Acta Cybernetica, Vol. 10, Nr. 3, Szeged, 1992

Alternation bounds for tree automata

Kai Salomaa *

Abstract

We consider alternation depth bounds for tree automata, that is, we limit the number of alternations of existential and universal computation steps. We show that a constant bound guarantees that the forest recognized is regular, whereas already a logarithmic bound enables the automata to recognize a strictly larger class of forests. As a corollary we obtain results on other types of alternation bounds for tree automata. We consider also commutation properties of independent computation steps of an alternating tree automaton.

1 Introduction

An alternating computation generalizes a nondeterministic one by allowing the automaton to both make existential choices and branch the computation universally. Alternation has been used to model parallelism of various machine models, cf. e.g. [2], [6], [8], [9], [10]. The alternating computations of tree automata are particularly interesting because there parallelism occurs on two levels: the automaton reads in parallel independent subtrees of the input and, furthermore, the computation can branch universally.

It is well known that alternating finite automata recognize only the regular languages. Also alternating top-down finite tree automata recognize just the regular forests, cf. [18]. On the other hand, alternation increases dramatically the computational power of finite bottom-up tree automata, cf. [14], [15], [17]. These automata define as yield-languages even all recursively enumerable languages. Other alternating tree automaton models are studied in [11], [12], [16].

Because of the great computational power of alternating tree automata it seems natural to consider restrictions on the alternating computations that would limit the family of recognized forests. Alternation bounds on various Turing machine models and multihead finite automata have been investigated in [3], [5], [6], [7], [10], [13]. There one restricts the width or leaf-size of the computation trees, (or equivalently the number of universal computation steps in an alternating computation.) Also bounds that restrict the size of the alternating computation trees have been investigated, this corresponds to limiting together the length and parallelism of the computation.

Alternation depth of Turing machines, that is the number of alternations of existential and universal configurations in a computation path, is considered in [2]. A similar measure called alternation size which restricts the total number of alternations in a computation tree is defined in [1], and the alternation size of a

*Department of Mathematics, University of Turku, SF-20500 Turku, Finland

Turing machine is shown to correspond closely to the number of reversals of an auxiliary pushdown automaton.

Here we study alternation depth bounds in the context of bottom-up tree automata. For tree automata, the configurations cannot be divided into existential and universal ones as in the case of Turing machines (a tree configuration can contain an arbitrary number of states) and, therefore, we will in fact limit the alternation of existential and universal computation steps in any branch of the computation tree. Intuitively, one may think the edges of the computation tree to be labeled by E or U depending on whether the computation step is existential or universal. On a path from the root to a leaf a sequence of consequtive symbols E (respectively U) that is limited on both sides by symbols U (resp. E) is said to be an existential (resp. universal) computation segment. In an alternation bounded computation tree the number of computation segments on each path is limited by a function on the size of the input.

Clearly even a computation tree with a constant alternation bound may contain an arbitrary number of universal computation steps and thus its width can be linear in the size of the input. However, it is shown that forests recognized by tree automata with a constant alternation bound are regular. This illustrates the fact that the power of alternating tree automata is not only due to an unlimited number of parallel computations but also to the capability of alternating existential and universal computation steps and reading independent subtrees of the input differently in distinct branches of the computation. As a special case it follows that also automata with constant width computation trees define only the regular forests.

On the other hand, it is shown that already a logarithmic alternation bound enables the finite tree automata to recognize also nonregular forests. The main open question is whether the same is true for some sublogarithmic (nonconstant) functions.

In order to establish the above results we need to consider the commutation properties of independent computation steps in an alternating computation. (That is, computation steps at independent nodes of the input tree.) In general, the order in which the recognizer reads independent subtrees of the input can be essential. However, clearly two existential (i.e., nondeterministic) independent computation steps always commute, and the same is true for universal computation steps. Furthermore, a universal and existential computation step semi-commute, that is, a universal computation step may be assumed to be performed first.

In Section 2 we recall the definition of an alternating tree recognizer and establish the above commutation properties. Alternation bounded computations are defined in Section 3 and there it is also shown that constant bounds define only the regular forests. Sections 4 and 5 consider the logarithmic bound and some related alternation measures.

2 Alternating tree automata

The reader is assumed to be familiar with trees and tree automata, cf. e.g. [4]. Here we recall the definition of an alternating finite bottom-up tree recognizer, cf. [14], [15], [17]. Since we will consider computations with a bounded number of alternations of existential and universal computation steps, we use a slightly different definition from that of *op. cit.* There at each computation step the automaton was able to make both an existential and a universal choice. It is shown that in terms of recognition power these models are essentially equivalent.

The set of (nonempty) words over a set X is denoted by X^* (respectively X^+)

and the empty word by λ. The length of a word w is denoted $|w|$. The catenation of $A, B \subseteq X^*$ is defined by $AB = \{w \in X^* | (\exists w_1 \in A)(\exists w_2 \in B) w = w_1 w_2\}$. The set of subsets of X is denoted $\wp(X)$. If X is finite, the cardinality of X is denoted by $\#X$. If X is a (proper) subset of Y this is denoted by $X \subseteq Y (X \subset Y)$. The symbol N_+ stands for the set of positive integers.

A **tree domain** D is a nonempty subset of N_+^* that satisfies the conditions (TD1) and (TD2) below.

(TD1) If $u \in D$, then every prefix of u belongs to D.

(TD2) For every $u \in D$ there exists $i \in N_+ \cup \{0\}$ such that $uj \in D$ iff $1 \leq j \leq i$. ($i = 0$ if u has no daughters.)

Let A be a set. An **A-labeled tree** is a mapping $t : D \rightarrow A$, where D is a tree domain. The elements of D are called nodes of the tree t and D is denoted $\mathrm{dom(t)}$. A node u is said to be labeled by $t(u) (\in A)$. We use freely notions such as the height, root, a leaf, and a subtree of a tree. The height of t is denoted $\mathrm{hg(t)}$ (a tree with one node is defined to have height zero), and the set of leaves of t is denoted $\mathrm{leaf(t)}$. The subtree of t at node u is denoted $\mathrm{t/u}$. A node v is a successor of $u \in \mathrm{dom(t)}$ if u is a prefix of v, and u and v are said to be **independent**, $u \| v$, if neither one is a successor of the other. If $u_1, \ldots, u_m \in \mathrm{dom(t)}$ are pairwise independent, then $t(u_1 \leftarrow t_1, \ldots, u_m \leftarrow t_m)$ denotes the tree obtained from t by replacing t/u_i with $t_i, i = 1, \ldots, m$.

In the term notation one assumes that a node with i daughters (immediate successors) is always labeled by a symbol of rank i. Letters Σ and Ω denote here finite ranked alphabets and the set of m-ary, $m \geq 0$, symbols of Σ is denoted by Σ_m. The rank of an element $\sigma \in \Sigma_m$ is denoted $\mathrm{rank}_\Sigma(\sigma)$ or just $\mathrm{rank}(\sigma)$ if the alphabet Σ is known. Let A be a (finite) set. The set **ΣA-trees**, $F_\Sigma(A)$, is the smallest set such that (i) $\Sigma_0 \cup A \subseteq F_\Sigma(A)$, and (ii) if $\sigma \in \Sigma_m, m \geq 1, t_1, \ldots, t_m \in F_\Sigma(A)$, then $\sigma(t_1, \ldots, t_m) \in F_\Sigma(A)$. The set of **Σ-trees**, F_Σ, is defined to be $F_\Sigma(\emptyset)$. Subsets of F_Σ are called **Σ-forests**. Let $t, t_1, \ldots, t_m \in F_\Sigma(A)$ and $a_1, \ldots, a_m \in A$. Then $t(a_1 \leftarrow t_1, \ldots, a_m \leftarrow t_m)$ denotes the ΣA-tree obtained from t by replacing every occurrence of a symbol a_i with t_i. To a given Σ-tree t one associates in the natural way the corresponding tree domain $\mathrm{dom}(t)$. For $t \in F_\Sigma$ we denote $\mathrm{size(t)} = \#\mathrm{dom}(t)$. A ($\Sigma-$)tree t is said to be **balanced** if every path from the root of t to a leaf has equal length.

We still recall the notion of tree homomorphism, cf. [4]. For every $m \geq 1$, let $\Xi_m = \{\xi_1, \ldots, \xi_m\}$ be a set of variables. Assume that for every $m \geq 0$ such that $\Sigma_m \neq \emptyset$ we are given a mapping $h_m : \Sigma_m \rightarrow F_\Omega(\Xi_m)$. Then the mappings h_m determine inductively a **tree homomorphism** $h : F_\Sigma(A) \rightarrow F_\Omega(A)$ as follows.

(i) $h(a) = a$ for $a \in A$.

(ii) Let $m \geq 0, \sigma \in \Sigma_m, t_1, \ldots, t_m \in F_\Sigma(A)$. Then

$$h(\sigma(t_1, \ldots, t_m)) = h_m(\sigma)(\xi_1 \leftarrow h(t_1), \ldots, \xi_m \leftarrow h(t_m)).$$

It is clear that Σ-trees can be seen as Σ-labeled trees (graphs) where each node having i daughters is labeled by an element of rank i and conversely every Σ-labeled tree with the above property corresponds to a unique Σ-tree. In the following, we speak about trees using interchangeably the above notions of a Σ-labeled graph and an element of F_Σ.

Definition 2.1 *An* **alternating (bottom-up) tree recognizer** *is a four-tuple* $\mathbb{A} = (\Sigma, A, A', G)$, *where*

(i) Σ *is a finite ranked alphabet,*

(ii) A *is a finite set of states,*

(iii) $A' \subseteq A$ *is the a set of final states, and*

(iv) G is the state transition relation defined as follows. The relation G associates with every $\sigma \in \Sigma_m, m \geq 0$, a mapping

$$\sigma_G : A^m \to \wp(A) \times \{E, U\}.$$

The class of alternating tree recognizers is denoted by **ATR**. Let **A** be as in Definition 2.1. Elements of $F_\Sigma(A)$ are called **A-configurations. A configuration tree of A** is a tree where the nodes are labeled **A**-configurations, and the set of configuration trees of **A** is denoted by CT(**A**). Let $T \in$ CT (**A**). Then conf(T) denotes the set of all **A**-configurations labeling some node of T.

Let K be an **A**-configuration. Subtrees of K of the form $\sigma(a_1, \ldots, a_m)$, $m \geq 0$, $\sigma \in \Sigma_m, a_1, \ldots, a_m \in A$ are said to be **active subtrees**, and the set of (occurrences of) active subtrees is denoted by act(K). Let $Z \in \{E, U\}$ and assume that

$$\sigma_G(a_1, \ldots, a_m) = (B, Z), \tag{1}$$

where $\sigma \in \Sigma_m, a_1, \ldots, a_m \in A$ and $B \subseteq A$. Then we denote

$$\sigma_{Z(G)}(a_1, \ldots, a_m) = B,$$

or simply,

$$\sigma_Z(a_1, \ldots, a_m) = B$$

if G is known. We also denote $G = E(G) \cup U(G)$ or simply $G = E \cup U$. This notation can be justified by the fact that always exactly one of the sets $\sigma_E(a_1, \ldots, a_m)$ and $\sigma_U(a_1, \ldots, a_m)$ is defined.

If in (1) $Z = E$, we say that the active subtree $f = \sigma(a_1, \ldots, a_m)$ is **existential (or of type E)**, and if $Z = U, f$ is said to be **universal (or of type U)**. Also $\sigma_Z(a_1, \ldots, a_m)$ is denoted simply by f_Z. If f_Z consists of only one element, we say informally that the corresponding computation step is deterministic. When considering specific examples, the state-transition relation G is usually convenient to define by listing all nonempty sets f_Z where f is an active subtree.

Intuitively, $\sigma_Z(a_1, \ldots, a_m)$ denotes the set of next states the automaton **A** reaches after reading the input symbol σ in states $a_1, \ldots, a_m$. If $\sigma(a_1, \ldots, a_m)$ is existential, the automaton chooses nondeterministically one of the next states in which it continues the computation. If $Z = U$, the computation has to be continued in all states of $\sigma_U(a_1, \ldots, a_m)$. This is defined formally below.

Definition 2.2 *The transition relation $\Rightarrow_{\mathbf{A}}$ of a recognizer* **A** $\in$ *ATR is the binary relation on* CT(**A**) *defined as follows. Let $T_1, T_2 \in$* CT(**A**). *Then $T_1 \Rightarrow_{\mathbf{A}} T_2$ if T_2 is obtained from T_1 as follows. Let n be a leaf of T_1 that is labeled by an* **A**-*configuration K. Let $f \in act(K)$ be of type $Z, Z \in \{E, U\}$, and $f_Z = \{a_1, \ldots, a_m\}, a_i \in A, i = 1, \ldots, m$. If $Z = E$, then T_2 is obtained from T_1 by attaching for the node n a daughter labeled by $K(f \leftarrow a_i)$ for some $i \in \{1, \ldots, m\}$. If $Z = U$, then in T_2 the node n has m daughters labeled by the configurations $K(f \leftarrow a_1), \ldots, K(f \leftarrow a_m)$.*

Let $K \in F_\Sigma(A)$. The set of K-computation trees of **A** is defined by

$$\text{COM}(\mathbf{A}, K) = \{T \in \text{CT}(\mathbf{A}) | K \Rightarrow^*_{\mathbf{A}} T\}.$$

Above K denotes the configuration tree with one node labeled by K. A computation tree is **accepting** if all its leaves are labeled by elements of A', and the set of accepting K-computation trees is denoted ACOM$(\mathbf{A}, K)$. We say that $\mathbf{A}$ accepts a configuration K if ACOM$(\mathbf{A}, K) \neq \emptyset$. The set of accepted $\mathbf{A}$-configurations is denoted by ACC$(\mathbf{A})$. The forest recognized by the recogniser $\mathbf{A}$ is

$$L(\mathbf{A}) = F_\Sigma \cap \mathrm{ACC}(\mathbf{A}).$$

The family of forests recognized by alternating tree recognizers is denoted $\mathbf{L}(\mathbf{ATR})$.

Let $T_1, T_2 \in$ COM$(\mathbf{A}, t)$ for some input t and $T_1 \Rightarrow_{\mathbf{A}} T_2$. We use the notation $T_1 \Rightarrow_{\mathbf{A}}{}^E T_2$ (resp. $T_1 \Rightarrow_{\mathbf{A}}{}^U T_2$) to indicate that T_2 is obtained from T_1 using an existential (resp. universal) computation step in some configuration labeling a leaf of T_1. A computation tree T is said to be **complete** if each leaf of T is labeled by an element of A (i.e., each branch of the computation of T has reached the root of t.)

Example **2.3** *Let* $\mathbf{A} = (\Sigma, A, A', G) \in ATR$ *where* $\Sigma = \Sigma_0 \cup \Sigma_2, \Sigma_0 = \{\tau, \gamma\}, \Sigma_2 = \{\omega\}, A = \{a, b\}, A' = \{a\}$ *and the relation* $G = E(G) \cup U(G)$ *is defined by the following:*

$$\tau_{U(G)} = \{a, b\}, \gamma_{E(G)} = \{a, b\}, \omega_{E(G)}(a, a) = \omega_{E(G)}(b, b) = \{a\},$$

$$\omega_{U(G)}(a, b) = \omega_{U(G)}(b, a) = \{b\}.$$

Let $t = \omega(\tau, \gamma)$. We construct an accepting t-computation tree of $\mathbf{A}$. Below we denote a computation tree T with leaves labeled by configurations $K_1, \ldots, K_m$, $m \geq 1$, simply by $[K_1, \ldots, K_m]$.

$$\begin{aligned} t \;\; &\Rightarrow_{\mathbf{A}}{}^U \;\; [\omega(a, \gamma), \omega(b, \gamma)] \Rightarrow_{\mathbf{A}}{}^E [\omega(a, a), \omega(b, \gamma)] \\ &\Rightarrow_{\mathbf{A}}{}^E \;\; [\omega(a, a), \omega(b, b)] \Longrightarrow_{\mathbf{A}}{}^E [a, \omega(b, b)] \Longrightarrow_{\mathbf{A}}{}^E [a, a]. \end{aligned}$$

It is easy to see that this is the only accepting t-computation tree of $\mathbf{A}$, i.e., in an accepting computation on t, $\mathbf{A}$ must always read the leaf labeled by τ before the leaf γ. (Of course, the computation steps in different branches of the computation tree can be performed in arbitrary order.)

Let $\mathbf{A} = (\Sigma, A, A', G) \in$ ATR. The recogniser $\mathbf{A}$ is said to be **nondeterministic** if all active subtrees of $\mathbf{A}$ are existential. If, furthermore, for all active subtrees f the set f_E contains at most one element, the recogniser $\mathbf{A}$ is said to be **deterministic**. If $\mathbf{A}$ is deterministic and $f_E = \{b\}, b \in A$, we denote f_E simply by b. (A deterministic recogniser could of course also be defined as a special case of a universal one.) It is clear that this definition of nondeterministic and deterministic tree recognizers is equivalent to the usual definitions, cf. [4]. Nondeterministic and determinisitic bottom-up recognisers both recognise the family of regular forests **REG**.

It is known from [14], [15], [17] that alternating bottom-up tree recognizers recognise also nonregular forests. In order to simplify some constructions, in *op. cit.* one uses an automaton model that in each computation step can branch the computation both existentially and universally. In the following we refer to this model as the **generalized alternating automaton** (or recogniser). The definition used here is more restricted as each computation step has to be purely existential or universal. The automaton model of Definition 2.1 cannot straightforwardly simulate the generalised alternating recognisers, however the models are "essentially equivalent" in the sense described below.

Let Σ be a ranked alphabet and M a Σ-forest recognised by a generalised alternating tree automaton **B**. Define the ranked alphabet $\Omega = \Sigma \cup \Lambda$ where $\Lambda = \{\sigma' | \sigma \in \Sigma\}$ and $\Omega_m = \Sigma_m$ if $m \neq 1$ and $\Omega_1 = \Sigma_1 \cup \Lambda$. Let $h : F_\Sigma \to F_\Omega$ be the tree homomorphism defined by

$$h_m(\sigma) = \sigma'(\sigma(\xi_1, \ldots, \xi_m)), m \geq 0, \sigma \in \Sigma_m.$$

Intuitively for $t \in F_\Sigma, h(t)$ is obtained simply by attaching above each node labeled by a symbol σ a node labeled with the unary symbol σ'. We claim that an automaton **A** of Definition 2.1 can recognise the Ω-forest $h(M)$. The proof of this result is essentially the same as the proof of Theorem 4.4 of [14] and we just explain it intuitively below. At a node labeled by σ, **A** performs the existential choice of the generalised alternating automaton **B** recognising M, and then at the node above labeled by σ' the recognizer **A** simulates the universal choice of **B**. Note that by Lemma 2.6 below, without restriction one can assume that **A** performs the universal computation step in σ' immediately after reading the symbol σ, (this could also be seen directly as in the proof given in [14].) Thus it is clear that all computations of **A** on a tree $h(t), t \in F_\Sigma$, correspond to a computation of the generalised automaton on t, and $L(\mathbf{A}) = L(\mathbf{B})$.

Intuitively, the above result means that corresponding to every forest L recognised by a generalised alternating automaton, the family L(ATR) contains a forest essentially similar to L. Now by the results of [14], [15], [17] it is immediate that L(ATR) contains forests that are not regular, and even not context-free (that is, algebraic). Also for instance the emptiness and equivalence problems are undecidable for L(ATR). The tree homomorphism h above does not affect the yield (or frontier), cf. [4], of a forest and hence by [17] it follows that every recursively enumerable language is the yield of a forest in L(ATR).

Finally we define a complete recognizer. An ATR-recognizer is said to be complete if $f_Z \neq \emptyset$ for all active subtrees f of type $Z, Z \in \{E, U\}$. The proof of the following lemma is then immediate.

Lemma 2.4 *Every forest of L(ATR) can be recognized by a complete ATR-recognizer.*

Example 2.5 *Let* $\mathbf{A} = (\Sigma, A, A', G) \in ATR$ *where* $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2, \Sigma_0 = \{\tau, \gamma\}, \Sigma_1 = \{\sigma\}, \Sigma_2 = \{\omega\}, A = \{a, b, b_1, b_2, c, c_1, c_2, d, d_1, d_2, e, e_1, e_2, f\}, A' = \{f\}$ *and the relation* $G = E \cup U$ *is defined by the following:*

(i) $\tau_U = \{a, b\}$
(ii) $\gamma_E = \{d_1, d_2, e\}$
(iii) $\sigma_E(a) = \{c\}$
(iv) $\sigma_U(c) = \{c_1, c_2\}$
(v) $\sigma_E(b) = \{b_1, b_2\}$
(vi) $\sigma_U(b_i) = \{b_i\}, i = 1, 2$
(vii) $\sigma_U(d_i) = \{d_i\}, i = 1, 2$
(viii) $\sigma_U(e) = \{e_1, e_2\}$
(ix) $\omega_E(x, y) = \{f\}$ *if* $(x, y) = (c_i, d_i)$ *or* $(x, y) = (b_i, e_i), i \in \{1, 2\}$.

The relation G is undefined in all cases not covered by (i)-(ix). We denote $t = \omega(\sigma\sigma(\tau), \sigma(\gamma))$ and construct an accepting t-computation tree of **A**. As in the previous example we denote a computation tree with the sequence of configurations labeling its leaves:

$$\begin{aligned}
t \;& \Rightarrow_{\mathbf{A}}{}^{U} [\omega(\sigma\sigma(a), \sigma(\gamma)), \omega(\sigma\sigma(b), \sigma(\gamma))] \\
& (\Rightarrow_{\mathbf{A}}{}^{E})^2 [\omega(\sigma(c), \sigma(\gamma)), \omega(\sigma\sigma(b), \sigma(e))] \\
& (\Rightarrow_{\mathbf{A}}{}^{U})^2 [\omega(c_1, \sigma(\gamma)), \omega(c_2, \sigma(\gamma)), \omega(\sigma\sigma(b), e_1), \omega(\sigma\sigma(b), e_2)] \\
& (\Rightarrow_{\mathbf{A}}{}^{E})^4 [\omega(c_1, \sigma(d_1)), \omega(c_2, \sigma(d_2)), \omega(\sigma(b_1), e_1), \omega(\sigma(b_2), e_2)] \\
& (\Rightarrow_{\mathbf{A}}{}^{U})^4 [\omega(c_1, d_1), \omega(c_2, d_2), \omega(b_1, e_1), \omega(b_2, e_2)] \\
& (\Rightarrow_{\mathbf{A}}{}^{E})^4 [f, f, f, f]
\end{aligned}$$

It can be verified that the computation tree constructed above is the unique accepting t-computation tree of **A**. Of course the computation steps in parallel branches of the computation tree can be performed in different order, but the resulting computation tree will always be unique (if it is accepting.) Also one sees that in the computation starting from the configuration $\omega(\sigma\sigma(a), \sigma(\gamma))$ one must read both σ-symbols in the subtree $\sigma\sigma(a)$ before the symbol γ whereas in the computation starting from $\omega(\sigma\sigma(b), \sigma(\gamma))$ the automaton necessarily has to read the subtree $\sigma(\gamma)$ before continuing the computation in the subtree $\sigma\sigma(b)$.

This example illustrates the fact that in an alternating computation the order in which subtrees of the input are processed can be very essential. This is the reason why an alternating computation cannot in general be simulated by a nondeterministic one using a subset construction. The above example was constructed to be as simple as possible and here of course $L(\mathbf{A})$ is regular, (in fact $L(\mathbf{A}) = \{t\}$.) For examples of alternating tree recognizers defining nonregular forests see [14], [15], [17].

In Examples 2.3 and 2.5 we noticed that the order in which independent subtrees of the input are read can be important. To conclude this section we investigate when computation steps in independent subtrees commute.

As was done in the previous examples, it is many times convenient to denote a computation tree T with the sequence of configurations $[K_1, \ldots, K_m]$ labeling the leaves of T. The configurations $K_i, i = 1, \ldots, m$, contain all information needed to continue the computation of T and also their order is irrelevant. We say that computation trees T_1 and T_2 are equivalent if the leaves of both T_1 and T_2 are labeled by the same sequence of configurations. In this case T_1 can be completed to an accepting computation tree iff the same holds for T_2. In general the computation tree gives also the structure of the computation and we cannot for all purposes replace it with the sequence of its leaves.

An arbitrary sequence of **A**-configurations $[K_1, \ldots, K_m]$ does not necessarily correspond to any computation tree but we can extend the relation $\Rightarrow_{\mathbf{A}}$ in the natural way for arbitrary sequences of configurations:

$$[K_1, \ldots, K_m] \Rightarrow_{\mathbf{A}} [H_1, \ldots, H_n]$$

iff for some $i \in \{1, \ldots, m\}$ there exists a computation tree T with leaves labeled by $M_1, \ldots, M_r$ such that $K_i \Rightarrow_{\mathbf{A}} T$ and the multiset $\{K_1, \ldots, K_{i-1}, M_1, \ldots, M_r, K_{i+1}, \ldots, K_m\}$ equals to the multiset $\{H_1, \ldots, H_n\}$. (Here T is a tree of height one with the root labeled by K_i.) Note in particular that if $[K_1, \ldots, K_m]$ is the sequence of leaves of a computation tree T_1 and $[H_1, \ldots, H_n]$ are the leaves of T_2 then $[K_1, \ldots, K_m] \Rightarrow_{\mathbf{A}}{}^{*} [H_1, \ldots, H_n]$ holds if $T_1 \Rightarrow_{\mathbf{A}}{}^{*} T_2$.

In the next lemma we prove commutation properties of alternating computations. There it is notationally more convenient to consider sequences of configurations instead of computation trees. Using the above definition the alternating computations can be defined also for sequences that do not correspond to any computation tree and we thereby prove a slightly more general result. We still introduce some notation. Let $\mathbf{A} = (\Sigma, A, A', G) \in ATR$. Let $Z \in \{E, U\}$ and $K_i, H_j \in F_\Sigma(A), i = 1, \ldots, r, j = 1, \ldots, s$. Then the notation $[K_1, \ldots, K_r] \Rightarrow_{\mathbf{A}}{}^Z(u, i)[H_1, \ldots, H_s], i \in \{1, \ldots, r\}, u \in \mathrm{dom}(K_i)$, is used to denote that the sequence of configurations $[H_1, \ldots, H_s]$ is obtained from $[K_1, \ldots, K_r]$ by applying a Z-computation step at node u in the configuration K_i. If $r = 1$ or i is otherwise clear we denote $\Rightarrow_{\mathbf{A}}{}^Z(u, i)$ simply by $\Rightarrow_{\mathbf{A}}{}^Z(u)$.

Lemma 2.6 *Let* $\mathbf{A} = (\Sigma, A, A', G) \in ATR$ *and* $K \in F_\Sigma(A), u, v \in \mathrm{dom}(K)$. *Assume that* $u \| v$ *and* $K/u, K/v \in \mathrm{act}(K)$. *Denote* $K/u = f, K/v = g$, *and let* $f_X = \{a_1, \ldots, a_m\}, g_Y = \{b_1, \ldots, b_n\}, m, n \geq 1, X, Y \in \{E, U\}$.

(i) Assume that f and g are existential, (i.e., $X = Y = E$.) Let

$$[K] \Rightarrow_{\mathbf{A}}{}^E(u)[K_1] \Rightarrow_{\mathbf{A}}{}^E(v)[K_2].$$

Then there exists $K' \in F_\Sigma(A)$ such that

$$[K] \Rightarrow_{\mathbf{A}}{}^E(v)[K'] \Rightarrow_{\mathbf{A}}{}^E(u)[K_2].$$

(ii) Assume that f is existential and g universal. Let

$$[K] \Rightarrow_{\mathbf{A}}{}^E(u)[K_1] \Rightarrow_{\mathbf{A}}{}^U(v)[H_1, \ldots, H_n].$$

Then there exist configurations $H'_i, i = 1, \ldots, n$, such that

$$[K] \Rightarrow_{\mathbf{A}}{}^U(v)[H'_1, \ldots, H'_n](\Rightarrow_{\mathbf{A}}{}^E(u))^n[H_1, \ldots, H_n].$$

(iii) Assume that f and g are universal. Suppose that

$$\begin{aligned} &[K] \Rightarrow_{\mathbf{A}}{}^U(u)[K_1, \ldots, K_m] \Rightarrow_{\mathbf{A}}{}^U(v, i) \\ (*) \quad &[K_1, \ldots, K_{i-1}, K_i(v \leftarrow b_1), \ldots, K_i(v \leftarrow b_n), K_{i+1}, \ldots, K_m] \Rightarrow_{\mathbf{A}}{}^*[M_1, \ldots, M_r], \end{aligned}$$

where $v \notin \mathrm{dom}(M_j)$ or $M_j(v) \in A, 1 \leq j \leq r$, i.e., in each of the configurations M_j the active subtree g has been read.

Then there exists a computation

$$\begin{aligned} &[K] \Rightarrow_{\mathbf{A}}{}^U(v)[K(v \leftarrow b_1), \ldots, K(v \leftarrow b_n)] \Rightarrow_{\mathbf{A}}{}^U(u, 1) \\ &[K_1(v \leftarrow b_1), \ldots, K_m(v \leftarrow b_1), K(v \leftarrow b_2), \ldots, K(v \leftarrow b_n)](\Rightarrow_{\mathbf{A}}{}^U(u))^{n-1} \\ (**) \quad &[K_1(v \leftarrow b_1), \ldots, K_m(v \leftarrow b_1), \ldots, K_1(v \leftarrow b_n), \ldots, K_m(v \leftarrow b_n)] \\ &\Rightarrow_{\mathbf{A}}{}^*[M_1, \ldots, M_r]. \end{aligned}$$

Proof. (i) This is immediate since $K_1 = K(u \leftarrow a_i)$ and $K_2 = K(u \leftarrow a_i, v \leftarrow b_j), i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\}$, and clearly we can choose $K' = K(v \leftarrow b_j)$.
(ii) Now $K_1 = K(u \leftarrow a_i), i \in \{1, \ldots, m\}$, and $H_j = K(u \leftarrow a_i, v \leftarrow b_j), j = 1, \ldots, n$. Thus we have

$$\begin{aligned} [K] \; &\Rightarrow_{\mathbf{A}}{}^U(v)[K(v \leftarrow b_1), \ldots, K(v \leftarrow b_n)] \\ &\Rightarrow_{\mathbf{A}}{}^E(u, 1)[K(u \leftarrow a_i, v \leftarrow b_1), K(v \leftarrow b_2), \ldots, K(v \leftarrow b_n)] \\ &\Rightarrow_{\mathbf{A}}{}^E(u, 2) \ldots \Rightarrow_{\mathbf{A}}{}^E(u, n)[H_1, \ldots, H_n]. \end{aligned}$$

(iii) Clearly $K_i = K(u \leftarrow a_i), i = 1,\ldots,m$. Let $N_{1j},\ldots,N_{sj}$ be the configurations from the sequence $M_1,\ldots,M_r$ that are successors of $K_j, j \in \{1,\ldots,i-1, i+1,\ldots,m\}$. Since the configurations $K_i(v \leftarrow b_1),\ldots,K_i(v \leftarrow b_n)$ appear in both (*) and (**), it is sufficient to show that

$$[K_j(v \leftarrow b_1),\ldots K_j(v \leftarrow b_n)] \Rightarrow_{\mathbf{A}}{}^*[N_{1j},\ldots,N_{sj}], j \in \{1,\ldots,m\}, j \neq i. \qquad (2)$$

Let $j \in \{1,\ldots,i-1,i+1,\ldots,m\}$. Let $H_1,\ldots,H_q$ be the configurations appearing in each branch of the computation $[K_j] \Rightarrow_{\mathbf{A}}{}^*[N_{1j},\ldots,N_{sj}]$ just before the automaton reads the subtree g at node v. Thus $[K_j] \Rightarrow_{\mathbf{A}}{}^*[H_1,\ldots,H_q]$ using only computation steps at nodes that are independent with v. From this it follows that

$$[K_j(v \leftarrow b_r)] \Rightarrow_{\mathbf{A}}{}^*[H_1(v \leftarrow b_r),\ldots,H_q(v \leftarrow b_r)], r = 1,\ldots,n. \qquad (3)$$

By the choice of $H_1,\ldots,H_q$,

$$[H_1(v \leftarrow b_1),\ldots,H_1(v \leftarrow b_n),\ldots,H_q(v \leftarrow b_1),\ldots,H_q(v \leftarrow b_n)]$$

$$\Rightarrow_{\mathbf{A}}{}^*[N_{1j},\ldots,N_{sj}]$$

and hence (2) holds by (3). (Note that one may arbitrarily permute the configurations $P_1,\ldots P_x$ in a sequence $[P_1,\ldots,P_x]$.) Q. E. D.

In the above lemma, case (i) states that one may always permute two independent (i.e., corresponding to independent nodes) existential computation steps and (ii) states that an existential computation step followed by a universal computation step may be replaced by first making the universal step and thereafter the corresponding existential computation steps. Case (iii) states that always two independent universal computation steps commute. This is the most complicated case as one does not directly obtain identical configurations but has to consider the computation so far that in each branch both universal active subtrees have been read. This is not a restriction when considering accepting computation trees where each branch ends at the root of the input.

The fourth case would be a universal computation step followed by an independent existential computation step. These cannot (in general) be permuted since in each universal branch the automaton can make different existential choices. Thus one can say that independent existential and universal computation steps semicommute: one may always replace EU with UE but not in the other direction.

3 Depth bounded alternation

As observed in the previous section, alternation is a very powerful mode of computation for bottom-up tree automata. For this reason we consider alternating computations where in each path the number of alternations of the existential and universal computation steps is bounded by some function on the size of the input tree.

Definition 3.1 *We define a mapping alt: $\{E,U\}^+ \to N_+$ as follows. Let $u \in \{E,U\}^+$. Then alt(u) is the least integer n such that we can write*

$$u = u_1 \ldots u_n,\ u_i \in E^+ \cup U^+, i = 1,\ldots,n.$$

Definition 3.2 *Let* $\mathbf{A} = (\Sigma, A, A', G) \in ATR, t \in F_\Sigma$ *and* $T \in COM(\mathbf{A}, t)$. *We define a mapping* $\phi_T : dom(T) \to \{E, U\}^*$ *inductively as follows.*

(i) $\phi_T(\lambda) = \lambda$.

(ii) Let $u \in dom(T)$ *be labeled by an* $\mathbf{A}$*-configuration* K. *Suppose that in the computation of* T *in the configuration* K *the recognizer reads an active subtree of type* $Z, Z \in \{E, U\}$. *Let* $n = \max\{i | ui \in dom(T)\}$. *Then for every* $j = 1, \ldots, n$:

$$\phi_T(uj) = \phi_T(u)Z.$$

(Note that $n \leq 1$ *if* $Z = E$.*)*

Now we define the function alt: $COM(\mathbf{A}, t) \to N_+$ *by*

$$\text{alt}\ (T) = \max\{\text{alt}\ (\phi_T(u)) | u \in \text{leaf}\ (T)\}.$$

Let u be a node of a computation tree T. Then $\phi_T(u)$ gives the sequence in which existential and universal computation steps are performed along the path from the root of T to u. Thus $\text{alt}(\phi_T(u))$ denotes the number of existential and universal computation segments in the computation corresponding to the node u. Now depth bounded alternating computations can be defined by restricting the value $\text{alt}(T)$.

Definition 3.3 *Let* $\mathbf{A} = (\Sigma, A, A', G) \in ATR, t \in F_\Sigma$, *and* $\theta : N_+ \to N_+$ *be a function. The set of* θ-**bounded t-computation trees of A** *is*

$$\text{COM}(\mathbf{A}, t)[\theta] = \{T \in \text{COM}(\mathbf{A}, t) | \text{alt}(T) \leq \theta(\text{size}(t))\}.$$

A forest $L \subseteq F_\Sigma$ *is* θ-**bounded recognized by A if**

(i) $L = L(\mathbf{A})$, *and*

(ii) for every $t \in L$, $COM(\mathbf{A}, t)[\theta] \cap ACOM(\mathbf{A}, t) \neq \emptyset$.

In this case we denote $L = L(\mathbf{A})[\theta]$. *The family of forests* θ*-bounded recognized by alternating tree recognizers is denoted* $L(ATR)[\theta]$.

Thus L is θ-bounded recognized by $\mathbf{A}$ if each tree t of L has an accepting computation tree with alternation depth at most $\theta(\text{size}(t))$ and any tree not in L does not have an accepting computation. If $L(\mathbf{A})[\theta]$ is defined, we say also that the recognizer $\mathbf{A}$ is θ-**bounded.** Note that if one would define $L(\mathbf{A})[\theta]$ just to consist of trees $t \in F_\Sigma$ such that there exists an accepting computation tree in $\text{COM}(\mathbf{A}, t)[\theta]$, then the automaton would be able to use the counting properties of the function θ to check properties of the inputs. This would clearly be unnatural, especially if the function θ is not well behaved. Note that $L(\mathbf{A})[\theta]$ is not defined if $L(\mathbf{A})$ contains trees that cannot be accepted in θ-bounded computations.

Let t and $\mathbf{A}$ be as in Example 2.5 and let T be the t-computation tree considered there. Then $\text{alt}(T) = 6$. Thus $L(\mathbf{A})[\theta] = \{t\}$ for every function θ such that $\theta(6) \geq 6$. (Note that $\text{size}(t) = 6$.)

Lemma 3.4 *For every function* θ, *the family* $L(ATR)[\theta]$ *is closed with respect to intersection with regular sets.*

Proof. This is seen easily by adding to the states of an ATR-recognizer second components that simulate the computation of a deterministic recogniser for the regular forest in question. Clearly the simulation can be done at the same time preserving the type (existential or universal) of each computation step. Q.E.D.

Theorem 3.5 *Suppose that $\theta_1(n) \leq \theta_2(n)$ almost everywhere, i.e., there exists $M \in N_+$ such that for all $n \geq M, \theta_1(n) \leq \theta_2(n)$. Then*

$$L(ATR)[\theta_1] \subseteq L(ATR)[\theta_2].$$

Proof. Let L be a Σ-forest θ_1-bounded recognized by $\mathbb{A} \in$ ATR. Denote $F(\geq) = \{t \in F_\Sigma \,|\, \text{size}(t) \geq M\}$ and $F(<) = \{t \in F_\Sigma \,|\, \text{size}(t) < M\}$. Now

$$L = (L(\mathbb{A})[\theta_1] \cap F(\geq)) \cup (L(\mathbb{A})[\theta_1] \cap F(<)).$$

By Lemma 3.4 there exists $\mathbb{B} \in$ ATR such that $L(\mathbb{B})[\theta_1] = L(\mathbb{A})[\theta_1] \cap F(\geq)$. By the choice of M, furthermore, $L(\mathbb{B})[\theta_1] = L(\mathbb{B})[\theta_2]$. Since $F(<)$ is finite, using $\mathbb{B}$ one can easily construct a recogniser $\mathbb{B}'$ such that

$$L(\mathbb{B}')[\theta_2] = L(\mathbb{B})[\theta_2] \cup (L(\mathbb{A})[\theta_1] \cap F(<)) = L.$$

Q.E.D.

Clearly $L(\text{ATR})[id] = L(\text{ATR})$ where id denotes the identity function. In the following we will consider constant and logarithmic alternation bounds. Let $c(k), k \geq 1$, denote the function that maps every element of N_+ to k. Then $L(\text{ATR})[c(1)] = \text{REG}$ because both the purely existential and purely universal tree automata recognise only the regular forests. Next we will show that in fact $L(\text{ATR})\,[c(k)] = \text{REG}$ for all $k \geq 1$. In the following we denote the function $c(k)$ simply by k.

A first idea for a regularity proof for the forests of $L(\text{ATR})[k]$ might be to simulate the k-bounded alternating computations by a deterministic tree recognizer using a subset construction where the sets would additionally contain the information which existential (resp. universal) segment of the computation one is simulating. (There can be at most k segments.) However, this approach does not work because it may be the case that in different branches of the computation a given node must be read in different segments.

To illustrate the difficulty, let us consider again from Example 2.5 the t-computation tree which is the unique accepting computation tree for the input $t = \omega(\sigma\sigma(\tau), \sigma(\gamma))$. For instance, in the left branch of the computation the symbol γ has to be read in the second existential segment whereas in the right branch it is necessarily read in the first existential segment.

It turns out that a deterministic automaton simulating the computations of $\mathbb{A}$ will need to store in the states the information concerning the partition into existential and universal segments of all possible computation trees of the input scanned so far, this will be called the *computation schema*. It will be seen that for k-bounded computations the number of distinct computation schemata is finite. Let $\mathbb{A} \in$ ATR, $t = \sigma(t_1, \ldots, t_m) \in F_\Sigma$ and T be a complete t-computation tree of $\mathbb{A}$. The computation tree T is obtained by combining t_i-computation trees, $i = 1, \ldots, m$, and finally in each branch reading the root symbol σ. Thus it is clear that one can construct an arbitrary (k-bounded) t-computation tree if one knows all possible (k-bounded) t_i-computation trees.

In the following $\mathbb{A} = (\Sigma, A, A', G) \in$ ATR is always assumed to be complete. By Lemma 2.4 this is not a restriction. (Clearly the analogy of Lemma 2.4 holds also for arbitrary θ-bounded computations.) We say that $K \in F_\Sigma(A)$ is an **existential** configuration if all active subtrees of K are existential and otherwise K is said to be **universal**. In particular, if $K \in A$ then $\text{act}(K) = \emptyset$ and hence K is existential.

Lemma 3.6 *Let K be a universal configuration. Then there exist unique existential configurations $K_1, \ldots, K_n$ such that*

$$[K](\Rightarrow_{\mathbb{A}}{}^{U})^*[K_1, \ldots, K_n]. \qquad (4)$$

(Of course $K_1, \ldots, K_n$ may be arbitrarily permuted.)

Proof. Since **A** is complete, there are existential configurations $K_1, \ldots, K_n$ such that (4) holds (one computes universal active subtrees until there are none left.) By Lemma 2.6 (iii) universal computation steps commute and hence the configurations $K_1, \ldots, K_n$ are unique.

Definition 3.7 *Let $t \in F_\Sigma$ and $k \geq 1$. The* **k-bounded t-computation schema** *of* **A**, *$SC(t, k, \mathbf{A})$ is the configuration tree S defined as follows. The root of S is labeled by t. Suppose that a node $u \in dom(S), |u| \leq k-1$, is labeled by a configuration K.*

(i) Suppose that K is universal and let $K_1, \ldots, K_n$ be the (by Lemma 3.6 unique) existential configurations such that $[K](\Rightarrow_{\mathbf{A}}{}^U)^*[K_1, \ldots, K_n]$. Then the node u has n daughters (immediate successors) labeled by $K_1, \ldots, K_n$.

(ii) Let K be existential, $K \notin A$. Denote by C the set of all configurations K' such that $[K](\Rightarrow_{\mathbf{A}}{}^E)^*[K']$ and K' is universal or $K' \in A$. Then for every $K' \in C$ the node u has a daughter labeled by K'.

(iii) If $K \in A$ then u is a leaf of S.

Finally, if $u \in dom(S)$ and $|u| = k$, then u has no daughters.

If u is a leaf of S and $S(u) \notin A$, then this branch corresponds to a computation that does not reach the root of the tree t in k existential and universal computation segments. These computations cannot be a part of any k-bounded computation on an input with subtree t and thus the corresponding branches can be pruned from the schema.

The **pruned schema**, $pr(S)$, is obtained from S by recursively repeating the following. Choose a leaf u of S labeled by an element not belonging to A and let v be the mother (immediate predecessor) of u. If the configuration $S(v)$ is existential then remove the node u. If $S(v)$ is universal then remove all daughters of v. (If a universal configuration K has a daughter leading to failure then the computation has failed already in K.) Note that $pr(S)$ is the empty tree iff there does not exist a complete k-bounded t-computation tree of **A**. In this case t is not a subtree of any tree of $L(\mathbf{A})[k]$.

The pruned computation schema $pr(\mathrm{SC}(t, k, \mathbf{A}))$ will be denoted by $\mathrm{PSC}(t, k, \mathbf{A})$ and in the following, when not otherwise mentioned, by a computation schema we always mean the pruned schema.

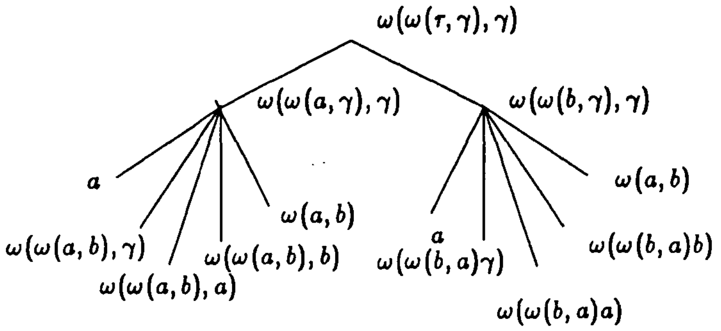
Suppose that $\mathrm{S} = \mathrm{PSC}(t, k, \mathbf{A})$. An **A**-configuration tree T is said to be a **configuration tree associated with the schema S** if T is constructed as follows. The root of T is labeled by t. Suppose that $u \in \mathrm{dom}(S) - \mathrm{leaf}(S)$ is labeled by a configuration K.

(i) If K is universal, then the node u has in T all the same daughters as in S.

(ii) If K is existential, then u has exactly one daughter labeled by some configuration that is a daughter of K in S.

Thus a configuration tree associated with the schema S is essentially a t-computation tree where some intermediate nodes in the existential and universal computation segments have been removed.

Example 3.8 *Let $\mathbf{A} = (\Sigma, A, A', G)$ be the recognizer from Example 2.3 and $t = \omega(\omega(r, \gamma), \gamma)$. Then the 2-bounded t-computation schema $SC(t, 2, \mathbf{A})$ is given in Figure 1. The configurations $a, \omega(\omega(a, \gamma), \gamma)$, and $\omega(\omega(b, \gamma), \gamma)$ are existential, all other configurations appearing in the schema are universal. The pruned computation schema $PSC(t, 2, \mathbf{A})$ is obtained by removing all leaves except the ones labeled by a. The schema $PSC(t, 2, \mathbf{A})$ has only one associated configuration tree and it equals to $PSC(t, 2, \mathbf{A})$.*



$\omega(\omega(\tau,\gamma),\gamma)$
$\omega(\omega(a,\gamma),\gamma)$
$\omega(\omega(b,\gamma),\gamma)$
a
$\omega(a,b)$
$\omega(\omega(a,b),\gamma)$
$\omega(\omega(a,b),a)$
$\omega(\omega(a,b),b)$
$\omega(a,b)$
a
$\omega(\omega(b,a)\gamma)$
$\omega(\omega(b,a)a)$
$\omega(\omega(b,a)b)$

Figure 1.

It is clear that every configuration tree associated with a schema corresponds to a computation tree of the recognizer **A**. In fact we have the correspondence also in the converse direction.

Let $\mathbf{A} = (\Sigma, A, A', G) \in$ ATR and $t \in F_\Sigma, k \geq 1$. We say that $T \in$ COM$(\mathbf{A}, t)$ is **normalized** if the following condition holds. If some leaf of t is universal (i.e., t is a universal configuration), then at the root of T the recognizer reads a universal active subtree.

Lemma 3.9 *Let* **A** *and t be as above and $k \geq 1$. Then there exists a complete normalized computation tree in COM$(\mathbf{A}, t)[k]$ with leaves labeled by $a_1, \ldots, a_n (a_i \in A)$ iff there is a configuration tree associated with the schema PSC$(t, k, \mathbf{A})$ having leaves $a_1, \ldots, a_n$.*

Proof. The proof in the "if"-direction is immediate since a configuration tree T associated to the schema is clearly a computation tree of **A** where some intermediate nodes are left out. According to the definition of PSC$(t, k, \mathbf{A})$ if t has a universal leaf-symbol, then the computation of T first branches universally, i.e., the corresponding computation tree is normalized.

Suppose then that $T \in$ COM$(\mathbf{A}, t)[k]$ is normalized and has leaves $a_1, \ldots, a_n$. Using the commutation properties of Lemma 2.6 we show that there exists an equivalent $T_1 \in$ COM$(\mathbf{A}, t)[k]$ (i.e., also T_1 has the leaves $a_1, \ldots, a_n$) that follows the computation in the schema PSC$(t, k, \mathbf{A})$.

Since T is normalized, if t contains a universal leaf-symbol the automaton first makes a universal computation step. By Lemma 2.6 universal computation steps commute with each other and semi-commute with existential computation steps. Thus in an equivalent computation tree one can first make all possible universal computation steps (in arbitrary order). Now the computation of T_1 begins as in the schema PSC$(t, k, \mathbf{A})$. Always in an existential configuration K the automaton makes an arbitrary number of consequtive existential computation steps that lead to some universal configuration K'. Thus K' is a daughter of K in PSC$(t, k, \mathbf{A})$. In K', **A** makes a universal computation step and thus again by Lemma 2.6 it can be made to read all universal active subtrees of K' successively yielding a number of existential configurations. By Lemma 3.6, these are exactly the daughters of

K' in PSC$(t, k, \mathbf{A})$. (Note that if a configuration K contains a universal active subtree f then by Lemma 2.6 in a K-computation tree $\mathbf{A}$ could always first read f. In general this could cause additional alternations of the existential and universal computation steps. However here this problem does not occur because the original computation step made in K' is universal and thus one can make thereafter an arbitrary number of universal steps "for free".)

Since T is k-bounded so is also T_1 (any operations above do not increase the existential-universal alternations.) Thus T is equivalent to a configuration tree associated with the schema PSC$(t, k, \mathbf{A})$. Q.E.D.

The previous lemma gives almost a criterion for checking whether $t \in L(\mathbf{A})[k]$ using only the schema PSC$(t, k, \mathbf{A})$. There is still the restriction that the computation tree has to be normalized. This restriction can be removed by considering $(k+1)$-bounded schemata.

Lemma 3.10 *Let* $\mathbf{A} = (\Sigma, A, A', G) \in ATR$, $k \geq 1$, *and* $L = L(\mathbf{A})[k]$. *Then* $t \in L$ *iff there exists a configuration tree* W *associated with the schema* $PSC(t, k+1, \mathbf{A})$ *such that all leaves of* W *are labeled by elements of* A'.

Proof. Let $t \in L$ and $T \in$ COM$(\mathbf{A}, t)[k]$ be accepting. Assume that at least one leaf of t is labeled by a universal symbol. Then, by Lemma 2.6, T can be transformed to an equivalent computation tree T_1 where first the automaton performs all possible universal computation steps, i.e., T_1 is normalized. Furthermore, from the proof of Lemma 2.6 it follows that T_1 is $(k+1)$-bounded. Moving a number of universal computation steps to the beginning may introduce an additional universal computation segment if the computation of T starts existentially. (Of course it is also possible that alt$(T_1) < k$, but for our purposes it is sufficient just to know the upper bound alt$(T_1) \leq k+1$.) On the other hand, if all leaves of t are existential then already the computation tree T is normalized. Thus in both cases by Lemma 3.9 there exists a configuration tree W associated with the schema PSC$(t, k+1, \mathbf{A})$ such that the leaves of W are labeled by elements of A' (since T_1 is accepting).

Conversely assume that W as above exists. Then by Lemma 3.9, there exists an accepting computation tree in COM$(\mathbf{A}, t)[k+1]$. Thus $t \in L(\mathbf{A})$ and there necessarily exists also an accepting k-bounded t-computation tree. Q.E.D.

According to Lemmas 3.9 and 3.10 the schema PSC$(t, k+1, \mathbf{A})$ contains the information on all complete k-bounded t-computation trees of $\mathbf{A}$. We want to define a deterministic tree automaton that stores the schemata in its states. For this purpose we need to consider the composition of schemata.

Definition 3.11 *Let* $\sigma \in \Sigma_m, t_1, \ldots, t_m \in F_\Sigma$ *and* $S_i = PSC(t_i, k, \mathbf{A}), k \geq 1$. *We define the* σ-**composition** *of the schemata* $S_i, i = 1, \ldots, m, \sigma(S_1, \ldots, S_m)$, *as follows. First we construct a tree* S *(that will be the corresponding unpruned schema). The root of* S *is labeled by* $\sigma(t_1, \ldots, t_m)$. *Suppose that a node* $u \in dom(S)$, $|u| \leq k-1$, *is labeled by* $\sigma(K_1, \ldots, K_m)$ *where* $K_i = S_i(v_i), i = 1, \ldots, m$, *and the node* $v_i \in dom(S_i)$ *has* r_i *daughters. (Note that* $t_i = S_i(\lambda), i = 1, \ldots, m$.)

(i) Let $\sigma(K_1, \ldots, K_m)$ *be existential (i.e.,* $K_1, \ldots, K_m$ *are all existential). Then the node* u *has daughters labeled by all configurations* $\sigma(K'_1, \ldots, K'_m)$ *where*

(ia) K'_i *is a daughter of* K_i *(in* S_i*) or* $K'_i = K_i$, *and,*

(ib) there exists at least one j *such that* $K'_j \neq K_j$ *and* $K'_j \notin A$, *(i.e.,* K'_j *is universal.)*

(ic) Furthermore, if a_i *is a daughter of* K_i *in* $S_i, i = 1, \ldots, m$, *and* $\sigma(a_1, \ldots, a_m)$ *is existential, then* u *has daughters labeled by all elements of* $\sigma_E(a_1, \ldots, a_m)$. *If* $\sigma(a_1, \ldots, a_m)$ *is universal, then* u *has a daughter labeled by* $\sigma(a_1, \ldots, a_m)$. *(Note*

that these conditions guarantee that all daughters of $\sigma(K_1, \ldots, K_m)$ are universal configurations or elements of A.)

(ii) Suppose that $\sigma(K_1, \ldots, K_m)$ is universal and that K_j is universal iff $j \in \{i_1, \ldots, i_c\}, c \geq 1, 1 \leq i_1 < \ldots < i_c \leq m$. Then the node u has $r_{i_1} \ldots r_{i_c}$ daughters labeled by the configurations $\sigma(K'_1, \ldots, K'_m)$ where

$$K'_j = \begin{cases} K_j & \text{if } j \notin \{i_1, \ldots, i_c\} \\ \text{some daughter of } K_j & \text{if } j \in \{i_1, \ldots, i_c\}. \end{cases}$$

Furthermore if for some $\sigma(K'_1, \ldots, K'_m)$ above $K'_i = a_i \in A, i = 1, \ldots, m$, and $\sigma(a_1, \ldots, a_m)$ is universal, then the node $\sigma(K'_1, \ldots, K'_m)$ is replaced by nodes labeled by elements of $\sigma_U(a_1, \ldots, a_m)$.

Finally, if $u \in dom(S)$ and $|u| = k$, then u is a leaf of S.

Now the composition $\sigma(S_1, \ldots, S_m)$ is defined to be $pr(S)$ where pr is the pruning function defined after Definition 3.7.

Clearly the σ-composition of k-bounded schemata is a tree of height at most k such that existential and universal configurations alternate as internal nodes in each branch and all leaves are labeled by elements of A. The composition of schemata respects the composition of trees as follows.

Lemma 3.12 *Let $k \geq 1, m \geq 1, \sigma \in \Sigma_m$, and $t_1, \ldots, t_m \in F_\Sigma$. Denote $t = \sigma(t_1, \ldots, t_m), S = PSC\ (t, k, \mathbf{A})$ and $S_i = PSC\ (t_i, k, \mathbf{A}), i = 1, \ldots, m$. Then*

$$S = \sigma(S_1, \ldots, S_m).$$

Proof. This follows straightforwardly from the definition of σ-composition. In the schema S the daughters of an existential node $\sigma(K_1, \ldots, K_m)$ are all universal configurations K such that $\sigma(K_1, \ldots, K_m)(\Rightarrow_{\mathbf{A}}{}^{E})^{+}K$. (Here K may be also existential if $K \in A$.) These are exactly the configurations where at least one K_i is replaced by its daughter in S_i as in Definition 3.11 (i) (where the case $K \in A$ is handled separately.)

Similarly, the daughters of a universal configuration $\sigma(K_1, \ldots, K_m)$ in S are exactly all configurations obtained from $\sigma(K_1, \ldots, K_m)$ by reading all the universal active subtrees. These are obtained from the daughters of universal configurations K_j as in Definition 3.11 (ii). (Note that if K_j is not universal then each $f \in \text{act}(K_j)$ is existential and K_j necessarily remains unchanged in the universal computation segment starting from $\sigma(K_1, \ldots K_m)$.)

Finally the branches in the composition $\sigma(S_1, \ldots, S_m)$ are terminated after the k^{th} level exactly as in the schema S. Q.E.D.

Next we define the reduced simplified computation schemata that will contain all essential information about the corresponding k-bounded computation trees. Intuitively, the reduced simplified schema is obtained by removing the labels of internal nodes and then identifying identical subtrees. This means that the set of reduced simplified schemata will be finite.

Let $S = \text{PSC}(t, k, \mathbf{A}), t \in F_\Sigma, k \geq 1$; the **simplified schema** corresponding to $S, \text{sim}(S)$, is defined by relabeling each internal existential and universal node respectively by E and U. The **reduced simplified schema**, redsim(S), is obtained by identifying identical subtrees of a given node of sim(S) recursively in the bottom-up direction.

Set $S_0 = \text{sim}(S)$. Suppose that $ui, uj \in \text{dom}(S_r), r \geq 0, u \in N_+^*, i, j \in N_+, i < j$, and $S_r/ui = S_r/uj$. Furthermore we assume that $S_r/vi_1 \neq S_r/vi_2$ always when

$i_1 \neq i_2$ and u is a proper prefix of v, i.e., u is chosen to be maximal. Then one defines S_{r+1} to be the tree obtained by removing the subtree S_r/uj from S_r. There exists $C \in N_+$ such that $S_r = S_{r+1}$ always when $r \geq C$ and we define

$$\text{redsim}(S) = S_C.$$

The construction of S_{r+1} from S_r was defined nondeterministically. However because u is always chosen to be maximal, it is clear that redsim(S) is well defined. (Equivalently one could consider some fixed order for the identification process.)

Now for each $k \geq 1$, the cardinality of the set

$$D(k) = \{\text{redsim}(\text{PSC}(t, k, \mathbf{A})) | t \in F_\Sigma\}$$

is finite. Already the simplified schema sim(S), $S = \text{PSC}(t, k, \mathbf{A})$, is a tree of height at most k where the nodes are labeled by elements of $A \cup \{E, U\}$. However, the number of daughters of a given node of sim(S) is in general unbounded (since t can be arbitrary). In redsim(S) one obtains a bound for the arity of the nodes (assuming that also k is fixed). In fact, $\#D(1) \leq 2(2^{\#A} - 1) + 1 = 2^{\#A+1} - 1$ (PSC$(t, k, \mathbf{A})$ may also be the empty schema), and in general $\#D(k+1) \leq 2^{[\#D(k)+\#A+1]}$.

Thus a finite automaton can use the reduced simplified schemata to remember all possible k-bounded computation trees of the input processed so far. We still need to define the compositions of simplified schemata. This is done completely analogously with Definition 3.11. In fact, these definitions could both be obtained as special cases from a more general notion of composition of schemata. However, we presented Definition 3.11 separately because it has a very clear intuitive meaning which makes also the idea behind the next definition more transparent.

Definition 3.13 *Let* $\mathbf{A} \in ATR$ *and* $k \geq 1$. *Let* $m \geq 1, \sigma \in \Sigma_m$, *and* $S_1, \ldots, S_m$ *be simplified schemata (i.e., computation schemata where the internal nodes are labeled just by* E *and* U*). Then the composition of* $S_1, \ldots, S_m$,

$$S = \sigma(S_1, \ldots, S_m)$$

is defined by the following. First we define a tree T *as follows. Nodes of* T *are labeled by elements of* A *or elements of the form* $\sigma(x_1, \ldots, x_m)$ *where* $x_i = (S_i(u_i), u_i), u_i \in dom(S_i)$. *An element* $\sigma(x_1, \ldots, x_m)$ *is said to be universal if*

$$\text{there exists } x_i = (S_i(u_i), u_i) \text{ such that } S_i(u_i) = U, \text{ or} \tag{5}$$

$$S_1(u_1), \ldots, S_m(u_m) \in A \text{ and } \sigma(S_1(u_1), \ldots, S_m(u_m)) \text{ is a universal active subtree.} \tag{6}$$

Otherwise $\sigma(x_1, \ldots, x_m)$ *is existential.*

The root of T *is labeled by* $\sigma((S_1(\lambda), \lambda), \ldots, (S_m(\lambda), \lambda))$. *Assume that a node* $u \in dom(T), |u| \leq k-1$, *is labeled by an element* $R = \sigma((S_1(u_1), u_1), \ldots, (S_m(u_m), u_m))$.

(i) Suppose that R *is existential. Then* u *has daughters labeled by elements*

$$\sigma((S_1(v_1), v_1), \ldots, (S_m(v_m), v_m)) \tag{7}$$

where $(a) v_i = u_i$ *or* $(b) v_i = u_i n, n \in N_+, u_i n \in dom(S_i)$, *and for at least one* $i \in \{1, \ldots, m\}$ *the case* (b) *holds with* $S_i(v_i) = U$. *Furthermore, if* $a_i \in A$ *is a daughter of the node* u_i *in* $S_i, i = 1, \ldots, m$, *and* $\sigma(a_1, \ldots, a_m)$ *is an existential* *active subtree of* **A**, *then u has also daughters labeled by elements of $\sigma_E(a_1, \ldots, a_m)$. If $\sigma(a_1, \ldots, a_m)$ is a universal active subtree, then u has a daughter labeled by $\sigma((a_1, u_1n_1), \ldots, (a_m, u_mn_m))$ where $a_i = S_i(u_in_i), n_i \in N_+, i = 1, \ldots, m$.*

(ii) Suppose that R is universal. Then u has daughters labeled by elements

$$\sigma((S_1(v_1), v_1), \ldots, (S_m(v_m), v_m)) \tag{8}$$

where (a)$v_i = u_i$ if $S_i(u_i) = E$, and (b)v_i is a daughter of u_i(in S_i) if $S_i(u_i) = U, i = 1, \ldots, m$. Furthermore if for some element as in (8), $S_i(v_i) = a_i \in A, i = 1, \ldots, m$, and $\sigma(a_1, \ldots, a_m)$ is a universal active subtree of **A** *then this node is replaced by nodes labeled by elements of $\sigma_U(a_1, \ldots, a_m)$.*

Next we relabel the existential inner nodes of T by E and the universal inner nodes by U. The nodes of T are said to be universal or existential according to (5) and (6). The labels of leaves of T are left unchanged in the relabeling. We denote by T_1 the tree obtained from T as the result of the relabeling.

Now the composition S is obtained by pruning the tree T_1, i.e.,

$$\sigma(S_1, \ldots, S_m) = pr(T_1).$$

Here for the definition of the pruning function pr one considers the internal nodes of T_1 labeled by E to be existential and those labeled by U to be universal. Thus in $\sigma(S_1, \ldots, S_m)$ all leaves are labeled by elements of A and it is a simplified computation schema. Note that the composition $\sigma(S_1, \ldots, S_m)$ need not be reduced even if the schemata $S_1, \ldots, S_m$ are reduced.

Lemma 3.14 *Let $\sigma \in \Sigma_m$, and $S_1, \ldots, S_m$ be k-bounded computation schemata of* **A**, *$k \geq 1$. Then*

$$\mathrm{sim}(\sigma(S_1, \ldots, S_m)) = \sigma(\mathrm{sim}(S_1), \ldots, \mathrm{sim}(S_m)).$$

Proof. This follows immediately from the Definitions 3.11 and 3.13. For the σ-composition of the computation schemata $S_1, \ldots, S_m$ (in Definition 3.11) one uses the configurations labeling the internal nodes of $S_i, i = 1, \ldots, m$, only to determine whether the node is existential or universal. Hence it does not make a difference whether the configurations are replaced by the symbols E and U before or after the composition. Q.E.D.

Lemma 3.15 *Let $\sigma \in \Sigma_m$, and $S_1, \ldots, S_m$ be simplified k-bounded computation schemata of* **A**, *$k \geq 1$. Then*

$$\mathrm{red}(\sigma(\mathrm{red}(S_1), \ldots, \mathrm{red}(S_m))) = \mathrm{red}(\sigma(S_1, \ldots, S_m)).$$

Proof. Denote $R_1 = \sigma(\mathrm{red}\ (S_1), \ldots, \mathrm{red}(S_m))$ and $R_2 = \sigma(S_1, \ldots, S_m)$. Since $\mathrm{red}(S_i)$ is obtained by identifying some identical subtrees of $S_i, i = 1, \ldots, m$, it follows that R_1 is obtained from R_2 by identifying some subtrees. Thus it is clear that $\mathrm{red}(R_1) = \mathrm{red}(R_2)$. Q.E.D.

Lemma 3.16 *Let $k \geq 1, m \geq 1, \sigma \in \Sigma_m, t_1, \ldots, t_m \in F_\Sigma$, and denote $t = \sigma(t_1, \ldots, t_m)$. Then*

$$\mathrm{redsim}(\mathrm{PSC}(t, k, \mathbf{A})) = \tag{9}$$

$$\mathrm{red}(\sigma(\mathrm{redsim}(\mathrm{PSC}(t_1, k, \mathbf{A})), \ldots, \mathrm{redsim}(\mathrm{PSC}(t_m, k, \mathbf{A})))).$$

Proof. Denote $S = \mathrm{PSC}(t, k, \mathbf{A})$ and $S_i = \mathrm{PSC}(t_i, k, \mathbf{A}), i = 1, \dots, m$. By Lemma 3.12,

$$S = \sigma(S_1, \dots, S_m).$$

Thus by Lemma 3.14,

$$\mathrm{sim}(S) = \sigma(\mathrm{sim}(S_1), \dots, \mathrm{sim}(S_m))$$

and (9) follows from Lemma 3.15. Q.E.D.

Now using Lemma 3.16, corresponding to an alternating recogniser $\mathbf{A}$ we can construct a deterministic tree recognizer that arrives at the root of an input tree t in the state $\mathrm{redsim}(\mathrm{PSC}(t, k, \mathbf{A}))$.

Theorem 3.17 *For every $k \geq 1$,*

$$L(\mathrm{ATR})[k] = \mathrm{REG}.$$

Proof. Clearly it is sufficient to show that $L(\mathrm{ATR})[k] \subseteq \mathrm{REG}$. Let $\mathbf{A} = (\Sigma, A, A', G) \in \mathrm{ATR}$ and suppose that $L = L(\mathbf{A})[k], k \geq 1$.

Denote by A-SCHEMA$(k, \mathbf{A})$ the set of all k-bounded computation schemata of $\mathbf{A}, S = \mathrm{PSC}(t, k, \mathbf{A})$, such that S has an associated configuration tree with all leaves labeled by elements of A'. Now we construct a deterministic recognizer

$$\mathbf{B} = (\Sigma, B, B', H)$$

where

(i) $B = \{\mathrm{redsim}(\mathrm{PSC}(t, k + 1, \mathbf{A})) | t \in F_\Sigma\}$,

(ii) $B' = \{\mathrm{redsim}(\mathrm{PSC}(t, k + 1, \mathbf{A})) | t \in F_\Sigma, PSC(t, k + 1, \mathbf{A}) \in A - \mathrm{SCHEMA}(k + 1, \mathbf{A})\}$,

(iii) the relation H is defined by

(a) $\sigma_{E(H)} = \mathrm{redsim}(\mathrm{PSC}(\sigma, k + 1, \mathbf{A}))$ if $\sigma \in \Sigma_0$,

(b) $\sigma_{E(H)}(S_1, \dots, S_m) = \mathrm{red}(\sigma(S_1, \dots, S_m))$,

if $m \geq 1, \sigma \in \Sigma_m, S_1, \dots, S_m \in B$. (Here $\sigma(S_1, \dots, S_m)$ denotes of course the σ-composition of simplified schemata.)

The set of final states B' is well defined. If $S = \mathrm{PSC}(t, k + 1, \mathbf{A})$ then S is of course not determined by $\mathrm{redsim}(S)$. However, using $\mathrm{redsim}(S)$ one can determine whether $S \in A - \mathrm{SCHEMA}(k + 1, \mathbf{A})$. One constructs the associated trees of $\mathrm{redsim}(S)$ by taking all successors of universal nodes and exactly one successor of an existential node. Since $\mathrm{redsim}(S)$ is obtained from S by relabeling internal nodes and identifying identical subtrees it is clear that there exists an associated tree of $\mathrm{redsim}(S)$ with all leaves labeled by elements of A' iff $S \in A - \mathrm{SCHEMA}(k + 1, \mathbf{A})$. This observation also guarantees that the construction of $\mathbf{B}$ is effective.

Now we claim that for every $t \in F_\Sigma$ the recognizer $\mathbf{B}$ reaches the root of t in the state $\mathrm{redsim}(\mathrm{PSC}(t, k + 1, \mathbf{A}))$. If $t \in \Sigma_0$ this follows from the definition of H. Suppose then that $m \geq 1, \sigma \in \Sigma_m, t = \sigma(t_1, \dots, t_m)$ and the claim holds for $t_1, \dots, t_m$. Then $\mathbf{B}$ reaches the root of t in the state

$$\begin{aligned} & \sigma_{E(H)}(\mathrm{redsim}(\mathrm{PSC}(t_1, k + 1, \mathbf{A})), \dots, \mathrm{redsim}(\mathrm{PSC}(t_m, k + 1, \mathbf{A}))) \\ = \; & \mathrm{red}(\sigma(\mathrm{redsim}(\mathrm{PSC}(t_1, k + 1, \mathbf{A})), \dots, \mathrm{redsim}(\mathrm{PSC}(t_m, k + 1, \mathbf{A})))) \\ = \; & \mathrm{redsim}(\mathrm{PSC}(t, k + 1, \mathbf{A})). \end{aligned}$$

The second equality follows from Lemma 3.16. From Lemma 3.10 it follows that $\mathrm{redsim}(\mathrm{PSC}(t, k + 1, \mathbf{A})) \in B'$ iff $t \in L(\mathbf{A})[k]$. Thus $L(\mathbf{B}) = L(\mathbf{A})[k]$. Q.E.D.

4 The logarithmic bound

In this section we show that a logarithmic alternation depth bound defines a family of forests strictly larger than the regular forests. We denote by log the function $n \to \lceil \log_2(n) \rceil$ where $\lceil \log_2(n) \rceil$ is the smallest positive integer not less than the 2-based logarithm of n.

Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ where $\Sigma_0 = \{\gamma\}, \Sigma_1 = \{\sigma\}$ and $\Sigma_2 = \{\omega\}$, and denote $\Omega = \Sigma_0 \cup \Sigma_2$. Define the tree homomorphism $h : F_\Omega \to F_\Sigma$ by the following:

$$h_0(\gamma) = \gamma \text{ and } h_2(\omega) = \sigma(\omega(x_1, x_2)).$$

The Σ-tree $h(t)$ is obtained from an Ω-tree t simply by attaching above every ω-node of t a node labeled by the unary symbol σ.

Lemma 4.1 *Let* Σ, Ω *and* h *be as above and denote*

$$L = \{h(r) | r \in F_\Omega \text{ and } r \text{ is balanced}\}.$$

Then $L \in L(ATR)[2\log]$.

Proof. Clearly the set $h(F_\Omega)$ is regular. Hence by Lemma 3.4 it is sufficient to construct a recognizer $\mathbf{A} = (\Sigma, A, A', G) \in$ ATR such that

$$\text{for every } t \in h(F_\Omega) : t \in L(\mathbf{A}) \text{ if and only if } t \text{ is balanced,} \tag{10}$$

and for every balanced tree $h(r), r \in F_\Omega$,

$$\text{COM}(\mathbf{A}, h(r))[2\log] \cap \text{ ACOM}(\mathbf{A}, h(r)) \neq \emptyset. \tag{11}$$

That is, we can assume that the inputs are of the form $h(r), r \in F_\Omega$. Choose

$$\begin{aligned} A &= \{c_i, d_i, e_i, f_i, g_i | i = 1, 2, 3\}, \text{ and} \\ A' &= A - \{d_1, d_2, d_3\}. \end{aligned}$$

The state-transition relation $G = E \cup U$ is defined by the following. Below addition is always performed modulo three.

$$\gamma_E = \{c_1\}; \tag{12}$$

$$\omega_U(c_i, c_i) = \{e_i, g_i\}, i = 1, 2, 3; \tag{13}$$

$$\sigma_E(e_i) = \{c_{i+1}, d_i\}, i = 1, 2, 3; \tag{14}$$

$$\sigma_E(x) = \{x\} \text{ if } x \in \{d_1, d_2, d_3, f_1, f_2, f_3, g_1, g_2, g_3\}; \tag{15}$$

$$\omega_E(g_i, g_i) = \{g_i\}, i = 1, 2, 3; \tag{16}$$

$$\omega_E(d_i, d_i) = \{d_i\}, i = 1, 2, 3; \tag{17}$$

$$\omega_E(x, y) = \{f_i\} \text{ if } \{x, y\} = \{d_i, g_i\}, i \in \{1, 2, 3\}; \tag{18}$$

$$\omega_E(x,y) = \{f_i\} \text{ if } x,y \in \{d_i, g_i, f_i\} \text{ and } f_i \in \{x,y\}, i \in \{1,2,3\}. \tag{19}$$

The state-transition relation is undefined in all other cases. We say that a configuration $K \in F_\Sigma(A)$ is well formed if $K = h(r)$ for some ΩA-tree r. (The tree homomorphism h is extended to $F_\Omega(A)$ by setting $h(a) = a$ for all $a \in A$.) Let $K \in F_\Sigma(\{a_1, \ldots, a_n\})$ be such that K does not contain the nullary symbol γ, and $a_1, \ldots, a_n \in A$. If each element $a_i, i = 1, \ldots, n$, occurs at least once in K, it is called an $[a_1, \ldots, a_n]$-configuration. First we show that (11) holds.

Claim 1 *Let K_1 be a well formed balanced $[c_i]$-configuration, $i \in \{1,2,3\}$, and denote $m = \text{hg}(K_1)$. We claim that there exists $T \in$ ACOM$(\mathbb{A}, K_1)$ such that $alt(T) \leq m$.*

Proof of Claim 1. Since K_1 is well formed, each subtree of K_1 of height two is of the form $\sigma(\omega(c_i, c_i))$. In the computation of T the recogniser reads first all (universal) active subtrees $\omega(c_i, c_i)$ in arbitrary order using the rule (13). Thus one obtains one $[e_i]$-configuration $K(1)$, one $[g_i]$-configuration $K(2)$ and a number of $[e_i, g_i]$-configurations $K(3)$. In each configuration $K(3)$ the recogniser reads all active subtrees $\sigma(e_i)$ making the existential choice d_i, this results in a $[d_i, g_i]$-configuration $K(4)$. Since $K(4)$ contains both states d_i and g_i, it follows that the recogniser reaches the root of $K(4)$ in the accepting state f_i by the deterministic rules (15)-(19). Similarly the computation starting from $K(2)$ reaches the root in the state g_i using rules (15) and (16). Finally, in $K(1)$ the recogniser makes in each active subtree $\sigma(e_i)$ the existential choise c_{i+1}, which yields a $[c_{i+1}]$-configuration K_2. Furthermore K_2 is balanced because K_1 is balanced.

Above the computations starting from configurations $K(2)$, $K(3)$ and $K(4)$ are purely existential. Thus there exists a K_1-computation tree T_1 such that alt$(T_1) = 2$ and one leaf of T_1 is labeled by K_2 and all other leaves by elements of A'. Now hg$(K_2) = $ hg$(K_1) - 2$. By inductive reasoning it follows that T_1 can be completed to an accepting computation tree T, where alt$(T) = m = $ hg(K_1). (Since K_1 is well formed, m is even. The configuration $K_{(m/2+1)}$ will be of the form $c_j, j \in \{1,2,3\}$.) This concludes the proof of the claim.

Now let $t \in h(F_\Omega)$ be balanced. We construct $T \in$ COM$(\mathbb{A}, t)$ as follows. First the recogniser reads the leaves of t using the rule (11) yielding a $[c_1]$-configuration K_1. By Claim 1 there exists an accepting K_1-computation tree T_1 such that alt$(T_1) = $ hg$(K_1)(=$ hg$(t))$ where furthermore in each branch the first computation segment is universal. Thus T can be constructed so that

$$\text{alt}(T) = \text{hg}(t) + 1.$$

(The first computation segment corresponding to rules (12) is existential.) Since t is balanced, hg$(t) < 2\log(\text{size}(t))$ and (12) holds.

It remains to verify that also (10) holds. The "if" direction follows from (11). The intuitive idea of the proof in the "only if" direction is to show that in an accepting t-computation tree there necessarily exists a branch where the recogniser essentially reads the input in a layered fashion as in the proof of Claim 1 and thus checks that the input tree t is balanced. First we prove a number of claims. Denote

$$Q_i = \{c_i, d_i, g_i, f_i\}, i = 1,2,3.$$

Claim 2 *Let $K \in F_\Sigma(A)$ and assume that K contains elements of Q_i and Q_j, $i \neq j$. Then K is not accepting.*

Proof of Claim 2. Let $T \in \mathrm{COM}(\mathbf{A}, K)$ be arbitrary and let H label a node of T. If H contains an element of $Q_k, k \in \{1,2,3\}$, then one daughter of H in T also contains an element of Q_k. This follows immediately from the definition of the rules that read elements of Q_k, note that in rule (13) one can choose the daughter corresponding to g_k. Now since K contains elements of both Q_i and Q_j, the computation tree T contains a branch where each configuration has elements of Q_i and Q_j and this computation cannot terminate successfully.

Claim 3 *Let $K \in F_\Sigma(A)$ and assume that states e_i and c_{i+2} appear in K, ($i+2$ is computed modulo 3.) Then K is not accepting.*

Proof of Claim 3. We can assume that e_i appears in an active subtree $r_1 = \sigma(e_i)$ and c_{i+2} in an active subtree $r_2 = \omega(c_{i+2}, c_{i+2})$ because otherwise the computation is blocked already in the states in question. Let T be an arbitrary K-computation tree of $\mathbf{A}$. Assume that in T the recogniser reads r_1 before r_2. The existential rule (14) yields the state c_{i+1} or d_i which cannot appear together with c_{i+2} in an accepting configuration by Claim 2. Thus necessarily the recogniser reads first r_2 using the universal rule (13). Consider the branch of the computation corresponding to the state g_{i+2}. From rules (15), (16), (18) and (19) it follows that all configurations in this branch contain one of the states g_{i+2} or f_{i+2}, (g_{i+2} can only be deleted by changing it to f_{i+2} using rule (18) or (19).) So when the recogniser reads the active subtree r_1 at an arbitrary time in the computation, both existential choices c_{i+1} and d_i yield a configuration that is not accepting by Claim 2. Thus $T \notin \mathrm{ACOM}(\mathbf{A}, K)$.

Claim 4 *Denote $D = \{c_1, c_2, c_3, e_1, e_2, e_3, d_1, d_2, d_3\}$. Assume that all leaves of a configuration K are labeled by elements of D and K contains at least one element of $\{d_1, d_2, d_3\}$. Then K is not accepting.*

Proof of Claim 4. Let $T \in \mathrm{COM}(\mathbf{A}, K)$ be arbitrary. Consider the branch B of T that in universal computation steps (13) follows the choice e_i. All configurations in this branch have leaves labeled by elements of D and furthermore contain at least one element of $\{d_1, d_2, d_3\}$. This is because the elements d_i can be deleted only by rules (18) and (19) which do not become applicable as the configurations do not contain elements g_i or f_i. Thus B cannot end with an accepting final state.

Claim 5 *Let K be an $\mathbf{A}$-configuration with all leaves labeled by $e_i, i \in \{1,2,3\}$, and $hg(K) > 1$. Assume that T is an accepting K-computation tree of $\mathbf{A}$. Then T contains a configuration K_1 with all leaves labeled by e_{i+1}. Furthermore,*

$$K = K_1(e_{i+1} \leftarrow \omega(\sigma(e_i), \sigma(e_i))). \tag{20}$$

Proof of Claim 5. Necessarily the mother (immediate predecessor) of each node e_i is labeled by σ because otherwise the computation would be blocked in the state e_i. The computation of T first reads an arbitrary number of the active subtrees $\sigma(e_i)$ making the choice c_{i+1}. The existential choice d_i is prohibited by Claim 4. In the states c_{i+1} the recogniser can then apply only the universal rule (13). Consider the branch B of the computation that corresponds to universal choices e_{i+1}. Note that above the recogniser needs not read all subtrees $\sigma(e_i)$ before starting to read the subtrees $\omega(c_{i+1}, c_{i+1})$. However, before continuing the computation from the states e_{i+1} the recogniser must read all active subtrees $\sigma(e_i)$ and $\omega(c_{i+1}, c_{i+1})$. This is seen as follows.

The state e_{i+1} can only be read by rule (14) where by Claim 4 furthermore the recogniser needs to make the existential choice c_{i+2}. (The current configuration contains only states e_i, c_{i+1}, e_{i+1}.) By Claims 2 and 3, the state c_{i+2} cannot appear with c_{i+1} or e_i in an accepting configuration. Thus we can choose K_1 to be the configuration that appears in the branch B just before the first symbol e_{i+1} is read. Then all leaves of K_1 are labeled by e_{i+1} and also clearly (20) holds. The assumption $\mathrm{hg}(K) > 1$ prohibits the possibility that $K = \sigma(e_i)$.

Now we can proceed to prove that the "only if" side of (10) holds. Assume that $t \in L(\mathbf{A})$ and let $T \in \mathrm{COM}(\mathbf{A}, t)$ be accepting. Without restriction we can assume that in T the recognizer first reads all leaf symbols γ. Note that the rule (12) is deterministic so it commutes with all other rules. Thus one obtains a configuration $K = t(\gamma \leftarrow c_1)$. Consider the branch of T that corresponds to the universal choices e_1 in the computation steps (13) in the configuration K. (Note that (13) is the only computation step applicable in K.) In this branch of the computation the recognizer must read all states c_1 before reading any of the states e_1 by rule (14). (This is seen using Claims 2-4 exactly as in the proof of Claim 5.) Thus one obtains a configuration K_1 with all leaves labeled by e_1 such that

$$t = K_1(e_1 \leftarrow \omega(\gamma, \gamma)).$$

Denote by $[j]$ the smallest positive integer congruent to j modulo 3. By Claim 5, if $K_i \in \mathrm{conf}(T)$ is a configuration with all leaves labeled by $e_{[i]}$ and having height at least two, there exists a configuration $K_{i+1} \in \mathrm{conf}(T)$ with all leaves labeled by $e_{[i+1]}$ such that

$$K_i = K_{i+1}(e_{[i+1]} \leftarrow \omega(\sigma(e_{[i]}), \sigma(e_{[i]}))). \tag{21}$$

Denote $m = (\mathrm{hg}(K_1) + 1)/2$. Then $K_m = \sigma(e_{[m]})$. (Since $t \in h(F_\Omega)$, it is easy to see that the string of symbols labeling a path from a leaf of K_1 to the root always belongs to $(\sigma\omega)^*\sigma$. Hence $\mathrm{hg}(K_1)$ is odd and the last configuration in the chain defined by (21) is $\sigma(e_{[m]})$.) From (21) it follows that K_1 and hence t is balanced. Q.E.D.

In Lemma 4.1 the function $2\log$ can be reduced by an arbitrary constant factor. The construction in the proof is independent of the rank of the elements ω and by increasing $\mathrm{rank}(\omega)$ the number of distinct existential and universal segments in a computation on an input t can be made to be smaller than $C^{-1}\log(\mathrm{size}(t))$ for any natural number C.

Let $m \geq 2$ and define $\Gamma = \Gamma_0 \cup \Gamma_1 \cup \Gamma_m$, where $\Gamma_0 = \{\gamma\}, \Gamma_1 = \{\sigma\}$ and $\Gamma_m = \{\overline{\omega}\}$, i.e., Γ is as Σ in Lemma 4.1 except the binary symbol ω is replaced by $\overline{\omega}$ of rank m. Define $L(m)$ to be the Γ-forest that is obtained from the forest L of Lemma 4.1 by relabeling each ω-node with $\overline{\omega}$ and attaching for it $m-2$ additional copies of the subtrees. In other words, $L(m)$ consists of all balanced Γ-trees t such that the string of labels of each branch from the root of t to a leaf belongs to $(\sigma\overline{\omega})^*\gamma$. Define $\mathbf{A} = (\Gamma, A, A', G(m)) \in \mathrm{ATR}$ otherwise exactly as in Lemma 4.1 except the rules (13), (16), (17), (18) and (19) are replaced by the following:

$$\overline{\omega}_U(c_i, \ldots, c_i) = \{e_i, g_i\}, i = 1, 2, 3; \tag{13'}$$

$$\overline{\omega}_E(g_i, \ldots, g_i) = \{g_i\}, i = 1, 2, 3; \tag{16'}$$

$$\overline{\omega}_E(d_i, \ldots, d_i) = \{d_i\}, i = 1, 2, 3; \tag{17'}$$

$$\overline{\omega}_E(x_1, \ldots, x_m) = \{f_i\} \text{ if } \{x_1, \ldots, x_m\} = \{d_i, g_i\}, i \in \{1, 2, 3\}; \tag{18'}$$

$$\overline{\omega}_E(x_1, \ldots, x_m) = \{f_i\} \text{ if } x_1, \ldots, x_m \in \{d_i, g_i, f_i\}, \text{ and} \tag{19'}$$

$$f_i \in \{x_1, \ldots, x_m\}, i \in \{1, 2, 3\}.$$

Then exactly as in the proof of Lemma 4.1 it is seen that

$$L(\mathbb{A}) = L(m),$$

and furthermore for every $t \in L(m)$ there exists $T \in \mathrm{ACOM}(\mathbb{A}, t)$ such that $\mathrm{alt}(T) = \mathrm{hg}(t) + 1$. Let $C \in N_+$ be arbitrary and choose $m > 2^C$. Then for every $t \in L(m)$,

$$\mathrm{hg}(t) < (2/C)\log(\mathrm{size}(t)).$$

Denote by $C^{-1}\log$ the function $n \to \lceil C^{-1}\log_2(n)\rceil$. Thus we have:

Theorem 4.2 *For every $C \in N_+$:*

$$REG \subset L(ATR)[C^{-1}\log].$$

(Here $\subset$ denotes strict inclusion.)

5 Conclusions

Here we briefly discuss open questions and results on other types of alternation bounds. We have shown that

$$\mathrm{REG} = L(\mathrm{ATR})[k] \subset L(\mathrm{ATR})[C^{-1}\log]$$

for all constants k and C. A central open question is whether it is possible to separate $L(\mathrm{ATR})[\theta]$ from REG for some sublogarithmic function θ. Also we do not know whether $L(\mathrm{ATR})[\log] \subset L(\mathrm{ATR})$. We conjecture that the simple forest $L = \{\omega(\sigma^n(\gamma), \sigma^n(\gamma)) | n > 0\}$ does not belong to $L(\mathrm{ATR})[\log]$ but do not have a proof for this. It is easy to see that $L \in L(\mathrm{ATR})$, cf. [14], [15].

One can restrict the computation trees of an alternating recognizer in many different ways. A natural variant of Definition 3.3 would be to require that the number of distinct existential and universal computation segments corresponding to any given path from a leaf to the root in the input tree is bounded by some function θ. Similarly as in Definitions 3.2 and 3.3, in every branch of a computation tree one can associate a word w over $\{E, U\}$ to the computation steps performed on a given path from a leaf to the root in the input tree t. Then one requires that for all such words w, $\mathrm{alt}(w)$ is at most $\theta(\mathrm{size}(t))$. With this definition it is not difficult to see that already a constant bound (in fact even the constant 2) allows the alternating automata to recognize forests that are not regular. The detailed construction is omitted here. Note that since the computations in independent subtrees can be performed in arbitrary order, a t-computation tree may have $O(\mathrm{size}(t))$ computation segments (in the sense of Definition 3.2) even if the computation on any fixed path of t has only 2 segments.

Also one can restrict the width of the computation trees or, equivalently, the number of universal computation steps analogously with the bounds on parallelism considered in [5], [7]. Let θ be a function on the natural numbers, $\mathbb{A} = (\Sigma, A, A', G) \in \mathrm{ATR}$ and T be a computation tree of $\mathbb{A}$. We denote by $\#T$ the number of leaves of T. We say that the recogniser $\mathbb{A}$ accepts a Σ-forest L with the width-bound θ if $L = L(\mathbb{A})$ and for every $t \in L$ there exists $T \in \mathrm{ACOM}(\mathbb{A}, t)$ such that $\#T \leq \theta(\mathrm{size}(t))$. This is denoted $L = L(\mathbb{A})[\theta]_w$. The family of forests recognized with the width bound θ is denoted $L(\mathrm{ATR})[\theta]_w$. As a corollary of Theorem 3. 17 we have:

Theorem 5.1 $L(\mathrm{ATR})[k]_w = \mathrm{REG}$. *(Again we denote the constant function $c(k)$ simply by k.)*

Proof. Suppose that $L = L(\mathbf{A})[k]_w$, $\mathbf{A} = (\Sigma, A, A', G)$. Without restriction we can assume that if f is an active subtree of type Z of $\mathbf{A}$, $Z \in \{E, U\}$, and f_Z consists of only one element of A, then f is existential, i.e., $Z = E$. (A suitable modification of the relation G does not change the number of leaves of any computation tree.) Thus for every computation tree T of $\mathbf{A}$ we have

$$\mathrm{alt}(T) \leq 2(\#T) - 1.$$

($\#T$ is at least the number of universal computation steps in T plus one.) Thus $L = L(\mathbf{A})[2k-1]$ and L is regular by Theorem 3.17. Q.E.D.

Also the question whether $L(\mathrm{ATR})[\log]_w$ contains nonregular forests remains open. Note that this does not follows from the results of the previous section because in the construction of Lemma 4.1 the recognizer uses $O(\mathrm{size}(t))$ universal computation steps on an input t.

References

[1] G. Buntrock and A. Hoene, Reversals and alternation, Proc. of 6th STACS, Lect. Notes Comput. Sci. 349 (1989) 218-228.

[2] A.K. Chandra, D.C. Kozen and L.J. Stockmeyer, Alternation, J. Assoc. Comput. Mach. 28 (1981) 114-133.

[3] E.M. Gurari and O.H. Ibarra, (Semi)Alternating stack automata, Math. Systems Theory 15 (1982) 211-224.

[4] F. Gécseg and M. Steinby, Tree automata, Akadémiai Kiadó, Budapest, 1984.

[5] J. Hromkovic, Tradeoffs for language recognition on parallel computing models, Proc. of 13th ICALP, Lect. Notes Comput. Sci. 226 (1986) 157-166.

[6] J. Hromkovic, On the power of alternation in automata theory, J. Comput. System Sci. 31 (1985) 28-39.

[7] K.N. King, Measures of parallelism in alternating computation trees, Proc. of 13th Ann. ACM Symp. on Theory of Computing (1981) 189-201.

[8] K.N. King, Alternating multihead finite automata, Theoret. Comput. Sci. 61 (1988) 149-174.

[9] R.E. Ladner, R.J. Lipton and L.J. Stockmeyer, Alternating pushdown and stack automata, SIAM J. Comput. 13 (1984) 135-155.

[10] H. Matsuno, K. Inoue, H. Taniguchi and I. Takanami, Alternating simple multihead finite automata, Theoret. Comput. Sci. 36 (1985) 291-308.

[11] D.E. Muller, A. Saoudi and P.E. Schupp, Alternating automata, the weak monadic theory of the tree, and its complexity, Proc. of 13th ICALP, Lect. Notes Comput. Sci. 226 (1986) 275-283.

[12] D.E. Muller and P.E. Schupp, Alternating automata on infinite trees, Theoret. Comput. Sci. 54 (1987) 267-276.

[13] W.L. Ruzzo, Tree-size bounded alternation, J. Comput. System Sci. 21 (1980) 218-235.

[14] K. Salomaa, Alternating bottom-up tree recognizers, Proc. of 11th CAAP, Lect. Notes Comput. Sci. 214 (1986) 158-171.

[15] K. Salomaa, Yield-languages recognized by alternating tree recognizers, RAIRO Inform. Théor. 22 (1988) 319-339.

[16] K. Salomaa, Alternating tree pushdown automata, Ann. Univ. Turku Ser. AI 192 (1988).

[17] K. Salomaa, Representation of recursively enumerable languages using alternating finite tree recognizers, Proc. of 7th FCT, Lect. Notes Comput. Sci. 380 (1989) 372-383.

[18] G. Slutzki, Alternating tree automata, Proc. of 8th CAAP, Lect. Notes Comput. Sci. 159 (1983) 392-404.

Received June 20, 1991.