Deciding low levels of tree-automata hierarchy

Igor Walukiewicz

LaBRI
Bordeaux University
Domaine Universitaire, bâtiment A30, 351 cours de la Libration
33405 Talence Cedex
France

igw@labri.fr

Abstract

The paper discusses the hierarchy of indices of finite automata over infinite objects. This hierarchy corresponds exactly to the hierarchy of alternations of least and greatest fixpoints in the mu-calculus. It is also connected to quantifier hierarchies in monadic second-order logic. The open question is to find a procedure that given a regular tree language decides its level in the index hierarchy. Here, decision procedures are presented for low levels of the hierarchy. It is shown that these procedures have optimal complexity.

1 Introduction

Finite state automata running in infinite time constitute a fundamental model in the theory of verification of concurrent systems. One complexity measure obviously suggested by this model is the number of states, but more subtle criteria refer to the behavior of automaton and are specified in terms of positive and negative constraints on events which occur infinitely often. The depth of nesting of positive and negative conditions is reflected in the concept of the *index of an automaton*. Interestingly, the hierarchy of indices has a counterpart in the hierarchy of alternations of the least and greatest fixed points in the μ-calculus and quantifier hierarchies in monadic second-order logic.

Wagner [20], as early as in 1977, established the strictness of the hierarchy of indices for deterministic automata on infinite words. An analogous hierarchy for nondeterministic automata is easily seen to collapse to the level of Büchi automata. That is, nondeterminism can help to reduce the complexity of the acceptance condition reflected by the index of an automaton. The situation turns out to be different for automata on infinite trees. The power of such automata has been recognized since the seminal paper by Rabin [16]. The strictness of the hierarchy for both deterministic and nondeterministic

automata on infinite trees was proved in 1986 [10]. About the same time, Muller and Schupp introduced alternating tree automata [9]. The hierarchy problem for the μ-calculus (or equivalently alternating automata on trees of arbitrary branching) was solved ten years later by Bradfield [4]. It was then refined to the case of binary trees [5]. At the same time Arnold [1] gave a very beautiful proof of this result based on a diagonal argument and Banach fix-point theorem.

Once the hierarchy problems are resolved, the next challenge can be to provide algorithms for determining the level in the hierarchy of a given recognizable language. For word automata, polynomial–time algorithms for computing the index of an automaton presented by Muller or parity condition were given in [21] and [12], respectively. [1] For tree automata not much is known. Urbański [19] showed that it is decidable if a *deterministic* Rabin tree automaton is equivalent to a nondeterministic Büchi one. A slightly different approach [13] gives a PTIME algorithm for the problem. Otto [14] has shown that it is decidable if a μ-calculus formula is equivalent to a formula without fixpoints. This question is the same as asking whether a given alternating automaton is equivalent to a weak automaton of a very restricted shape.

In this paper we give a unified presentation of the above mentioned results together with some new results on the lowest levels of the hierarchy of nondeterministic and alternating automata. We show how to decide if a nondeterministic (or alternating) automaton recognizes a $(0,0)$ or a $(1,1)$ level language. We also provide the optimal complexity bounds for the problem. Büchi automata correspond to level $(0,1)$ in the hierarchy of indices. At present it is not known how to decide $(0,1)$ level for nondeterministic automata. It is also not known how to decide higher index levels for deterministic tree automata. In this paper we consider binary trees. The extension to trees of arbitrary degree seems possible, but is out of scope of this short article. Independently, the result about $(0,0)$ and $(1,1)$ levels was developed by Küsters and Wilke [8]. Their proof is different and they also work out the case of trees of arbitrary degree.

In the next section we introduce automata on words and trees. We also define the index hierarchy. In Section 3 we give an overview of the procedure for deciding the level of a language in the index hierarchy for deterministic word automata. Then we turn into nondeterministic tree automata. In Section 4 we show an EXPTIME lower bound on the complexity of deciding a level of a language in the index hierarchy of nondeterministic automata on trees. In the next section we give an EXPTIME algorithm deciding if a language given by an alternating automaton is on $(0,0)$ level of the hierarchy. By duality we get also an algorithm for $(1,1)$ level. After this we briefly sketch the procedure for deciding if a language given by a deterministic automaton is on $(0,1)$ level. The details of this procedure can be found in [13].

[1] Another proof of the result stated as Corollary 15 in [12] appeared later in [6].

2 Preliminaries

Automata on infinite words.

An infinite word over a finite alphabet Σ is a function $u : \mathbb{N} \to \Sigma$. By Σ^ω we denote the set of all infinite words over Σ. A *nondeterministic parity automaton* on Σ^ω is a tuple:

$$\mathcal{A} = \langle Q, \Sigma, q_I, \delta \subseteq Q \times \Sigma \to \mathcal{P}(Q), \Omega : Q \to \mathbb{N} \rangle$$

where Q is a finite set of *states* with an *initial state* q_I, δ is the *transition function*, and $\Omega : Q \to \mathbb{N}$ is the *rank* function. A *deterministic parity automaton* is a nondeterministic automaton such that $\delta(q, a)$ is a singleton or an empty set, for every $(q, a) \in Q \times \Sigma$.

A *run* of an automaton $\mathcal{A}$ on an infinite word $u \in \Sigma^\omega$ can be presented as an infinite word $\rho \in Q^\omega$ such that $\rho(0) = q_I$, and $\rho(m+1) \in \delta(\rho(m), u(m))$, for every $m \in \mathbb{N}$. The run ρ is *accepting* if $\liminf_{n\to\infty} \Omega(\rho(n))$ is *even*; in other words, the smallest rank repeating infinitely often is even. The language $L(\mathcal{A})$ recognized by $\mathcal{A}$ consists of those words in Σ^ω for which there exists an accepting run. A language $L \subseteq \Sigma^\omega$ is *recognizable* if it is recognized by a nondeterministic parity automaton.

Automata on infinite trees

A full binary infinite tree over an alphabet Σ is a function $t : \{0, 1\}^* \to \Sigma$. We write $\mathit{Trees}(\Sigma)$ for the set of all Σ labelled trees. We write $w0$ for the word w extended with 0, similarly for $w1$. We can think of $w0$ as the left son of w and of $w1$ as the right son of w. For $M \in \mathbb{N}$ we write $t|_M$ for a finite tree that is a restriction of t to the nodes of depth at most M.

A *nondeterministic parity automaton* on $\mathit{Trees}(\Sigma)$ is a tuple:

$$\mathcal{A} = \langle Q, \Sigma, q_I, \delta \subseteq Q \times \Sigma \to \mathcal{P}(Q \times Q), \Omega : Q \to \mathbb{N} \rangle$$

where the only difference with automata on words is in the type of transition function. A *run* of $\mathcal{A}$ on a tree $t \in \mathit{Trees}(\Sigma)$ is itself a Q–valued tree $r : \{0, 1\}^* \to Q$ such that $r(\varepsilon) = q_I$, and, for each $w \in \{0, 1\}^*$, we have $(r(w), a, r(w0), r(w1)) \in \delta$, whenever $t(w) = a$. A *path* in r is *accepting* if the smallest rank occurring infinitely often along it is even. More formally, for a path $P = p_0 p_1 \ldots \in \{0, 1\}^\omega$, this means that $\liminf_{n\to\infty} \Omega(r(p_0 p_1 \ldots p_n))$ is even. A *run is accepting* if so are all its paths. The tree language $L(\mathcal{A})$ *recognized* by $\mathcal{A}$ consists of those trees in $\mathit{Trees}(\Sigma)$ that admit an accepting run. We call a tree language $L \subseteq T_\Sigma$ *regular* if it is recognized by a nondeterministic parity tree automaton.

Alternating automata

It is the easiest to define alternating automata with the help of parity games. So we start with a short presentation of games of this kind.

A *parity game* $G = \langle V, V_0, V_1, E \subseteq V \times V, \Omega : V \to \{0, \ldots, d\}\rangle$ is a bipartite labelled graph with the partition (V_0, V_1) of the set of vertices V. We say that a vertex v' is a *successor* of a vertex v if $E(v, v')$ holds. We do not require that V is finite but we require that the set of ranks assigned to the vertices is finite, i.e., the range of Ω is finite.

A *play* from some vertex $v_0 \in V_0$ proceeds as follows: first player 0 chooses a successor v_1 of v_0, then player 1 chooses a successor v_2 of v_1, and so on ad infinitum unless one of the players cannot make a move. If a player cannot make a move he looses. The result of an infinite play is an infinite path $v_0, v_1, v_2, \ldots$ This *path is winning* for player 0 if the sequence $\Omega(v_0), \Omega(v_1), \ldots$ satisfies the parity condition. The play from vertices of V_1 is defined similarly but this time player 1 starts.

A *strategy* σ for player 0 is a function assigning to every sequence of vertices $\boldsymbol{v}$ ending in a vertex from V_0 a successor vertex $\sigma(\boldsymbol{v}) \in V_1$. A strategy is *memoryless* iff $\sigma(\boldsymbol{v}) = \sigma(\boldsymbol{w})$ whenever $\boldsymbol{v}$ and $\boldsymbol{w}$ end in the same vertex. A *strategy is winning* iff it guarantees a win for player 0 whenever he follows the strategy. Similarly we define a strategy for player 1.

An *alternating tree automaton* is a tuple:

$$\mathcal{A} = \langle Q, Q_\exists, Q_\forall, \Sigma, q^0, \delta : Q \times \Sigma \to \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega\rangle$$

There are two differences with respect to nondeterministic automata. First, the set Q of states is partitioned into existential and universal states, $Q_\exists$ and $Q_\forall$ respectively. Next, the transition function has different type. To execute a transition $(q', d) \in Q \times \{0, 1, \varepsilon\}$ in a vertex v means to go to the vertex vd and change the state to q'. So, if $d = \varepsilon$ then the automaton stays in v, if $d = 0$ then it moves to the left son of v. The idea is that if the automaton is in an existential state q and in a vertex labelled by a then it chooses a transition from $\delta(q, a)$ which it is going to execute. If q is universal then the choice is made by the opponent; this is equivalent to saying that the automaton has to execute all the transitions from $\delta(q, a)$.

It is the simplest to formalize the notion of a run and an acceptance of an alternating automaton $\mathcal{A}$ in terms of games. Given a tree t we define the *acceptance game* $G_{\mathcal{A},t}$:

- the set V_0 of vertices for player 0 is $\{0, 1\}^* \times Q_\exists$,
- the set V_1 of vertices for player 1 is $\{0, 1\}^* \times Q_\forall$,
- from each vertex (v, q) and $(q', d) \in \delta(q, t(v))$ there is an edge to (vd, q').
- the acceptance condition is given by $\Omega(v, q) = \Omega(q)$

We say that $\mathcal{A}$ *accepts* a tree t iff player 0 has a winning strategy in the game $G_{\mathcal{A},t}$. The *language recognized by* $\mathcal{A}$ is the set of trees accepted by $\mathcal{A}$.

Alternating automata on words are defined similarly but since there is only one successor of each position then the transition function has the type: $\delta : Q \times \Sigma \to \mathcal{P}(Q \times \{0, \varepsilon\})$.

Hierarchy of Mostowski indices.

The *Mostowski index* of an automaton $\mathcal{A}$ with the acceptance condition given by Ω is the pair $(\min(\Omega(Q)), \max(\Omega(Q)))$. We may assume without a loss of generality that $\min(\Omega(Q)) \in \{0, 1\}$. (Otherwise we can scale down the rank by $\Omega(q) := \Omega(q) - 2$.) Therefore, for any type of automata, the Mostowski indices induce a hierarchy depicted in Figure 1.

For nondeterministic $(1, 1)$ automata it is necessary to assume that there is a special state $\top$ from which every tree is accepted. This assumption is not need for automata with other indices as the language of all trees can be accepted by $(0, 0)$ automaton. The assumption is also not needed for $(1, 1)$ alternating automaton as the language of all trees is accepted from an universal state for which the transition function gives the empty set of moves.

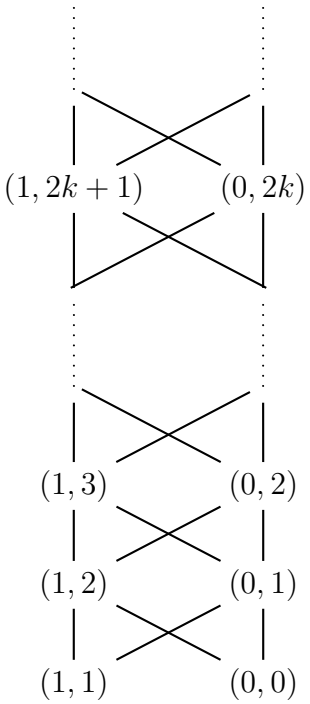

Fig. 1. Hierarchy of Mostowski indices

Automata of index $(0, 1)$ are traditionally called *Büchi automata* and presented by $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$, where F is the set of states of rank 0 (called *accepting states*). Note that a path in a run of a Büchi automaton is accepting if some accepting state occurs infinitely often.

A hierarchy is *strict* if there is an automaton at each level that cannot be simulated by any automaton of any lower level. As it was mentioned in the introduction, the hierarchy is known to be strict for deterministic automata on words [20], and for all kinds of automata on infinite trees. In contrast, for nondeterministic word automata the hierarchy collapses to the level $(0, 1)$ level (Büchi automata) [15], and for the alternating automata even to the intersection of levels $(0, 1)$ and $(1, 2)$ [2].

Connections to the mu-calculus

There is a very close connection between the index hierarchies and the hierarchy in the μ-calculus. In the later we measure the complexity of a formula by counting the number of alternations between the least, μ, and the greatest, ν, fix-point operators. So Σ_1^μ is the set of formulas only with μ, Π_1^μ are the formulas only with ν and Σ_2^μ are the μ-closures of Π_1^μ formulas. For a formal definition of the hierarchy we refer the reader to [11,3].

A language defined by a μ-calculus formula is a set of trees where the formula holds in the root. We have [3]:

Theorem 2.1 *A language of binary trees is definable by a Σ_n^μ formula iff it is the language of some alternating automaton of index $(1, n)$. Similarly for Π_n^μ and $(0, n-1)$ automata.*

In this paper we are also interested in the indices of nondeterministic tree automata. These are different than the indices of alternating automata mentioned in the above theorem. Still for the small levels that we consider here, the two are the same [2].

Theorem 2.2 *For any of the indices $(0, 0)$, $(1, 1)$ and $(0, 1)$: if a tree language is recognized by an alternating automaton with one of these indices then it is recognized by a nondeterministic automaton with the same index.*

The theorem is not true for index $(1, 2)$ or any bigger index.

Theorem 2.1 gives a connection between formulas with fixpoints and automata. We will be also interested in the formulas of the modal logic, i.e., the formulas with no fix-point at all. These correspond to *strict tree automata*. A strict tree automaton is a tree automaton with a partial order $\leq$ on states, and such that all possible transitions from a state q lead to states strictly smaller than q. The following easy fact makes a desired connection.

Fact 2.3 *A language of binary trees is definable by a modal formula iff it is the language of some strict automaton.*

3 Deciding hierarchies for words

As we have mentioned above the Mostowski hierarchy over words is infinite only for deterministic automata. In this section we shortly summarize the results from [12] showing how to calculate the deterministic index of a given language.

To see the examples of the strictness of the hierarchy consider for each $n \in \mathbb{N}$ an alphabet $\Sigma_n = \{1, \ldots, n\}$. Then we define the languages:

$$M_n = \{w \in \Sigma_n^\omega : \liminf_{n\to\infty} w(n) \text{ is even}\}$$
$$N_n = \{w \in \Sigma_n^\omega : \liminf_{n\to\infty} w(n) \text{ is odd}\}$$

So, M_n contains words where the smallest number appearing infinitely often

is even, and N_n contains words where this number is odd.

It is easy to see that M_n can be recognized by a $(1, n)$ deterministic automaton and N_n can be recognized by a $(0, n-1)$ deterministic automaton. The proof that there are no simpler automata for those languages, follows from a more general lemma presented below. It shows a connection between the Mostowski index of an ω-word language and the shape of a deterministic parity automaton recognizing the language. Roughly speaking, it says that in the graph of an automaton recognizing a "hard" language there must be a subgraph, called a flower, "witnessing" this hardness.

Definition 3.1 *Let $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \Omega \rangle$ be a deterministic parity automaton on words. The* graph of $\mathcal{A}$ *is the graph obtained by taking Q as the set of vertices and adding an edge from q to q' whenever $\langle q, a, q' \rangle \in \delta$, for some letter a.*

A path *in a graph is a sequence of vertices $v_1, \ldots, v_j$, such that, for every $i = 1, \ldots, j-1$ there is an edge from v_i to v_{i+1} in the graph. A* maximal strongly connected component *of a graph is a maximal subset of vertices of the graph, such that, for every two vertices v_1, v_2 in the subset there is a path from v_1 to v_2 and from v_2 to v_1.*

For an integer k, a k-loop *in $\mathcal{A}$ is a path $v_1, \ldots, v_j$ in the graph of $\mathcal{A}$ with $v_1 = v_j$, $j > 1$ and $k = \min\{\Omega(v_i) : i = 1, \ldots, j\}$. Observe that a k-loop must necessarily go through at least one edge.*

Given integers m and n, a state $q \in Q$ is a m-n-flower *in $\mathcal{A}$ if for every $k \in \{m, \ldots, n\}$ there is, in the graph of $\mathcal{A}$, a k-loop containing q.*

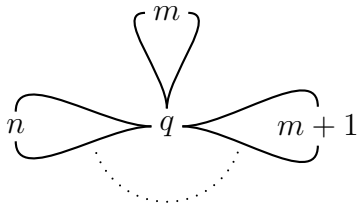

Fig. 2. m-n-flower

Definition 3.2 *We say that a language $L \subseteq \Sigma^\omega$* admits *an m-n-flower if there exists a deterministic Mostowski automaton $\mathcal{A}$, such that, $L = L(\mathcal{A})$ and $\mathcal{A}$ has an m-n-flower q for some q not a useless state in $\mathcal{A}$ (i.e. q occurring in some accepting run of $\mathcal{A}$).*

The delicate point about the above definition is that it talks about existence of a deterministic automaton for the language. Intuitively we are interested in the minimal automaton for the language, but for automata on infinite words the notion of minimality is not very convenient to work with. In particular there are languages with several different minimal deterministic automata recognizing them.

Lemma 3.3 (Flower Lemma) *For every $n \in \mathbb{N}$ and $L \subseteq \Sigma^\omega$: (1) if L is $(1, n+1)$-unfeasible then L admits a $2i$-$(2i+n)$-flower, for some i; (2) if L is $(0, n)$-unfeasible then L admits a $(2i+1)$-$(2i+1+n)$-flower, for some i.*

A priori we don't know how to find a deterministic automaton with a flower. Fortunately it turns out that it is enough to take any deterministic automaton for the language and then normalize it in some way. The resulting automaton is guaranteed to have as big flower as the index the language requires. This normalization procedure can be done in a quadratic time [12].

Corollary 3.4 *The problem of establishing the index of the language accepted by a deterministic automaton $\mathcal{A}$ with a Mostowski condition can be solved in time $\mathcal{O}(|\mathcal{A}|^2)$.*

4 The lower bound for nondeterministic tree automata

In this section we show that the hierarchy questions for nondeterministic tree automata are EXPTIME-hard.

Theorem 4.1 *For every $i \in \mathbb{N}$. The problem of deciding if a given nondeterministic automaton $\mathcal{A}$ accepts a $(1, i)$ language is EXPTIME-hard. Similarly for the $(0, i)$ class.*

Proof. We will reduce the universality problem for tree languages: given a nondeterministic automaton $\mathcal{A}$ decide if $L(\mathcal{A})$ accepts every tree (i.e. $L(\mathcal{A}) = \mathit{Trees}_\Sigma$). This problem is known to be EXPTIME-hard even for automata over finite trees [18].

Fix a language $L_{>i}$ not belonging to $(1, i)$ level of the hierarchy. Such a language exists as the hierarchy is infinite. For a given automaton $\mathcal{A}$ construct an automaton $\mathcal{B}$ that accepts a tree if either:

- the left subtree is accepted by $\mathcal{A}$ and the right subtree is arbitrary, or
- the left subtree is arbitrary and the right subtree is in $L_{>i}$.

This is schematically presented in Figure 3. Observe that the behaviour of $\mathcal{B}$ does not depend on the label of the root of a tree. We claim that $L(\mathcal{B})$ is on $(1, i)$ level iff $L(\mathcal{A}) = \mathit{Trees}_\Sigma$.

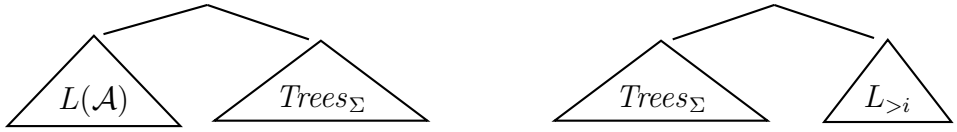

Fig. 3. The language $L(\mathcal{B})$

If $L(\mathcal{A}) = \mathit{Trees}_\Sigma$ then $L(\mathcal{B}) = \mathit{Trees}_\Sigma$, so $L(\mathcal{B})$ is a $(1, i)$ language as it is a $(1, 1)$ language.

If $L(\mathcal{A}) \neq \mathit{Trees}_\Sigma$ then we show that $L(\mathcal{B})$ cannot be a $(1, i)$ language. Suppose for a contradiction that $L(\mathcal{B})$ is recognized by a $(1, i)$ automaton $\mathcal{C}$.

Take a tree $t \notin L(\mathcal{A})$. If a tree from $L(\mathcal{B})$ has t as the left subtree then it has to have a tree from $L_{>i}$ in the right subtree. Let S_t be the set of states of $\mathcal{C}$ from which it accepts t. Let S_r be the set of possible right states when the left state is in S_t, i.e., $S_r = \{q_r : \exists_{a\in\Sigma, q_l\in S_t}\ (q_l, q_r) \in \delta(q_I^{\mathcal{C}}, a)\}$; here $q_I^{\mathcal{C}}$ is the initial state of $\mathcal{C}$. Let $\mathcal{C}(S_r)$ denote the automaton $\mathcal{C}$ where all states from S_r are initial states. Directly from the definitions we have that $L(\mathcal{C}(S_r)) = L_{>i}$. But this is impossible because $\mathcal{C}(S_r)$ is a $(1, i)$ automaton and $L_{>i}$ is not a $(1, i)$ language. □

5 The case of strict tree automata

In this section we present the decidability result for strict automata. This is a sub-level of both $(0, 0)$ and $(1, 1)$ levels. Actually this is precisely the intersection of the two levels. The interest in this level is mainly because of the connections to modal logic. From Fact 2.3 we know that this level is equivalent to definability in modal logic.

The main concept that we will need in the following is that of a type of a tree with respect to a given automaton.

Definition 5.1 *Fix an automaton $\mathcal{A}$. A* type *of a tree t is the set of states from which $\mathcal{A}$ accepts t: $Type_{\mathcal{A}}(t) = \{q \in Q : t \in L(\mathcal{A}(q))\}$. We will omit the subscript when the automaton is clear from the context. We will use $Types(\mathcal{A})$ for the set of all types of $\mathcal{A}$.*

The following simple lemma gives a useful characterization of languages recognized by strict automata. Recall that $t|_M$ denotes the restriction of t to nodes of depth at most M.

Lemma 5.2 *For every regular language L: L is recognizable by a strict automaton iff there is a bound M such that for every $t \in L$ and for every t', if $t|_M = t'|_M$ then $t' \in L$.*

Proof. Suppose L is recognized by a strict automaton $\mathcal{A}$. Such an automaton can look at the nodes of the tree at the depth at most equal to its size. So the size of the automaton gives an upper bound on M.

Conversely, suppose that L has a bound M. There are finitely many trees of depth M. So we can enumerate all depth M trees which are prefixes of trees in L. Then we construct a strict automaton recognizing all these prefixes. □

To test if a given automaton $\mathcal{A} = \langle Q^{\mathcal{A}}, \Sigma^{\mathcal{A}}, q_I^{\mathcal{A}}, \delta^{\mathcal{A}}, F^{\mathcal{A}}\rangle$ is equivalent to a strict automaton we construct an automaton $\mathcal{B}_{\mathcal{A}}$ on finite words over the alphabet $\Sigma^{\mathcal{B}} = \Sigma \times \{0, 1\}$:

$$\mathcal{B}_{\mathcal{A}} = \langle \mathcal{P}(Q^{\mathcal{A}}), \Sigma^{\mathcal{B}}, \{q_I^{\mathcal{A}}\}, \delta^{\mathcal{B}}, F^{\mathcal{B}}\rangle$$

where:

- $\delta^{\mathcal{B}}(S,(a,l)) = \{\{q_l : \exists_{q\in S}\exists_{q_r\in\tau}\ (q_l,q_r) \in \delta^{\mathcal{A}}(q,a)\} : \tau \in \mathit{Types}(\mathcal{A})\}$; and similarly for (a,r) letter.
- $F^{\mathcal{B}} = \{S : L(\mathcal{A}(S)) \neq \emptyset \wedge L(\mathcal{A}(S)) \neq \mathit{Trees}_\Sigma\}$

Intuitively, $\delta(S,(a,l))$ describes a situation when we want to accept from a state from S being in a node of the tree labelled by a and planing to check only the left subtree while assuming that the right subtree has some type. The transition function gives for each possible type to the right a set of all states from which we can accept the left subtree and have the whole tree accepted from some state in S. An element of the acceptance set F is a set of states from which $\mathcal{A}$ can accept some tree but from which it cannot accept all the trees.

Lemma 5.3 *$L(\mathcal{A})$ is not recognizable by a strict automaton iff $\mathcal{B}_\mathcal{A}$ accepts arbitrary long words.*

Proof. For the right to left direction we need to find for every bound M two trees t and t' such that $t|M = t'|M$ but $t \in L(\mathcal{A})$ and $t' \notin L(\mathcal{A})$. Consider a word $w = w_0,\ldots,w_M \in L(\mathcal{B}_\mathcal{A})$. Let $r_w = r_w(0)r_w(1)\ldots r_w(M+1)$ be an accepting run of $\mathcal{B}_\mathcal{A}$ on w.

For every suffix v of w we construct trees t_v and t'_v such that t_v is accepted from some state in $r(M+1-|v|)$ and t'_v is not accepted from any state in $r(M+1-|v|)$.

- If $v = \varepsilon$ is the empty suffix then for t_v we take a tree accepted from some state in $r_w(M+1)$ and for t'_v we take a tree not accepted from any state in $r_w(M+1)$.
- Suppose $v = w_i u$ with $w_i = (a_i, d_i)$. Consider the case when $d_i = l$, the other case is symmetric. Let τ_i be the type such that $r_w(i+1) = \{q_l : \exists q \in r_w(i).\ \exists q' \in \tau_i.\ (q_l,q') \in \delta(q,a)\}$. The tree t_v is the tree with the root labelled a_i with t_u as the left subtree and a tree of type τ_i as the right subtree. The tree t'_v is defined similarly but now it has t'_u as the left subtree.

By induction on the length of v one can show that t_v is accepted from some state in $r(M+1-|v|)$ and t'_v is not accepted from any state in $r(M+1-|v|)$. For the required t and t' we can then take t_w and t'_w.

For the left to right direction we need to show that if $L(\mathcal{A})$ is not recognizable by a strict automaton then $\mathcal{B}_\mathcal{A}$ accepts arbitrary long words. Given $M \in \mathbb{N}$ we are going to construct a word of length M recognized by $\mathcal{B}_\mathcal{A}$. From Lemma 5.2 we know that $L(\mathcal{A})$ does not have a bound. Let us then take trees $t \in L(\mathcal{A})$ and $t' \notin L(\mathcal{A})$ such that $t|M = t'|M$. Let $s_1,\ldots,s_k$ be the sequence of all the nodes on level M in t. One can think of t' as obtained from t by substituting some subtrees into some of these nodes. We construct a sequence of trees as follows:

- t_0 is t,
- t_{i+1} is obtained form t_i by substituting in s_i the subtree of t' rooted in s_i.

By the construction $t_k = t'$. Hence, there must be an index i such that $t_i \in L(\mathcal{A})$ and $t_{i+1} \notin L(\mathcal{A})$. So the change of a subtree in the node s_i prevents t_i from being accepted.

We use t_i and t_{i+1} to construct a word accepted by $\mathcal{B}_\mathcal{A}$. Let $u_0, \ldots, u_M$ be the path from the root to s_i. The word $w_0, \ldots, w_M$ is defined by $w_j = (a_j, d_j)$ where a_j is the label of the node u_j and d_j is the direction to the son which is on the path to s_i. It can be shown by induction that there is a run of $\mathcal{B}_\mathcal{A}$ on w that goes only only through accepting states. □

Lemma 5.4 *Automaton $\mathcal{B}_\mathcal{A}$ accepts arbitrary long words iff it accepts a word of length $> 2^{|\mathcal{A}|}$.*

Proof. The automaton $\mathcal{B}_\mathcal{A}$ has the property that from states not in F it does not accept anything. If $\mathcal{B}_\mathcal{A}$ accepts a path of length $> 2^{|\mathcal{A}|}$ then there is a cycle in the graph of $\mathcal{B}_\mathcal{A}$ staying in the states from F. This cycle can be used to produce arbitrary long words accepted by $\mathcal{B}_\mathcal{A}$. □

We obtain the following corollary that was originally shown in [14].

Corollary 5.5 *The problem of deciding whether a language of a given alternating automaton is definable by a strict automaton is EXPTIME-complete.*

6 The $(0,0)$ and $(1,1)$ cases

In this section we deal with the lowest levels of the hierarchy. We show that the problem of deciding membership in these levels is EXPTIME-complete.

Lemma 6.1 *If L is a $(1,1)$ language then for every $t \in L$ there is a bound M s.t. for every tree t' with $t|_M = t'|_M$ we have $t' \in L$*

Proof. Take a $(1,1)$ automaton for L. If this automaton accepts t then it looks only on a finite part of t. □

Remark 6.2 *The implication in the other direction also holds. That is, if L is regular and every tree in L has a bound then L is a $(1,1)$ language. This follows from the results below.*

Recall that *Types*$(\mathcal{A})$ stands for the set of types of the automaton $\mathcal{A}$. Directly from the definition it follows that the types of the sons of the root together with the label in the root determine the type of the root. We write $(\tau_0, \tau_1) \xrightarrow{b} \tau$ to mean that a node has type τ if it is labelled by b and the left and the right sons of a node have types τ_0 and τ_1 respectively.

Let $S \subseteq$ *Types*$(\mathcal{A})$ be a set of types of $\mathcal{A}$. We will use $L(S)$ to denote the set of the trees having one of the types in S. Suppose that $L(S)$ is a $(0,0)$ language. We are going to describe a direct construction of a $(0,0)$ automaton for $L(S)$.

Consider an automaton $\mathcal{C}_S = \langle \mathit{Types}(\mathcal{A}) \setminus \emptyset, \Sigma, S, \delta_c, \Omega_c \rangle$, where Ω_c assigns 0 to each state, and δ_c is defined by:

$$(\tau_0, \tau_1) \in \delta_c(\tau, a) \quad \text{iff} \quad (\tau_0, \tau_1) \xrightarrow{a} \tau$$

Observe that this automaton has a set of initial states.

Lemma 6.3 *If $L(S)$ is a $(0,0)$ language then $L(S) = L(\mathcal{C}_S)$.*

Proof. The inclusion $\subseteq$ is easy. Having a tree $t \in L(S)$ just assign to each node its type. This assignment is a run of $\mathcal{C}_S$. By definition every run of $\mathcal{C}_S$ is accepting.

For the other inclusion suppose conversely that there is a tree $t \in L(\mathcal{C}_S) \setminus L(S)$. Then $t \in L(\mathcal{C}_S) \cap \overline{L(S)}$, where $\overline{L(S)}$ is the complement of $L(S)$. By assumption $\overline{L(S)}$ is a $(1,1)$ language so, by Lemma 6.1, the tree t has a bound M. Take an accepting run $r : \{0,1\}^* \to \mathit{Types}(\mathcal{A})$ of $\mathcal{C}_S$ on t. Let $s_1, \ldots, s_k$ be all the nodes from t of depth $M+1$. Let t' be a tree where we substitute in each node s_i a tree of type $r(s_i)$. By definition of $\mathcal{C}_S$ we know that t' has one of the types in S. So $t' \in L(S)$. This is a contradiction as $t|M = t'|M$ and $t \in \overline{L(S)}$.

Theorem 6.4 *It is an EXPTIME-complete problem to decide if an alternating or a nondeterministic tree automaton accepts a $(0,0)$ language. Similarly for $(1,1)$ languages.*

Proof. The lower bound follows from Theorem 4.1. So it remains to show the upper bound. Given an automaton $\mathcal{A}$, let $S \subseteq \mathit{Types}(\mathcal{A})$ be the types containing the initial state. Hence, $L(\mathcal{A}) = L(S)$. It is enough to construct $\mathcal{C}_S$ and then check whether $L(\mathcal{A}) = L(\mathcal{C}_S)$. By Lemma 6.3 the equality holds iff $L(\mathcal{A})$ is a $(0,0)$ language.

By the definition of $L(\mathcal{C}_S)$ we know that $L(\mathcal{A}) \subseteq L(\mathcal{C}_S)$ always holds. So it is remains to check if $L(\mathcal{C}_S) \subseteq L(\mathcal{A})$. For this we check that $L(\mathcal{C}_S) \cap L(\overline{\mathcal{A}}) = \emptyset$ where $\overline{\mathcal{A}}$ is the automaton accepting the complement of $L(\mathcal{A})$. As $\mathcal{A}$ is an alternating automaton of size n, we can construct $\overline{\mathcal{A}}$ which is a nondeterministic automaton of size $\mathcal{O}(n!)$ and $\mathcal{O}(n)$ size acceptance condition. Automaton $\mathcal{C}_S$ has the size $\mathcal{O}(2^n)$ and trivial acceptance condition (every run is accepting). So the test $L(\mathcal{C}_S) \cap L(\overline{\mathcal{A}}) = \emptyset$ can be done in $2^{\mathcal{O}(n \log n)}$ time.

The case for $(1,1)$ level follows by duality. The complement of a $(1,1)$ language is a $(0,0)$ language. Hence, given an automaton $\mathcal{A}$ we can construct an alternating automaton $\overline{\mathcal{A}}$ for the complement an check whether it recognizes a $(0,0)$ language. The construction of $\overline{\mathcal{A}}$ can be done in linear time. □

7 The $(0,1)$ case

At present it is known how to decide $(0,1)$ level (Büchi level) only for deterministic tree languages. For such languages the lower bound from Theorem 4.1 does not hold. Actually, as we will see here, we can decide in PTIME if the

language of a given deterministic automaton is on $(0,1)$ level (provided we know that all states of the given automaton are productive.) The results presented here come from [13].

We start from a very useful characterization of deterministic tree languages in terms of paths in trees.

A *labeled path* in a tree $t : \{l,r\}^* \to \Sigma$ is an infinite sequence $\sigma_0 p_1 \sigma_1 p_2 \sigma_2 \ldots$, such that $\sigma_i \in \Sigma$, $p_i \in \{0,1\}$, and $t(p_1 \ldots p_i) = \sigma_i$ (so in particular $t(\varepsilon) = \sigma_0$). Note that a labeled path is an infinite word over an alphabet $\{0,1\} \cup \Sigma$. We let $\mathrm{Paths}(t)$ denote the set of all labeled paths in t, and, for a tree language L, $\mathrm{Paths}(L) = \bigcup_{t \in L} \mathrm{Paths}(t)$. For a word language $K \subseteq (\{0,1\} \cup \Sigma)^\omega$ we define a tree language:

$$\forall K = \{t \in T_\Sigma : \mathrm{Paths}(t) \subseteq K\}$$

Proposition 7.1 *The following conditions are equivalent for a tree language $L \subseteq T_\Sigma$.*

(i) *L is deterministically recognizable.*

(ii) *L is recognizable, and $L = \forall(\mathrm{Paths}(L))$.*

(iii) *$L = \forall K$, for some recognizable language K of infinite words.*

Take the graph of an automaton $\mathcal{A}$ on infinite words. We say that a state of $\mathcal{A}$ is correctly reachable if it is reachable on a word ending in a letter from Σ. We say that $\mathcal{A}$ admits a *split*[2] if, for some correctly reachable state q_0, there are two loops: $q_0 \xrightarrow{0} q_1 \xrightarrow{w} q_0$ and $q_0 \xrightarrow{1} q_2 \xrightarrow{v} q_0$, where w and v are some words in $\Sigma(\{0,1\}\Sigma)^*$, such that the highest ranks occurring on these loops are of different parity, and the smaller of the two is *even*.

Example: Let $\Sigma = \{a,b\}$ and let L be the set of all infinite words of the form $\sigma_0 p_1 \sigma_1 p_2 \sigma_2 p_3 \ldots$, with $\sigma_i \in \Sigma$ and $p_i \in \{0,1\}$, in which b occurs infinitely often. The language L can be recognized by a deterministic automaton with states q (initial), q_a, and q_b of ranks $\Omega(q) = \Omega(q_a) = 1$ and $\Omega(q_b) = 0$, and transitions $q \xrightarrow{a} q_a$, $q \xrightarrow{b} q_b$, and $q_a, q_b \xrightarrow{0,1} q$. This automaton has a split in state q_a. Rabin [17] showed that the set of trees whose all paths are outside L, i.e., on each path, b occurs only finitely often, cannot be recognized by a Büchi automaton. This fact can be generalized as follows.

Lemma 7.2 *A* deterministic *word automaton for $\overline{\mathrm{Paths}(L)}$ admits a split iff $\forall\,\mathrm{Paths}(L)$ cannot be recognized by a Büchi tree automaton.*

Hence to decide if a deterministic tree automaton $\mathcal{A}$ accepts a Büchi language we can proceed as follows. At first we convert $\mathcal{A}$ into a deterministic parity word automaton for $\mathrm{Paths}(L(\mathcal{A}))$. The construction is easy and does not increase the number of the automaton's states, however it requires knowing which states of $\mathcal{A}$ are productive. Once the automaton for $\mathrm{Paths}(L(\mathcal{A}))$ is constructed, we obtain a deterministic automaton for $\overline{\mathrm{Paths}(L(\mathcal{A}))}$ by simply

[2] This concept is similar to that of *gadget* used in [19].

scaling up the rank by 1. Now, it is easy to detect in polynomial time if a word automaton has a split. This gives:

Theorem 7.3 *It is decidable in polynomial time if a deterministically recognizable tree language (presented by a deterministic parity automaton without unproductive states) can be recognized by a Büchi automaton.*

Let us remark that checking if a state of an automaton is productive is as difficult for deterministic as for nondeterministic automata. For automata with parity conditions the problem is known to be in NP∩co-NP [7].

8 Conclusions

We have considered the lowest levels of index hierarchies for automata on binary trees. At present, for deterministic automata we can decide all levels up to $(0, 1)$. For nondeterministic and alternating automata we can decide $(0, 0)$ level, $(1, 1)$ level, and the intersection of the two. It would be also interesting to show these results for trees of arbitrary degree. For strict automata, or equivalently for the intersection of $(0, 0)$ and $(1, 1)$ levels, this was done by Otto [14]. We conjecture that a modification of the proofs presented here should give the results for $(0, 0)$ and $(1, 1)$ level. Independently Küsters and Wilke [8] have shown the same results for $(0, 0)$ and $(1, 1)$ levels; they have also worked out the extension to trees of arbitrary branching.

References

[1] A. Arnold. The mu-calculus alternation-depth hierarchy is strict on binary trees. *RAIRO–Theoretical Informatics and Applications*, 33:329–339, 1999.

[2] A. Arnold and D. Niwiński. Fixed point characterisation of weak monadic logic definable sets of trees. In M. Nivat and A. Podelski, editors, *Tree Automata and Languages*, pages 159–188. Elsevier, 1992.

[3] A. Arnold and D. Niwiski. *The Rudiments of the Mu-Calculus*, volume 146 of *Studies in Logic*. North-Holand, 2001.

[4] J. Bradfield. The modal mu-calculus alternation hierarchy is strict. *Theoretical Computer Science*, 195:133–153, 1997.

[5] J. Bradfield. Fixpoint alternation: Arithmetic, transition systems, and the binary tree. *RAIRO–Theoretical Informatics and Applications*, 33:341–356, 1999.

[6] O. Carton and R. Maceiras. Computing the rabin index of a parity automaton. *RAIRO-Theoretical Informatics and Applications*, 33:495–505, 1999.

[7] E. A. Emerson, C. Jutla, and A. Sistla. On model-checking for fragments of μ-calculus. In *CAV'93*, volume 697 of *LNCS*, pages 385–396, 1993.

[8] R. Ksters and T. Wilke. Deciding the first level of the mu-calculus alternation hierarchy. In *Foundations of Software Technology and Theoretical Computer Science: 22th Conference*, Lecture Notes in Computer Science, Kanpur, India, 2002. To appear.

[9] D. Muller and P. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[10] D. Niwiński. On fixed-point clones. In *Proc. 13th ICALP*, volume 226 of *LNCS*, pages 464–473, 1986.

[11] D. Niwiński. Fixed point characterization of infinite behaviour of finite state systems. *Theoretical Computer Science*, 189:1–69, 1997.

[12] D. Niwiński and I. Walukiewicz. Relating hierarchies of word and tree automata. In *STACS'98*, volume 1373 of *LNCS*. Springer-Verlag, 1998.

[13] D. Niwiński and I. Walukiewicz. A gap property of deterministic tree languages. To appear in TCS, 2002.

[14] M. Otto. Eliminating recursion in the mu-calculus. In *STACS'99*, volume 1563 of *LNCS*, pages 531–540, 1999.

[15] D. Park. Concurrency and automata on inifinte sequences. In *5th Gi Conference on Theoretical Computer Science*, volume 104 of *LNCS*, pages 167–183, 1981.

[16] M. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.

[17] M. Rabin. Weakly definable relations and special automata. In Y.Bar-Hillel, editor, *Mathematical Logic in Foundations of Set Theory*, pages 1–23. 1970.

[18] H. Seidl. Deciding equivalence of finite tree automata. *SIAM Journal of Computing*, 19:424–437, 1990.

[19] T. Urba"nski. On deciding if deterministic rabin language is in büchi class. In *ICALP'00*, LNCS, 2000. to appear.

[20] K. Wagner. Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen. *J. Inf. Process. Cybern. EIK*, 13:473–487, 1977.

[21] T. Wilke and H. Yoo. Computing the Rabin index of a regular language of infinite words. *Information and Computation*, 130(1):61–70, 1996.