WEAK INDEX VERSUS BOREL RANK

FILIP MURLAK [1]

[1] Warsaw University
E-mail address: fmurlak@mimuw.edu.pl

ABSTRACT. We investigate weak recognizability of deterministic languages of infinite trees. We prove that for deterministic languages the Borel hierarchy and the weak index hierarchy coincide. Furthermore, we propose a procedure computing for a deterministic automaton an equivalent minimal index weak automaton with a quadratic number of states. The algorithm works within the time of solving the emptiness problem.

1. Introduction

Finite automata on infinite trees are one of the basic tools in the verification of non-terminating programs. Practical applicability of this approach relies on the simplicity of the automata used to express the specifications. On the other hand it is convenient to write the specifications in an expressive language, e. g. μ-calculus. This motivates the search for automatic simplifications of automata. An efficient, yet reasonably expressive, model is offered by weak alternating automata. It was essentially showed by Rabin [18] that a language L can be recognized by a weak automaton if and only if both L and $L^{\complement}$ can be recognized by nondeterministic Büchi automata. Arnold and Niwiński [2] proposed an algorithm that, given two Büchi automata recognizing a language and its complement, constructs a doubly exponential alternation free μ-calculus formula defining L, which essentially provides an equally effective translation to a weak automaton. Kupferman and Vardi [7] gave an immensely improved construction that involves only quadratic blow-up.

A more refined construction could also simplify an automaton in terms of different complexity measures. A measure that is particularly important for theoretical and practical reasons is the Mostowski–Rabin index. This measure reflects the alternation depth of positive and negative events in the behaviour of a verified system. The index orders automata into a hierarchy that was proved strict for deterministic [21], nondeterministic [13], alternating [4, 8], and weak alternating automata [9]. Computing the least possible index for a given automaton is called the index problem. Unlike for ω-words, where the solution was essentially given already by Wagner [21], for trees this problem in its general form

SYMPOSIUM
ON THEORETICAL
ASPECTS
OF COMPUTER
SCIENCE

remains unsolved. For deterministic languages, Niwiński and Walukiewicz gave algorithms to compute the deterministic and nondeterministic indices [14, 16].

The theoretical significance of the weak index is best reflected by its coincidence with the quantifier alternation depth in the weak monadic second order logic [9]. Further interesting facts are revealed by the comparison with the Borel rank. In 1993 Skurczyński gave examples of Π_n^0 and Σ_n^0-complete languages recognized by weak alternating automata with index $(0, n)$ and $(1, n + 1)$ accordingly [19]. In [5] it was shown that weak $(0, n)$-automata can only recognize Π_n^0 languages (and dually, $(1, n + 1)$-automata can only recognize Σ_n^0 languages), and it was conjectured that the weak index and the Borel hierarchies actually coincide. Here we prove that the conjecture holds for deterministic languages. Consequently, the algorithm calculating the Borel rank for deterministic languages [11] can be also used to compute the weak index. Since all deterministic languages are at the first level of the alternating hierarchy, this completes the picture for the deterministic case. We also provide an effective translation to a weak automaton with a quadratic number of states and the minimal index.

2. Automata

We will be working with deterministic and weak automata, but to have a uniform framework, we first define automata in their most general alternating form.

A *parity game* is a perfect information game of possibly infinite duration played by two players, Adam and Eve. We present it as a tuple $(V_\exists, V_\forall, E, v_0, \mathrm{rank})$, where $V_\exists$ and $V_\forall$ are (disjoint) sets of positions of Eve and Adam, respectively, $E \subseteq V \times V$ is the relation of possible moves, with $V = V_\exists \cup V_\forall$, $p_0 \in V$ is a designated initial position, and $\mathrm{rank} : V \to \{0, 1, \ldots, n\}$ is the ranking function.

The players start a play in the position v_0 and then move a token according to relation E (always to a successor of the current position), thus forming a path in the graph (V, E). The move is selected by Eve or Adam, depending on who is the owner of the current position. If a player cannot move, she/he looses. Otherwise, the result of the play is an infinite path in the graph, $v_0, v_1, v_2, \ldots$. Eve wins the play if the highest rank visited infinitely often is even, otherwise Adam wins.

An *alternating automaton* $A = \langle \Sigma, Q_\exists, Q_\forall, q_0, \delta, \mathrm{rank} \rangle$, consists of a finite input alphabet Σ, a finite set of states Q partitioned into existential states $Q_\exists$ and universal states $Q_\forall$ with a fixed initial state q_0, a transition relation $\delta \subseteq Q \times \Sigma \times \{0, 1, \varepsilon\} \times Q$, and a ranking function $\mathrm{rank} : Q \to \omega$. Instead of $(p, \sigma, d, q) \in \delta$, one usually writes $p \xrightarrow{\sigma, d} q$.

An input tree t is accepted by A iff Eve has a winning strategy in the parity game $\langle Q_\exists \times \{0, 1\}^*, Q_\forall \times \{0, 1\}^*, (q_0, \varepsilon), E, \mathrm{rank}' \rangle$, where $E = \{((p, v), (q, vd)) : v \in \mathrm{dom}(t), (p, t(v), d, q) \in \delta\}$ and $\mathrm{rank}'(q, v) = \mathrm{rank}(q)$. The computation tree of A on t is obtained by unravelling the graph above from the vertex (q_0, ε) and labelling the node $(q_0, \varepsilon), (q_1, d_1), (q_2, d_2), \ldots, (q_n, d_n)$ with q_n. The result of the parity game above only depends on the computation tree.

An automaton is called *deterministic* iff Eve has no choice at all, and Adam can only choose the direction: left or right (no ε-moves). Formally, it means that $Q_\exists = \emptyset$, and $\delta : Q \times \Sigma \times \{0, 1\} \to Q$. For deterministic automata, the computation tree is a full binary tree. The transitions are often written as $p \xrightarrow{\sigma} q_0, q_1$, meaning $p \xrightarrow{\sigma, d} q_d$ for $d = 0, 1$.

A *weak automaton* is an alternating automaton satisfying the condition

$$p \xrightarrow{\sigma, d} q \quad \implies \quad \mathrm{rank}\, p \leq \mathrm{rank}\, q\,.$$

A more elegant definition of the class of weakly recognizable languages is obtained by using *weak parity games* in the definition of acceptance by alternating automata. In those games Eve wins a play if the highest rank used at least once is even. For the purpose of the following lemma, let us call the first version *restricted alternating automata*. Later, we will stick to the second definition.

Lemma 2.1. *For every L it holds that L is recognized by a restricted alternating (ι, κ)-automaton iff it is recognized by a weak alternating (ι, κ)-automaton.*

Proof. Every restricted automaton can be transformed into an equivalent weak automaton by simply changing the acceptance condition to weak. Let us, then, concentrate on the converse implication.

Fix a weak automaton A using ranks (ι, κ). To construct a restricted automaton we will take one copy of A for each rank: $A^{(\iota)}, A^{(\iota+1)}, \ldots, A^{(\kappa)}$. By $q^{(i)}$ we will denote the counterpart of A's state q in $A^{(i)}$. We set $\operatorname{rank} q^{(i)} = i$. We want the number of the copy the computation is in to reflect the highest rank seen so far. To obtain that, we set the initial state of the new automaton to $q_0^{(\operatorname{rank} q_0)}$, and for each i and each transition $p \xrightarrow{\sigma, d} q$ in A we add a transition $p^{(i)} \xrightarrow{\sigma, d} q^{(\max(i, \operatorname{rank} q))}$. For each i and q, $q^{(i)}$ is universal iff q is universal. Checking the equivalence is straightforward. ∎

For deterministic automata we will assume that all states are productive, i. e., are used in some accepting run, save for one all-rejecting state $\bot$, and that all transitions are productive or go to $\bot$, i. e., whenever $q \xrightarrow{\sigma} q_1, q_2$, then either q_1 and q_2 are productive, or $q_1 = q_2 = \bot$. The assumption of productivity is vital for our proofs. Thanks to this assumption, in each node of an automaton's run we can plug in an accepting sub-run.

Transforming a given automaton into such a form of course needs calculating the productive states, which is equivalent to deciding a language's emptiness. The latter problem is known to be in $\text{NP} \cap \text{co-NP}$, but it has no polynomial solutions yet. Therefore we can only claim that our algorithms are polynomial for the automata that underwent the above preprocessing. We will try to mention it whenever particularly important.

3. Two Hierarchies

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 $\min \operatorname{rank} Q$ is either 0 or 1. Thus, the indices are elements of $\{0, 1\} \times \omega \setminus \{(1, 0)\}$. For an index (ι, κ) we shall denote by $\overline{(\iota, \kappa)}$ the *dual index*, i. e., $\overline{(0, \kappa)} = (1, \kappa + 1)$, $\overline{(1, \kappa)} = (0, \kappa - 1)$. Let us define an ordering of indices with the following formula:

$$(\iota, \kappa) < (\iota', \kappa') \text{ if and only if } \kappa - \iota < \kappa' < \iota'.$$

In other words, one index is greater than another if and only if it "uses" more ranks. This means that dual indices are incomparable. *The Mostowski–Rabin index hierarchy* for a certain class of automata consists of ascending sets (levels) of languages recognized by (ι, κ)-automata.

Here, we are mainly interested in the *weak index hierarchy*, i. e., the hierarchy of languages recognized by weak (ι, κ)-automata. The strictness of this hierarchy was established by Mostowski [9] via equivalence with the quantifier-alternation hierarchy for the weak
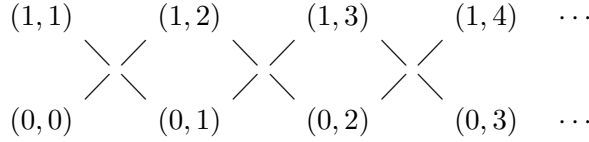
$$(1,1) \qquad (1,2) \qquad (1,3) \qquad (1,4) \quad \cdots$$

$$(0,0) \qquad (0,1) \qquad (0,2) \qquad (0,3) \quad \cdots$$

Figure 1: The Mostowski–Rabin index hierarchy

$$\Sigma_0^0 \qquad \Sigma_1^0 \qquad \Sigma_2^0 \qquad \Sigma_3^0 \quad \cdots$$

$$\Delta_1^0 \qquad \Delta_2^0 \qquad \Delta_3^0$$

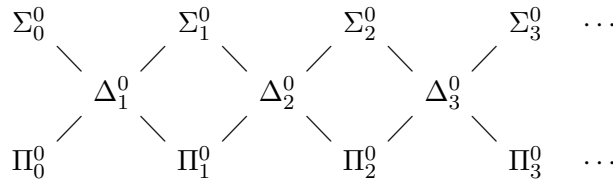$$\Pi_0^0 \qquad \Pi_1^0 \qquad \Pi_2^0 \qquad \Pi_3^0 \quad \cdots$$

Figure 2: The Borel hierarchy

monadic second order logic, whose strictness was proved by Thomas [20]. The *weak index problem*, i. e., computing the minimal weak index needed to recognize a given weak language, for the time being remains unsolved just like other versions of the index problem.

The weak index hierarchy is closely related to the Borel hierarchy. We will work with the standard Cantor-like topology on T_Σ induced by the metric

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

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.

For a topological space X, the initial (finite) levels of the *Borel hierarchy* are defined as follows:

- $\Sigma_1^0(X)$ – open subsets of X,
- $\Pi_k^0(X)$ – complements of the sets from $\Sigma_k^0(X)$,
- $\Sigma_{k+1}^0(X)$ – countable unions of sets from $\Pi_k^0(X)$.

For instance, $\Pi_1^0(X)$ are the closed sets, $\Sigma_2^0(X)$ are F_σ sets and $\Pi_2^0(X)$ are G_δ sets. By convention $\Sigma_0^0(X) = \{\emptyset\}$ and $\Pi_0^0(X) = \{X\}$.

A straightforward inductive argument shows that the classes defined above are closed under inverse images of continuous functions. Let $\mathcal{C}$ be one of those classes. A set A is called $\mathcal{C}$-*hard*, if each set in $\mathcal{C}$ is an inverse image of A under some continuous function. If additionally $A \in \mathcal{C}$, A is $\mathcal{C}$-*complete*.

We start the discussion of the relations between the index of a weak automaton and the Borel rank of the language it recognises by recalling Skurczyński's results. For a tree $t : \{0,1\}^* \to \Sigma$ and a node $v \in \{0,1\}^*$ let $t.v$ denote the tree rooted in v, i. e., $t.v(w) = t(vw)$. Let us define a sequence of languages:

- $L_{(0,1)} = \{t\}$, where $t \in T_{\{a,b\}}$ is the tree with no b's,
- $L_{(1,n+1)} = L_{(0,n)}^{\complement}$ for $n \geq 1$,
- $L_{(0,n+1)} = \{t \in T_{\{a,b\}} : \forall_k \ t.0^k 1 \in L_{(1,n+1)}\}$ for $n \geq 1$.

Theorem 3.1 (Skurczyński [19]). *For each $n \geq 1$,*

- $L_{(0,n)}$ *is a Π_n^0-complete language recognized by a weak $(0,n)$-automaton,*

- $L_{(1,n+1)}$ *is a Σ_n^0-complete language recognized by a weak $(1, n + 1)$-automaton.*

We will now show that this construction is as efficient as it can be: ranks $(0, n)$ are necessary to recognize any Π_n^0-hard language (if it can be weakly recognized at all).

We will actually prove a bit stronger result. We will consider *weak game languages* $W_{[\iota,\kappa]}$, to which all languages recognized by weak $[\iota, \kappa]$-automata can be reduced, and show that $W_{[0,n]} \in \Pi_n^0$ and $W_{[1,n+1]} \in \Sigma_n^0$ (by Skurczyński's results, they are hard for these classes). The languages $W_{[\iota,\kappa]}$ are natural weak counterparts of strong game languages that prove the strictness of the strong alternating index hierarchy. Lately Arnold and Niwiński proved that the strong game languages also form a strict hierarchy with respect to continuous reductions, but they are all non-Borel [3].

Fix a natural number N. For $\iota = 0, 1$ and $\kappa \geq \iota$, let $\mathcal{T}_{(\iota,\kappa)}$ denote the set of full N-ary trees over the alphabet $\{\exists, \forall\} \times \{\iota, \iota+1, \ldots, \kappa\}$. Let $W_{(\iota,\kappa)} \subseteq \mathcal{T}_{(\iota,\kappa)}$ be the set of all trees t for which Eve has a winning strategy in the *weak* parity game $G_t = \langle V_\exists, V_\forall, E, v_0, \mathrm{rank} \rangle$, where $V_\theta = \{v \in \mathrm{dom}\, t \colon t(v) = (\theta, j) \text{ for some } j\}$, $E = \{(v, vk) \colon v \in \mathrm{dom}\, t,\ k < N\}$, $v_0 = \varepsilon$, $\mathrm{rank}(v) = j$ iff $t(v) = (\theta, j)$ for some θ.

Theorem 3.2. *For each n, $W_{(0,n)} \in \Pi_n^0(\mathcal{T}_{(0,n)})$ and $W_{(1,n+1)} \in \Sigma_n^0(\mathcal{T}_{(1,n+1)})$.*

Proof. We will proceed by induction on n. For $n = 0$ the claim is obvious: $W_{(0,0)} = \mathcal{T}_{(0,0)} \in \Pi_0^0(\mathcal{T}_{(0,0)})$, $W_{(1,1)} = \emptyset \in \Sigma_0^0(\mathcal{T}_{(1,1)})$.

Take $n > 0$. For each $t \in W_{(1,n+1)}$ there exists a strategy σ for Eve, such that it guarantees that the play reaches a node with the rank greater or equal to 2. By König lemma, this must happen in a bounded number of moves. Basing on this observation we will provide a Σ_n^0 presentation of $W_{(1,n+1)}$.

Let k-*antichain* be a subset of the nodes on the level k. Let $\mathcal{A}$ denote the set of all possible k-antichains for all $k < \omega$. Obviously this set is countable. For a k-antichain A let W_A denote the set of trees such that there exists a strategy for Eve that guarantees visiting a node with the rank ≥ 2 during the initial k moves and reaching a node from A. This set is a clopen. We have a presentation

$$W_{(1,n+1)} = \bigcup_{A \in \mathcal{A}} \left(W_A \cap \bigcap_{v \in A} \{t : t'.v \in W_{(0,n-1)}\} \right),$$

where t' is obtained from t by decreasing all the ranks by 2 (if the result is -1, take 0). The claim follows by induction hypothesis and the continuity of $t \mapsto t'$ and $t \mapsto t.v$.

Now, it remains to see that $W_{(0,n)} \in \Pi_n^0(\mathcal{T}_{(0,n)})$. For this, note that

$$W_{(0,n)} = \left\{ t : t'' \in (W_{(1,n+1)})^{\complement} \right\},$$

where t'' is obtained from t by swapping $\exists$ and $\forall$, and increasing ranks by 1. The claim follows by the continuity of $t \mapsto t''$. ∎

As a corollary we get the promised improvement of Skurczyński's result.

Corollary 3.3. *For every weak alternating automaton A with index $(0, n)$ (resp. $(1, n+1)$) it holds that $L(A) \in \Pi_n^0$ (resp. $L(A) \in \Sigma_n^0$).*

Proof. Let A be an automaton with priorities inside $[\iota, \kappa]$. For sufficiently large N we may assume without loss of generality that the computation trees of the automaton are N-ary trees. By assigning to an input tree the run of A, one obtains a continuous function reducing $L(A)$ to $W_{(\iota,\kappa)}$. Hence, the claim follows from the theorem above. ∎

$$\Sigma_0^0 \quad\quad \Sigma_1^0 \quad\quad \Sigma_2^0$$

$$\Delta_1^0 \quad\quad \Delta_2^0 \quad\quad \Delta_3^0 = \Sigma_3^0$$

$$\Pi_0^0 \quad\quad \Pi_1^0 \quad\quad \Pi_2^0 \quad\quad \Pi_3^0 \quad | \quad \text{non-Borel}$$

$$(1,1) \quad\quad (1,2) \quad\quad (1,3)$$

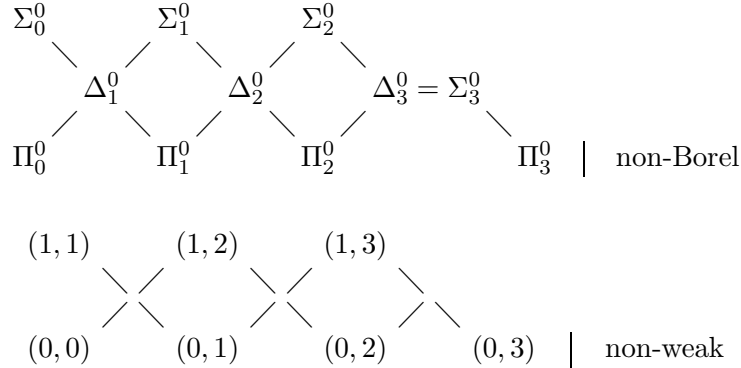$$(0,0) \quad\quad (0,1) \quad\quad (0,2) \quad\quad (0,3) \quad | \quad \text{non-weak}$$

Figure 3: The Borel hierarchy and weak index hierarchy for deterministic tree languages.

In fact the corollary follows also from Mostowski's theorem on equivalence of weak automata and weak monadic second order logic on trees [9]. The present proof of Theorem 3.2 is actually just a repetition of Mostowski's proof in the setting of the Borel hierarchy. An entirely different proof can be found in [5].

We believe that the converse implication is also true: a weakly recognizable Π_n^0-language can be recognized by a weak $(0, n)$-automaton (and dually for Σ_n^0).

Conjecture 3.4. For weakly recognizable languages the weak index hierarchy and the Borel hierarchy coincide.

In this paper we show that the conjecture holds true when restricted to deterministic languages.

4. The Deterministic Case

In 2002 Niwiński and Walukiewicz discovered a surprising dichotomy in the family of deterministic languages: a deterministic language is either very simple or very sophisticated.

Theorem 4.1 (Niwiński, Walukiewicz [15]). *For a deterministic automaton A with n states, $L(A)$ is either recognizable with a weak alternating $(0, 3)$-automaton with $\mathcal{O}(n^2)$ states (and so Π_3^0) or is non-Borel (and so not weakly recognizable). The equivalent weak automaton can be constructed within the time of solving the emptiness problem.*

An important tool used in the proof of the Gap Theorem (Theorem 4.1) is the technique of difficult patterns. In the topological setting the general recipe goes like this: for a given class identify a pattern that can be unravelled to a language complete for this class; if an automaton does not contain the pattern, then $L(A)$ should be in the dual class. The same technique was later applied to obtain effective characterisations of the remaining Borel classes of deterministic languages [11].

Let us define the patterns used in these characterisations. A *loop* in an automaton is a sequence of states and transitions:

$$p_0 \xrightarrow{\sigma_1,d_1} p_1 \xrightarrow{\sigma_2,d_2} \ldots \xrightarrow{\sigma_n,d_n} p_0 \,.$$

A loop is called *accepting* if $\max_i \operatorname{rank}(p_i)$ is even. Otherwise it is *rejecting*.

A (ι,κ)-*flower* is a sequence of loops $\lambda_\iota, \lambda_{\iota+1}, \ldots, \lambda_\kappa$ starting in the same state p, such that the highest rank appearing on λ_i has the same parity as i and it is higher than the highest rank on λ_{i-1} for $i = \iota, \iota+1, \ldots, \kappa$.

A *weak* (ι,κ)-*flower* is a sequence of loops $\lambda_\iota, \lambda_{\iota+1} \ldots, \lambda_\kappa$ such that λ_{i+1} is reachable from λ_i, and λ_i is accepting iff i is even.

A *split* is a pair of loops $p \xrightarrow{\sigma,0} p_0 \longrightarrow \ldots \longrightarrow p$ and $p \xrightarrow{\sigma,1} p_1 \longrightarrow \ldots \longrightarrow p$ such that the highest ranks occurring on them are of different parity and the highest one is odd.

A state q is *replicated* by a loop $p \xrightarrow{\sigma,d_0} p_0 \longrightarrow \ldots \longrightarrow p$ if there exists a path $p \xrightarrow{\sigma,d_1} p_1 \longrightarrow \ldots \longrightarrow q$ such that $d_0 \neq d_1$. We will say that a loop or a flower is replicated by a loop λ if it contains a state replicated by λ.

Proposition 4.2 (Niwiński, Walukiewicz [15]; Murlak [11])**.** *Let A be a deterministic automaton.*

   (1) *$L(A) \in \Pi^0_1$ iff A contains no weak $(1,2)$-flower.*
   (2) *$L(A) \in \Sigma^0_1$ iff A contains no weak $(0,1)$-flower.*
   (3) *$L(A) \in \Pi^0_2$ iff A contains no $(0,1)$-flower.*
   (4) *$L(A) \in \Sigma^0_2$ iff A contains neither $(1,2)$-flower nor a weak $(1,2)$-flower replicated by an accepting loop.*
   (5) *$L(A) \in \Sigma^0_3$ iff A contains no $(0,1)$-flower replicated by an accepting loop.*
   (6) *$L(A) \in \Pi^0_3$ iff A contains no split.*

In particular, the Borel rank of $L(A)$ is computable within the time of finding the productive states of A.

The patterns defined above were originally introduced to capture the index complexity of recognizable languages. Niwiński and Walukiewicz used flowers to solve the deterministic index problem for word languages [14]. Their result may easily be adapted to trees (see [11] for details).

Theorem 4.3. *For a deterministic tree automaton A the language $L(A)$ is recognized by a deterministic (ι,κ)-automaton iff A does not contain a $\overline{(\iota,\kappa)}$-flower. An equivalent minimal index automaton with the same number of states can be constructed within the time of solving the emptiness problem.*

The weak flowers provide an analogous characterisation of the weak deterministic index.

Proposition 4.4 ([11])**.** *A deterministic automaton A is equivalent to a weak deterministic (ι,κ)-automaton iff it does not contain a weak $\overline{(\iota,\kappa)}$-flower. An equivalent minimal index automaton with the same number of states can be constructed within the time of solving the emptiness problem.*

Proof. If the automaton contains a weak (ι,κ)-flower, for each weak $\overline{(\iota,\kappa)}$-automaton one can build a cheating tree (see [11] for details). For the converse implication, construct a weak deterministic (ι,κ)-automaton by modifying the ranks of the given deterministic automaton. Set rank q to the lowest number m such that there exists a weak (m,κ)-flower with a path from q to λ_m. $\blacksquare$

5. The Power of the Weak

In this section we finally turn to the weak recognizability of deterministic languages. First we give sufficient conditions for a deterministic automaton to be equivalent to a weak alternating automaton of index $(0,2)$, $(1,3)$, and $(1,4)$. This is the first step to the solution of the weak index problem for deterministic automata.

Proposition 5.1. *For each deterministic $(1,2)$-automaton with n states one can construct an equivalent weak $(0,2)$-automaton with $2n+1$ states.*

Proof. Fix a deterministic $(1,2)$-automaton A. We will construct a weak $(0,2)$-automaton B such that $L(A) = L(B)$. Basically, for each node v the automaton B should check whether on each path in the subtree rooted in v the automaton A will reach a state with rank 2. This can be done as follows. Take two copies of A. In the first copy, all states are universal and have rank 0. The transitions are like in A plus for each state $q^{(1)}$ there is an ε-transition to $q^{(2)}$, the counterpart of $q^{(1)}$ in the second copy. In the second copy all states are universal and have rank 1. For the states with rank 1 in A, the transitions are like in A. For the states with rank 2 in A, there is just one transition to an all-accepting state $\top$ (rank 2 in B). ∎

Before we proceed with the conditions, let us show a useful property of the replication.

Lemma 5.2 (Replication Lemma). *A state occurs in infinitely many incomparable nodes of an accepting run iff it is productive and is replicated by an accepting loop.*

Proof. If a state p is replicated by an accepting loop, then by productivity one may easily construct an accepting run with infinitely many incomparable occurrences of p. Let us concentrate on the converse implication.

Let p occur in an infinite number of incomparable nodes $v_0, v_1, \ldots$ of an accepting run 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. Since v_i are incomparable, v_i is not on π. Let the word α_i be the sequence of states labeling the path from the last common node of π and π_i to v_i. Cutting the loops off if needed, we may assume that $|\alpha_i| \leq |Q|$ for all $i \in \omega$. Consequently, there exist a word α repeating infinitely often in the sequence $\alpha_0, \alpha_1, \ldots$. Moreover, the path π is accepting, so the starting state of α must lay on an accepting productive loop. This loop replicates p. ∎

Proposition 5.3. *For each deterministic $(0,1)$-automaton with n states which contains no weak $(1,2)$-flower replicated by an accepting loop one can construct effectively an equivalent weak $(1,3)$-automaton with $3n+1$ states.*

Proof. Let A be a deterministic $(0,1)$-automaton which contains no weak $(1,2)$-flower replicated by an accepting loop. Let us call a state of A *relevant* if it has the highest rank on some loop. We may change the ranks of productive irrelevant states to 0, and assume from now on that all odd states are relevant. 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 must occur in infinitely many incomparable nodes (otherwise we would get a rejecting path). By the Replication Lemma p is replicated by an accepting loop. As p is odd and relevant, it lies on some nontrivial rejecting loop. Since p is also productive, some accepting loop can be reached from p. Hence, A contains a weak $(1,2)$-flower replicated by an accepting loop - a contradiction

Now, we can easily construct a weak $(1,3)$-automaton recognising $L(A)$. Intuitively, we will simulate A and check if A's odd states occur finitely many times. This can be done as follows. Take three copies of A. In the first copy all the states are universal and have rank 1. The transitions are just like in A, only they go to the second copy of A. In the second copy of A, all the states are existential and have rank 1. From each state $q^{(2)}$ there are two ε-transitions to $q^{(1)}$ in the first copy and to $q^{(3)}$ in the third copy. Finally, in the third copy of A all the states are universal and have rank 2. The transitions from the states ranked 0 in A are just like in A, and from the states ranked 1 in A they go to an all-rejecting state $\perp$ (rank 3 in B). It is easy to see that B recognizes $L(A)$.   ∎

Proposition 5.4. *For each automaton with n states containing no $(0,1)$-flower replicated by an accepting loop one can construct an equivalent weak alternating $(1,4)$-automaton with $\mathcal{O}(n^2)$ states.*

Proof. Let A be an automaton without $(0,1)$-flower replicated by an accepting loop. Consider the DAG of strongly connected components of A. For each SCC X containing at least one loop we will construct a weak automaton B_X recognising the languages of trees t such that each path of A's run on t that enters X either leaves X or is accepting. Obviously, the conjunction of such automata recognizes exactly $L(A)$. Let us first consider components replicated by an accepting loop. By the hypothesis, such a component must not contain a $(0,1)$-flower. Therefore we may assume that X only uses ranks 1 and 2. To obtain B_X take a copy of A. The states outside X can be divided into three disjoint groups: those that can be reached from X, those from which X can be reached, and the rest. Give the states from the first group the rank 4, and the states from the second and third group the rank 2. Finally, following the method from Proposition 5.1, replace X with an equivalent weak alternating subautomaton using ranks 2,3, and 4. The constructed automaton has $\mathcal{O}(n)$ states.

The case of X not replicated by an accepting loop is more tricky. The key property follows from the Replication Lemma. Let ρ_X denote the restriction of the run ρ to the nodes labeled with a state from X or having a descendant labeled with a state from X. By the Replication Lemma, this tree has only finitely many branches (some of them may be infinite). What B_X should do is to guess a node v on each path such that in the subtree rooted in v, ρ_X is either empty or consists of one infinite accepting branch. In the latter case we may additionally demand that on this infinite path the highest rank that ever occurs, occurs infinitely many times.

B_X consists of the component C_{guess} realising the guessing, the component $C_{A \setminus X}$ checking that no path of the computation enters X, and components $C_{X,r}$ for all ranks r used in X, which check that in a given subtree of the run ρ there is exactly one branch of ρ_X and that on this branch r occurs infinitely often and no higher rank is used.

To construct C_{guess}, take a copy of A and declare all the states universal and set their ranks to 1. For each q add a fresh existential state q' of rank 1 with an ε-transition to q and either to $q^{A \setminus X} \in C_{A \setminus X}$ if $q \notin X$ (ρ_X is empty) or to $q^{X,r} \in C_{X,r}$ for all r if $q \in X$ (ρ_X is one infinite accepting path). Finally replace each transition $p \xrightarrow{\sigma} p_0, p_1$ with $\xrightarrow{\sigma} p_0', p_1'$.

The component $C_{A \setminus X}$ is a copy of A with all ranks equal 2, and the SCC X replaced with one all-rejecting state $\perp$ with rank 3.

Finally, let us now describe the automaton $C_{X,r}$. The automaton, staying in rank 2, works its way down the input tree just like A would, with the following modifications:

$$\Sigma_0^0 = (1,1) \qquad\qquad \Sigma_1^0 = (1,2) \qquad\qquad \Sigma_2^0 = (1,3)$$

$$\Sigma_1^0 \cap \Pi_1^0 \qquad\qquad \Sigma_2^0 \cap \Pi_2^0 \qquad\qquad \Sigma_3^0 \cap \Pi_3^0$$
$$\|\qquad\qquad\qquad \|\qquad\qquad\qquad \|$$
$$(1,2) \cap (0,1) \qquad\qquad (1,3) \cap (0,2) \qquad\qquad (1,4) \cap (0,3)$$

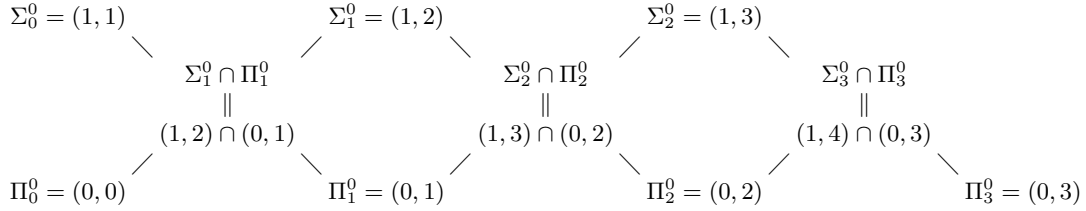$$\Pi_0^0 = (0,0) \qquad\qquad \Pi_1^0 = (0,1) \qquad\qquad \Pi_2^0 = (0,2) \qquad\qquad \Pi_3^0 = (0,3)$$

Figure 4: For deterministic tree languages the hierarchies coincide.

- if A enters a state in X with rank greater than r, $C_{X,r}$ moves to an all rejecting state $\perp$ (rank 3),
- if A takes a transition exiting X on both branches or staying in X on both branches, $C_{X,r}$ moves to $\perp$,
- if A takes a transition whose left branch leaves X and the right branch stays inside, $C_{X,r}$ sends to the right a $(3,4)$-component looking for a state from X with the rank r, and moves on to the right subtree (and symmetrically).

In order two see that $C_{X,r}$ does the job, it is enough to observe that if the $(3,4)$ component always succeeds to find a state from X with the rank r, then on the unique path that stays forever in X the rank r repeats infinitely often.

The $(3,4)$-component of $C_{X,r}$ can be constructed in such a way that it has $|X| + 2$ states, and so in this case B_X has at most $2|X|(|X| + 2) + 3n \leq 2|X|^2 + 7n$ states.

In both cases, the number of states of B_X can be bounded by $c_1|X|^2 + c_2 n$ for fixed constants c_1 and c_2, independent of X. Since the SCCs are disjoint, the number of states of the conjunction of B_X's is at most

$$1 + \sum_{X \in A} (c_1|X|^2 + c_2 n) \leq 1 + c_1 \left(\sum_{X \in A} |X| \right)^2 + c_2 n^2 \leq (c_1 + c_2)n^2 + 1 \,.$$

$\blacksquare$

We have now collected all the ingredients for the solution of the weak index problem for deterministic languages. What is left to be done is to glue together the sufficient conditions for index easiness and Borel hardness using Corollary 3.3.

Theorem 5.5. *For deterministic languages the Borel hierarchy and the weak index hierarchy coincide (Fig. 4) and are decidable within the time of solving emptiness problem. For a deterministic automaton with n states, an equivalent minimal index automaton with $\mathcal{O}(n^2)$ states can be constructed effectively within the time of solving the emptiness problem.*

Proof. We will abuse the notation and write (ι, κ) to denote the class of languages recognized by weak (ι, κ)-automata. All the classes considered here are relativised to the deterministic languages.

By the two versions of the Gap Theorem we have the equality and decidability of the classes of the classes Π_3^0 and $(0,3)$.

Let us continue with the third level. Let us see that $\Sigma_3^0 = (1,4)$. We will show that both these classes are equal to the class of languages recognized by deterministic automata without a $(0,1)$-flower replicated by an accessible loop. If a deterministic automaton A does not contain this pattern, then it is equivalent to a weak $(1,4)$-automaton and by Corollary 3.3 recognizes a Σ_3^0 language. If A does contain this pattern, then by Proposition 4.2 it is

not Σ_3^0 and so is not equivalent to a weak $(1,4)$-automaton. The decidability follows easily, since checking for the pattern above can be done effectively (in polynomial time).

For the equality $\Pi_2^0 = (0,2)$, prove that both classes are equal to the class of languages recognized by deterministic automata without a $(0,1)$-flower. Proceed just like before, only use Proposition 5.1 instead of Proposition 5.4. Analogously, using Proposition 5.3, show that both Σ_2^0 and $(1,3)$ are equal to the class of languages recognized by deterministic automata admitting neither a $(1,2)$-flower nor a weak $(1,0)$-flower replicated by an accepting loop.

For the first level use the characterisation given by Proposition 4.4. The level zero is trivial. ∎

Acknowledgments

References

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

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

[3] A. Arnold, D. Niwiński. Continuous separation of game languages. Manuscript, submitted, 2006.

[4] J. C. Bradfield. The modal mu-calculus alternation hierarchy is strict. Theoret. Comput. Sci. **195** (1998) 133–153.

[5] J. Duparc, F. Murlak. On the topological complexity of weakly recognizable tree languages. Proc. FCT 2007, LNCS 4639 (2007) 261-273.

[6] E. A. Emerson, C. S. Jutla. The complexity of tree automata and logics of programs. *Proc. FoCS '88*, IEEE Computer Society Press 1988, 328–337.

[7] O. Kupferman, M. Vardi. The weakness of self-complementation. *Proc. STACS '99*, LNCS 1563 (1999) 455–466.

[8] G. Lenzi. A hierarchy theorem for the mu-calculus. *Proc. ICALP '96*, LNCS 1099 (1996) 87–109.

[9] A. W. Mostowski. Hierarchies of weak automata and weak monadic formulas. Theoret. Comput. Sci. **83** (1991) 323-335.

[10] D. E. Muller, A. Saoudi, P. E. Schupp. Alternating automata. The weak monadic theory of the tree, and its complexity. *Proc. ICALP '86*, LNCS 226 (1986) 275–283.

[11] F. Murlak. On deciding topological classes of deterministic tree languages. *Proc. CSL '05*, LNCS 3634 (2005) 428–441.

[12] J. Neumann, A. Szepietowski, I. Walukiewicz. Complexity of weak acceptance conditions in tree automata. IPL 84 (2002) 181–187.

[13] D. Niwiński. On fixed point clones. *Proc. ICALP '86*, LNCS 226 (1986) 464–473.

[14] D. Niwiński, I. Walukiewicz. Relating hierarchies of word and tree automata. *Proc. STACS '98*, LNCS 1373 (1998) 320–331.

[15] D. Niwiński, I. Walukiewicz. A gap property of deterministic tree languages. Theoret. Comput. Sci. **303** (2003) 215–231.

[16] D. Niwiński, I. Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Proc. WoLLiC '04*, Electronic Notes in Theoret. Comp. Sci. 2005, 195–208.

[17] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. Trans. Amer. Soc. **141** (1969) 1–35.

[18] M. O. Rabin. Weakly definable relations and special automata. *Mathematical Logic and Foundations of Set Theory*, North-Holland 1970, 1–70.

[19] J. Skurczyński. The Borel hierarchy is infinite in the class of regular sets of trees. Theoret. Comput. Sci. **112** (1993) 413–418.

[20] W. Thomas. A hierarchy of sets of infinite trees. *Proc. Theoretical Computer Science*, LNCS 145 (1982) 335–342.

[21] K. Wagner. On ω-regular sets. Inform. and Control **43** (1979) 123–177.