Complexity of weak acceptance conditions in tree automata

Jakub Neumann[1], Andrzej Szepietowski[1], Igor Walukiewicz[2]

Abstract

Weak acceptance conditions for automata on infinite words or trees are defined in terms of the set of states that appear in the run. This is in contrast with, more usual, strong conditions that are defined in terms of states appearing infinitely often on the run. Weak conditions appear in the context of model-checking and translations of logical formalisms to automata. We study the complexity of the emptiness problem for tree automata with weak conditions. We also study the translations between automata with weak and strong conditions.

1 Introduction

An acceptance condition of an automaton with a set S of states is a set of infinite sequences $Acc \subseteq S^\omega$. A weak acceptance condition is a condition that does not distinguish between two sequences with the same states in them. In contrast, a more usual strong acceptance condition does not distinguish between two sequences which have the same states appearing infinitely often. Similarly as for strong conditions one can define Muller, Rabin, Streett and Mostowski weak conditions.

We study the complexity of deciding the emptiness problem of automata with weak conditions. The problem is PTIME-complete for Mostowski conditions and PSPACE-complete for Rabin and Streett conditions. This can be

[1]Institute of Mathematics University of Gdańsk ul Wita Stwosza 57, 80952 Gdańsk, Poland

[2]LaBRI, Université Bordeaux I, Domaine Universitaire, batiment A30, 351, cours de la Libération, 33405 Talence Cedex, France

compared with the fact that the problem for strong Rabin and Streett conditions is NP-complete and coNP-complete respectively [2]. Given this result we study the complexity of translations from automata with weak conditions to automata with strong conditions. We show that in some cases, like that of Rabin and Streett conditions, the translations need to be exponential.

Weak acceptance conditions were first considered by Staiger and Wagner [6] for automata on infinite sequences. Weak conditions for automata on infinite trees were considered in [11], [7], [9], and [10]. In particular Mostowski [7] shows a direct correspondence between the index of a weak condition and the alternation depth of weak second order quantifiers. Skurczyński [10] considered the relations between tree automata with weak condition and some restricted versions of branching temporal logic.

A related notion of weak automata was introduced by Muller et. al. [8]. It is easy to see that weak automata are equivalent to automata with weak Mostowski conditions (see Section 4). In [8] weak alternating automata were used to give simpler proofs of the complexity bounds for the satisfiability problem of some temporal and dynamic logics. This approach was then extended in [5] also to model-checking problems.

2 Preliminaries

2.1 Automata on trees

The set $\{0,1\}^*$ of all finite strings over alphabet $\{0,1\}$ may be viewed as an infinite binary tree where the root node is the empty string ε. Let Σ be a finite alphabet. A Σ-*tree* is a function $T : \{0,1\}^* \to \Sigma$.

A *finite automaton* $\mathcal{A}$ on Σ-trees is a tuple $\langle \Sigma, S, \delta, s^0, Acc\rangle$, where: S is the finite set of states of the automaton, $\delta : S \times \Sigma \to \mathcal{P}(S \times S)$ is the (nondeterministic) transition function, s^0 is the initial state of the automaton, $Acc \subseteq S^\omega$ is the acceptance condition (here S^ω is the set of all infinite strings over S, i.e., functions $\mathbb{N} \to S$.)

A *run* of $\mathcal{A}$ on a Σ-tree T is a function $r : \{0,1\}^* \to S$ such that: $r(\varepsilon) = s^0$ and for all $x \in \{0,1\}^*$, $(r(x0), r(x1)) \in \delta(r(x), T(x))$. The run is *accepting* iff for every path $\varepsilon, v_1, v_2, \ldots$ in T the sequence $r(\varepsilon)r(v_1)r(v_2)\ldots$ is in Acc. A tree T is *accepted* by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on T. By $L(\mathcal{A})$ we denote the set of trees accepted by $\mathcal{A}$.

2.2 Types of acceptance conditions

The most common way to define an acceptance condition $Acc \subseteq S^\omega$ is to put some restrictions on states appearing infinitely often in the sequences. Here we call such conditions *strong conditions*. In contrast, *weak conditions* put restrictions on all the states that appear in the sequence. More formally, if we have an infinite sequence of states $\sigma : \mathbb{N} \to S$ then we can define two sets:

$$\begin{aligned} \sigma(\mathbb{N}) =& \{s \in S \mid \sigma(i) = s \text{ for some } i \in \mathbb{N}\} \\ \mathrm{Inf}(\sigma) =& \{s \in S \mid \sigma(i) = s \text{ for infinitely many } i \in \mathbb{N}\} \end{aligned}$$

Strong conditions define a set of sequences σ through the properties of the set $\mathrm{Inf}(\sigma)$. Weak conditions refer to the properties of $\sigma(\mathbb{N})$. Here are the weak forms of some standard [12, 13] acceptance conditions:

weak Muller condition is specified by a set $\mathcal{F} \subseteq \mathcal{P}(S)$. We have $Acc = \{\sigma : \sigma(\mathbb{N}) \in \mathcal{F}\}$.

weak Rabin condition is specified by a set $\{(R_1, G_1), \ldots, (R_k, G_k)\}$, where $R_i, G_i \subseteq S$. We have $Acc = \{\sigma : \exists i.\ \sigma(\mathbb{N}) \cap R_i = \emptyset \text{ and } \sigma(\mathbb{N}) \cap G_i \neq \emptyset\}$

weak Streett condition is also specified by a set $\{(R_1, G_1), \ldots, (R_k, G_k)\}$, where $R_i, G_i \subseteq S$. We have $Acc = \{\sigma : \forall i.\ \sigma(\mathbb{N}) \cap R_i = \emptyset \text{ or } \sigma(\mathbb{N}) \cap G_i \neq \emptyset\}$

weak Mostowski condition Is specified by a function $\Omega : S \to \mathbb{N}$. We have $Acc = \{\sigma : \min(\sigma(\mathbb{N})) \text{ is even}\}$

Rabin condition is sometimes called "pairs condition"; Streett condition is called "complemented pairs condition"; Mostowski condition is called "parity condition".

A *weak automaton* is a Mostowski automaton $\mathcal{A} = \langle \Sigma, S, \delta, s^0, \Omega : S \to \{0, 1\} \rangle$ for which there exists a quasi-order (reflexive and transitive relation) on states $\preccurlyeq$ such that: (i) whenever $(s_0, s_1) \in \delta(s, a)$ then $s_0 \preccurlyeq s$ and $s_1 \preccurlyeq s$; (ii) if $s \preccurlyeq s'$ and $s' \preccurlyeq s$ then $\Omega(s) = \Omega(s')$. Observe that on each path a run of $\mathcal{A}$ is eventually trapped in some equivalence class defined by $\preccurlyeq$. Acceptance is then determined according to the value of Ω for this class (Ω has the same value on all the states in the same class).

2.3 Games

A *game* $G = \langle V, V_0, V_1, E \subseteq V \times V, Acc \subseteq V_0^\omega \rangle$ is a bipartite labelled graph with the partition $V_0, V_1 \subseteq V$. We say that a vertex v' is a *successor* of a vertex v if $E(v, v')$ holds. The set Acc defines the winning condition of the game.

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 $v_0, v_1, v_2, \dots$ This *path is winning* for player 0 if the projection of the sequence to V_0 belongs to Acc. Otherwise player 1 is the winner.

A *strategy* σ for player 0 is a function assigning to every sequence of vertices $\vec{v}$ ending in a vertex v from V_0 a vertex $\sigma(\vec{v}) \in V_1$ which is a successor of v. A strategy is *memoryless* iff $\sigma(\vec{v}) = \sigma(\vec{w})$ whenever $\vec{v}$ and $\vec{w}$ end in the same vertex. A *strategy is winning* from a vertex v iff it guarantees a win for player 0 whenever he follows the strategy. Similarly we define a strategy for player 1. We will often consider strategies which are partial functions. To fit our definition one can assume that these are total functions whose values for some elements don't matter.

A nondeterministic automaton $\mathcal{A} = \langle \Sigma, S, \delta, s^0, Acc \rangle$ defines a game $G(\mathcal{A}) = \langle S \cup \delta, S, \delta, E, Acc \rangle$, where the set of vertices of player 0 is the set of states of the automaton; the set of vertices of player 1 is the set of its transitions; Acc consists of those sequences over $S \cup \delta$ which projection on S is accepted by the acceptance condition of the automaton; finally E is defined by:

- $(s, (s, a, s', s'')) \in E$ if $(s', s'') \in \delta(s, a)$,
- $((s, a, s', s''), s') \in E$ and $((s, a, s', s''), s'') \in E$.

Immediately from the definitions we get the following two facts relating automata and games.

Fact 1 For an automaton $\mathcal{A}$ with an initial state s^0: $L(\mathcal{A}) \neq \emptyset$ iff there is a strategy for player 0 in $G(\mathcal{A})$ which is winning from the vertex s^0.

Fact 2 For every finite game G and its vertex v there is an automaton $\mathcal{A}(G, v)$ over one letter alphabet such that: $L(\mathcal{A}(G, v)) \neq \emptyset$ iff player 0 has winning strategy from vertex v in G. The size of $\mathcal{A}(G, v)$ is linear in the size of G. The acceptance condition of $\mathcal{A}(G, v)$ is of the same kind as the winning condition in G.

3 The emptiness problem

The emptiness problem for tree automata is: "given $\mathcal{A}$ decide if $L(\mathcal{A}) \neq \emptyset$". In this section we will establish the computational complexity of this problem for automata with various kinds of weak accepting conditions. By the facts mentioned above the emptiness problem is linear time equivalent to finding a winner in respective games.

Fact 3 The emptiness problem is PTIME-complete for automata with weak parity conditions.

Proof
PTIME-hardness is easy by reduction of the alternating reachability problem (see [3]).

For the membership in PTIME it is enough, by Fact 1, to show an algorithm for deciding a winner in parity games with weak parity conditions. Let $G = \langle V, V_0, V_1, E, \Omega \rangle$ be a game where $\Omega : V_0 \to \mathbb{N}$ is the function defining a weak parity condition. Without a loss of generality we can assume that the range of Ω is $\{0, \ldots, n\}$ for some n. For each $i = 0, \ldots, n$ the algorithm calculates the set of vertices B_i such that: $v \in B_i$ iff $\Omega(v) = i$ and there is a strategy for player 0 which guarantees that every play starting from v satisfies:

- for even i: if the play reaches a vertex outside $\{v' : \Omega(v') \geq i\}$ then the first such vertex must belong to $\bigcup_{j<i} B_j$;
- for odd i: the play must reach a vertex outside $\{v' : \Omega(v') \geq i\}$, and the first such vertex must belong to $\bigcup_{j<i} B_j$.

So for each i we need to solve an alternating reachability problem (for even i we rather calculate the complement of B_i). One can check that there is a winning strategy from a vertex v iff $v \in B_{\Omega(v)}$. □
□

Theorem 4
The emptiness problem is PSPACE-complete for automata with weak Rabin and weak Streett conditions.

The theorem is a consequence of the two following facts.

Fact 5 A problem of establishing a winner in a game with weak acceptance condition (Muller, Rabin, or Streett) can be solved by a deterministic $\Omega(n^2)$ space bounded Turing machine, where n is the number of positions in the game.

Proof
Let $G = \langle V, V_0, V_1, E, Acc\rangle$ be a game and let $Acc = Acc_{\mathcal{F}}$ be a Muller acceptance condition defined by a set $\mathcal{F} \subseteq \mathcal{P}(V_0)$. Observe that our algorithm will not depend on the size of representation of Acc.

For a set $B \subseteq V_0$ denote by Acc_B the Muller condition defined by the set $\mathcal{F}_B = \{S : S \cup B \in Acc\}$. For each $B \subseteq V_0$ define the set $\text{Accept}(B)$ consisting of the vertices v from which player 0 has a winning strategy in G with the winning condition Acc_B. In other words, we consider plays from v but we assume that B is the set of vertices that was already visited on the way to v.

We have that $v \in \text{Accept}(\emptyset)$ if player 0 has a winning strategy in the game G. We also have that $v \in \text{Accept}(V_0)$ iff $V_0 \in Acc$ and there is strategy allowing player 0 to always make a move.

We want to show an algorithm to compute $\text{Accept}(B)$ for all sets $B \subseteq V_0$. As noted above, calculating $\text{Accept}(V_0)$ is equivalent to an alternating reachability problem, so it can be solved in linear time and space. To decide if $v \in \text{Accept}(B)$ for some other B, we first calculate the set $F = \{v' \in V_0 \setminus B : v' \in \text{Accept}(B \cup \{v'\})\}$. Then we proceed as follows:

- If $B \notin \mathcal{F}$ then $v \in \text{Accept}(B)$ holds iff player 0 has a strategy from v to reach a vertex outside B and with the first such vertex being in F.

- If $B \in \mathcal{F}$ then $v \in \text{Accept}(B)$ holds iff player 0 has a strategy from v to stay in vertices in B or to leave to some vertex in F.

It is not difficult to check that the algorithm is correct.

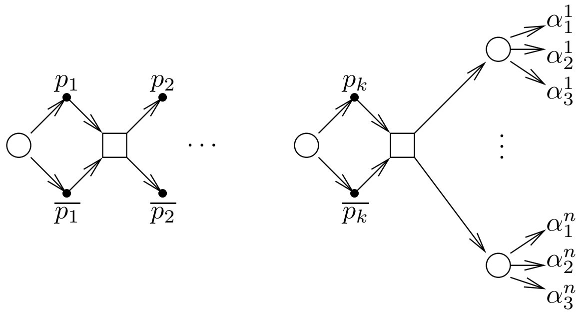
Calculating $\text{Accept}(B)$ is an alternating reachability problem, once we know F. We can calculate F using recursion. As the depth of recursion is bounded by the size of V_0, we can calculate $\text{Accept}(\emptyset)$ in the space $\mathcal{O}(|V|^2)$.
$\Box$ $\Box$

Fact 6 The problem of finding a winner in games with weak Rabin or Streett conditions is PSPACE-hard.

Proof
We reduce the QBF (Quantified Boolean Formulas) problem. Let $X = \{x_1, \overline{x}_1, \ldots, x_k, \overline{x}_k\}$ be a set of literals and let

$$\exists x_1.\forall x_2 \ldots \exists x_k.\ (\alpha_1^1 \vee \alpha_2^1 \vee \alpha_3^1) \wedge \cdots \wedge (\alpha_1^n \vee \alpha_2^n \vee \alpha_3^n)$$

be a QBF formula where each $\alpha_j^i \in X$ is a literal. Consider the game:



The vertices for player 0 and player 1 are denoted by circles and squares respectively. The rest of the vertices has degree 1 so it does not matter who makes a move in them. Moreover there are self loops, that are not shown, in each of the vertices α_j^i. The graph of the game is not bipartite but it can be easily made so by introducing some dummy vertices of degree 1.

The intuition behind the game is easy. The players choose a valuation of variables (who chooses a value depends on the quantifier binding the variable in the formula). Then, player 1 points to a clause which he thinks is not satisfied in the chosen valuation. Finally, player 0 has to show that some literal from the chosen clause belongs to the chosen valuation.

The relevant Rabin condition is:

$$(\{\overline{p_1}\}, \{x_1\}), (\{p_1\}, \{\overline{x_1}\}), \ldots, (\{\overline{p_k}\}, \{x_k\}), (\{p_k\}, \{\overline{x_k}\})$$

The condition says that if x_i is chosen in the play then the play does not go through $\overline{p}_i$; and similarly for $\overline{x}_i$. Observe that on every play there is only one appearance of a literal, i.e., an element of X. It should be clear that player 0 has a winning strategy iff the given formula is satisfiable.

The relevant Streett condition is:

$$\{(\{p_i\}, X \setminus \{x_i\}), (\{\overline{p}_i\}, X \setminus \{\overline{x}_i\}) : i = 1, \ldots, k\}$$

The condition says that if p_i does not appear in the play then x_i cannot appear in the play; and similarly for $\overline{p}_i$. It should be clear that player 0 has a winning strategy in the game with this condition iff the given formula is satisfiable. □ □

Corollary 7 Unlike the case of games with strong Rabin conditions [1, 4], the memoryless determinacy theorem does not hold for games with weak Rabin conditions.

Proof
In the reduction above consider the formula $\forall x_2.\exists x_3.\ x_2 \Leftrightarrow x_3$. Then in the above game, player 0 has a winning strategy, but in this strategy he has to choose p_3 if player 1 has chosen p_2 and to choose $\overline{p}_3$ otherwise. □ □

4 Translations

In this section we will consider the translations between weak automata and automata with weak conditions of various types. The translation from weak automata to automata with weak parity conditions is straightforward and does not produce any blowup.

Fact 8 For every weak automaton $\mathcal{A}$ there is an equivalent automaton with weak parity conditions with the same set of states as $\mathcal{A}$.

The translations in the opposite direction are slightly more difficult. Later we will show that the blowups in the given translations are in general necessary.

Fact 9 Let $\mathcal{A}$ be an automaton with a set of states S and a winning condition *Acc*.

1. If *Acc* is a weak Muller condition then there is an equivalent weak automaton $\mathcal{A}'$ of size $|S|2^{|S|}$.

2. If *Acc* is a weak parity condition with the range $\{i, \ldots, j\}$ then there is an equivalent weak automaton $\mathcal{A}'$ of size $|S|(j - i + 1)$.

3. If *Acc* is a weak Rabin or Streett condition given by a set of pairs $\{(R_i, G_i)\}_{i=1,\ldots,k}$ then there is an equivalent weak automaton $\mathcal{A}'$ of size $|S|3^k$.

Proof
For the first statement, about a Muller condition $\mathcal{F} \subseteq \mathcal{P}(S)$, we construct an automaton $\mathcal{A}'$ with the set of states $S \times \mathcal{P}(S)$. Intuitively, the second component will contain all the states seen so far. The initial state of $\mathcal{A}'$ is $(q^0, \{q^0\})$, where q^0 is the initial state of $\mathcal{A}$. We have $((q_0, B \cup \{q_0\}), (q_1, B \cup \{q_1\})) \in \delta'((q, B), a)$ iff $(q_0, q_1) \in \delta(q, a)$. We make a state (q, B) accepting if $B \in \mathcal{F}$. It is easy to show that $\mathcal{A}'$ is a weak automaton and that $L(\mathcal{A}) = L(\mathcal{A}')$.

In order to prove the second and the third statements of the fact we will show how to identify some of the states of the automaton $\mathcal{A}'$.

If the accepting condition of $\mathcal{A}$ is a parity condition $\Omega : S \to \mathbb{N}$ then for the automaton $\mathcal{A}'$ we have that (q, B) is accepting iff $\min(B) = \min\{\Omega(q) : q \in B\}$ is even. Because $\min(\min(B), \Omega(q)) = \min(B \cup \{q\})$ it is enough to know $\min(B)$ and q to calculate $\min(B \cup \{q\})$. So in the case of the parity condition we can reduce the set of states of $\mathcal{A}'$ to $S \times \{i, \dots, j\}$ where $\{i, \dots, j\}$ is the range of Ω. The second component is used to keep the current minimum.

If the accepting condition is a Rabin condition $\{(R_i, G_i)\}_{i=1,\dots,k}$ then a state (q, B) of $\mathcal{A}'$ is accepting iff there is i such that $B \cap R_i = \emptyset$ and $B \cap G_i \neq \emptyset$. Consider the function $f : \mathcal{P}(S) \to (\{1, \dots, k\} \to \{0, 1, \bot\})$ defined by

$$f(B)(i) = \begin{cases} 0 & \text{if } B \cap (R_i \cup G_i) = \emptyset \\ \bot & \text{if } B \cap R_i \neq \emptyset \\ 1 & \text{otherwise} \end{cases}$$

We have that (q, B) is an accepting state of $\mathcal{A}'$ iff $f(B)(i) = 1$ for some i. It is not difficult to define a function Update such that $f(B \cup \{q\}) = \text{Update}(f(B), q)$. So we can reduce the set of states of $\mathcal{A}'$ to $S \times (\{1, \dots, k\} \to \{0, 1, \bot\})$.

Finally, the case of Streett condition $\{(R_i, G_i)\}_{i=1,\dots,k}$ is very similar but now we define

$$f(B)(i) = \begin{cases} 0 & \text{if } B \cap (R_i \cup G_i) = \emptyset \\ 1 & \text{if } B \cap G_i \neq \emptyset \\ \bot & \text{otherwise} \end{cases}$$

We have that (q, B) is accepting iff $f(B)(i) \neq \bot$ for all i. The update function is also easily definable in this case. $\square$ $\square$

It is easy to show that for every n there is a language L_n which is recognized by a $\mathcal{O}(n)$ size automaton with weak Muller conditions but not recognized by any automaton with strong conditions with less than 2^n states. Below we show that the exponential blowup is also necessary in the case of Rabin and Streett conditions.

Fact 10 For every n there are languages L_n^R and L_n^S such that:

- there is an $\mathcal{O}(n)$ size automaton with weak Rabin (resp. Streett) conditions accepting L_n^R (resp. L_n^S).
- every automaton with strong conditions recognizing L_n^R or L_n^S has $\Omega(2^n)$ states.

Proof
Fix $n \in \mathbb{N}$ and consider the alphabet $\Sigma_n = \{p_i, q_i : i = 0, \ldots, n-1\} \cup \{\lambda\}$.

Let L_n^R be the set of trees over Σ_n such that for every path there is $i \in \{0, \ldots, n-1\}$ with q_i and no p_i on the path. This property can be expressed by a weak Rabin condition $\{(\{p_0\}, \{q_0\}), \ldots, (\{p_{n-1}\}, \{q_{n-1}\})\}$. It is easy to construct an automaton with $2n + 1$ states and a weak Rabin condition of the above form accepting L_n^R.

We are going to show that every automaton with strong conditions accepting L_n^R needs 2^n states. Take a subset $\Gamma \subseteq \{0, \ldots, n-1\}$. Construct a tree $T_\Gamma \in L_n^R$ with a node v on the leftmost path such that:

1. the set of labels of the nodes between the root and v is precisely $\{p_i : i \in \Gamma\}$;
2. on every path from v, the sequence of labels is of the form $\lambda^* q_i \lambda^\omega$ with $i \notin \Gamma$;
3. for every $i \notin \Gamma$ there is a node labelled q_i after v.

Let $\mathcal{A}$ be an automaton accepting L_n^R. Consider an accepting run of $\mathcal{A}$ on T_Γ and let s_Γ be the state labelling the node v in this run. It should be clear that if $\Gamma_1 \neq \Gamma_2$ then s_{Γ_1} and s_{Γ_2} must be different. This shows that there must be at least 2^n states in $\mathcal{A}$.

Let L_n^S be the set of trees of such that for every $i \in \{0, \ldots, n-1\}$: if p_i appears on the path then q_i appears on the path. This condition can be expressed by a Street condition $\{(\{p_0\}, \{q_0\}), \ldots, (\{p_{n-1}\}, \{q_{n-1}\})\}$. This is the same set of pairs as before but now considered as a Streett condition. Similarly to the case of Rabin conditions one can show that there is an automaton recognizing L_n that has weak Streett conditions and $2n+1$ states. Also a similar argument shows that every automaton with strong conditions recognizing the language must have 2^n states. $\square$ $\square$

The translations from Fact 9 show that if an automaton with weak conditions accepts some tree then it accepts a regular tree. Below we show that sometimes the smallest such a tree is exponentially bigger than the automaton.

Fact 11 For every n there are automata with $\mathcal{O}(n)$ states and with weak Rabin or Streett conditions such that any regular tree accepted by these automata has 2^n non-isomorphic subtrees.

Proof
Consider the alphabet $\Sigma_n = \{p_i, q_i : i = 0, \dots, n-1\} \cup \{\lambda\}$ as in the previous fact.

Let L_1 be the language of trees t over Σ such that $t(\varepsilon) = \lambda$, and for every $i = 0, \dots, n/2-1$ we have:

$$t(\{0,1\}^i 0) = p_{2i} \quad \text{and} \quad t(\{0,1\}^i 1) = p_{2i+1}$$

Let L_2 be the language of trees t such that $t(\{0,1\}^{n/2}0^i) \in \{q_{2i}, q_{2i+1}\}$ (for $i = 0, \dots, n/2-1$) and $t(v) \in \{p_0, \dots, p_{n-1}\}$ for all other nodes v.

Consider the language L_n^S from the previous lemma. It is not difficult to construct an automaton recognizing $L_n^S \cap L_1 \cap L_2$ that has $5n$ states and a weak Streett condition with n pairs. It is also not difficult to check that for every tree t in this language, if $v, v' \in \{0,1\}^{n/2}$ are two different nodes at level $n/2$ then the subtrees of t rooted at v and v' must be different. This settles the case of Streett conditions.

Let L_3 be the language of trees t over Σ such that $t(\{0,1\}^{n/2}0^i1) \in \{q_{2i}, q_{2i+1}\}$ (for $i = 0, \dots, n/2-1$) and $t(v) \in \{p_0, \dots, p_{n-1}\}$ for all other nodes v. Let K_n^R consist of trees t such that for every path P of t: either only the letters from $\{\lambda, p_0, \dots, p_{n-1}\}$ appear in P or there is i such that q_i appears on P and no p_i appears on P. This language is very similar to the language L_n^R from the previous lemma, but here we allow also paths where no q_i appears. It is not difficult to find an automaton recognizing $K_n^R \cap L_1 \cap L_3$ that has $5n$ states and a weak Rabin condition with $n+1$ pairs. The same argument as before shows that every tree in this language has at least $2^{n/2}$ non-isomorphic subtrees. $\square$

References

[1] E. A. Emerson. Automata, tableaux, and temporal logics. In *Workshop on Logics of Programs*, pages 79–87, 1985.

[2] E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs. In *29th IEEE Symposium on Fundation of Computer Science*, pages 368–377, 1988.

[3] N. Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22:384–406, 1981.

[4] N. Klarlund. Progress measures, immediate determinacy and a subset construction for tree automata. In *LICS '92*, pages 382–393, 1992.

[5] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[6] K. Wagner L. Staiger. Automatentheoretische und automatenfreie charakterisierungen topologischer klassen regulärer folgenmengen. *EIK*, 10:379–392, 1974.

[7] A. W. Mostowski. Hierarchies of weak automata and weak monadic formulas. *Theoretical Computer Science*, 83:323–335, 1991.

[8] D. E. Muller, A. Saoudi, and P. E. Schupp. Alternating automata, the weak monadic theory of the tree and its complexity. *TCS*, 97:233–244, 1992.

[9] J. Skurczyński. *Tree automata with weak acceptance conditions*. PhD thesis, University of Gdańsk, 1989.

[10] J. Skurczyński. A negative result for chain logic. Technical report, University of Gdańsk, 1997.

[11] S. Miyano T. Hayashi. Finite tree automata on infinite trees. *Bull. of Informatics and Cybernetics*, 21:71–82, 1985.

[12] W. Thomas. Automata on infinite objects. In J. van Leeuven, editor, *Handbook of Theoretical Computer Science Vol.B*, pages 133–192. Elsevier, 1990.

[13] W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3. SV, 1997.