s(\mathcal{A}_i))) \cap 2\text{pr}(\Sigma^{LA}).$$

By coding two-pruned trees into two strings we can simulate the computations of $\mathcal{C}_i$ using a deterministic two-tape finite automaton $\mathcal{D}_i$. This is done similarly as in the proof of Lemma 5.6 of [14]. The only difference is that instead of deterministic synchronized automata we are considering globally deterministic automata. However, this does not change anything because the finite control of $\mathcal{D}_i$ has reading heads on both tapes (representing the paths of a two-pruned tree), and thus corresponding to an arbitrary synchronization cut, from the nondeterministic choices of $\mathcal{C}_i$ on the two paths, $\mathcal{D}_i$ can deterministically make the unique possible global choice. Thus $\mathcal{D}_1$ and $\mathcal{D}_2$ accept the same inputs if and only if $M_s(\mathcal{C}_1) = M_s(\mathcal{C}_2)$.

Since equivalence of deterministic multitape automata is decidable [8], it follows that we can effectively decide whether $M_s(\mathcal{C}_1) = M_s(\mathcal{C}_2)$. Since the mapping β_{wf} is bijective, it follows that also the condition (7) is decidable. □

Using the easy Lemma 3.3 of [14], we see that the above theorem implies also the decidability of equivalence for $D_e ST$. This result was established differently in [14].

Corollary 4.1 *Equivalence is decidable for deterministic equality-synchronized tree automata.*

We have established that for deterministic prefix- and equality-synchronized automata equivalence is decidable and it is undecidable in the nondeterministic cases. The family of tree languages defined by globally deterministic synchronized automata lies strictly between the corresponding nondeterministic and deterministic families. It is an open question whether equivalence of globally deterministic tree automata can be decided effectively.

References

1. B. Bogaert and S. Tison, Equality and disequality constraints on direct subterms in tree automata, in: *Proc. of the 9th Symposium on Theoretical Aspects of Computer Science,* Lect. Notes Comput. Sci. **577**, Springer-Verlag, 1992, pp. 161–171.
2. A.-C. Caron, J.-L. Coquidé and M. Dauchet, Encompassment properties and automata with constraints, in: *Proc. of 5th RTA,* Lect. Notes Comput. Sci. **690**, Springer-Verlag, 1993, pp. 328–342.
3. A.-C. Caron, H. Comon, J.-L. Coquidé, M. Dauchet and F. Jacquemard, Pumping, cleaning and symbolic constraints solving, in: *Proc. of 21st ICALP,* Lect. Notes Comput. Sci. **820**, Springer-Verlag, 1994, pp. 436–449.
4. Z. Fülöp and S. Vágvölgyi, Variants of top-down tree transducers with look-ahead, *Math. Systems Theory* **21** (1989) 125–145.
5. Z. Fülöp and S. Vágvölgyi, Iterated deterministic top-down look-ahead, in: *Proc. of 7th FCT,* Lect. Notes Comput. Sci. **380**, Springer-Verlag, 1989, pp. 175–184.
6. Z. Fülöp and S. Vágvölgyi, A characterization of irreducible sets modulo left-linear term rewriting systems by tree automata, *Fundam. Inf.* **13** (1990) 211-226.
7. F. Gécseg and M. Steinby, *Tree automata,* Akadémiai Kiadó, Budapest, 1984.
8. T. Harju and J. Karhumäki, The equivalence problem of multitape finite automata, *Theoret. Comput. Sci.* **78** (1991) 347–355.
9. J. Hromkovič, How to organize the communication among parallel processes in alternating computations, unpublished manuscript, Comenius University, Bratislava, 1986.
10. J. Hromkovič, J. Karhumäki, B. Rovan and A. Slobodová, On the power of synchronization in parallel computations, *Discrete Appl. Math.* **32** (1991) 155–182.
11. J. Hromkovič, B. Rovan, A. Slobodová, Deterministic versus nondeterministic space in terms of synchronized alternating machines, *Theoret. Comput. Sci.* **132** (1994) 319–336.
12. O. Ibarra and N. Trân, Synchronized finite automata and 2DFA reductions, *Theoret. Comput. Sci.* **115** (1993) 261–275.
13. K. Salomaa, Yield-languages recognized by alternating tree recognizers, *RAIRO Inform. Théor.* **22** (1988) 319–339.
14. K. Salomaa, Synchronized tree automata, *Theoret. Comput. Sci.* **127** (1994) 25–51.
15. G. Slutzki, Alternating tree automata, *Theoret. Comput. Sci.* **41** (1985) 305–318.
16. G. Slutzki and S. Vágvölgyi, A hierarchy of deterministic top-down tree transformations, in: *Proc. of 9th FCT,* Lect. Notes Comput. Sci. **710**, Springer-Verlag, 1993, pp. 440–451.
17. S. Vágvölgyi, Top-down tree transducers with two-way tree walking look-ahead, *Theoret. Comput. Sci.* **93** (1992) 43–74.