On Deciding Topological Classes of Deterministic Tree Languages

Filip Murlak *

Institute of Informatics, Warsaw University
ul. Banacha 2, 02–097 Warszawa, Poland
fmurlak@mimuw.edu.pl

Abstract. It has been proved by Niwiński and Walukiewicz that a deterministic tree language is either Π^1_1-complete or it is on the level Π^0_3 of the Borel hierarchy, and that it can be decided effectively which of the two takes place. In this paper we show how to decide if the language recognized by a given deterministic tree automaton is on the Π^0_2, the Σ^0_2, or the Σ^0_3 level. Together with the previous results it gives a procedure calculating the exact position of a deterministic tree language in the topological hierarchy.

Keywords: deterministic tree automata, index hierarchy, Borel hierarchy

1 Introduction

Tree automata, introduced by Rabin [14] in order to prove decidability of second order monadic logic of two successors, are today – together with μ-calculus – the basic tool in modeling and verification of concurrent systems. A tree represents all possible behaviours of an analysed system and an automaton is a coded correctness condition. An interesting measure of complexity of such a condition is the nesting depth of positive and negative constraints on the events occurring infinitely often. The formalization of that criterion gives the notion of the index of an automaton. The languages recognized by automata of different indices constitute an ascending hierarchy. This hierarchy was proved to be strict for the classes of deterministic [19], nondeterministic [9], alternating [1, 7] and weak alternating automata [8].

Another approach to estimating the complexity of a language is to calculate its level in the topological hierarchy. Skurczyński [16] proved that the finite part of the Borel hierarchy is strict for languages recognized by weak alternating automata. Deterministic tree languages surprisingly turned out to be either Π^1_1-complete (hence non-Borel) or on the Π^0_3 level of the Borel hierarchy. The paper by Niwiński and Walukiewicz [11] containing the proof of the above suggests also that two basic complexity criteria, combinatorial and topological, are closely related – at least for deterministic automata.

* Supported by KBN Grant 4 T11C 042 25.

While the efficiency of verification methods depends on the brevity of the correctness conditions, they often result redundant when modeling real systems. Therefore, developing algorithmic methods for calculating actual automata's complexity would be interesting. So far, there have been presented procedures calculating the index of tree languages consisting of trees which have all the paths in a given regular ω-language [6, 10], deciding if a deterministic automaton is equivalent to a Büchi automaton [18], and calculating the (nondeterministic) index of deterministic automata [12]. The μ-calculus approach resulted in a procedure deciding if a given formula of modal μ-calculus is equivalent to a formula of modal logic [13].

In this paper we concentrate on algorithmic calculation of a language's position in the topological hierarchy and its connections with the deterministic index hierarchy. In Sect. 2 and Sect. 3 we remind the basic notions of automata theory and present simple criteria determining a language's position in the deterministic index hierarchy and weak deterministic index hierarchy. Section 4 recalls some necessary topological notions. In Sect. 5 we show how to decide if a deterministic language is in the classes Σ_2^0, Π_2^0 and Σ_3^0. When combined with the previous characteristics by Niwiński and Walukiewicz, it provides a complete procedure calculating the position of a deterministic language in the topological hierarchy.

2 Basic Notions

We shall use the symbol ω to denote the set of natural numbers $\{0, 1, 2, \ldots\}$. For an alphabet X, X^* is the set of finite words over X and X^ω is the set of infinite words over X. The concatenation of words $u \in X^*$ and $v \in X^* \cup X^\omega$ will be denoted by uv, and the empty word by ε. The concatenation is naturally generalized for infinite sequences of words $v_1 v_2 v_3 \ldots$. The concatenation of sets $A, B \subseteq X^*$ is $AB = \{uv : u \in A, v \in B\}$.

A *binary tree* is any subset of $\{0,1\}^*$ closed under the prefix relation. An element of a tree is usually called a *node*. A *leaf* is any node of a tree which is not a (strict) prefix of some other node. We shall be dealing mainly with *labeled trees* over Σ which are functions $t : \operatorname{dom} t \to \Sigma$ such that $\operatorname{dom} t$ is a tree. The symbol T_Σ will denote the set of full infinite binary trees over Σ, i. e. functions $\{0,1\}^* \to \Sigma$.

For any trees t, s and v, a node of t, the result of the *substitution* of v with s is a tree t' whose domain is the set $\operatorname{dom} t \cup v \operatorname{dom} s$ and

$$t'(u) = \begin{cases} s(u') \text{ if } u = vu' \text{ for some } u' \\ t(u) \quad \text{otherwise} \end{cases} .$$

Note that u' may be empty, so if $t(u) \neq s(\varepsilon)$, for the label of u in t' we choose $s(\varepsilon)$. We find it more convenient since the state of an automaton in a node depends on every predecessor of the node, but not on the node itself.

The concatenation of tree languages A, B is a tree language AB consisting of all trees obtained from any $t \in A$ by substituting every leaf u of t with any tree $s_u \in B$. The concatenation of infinite sequence of tree languages is a natural

generalization of the above. A more precise definition requires an auxiliary notion of a limit. Let $t_0, t_1, \ldots$ be a sequence of trees such that

- $\operatorname{dom} t_0 \subseteq \operatorname{dom} t_1 \subseteq \ldots$,
- $\forall v \in \bigcup_{m\in\omega} \operatorname{dom} t_m \;\; \exists n_v \;\; \forall n \geq n_v \;\; t_n(v) = t_{n_v}(v)$.

The *limit* $t = \lim t_n$ is defined as follows:

- $\operatorname{dom} t = \bigcup_{m\in\omega} \operatorname{dom} t_m$,
- $t(v) = t_{n_v}(v)$.

An *infinite concatenation* of tree languages $L_0 L_1 \ldots$ consists of the limits of all sequences $t_0, t_1, \ldots$ such that $t_0 \in L_0$ and $t_{n+1} \in \{t_n\} L_{n+1}$ for all n.

The concatenation of trees s, t is the only element of the concatenation $\{s\}\{t\}$. Similarly, the concatenation of infinite sequence of trees $t = t_1 t_2 t_3 \ldots$ is the only element of $\{t_1\}\{t_2\}\{t_3\}\ldots$.

For $v \in \operatorname{dom} t$ we define $t.v$ as a subtree of t rooted in v, i. e. $\operatorname{dom}(t.v) = \{u : vu \in \operatorname{dom} t\}$, $t.v(u) = t(vu)$. A *segment* of a tree t between u and uv is the restriction of the function $t.u$ to the set $\{w \in \operatorname{dom}(t.u) : v \text{ is not a strict prefix of } w\}$.

A *(nondeterministic) automaton on words* is a tuple $A = \langle \Sigma, Q, \delta, q_0, \operatorname{rank}\rangle$, where Σ is a (finite) input alphabet, Q is the set of states, $\delta \subseteq Q \times \Sigma \times Q$ is the relation of transition and $q_0 \in Q$ is the initial state. The meaning of the function $\operatorname{rank} : Q \to \omega$ will be explained later. Instead of $(q, \sigma, q_1) \in \delta$ one usually writes $q \xrightarrow{\sigma} q_1$. A *run* of an automaton A on a word $w \in \Sigma^\omega$ is a word $\rho_w \in Q^\omega$ such that $\rho_w(0) = q_0$ and if $\rho_w(n) = q$, $\rho_w(n+1) = q_1$, and $w(n) = \sigma$, then $q \xrightarrow{\sigma} q_1$. A run ρ_w is *accepting* if the highest rank repeating infinitely often in ρ_w is even; otherwise ρ_w is *rejecting*. A word is *accepted* by A if there exist an accepting run on it. The language of words accepted by A is denoted by $L(A)$. One says that L is *recognized* by A if $L = L(A)$. An automaton is *deterministic* if its relation of transition is a function $Q \times \Sigma \to Q$. Note, that we do not let the transition relation be a partial function, and so there is a run – accepting or not – on every word. We call a language deterministic if it is recognized by a deterministic automaton.

An *(nondeterministic) automaton on trees* is a tuple $A = \langle \Sigma, Q, \delta, q_0, \operatorname{rank}\rangle$, the only difference being that $\delta \subseteq Q \times \Sigma \times Q \times Q$. Like before, $q \xrightarrow{\sigma} q_1, q_2$ means $(q, \sigma, q_1, q_2) \in \delta$. A *run* of A on a tree $t \in T_\Sigma$ is a tree $\rho_t \in T_Q$ such that $\rho_t(\varepsilon) = q_0$ and if $\rho_t(v) = q$, $\rho_t(v0) = q_1$, $\rho_t(v1) = q_2$ and $t(v) = \sigma$, then $q \xrightarrow{\sigma} q_1, q_2$. A path π of the run ρ_t is *accepting* if the highest rank repeating infinitely often in π is even; otherwise π is *rejecting*. A run is called accepting if all its paths are accepting. If at least one of them is rejecting, so is the whole run. An automaton is called deterministic if its transition relation is a function $Q \times \Sigma \to Q \times Q$.

An automaton is called *weak* if the rank of visited states does not increase during the run, i. e. whenever there is a transition $p \xrightarrow{\sigma} q_1, q_2$, then $\operatorname{rank} p \geq \operatorname{rank} q_1$ and $\operatorname{rank} p \geq \operatorname{rank} q_2$.

The symbol G_A denotes a directed edge-labeled graph representing the transition relation of A. The set of vertices is Q and whenever in A there is a transition

$p \xrightarrow{\sigma} q_1, q_2$, in G_A there is an edge between p and q_1 labeled with $(\sigma, 0)$ and an edge between p and q_2 labeled with $(\sigma, 1)$. For sake of brevity we shall write $p \xrightarrow{\sigma,0} q_1$ and $p \xrightarrow{\sigma,1} q_2$. A state is called *productive* if it is used in some accepting run. The *productive graph* G_A^+ is analogous to G_A, only now the set of vertices is restricted to productive states and when defining the edges we demand that *all* of the states p, q_1, q_2 are productive. We shall call a path in G_A *productive* if it is also a path in G_A^+.

A *partial run* of A is a segment of any run of A. A partial run ρ *realizes* a finite path π in the graph G_A^+ if it is a segment of an accepting run ρ' between two nodes x and y such that ρ' agrees with π between x and y. More precisely, if $\pi = p_0 \xrightarrow{\sigma_1,d_1} \ldots \xrightarrow{\sigma_m,d_m} p_m$, then $y = xd_1d_1 \ldots d_m$, $\rho'(x) = p_0$ and $\rho'(xd_1 \ldots d_i) = p_i$ for all i. Note that, since ρ is a segment of an accepting run, all its infinite paths are accepting. A tree segment f *realizes* a path π if the corresponding partial run ρ_f realizes π.

When analysing the way an automaton works, one finds it useful to let the automaton begin its run in states other than initial. An automaton starting in the state q will be denoted by A_q.

The *index of an automaton* A is a pair $(\min \operatorname{rank} Q, \max \operatorname{rank} Q)$. Scaling down the rank function if necessary one may assume that $\min \operatorname{rank} Q \in \{0, 1\}$. For an index (i, j) we shall denote by $\overline{(i, j)}$ *the dual index*, i. e. $\overline{(0, j)} = (1, j+1)$, $\overline{(1, j)} = (0, j-1)$. *The index hierarchy* for a certain class of automata consists of (roughly speaking) ascending sets (levels) of languages recognized by (i, j)-automata, where $(i, j) \in \{0, 1\} \times \omega$. It is known that index hierarchies are strict for deterministic [19], nondeterministic [9], alternating [1, 7] and weak alternating automata [8].

3 Deterministic Index Hierarchies

Given a deterministic language, one may ask about its *deterministic index*, i. e. the exact position in the index hierarchy of deterministic automata. This question can be answered effectively. Here we follow the method introduced by Niwiński and Walukiewicz [10].

A sequence of loops $\lambda_i, \lambda_{i+1}, \ldots, \lambda_j$ in a graph of an automaton is called *an alternating chain* if the highest rank appearing on λ_k has the same parity as k and it is higher then the highest rank on λ_{k-1}. A (i, j)-*flower* is an alternating chain $\lambda_i, \lambda_{i+1}, \ldots, \lambda_j$ such that all loops start in the same state q. Let $\mathrm{Paths}(L) \subseteq \Sigma^\omega$ be the set of paths of trees from L and $\mathrm{Paths}'(L) \subseteq (\Sigma \times \{0, 1\})^\omega$ denote the language of generalized paths of L,

$$\mathrm{Paths}'(L) = \{\langle(\sigma_1, d_1), (\sigma_2, d_2), \ldots\rangle : \exists t \in L\, t(d_1 d_2 \ldots d_{i-1}) = \sigma_i\} \ .$$

Niwiński and Walukiewicz show that a language L is recognized by a $\underline{(i, j)}$-automaton iff no deterministic automaton recognizing $\mathrm{Paths}(L)$ contains a $\overline{(i, j)}$-flower. As an intermediate pass they prove the following fact.

Theorem 1 (Niwiński, Walukiewicz [10]). *A deterministic automaton on words is equivalent to a deterministic (i,j)-automaton iff it does not contain a $\overline{(i,j)}$-flower.*

For a deterministic tree automaton A, the graph G_A^+ can be treated as a deterministic automaton recognizing Paths'$(L(A))$. Conversely, given a deterministic word automaton recognizing Paths'$(L(A))$, one may interpret it as a graph of a tree automaton, obtaining thus a deterministic automaton recognizing $L(A)$. Hence, applying Theorem 1 one gets the following result.

Proposition 1. *For a deterministic tree automaton A the language $L(A)$ is recognized by a deterministic (i,j)-automaton iff G_A^+ does not contain a $\overline{(i,j)}$-flower.*

Similarly, one can calculate the exact position of a deterministic language in the hierarchy of weak deterministic automata. A *weak (i,j)-flower* is a sequence of loops $\lambda_i, \lambda_{i+1} \ldots, \lambda_j$ such that λ_k is reachable in G_A^+ from λ_{k+1}, and λ_k is accepting iff k is even. Intuitively, the notion is to provide long enough alternation of rank parity. Therefore we have to extend it to cover the case when i is odd and instead of λ_i there is an unproductive state r reachable in G_A from λ_{i+1}.

Proposition 2. *For any deterministic tree automaton A the language $L(A)$ can be recognized by a weak deterministic (i,j)-automaton iff G_A^+ does not contain a weak $\overline{(i,j)}$-flower.*

Proof. ($\Rightarrow$) Let us suppose that G_A^+ contains a weak (i',j')-flower, $(i',j') = \overline{(i,j)}$. Let $g_{j'}$ be a tree segment realizing some path from the initial state q_0 to a state $r_{j'}$ on $\lambda_{j'}$. By induction, let g_k realize a path from the state $r_{k+1} \in \lambda_{k+1}$ to a state $r_k \in \lambda_k$ for $k = j'-1, \ldots, i'$. Finally, let f_k realize the loop λ_k (from r_k to r_k) for all k. Let B be a weak deterministic automaton recognizing $L(A)$. Clearly, we can choose numbers $n_{i'}, \ldots, n_{j'}$ so that the run on $g_{j'}\,(f_{j'})^{n_{j'}} g_{j'-1}\,(f_{j'-1})^{n_{j'-1}} \ldots g_{i'}\,(f_{i'})^{n_{i'}}$ would need $j'-i'$ changes of rank parity and thus $j'-i'+1$ different ranks. Consequently, the index of B cannot be (i,j).

($\Leftarrow$) A weak deterministic (i,j)-automaton is obtained by setting $\mathrm{rank}\,(q)$ equal to the highest number m such that there exists a weak (ι,m)-flower with a path from q to λ_m (recall that an unproductive state is a weak $(1,1)$-flower). □

Finally, for a deterministic language one may want to calculate its *nondeterministic index*, i.e. the position in the hierarchy of nondeterministic automata. This may be lower than the deterministic index, due to greater expressive power of nondeterministic automata. Consider for example the language $L_M^{0^\omega}$ consisting of trees whose leftmost paths are in a regular ω-language M. It can be recognised by a (nondeterministic) Büchi automaton, but its deterministic index is equal to the deterministic index of M, which can be arbitrarily high.

The problem transpired to be rather difficult and has only just been solved in [12]. The analogous problem for nondeterministic languages remains open.

4 Topological Hierarchy

We start with a short recollection of elementary notions of descriptive set theory. For further information see [5].

Let 2^ω be the set of infinite binary sequences with a metric given by the formula

$$d(u,v) = \begin{cases} 2^{-\min\{i\in\omega\,:\,u_i\neq v_i\}} & \text{iff } u \neq v \\ 0 & \text{iff } u = v \end{cases}$$

and T_Σ be the set of infinite binary trees over Σ with a metric

$$d(s,t) = \begin{cases} 2^{-\min\{|x|\,:\,x\in\{0,1\}^*,\, s(x)\neq t(x)\}} & \text{iff } s \neq t \\ 0 & \text{iff } s = t \end{cases} .$$

Both 2^ω and T_Σ, with the topologies induced by the above metrics, are Polish spaces (complete metric spaces with countable dense subsets). In fact, both of them are homeomorphic to the Cantor discontinuum.

The class of Borel sets of a topological space X is the closure of the class of open sets of X by countable sums and complementation. Within this class one builds so called *Borel hierarchy*. The initial (finite) levels of it are defined as follows:

Σ^0_1 – open relations, i. e. open subsets of X^n for some $n < \omega$,
Π^0_k – complements of relations from Σ^0_k,
Σ^0_{k+1} – countable sums of relations form Π^0_k.

For example, Π^0_1 are closed relations, Σ^0_2 are F_σ relations, and Π^0_2 are G_δ relations.

Even more general classes of sets form the *projective hierarchy*. We will need only its lowest level:

Σ^1_1 – *analytical* sets, i. e. projections of Borel relations,
Π^1_1 – complements of relations from Σ^1_1.

Let $\varphi : X \to Y$ be a continuous map of topological spaces. One says that φ *reduces* $A \subseteq X$ to $B \subseteq Y$, if $\forall x \in X\ x \in A \leftrightarrow \varphi(x) \in B$. Note that if B is in a certain class of the above hierarchies, so is A. For any class $\mathcal{C}$ a set B is $\mathcal{C}$*-hard*, if for any set $A \in \mathcal{C}$ there exists a reduction of A to B. The topological hierarchy is strict for Polish spaces, so if a set is $\mathcal{C}$-hard, it cannot be in any lower class. If a $\mathcal{C}$-hard set B is also an element of $\mathcal{C}$, then it is $\mathcal{C}$*-complete*.

For a deterministic automaton A one may define a function $\varphi_A : T_\Sigma \to T_{\text{im (rank)}}$ so that $\varphi_A(t)(v) = \text{rank}\,(\rho_t(v))$, where ρ_t is the run of A on t. Note that φ_A is a continuous map that reduces $L(A)$ to the set P of all trees satisfying A's parity condition.

We shall continue with a handful of examples which will turn out useful later.

Example 1. Consider the set $P_{(1,2)} \subseteq T_{\{1,2\}}$ consisting of trees having infinitely many 2s on every path. For each $n < \omega$ let G_n be the set of all trees that have at least one 2 below the level n on every path. From König lemma it follows that each G_n is open. Clearly, $P_{(1,2)} = \bigcap_{n\in\omega} G_n$ and so it is a Π^0_2 set.

Example 2. Let $P^{fin}_{(0,1)} \subseteq T_{\{0,1\}}$ be the set of trees in which there are only finitely many 1s. For any $n < \omega$ a set $F_n \subseteq T_{\{0,1\}}$ consisting of trees having no 1s below the level n is closed. $P^{fin}_{(0,1)} = \bigcup_{n\in\omega} F_n$, hence $P^{fin}_{(0,1)} \in \Sigma^0_2$.

Example 3. Let $L_a^{0^*1^\omega} \subseteq T_{\{a,b\}}$ be the set of trees which have an a on every path from the set 0^*1^ω. Suppose that it is a Σ^0_2 set. Let $L_a^{0^*1^\omega} = \bigcup_{n\in\omega} F_n$, F_n is closed for all n. We claim that for every n there exists m_n such that in every tree from F_n the letter a occurs on the path 0^n1^ω above the level m_n. If there was no such number, then we could find a sequence t_k of trees having no letters a on the path 0^n1^ω above the level l_k, where $l_1 < l_2 < l_3 < \ldots$. As $T_{\{a,b\}}$ is compact, there exists a subsequence t_{k_i} convergent in F_n. However the limit of t_{k_i} cannot be in F_n for it has no letter a on the path 0^n1^ω. Now, consider a tree t with a in nodes $0^n1^{m_n+1}$ and b in other nodes. Clearly, $t \in L_a^{0^*1^\omega}$, but $t \notin \bigcup_{n\in\omega} F_n$. This way we have shown that $L_a^{0^*1^\omega} \notin \Sigma^0_2$.

Example 4. In quite a similar way one proves that the set $Q = (0^*1)^*0^\omega$ is not in Π^0_2 (in fact, it is Σ^0_2-complete).

Example 5. Let $L_Q^{0^*1^\omega}$ denote the language of trees such that the rightmost path from every node of the form 0^* belongs to the language Q defined above. We shall see that it is Π^0_3-complete, and therefore it is not in Σ^0_3. Let us take any $M = \bigcup_{i<\omega} X_i$ with X_i in Σ^0_2. Since Q is Σ^0_2-complete, for each i there exists f_i reducing X_i to Q. One easily defines a continuous reduction of M to $L_Q^{0^*1^\omega}$ assigning to each t a tree having the word $f_i(t)$ on the path 0^i1^ω for all i, and 0s in all the other nodes.

5 Deciding Levels of Topological Hierarchy

The basic tool for investigating automata's properties is the technique of gadgets or difficult patterns in the graph of an automaton. In the topological context, the general recipe goes like this. For every class identify a gadget satisfying the following condition: if the gadget appears in an automaton A, then it provides a reduction of some difficult language to $L(A)$; otherwise $L(A)$ is in the class considered.

Wagner used this technique successfully in his solution of the general problem of continuous reductions between ω-languages [19]. For infinite words, the Borel hierarchy collapses at the level Δ^0_3 and below it is strict. The levels Π^0_1 and Σ^0_1 correspond to weak deterministic $(1,2)$ and $(0,1)$ automata; the class Δ^0_2 consists of all weak deterministic automata; Π^0_2 and Σ^0_2 are exactly deterministic Büchi and co-Büchi languages. We shall see that the situation for trees is slightly different.

We start with the gap property for deterministic tree languages. An automaton A admits a *split* if in G^+_A there are two loops $q_0 \stackrel{\rho,0}{\to} q_1 \to \ldots \to q_0$ and $q_0 \stackrel{\sigma,1}{\to} q_2 \to \ldots \to q_0$ such that the highest ranks occurring on them are of different parity and the higher is odd.

Theorem 2 (Niwiński, Walukiewicz [11]). *For a deterministic automaton A, $L(A)$ is on the level Π^0_3 of the Borel hierarchy iff A does not admit split; otherwise $L(A)$ is Π^1_1-complete (hence non-Borel).*

Owing to this result, it is enough to decide if a language is on the levels Σ^0_1, Π^0_1, Σ^0_2, Π^0_2, Σ^0_3 and use the split criterion to get complete information on its position in the topological hierarchy. Before dealing with this task we shall see that it does not get any easier, and so, not only there exist non-Borel tree languages but even the Borel hierarchy for trees is higher than for words.

Proposition 3. *The Borel hierarchy for deterministic tree languages is strict below Π^0_3.*

Proof. The language $L_a^{0^*}$ consisting of trees having an a on the leftmost path is open, but obviously is not closed, $L_a^{0^*} \in \Sigma^0_1 \setminus \Pi^0_1$. An example of a language from $\Pi^0_1 \setminus \Sigma^0_1$ can be $\{t_0\}$, where $t_0(v) = 0$ for every node v. The set Q can be reduced to $P^{fin}_{(0,1)}$ by a map $\{0,1\}^\omega \ni w \mapsto t_w \in T_{0,1}$ where t_w is a tree whose leftmost path is w and having 0 in all the other nodes. Hence $P^{fin}_{(0,1)} \in \Sigma^0_2 \setminus \Pi^0_2$. It can also be easily seen that $L_a^{0^*1^\omega}$ is a Π^0_2 set and we have already proved that it is not a Σ^0_2. Finally, the language $L_Q^{0^*1^\omega}$ has been shown not to be in the Σ^0_3 class, but clearly is in Π^0_3. □

Having seen the strictness of our confined hierarchy, we shall continue with the characterization of its levels. The description of the open and the closed languages is probably well known, so we state it here, together with a short proof, just for the sake of completeness.

Proposition 4. *For any deterministic tree automaton A*

1. *$L(A)$ is closed iff A is equivalent to a weak deterministic $(1,2)$-automaton* [1],
2. *$L(A)$ is open iff A is equivalent to a weak deterministic $(0,1)$-automaton.*

Proof. We will prove only (1). First, suppose that A is not equivalent to a weak deterministic $(1,2)$-automaton. It follows from Proposition 2 that in G^+_A there must be an accepting loop λ_2 reachable from a rejecting loop λ_1. Let g_1 realize a path from q_0 to some $q_1 \in \lambda_1$, g_2 realize a path form q_1 to some $q_2 \in \lambda_2$, and f_1, f_2 realize loops λ_1 (from q_1 to q_1), λ_2 (from q_2 to q_2) respectively. Consider $t_n = g_1(f_1)^n g_2(f_2)^\omega$ and $t = g_1(f_1)^\omega$. Clearly, $t_n \in L(A)$ and $t_n \to t$ when $n \to \infty$, but $t \notin L(A)$. Hence $L(A)$ is not closed.

Now, if $L(A)$ is recognized by a weak deterministic automaton B, then it is the inverse image of a point under the continuous map φ_B and so it is closed. □

The combinatorial characterization of Π^0_2-languages transpires to be equally elegant.

[1] Recall that we do not let an automaton stop. If we did, there should be $(0,0)$ instead of $(1,2)$.

Theorem 3. *For a deterministic tree automaton A, the language $L(A)$ is on the level Π^0_2 of the Borel hierarchy iff A is equivalent to a deterministic Büchi automaton.*

Proof. ($\Rightarrow$) Suppose that $L(A)$ is not recognized by a deterministic Büchi automaton. From Proposition 1 it follows that in G^+_A there exist an accepting loop λ_0 and a rejecting loop λ_1 forming a $(0,1)$-flower. The loops λ_0 and λ_1 cannot be equal, so there is a node q lying on both loops, such that the next edges going out of q in λ_0 and λ_1 have different labels. Let us assume first that the edges are labeled with different letters $a \neq b$:

$$\lambda_0 : q \xrightarrow{a,d_0} r \longrightarrow \ldots \longrightarrow q \ ,$$

$$\lambda_1 : q \xrightarrow{b,d_1} s \longrightarrow \ldots \longrightarrow q \ .$$

Let f_0, f_1 be tree segments realizing the loops λ_0 and λ_1 respectively (both from q to q). Note that f_0 and f_1's roots are labeled with different letters a and b. Consider a map $\varphi : 2^\omega \to T_\Sigma$ defined by the formula

$$\varphi(x_0x_1x_2\ldots) = ff_{x_0}f_{x_1}f_{x_2}\cdots \ ,$$

where f is a tree segment realizing a path from q_0 to q. The map φ is continuous, since $d(\varphi(x),\varphi(x')) \leq d(x,x')^{|\lambda_0|}$. Thus we have reduced $(0^*1)^*0^\omega$ to $L(A)$, which, by Example 4, implies that $L(A)$ is not a Π^0_2 language.

The second case is slightly more sophisticated. We have

$$\lambda_0 : q \xrightarrow{a,0} r \longrightarrow \ldots \longrightarrow q$$

$$\lambda_1 : q \xrightarrow{a,1} s \longrightarrow \ldots \longrightarrow q$$

(or dual). Consider a path in the graph G^+_A along the edges of the loop λ_1 starting from q. We claim that it must reach a node q' such that there is an edge e from q' to q''' labeled with the same direction (0 or 1) as the edge in the loop but with a different letter, e.g.

$$\lambda_1 : q \xrightarrow{a,1} s \longrightarrow \ldots \longrightarrow q' \xrightarrow{b,0} q'' \longrightarrow \ldots \longrightarrow q$$
$$\searrow^{c,0}$$
$$q'''$$

.

Were there no such an edge, all the runs starting in q would loop on λ_1 and q would be unproductive. Let π_0 be the path $q \xrightarrow{a,1} s \longrightarrow \ldots \longrightarrow q' \xrightarrow{e} q'''$. The state q''' is productive, so we can extend π_0 to an infinite accepting path π in G^+_A. For f_0 we choose a tree segment realizing both λ_0 and π. This is possible owing to the fact, that λ_0 and π start with edges labeled with the same letter and different directions. As before, f_1 can be any tree segment realizing λ_1. Now we can continue like in the previous case.

($\Leftarrow$) $L(A)$ can be reduced to $P_{(1,2)}$, which is a Π^0_2-set. □

We shall now continue with describing the Σ_2^0 languages. Recall the language $L_a^{0^*1^\omega} \subseteq T_{\{a,b\}}$ consisting of trees which have an a on every path from the set 0^*1^ω. Even though one may easily construct a deterministic $(0,1)$-automaton recognizing this language, it is not a Σ_2^0 set. Since a simple analog of the Π_2^0 case condition has shown insufficient, a more careful analysis of the automaton's graph is needed. We will say that a node $v \in G_A^+$ is *accessible with a split* if in G_A^+ there exist an accepting loop $u_1 \xrightarrow{\sigma,d_0} u_2 \longrightarrow \ldots \longrightarrow u_1$ and a path $u_1 \xrightarrow{\sigma,d_1} u_2' \longrightarrow \ldots \longrightarrow v$, where $d_0 \neq d_1$. We will say that a loop or a flower is accessible with a split, meaning that it contains a node accessible with a split.

Theorem 4. *For a deterministic tree automaton A, the language $L(A)$ is on the level Σ_2^0 of the Borel hierarchy iff A is equivalent to a deterministic $(0,1)$-automaton and G_A^+ does not contain a rejecting loop accessible with a split.*

Proof. ($\Rightarrow$) Let us suppose that $L(A)$ is a Σ_2^0 language. To prove the equivalence to a deterministic $(0,1)$-automaton follow the dual version of the method used in the previous theorem. There exist a rejecting loop λ_1 and an accepting loop λ_0 forming a $(1,2)$-flower. Find tree segments f_0, f_1 realizing λ_0, λ_1. Make sure they are different by finding an accepting path π leaving λ_1. The map φ defined in the previous proof reduces $(1^*0)^\omega$ to $\operatorname{im}\varphi \cap L(A)$. Were $L(A)$ a Σ_2^0 set, so would $(1^*0)^\omega$, which, by Example 4, is not true. Let us now concentrate on the second part of the condition. Suppose that G_A^+ does contain a rejecting loop λ_1 accessible with a split from an accepting loop λ_0 along a path π. For $n \in \omega$ let π_n be an infinite accepting path having a prefix $\pi(\lambda_1)^n$ but no prefixes $\pi(\lambda_1)^m$ for $m > n$ (find an edge leaving the rejecting loop λ_1 just as it was done in the second case of the previous proof) and $\pi_\omega = \pi(\lambda_1)^\omega$. For each $\alpha \in \omega+1 = \omega \cup \{\omega\}$ consider a tree segment f_α realizing both λ_0 and π_α, this being possible since the first edges of both paths are labeled with the same letter σ and different directions d_0, d_1. For any $x = (x_1, x_2, \ldots) \in (\omega+1)^\omega$ let $t_x = f f_{x_1} f_{x_2} \ldots$, where f is a tree segment realizing a path from the initial state q_0 to u_1. We shall define a continuous map $\varphi : T_{\{a,b\}} \to T_\Sigma$. For $s \in T_{\{a,b\}}$ let $y_i = \min(\{|w| : w \in 0^i 1^*,\ s(w) = a\} \cup \omega)$. Let $z_i = \max(y_i - i, 0)$ if $y_i < \omega$ and $z_i = \omega$ if $y_i = \omega$. Let us now set $\varphi(s) = t_z$, where $z = (z_0, z_1, \ldots)$. The map φ reduces $L_a^{0^*1^\omega}$ to $L(A)$. However, we have already shown that $L_a^{0^*1^\omega}$ is not a Σ_2^0 language. Hence G_A^+ cannot contain a rejecting loop accessible with a split.

($\Leftarrow$) Investigating the proof of Theorem 1 one easily observes that the reduction is careful enough not to introduce any rejecting loops accessible with a split, provided there are no such loops in the original automaton. Therefore, we may assume that A is a $(0,1)$-automaton such that G_A^+ does not contain a rejecting loop accessible with a split. A state is called *relevant* if it has the highest rank on some productive loop. We may change the ranks of productive irrelevant states to 0, and assume from now on that the odd states are either relevant or unproductive. We claim that the odd states occur only finitely many times on accepting runs of A. Suppose that an odd state p occurs infinitely many times in an accepting run ρ. Then it appears in an infinite number of incomparable nodes $v_0, v_1, \ldots$ of ρ. Let π_i be a path of ρ going through the node v_i. Since

2^ω is compact, we may assume, passing to a subsequence, that the sequence π_i converges to a path π. As v_i are incomparable, at most one of them, say v_{i_0}, may lie on π. Let us remove π_{i_0} from the sequence π_i. Consider the node w_i in which p occurs for the first time on π_i after leaving π and let π_i^0 be the path from the last common node of π and π_i to w_i. Cutting the loops off if needed, we may assume that $|\pi_i^0| \leq |Q|$ for all $i \in \omega$. Subsequently, there exist a path π^0 repeating infinitely often in the sequence $\pi_0^0, \pi_1^0, \ldots$. Moreover, the path π is accepting, so the starting node of π^0 must lay on an accepting productive loop. As p is productive, the assumption implies that it is relevant and, being odd, lies on some productive rejecting loop. Hence, G_A^+ contains a rejecting loop accessible with a split – a contradiction. This way we have shown that φ_A reduces $L(A)$ to $P_{0,1}^{fin}$, and so $L(A)$ is a Σ_2^0 language. □

Let us now consider the class Σ_3^0. Every deterministic Σ_3^0 language is, due to Theorem 2, in the $\Delta_3^0 = \Pi_3^0 \cap \Sigma_3^0$ class. Below we present a combinatorial description of this class of languages.

Theorem 5. *For a deterministic tree automaton A, $L(A)$ is a Σ_3^0 set iff G_A^+ does not contain a $(0,1)$-flower accessible with a split.*

Proof. First let us suppose that G_A^+ contains a $(0,1)$-flower accessible with a split. Following the method used in Theorem 3 one easily finds a map reducing the language $L_Q^{0^*1^\omega}$ to $L(A)$. Subsequently, $L(A)$ is not a Σ_3^0 language.

Now, suppose that G_A^+ does not contain a $(0,1)$-flower accessible with a split. We shall find a Σ_3^0 representation of the set $R \subseteq T_Q$ of accepting runs of A. The theorem will follow since the map $T_\Sigma \ni t \mapsto \rho_t \in T_Q$ is continuous. Let us consider, then, the set $\mathcal{X}$ of strongly connected components of G_A^+. Recall that they form a directed acyclic graph, i. e. no path returns to a component it has left. The language R can be expressed by the following formula

$$R = \bigcap_{X \in \mathcal{X}} R_X \ ,$$

where R_X is the set of runs whose every path staying forever in X is accepting. Owing to this simple observation, it is enough to prove that the sets R_X are Σ_3^0.

Let Π_X denote the set of all paths from the initial state q_0 to some state in X containing only one state from X. Note that Π_X is countable for every X. Let us first suppose that X is accessible with a split. For $\pi \in \Pi_X$ let $R_{X,\pi}$ denote the set of runs whose every path going along π either leaves X or is accepting. By the hypothesis, X contains no $(0,1)$-flowers, and so it can be replaced by an equivalent component X' using only ranks 1 and 2. Therefore, given $q \in X'$, the set of runs of A_q, whose all paths are accepting or leave X', is a Π_2^0 set. Obviously, so is $R_{X,\pi}$. As it also holds that

$$R_X = \bigcap_{\pi \in \Pi_X} R_{X,\pi} \ ,$$

R_X is a Π_2^0 set.

The case of X not accessible with a split is slightly fastidious. Let ρ be an accepting run of A. Consider ρ_X, a subtree of ρ formed by the nodes which have a successor whose labeling state is in X. No state from X is accessible with a split, therefore it cannot appear in infinitely many incomparable nodes of ρ. Hence, ρ_X has only finitely many branches. Let ρ_X^0 denote the tree ρ_X restricted to the highest level below which there are no branching points. Let R_X^0 denote the set of all such trees; note that, although R_X may be uncountable, R_X^0 is countable. Obviously,

$$R_X = \bigcup_{s \in R_X^0} R_{X,s} \ ,$$

where $R_{X,s}$ is the set of runs from R_X coinciding with a tree $s \in R_X^0$ in its domain. Observe that $R_{X,s}$ is equal to the set of runs ρ' satisfying the following conditions:

(1) ρ' coincides with s in its domain,
(2) the states from X appear only in successors of leaves of s,
(3) in every subtree of ρ' rooted in a leaf of s the states from X appear infinitely often on at most one path,
(4) in every subtree of ρ' rooted in a leaf of s the highest rank of the states from X appearing infinitely often is even.

The condition (1) obviously defines an open set. The condition (2) defines a closed set and so does the condition (3), because it is equivalent to saying that no node of the subtree has both children in X. The condition (4) is of the $B(\Sigma_2^0)$ type. By $B(\Sigma_2^0)$ we mean the closure of Σ_2^0 by the finite Boolean operations; it is clearly a subclass of Σ_3^0. Hence $R_{X,s}$ is a Σ_3^0 set and so is R_X. □

As a conclusion we obtain the main result of this paper.

Corollary 1. *The problem of calculating the exact position in the topological hierarchy of a language recognized by a deterministic tree automaton is decidable within the time of finding the productive states of a deterministic automaton.*

Proof. From Proposition 4 it follows that the language recognised by a deterministic automaton A is closed iff A is equivalent to a weak $(1,2)$-automaton. This, by Proposition 2, can be reformulated as follows: $L(A)$ is closed iff G_A^+ does not contain a weak $(0,1)$-flower. Now, to decide whether a deterministic automaton recognises a closed set, first determine its productive states, then build its productive graph and check for weak $(0,1)$-flowers. Note that two last steps can be easily done in polynomial time. The case of open languages is entirely dual.

For Π_2^0 and Σ_2^0 levels follow analogous argument only now using Theorem 3 and Theorem 4 respectively, and Proposition 1. For Π_3^0 and Σ_3^0 levels use the gap property and Theorem 5.

This way for a given deterministic language one obtains its exact level in the topological hierarchy. □

In general, deciding the topological complexity of a deterministic tree language is as difficult as calculating the unproductive states of an automaton, the latter being equivalent to deciding a language's emptiness. In 1969 Rabin [14] showed that the emptiness problem is decidable, and in 1988 Emerson and Jutla [3] presented an algorithm with time complexity $\mathcal{O}((nd)^{3d})$, where n is the number of states and d is the number of ranks used. The emptiness problem can be easily reduced to solving parity games. The late nineties brought improved algorithms for this problem by Browne et al. [2] with complexity $\mathcal{O}(d^2mn^{\frac{d}{2}})$ and by Seidl [15] with complexity $\mathcal{O}(dm(\frac{n+d}{d})^{\frac{d}{2}})$, where n, m, and d are the numbers of vertices, edges, and ranks in the game graph. The investigation of parity games resulted in polynomial algorithms in plenty of special cases, but so far it is not known if the original problem is polynomial. One of the last achievements in this field is the procedure by Jurdziński and Vöge [4] which is apparently quite efficient, however its complexity has not, at present, got any nontrivial upper bounds.

Acknowledgment

The author would like to thank Damian Niwiński for helpful comments and discussions, and the anonymous referees for helpful comments.

References

1. Bradfield, J. C.: The modal mu-calculus alternation hierarchy is strict. Theoret. Comput. Sci. **195** (1998) 133–153
2. Browne, A., Clarke, E. M., Jha, S., Long, D. E., Marrero, W.: An improved algorithm for the evaluation of fixpoint expressions. Theoret. Comput. Sci. **178** (1997) 237–255
3. Emerson, E. A., Jutla, C. S.: The complexity of tree automata and logics of programs. In: Proc. FoCS '88. IEEE Computer Society Press (1988) 328–337
4. Jurdziński, M., Vöge, J.: A discrete strategy improvement algorithm for solving parity games. In: Proc. CAV 2000. Lecture Notes in Computer Science, Vol. 1855. Springer-Verlag (2000) 202–215
5. Kechris, A. S.: Classical Descriptive Set Theory. Graduate Texts in Mathematics, Vol. 156. Springer-Verlag (1995)
6. Kupferman, O., Safra, S., Vardi, M.: Relating Word and Tree Automata. In: 11th IEEE Symp. on Logic in Comput. Sci. (1996) 322–332
7. Lenzi, G.: A hierarchy theorem for the mu-calculus. In: auf der Heide, F. M., Monien, B. (eds.): Proc. ICALP '96. Lecture Notes in Computer Science, Vol. 1099. Springer-Verlag (1996) 87–109
8. Mostowski, A. W.: Hierarchies of weak automata and weak monadic formulas. Theoret. Comput. Sci. **83** (1991) 323-335.
9. Niwiński, D.: On fixed point clones. In: Kott, L. (ed.): 13th ICALP '86. Lecture Notes in Computer Science, Vol. 226. Springer-Verlag (1986) 464–473
10. Niwiński, D., Walukiewicz, I.: Relating hierarchies of word and tree automata. In: STACS '98. Lecture Notes in Computer Science, Vol. 1373. Springer-Verlag (1998) 320–331

11. Niwiński, D., Walukiewicz, I.: A gap property of deterministic tree languages. Theoret. Comput. Sci. **303** (2003) 215–231
12. Niwiński, D., Walukiewicz, I.: Deciding nondeterministic hierarchy of deterministic tree automata. In: Proc. WoLLiC 2004 (to appear in Electronic Notes in Theoretical Computer Science)
13. Otto, M.: Eliminating recursion in μ-calculus. In: STACS'99. Lecture Notes in Computer Science, Vol. 1563. Springer-Verlag (1999) 531–540
14. Rabin, M. O.: Decidability of second-order theories and automata on infinite trees. Trans. Amer. Soc. **141** (1969) 1–35
15. Seidl, H.: Fast and simple nested fixpoints. Information Processing Letters **59** (1996) 303–308
16. Skurczyński, J.: The Borel hierarchy is infinite in the class of regular sets of trees. Theoret. Comput. Sci. **112** (1993) 413–418
17. Thomas, W.: Languages, automata, and logic. In: Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, Vol. 3. Springer-Verlag (1997) 389–455
18. Urbański, T. F.: On deciding if deterministic Rabin language is in Büchi class. In: Montanari, J. R. U., Welzl, E. (eds.): Proc. ICALP 2000. Lecture Notes in Computer Science, Vol. 1853. Springer-Verlag (2000) 663–674
19. Wagner, K.: Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen. J. Inf. Process. Cybern. EIK **13** (1977) 473–487