Index problems for game automata

Alessandro Facchini, IDSIA, Switzerland
Filip Murlak, University of Warsaw
Michał Skrzypczak[1], University of Warsaw

For a given regular language of infinite trees, one can ask about the minimal number of priorities needed to recognize this language with a non-deterministic, alternating, or weak alternating parity automaton. These questions are known as, respectively, the non-deterministic, alternating, and weak Rabin-Mostowski index problems. Whether they can be answered effectively is a long-standing open problem, solved so far only for languages recognizable by deterministic automata (the alternating variant trivializes).

We investigate a wider class of regular languages, recognizable by so-called game automata, which can be seen as the closure of deterministic ones under complementation and composition. Game automata are known to recognize languages arbitrarily high in the alternating Rabin-Mostowski index hierarchy; that is, the alternating index problem does not trivialize any more.

Our main contribution is that all three index problems are decidable for languages recognizable by game automata. Additionally, we show that it is decidable whether a given regular language can be recognized by a game automaton.

CCS Concepts: •**Theory** → **Languages;**

Additional Key Words and Phrases: Automata over infinite trees, Alternation, Parity games, Rabin-Mostowski index

ACM Reference Format:
Alessandro Facchini, Filip Murlak, Michał Skrzypczak, 2015. Index problems for game automata. *ACM Trans. Comput. Logic* 0, 0, Article 0 (2016), 39 pages.
DOI: 0000001.0000001

1. INTRODUCTION

Finite state automata running over infinite words and infinite binary trees lie at the core of the seminal works of Büchi [Büchi 1962] and Rabin [Rabin 1969]. Known to be equivalent to the monadic second-order (MSO) logic and the modal μ-calculus on both classes of structures, they subsume all standard linear and branching temporal logics. Because of these properties, they constitute fundamental tools in the theory of verification and model-checking, where the model-checking problem is reduced to the non-emptiness problem for automata: a given formula is translated into an automaton recognizing its models. From this perspective, a natural question is, which parameter in the definition of an automaton reflects the complexity of the language recognized by it. A naïve approach is to look at the number of states; a more meaningful one is to consider the infinitary behaviour of an automaton, captured by the complexity of its acceptance condition.

Out of different acceptance conditions proposed for tree automata, Büchi, Muller, Rabin, Streett, and parity [Mostowski 1991a; Mostowski 1984], the last one has proved to be the most appropriate, as it enabled unveiling the subtle correspondences

The third author has been supported by Poland's National Science Centre (decision DEC-2012/05/N/ST6/03254).

DOI: 0000001.0000001

between games, automata, and the modal μ-calculus [Arnold and Niwiński 2001; Emerson and Jutla 1991]. In a parity automaton, each state is assigned a natural number, called its priority. A sequence of states is said to be accepting if the lowest priority occurring infinitely often is even (min-parity condition). The pair (i, j) consisting of the minimal priority i and the maximal priority j in a given automaton is called its Rabin-Mostowski index. The index of a language is the minimal index of a recognizing automaton. Practical importance of this parameter comes from the fact that the best known algorithms deciding emptiness of (non-deterministic) automata are exponential in the number of priorities.

Given a regular tree language, what is the minimal range of priorities needed to recognize it? The answer to this question depends on which mode of computation is used, i.e, whether the automata are deterministic, non-deterministic, alternating, or weak alternating. While weak alternating and deterministic automata are weaker, non-deterministic and alternating parity automata recognize all regular tree languages. Still, alternating automata often need less priorities than non-deterministic ones. Thus, for each of these four classes there is the respective index problem.

$\mathfrak{C}$ **Index Problem:** *Given i, j and a regular language L, decide if L is recognized by an automaton in class $\mathfrak{C}$ of Rabin-Mostowski index (i, j).*

The solution of this problem for the most important cases—when $\mathfrak{C}$ is the class of non-deterministic, alternating, or weak alternating automata—seems still far away. The results of [Otto 1999; Küsters and Wilke 2002; Walukiewicz 2002], later extended in [Bojańczyk and Place 2012], show that it is decidable if a given regular tree language can be recognized by a combination of reachability and safety conditions (which corresponds to the Boolean combination of open sets). It is also known that the non-deterministic (min-parity) index problem is decidable for $(i, j) = (1, 2)$, and for $(i, j) = (0, 1)$ if the input language is given by an alternating automaton of index $(1, 2)$ [Vanden Boom 2012; Kuperberg 2012; Colcombet et al. 2013]. The non-deterministic index problem has been reduced to the uniform universality problem for so-called distance-parity automata [Colcombet and Löding 2008], but decidability of the latter problem remains open.

The index problems become easier when we restrict the input to languages recognized by deterministic automata. This is mostly due to the fact that in a deterministic automaton, each sub-automaton can be substituted with any automaton recognizing a language of the same index, without influencing the index of the whole language. This observation has been essential in providing a full characterization of the combinatorial structure of a language L in terms of certain patterns in a deterministic automaton recognizing L. This so-called pattern method [Murlak 2008a] has been successfully used for solving all four index problems for languages recognized by deterministic automata:

THEOREM 1.1. *For languages recognized by deterministic automata the following problems are decidable:*

(*1*) *the* deterministic *index problem [Niwiński and Walukiewicz 1998];*
(*2*) *the* non-deterministic *index problem [Urbański 2000; Niwiński and Walukiewicz 2005];*
(*3*) *the* alternating *index problem [Niwiński and Walukiewicz 2003]; and*
(*4*) *the* weak alternating *index problem [Murlak 2008b].*

The pattern method cannot be applied in general to non-deterministic or alternating automata; the reason is that both these types of automata naturally implement set-theoretic union of languages and union is not an operation that preserves the in-

dex of languages. But how far can we push the pattern method beyond deterministic automata?

In this paper we give a precise answer to this question. We present a syntactic class of automata for which substitution preserves the index of languages—we call them *game automata*—and show that it is the largest such class satisfying natural closure conditions. Relying on the first property we extend Theorem 1.1 (2), (3), (4) and prove the following.

THEOREM 1.2. *For languages recognized by game automata the following problems are decidable:*

(*1*) *the* non-deterministic *index problem,*
(*2*) *the* alternating *index problem,*
(*3*) *the* weak alternating *index problem.*

Decidability of the non-deterministic index problem for languages recognized by game automata is obtained via an easy reduction to the non-deterministic index problem for deterministic automata (Section 4).

As game automata recognize the game languages $W_{i,j}$ [Arnold 1999], the alternating index problem does not trivialize, unlike for deterministic automata, and is much more difficult than the non-deterministic index problem. We solve it by providing a recursive procedure computing the alternating index of the language recognized by a given game automaton (Section 5).

Similar techniques are applied to solve the weak alternating index problem (Section 6).

Finally, we give an effective characterization of languages recognized by game automata, within the class of all regular languages (Section 7). As the characterization effectively yields an equivalent game automaton, we obtain procedures computing the alternating, weak alternating, and non-deterministic index for a given alternating automaton equivalent to some game automaton.

This paper collects results from two conference papers: [Facchini et al. 2013] and [Facchini et al. 2015]. Additionally, it contains a discussion of the maximality of the class of game automata, which adapts a reasoning from [Duparc et al. 2011] to the index problem.

2. PRELIMINARIES

To simplify the presentation of inductive arguments, all our definitions allow partial objects: trees have leaves, automata have exits (where computation stops) and games have final positions (where the play stops and no player wins). The definitions become standard when restricted to *total* objects: trees without leaves, automata without exits, and games without final positions. We also do not distinguish the initial state of an automaton but treat it as an additional parameter for the recognized language.

2.1. Trees

For a function f we write $\mathrm{dom}(f)$ for the domain of f and $\mathrm{rg}(f)$ for the range of f. For a finite alphabet A, we denote by PTr_A the set of partial trees over A, i.e., functions $t\colon \mathrm{dom}(t) \to A$ from a prefix-closed subset $\mathrm{dom}(t) \subseteq \{\mathtt{L},\mathtt{R}\}^*$ to A. By Tr_A we denote the set of *total* trees, i.e., trees t such that $\mathrm{dom}(t) = \{\mathtt{L},\mathtt{R}\}^*$. For a direction $d \in \{\mathtt{L},\mathtt{R}\}$ by $\bar{d}$ we denote the opposite direction. For $v \in \mathrm{dom}(t)$, $t\restriction_v$ denotes the subtree of t rooted at v. The sequences $u, v \in \{\mathtt{L},\mathtt{R}\}^*$ are naturally ordered by the prefix relation: $u \preceq v$ if u is a prefix of v.

A tree that is not total contains *holes*. A *hole* of a tree t is a minimal sequence $h \in \{\mathtt{L},\mathtt{R}\}^*$ that does not belong to $\mathrm{dom}(t)$. By $\mathrm{holes}(t) \subseteq \{\mathtt{L},\mathtt{R}\}^*$ we denote the set of holes

of a tree t. If h is a hole of $t \in \mathrm{PTr}_A$, for $s \in \mathrm{PTr}_A$ we define the partial tree $t[h := s]$ obtained by putting the root of s into the hole h of t.

2.2. Games

A *parity game* $\mathbf{G}$ is a tuple $\langle V = V_\exists \cup V_\forall, v_I, F, E, \Omega \rangle$, where

— V is a countable *arena*;
— $V_\exists, V_\forall \subseteq V$ are positions of the game *belonging*, respectively, to player $\exists$ and player $\forall$, $V_\exists \cap V_\forall = \emptyset$;
— $v_I \in V$ is the initial position of the game;
— F is a countable set of *final positions*, $F \cap V = \emptyset$;
— $E \subseteq V \times (V \cup F)$ is the transition relation;
— $\Omega \colon V \to \{i, \ldots, j\} \subseteq \mathbb{N}$ is a *priority function*.

We assume that all parity games are finitely branching (for each $v \in V$ there are only finitely many $u \in V \cup F$ such that $(v, u) \in E$), and that there are no dead-ends (for each $v \in V$ there is at least one $u \in V \cup F$ such that $(v, u) \in E$).

A *play* in a parity game $\mathbf{G}$ is a finite or infinite sequence π of positions starting from v_I. If π is finite then $\pi = v_I v_1 \ldots v_n$ and v_n is required to be a final position (that is $v_n \in F$). In that case v_n is called the *final position of* π. An infinite play π is *winning* for $\exists$ if $\liminf_{n\to\infty} \Omega(\pi(n))$ is even. Otherwise π is winning for $\forall$.

A (positional) *strategy* σ for a player $P \in \{\exists, \forall\}$ in a game $\mathbf{G}$ is defined as usual, as a function assigning to every P's position $v \in V_P$ the chosen successor $\sigma(v) \in V \cup F$ such that $(v, \sigma(v)) \in E$. Strategies can be also seen as trees labelled with positions and final positions: we label the root with the initial position v_I, and then for each node labelled with a (non-final) position of the player P we add one child, corresponding to the move determined by the strategy, and for each node labelled with a (non-final) position of the opponent we add a child for each possible move. A play π *conforms to* σ if whenever π visits a vertex $v \in V_P$, the next position of π is $\sigma(v)$; that is, if π is a prefix of a branch of the strategy σ viewed as a tree. We say that a strategy σ is *winning* for P if every infinite play conforming to σ is winning for P. For a winning strategy σ we define the *guarantee of* σ as the set of all final positions that can be reached in plays conforming to σ (the labels of the leaves of σ viewed as a tree). Due to final positions, both players can have a winning strategy; in such case the intersection of their guarantees is nonempty, as two winning strategies used against each other must lead the play to a final position. Like for parity games without final positions, at least one player has a (positional) winning strategy [Emerson and Jutla 1991; Mostowski 1991a].

2.3. Automata

For the purpose of the inductive argument we incorporate into the definition of automata a finite set of *exits*. Therefore, an alternating automaton $\mathcal{A}$ is defined as a tuple $\langle A, Q, F, \delta, \Omega \rangle$, where A is a finite alphabet, Q is a finite set of states, F is a finite set of exits disjoint from Q, $\Omega \colon Q \to \mathbb{N}$ is a function assigning to each state of $\mathcal{A}$ its priority, and δ assigns to each pair $(q, a) \in Q \times A$ the transition $b = \delta(q, a)$ built using the grammar

$$b \ ::= \ \top \ \mid \ \bot \ \mid \ (q, d) \ \mid \ (f, d) \ \mid \ b \vee b \ \mid \ b \wedge b$$

for states $q \in Q$, $f \in F$, and directions $d \in \{\mathtt{L}, \mathtt{R}\}$.

For an alternating automaton $\mathcal{A}$, a state $q_I \in Q$, and a partial tree $t \in \mathrm{PTr}_A$ we define the game $\mathbf{G}(\mathcal{A}, t, q_I)$ as follows:

— $V = \mathrm{dom}(t) \times (S_\delta \cup Q)$, where S_δ is the set of all subformulae of formulae in $\mathrm{rg}(\delta)$; all positions of the form $(v, b_1 \vee b_2)$ belong to $\exists$ and the remaining ones to $\forall$;[2]
— $F = (\mathrm{holes}(t) \times (Q \cup F)) \cup \mathrm{dom}(t) \times F$;
— $v_I = (\epsilon, q_I)$;
— E contains the following pairs (for all $v \in \mathrm{dom}(t)$):
  — $\big((v, b), (v, b)\big)$ for $b \in \{\top, \bot\}$,
  — $\big((v, b), (v, b_i)\big)$ for $b = b_1 \wedge b_2$ or $b = b_1 \vee b_2$,
  — $\big((v, (q, d)), (vd, q)\big)$ for $d \in \{\mathtt{L}, \mathtt{R}\}$, $q \in Q \cup F$,
  — $\big((v, q), (v, \delta(q, t(v)))\big)$ for $q \in Q$;
— $\Omega(v, \top) = 0$, $\Omega(v, \bot) = 1$, $\Omega(v, q) = \Omega_{\mathcal{A}}(q)$ for $q \in Q$, $v \in \mathrm{dom}(t)$, and for other positions Ω is $\max(\mathrm{rg}(\Omega_{\mathcal{A}}))$, where $\Omega_{\mathcal{A}}$ is the priority function of $\mathcal{A}$.

An automaton $\mathcal{A}$ is *total* if $F = \emptyset$. A total automaton $\mathcal{A}$ *accepts* a total tree $t \in \mathrm{Tr}_A$ from $q_I \in Q$ if $\exists$ has a winning strategy in $\mathbf{G}(\mathcal{A}, t, q_I)$. By $L(\mathcal{A}, q_I)$ we denote the set of total trees accepted by a total automaton $\mathcal{A}$ from a state q_I. A total automaton $\mathcal{A}$ *recognizes* a language $L \subseteq \mathrm{Tr}_A$ if $L(\mathcal{A}, q_I) = L$ for some $q_I \in Q$. A state $q \in Q$ is *non-trivial* if $\emptyset \subsetneq L(\mathcal{A}, q) \subsetneq \mathrm{Tr}_A$. Without loss of generality, when a total automaton $\mathcal{A}$ recognizes a non-trivial language, i.e. $L(\mathcal{A}, q_I) \notin \{\emptyset, \mathrm{Tr}_A\}$ for some $q_I \in Q$, we implicitly assume that $\mathcal{A}$ has only non-trivial states.

The *(Rabin-Mostowski) index* of an automaton $\mathcal{A}$ is the pair (i, j) where i is the minimal and j is the maximal priority of the states of $\mathcal{A}$ ($\bot$ and $\top$ are counted as additional looping states with odd and even priority, respectively). In that case $\mathcal{A}$ is called an (i, j)-automaton.

An automaton $\mathcal{A}$ is *deterministic* if all its transitions are deterministic, i.e., of the form $\top$, $\bot$, (q_d, d), or $(q_{\mathtt{L}}, \mathtt{L}) \wedge (q_{\mathtt{R}}, \mathtt{R})$, for $d \in \{\mathtt{L}, \mathtt{R}\}$. Similarly, $\mathcal{A}$ is *non-deterministic* if its transitions are (multifold) disjunctions of deterministic transitions.

An automaton $\mathcal{A}$ is *weak* if whenever $\delta(q, a)$ contains a state q' then $\Omega(q) \leq \Omega(q')$. For weak automata, allowing *trivial* transitions $\top$ or $\bot$, interferes with the index much more than for strong automata: essentially, it adds one more change of priority. To reflect this, when defining the index of the automaton, we count $\bot$ and $\top$ as additional looping states with priorities assigned so that the weakness condition above is satisfied: $\bot$ gets the lowest *odd* priority ℓ such that $\bot$ is accessible only from states of priority at most ℓ, and dually for $\top$. That is, if the automaton uses priorities $i, i+1, \ldots, 2k-1$, we can use $\bot$ for free (with priority $2k-1$), but for $\top$ we may need to pay with an additional priority $2k$, yielding index $(i, 2k)$. To emphasize the fact that an automaton in question is weak, we often call its index the *weak index*.

2.4. Compositionality

Let $\mathcal{A} = \langle A, Q, F, \delta, \Omega \rangle$ be an alternating automaton and $Q' \subseteq Q$ be a set of states. By $\mathcal{A} \restriction_{Q'}$ we denote the *restriction of $\mathcal{A}$ to Q'* obtained by replacing the set of states by Q', the set of exits by $F \cup (Q - Q')$, the priority function by $\Omega \restriction_{Q'}$, and the transition function by $\delta \restriction_{Q' \times A}$. Let us stress that in the restricted automaton exits are either original exits or original states not in Q' (see Fig. 1). We say that $\mathcal{B}$ is a *sub-automaton of $\mathcal{A}$* (denoted $\mathcal{B} \subseteq \mathcal{A}$) if $\mathcal{B} = \mathcal{A} \restriction_{Q^{\mathcal{B}}}$.

For automata $\mathcal{A}$, $\mathcal{B}$ over an alphabet A with $Q^{\mathcal{A}} \cap Q^{\mathcal{B}} = \emptyset$, we define the composition $\mathcal{A} \cdot \mathcal{B}$ as the automaton over A, with states $Q = Q^{\mathcal{A}} \cup Q^{\mathcal{B}}$, exits $\big(F^{\mathcal{A}} \cup F^{\mathcal{B}}\big) - Q$, transitions $\delta^{\mathcal{A}} \cup \delta^{\mathcal{B}}$, and priorities $\Omega^{\mathcal{A}} \cup \Omega^{\mathcal{B}}$. Note here that some exits of $\mathcal{A}$ may be states of $\mathcal{B}$ and *vice versa*.

[2] Positions $(v, (q, d)), (v, q), (v, \bot), (v, \top)$ offer no choice, so their owner is irrelevant.

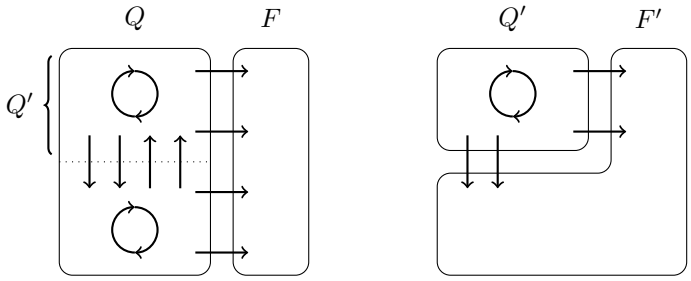

Fig. 1. An alternating automaton $\mathcal{A}$ with states Q and exits F; and the restriction $\mathcal{A}\restriction_{Q'}$ for $Q' \subseteq Q$. The edges illustrate the transitions of $\mathcal{A}$.

FACT 1. *If $\mathcal{A}$ is an alternating automaton and $Q = Q_1 \cup Q_2$ is a partition of the states of $\mathcal{A}$ then $\mathcal{A}\restriction_{Q_1} \cdot \mathcal{A}\restriction_{Q_2} = \mathcal{A}$.*

3. GAME AUTOMATA

Let $\mathcal{A}$ and $\mathcal{B}$ be automata over the same alphabet. For an occurrence of a state (or an exit) q in a transition $\delta(p,a)$ of $\mathcal{A}$, and a state $q_0^{\mathcal{B}}$ of $\mathcal{B}$, the substitution $\mathcal{A}_{\mathcal{B}}$ is obtained by taking the disjoint union of $\mathcal{A}$ and $\mathcal{B}$ (the state space is the disjoint union of $Q^{\mathcal{A}}$ and $Q^{\mathcal{B}}$, etc.), and replacing the occurrence of q in $\delta(p,a)$ with $q_0^{\mathcal{B}}$. The mapping $\mathcal{B} \mapsto \mathcal{A}_{\mathcal{B}}$ induces an operation on recognized languages, but it need not preserve coarser equivalence relations, like having the same index. For a class of automata $\mathfrak{C}$ over a common alphabet, we say that *substitution preserves (alternating, nondeterministic, etc.) index in* $\mathfrak{C}$ if for all $\mathcal{A}, \mathcal{B}_1, \mathcal{B}_2, \in \mathfrak{C}$, if $L(\mathcal{B}_1, q_0^{\mathcal{B}_1})$ and $L(\mathcal{B}_2, q_0^{\mathcal{B}_2})$ have the same (alternating, nondeterministic, etc.) index, then so do $L(\mathcal{A}_{\mathcal{B}_1}, q_0^{\mathcal{A}})$ and $L(\mathcal{A}_{\mathcal{B}_2}, q_0^{\mathcal{A}})$ for any $q_0^{\mathcal{A}}$.

As pointed out in the introduction, the operation of union does not preserve index. The same is true for intersection.

Example 3.1. Take $A = \{0,1,2\}$ and consider ω-word languages $(A^*(1+2))^\omega$ and $(A^*2)^\omega$. Both these languages can be recognized by deterministic automata of index $(1,2)$, and not lower than this. Taking union with A^*0^ω, we obtain $(A^*(1+2))^\omega \cup A^*0^\omega = A^\omega$, and $(A^*2)^\omega \cup A^*0^\omega$. To recognize the language $(A^*2)^\omega \cup A^*0^\omega$, a deterministic automaton requires three priorities and an alternating one needs two. This makes it much more complex than the whole space A^ω, which can be recognized by a deterministic automaton with a single state, whose priority is 0. Similarly, intersecting $(A^*(1+2))^\omega$ and $(A^*2)^\omega$ with $(A^*(0+1))^\omega$ we obtain respectively $A^*(0^*1)^\omega$, and the empty set, which have very different complexity. This example can be transferred to trees by encoding ω-words as sequences of labels on the left-most branches.

Example 3.1 illustrates a more general phenomenon. The following notion is designed to capture how an automaton can simulate union or intersection: we call a transition $\delta(q,a)$ *ambiguous* if it contains two occurrences of some direction $d \in \{\mathtt{L}, \mathtt{R}\}$. Recall that a transition is trivial if it is of the form $\bot$ or $\top$; as discussed in Section 2.3, trivial transitions are just a convenient notation for immediate acceptance and rejection, and can be easily replaced with looping states of appropriate priority.

FACT 2. *Let $\mathfrak{C}$ be a class of (alternating) automata without trivial transitions, over a fixed alphabet A containing at least two letters, that*

— is closed under substitution, and
— contains automata recognizing $\emptyset$, Tr_A, and some language X of non-trivial index.

It is only possible that substitution preserves index in $\mathfrak{C}$, if no automaton in $\mathfrak{C}$ contains an ambiguous transition.

PROOF. Let us take an arbitrary automaton $\mathcal{A} \in \mathfrak{C}$ and a state p of $\mathcal{A}$. Assume that $\delta(p,a)$ contains $(\mathtt{L}, p_a)$ and $\delta(p,b)$ contains $(\mathtt{L}, p_b)$ for some distinct letters $a, b \in A$ and some states p_a, p_b (the remaining three cases are symmetric). Starting from the automaton $\mathcal{A}$ and the automata recognizing $\emptyset$, Tr_A, and X, we obtain by substitution automata $\mathcal{A}_a, \mathcal{A}_b, \mathcal{A}'_a, \mathcal{A}'_b \in \mathfrak{C}$ such that

$$L(\mathcal{A}_a, p) = a(X, \mathrm{Tr}_A) \cup b(\mathrm{Tr}_A, \mathrm{Tr}_A), \qquad L(\mathcal{A}'_a, p) = a(X, \mathrm{Tr}_A),$$
$$L(\mathcal{A}_b, p) = a(\mathrm{Tr}_A, \mathrm{Tr}_A) \cup b(X, \mathrm{Tr}_A), \qquad L(\mathcal{A}'_b, p) = b(X, \mathrm{Tr}_A),$$

where for $c \in A$ and $Y, Z \subseteq \mathrm{Tr}_A$,

$$c(Y,Z) = \left\{ t \in \mathrm{Tr}_A \;\middle|\; t(\epsilon) = c,\ t{\restriction_{\mathtt{L}}} \in Y, t{\restriction_{\mathtt{R}}} \in Z \right\}.$$

Note that $L(\mathcal{A}_a, p) \cup L(\mathcal{A}_b, p) = \mathrm{Tr}_A$ and $L(\mathcal{A}'_a, p) \cap L(\mathcal{A}'_b, p) = \emptyset$.

Let $\mathcal{B} \in \mathfrak{C}$ and let q_0 be a state of $\mathcal{B}$ such that for some c, $\delta(q_0, c)$ is an ambiguous transition. By substituting appropriately the automata recognizing $\emptyset$ and Tr_A we can assume that $\delta(q_0, c) = (d, q_1) \vee (d, q_2)$ or $\delta(q_0, c) = (d, q_1) \wedge (d, q_2)$ for some states q_1, q_2, and no tree with a label $c' \neq c$ in the root is accepted from q_0. Assume $\delta(q_0, c) = (d, q_1) \vee (d, q_2)$, and let $\mathcal{B}'$ be the result of replacing the occurrence of q_1 with the state p of $\mathcal{A}_a$. Now, $L(\mathcal{A}_a, p)$ and $L(\mathcal{A}_b, p)$ have the same index, but by substituting in $\mathcal{B}'$ at (d, q_2) the automaton $\mathcal{A}_a$ or $\mathcal{A}_b$ (with initial state p), we get two languages of different index. For $\delta(q_0, c) = (d, q_1) \wedge (d, q_2)$ the argument is analogous but uses the other two automata. □

In the light of this result we propose the following definition.

Definition 3.2. A *game automaton* is an alternating automaton without ambiguous transitions; that is, it has only transitions of the following forms:

$$\top,\ \ \bot,\ \ (q_{\mathtt{L}}, \mathtt{L}),\ \ (q_{\mathtt{R}}, \mathtt{R}),\ \ (q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R}),\ \ (q_{\mathtt{L}}, \mathtt{L}) \wedge (q_{\mathtt{R}}, \mathtt{R})$$

for $q_{\mathtt{L}}, q_{\mathtt{R}} \in Q \cup F$.

In the course of the paper we shall see that for game automata substitution preserves the non-deterministic index (Proposition 4.2), the alternating index (Proposition 5.18), and the weak alternating index (Proposition 6.9). Together with Fact 2 this will imply that game automata are the largest non-trivial subclass of alternating automata closed under substitution for which substitution preserves the index.

The class of languages recognized by game automata is closed under complementation: the usual complementation procedure of increasing the priorities by one and swapping existential and universal transitions works. However they are neither closed under union nor intersection. For instance, let $L_c = \{t \in T_{\{a,b\}} \colon t(\mathtt{L}) = t(\mathtt{R}) = c\}$ for $c = a, b$. Obviously, L_a and L_b are recognizable by game automata, but $L_a \cup L_b$ is not. Note that the last example also shows that game automata do not recognize all regular languages. On the other hand they extend across the whole alternating index hierarchy, as they recognize so-called game languages $W_{i,j}$. We discuss this in more detail in Section 5.3.

The main similarity between game automata and deterministic automata is that their acceptance can be expressed in terms of *runs*, which are relabelings of input trees induced uniquely by transitions. For a total game automaton $\mathcal{A}$ and an initial state q_I, with each partial tree t one can associate the *run*

$$\rho(\mathcal{A}, t, q_I) \colon \mathrm{dom}(t) \cup \mathrm{holes}(t) \to Q^{\mathcal{A}} \cup \{\top, \bot, *\}$$

such that $\rho(\varepsilon) = q_I$ and for all $v \in \mathrm{dom}(t)$, if $\rho(v) = q$, $\delta(q, t(v)) = b_v$, then

— if b_v is $(q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R})$ or $(q_{\mathtt{L}}, \mathtt{L}) \wedge (q_{\mathtt{R}}, \mathtt{R})$, then $\rho(v\mathtt{L}) = q_{\mathtt{L}}$ and $\rho(v\mathtt{R}) = q_{\mathtt{R}}$;
— if $b_v = (q_d, d)$ for some $d \in \{\mathtt{L}, \mathtt{R}\}$, then $\rho(vd) = q_d$ and $\rho(v\bar{d}) = *$;
— if $b_v = \bot$ then $\rho(v\mathtt{L}) = \rho(v\mathtt{R}) = \bot$, and dually for $\top$;

and if $\rho(v) \in \{\top, \bot, *\}$, then $\rho(v\mathtt{L}) = \rho(v\mathtt{R}) = *$. Observe that $\rho(v)$ is uniquely determined by the labels of t on the path leading to v.

The run $\rho = \rho(\mathcal{A}, t, q_I)$ is naturally interpreted as a game $\mathbf{G}_\rho(\mathcal{A}, t, q_I)$ with positions $\mathrm{dom}(t) - \rho^{-1}(*)$, final positions $\mathrm{holes}(t)$, where edges follow the child relation and loop on $\rho^{-1}(\{\top, \bot\})$, priority of v is $\Omega^{\mathcal{A}}(\rho(v))$ with $\Omega^{\mathcal{A}}(\bot) = 1$, $\Omega^{\mathcal{A}}(\top) = 0$, and the owner of v is $\exists$ iff $\delta(\rho(v), t(v)) = (q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R})$ for some $q_{\mathtt{L}}, q_{\mathtt{R}} \in Q^{\mathcal{A}}$. Clearly $\mathbf{G}_\rho(\mathcal{A}, t, q_I)$ is equivalent to $\mathbf{G}(\mathcal{A}, t, q_I)$. Note that the arena of $\mathbf{G}_\rho(\mathcal{A}, t, q_I)$ is a subtree of t (with additional edges looping over positions labelled with $\top$ or $\bot$ in the run of $\mathcal{A}$ on t). Consequently, a strategy in $\mathbf{G}_\rho(\mathcal{A}, t, q_I)$ can be also viewed as a subtree of t (we can ignore the looping edges, as there is no choice there anyway. If t is total, we say that ρ is *accepting*, if $\exists$ has a winning strategy in $\mathbf{G}_\rho(\mathcal{A}, t, q_I)$.

A direct consequence of the acceptance being definable in terms of runs is that each part of the automaton matters: for each state q reachable from the given initial state, one can find a family of trees that spans all possible behaviours of the automaton from the state q, and the outcome of the computation depends exclusively on this behaviour. As we show below, it can be done by taking all full trees extending an appropriately chosen partial tree with a single hole, corresponding to the state q. Let $t \in \mathrm{PTr}_A$ be a partial tree and $\rho = \rho(\mathcal{A}, t, q_I)$ be the run of an automaton $\mathcal{A}$ on t. We say that t *resolves* $\mathcal{A}$ *from* $q_I \in Q^{\mathcal{A}}$ if $\rho(h) \neq *$ for each hole h of t and whenever $t \restriction_{vd}$ is the only total tree in $\{t \restriction_{v\mathtt{L}}, t \restriction_{v\mathtt{R}}\}$, either $\rho(vd) = *$ or vd is losing for the owner of v in $\mathbf{G}_\rho(\mathcal{A}, t, q_I)$. The following establishes the property we discussed above and its analogue for transitions.

FACT 3. *Assume that t resolves $\mathcal{A}$ from q_I and $\rho = \rho(\mathcal{A}, t, q_I)$. If t has a single hole h then $t[h := s] \in L(\mathcal{A}, q_I)$ iff $s \in L(\mathcal{A}, \rho(h))$ for all $s \in \mathrm{Tr}_A$.*

If t has two holes h, h' whose closest common ancestor u satisfies $\delta_{\mathcal{A}}(\rho(u), t(u)) = (q_{\mathtt{L}}, \mathtt{L}) \wedge (q_{\mathtt{R}}, \mathtt{R})$ for some $q_{\mathtt{L}}, q_{\mathtt{R}}$, then $t[h := s, h' := s'] \in L(\mathcal{A}, q_I)$ iff $s \in L(\mathcal{A}, \rho(h))$ and $s' \in L(\mathcal{A}, \rho(h'))$ for all s, s'; dually for $(q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R})$.

PROOF. Let us prove the first claim. There are two cases.

— One of the players $P \in \{\exists, \forall\}$ has a winning strategy σ in the game associated to ρ such that node h does not belong to σ viewed as a subtree of t. In that case, there exists an ancestor u of h such that the player P owns u and σ moves to from u to ud such that ud is not ancestor of h. In that case σ is a winning strategy for P in the subtree under ud, which contradicts the definition of a resolving tree.
— Whenever σ is a winning strategy for a player $P \in \{\exists, \forall\}$ in the game associated to ρ, then the node corresponding to h belongs to σ viewed as a subtree of t. Take any total tree s. If $t[h := s] \in L(\mathcal{A}, q_I)$ then $\exists$ has a winning strategy in the game associated with $\rho(\mathcal{A}, t[h := s], q_I)$. In particular, she can win from the position h in this game. Therefore, by the definition, $s \in L(\mathcal{A}, \rho(h))$. If $t[h := s] \notin L(\mathcal{A}, q_I)$ then the property is symmetrical: $\forall$ has a winning strategy and $s \notin L(\mathcal{A}, \rho(h))$.

For the second claim, it follows easily that in this case the trees $t \restriction_{u\mathtt{L}}$, $t \restriction_{u\mathtt{R}}$ and the tree obtained by putting a hole in t instead of u, resolve $\mathcal{A}$ from $q_{\mathtt{L}}$, $q_{\mathtt{R}}$, and q_I, respectively. We obtain the second claim by applying the first claim three times. □

4. NON-DETERMINISTIC INDEX PROBLEM

The decidability of the non-deterministic index problem for languages recognized by game automata is an immediate consequence of the decidability of the non-deterministic index problem for deterministic tree languages [Niwiński and Walukiewicz 2005] and the following observation.

PROPOSITION 4.1. *For each game automaton $\mathcal{A}$ and a state $q_I^{\mathcal{A}} \in Q^{\mathcal{A}}$ one can effectively construct a deterministic automaton $\mathcal{D}$ with an initial state $q_I^{\mathcal{D}}$, such that $L(\mathcal{A}, q_I^{\mathcal{A}})$ is recognized by a non-deterministic automaton of index (i,j) if and only if so is $L(\mathcal{D}, q_I^{\mathcal{D}})$.*

PROOF. Essentially, $\mathcal{D}$ recognizes the set of winning strategies for $\exists$ in games induced by the runs of $\mathcal{A}$. For two total trees $t \in \mathrm{Tr}_A$, $s \in \mathrm{Tr}_B$ let $t \otimes s \in \mathrm{Tr}_{A \times B}$ be given by $(t \otimes s)(v) = (t(v), s(v))$. Let $W^{\exists}_{\mathcal{A}, q_I}$ be the set of all total trees $t \otimes s$ over the alphabet $A^{\mathcal{A}} \times \{\mathtt{L}, \mathtt{R}, \star\}$ such that s encodes a winning strategy for $\exists$ in the game $\mathbf{G}_\rho(\mathcal{A}, t, q_I)$ in the following sense: if $s(v) \in \{\mathtt{L}, \mathtt{R}\}$, $\exists$ should choose $v \cdot s(v)$, and $s(v) = \star$ means that $\exists$ has no choice in v. It is easy to see that $W^{\exists}_{\mathcal{A}, q_I}$ can be recognized by a deterministic automaton $\mathcal{D}$. It inherits the state-space and the priority function from $\mathcal{A}$ and its transitions are modified as follows: for all $q \in Q$, $a \in A$, $d \in \{\mathtt{L}, \mathtt{R}\}$, if $\delta_{\mathcal{A}}(q, a) = (q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R})$ for some $q_{\mathtt{L}}, q_{\mathtt{R}}$, then

$$\delta(q, (a, d)) = (q_d, d)\,, \quad \delta(q, (a, \star)) = \bot\,,$$

otherwise,

$$\delta(q, (a, d)) = \bot\,, \quad \delta(q, (a, \star)) = \delta_{\mathcal{A}}(q, a)\,.$$

It is easy to check that $L(\mathcal{D}, q_I) = W^{\exists}_{\mathcal{A}, q_I}$.

Note that

$$L(\mathcal{A}, q_I^{\mathcal{A}}) = \left\{ t \in \mathrm{Tr}_{A^{\mathcal{A}}} \;\middle|\; \exists s.\ t \otimes s \in W^{\exists}_{\mathcal{A}, q_I^{\mathcal{A}}} \right\}.$$

Hence, if $W^{\exists}_{\mathcal{A}, q_I^{\mathcal{A}}} = L(\mathcal{B}, q_I^{\mathcal{B}})$ for some non-deterministic automaton $\mathcal{B}$ then $L(\mathcal{A}, q_I^{\mathcal{A}}) = L(\mathcal{B}', q_I^{\mathcal{B}})$, where $\mathcal{B}'$ is the standard projection of $\mathcal{B}$ on the alphabet $A^{\mathcal{A}}$: for all $q \in Q^{\mathcal{A}}$ and $a \in A^{\mathcal{A}}$, $\delta^{\mathcal{B}'}(q, a) = \delta^{\mathcal{B}}(q, (a, \mathtt{L})) \vee \delta^{\mathcal{B}}(q, (a, \mathtt{R})) \vee \delta^{\mathcal{B}}(q, (a, \star))$. The projection does not influence the index.

For the other direction, the proof is based on the following observation. For $t \in \mathrm{Tr}_{A^{\mathcal{A}}}$ and $s \in \mathrm{Tr}_{\{\mathtt{L}, \mathtt{R}, \star\}}$ let $t \odot s \in \mathrm{Tr}_{A^{\mathcal{A}}}$ be the tree obtained from t by the following operation: for each v, if $\rho_{t, q_I}(v) = q$, $\delta(q, t(v)) = (q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R})$, and $s(v) = \mathtt{L}$, then replace the subtree of t rooted at $v\mathtt{R}$ by some fixed regular tree in the complement of $L(\mathcal{A}, q_{\mathtt{R}})$; dually for $s(v) = \mathtt{R}$. (Recall that $\mathcal{A}$ has only non-trivial states, so $L(\mathcal{A}, q_{\mathtt{R}}) \subsetneq \mathrm{Tr}_{A^{\mathcal{A}}}$; by the Rabin's theorem the complement of $L(\mathcal{A}, q_{\mathtt{R}})$ contains a regular tree.) If s encodes a strategy σ_s for $\exists$ in $\mathbf{G}_\rho(\mathcal{A}, t, q_I^{\mathcal{A}})$, then σ_s is winning if and only if $t \odot s \in L(\mathcal{A}, q_I^{\mathcal{A}})$. Hence, $t \otimes s \in W^{\exists}_{\mathcal{A}, q_I^{\mathcal{A}}}$ if and only if s encodes a strategy for $\exists$ in $\mathbf{G}_\rho(\mathcal{A}, t, q_I^{\mathcal{A}})$ and $t \odot s \in L(\mathcal{A}, q_I^{\mathcal{A}})$. These conditions can be checked by a non-deterministic automaton of index (i,j) as soon as $L(\mathcal{A}, q_I^{\mathcal{A}})$ can be recognized by such an automaton.

Indeed, assume that $L(\mathcal{A}, q_I) = L(\mathcal{B}, q_I^{\mathcal{B}})$ for some non-deterministic automaton $\mathcal{B}$ of index (i,j). To construct a non-deterministic automaton $\mathcal{C}$ of index at most (i,j) recognizing $W^{\exists}_{\mathcal{A}, q_I}$, we first define a sequence of auxiliary languages and argue that

each of them can be recognized by such an automaton. Let

$$\begin{aligned}
\mathrm{St} &= \{t \otimes s : \ s \text{ is a strategy for } \exists \text{ in } \mathbf{G}_\rho(\mathcal{A}, t, q_I)\}\,,\\
\mathrm{StE} &= \{t \otimes s \otimes t' : \ t \otimes s \in \mathrm{St} \wedge t \odot s = t'\}\,,\\
\mathrm{StEW} &= \left\{t \otimes s \otimes t' \in \mathrm{StE} : \ t' \in L(\mathcal{B}, q_I^B) = L(\mathcal{A}, q_I)\right\},\\
\mathrm{StW} &= \{t \otimes s \in \mathrm{St} : \ t \odot s \in L(\mathcal{A}, q_I)\}\,.
\end{aligned}$$

Then,

— St corresponds to a safety condition that can be verified both by a deterministic automaton of index $(0,1)$ and by a deterministic automaton of index $(1,2)$,
— StE additionally enforces that the respective subtrees equal t_q, as above it can be checked both by a deterministic automaton of index $(0,1)$ and by a deterministic automaton of index $(1,2)$,
— StEW can be recognized by a product of automata recognizing StE and $\mathcal{B}$—the resulting non-deterministic automaton can be constructed in such a way that its index is (i,j),
— StW is obtained as the projection of StEW onto the first two coordinates, as such can also be recognized by a non-deterministic (i,j)-automaton.

It remains to show that

$$W^{\exists}_{\mathcal{A},q_I} = \mathrm{StW}$$

First assume that $t \otimes s \in W^{\exists}_{\mathcal{A},q_I}$. In that case s encodes a winning strategy σ for $\exists$ in $\mathbf{G}_\rho(\mathcal{A}, t, q_I)$. Let $t' = t \odot s$ and $D = \mathrm{dom}(\sigma)$ be the set of vertices belonging to σ. Note that if $v \in D$ then $t(v) = t'(v)$, so also $\rho_{t,q_I}(v) = \rho_{t',q_I}(v)$. Therefore, the strategy σ is also winning in $\mathbf{G}_\rho(\mathcal{A}, t', q_I)$. So $t' \in L(\mathcal{A}, q_I)$, which implies that $t \otimes s \otimes t' \in \mathrm{StEW}$ and $t \otimes s \in \mathrm{StW}$.

Now assume that $t \otimes s \in \mathrm{StW}$. Let $t' = t \odot s$ and σ be the strategy for $\exists$ in $\mathbf{G}_\rho(\mathcal{A}, t, q_I)$ encoded by s. By the definition of StEW we obtain that $t' \in L(\mathcal{A}, q_I)$ so there exists a winning strategy σ' for $\exists$ in $\mathbf{G}_\rho(\mathcal{A}, t', q_I)$. Similarly as above, let D (resp. D') be the set of vertices in σ (resp. σ'). If $D' \not\subseteq D$ then there exists a minimal (w.r.t. the prefix order) vertex $v \in D' - D$. By the definition of $t \odot s$ we obtain that $t' \restriction_v$ is t_q for $q = \rho(\mathcal{A}, t, q_I)(v)$. Therefore, since $t_q \notin L(\mathcal{A}, q)$, so there is no winning strategy for $\exists$ in $\mathbf{G}_\rho(\mathcal{A}, t_q, q)$ and we obtain a contradiction. Therefore $D' \subseteq D$ and for every $v \in D'$ we have $\rho(\mathcal{A}, t, q_I)(v) = \rho(\mathcal{A}, t', q_I)(v)$, so σ' is also a strategy in $\mathbf{G}_\rho(\mathcal{A}, t', q_I)$. Since strategies form an anti-chain with respect to inclusion, so $\sigma = \sigma'$, $t' \in L(\mathcal{A}, q_I)$, and $t \otimes s \in W^{\exists}_{\mathcal{A},q_I}$. □

As a direct corollary from the proof of Proposition 4.1 and the criteria for the non-deterministic index of deterministic languages [Niwiński and Walukiewicz 2005], we obtain the converse of Fact 2 for the non-deterministic index.

PROPOSITION 4.2. *For game automata, substitution preserves the non-deterministic index.*

PROOF. For a given game automaton $\mathcal{A}$, let $\mathcal{A}'$ be the deterministic automaton constructed in the proof of Proposition 4.1. Recall that $\mathcal{A}'$ is obtained from $\mathcal{A}$ by adding some transitions of the form $\bot$, and uncoupling each disjunctive transition over letter a into two non-branching transitions over letters $(a, \mathtt{L})$ and $(a, \mathtt{R})$; the state-space remains the same. It follows that the construction commutes with substitution: $(\mathcal{A}_\mathcal{B})'$ coincides with $(\mathcal{A}')_{\mathcal{B}'}$ for all game automata $\mathcal{A}$ and $\mathcal{B}$.

The construction was designed to preserve the non-deterministic index of the recognized language. Consequently, assuming that substitution preserves the index for

deterministic automata, we can show that the same holds for game automata. Indeed, if $L(\mathcal{B}, q_I^{\mathcal{B}})$ and $L(\mathcal{C}, q_I^{\mathcal{C}})$ have the same index, then $L(\mathcal{B}', q_I^{\mathcal{B}})$ and $L(\mathcal{C}', q_I^{\mathcal{C}})$ have the same index. If substitution preserves the index for deterministic automata, we can conclude that $L((\mathcal{A}_{\mathcal{B}})', q_I^{\mathcal{A}}) = L(\mathcal{A}'_{\mathcal{B}'}, q_I^{\mathcal{A}})$ and $L((\mathcal{A}_{\mathcal{C}})', q_I^{\mathcal{A}}) = L(\mathcal{A}'_{\mathcal{C}'}, q_I^{\mathcal{A}})$ have the same index. Hence, $L(\mathcal{A}_{\mathcal{B}}, q_I^{\mathcal{A}})$ and $L(\mathcal{A}_{\mathcal{C}}, q_I^{\mathcal{A}})$ have the same index.

Preservation of the index for deterministic automata follows immediately from the characterization of the levels of the index hierarchy among deterministic languages [Niwiński and Walukiewicz 2005]: it asserts that for every (i, j), a language $L(\mathcal{B}, q_I^{\mathcal{B}})$ is recognized by a non-deterministic automaton of index (i, j) if and only if the automaton $\mathcal{B}$ does not contain a certain characteristic, strongly connected subgraph reachable from $q_I^{\mathcal{B}}$. Consequently, if $L(\mathcal{B}, q_I^{\mathcal{B}})$ and $L(\mathcal{C}, q_I^{\mathcal{C}})$ have the same index, $\mathcal{B}$ and $\mathcal{C}$ contain the same characteristic subgraphs reachable from the respective initial states. By the definition of substitution, no strongly connected subgraph in $\mathcal{A}_{\mathcal{B}}$ can use states from $\mathcal{A}$ and from $\mathcal{B}$. Consequently, $\mathcal{A}_{\mathcal{B}}$ and $\mathcal{A}_{\mathcal{C}}$ contain the same characteristic subgraphs reachable from $q_I^{\mathcal{A}}$, and so $L(\mathcal{A}_{\mathcal{B}}, q_I^{\mathcal{A}})$ and $L(\mathcal{A}_{\mathcal{C}}, q_I^{\mathcal{A}})$ have the same index. □

5. ALTERNATING INDEX PROBLEM

In this section we show that the alternating index problem is decidable for game automata. Let us start with some notation.

Definition 5.1. For $i < j \in \mathbb{N}$, let $\mathrm{RM}(i, j)$ denote the class of languages recognized by alternating tree automata of index (i, j). Let

$$\begin{aligned}\mathbf{\Pi}_j^{RM} &= \mathrm{RM}(0, j),\\ \mathbf{\Sigma}_j^{RM} &= \mathrm{RM}(1, j+1),\\ \mathbf{\Delta}_j^{RM} &= \mathrm{RM}(0, j) \cap \mathrm{RM}(1, j+1).\end{aligned}$$

The above classes are naturally ordered by inclusion.

The result we prove not only gives decidability of the alternating index problem but also shows that languages recognizable by game automata collapse inside the $\mathbf{\Delta}_i^{RM}$ classes. To express it precisely we recall the so-called *comp* classes [Arnold and Santocanale 2005] that can be defined in terms of strongly connected components (SCCs) of a graph naturally associated with each alternating automaton.

Definition 5.2. Let $\mathcal{A}$ be an alternating automaton. Let $\mathrm{Gph}(\mathcal{A})$ be the directed edge-labelled graph over the set of vertices Q such that there is an edge $p \xrightarrow{(a,d)} q$ whenever (q, d) occurs in $\delta(p, a)$. Additionally, vertices of $\mathrm{Gph}(\mathcal{A})$ are labelled by values of Ω. We write $p \xrightarrow{w} q$ if there is a path in $\mathrm{Gph}(\mathcal{A})$ whose edge-labels yield the word w.

Definition 5.3. An alternating automaton $\mathcal{A}$ is in $\mathrm{Comp}(i, j)$ if (ignoring edge-labels) each SCC in $\mathrm{Gph}(\mathcal{A})$ has priorities between i and j or between $i + 1$ and $j + 1$.

It follows from the definition that each $\mathrm{Comp}(i, j)$ automaton is a $(i, j + 1)$ automaton, and can be transformed into an equivalent $\mathrm{Comp}(i+1, j+2)$ automaton by scaling the priorities. We write Comp_j for the class of languages recognized by $\mathrm{Comp}(0, j)$ automata. We then have

$$\mathbf{\Pi}_j^{RM} \cup \mathbf{\Sigma}_j^{RM} \subseteq \mathrm{Comp}_j \subseteq \mathbf{\Delta}_{j+1}^{RM}.$$

The class Comp_0 corresponds to the class of weak alternating automata. An important result obtained in [Rabin 1970] states that $\mathbf{\Delta}_1^{RM}$ coincides with the class of languages definable in weak monadic second order logic (WMSO). Since WMSO defin-

ability and weak recognizability are coextensive concepts [Muller et al. 1986], Rabin's result proves that classes Comp_0 and $\mathbf{\Delta}_1^{RM}$ coincide. However, as shown by Arnold and Santocanale [Arnold and Santocanale 2005], for higher levels the inclusion is strict

$$\mathrm{Comp}_j \subsetneq \mathbf{\Delta}_{j+1}^{RM} \quad \text{for } j > 0\,,$$

i.e., there are examples of regular languages in $\mathbf{\Delta}_{j+1}^{RM}$ but not in Comp_j. It turns out that, as a consequence of our characterization, in the case of languages recognizable by game automata the respective classes Comp_j and $\mathbf{\Delta}_{j+1}^{RM}$ coincide for all levels.

THEOREM 5.4. *For each game automaton $\mathcal{A}$ and an initial state q_I, the language $L(\mathcal{A}, q_I)$ belongs to exactly one of the classes:* Comp_0, $\mathbf{\Pi}_i^{RM} - \mathbf{\Sigma}_i^{RM}$, $\mathbf{\Sigma}_i^{RM} - \mathbf{\Pi}_i^{RM}$, *or* $\mathrm{Comp}_i - \big(\mathbf{\Pi}_i^{RM} \cup \mathbf{\Sigma}_i^{RM}\big)$, *for $i > 0$. Moreover, it can be effectively decided which class it is and an automaton from this class can be constructed.*

The rest of this section is devoted to showing this result. Section 5.1 describes a recursive procedure to compute the *class* of the given language $L(\mathcal{A}, q_I)$, i.e., $\mathbf{\Pi}_i^{RM}$, $\mathbf{\Sigma}_i^{RM}$, or Comp_i, depending on which of the possibilities holds. Sections 5.2, 5.3 show that the procedure is correct. The estimation of Section 5.2 is in fact an effective construction of an automaton from the respective class.

5.1. The algorithm

Let $\mathcal{A}$ be an alternating automaton of index (i, j). For $n \in \mathbb{N}$ we denote by $\mathcal{A}^{\geq n}$ the sub-automaton obtained from $\mathcal{A}$ by restricting to states of priority at least n. Observe that the index of $\mathcal{A}^{\geq n}$ is at most (n, j). A sub-automaton $\mathcal{B} \subseteq \mathcal{A}$ is an *n-component of* $\mathcal{A}$ if $\mathrm{Gph}(\mathcal{B})$ is a strongly connected component of $\mathrm{Gph}(\mathcal{A}^{\geq n})$. We say that $\mathcal{B}$ is *non-trivial* if $\mathrm{Gph}(\mathcal{B})$ contains at least one edge. Our algorithm computes the class of each n-component $\mathcal{B}$ of $\mathcal{A}$, based on the classes of $(n+1)$-components of $\mathcal{B}$ and transitions between them. (We shall see that for n-components the class does not depend on the initial state.)

We begin with a simple preprocessing. An automaton $\mathcal{A}$ is *priority-reduced* if for all $n > 0$, each n-component of $\mathcal{A}$ is non-trivial and contains a state of priority n.

LEMMA 5.5. *Each game automaton can be effectively transformed into an equivalent priority-reduced game automaton.*

PROOF. We iteratively decrease priorities in the n-components of $\mathcal{A}$, for $n \geq 1$. As long as there is an n-component that is not priority-reduced, pick any such n-component, if it is trivial, set all its priorities to $n-1$, if it is non-trivial but does not contain a state of priority n, decrease all its priorities by 2 (this does not influence the recognized language). After finitely many steps the automaton is priority-reduced. Note that no trivial states are introduced. □

The main algorithm uses three simple notions. An $(n+1)$-component $\mathcal{B}_0$ of $\mathcal{B}$ is *$\exists$-branching* if $\mathcal{B}$ contains a transition

$$\delta(p, a) = (q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R})$$

with $p, q_{\mathtt{L}} \in Q^{\mathcal{B}_0}$ or $p, q_{\mathtt{R}} \in Q^{\mathcal{B}_0}$. For $\forall$ replace $\vee$ with $\wedge$.

For a class K, operations $K^{\exists}$ and $K^{\forall}$ are defined as

$$\big(\mathbf{\Pi}_m^{RM}\big)^{\exists} = \big(\mathbf{\Sigma}_{m-1}^{RM}\big)^{\exists} = \big(\mathrm{Comp}_{m-1}\big)^{\exists} = \mathbf{\Pi}_m^{RM},$$
$$\big(\mathbf{\Sigma}_m^{RM}\big)^{\forall} = \big(\mathbf{\Pi}_{m-1}^{RM}\big)^{\forall} = \big(\mathrm{Comp}_{m-1}\big)^{\forall} = \mathbf{\Sigma}_m^{RM}.$$

We write $\bigvee_{\ell=1}^{k} K_\ell$ for the largest class among $K_1, K_2, \ldots, K_\ell$ if it exists, or Comp_m if among these classes there are two maximal ones, $\mathbf{\Pi}_m^{RM}$ and $\mathbf{\Sigma}_m^{RM}$.

Let $\mathcal{A}$ be a priority-reduced game automaton of index (i, j). The algorithm starts from $n = j$ and proceeds downward. Let $\mathcal{B}$ be an n-component.

— If $\mathcal{B}$ has only states of priority n, set $\mathrm{class}(\mathcal{B}) = \mathrm{Comp}_0$.
— If $\mathcal{B}$ has no states of priority n, it coincides with a single 1-component $\mathcal{B}_1$. Set $\mathrm{class}(\mathcal{B}) = \mathrm{class}(\mathcal{B}_1)$.
— Otherwise, assume that n is even (for odd n replace $\exists$ with $\forall$). Let $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$, be the $(n+1)$-components of $\mathcal{B}$ that are $\exists$-branching, and let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{k'}$ be the ones that are not $\exists$-branching. We set

$$\mathrm{class}(\mathcal{B}) = \bigvee_{\ell=1}^{k} \mathrm{class}(\mathcal{B}_\ell)^{\exists} \vee \bigvee_{\ell=1}^{k'} \mathrm{class}(\mathcal{C}_\ell)\,,$$

Let $\mathrm{class}(\mathcal{A}, q_I) = \bigvee_{\ell=1}^{k} \mathrm{class}(\mathcal{A}_\ell)$ where $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ are the i-components of $\mathcal{A}$ reachable from q_I in $\mathrm{Gph}(\mathcal{A})$.

5.2. Upper bounds

In this subsection we show that $L(\mathcal{A}, q_I)$ can be recognized by a $\mathrm{class}(\mathcal{A}, q_I)$-automaton. The argument will closely follow the recursive algorithm, pushing through an invariant guaranteeing that each n-component $\mathcal{B}$ of $\mathcal{A}$ can be replaced with an "equivalent" $\mathrm{class}(\mathcal{B})$-automaton. The notion of equivalence for non-total automata is formalized by simulations.

Definition 5.6. An alternating automaton $\mathcal{S}$ *simulates* a game automaton $\mathcal{A}$ if $F^{\mathcal{S}} \subseteq F^{\mathcal{A}}$ and there exists an embedding $\iota\colon Q^{\mathcal{A}} \to Q^{\mathcal{S}}$ (usually $Q^{\mathcal{A}} \subseteq Q^{\mathcal{S}}$) such that for all $t \in \mathrm{Tr}_A$, $q_I^{\mathcal{A}} \in Q^{\mathcal{A}}$, and for each winning strategy σ for player P in $\mathbf{G}(\mathcal{A}, t, q_I^{\mathcal{A}})$ there is a winning strategy σ^S for P in $\mathbf{G}(\mathcal{S}, t, \iota(q_I^{\mathcal{A}}))$ such that the guarantee of σ^S is contained in the guarantee of σ, and if there is an infinite play conforming to σ^S then there is an infinite play conforming to σ.

Note that if $\mathcal{A}$ and $\mathcal{S}$ are total and $\mathcal{S}$ simulates $\mathcal{A}$ then $L(\mathcal{A}, q_I^{\mathcal{A}}) = L(\mathcal{S}, \iota(q_I^{\mathcal{A}}))$.

LEMMA 5.7. *For each n-component $\mathcal{B}$ of a game automaton $\mathcal{A}$, $\mathcal{B}$ can be simulated by a $\mathrm{class}(\mathcal{B})$-automaton.*

PROOF. Assume that the index of $\mathcal{A}$ is (i, j). We proceed by induction on $n = j, j-1, \ldots, i$. If all states of $\mathcal{B}$ have priority n or all have priority strictly greater than n, the claim is immediate. Let us assume that neither is the case. By symmetry it is enough to give the construction for even n.

Suppose $\mathcal{B}$ has only $\exists$-branching $n+1$ components, $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$. Then $\mathrm{class}(\mathcal{B}) = \bigvee_\ell \mathrm{class}(\mathcal{B}_\ell)^{\exists} = \mathbf{\Pi}_m^{RM}$ for some $m \geq 1$. By the inductive hypothesis we get a $\mathrm{class}(\mathcal{B}_\ell)$-automaton $\mathcal{B}_\ell^S$, simulating $\mathcal{B}_\ell$. Since $\mathbf{\Pi}_m^{RM} \geq \mathrm{class}(\mathcal{B}_\ell)^{\exists}$, $\mathcal{B}_\ell^S$ can be assumed to be an $(n, n+m)$-automaton. Hence, we can put

$$\mathcal{B}^S = \mathcal{B}\restriction_{\Omega^{-1}(n)} \cdot \mathcal{B}_1^S \cdot \mathcal{B}_2^S \cdot \ldots \cdot \mathcal{B}_k^S$$

to get an $(n, n+m)$-automaton simulating $\mathcal{B}$.

Now, assume that $\mathcal{B}$ contains also $n+1$ components $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{k'}$ that are not $\exists$-branching. Repeating the construction above would now result in an automaton of index $\bigvee_\ell \mathrm{class}(\mathcal{B}_\ell)^{\exists} \vee \bigvee_\ell \mathrm{class}(\mathcal{C}_\ell)^{\exists}$, potentially higher than $\mathrm{class}(\mathcal{B}) = \bigvee_\ell \mathrm{class}(\mathcal{B}_\ell)^{\exists} \vee \bigvee_\ell \mathrm{class}(\mathcal{C}_\ell)$. Hence, instead of $\mathcal{C}_\ell^S$ we shall use $\mathcal{C}_\ell^R \cdot \mathcal{C}_\ell^T$, where

— $\mathcal{C}_\ell^T$ is a copy of $\mathcal{C}_\ell^S$ with each transition leading to an exit of $\mathcal{C}_\ell^S$ that is not an exit of $\mathcal{B}$, replaced with a transition to $\top$ (losing for $\forall$);

— $\mathcal{C}_\ell^R$ is $\mathcal{C}_\ell^S$ with all priorities set to n and additional ε-transitions (which can be eliminated in the usual way): for each state q of $\mathcal{C}_\ell^R$ allow $\forall$ to decide to stay in q or move to the copy of q in $\mathcal{C}_\ell^T$ (treated as an exit in $\mathcal{C}_\ell^R$).

Thus,

$$\mathcal{B}^S = \mathcal{B}\restriction_{\Omega^{-1}(n)} \cdot \mathcal{B}_1^S \cdot \ldots \cdot \mathcal{B}_k^S \cdot \mathcal{C}_1^R \cdot \mathcal{C}_1^T \cdot \ldots \cdot \mathcal{C}_{k'}^R \cdot \mathcal{C}_{k'}^T .$$

The composition of automata $\mathcal{B}\restriction_{\Omega^{-1}(n)}$, $\mathcal{B}_\ell^S$, $\mathcal{C}_\ell^R$ gives a $\mathrm{class}(\mathcal{B})$-automaton (each $\mathcal{C}_\ell^S$ was replaced with an (n,n)-automaton $\mathcal{C}_\ell^R$). This is further composed with $\mathrm{class}(\mathcal{C}_\ell)$-automata $\mathcal{C}_\ell^T$ in a loop-less way. Hence, $\mathcal{B}^S$ is a $\mathrm{class}(\mathcal{B})$-automaton.

Let us see that $\mathcal{B}^S$ simulates $\mathcal{B}$. Let ι be defined as identity on $\mathcal{B}\restriction_{\Omega^{-1}(n)}$, on $Q^{\mathcal{B}_\ell}$ as the embedding $Q^{\mathcal{B}_\ell} \to Q^{\mathcal{B}_\ell^S}$, and on $Q^{\mathcal{C}_\ell}$ as the embedding $Q^{\mathcal{C}_\ell} \to Q^{\mathcal{C}_\ell^R}$. Consider a tree $t \in \mathrm{Tr}_A$, a state $q_I^{\mathcal{B}}$ of $\mathcal{B}$, and games $\mathbf{G}(\mathcal{B}, t, q_I^{\mathcal{B}})$ and $\mathbf{G}(\mathcal{B}^S, t, \iota(q_I^{\mathcal{B}}))$.

First, consider a strategy σ for $\exists$ in $\mathbf{G}(\mathcal{B}, t, q_I^{\mathcal{B}})$. We decompose this strategy into parts corresponding to the sub-automata $\mathcal{B}_\ell$ and $\mathcal{C}_\ell$, for each part we use the fact that $\mathcal{B}_\ell^S$ simulates $\mathcal{B}_\ell$ and $\mathcal{C}_\ell^S$ simulates $\mathcal{C}_\ell$. This gives us a strategy for $\exists$ on parts of $\mathbf{G}(\mathcal{B}^S, t, \iota(q_I^{\mathcal{B}}))$ corresponding to sub-automata $\mathcal{B}_\ell^S$, $\mathcal{C}_\ell^R$, $\mathcal{C}_\ell^T$. Outside of $\mathcal{B}_\ell^S$, $\mathcal{C}_\ell^R$, and $\mathcal{C}_\ell^T$, $\exists$ has the same choices in $\mathcal{B}^S$ as in $\mathcal{B}$. Therefore, she can make her choices according to σ. This gives a complete strategy σ^S. Now consider any play conforming to σ^S. Such a play either visits infinitely many times a state of priority n in $\mathcal{B}^S$, and so is winning for $\exists$, or from some point on it stays in some sub-automaton $\mathcal{B}_\ell^S$, $\mathcal{C}_\ell^R$ or $\mathcal{C}_\ell^T$. In this case the play is also winning for $\exists$, by the assumption on σ and by the fact that all the changes of priorities in $\mathcal{C}_\ell^R$'s and transitions in $\mathcal{C}_\ell^T$'s are favourable to $\exists$. By the definition of σ^S, the guarantee of σ^S is contained in the guarantee of σ, and if there is an infinite play conforming to σ^S then there is an infinite play conforming to σ.

For a winning strategy σ for $\forall$ in $\mathbf{G}(\mathcal{B}, t, q_I^{\mathcal{B}})$, we construct a winning strategy σ^S for $\forall$ in $\mathbf{G}(\mathcal{B}^S, t, \iota(q_I^{\mathcal{B}}))$ as follows:

— in positions corresponding to states of priority n in $\mathcal{B}$ the strategy σ^S follows the decisions of σ;
— in components $\mathcal{B}_\ell^S$, $\mathcal{C}_\ell^R$, $\mathcal{C}_\ell^T$ the strategy σ^S simulates σ (using the fact that $\mathcal{C}_\ell^R$ and $\mathcal{C}_\ell^T$ have the same states and exits as the automaton $\mathcal{C}_\ell^S$ that simulates $\mathcal{C}_\ell$) with the following exception: $\forall$ immediately moves from $\mathcal{C}_\ell^R$ to $\mathcal{C}_\ell^T$ whenever each extension of the current play, conforming to the simulating strategy, stays forever in $\mathcal{C}_\ell^R$ (possibly reaching an exit that is also an exit of $\mathcal{B}^S$).

An easy inductive argument shows that

(1) each position (v,p) with $p \in \mathcal{B}\restriction_{\Omega^{-1}(n)}$ that is reached in some play conforming to σ^S is also reached in some play in $\mathbf{G}(\mathcal{B}, t, q_I^{\mathcal{B}})$ conforming to σ;
(2) whenever a play conforming to σ^S enters $\mathcal{B}_\ell^S$ (resp. $\mathcal{C}_\ell^R$) in a position (v,p), then $p = \iota(q)$ for some $q \in \mathcal{B}_\ell$ (resp. $q \in \mathcal{C}_\ell$) and (v,q) is reached in some play in $\mathbf{G}(\mathcal{B}, t, q_I^{\mathcal{B}})$ conforming to σ.

Consider any play b^S conforming to σ^S.

Assume that b^S is a finite play leading to a final position (v,f). Unless (v,f) is entered directly from some $\mathcal{C}_\ell^T$, by the two observations above (and by the definition of σ^S) it follows that (v,f) can also be reached in some play conforming to σ. Assume that (v,f) is entered directly from some $\mathcal{C}_\ell^T$. Let $(w,\iota(q))$ be the last moment when b^S entered $\mathcal{C}_\ell^R$ (recall that $\mathcal{C}_\ell^T$ is only entered from $\mathcal{C}_\ell^R$). Since σ^S in $\mathcal{C}_\ell^R$ and $\mathcal{C}_\ell^T$ mimics the simulating strategy in $\mathcal{C}_\ell^S$, the final position (v,f) can be reached in some play in

$\mathbf{G}(\mathcal{B}, t, q_I^{\mathcal{B}})$ starting in (w, q), conforming to σ. By observation 2 it follows that (v, f) is reached in a play conforming to σ and starting in $(\varepsilon, q_I^{\mathcal{B}})$.

The remaining case is when b^S is an infinite play. Should b^S visit infinitely often positions of priority n, by the observation 2 and by the definition of σ^S we would define a play in $\mathbf{G}(\mathcal{B}, t, q_I^{\mathcal{B}})$ conforming to σ that visits infinitely often positions of priority n. This is impossible since σ is winning for $\forall$. It follows that from some point on b^S stays in some sub-component. If the sub-component is $\mathcal{B}_\ell^S$, $\forall$ wins as he is playing with a winning strategy in $\mathcal{B}_\ell^S$. The other possibility is that b^S stays forever in $\mathcal{C}_\ell^R \cdot \mathcal{C}_\ell^T$ for some ℓ. Since $\mathcal{C}_\ell$ is not $\exists$-branching, in each transition of the form $\delta(p, a) = (q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R})$ with $p \in Q^{\mathcal{B}_\ell}$, at least one of the states $q_{\mathtt{L}}, q_{\mathtt{R}}$ is an exit state in $\mathcal{B}$, or both are outside of $\mathcal{C}_\ell$. Hence, after entering $\mathcal{C}_\ell$, σ becomes a single path in $\mathcal{C}_\ell$, with all the branchings (choices of $\exists$) going directly to exits of $\mathcal{B}$. In general, this path may end in a position belonging to $\exists$, such that both choices lead outside of $\mathcal{C}_\ell$ (not necessarily to exits of $\mathcal{B}$.) In our case the path must stay in $\mathcal{C}_\ell$ forever: since b^S is infinite and stays forever in $\mathcal{C}_\ell^R \cdot \mathcal{C}_\ell^T$, there is an infinite play conforming to the strategy simulating σ in $\mathcal{C}_\ell^S$ and, by Definition 5.6, an infinite play conforming to σ in $\mathcal{C}_\ell$. Consequently, all exits reachable with σ in $\mathcal{C}_\ell$ are also exits of $\mathcal{B}$. Hence, as soon as b^S enters $\mathcal{C}_\ell^R$ for the last time, σ^S tells $\forall$ to move to $\mathcal{C}_\ell^T$ where $\forall$ wins all infinite plays. □

It follows easily that $L(\mathcal{A}, q_I)$ can be recognized by a $\mathrm{class}(\mathcal{A}, q_I)$-automaton: the automaton can be obtained as a loop-less composition of the $\mathrm{class}(\mathcal{A}_\ell)$-automata simulating the i-components $\mathcal{A}_\ell$ of $\mathcal{A}$ reachable from q_I. In other words, the alternating index bounds as computed by the algorithm in Section 5.1 are correct.

5.3. Lower bounds

It remains to see that $L(\mathcal{A}, q_I)$ cannot be recognized by an alternating automaton of index lower than $\mathrm{class}(\mathcal{A}, q_I)$. Our proof uses the concept of topological hardness. A classical notion of topological hardness relies on the Borel hierarchy and the projective hierarchy [Kechris 1995], but these notions are not suitable for us, since most regular tree languages live on the same level of these hierarchies: $\mathbf{\Delta}_2^1$. We use a more refined notion based on continuous reductions [Wadge 1983] and so-called game languages [Arnold 1999; Bradfield 1998; Arnold and Niwiński 2007].

Definition 5.8. For $i < j$ consider the following alphabet

$$A_{i,j} = \{\exists, \forall\} \times \{i, i+1, \ldots, j\}.$$

With each $t \in \mathrm{PTr}_{A_{i,j}}$ we associate a parity game $\mathbf{G}_t$ where

— $V = \mathrm{dom}(t)$, $F = \mathrm{holes}(t)$,
— $E = \big\{(v, vd) \;\big|\; v \in \mathrm{dom}(t), d \in \{\mathtt{L}, \mathtt{R}\}\big\}$,
— if $t(v)=(P, n)$ then $\Omega(v)=n$ and $v \in V_P$ for $P \in \{\exists, \forall\}$.

Let $W_{i,j}$ be the set of *total trees* over $A_{i,j}$ such that $\exists$ has a winning strategy in $\mathbf{G}_t$.

Let us assume the usual Cantor-like topology on the space of trees, with the open sets defined as arbitrary unions of finite intersections of sets of the form $\{t \in \mathrm{Tr}_A \;|\; t(v) = a\}$ for $v \in \{\mathtt{L}, \mathtt{R}\}^*$ and $a \in A$. Topological hardness of languages can be compared using continuous reductions. A *continuous reduction* of $L_1 \subseteq X$ to $L_2 \subseteq Y$ is a continuous function $f\colon X \to Y$ such that $f^{-1}(L_2) = L_1$. The fact that L_1 can be continuously reduced to L_2 is denoted by $L_1 \leq_W L_2$. On Borel sets, the pre-order $\leq_W$ induces the so-called Wadge hierarchy (see [Wadge 1983]) which greatly refines the Borel hierarchy and has the familiar ladder shape with pairs of mutually dual classes alternating with

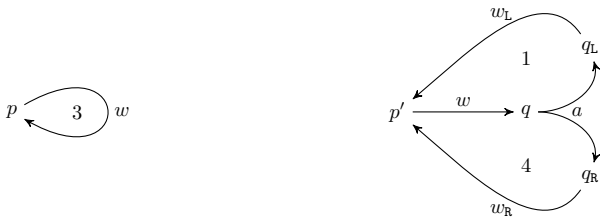

Fig. 2. A 3-loop rooted in p and a $(1,4)$-loop rooted in p'.

single self-dual classes. Here, we are interested in the following connection between continuous reductions, languages $W_{i,j}$, and the alternating index hierarchy.

FACT 4 ([ARNOLD 1999; BRADFIELD 1998; ARNOLD AND NIWIŃSKI 2007]). *For all $i < j$,*

(1) $W_{i,j}$ is regular and $W_{i,j} \in \mathrm{RM}(i,j)$,
(2) $L \leq_W W_{i,j}$ for each $L \in \mathrm{RM}(i,j)$,
(3) $W_{i,j} \not\leq_W W_{i+1,j+1}$,
(4) $W_{i,j} \in \mathbf{\Delta}^1_2$, $W_{0,1}$ is $\mathbf{\Sigma}^1_1$-complete, $W_{1,2}$ is $\mathbf{\Pi}^1_1$-complete.

This gives a criterion for proving index lower bounds.

COROLLARY 5.9. *If $W_{i,j} \leq_W L$ then $L \notin \mathrm{RM}(i+1, j+1)$.*

In consequence, in order to show that the index bound computed by the algorithm from Section 5.1 is tight, it suffices to show that if $\mathrm{RM}(i,j) \leq \mathrm{class}(\mathcal{A}, q_I)$, then $W_{i,j} \leq_W L(\mathcal{A}, q_I)$. We construct the reduction is three steps:

(1) we show that if the class computed by the algorithm (i.e. $\mathrm{class}(\mathcal{A}, q_I)$) is at least $\mathrm{RM}(i,j)$, then this is witnessed with a certain subgraph in $\mathrm{Gph}(\mathcal{A})$, called (i,j)-edelweiss;
(2) we introduce intermediate languages $\widehat{W}_{i,j}$, whose internal structure corresponds precisely to (i,j)-edelweisses, and in consequence $\widehat{W}_{i,j} \leq_W L(\mathcal{A}, q_I)$ if only $\mathcal{A}$ contains an (i,j)-edelweiss reachable from q_I;
(3) we prove that $W_{i,j} \leq_W \widehat{W}_{i,j}$.

The combinatorial core of the argument is the last step.

Definition 5.10. We say that in a game automaton $\mathcal{B}$ there is an *i-loop rooted in p* if there exists a word w such that on the path $p \xrightarrow{w} p$ in $\mathrm{Gph}(\mathcal{B})$ the minimal priority is i (see the left-hand side of Fig. 2).

A game automaton $\mathcal{B}$ contains an *(i,j)-loop for $\exists$ rooted in p* (see the right-hand side of Fig. 2), if there exist states $q, q_{\mathtt{L}}, q_{\mathtt{R}}$ of $\mathcal{B}$, a letter a, and words $w, w_{\mathtt{L}}, w_{\mathtt{R}}$ such that:

— $\delta(q,a) = (q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R})$;
— $p \xrightarrow{w} q$; $q_{\mathtt{L}} \xrightarrow{w_{\mathtt{L}}} p$; $q_{\mathtt{R}} \xrightarrow{w_{\mathtt{R}}} p$;
— on one of the paths $p \xrightarrow{w\,(a,\mathtt{L})\,w_{\mathtt{L}}} p$ or $p \xrightarrow{w\,(a,\mathtt{R})\,w_{\mathtt{R}}} p$ the minimal priority is i and on the other it is j.

For $\forall$ dually, with $\vee$ replaced with $\wedge$.

For an *even $j > i$*, $\mathcal{B}$ contains an *(i,j)-edelweiss rooted in p* (see Fig. 3 and Fig. 4) if for some even n it contains

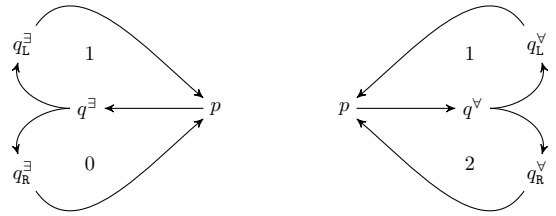

Fig. 3. $(0,1)$-edelweiss and $(1,2)$-edelweiss.

— $(n+k)$-loops for $k = i, i+1, \ldots, j-3$,
— $(n+j-2, n+j-1)$-loop for $\exists$, if $i \leq j-2$; and
— $(n+j-1, n+j)$-loop for $\forall$;

all rooted in p. For odd j swap $\forall$ and $\exists$ but keep n even.

LEMMA 5.11. *Let $\mathcal{A}$ be a game automaton and q_I a state of $\mathcal{A}$. If $\mathrm{class}(\mathcal{A}, q_I) \geq \mathrm{RM}(i,j)$ then $\mathcal{A}$ contains an (i,j)-edelweiss rooted in a state reachable from q_I.*

PROOF. Let us first assume that $(i,j) = (0,1)$. Analyzing the algorithm we see that the only case when $\mathrm{class}(\mathcal{A}, q_I)$ jumps to $\mathrm{RM}(0,1)$ is when for some even n there is an n-component $\mathcal{B}$ in $\mathcal{A}$, reachable from q_I, and containing states of priority n, such that some $n+1$ component $\mathcal{B}_\ell$ of $\mathcal{B}$ is $\exists$-branching in $\mathcal{B}$, i.e., $\mathcal{B}$ contains a transition of the form

$$\delta(p,a) = (q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R})$$

with $p, q_{\mathtt{L}} \in Q^{\mathcal{B}_\ell}$, $q_{\mathtt{R}} \in Q^{\mathcal{B}}$ (or symmetrically, $p, q_{\mathtt{R}} \in Q^{\mathcal{B}_\ell}$, $q_{\mathtt{L}} \in Q^{\mathcal{B}}$). Since $\mathcal{A}$ is priority-reduced, p is reachable from $q_{\mathtt{L}}$ within $\mathcal{B}_\ell$ via a state of priority $n+1$, and from $q_{\mathtt{R}}$ within $\mathcal{B}$ via a state of priority n. This gives an $(n, n+1)$-loop for $\exists$ (a $(0,1)$-edelweiss) rooted in a state reachable from q_I. The argument for $(1,2)$ is entirely dual.

Next, assume that $(i,j) = (0,2)$. It follows immediately from the algorithm that $\mathcal{A}$ contains an n-component $\mathcal{B}$ (reachable from q_I, containing states of priority n) such that n is even and there exists an $\exists$-branching $(n+1)$-component $\mathcal{B}_\ell$ in $\mathcal{B}$ such that $\mathrm{class}(\mathcal{B}_\ell) = \mathbf{\Sigma}_1^{RM}$ or $\mathrm{class}(\mathcal{B}_\ell) = \mathrm{Comp}_1$. In either case, $\mathrm{class}(\mathcal{B}_\ell) \geq \mathrm{RM}(1,2)$ and by the previous case $\mathcal{B}_\ell$ contains an $(n', n'+1)$-loop for $\forall$, for some odd $n' \geq n$. Since $\mathcal{A}$ is priority-reduced, for each state q in $\mathcal{B}_\ell$ and each r between n and $\Omega(q)$, there is a loop from q to q with the lowest priority r. Hence, the $(n', n'+1)$-loop can be turned into an $(n+1, n+2)$-loop. Thus, $\mathcal{B}_\ell$ contains an $(n+1, n+2)$-loop for $\forall$, rooted in a state p. We claim that $\mathcal{B}$ contains an $(n, n+1)$-loop for $\exists$, also rooted in p (giving a $(0,2)$-edelweiss rooted in p). Indeed, since $\mathcal{B}_\ell$ is $\exists$-branching, arguing like for $(0,1)$, we obtain an $(n, n+1)$-loop for $\exists$ rooted in a state p' in $\mathcal{B}_\ell$. Since $\mathcal{B}_\ell$ is an $n+1$-component, there are paths in $\mathcal{B}_\ell$ from p to p' and back; the lowest priority on these paths is at least $n+1$. Using these paths one easily transforms the $(n, n+1)$-loop rooted in p' into an $(n, n+1)$-loop rooted in p.

The inductive step is easy. Suppose that $j - i > 2$. Then, for some even n, $\mathcal{A}$ contains an $(n+i)$-component $\mathcal{B}$ (reachable from q_I, containing states of priority $n+i$), which has an $(n+i+1)$-component $\mathcal{B}_\ell$ such that $\mathrm{class}(\mathcal{B}_\ell) = \mathrm{RM}(i+1, j)$ or $\mathrm{class}(\mathcal{B}_\ell) = \mathrm{Comp}(i+1, j)$. Since for each state p in $\mathcal{B}_\ell$, $\mathcal{B}$ contains an $(n+i)$-loop rooted in p, we can conclude by the inductive hypothesis. □

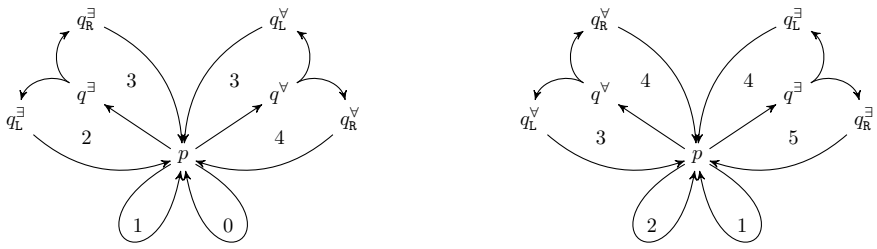

Fig. 4. $(0,4)$-edelweiss and $(1,5)$-edelweiss.

Definition 5.12. For $i \leq 2k-2$ consider the alphabet

$$\widehat{A}_{i,2k} = \{i, i+1, \ldots, 2k-3, e, a\}.$$

With each $t \in \mathrm{PTr}_{\widehat{A}_{i,2k}}$ we associate a parity game $\widehat{\mathbf{G}}_t$ with positions $\mathrm{dom}(t)$ and final positions $\mathrm{holes}(t)$ such that

— if $t(v) = a$, then in v player $\forall$ can choose to go to $v\mathtt{L}$ or to $v\mathtt{R}$, and $\Omega(v\mathtt{L}) = 2k-1$, $\Omega(v\mathtt{R}) = 2k$,
— if $t(v) = e$, then in v player $\exists$ can choose to go to $v\mathtt{L}$ or to $v\mathtt{R}$, and $\Omega(v\mathtt{L}) = 2k-2$, $\Omega(v\mathtt{R}) = 2k-1$,
— if $t(v) \in \{i, i+1, \ldots, 2k-3\}$, the only move from v is to $v\mathtt{L}$ and $\Omega(vL) = t(v)$.

For $i = 2k-1$, let $\widehat{A}_{i,2k} = \{a, \top\}$, and let $\widehat{\mathbf{G}}_t$ be defined like above, except that if $t(v) = \top$ then $\Omega(v) = 2k$ and the only move from v is back to v.

Let $\widehat{W}_{i,2k}$ be the set of all total trees over $A_{i,2k}$ such that $\exists$ has a winning strategy in $\widehat{\mathbf{G}}_t$.

The languages $\widehat{W}_{i,2k+1}$ are defined dually, with e, a and $\exists, \forall$ swapped, and $\top$ replaced with $\bot$.

LEMMA 5.13. *If a total game automaton $\mathcal{A}$ contains an (i,j)-edelweiss rooted in a state reachable from an initial state q_I then $\widehat{W}_{i,j} \leq_W L(\mathcal{A}, q_I)$.*

PROOF. We only give the proof for $(i,j) = (1,2)$; for other values of (i,j) the argument is entirely analogous. By the definition, $\mathcal{A}$ contains an $(1,2)$-loop for $\forall$, rooted in a state p reachable from q_I. Since $\mathcal{A}$ is a game automaton and has no trivial states, it follows that there exist

— a partial tree t_I resolving $\mathcal{A}$ from q_I, with a single hole v, labelled with p in $\rho(\mathcal{A}, t_I, q_I)$;
— a partial tree t_a resolving $\mathcal{A}$ from p with two holes v_1, v_2, such that in $\rho(\mathcal{A}, t_a, p)$ both holes are labelled p, the lowest priority on the path from the root to v_i is i, and the closest common ancestor u of v_1 and v_2 is labelled with a state q such that $\delta_{\mathcal{A}}(q, t(u)) = (q_{\mathtt{L}}, \mathtt{L}) \wedge (q_{\mathtt{R}}, \mathtt{R})$ for some $q_{\mathtt{L}}, q_{\mathtt{R}}$; and
— a total tree $t_\top \in L(\mathcal{A}, p)$.

Let us see how to build t_a. The paths $p \xrightarrow{w\,(a,\mathtt{L})\,w_{\mathtt{L}}} p$, $p \xrightarrow{w\,(a,\mathtt{R})\,w_{\mathtt{R}}} p$ guaranteed by Definition 5.10 give as a partial tree s with a single branching in some node u and two leaves v_1, v_2, which we replace with holes. For $\rho = \rho(\mathcal{A}, s, p)$, $\rho(v_1) = \rho(v_2) = p$ and $\delta(\rho(u), t(u)) = (q_{\mathtt{L}}, \mathtt{L}) \wedge (q_{\mathtt{R}}, \mathtt{R})$. At each hole of s, except v_1 and v_2, we substitute a total tree such that the run on the resulting tree with two holes resolves $\mathcal{A}$ from p, e.g., if vL

is a hole and $\delta(s(v), \rho(v)) = (q', \mathtt{L}) \vee (q'', \mathtt{R})$, we substitute at $v\mathtt{L}$ any tree that is not in $L(\mathcal{A}, q')$, relying on the assumption that $\mathcal{A}$ has no trivial states.

Let us define the reduction $g\colon \mathrm{Tr}_{\{a,\top\}} \to \mathrm{Tr}_{A^{\mathcal{A}}}$. Let $t \in \mathrm{Tr}_{\{a,\top\}}$. For $v \in \mathrm{dom}(t)$, define t_v co-inductively as follows: if $t(v) = \top$, set $t_v = t_\top$; if $t(v) = a$, then t_v is obtained by plugging in the holes v_1, v_2 of t_a the trees $t_{v\mathtt{L}}$ and $t_{v\mathtt{R}}$. Let $g(t)$ be obtained by plugging t_ε in the hole of t_I. It is easy to check that g continuously reduces $\widehat{W}_{1,2}$ to $L(\mathcal{A}, q_I)$. □

It remains to see that $W_{i,j} \leq_W \widehat{W}_{i,j}$. For the lowest level we give a separate proof.

LEMMA 5.14. *$W_{0,1} \leq_W \widehat{W}_{0,1}$ and $W_{1,2} \leq_W \widehat{W}_{1,2}$.*

PROOF. By the symmetry it is enough to prove the first claim. Let us take $t \in \mathrm{Tr}_{A_{0,1}}$. By König's lemma, Player $\exists$ has a winning strategy in $\mathbf{G}_t$ if and only if she can produce a sequence of finite strategies $\sigma_0, \sigma_1, \sigma_2, \ldots$ (viewed as subtrees of t) such that

(1) σ_0 consists of the root only;
(2) for each n the strategy σ_{n+1} extends σ_n in such a way that below each leaf of σ_n a non-empty subtree is added, and all the leaves of σ_{n+1} have priority 0.

Clearly, the union of such a sequence of finite strategies $(\sigma_n)_{n\in\mathbb{N}}$ is a total strategy for $\exists$ in $\mathbf{G}_t$. Additionally, the strategies σ_n witness that their union visits a node of priority 0 infinitely many times on each branch. Therefore, $\exists$ wins $\mathbf{G}_t$.

Consider the opposite direction: we assume that $\exists$ wins in $\mathbf{G}_t$ using a strategy σ and we want to define the strategies σ_n. Let σ_0 consist of the root only and let $\sigma_{n+1} \subseteq \sigma$ extend the strategy σ_n until the next node of priority 0 is seen on every branch. We need to prove that all the strategies σ_n are finite. Assume contrarily that σ_n is finite but σ_{n+1} is not. Let v be a leaf of σ_n such that $\sigma_{n+1}\restriction_v$ is infinite. By König's lemma we know that there exists an infinite branch π of σ_{n+1} such that $v \prec \pi$. In that case there is no node of priority 0 on π after v. Therefore, π treated as a play is winning for $\forall$ and is consistent with σ. It contradicts the assumption that σ was a winning strategy for $\exists$.

Using such approximating strategies σ_n we can define the required reduction. Let $(\tau_i)_{i\in\mathbb{N}}$ be the list of all finite unlabelled binary trees. Some of these trees naturally induce a strategy for $\exists$ in $\mathbf{G}_t$. For those we define $t_{\tau_i} \in \mathrm{Tr}_{\{e,\bot\}}$ co-inductively, as follows:

— $t_{\tau_i}(\mathtt{R}^j) = e$ for all j;
— if τ_j induces in $\mathbf{G}_t$ a strategy that is a legal extension of the strategy induced by τ_i in the sense of item 2) above, then the subtree of t_{τ_i} rooted at $\mathtt{R}^j\mathtt{L}$ is t_{τ_j};
— otherwise, all the nodes in this subtree are labelled with $\bot$.

Let $f(t) = t_{\sigma_0}$. By the initial observation, $t_{\sigma_0} \in \widehat{W}_{0,1}$ if and only if $\exists$ has a winning strategy in $\mathbf{G}_t$: a winning strategy for $\exists$ in t_{σ_0} corresponds to the successive choices of strategies $\sigma_0 \subseteq \sigma_1 \subseteq \ldots$.

Additionally, the function f is continuous: to determine the labels in nodes $\mathtt{R}^{n_1}\mathtt{L}\mathtt{R}^{n_2}\mathtt{L}\ldots\mathtt{R}^{n_k}$ and $\mathtt{R}^{n_1}\mathtt{L}\mathtt{R}^{n_2}\mathtt{L}\ldots\mathtt{R}^{n_k}\mathtt{L}$ we only need to know the restriction of t to the union of the domains of $\tau_{n_1}, \tau_{n_2}, \ldots, \tau_{n_k}$. Hence, f continuously reduces $W_{0,1}$ to $\widehat{W}_{0,1}$. □

Our aim now is to prove the following proposition, which forms the technical core of this section.

PROPOSITION 5.15. *For all i and $j \geq i+2$, $W_{i,j} \leq_W \widehat{W}_{i,j}$.*

The rest of this section is devoted to the proof of the proposition above. We begin by defining an auxiliary game $\tilde{\mathbf{G}}_t$ and proving that it is equivalent with $\mathbf{G}_t$. The structure of the game $\tilde{\mathbf{G}}_t$ corresponds to the possible choices of players in an edelweiss.

By duality we can assume that $j = 2k$. For $t \in \mathrm{Tr}_{A_{i,2k}}$, let us consider a game $\tilde{\mathbf{G}}_t$ defined as follows. The positions are pairs (v, σ), where v is a node of t, and σ is a finite strategy from v for $\forall$ (viewed as a subtree of $t \restriction_v$). Initially $v = \varepsilon$ is the root of t and $\sigma = \{\varepsilon\}$. In each round, in a position (v, σ), the players make the following moves:

— $\forall$ extends σ under leaves of priority $2k-1$ to σ' in such a way that on every path leading from a leaf of σ to a leaf of σ' all nodes have priority $2k$, except the leaf of σ', which has priority at most $2k-1$;
— $\exists$ has the following possibilities:
  — select a leaf v' of σ' with priority at most $2k-2$, and let the next round start with $(v', \{v'\})$, or
  — if σ' has some leaves of priority $2k-1$, continue with (v, σ').

A play is won by $\exists$ if she selects a leaf infinitely many times and the least priority of these leaves seen infinitely often is even, or $\forall$ is unable to extend σ in some round. Otherwise, the play is won by $\forall$.

LEMMA 5.16. *A player P has a winning strategy in $\mathbf{G}_t$ if and only if P has a winning strategy in $\tilde{\mathbf{G}}_t$.*

PROOF. For a winning strategy $\sigma_\exists$ for $\exists$ in $\mathbf{G}_t$, let $\tilde{\sigma}_\exists$ be the strategy in $\tilde{\mathbf{G}}_t$ in which $\exists$ selects a leaf v' in σ' if and only if $v' \in \sigma_\exists$. Consider an infinite play conforming to $\tilde{\sigma}_\exists$. If in the play $\exists$ selects a leaf infinitely many times, she implicitly defines a path in t conforming to $\sigma_\exists$, and so the play must be winning for $\exists$. Assume that $\exists$ selects a leaf only finitely many times. Then, $\forall$ produces an infinite sequence of finite strategies $\{v\} = \sigma_0 \subseteq \sigma_1 \subseteq \ldots$ in $\mathbf{G}_t$. Let σ_∞ be the union of these strategies. Consider the play π in $\mathbf{G}_t$ passing through v and conforming to σ_∞ and $\sigma_\exists$. Observe that for each σ_i, the strategy $\sigma_\exists$ must choose some path; hence, either $\exists$ selects a leaf of σ_i, or this path goes via a leaf of priority $2k-1$. Thus, π is infinite and by the rules of $\tilde{\mathbf{G}}_t$ priorities at most $2k-1$ are visited infinitely often. Since $\exists$ selects a leaf only finitely many times, priorities strictly smaller than $2k-1$ are visited finitely many times in π. Hence, π is won by $\forall$, what contradicts the assumption that $\sigma_\exists$ is winning for $\exists$.

Now, let $\sigma_\forall$ be a winning strategy for $\forall$ in $\mathbf{G}_t$. Then, for each $v \in \sigma_\forall$ there exists a finite sub-strategy σ' of $\sigma_\forall$ from v such that all internal nodes of σ' have priority $2k$ and leaves have priority at most $2k-1$. This shows that for each current strategy $\sigma \subseteq \sigma_\forall$, $\forall$ is able to produce a legal extension $\sigma' \subseteq \sigma_\forall$. Let $\tilde{\sigma}_\forall$ be a strategy of $\forall$ in $\tilde{\mathbf{G}}_t$ that extends every given σ by σ' as above. Consider any play conforming to $\tilde{\sigma}_\forall$. By the initial observation, the play is infinite, so priorities strictly smaller then $2k$ are visited infinitely often. If $\exists$ selects a leaf only finitely many times, priorities strictly smaller then $2k-1$ occur only finitely many times and $\forall$ wins. If $\exists$ selects a leaf infinitely many times, then the lowest priority seen infinitely often must be odd, as otherwise $\exists$ would show a losing path in $\sigma_\forall$. Hence, $\forall$ wins in this case as well. □

Now it remains to encode the game $\tilde{\mathbf{G}}_t$ as a tree $f(t) \in \mathrm{Tr}_{\widehat{A}_{i,2k}}$ in a continuous manner. The argument is similar to the one in Lemma 5.14. Let $(\tau_n)_{n\in\mathbb{N}}$ be the list of all unlabelled finite trees. For some pairs (v, τ_n), τ_n induces a strategy in $\mathbf{G}_t$ from the node v. For such (v, τ_n) we define $t^\forall_{v,\tau_n}$ and $t^\exists_{v,\tau_n}$ co-inductively, as follows:

— $t^\forall_{v,\tau_n}(\mathtt{R}^m) = a$ for all m;
— the subtree of $t^\forall_{v,\tau_n}$ rooted at $\mathtt{R}^m\mathtt{L}$ is $t^\exists_{v,\tau_m}$ if τ_m induces a strategy from v that is a legal extension of τ_n according to the rules of $\tilde{\mathbf{G}}_t$, and otherwise the whole subtree is labelled with e's (losing choice for $\forall$);

— $t^{\exists}_{v,\tau_n}(\mathtt{R}^m) = e$ for $m = 0, 1, \ldots, \ell$, where $v_0, v_1, \ldots, v_\ell$ are the leaves in the strategy induced by τ_n from v;
— the subtree of $t^{\exists}_{v,\tau_n}$ rooted at $\mathtt{R}^{\ell+1}$ is $t^{\forall}_{v,\tau_n}$ if the strategy induced by τ_n from v has some leaves of priority $2k-1$, otherwise the whole subtree is labelled with a's (losing choice for $\exists$);
— for $m \leq \ell$, consider the following cases to define the subtree s_m of $t^{\exists}_{v,\tau_n}$ rooted at $\mathtt{R}^m\mathtt{L}$:
  — if $\Omega(v_m) \in \{2k-1, 2k\}$ then s_m is labelled everywhere with a's (losing choice for $\exists$),
  — if $\Omega(v_m) = 2k-2$ then $s_m = t^{\forall}_{v_m,\{v_m\}}$,
  — if $\Omega(v_m) = r < 2k-2$ then $s_m(\varepsilon) = r$, the left subtree of s_m is $t^{\forall}_{v_m,\{v_m\}}$, and the right subtree of s_m is labelled with a's (irrelevant for $\mathbf{G}_t$).

Let $f(t)$ be $t^{\forall}_{\varepsilon,\{\varepsilon\}}$. Checking that f is continuous does not pose any difficulties. Lemma 5.16 implies that f reduces $W_{i,j}$ to $\widehat{W}_{i,j}$, which concludes the proof of Proposition 5.15.

5.4. Corollaries

As a by-product of the decision procedure we have described, we obtain the following characterization of the levels of the alternating index hierarchy for game languages.

PROPOSITION 5.17. *For a priority-reduced game automaton $\mathcal{A}$, $L(\mathcal{A}, q_I) \in \mathrm{RM}(i,j)$ if and only if there is no $(i+1, j+1)$-edelweiss reachable from q_I in $\mathcal{A}$.*

PROOF. One direction follows immediately from Lemma 5.11: if $L(\mathcal{A}, q_I) \notin \mathrm{RM}(i,j)$ then $\mathrm{class}(\mathcal{A}, q_I) \geq \mathrm{RM}(i+1, j+1)$ and $\mathcal{A}$ contains an $(i+1, j+1)$-edelweiss reachable from q_I. For the opposite direction assume that $\mathcal{A}$ contains an $(i+1, j+1)$-edelweiss reachable from q_I. By Lemma 5.13 it implies that $\widehat{W}_{i+1,j+1} \leq_W L(\mathcal{A}, q_I)$. Lemma 5.14 together with Proposition 5.15 imply that in that case $W_{i+1,j+1} \leq_W L(\mathcal{A}, q_I)$. By Corollary 5.9, it means that $L(\mathcal{A}, q_I) \notin \mathrm{RM}(i,j)$. □

A further corollary is the converse of Fact 2 for the alternating index.

PROPOSITION 5.18. *For game automata, substitution preserves the alternating index.*

PROOF. First note that without changing the outcome of the substitution $\mathcal{A}_{\mathcal{B}}$, we can always assume that the substituted state of $\mathcal{A}$ is an exit. We would like to use Proposition 5.17, but we first need to ensure that our automata are priority-reduced. The preprocessing that turns a given automaton into a priority reduced one, described in the proof of Lemma 5.5, works independently in each strongly-connected component of the automaton. Hence, as long as the substituted state of $\mathcal{A}$ is an exit, the preprocessing commutes with substitution; that is, $(\mathcal{A}_{\mathcal{B}})' = \mathcal{A}'_{\mathcal{B}'}$, where primes are used to denote the preprocessed automata. Consequently, we can assume that our initial automata are priority reduced, and so are the results of the substitution. The claim now follows from Proposition 5.17: since the characterization it offers is in terms of strongly connected subgraphs in the automaton, we can reason just like for non-deterministic index of deterministic automata in Proposition 4.2. If the languages recognized by automata $\mathcal{B}$ and $\mathcal{C}$ have the same index, then $\mathcal{B}$ and $\mathcal{C}$ contain the same edelweisses. By the definition of substitution, no strongly connected subgraph in $\mathcal{A}_{\mathcal{B}}$ can use states from $\mathcal{A}$ and from $\mathcal{B}$. Consequently, $\mathcal{A}_{\mathcal{B}}$ and $\mathcal{A}_{\mathcal{C}}$ contain the same edelweisses reachable from $q_I^{\mathcal{A}}$, and so $L(\mathcal{A}_{\mathcal{B}}, q_I^{\mathcal{A}})$ and $L(\mathcal{A}_{\mathcal{C}}, q_I^{\mathcal{A}})$ have the same index. □

6. WEAK ALTERNATING INDEX PROBLEM

In this section we provide a procedure computing the weak index for languages given via a game automaton recognizing them. Of course, a game language need not be weakly recognizable. This is because, as we have mentioned in Section 5, languages recognized by weak alternating automata coincide with the class $\mathrm{Comp}_0 \subseteq \mathbf{\Sigma}_1^{RM} \cap \mathbf{\Pi}_1^{RM}$, and, for each $i > 0$, there is a language recognized by a game automaton belonging to $\mathbf{\Sigma}_i^{RM} - \mathbf{\Pi}_i^{RM}$.

As an immediate corollary of the proof of Theorem 5.4, we have the following decidable characterization of being weakly recognizable for languages recognized by game automata.

FACT 5. *Let $\mathcal{A}$ be a game automaton, and q one of its states. The language $L(\mathcal{A}, q)$ is weakly recognizable if and only if $\mathcal{A}$ contains neither a $(0,1)$-edelweiss nor a $(1,2)$-edelweiss reachable from state q.*

PROOF. For the direction from left to right we reason as follows. Assume $L(\mathcal{A}, q)$ is weakly recognizable but contains, say, a $(0,1)$-edelweiss reachable from q. Then from Lemmas 5.13 and 5.14, and Corollary 5.9, we have that $L(\mathcal{A}, q) \notin \mathrm{Comp}_0$, a contradiction. The other direction is an immediate consequence of Lemma 5.11. □

This characterization of weak recognizability within the class of game automata was essentially already provided in [Niwiński and Walukiewicz 2003]. In this paper it is shown that a deterministic automaton recognizes a weakly recognizable language if and only if it does not contain a forbidden pattern called *split*, which corresponds to a $(1,2)$-edelweiss. Since game automata are closed under dualization, they can also contain a *dual split*, that is a $(0,1)$-edelweiss. Fact 5 is thence an immediate corollary from the proof of the result of [Niwiński and Walukiewicz 2003].

Analogously to what we have done in Section 5, in the aim of providing a precise formulation of the problem we want to solve, we start by introducing some useful notation.

Definition 6.1. For $i < j \in \mathbb{N}$, let $\mathbf{RM}^w(i,j)$ denote the class of languages recognized by weak alternating tree automata of index (i,j). Let

$$\begin{aligned} \mathbf{\Pi}_j^w &= \mathbf{RM}^w(0,j), \\ \mathbf{\Sigma}_j^w &= \mathbf{RM}^w(1,j+1), \\ \mathbf{\Delta}_j^w &= \mathbf{RM}^w(0,j) \cap \mathbf{RM}^w(1,j+1). \end{aligned}$$

These classes, naturally ordered by inclusion, constitute the *weak index hierarchy*. The *weak index of a language* L is the least class $\mathfrak{C}$ in the weak index hierarchy such that $L \in \mathfrak{C}$.

Now we can properly formulate the main result of this section.

THEOREM 6.2. *For a game automaton $\mathcal{A}$ and a state q, if $\mathcal{A}$ does not contain neither a $(0,1)$-edelweiss nor a $(1,2)$-edewelweiss reachable from the state q, then $L(\mathcal{A}, q)$ is weakly recognizable and its weak index can be computed effectively.*

The proof consists in a recursive procedure computing the weak class of $L(\mathcal{A}, q)$, denoted $\mathrm{wclass}(\mathcal{A}, q)$. The procedure itself is given in Subsection 6.1; Subsections 6.2 and 6.3 prove its correctness by providing upper and lower bounds, respectively. The upper bounds are simply constructions of a weak alternating automaton of appropriate weak index recognizing the language $L(\mathcal{A}, q)$. The lower bounds show that for lower indices such constructions are impossible; they are obtained by means of simple tools from descriptive set theory. In the final Subsection 6.5, we obtain as a Corollary that,

as for deterministic languages, the weak index and the Borel rank coincide for tree languages recognised by game automata.

6.1. The algorithm

Like for the strong index we assume without loss of generality that a given automaton $\mathcal{A}$ is priority-reduced (see Lemma 5.5). The procedure works recursively on the DAG of strongly-connected components, or SCCs, of $\mathcal{A}$ (maximal sets of mutually reachable states). We identify each SCC $\mathcal{B}$ of an automaton $\mathcal{A}$ with the automaton obtained by restricting $\mathcal{A}$ to the set of states in $\mathcal{B}$; the states outside of $\mathcal{B}$ accessible via a transition originating in $\mathcal{B}$ become the exits of the new automaton (cf. Subsection 2.3). Note that the resulting automaton is also priority-reduced. Our procedure computes $\mathrm{wclass}(\mathcal{A}, q)$ based on $\mathrm{wclass}(\mathcal{A}, p)$ for exits p of the SCC $\mathcal{B}$ containing q. Those classes are aggregated in a way dependent on the internal structure of $\mathcal{B}$, or more precisely, on the way in which the state p is reachable from $\mathcal{B}$. The aggregation is done by means of auxiliary operations on classes. Two most characteristic are

$$\left(\mathbf{\Pi}^w_{n-1}\right)^{\exists} = \left(\mathbf{\Delta}^w_n\right)^{\exists} = \left(\mathbf{\Sigma}^w_n\right)^{\exists} = \mathbf{\Sigma}^w_n, \quad \left(\mathbf{\Pi}^w_n\right)^{\forall} = \left(\mathbf{\Delta}^w_n\right)^{\forall} = \left(\mathbf{\Sigma}^w_{n-1}\right)^{\forall} = \mathbf{\Pi}^w_n.$$

We also use the bar notation for the dual classes,

$$\overline{\mathbf{\Pi}^w_n} = \mathbf{\Sigma}^w_n, \quad \overline{\mathbf{\Sigma}^w_n} = \mathbf{\Pi}^w_n, \quad \overline{\mathbf{\Delta}^w_n} = \mathbf{\Delta}^w_n,$$

and $\Phi \vee \Psi$ for the least class containing Φ and Ψ.

Let us now describe the conditions that will trigger applying the operations above to previously computed classes. We begin with some shorthand notation. Recall that an n-path is a path in which the minimal priority is n, and analogously for n-loop. Let q', q'' be a pair of states in $\mathcal{B}$. Let $\max_\Omega(q' \to q'')$ be the maximal n such that there exists an n-path from q' to q'' in $\mathcal{B}$. Observe that since $\mathcal{B}$ is an SCC, such n is well-defined (at least 0). Also, since the automaton is priority-reduced, for each $n' \leq \max_\Omega(q' \to q'')$ there exists an n'-path from q' to q'' in $\mathcal{B}$.

A $\forall$-*branching* transition in $\mathcal{B}$ is a transition of the form $\delta(q', a) = (q_{\mathtt{L}}, \mathtt{L}) \wedge (q_{\mathtt{R}}, \mathtt{R})$ with all three states q', $q_{\mathtt{L}}$, $q_{\mathtt{R}}$ in $\mathcal{B}$; dually for $\exists$. Note that these notions are similar but not entirely analogous to $\forall$-branching and $\exists$-branching components from Section 5.1.

We say that a state p is $(\exists, n)$-*replicated by* $\mathcal{B}$ if there are states q', q'' in $\mathcal{B}$ and a letter a such that $\delta(q', a) = (q'', \mathtt{L}) \vee (p, \mathtt{R})$ (or symmetrically) and $\max_\Omega(q'' \to q') \geq n$. Dually, p is $(\forall, n)$-*replicated* if the transition above has the form $\delta(q', a) = (q'', \mathtt{L}) \wedge (p, \mathtt{R})$ (or the symmetrical).

We can now describe the procedure. By duality we can assume that the minimal priority in $\mathcal{B}$ is 0. If $\mathcal{A}$ contains no loop reachable from q, set $\mathrm{wclass}(\mathcal{A}, q) = \mathbf{\Delta}^w_1$. If it contains an accepting loop reachable from q, but no rejecting loop reachable from q, set $\mathrm{wclass}(\mathcal{A}, q) = \mathbf{\Pi}^w_1$. Symmetrically, if it contains a rejecting loop reachable from q, but no accepting loop reachable from q, set $\mathrm{wclass}(\mathcal{A}, q) = \mathbf{\Sigma}^w_1$. Otherwise, consider the following two cases.

Assume first that $\mathcal{B}$ contains no $\forall$-branching transition. In that case, for every transition $\delta(q, a)$ of $\mathcal{B}$ that is controlled by $\forall$, at most one of the successors of $\delta(q, a)$ is a state of $\mathcal{B}$. Hence, $\mathcal{B}$ can be seen as a co-deterministic tree automaton (exits are removed from the transitions; if both states in a transition are exits, the transition is set to $\bot$). Thus, the automaton $\bar{\mathcal{B}}$ dual to $\mathcal{B}$ is a deterministic tree automaton. For deterministic tree automata it is known how to compute the weak index [Murlak 2008b]. Denote the weak index of $\bar{\mathcal{B}}$ as $\mathrm{wclass}(\bar{\mathcal{B}}, q)$.

Now, set $\mathrm{wclass}(\mathcal{A}, q)$ to

$$\mathbf{\Delta}^w_2 \vee \overline{\mathrm{wclass}(\bar{\mathcal{B}}, q)} \vee \bigvee_{p \in F} \mathrm{wclass}(\mathcal{A}, p) \vee \bigvee_{p \in F_{\exists,1}} \mathrm{wclass}(\mathcal{A}, p)^{\exists} \vee \bigvee_{p \in F_{\forall,0}} \mathrm{wclass}(\mathcal{A}, p)^{\forall} \qquad (1)$$

where $F \subseteq Q^{\mathcal{A}}$ is the set of exits of $\mathcal{B}$, $F_{\exists,1} \subseteq F$ is the set of states $(\exists, 1)$-replicated by $\mathcal{B}$, and similarly for $F_{\forall,0}$.

Assume now that $\mathcal{B}$ does contain an $\forall$-branching transition. By the hypothesis of the theorem, for every $\forall$-branching transition $\delta(q', a) = (q_{\mathtt{L}}, \mathtt{L}) \wedge (q_{\mathtt{R}}, \mathtt{R})$ in $\mathcal{B}$, it must hold that $\max_\Omega(q_{\mathtt{L}} \to q') \leq 1$ and $\max_\Omega(q_{\mathtt{R}} \to q') = 0$, or symmetrically. We call a state q'' (either $q_{\mathtt{L}}$ or $q_{\mathtt{R}}$) in an $\forall$-branching transition *bad* if $\max_\Omega(q'' \to q') = 0$. Let $\mathcal{B}^-$ be the automaton $\mathcal{B}$ with all these *bad* states in the $\forall$-branching transitions changed into $\top$, and let $\mathcal{A}^-$ be the automaton $\mathcal{A}$ with $\mathcal{B}$ replaced by $\mathcal{B}^-$. Observe that $\mathcal{B}^-$ contains no $\forall$-branching transitions. Let us put

$$\mathrm{wclass}(\mathcal{A}, q) = \mathbf{\Delta}_2^w \vee \big(\mathrm{wclass}(\mathcal{A}^-, q)\big)^\forall. \tag{2}$$

6.2. Upper bounds

In this section we prove that $\mathrm{wclass}(\mathcal{A}, q)$ is an upper bound for the weak index of $L(\mathcal{A}, q)$. More precisely, we show the following.

LEMMA 6.3. *If* $\mathrm{wclass}(\mathcal{A}, q) \leq \mathbf{RM}^w(i, j)$ *then the language* $L(\mathcal{A}, q)$ *can be recognised by a weak alternating automaton of index* (i, j).

Let us first deal with the lowest levels. The algorithm never returns $\mathbf{\Pi}_0^w = \mathbf{RM}^w(0, 0)$ nor $\mathbf{\Sigma}_0^w = \mathbf{RM}^w(1, 1)$, so the lowest (i, j) we need to consider are $(0, 1)$ and $(1, 2)$. Suppose that $(i, j) = (0, 1)$. Examining the algorithm we immediately see that this is possible only if automaton $\mathcal{A}$ does not contain a rejecting loop reachable from state q. Since our automaton is priority reduced, it means that it uses only priority 0. Hence, it is already a $(0, 1)$ weak automaton (not $(0, 0)$, because of allowed $\bot$ transitions). For $(i, j) = (1, 2)$ the argument is entirely analogous.

For higher indices we consider three cases, leading to three different constructions of weak alternating automata recognizing $L(\mathcal{A}, q)$.

6.2.1. $\mathcal{B}$ has no $\forall$-branching transitions and $(i, j) = (1, j)$ with $j \geq 3$. In an initial part of the weak automaton recognizing $L(\mathcal{A}, q)$ the players declare whether during the play on a given tree they would leave the $\mathcal{B}$ component or not. Since $\mathcal{B}$ has no $\forall$-branching transitions, as long as the play has not left $\mathcal{B}$, each choice of $\forall$ amounts to leaving $\mathcal{B}$ or staying in $\mathcal{B}$. Hence, each strategy of $\exists$ admits exactly one path staying in $\mathcal{B}$, finite or infinite. We first let $\exists$ declare $l_\exists \in \{\textit{leave}, \textit{stay}\}$, where *leave* means that the path is finite, and *stay* means that it is infinite.

— If $l_\exists = \textit{leave}$, we move to a copy of $\mathcal{B}$ with all the priorities set to 1. By Equation (1), for every exit f of $\mathcal{B}$ we have $\mathrm{wclass}(\mathcal{A}, f) \leq \mathbf{RM}^w(1, j)$. Therefore, we can compose this copy of $\mathcal{B}$ with all the automata for $L(\mathcal{A}, f)$ to obtain an automaton of index $(1, j)$.
— Assume that $l_\exists = \textit{stay}$. Given the special shape of $\exists$'s strategies, this means that $\exists$ claims that the play will only leave $\mathcal{B}$ if at some point $\forall$ chooses an exit f in a transition whose other end is in $\mathcal{B}$. Since the minimal priority in $\mathcal{B}$ is 0, all these exists are $(\forall, 0)$-replicated. We check $\exists$'s claim by substituting all other exits in transitions with rejecting states, i.e. weak alternating automata of index $(3, 3)$ (recall that j is at least 3). Thus, the only exits that are not substituted are the $(\forall, 0)$-replicated ones. Now, we ask $\forall$ whether he plans to take one of these exists: he declares $l_\forall \in \{\textit{leave}, \textit{stay}\}$, accordingly.
  — If $l_\forall = \textit{stay}$, the play moves to the weak alternating automaton of index $\overline{\mathrm{wclass}_{\mathrm{det}}(\bar{\mathcal{B}})}$, corresponding to the co-deterministic automaton $\mathcal{B}$ with the remaining exits removed from transitions (they were only present in transitions of the form $(q_{\mathtt{L}}, \mathtt{L}) \wedge (q_{\mathtt{R}}, \mathtt{R})$, with the other state in $\mathcal{B}$).
  — Assume that $l_\forall = \textit{leave}$. In that case we move to a copy of $\mathcal{B}$ with all the priorities set to 2. The only exits left are the $(\forall, 0)$-replicated ones. By Equation (1), for all

such exists f,

$$\mathrm{wclass}(\mathcal{A}, f) \leq \mathbf{RM}^w(0, j-2)\,,$$

for otherwise $\mathrm{wclass}(\mathcal{A}, f) \geq \mathbf{RM}^w(1, j-1)$, so $\big(\mathrm{wclass}(\mathcal{A}, p)\big)^\forall \geq \mathbf{RM}^w(0, j-1)$ and $\mathbf{RM}^w(0, j-1)$ is not smaller than $\mathbf{RM}^w(1, j)$. In particular, we can find a weak alternating automaton of index $(2, j)$ recognizing $L(\mathcal{A}, f)$. So the whole sub-automaton is a weak alternating automaton of index $(2, j)$.

6.2.2. $\mathcal{B}$ has no $\forall$-branching transitions and $(i, j) = (0, j)$ with $j \geq 2$. The simulation starts in a copy of $\mathcal{B}$ with all the priorities set to 0. If the play leaves $\mathcal{B}$ at this stage then we move to the appropriate automaton of index $(0, j)$. At any moment $\forall$ can pledge that:

— the play will no longer visit transitions $\delta(q', a)$ of the form $(f_{\mathtt{L}}, \mathtt{L}) \wedge (f_{\mathtt{R}}, \mathtt{R})$, $(f_{\mathtt{L}}, \mathtt{L}) \vee (f_{\mathtt{R}}, \mathtt{R})$, $(q_{\mathtt{L}}, \mathtt{L}) \vee (f_{\mathtt{R}}, \mathtt{R})$, $(f_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R})$, or $(q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R})$, where $\max_\Omega(q_{\mathtt{L}} \to q') = \max_\Omega(q_{\mathtt{R}} \to q') = 0$ and $f_{\mathtt{L}}$, $f_{\mathtt{R}}$ are exits of $\mathcal{B}$;
— in the transitions he controls, he will always choose the state in $\mathcal{B}$, and win regardless of $\exists$'s choices.

If the play stays forever in $\mathcal{B}$ but $\forall$ is never able to make such a pledge, he loses by the parity condition—it means that infinitely many times a loop from $q_{\mathtt{L}} \to q'$ or $q_{\mathtt{R}} \to q'$ is taken with $\max_\Omega(q_d \to q') = 0$ therefore, the minimal priority occurring infinitely often is 0.

After $\forall$ has made the above pledge, $\exists$ has the following choices:

— She can challenge the first part of $\forall$'s pledge, declaring that at least one such transition is reachable. In that case we move to a copy of $\mathcal{B}$ with all the priorities set to 1 and all the transitions controlled by $\exists$. In this copy, reaching any of the disallowed transitions entails acceptance—the play immediately moves to a $(2, 2)$ final component.
— She can accept the first part of $\forall$'s pledge.

After $\exists$ has accepted the first part of $\forall$'s pledge, we can assume that the rest of the game in $\mathcal{B}$ is a single infinite branch. Indeed, by the hypothesis of the theorem, for every $\exists$-branching transition $\delta(q', a) = (q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R})$ in $\mathcal{B}$ it must hold that $\max_\Omega(q_{\mathtt{L}} \to q') = \max_\Omega(q_{\mathtt{R}} \to q') = 0$; otherwise, $\mathcal{B}$ would contain $(0, 1)$-edelweiss.Thus, no $\exists$-branching transition can be reached, and since $\mathcal{B}$ contains no $\forall$-branching transitions at all, the game can continue in $\mathcal{B}$ in only one way.

Now $\exists$ must challenge the second part of $\forall$'s pledge. We ask her whether she plans to leave $\mathcal{B}$ or not, and she declares $l_\exists \in \{\mathit{leave}, \mathit{stay}\}$.

— If $l_\exists = \mathit{stay}$ then we proceed to the weak automaton of index $\mathrm{wclass}(\mathcal{B}, q)$, corresponding to $\mathcal{B}$ treated as a co-deterministic automaton. We are only interested in the behaviour of this automaton over trees in which there is exactly one branch in $\mathcal{B}$, and it is infinite. Over such trees we want to make sure that neither player ever chooses to exit. This is already ensured: when $\mathcal{B}$ is turned into a co-deterministic tree automaton, the exits are simply removed from transitions (if both states are exits, the transition is changed to a transition to a $(2, 2)$ automaton, but such transitions will never be used over trees we are interested in).
— If $l_\exists = \mathit{leave}$ then we move to a copy of $\mathcal{B}$ with all the priorities set to 1. The only available exits of $\mathcal{B}$ in this copy are those in transitions of the form $\delta(q', q) = (q_{\mathtt{L}}, \mathtt{L}) \vee (f, \mathtt{R})$ (or symmetrical) with $\max_\Omega(q_{\mathtt{L}} \to q') > 0$ (in other transitions the exits are removed, if both states are exits, they are replaced by a final $(2, 2)$-component); therefore $\mathrm{wclass}(\mathcal{A}, f) \leq \mathbf{RM}^w(1, j)$ and we can simulate it with a $(1, j)$-automaton.

6.2.3. $\mathcal{B}$ contains $\forall$-branching transitions. If $\mathcal{B}$ contains an $\forall$-branching transition, the algorithm returns $\mathrm{wclass}(\mathcal{A}, q)$ of the form $\mathbf{RM}^w(0, j)$. Let us construct a weak automaton of index $(0, j)$ that recognizes $L(\mathcal{A}, q)$. The automaton starts in a copy of $\mathcal{B}$ with all the priorities set to 0. At any moment $\forall$ can declare that no-one will ever take any *bad* transition in $\mathcal{B}$. If he cannot make such a declaration, it means that $\exists$ can force infinitely many bad transitions to be taken, and she wins. After $\forall$ has made such declaration, we need to recognize the language $L(\mathcal{A}^-, q)$ (note that the bad transitions in $\mathcal{A}^-$ are made directly losing for $\forall$). For this we can use a weak automaton of index $\mathrm{wclass}(\mathcal{A}^-) \leq \mathbf{RM}^w(0, j)$, already constructed.

6.3. Lower bounds

Now we prove lower bounds for the weak index computed by our procedure, as expressed by Lemma 6.4.

LEMMA 6.4. *If* $\mathrm{wclass}(\mathcal{A}, q) \geq \mathbf{RM}^w(i, j)$ *then* $L(\mathcal{A}, q)$ *cannot be recognised by a (total) weak alternating automaton of index* $(i + 1, j + 1)$.

For this we use a topological argument, relying on the following simple observation [Duparc and Murlak 2007], essentially proved already by Mostowski [Mostowski 1991b]. Let $\mathbf{\Pi}^0_n$, $\mathbf{\Sigma}^0_n$, and $\mathbf{\Delta}^0_n$ be the finite Borel classes; that is, $\mathbf{\Sigma}^0_1$ is the class of the open sets, $\mathbf{\Pi}^0_n$ consists of the complements of sets from $\mathbf{\Sigma}^0_n$, $\mathbf{\Delta}^0_n = \mathbf{\Sigma}^0_n \cap \mathbf{\Pi}^0_n$, and $\mathbf{\Sigma}^0_{n+1}$ consists of countable unions of sets from $\mathbf{\Pi}^0_n$.

FACT 6. *If* L *is recognizable by a weak alternating automaton of index* $(0, j)$ *then* $L \in \mathbf{\Pi}^0_j$. *Dually, for index* $(1, j + 1)$, *we have* $L \in \mathbf{\Sigma}^0_j$.

Thus, in order to show that a language is *not* recognizable by weak alternating automaton of index $(0, j)$ it is enough to show that it is not in $\mathbf{\Pi}^0_j$. This can be shown by providing a continuous reduction to L from some language not in $\mathbf{\Pi}^0_j$, e.g. a $\mathbf{\Sigma}^0_j$-complete language. We shall use languages introduced by Skurczyński [Skurczyński 1993].

One can define Skurczyński's languages by means of two dual operations on tree languages.

Definition 6.5. For $L \subseteq \mathrm{Tr}_A$ define

$$L^\forall = \left\{ t \in \mathrm{Tr}_A \;\middle|\; \forall_{n \in \mathbb{N}}\, t\restriction_{\mathtt{L}^n\mathtt{R}} \in L \right\}, \quad L^\exists = \left\{ t \in \mathrm{Tr}_A \;\middle|\; \exists_{n \in \mathbb{N}}\, t\restriction_{\mathtt{L}^n\mathtt{R}} \in L \right\}.$$

It is straightforward to check that these operations are monotone with respect to the Wadge ordering; that is,

$$L \leq_{\mathrm{W}} M \text{ implies } L^\forall \leq_{\mathrm{W}} M^\forall \text{ and } L^\exists \leq_{\mathrm{W}} M^\exists.$$

Moreover, for all $n > 0$,

— if L is $\mathbf{\Sigma}^0_n$-complete, $L^\forall$ is $\mathbf{\Pi}^0_{n+1}$-complete, and
— if L is $\mathbf{\Pi}^0_n$-complete, $L^\exists$ is $\mathbf{\Sigma}^0_{n+1}$-complete.

This allows us to define simple tree languages complete for finite levels of the Borel hierarchy.

Definition 6.6 (*[Skurczyński 1993]*). Consider the alphabet $A = \{\bot, \top\}$. Let

$$S_{(0,1)} = \left\{ t \in \mathrm{Tr}_A \;\middle|\; t(\epsilon) = \top \right\}^\forall, \quad S_{(1,2)} = \left\{ t \in \mathrm{Tr}_A \;\middle|\; t(\epsilon) = \bot \right\}^\exists.$$

The remaining languages are defined inductively,

$$S_{(0,j+1)} = (S_{(1,j+1)})^\forall, \quad S_{(1,j+1)} = (S_{(0,j-1)})^\exists.$$

For notational convenience, let $S_{(0,0)} = \mathrm{Tr}_A$ and $S_{(1,1)} = \emptyset$.

Note that the languages are dual to each other: $S_{(1,j+1)} = \mathrm{Tr}_A - S_{(0,j)}$. A straightforward reduction shows that $S_{(i',j')} \leq_{\mathrm{W}} S_{(i,j)}$ whenever (i,j) is at least (i',j'). But the crucial property is the following.

FACT 7 ([SKURCZYŃSKI 1993]). *$S_{(0,n)} \in \mathbf{\Pi}^0_n - \mathbf{\Sigma}^0_n$ and $S_{(1,n+1)} \in \mathbf{\Sigma}^0_n - \mathbf{\Pi}^0_n$.*

Summing up, from Facts 6 and 7 it follows immediately that if $S_{(i,j)} \leq_W L$ then L is not recognizable by a weak alternating automaton of index $(i+1, j+1)$.

Observe that $S_{(i,j)}$ can be recognised by a weak game automaton of index (i,j). One consequence of this—and Facts 6 and 7—is the strictness of the hierarchy.

COROLLARY 6.7. *The weak index hierarchy is strict, even when restricted to languages recognizable by game automata.*

Another consequence is that it is relatively easy to give the reductions we need to prove Lemma 6.4, summarized in the claim below.

CLAIM 1. *If* $\mathrm{wclass}(\mathcal{A}, q) \geq \mathbf{RM}^w(i,j)$ *then* $S_{(i,j)} \leq_{\mathrm{W}} L(\mathcal{A}, q)$.

We prove this claim by induction on the structure of the DAG of SCCs of $\mathcal{A}$ reachable from q, following the cases of the algorithm just like for the upper bound. One of the cases is covered by the procedure for deterministic automata, which we use as a black box. But in order to prove Lemma 6.4 we need to know that it preserves our invariant. And indeed, just like here, it is a step in the correctness proof: if the procedure returns at least $\mathbf{RM}^w(i,j)$, then $S_{(i,j)}$ continuously reduces to the recognised language [Murlak 2008b].

The remaining cases essentially correspond to the items in the following lemma.

LEMMA 6.8. *Assume that q is a state of $\mathcal{A}$, $\mathcal{B}$ is the SCC of $\mathcal{A}$ containing q, and p is a state of $\mathcal{A}$ reachable from q (from the same or different SCC).*

(1) $L(\mathcal{A}, p) \leq_{\mathrm{W}} L(\mathcal{A}, q)$.
(2) $L(\mathcal{A}^-, q) \leq_{\mathrm{W}} L(\mathcal{A}, q)$.
(3) *If an accepting loop is reachable from q, then* $S_{(0,1)} \leq_{\mathrm{W}} L(\mathcal{A}, q)$.
(4) *If a rejecting loop is reachable from q, then* $S_{(1,2)} \leq_{\mathrm{W}} L(\mathcal{A}, q)$.
(5) *If p is $(\forall, 0)$-replicated by $\mathcal{B}$ then* $(L(\mathcal{A}, p))^{\forall} \leq_{\mathrm{W}} L(\mathcal{A}, q)$.
(6) *If p is $(\exists, 1)$-replicated by $\mathcal{B}$ then* $(L(\mathcal{A}, p))^{\exists} \leq_{\mathrm{W}} L(\mathcal{A}, q)$.

PROOF. The proof is based on Fact 3. Let us begin with (1). Since all the states of $\mathcal{A}$ are non-trivial, we can construct a tree t with a hole h such that t resolves $\mathcal{A}$ from q and the state $\rho(\mathcal{A}, t, q)(h)$ is p. In that case $t[h := s] \in L(\mathcal{A}, q)$ if and only if $s \in L(\mathcal{A}, p)$. Therefore, the function $s \mapsto t[h := s]$ is a continuous reduction witnessing that $L(\mathcal{A}, p) \leq_{\mathrm{W}} L(\mathcal{A}, q)$.

For (2), recall that $\mathcal{A}^-$ is obtained from $\mathcal{A}$ by turning some choices for $\forall$ to $\top$; that is, some transitions $\delta(r, a)$ of the form $(r_{\mathtt{L}}, \mathtt{L}) \wedge (r_{\mathtt{R}}, \mathtt{R})$ are set to $(r_{\mathtt{L}}, \mathtt{L})$, $(r_{\mathtt{R}}, \mathtt{R})$, or $\top$. This means that if a node v of tree t has label a and gets state p in the associated run $\rho(\mathcal{A}^-, t, q)$, then $t \restriction_{v\mathtt{L}}$, $t \restriction_{v\mathtt{R}}$, or both of them, respectively, are immediately accepted by $\mathcal{A}^-$. In the corresponding run of the original automaton $\mathcal{A}$, however, these subtrees will be inspected by the players and we should make sure they are accepted. The way to do it is simple: since $r_{\mathtt{L}}$ and $r_{\mathtt{R}}$ are non-trivial in $\mathcal{A}$, we can replace these subtrees with $t_{r_{\mathtt{L}}} \in L(\mathcal{A}, r_{\mathtt{L}})$, or $t_{r_{\mathtt{R}}} \in L(\mathcal{A}, r_{\mathtt{R}})$, accordingly. This gives a continuous reduction of $L(\mathcal{A}^-, q)$ to $L(\mathcal{A}, q)$.

To prove (3), let us fix a state p on an accepting loop C, reachable from q. By (1) and transitivity of $\leq_{\mathrm{W}}$, it is enough to show that $S_{(0,1)} \leq_{\mathrm{W}} L(\mathcal{A}, p)$. Let t be a tree with hole h such that t resolves $\mathcal{A}$ from p, the state $\rho(\mathcal{A}, t, p)$ is p, and the states on the shortest

path from the root to h correspond to the accepting loop C. Since all states in $\mathcal{A}$ are non-trivial, we can also find a full tree $t' \notin L(\mathcal{A}, p)$. Let $t_0 = t'$ and $t_n = t[h := t_{n-1}]$ for $n > 0$, and let t_∞ be the tree defined co-inductively as

$$t_\infty = t[h := t_\infty]\,.$$

Then, $t_n \notin L(\mathcal{A}, p)$ for all $n \geq 0$, but $t_\infty \in L(\mathcal{A}, p)$. To get a continuous function reducing $S_{(0,1)}$ to $L(\mathcal{A}, p)$, map tree $s \in \mathrm{Tr}_{\{\bot,\top\}}$ to t_m, where $m = \min\left\{i \;\middle|\; s(\mathtt{L}^i\mathtt{R}) = \bot\right\}$, or to t_∞ if $\left\{i \;\middle|\; s(\mathtt{L}^i\mathtt{R}) = \bot\right\}$ is empty.

Item (4) is analogous.

For (5), let us assume that $\delta(q, a) = (q_\mathtt{L}, \mathtt{L}) \wedge (p, \mathtt{R})$ is the transition witnessing that p is $(\forall, 0)$-replicated by $\mathcal{A}$. Let us also fix the path $q_\mathtt{L} \to q$ with minimal priority 0. Now, let t be a tree with a hole h that resolves $\mathcal{A}$ from q and the value of the run of $\mathcal{A}$ in h is q. Similarly, let t' be the tree with a hole h' that resolves $\mathcal{A}$ from q_L and the value of the respective run is q. Let us construct a continuous function that reduces $(L(\mathcal{A}, p))^\forall$ to $L(\mathcal{A}, q)$. Assume that a given tree s has subtrees s_i under the nodes L^iR. Let us define co-inductively t_i as

$$t_i = a(t'[h' := t_{i+1}], s_i),$$

i.e. the tree with the root labelled by a and two subtrees: $t'[h' := t_{i+1}]$ and s_i. Finally, let $f(s)$ be $t[h := t_0]$. Note that the run $\rho(\mathcal{A}, f(s), q)$ labels the hole h of t by q'. Therefore, $f(t) \in L(\mathcal{A}, q)$ if and only if $t_0 \in L(\mathcal{A}, q')$ and $t_i \in L(\mathcal{A}, q)$ if and only if $t_{i+1} \in L(\mathcal{A}, q)$ and $s_i \in L(\mathcal{A}, p)$. Since the minimal priority on the path from t_i to t_{i+1} is 0, if no s_i belongs to $L(\mathcal{A}, p)$ then $f(t) \notin L(\mathcal{A}, q)$. Therefore, f is in fact the desired reduction.

The proof of (6) is entirely analogous. □

Using Lemma 6.8, and the guarantees for deterministic automata discussed earlier, we prove Lemma 6.4 as follows.

PROOF OF LEMMA 6.4. By induction on the recursion depth of the algorithm execution we prove that if $\mathrm{wclass}(\mathcal{A}, p) \geq \mathbf{RM}^w(i, j)$ then $S_{(i,j)} \leq_\mathrm{W} L(\mathcal{A}, p)$.

Let us start with the lowest level. Assume that $(i, j) = (0, 1)$ (for $(1, 2)$ the proof is analogous). Examining the algorithm we see that this is only possible if there is an accepting loop in $\mathcal{A}$, reachable from q. Then, by Lemma 6.8 Item (3), $S_{(0,1)} \leq_\mathrm{W} L(\mathcal{A}, q)$.

For higher levels we proceed by case analysis. First we cover the possible reasons why equation (1) can give at least $\mathbf{RM}^w(i, j)$. If $\overline{\mathrm{wclass}(\bar{\mathcal{B}}, q)} \geq \mathbf{RM}^w(i, j)$, the invariant follows immediately from the guarantees for deterministic automata, and the duality between indices and between Skurczyński languages. If $\mathrm{wclass}(\mathcal{A}, p) \geq \mathbf{RM}^w(i, j)$ for some $p \in F$, we use the fact that $L(\mathcal{A}, p) \leq_\mathrm{W} L(\mathcal{A}, q)$, and get $S_{(i,j)} \leq_\mathrm{W} L(\mathcal{A}, q)$ by transitivity. Then, assume that $\mathrm{wclass}(\mathcal{A}, p)^\exists \geq \mathbf{RM}^w(i, j)$ for some $p \in F_{\exists,1}$ (for $p \in F_{\forall,0}$ the proof is analogous). That means that $\mathrm{wclass}(\mathcal{A}, p) \geq \mathbf{RM}^w(0, j')$ such that $(\mathbf{RM}^w(0, j'))^\exists = \mathbf{RM}^w(1, j' + 2) \geq \mathbf{RM}^w(i, j)$. By the inductive hypothesis $S_{(0,j')} \leq_\mathrm{W} L(\mathcal{A}, p)$, so by the monotonicity of $\exists$ and Lemma 6.8 Item (6), $S_{(1,j'+2)} = \left(S_{(0,j')}\right)^\exists \leq_\mathrm{W} (L(\mathcal{A}, p))^\exists \leq_\mathrm{W} L(\mathcal{A}, q)$. But since $\mathbf{RM}^w(1, j' + 2) \geq \mathbf{RM}^w(i, j)$, by the Wadge ordering of Skurczyński's languages $S_{(i,j)} \leq_\mathrm{W} S_{(1,j'+2)}$, and consequently $S_{(i,j)} \leq_\mathrm{W} L(\mathcal{A}, q)$ follows by transitivity.

Finally, assume that $\mathrm{wclass}(\mathcal{A}, q)$ is computed according to (2); that is, the component $\mathcal{B}$ contains an $\forall$-branching transition $\delta(q', a) = (q_\mathtt{L}, \mathtt{L}) \wedge (q_\mathtt{R}, \mathtt{R})$. As we have already observed, the hypothesis of the theorem implies that in that case $\max_\Omega(q_\mathtt{L} \to q') = 0$ and $\max_\Omega(q_\mathtt{R} \to q') \leq 1$ (or symmetrically). That means that $q_\mathtt{R}$ is $\forall, 0$-replicated by $\mathcal{B}$, so by Lemma 6.8 Item (5), $(L(\mathcal{A}, q_\mathtt{R}))^\forall \leq_\mathrm{W} L(\mathcal{A}, q)$. But since $\mathcal{B}$ is strongly connected, q is reachable from $q_\mathtt{L}$ and $q_\mathtt{L}$, so by Lemma 6.8 Item 1 we have $L(\mathcal{A}, q) \leq_\mathrm{W} L(\mathcal{A}, q_\mathtt{R})$.

Since $\forall$ is monotone, we conclude that

$$(L(\mathcal{A},q))^{\forall} \leq_{\mathrm{W}} L(\mathcal{A},q)\,. \tag{3}$$

(Although it looks paradoxical, it is not the case since $(L^{\forall})^{\forall} \leq_{\mathrm{W}} L^{\forall}$ for all L.) Since $\mathrm{wclass}(\mathcal{A},q) \geq \mathbf{RM}^w(i,j)$, it must hold that $\mathrm{wclass}(\mathcal{A}^-,q) \geq \mathbf{RM}^w(1,j')$, such that $(\mathbf{RM}^w(1,j'))^{\forall} = \mathbf{RM}^w(0,j') \geq \mathbf{RM}^w(i,j)$. By the induction hypothesis, $S_{(0,j')} \leq_{\mathrm{W}} L(\mathcal{A}^-,q)$. Consequently, by Lemma 6.8 Item (2) and by transitivity, $S_{(0,j')} \leq_{\mathrm{W}} L(\mathcal{A},q)$. Thus, by the monotonicity of $\forall$, from (3) we get $S_{(0,j')} = \left(S_{(1,j')}\right)^{\forall} \leq_{\mathrm{W}} L(\mathcal{A},q)$, and we conclude by the Wadge ordering of Skurczyński's languages. □

6.4. Substitution preserves the weak alternating index

A quick look at how the weak index is computed suffices to show the converse of Fact 2 for weak alternating index.

PROPOSITION 6.9. *For game automata, substitution preserves the weak alternating index.*

PROOF. As we have argued in the proof of Proposition 5.18, we can assume that our automata are priority reduced and so are the results of the substitutions. Notice that $\mathrm{wclass}(\mathcal{A},q)$ depends only on the internal structure of the SCC containing q and the value of $\mathrm{wclass}(\mathcal{A},q)$ computed for the states outside this component. Therefore, substituting either of two game automata recognizing languages of the same weak index results in the same outcome of the procedure. The claim follows by the correctness of the procedure. □

6.5. Weak index VS Borel rank

Another notable feature of tree languages recognised by deterministic automata is that within this class, the properties of being Borel and being weakly recognizable are coextensive. Since the former is decidable [Niwiński and Walukiewicz 2003], the latter is also decidable. This correspondence can be made even more precise: for languages recognised by deterministic automata, the weak index and the Borel rank coincide [Murlak 2008b]. Notice that this implies that the Borel rank for deterministic languages is also decidable, a result originally proved in [Murlak 2005]. As a corollary of the work presented in the previous part of this Section, we obtain that the same is true for game automata.

COROLLARY 6.10. *Under restriction to languages recognised by game automata, the weak index hierarchy coincides with the Borel hierarchy, and both are decidable.*

PROOF. From [Murlak 2008b], we know that if $\mathrm{wclass}(\mathcal{A},q) \leq \mathbf{RM}^w(i,j)$ then $L(\mathcal{A},q) \leq_{\mathrm{W}} S_{(i,j)}$. The coincidence between weak index and Borel rank thence follows by applying Claim 1 and Fact 7. Decidability is a consequence of Theorem 6.2. □

7. RECOGNIZABILITY BY GAME AUTOMATA

In this section we give an effective characterization of the class of languages recognized by game automata within the class of all regular languages. The characterization is inspired by the one for deterministic automata [Niwiński and Walukiewicz 2003], however, due to the alternation of players, the arguments here are more involved.

We begin with a handful of definitions. Let us fix a finite alphabet A. A *trace* is a finite word w over $A \cup \{\mathtt{L},\mathtt{R}\}$, with letters from A on even positions, and directions from $\{\mathtt{L},\mathtt{R}\}$ on odd positions. If the last symbol of w is a letter, the trace is *labelled*, otherwise it is *unlabelled*. A trace w can be seen as a partial tree $t_w \in \mathrm{PTr}_A$ consisting of a single path: for a labelled trace $w = a_0 d_1 a_1 \dots d_k a_k$, $\mathrm{dom}(t_w) = \{d_1 d_2 \dots d_i \mid i \leq k\}$ and

$t_w(d_1 d_2 \dots d_i) = a_i$ for all $i \leq k$. Abusing the notation, we write w instead of $d_1 d_2 \dots d_k$. The tree t_w has two *final holes*, $w\mathtt{L}$ and $w\mathtt{R}$, and *side holes* $d_1 d_2 \dots d_{i-1}\bar{d}_i$ for $i \leq k$. For an unlabelled trace $w = a_0 d_1 a_1 \dots d_k$, t_w is defined similarly, but this time it has only one final hole: $d_1 d_2 \dots d_k$. We shall also write w for this hole.

A partial tree $t \in \mathrm{PTr}_A$ is a *realization* of a trace w if it is obtained from t_w by putting some *total* trees in all the side holes of t_w. If w is an unlabelled trace, t still has a hole w. We write $t(t')$ for the tree obtained by putting t' in the hole w, and $t^{-1}M$ for $\{t' \mid t(t') \in M\}$. Similarly, if w is a labelled trace, we write $t(t_\mathtt{L}, t_\mathtt{R})$ for the total tree obtained by putting $t_\mathtt{L}$, $t_\mathtt{R}$ in the holes $w\mathtt{L}$ and $w\mathtt{R}$, respectively, and we define $t^{-1}M$ as $\{(t_\mathtt{L}, t_\mathtt{R}) \mid t(t_\mathtt{L}, t_\mathtt{R}) \in M\}$. Additionally, $a^{-1}M$ stands for $t_a^{-1}M$ for the root-only tree t_a with $t_a(\varepsilon) = a$.

A language Z is *non-trivial* if neither Z nor its complement $Z^\complement$ is empty. The following notions are semantic counter-parts of states and transitions of game automata.

Definition 7.1. A *unary profile* is $*$ (standing for trivial) or a non-trivial regular tree language Z. A *binary profile* is $*$, $\emptyset$, $\mathrm{Tr}_A \times \mathrm{Tr}_A$, or a subset of $\mathrm{Tr}_A \times \mathrm{Tr}_A$ in one of the forms $Z_\mathtt{L} \times \mathrm{Tr}_A$, $\mathrm{Tr}_A \times Z_\mathtt{R}$, $(Z_\mathtt{L} \times \mathrm{Tr}_A) \cup (\mathrm{Tr}_A \times Z_\mathtt{R})$, or $Z_\mathtt{L} \times Z_\mathtt{R}$, for some non-trivial regular tree languages $Z_\mathtt{L}, Z_\mathtt{R}$; the first three (i.e. $*$, $\emptyset$, and $\mathrm{Tr}_A \times \mathrm{Tr}_A$) are called trivial and the remaining ones non-trivial.

We shall see that the binary profiles (except $*$) correspond to transitions of the form $\bot$, $\top$, $(q_\mathtt{L}, \mathtt{L})$, $(q_\mathtt{R}, \mathtt{R})$, $(q_\mathtt{L}, \mathtt{L}) \vee (q_\mathtt{R}, \mathtt{R})$, and $(q_\mathtt{L}, \mathtt{L}) \wedge (q_\mathtt{R}, \mathtt{R})$, respectively. As a first step, let us relate traces to profiles.

Definition 7.2. A trace w has non-trivial profile Z in a regular language M, if for each realization t of w either $t^{-1}M$ is trivial or $t^{-1}M = Z$, and for some realization t_0, $t_0^{-1}M = Z$; here Z is unary for unlabelled w and binary for labelled w.

An unlabelled trace w has profile $*$ in M if for each realization t of w, $t^{-1}M$ is trivial. A labelled trace wa has profile $Z \in \{\emptyset, \mathrm{Tr}_A \times \mathrm{Tr}_A\}$ in M if w has a non-trivial profile Z' and $a^{-1}Z' = Z$; if w has profile $*$, so does the trace wa.

Note that each trace, labelled or unlabelled, has at most one profile in M. We write p_M for the partial function assigning profiles to traces. We say that M is *locally game* if each trace has a profile in M. The following lemma shows that it is equivalent to assume that all unlabelled traces have profiles in M.

LEMMA 7.3. *Given the profiles of traces w, $wa\mathtt{L}$ and $wa\mathtt{R}$ in M, one can effectively compute the profile of wa in M.*

PROOF. Let us assume that w has a non-trivial profile $K \subseteq \mathrm{Tr}_A$, and $wa\mathtt{L}$ and $wa\mathtt{R}$ have profiles $K_\mathtt{L}$ and $K_\mathtt{R}$. Then, by Definition 7.2, wa cannot have profile $*$. Let $a^{-1}K = \big\{(s,t) \in \mathrm{Tr}_A \times \mathrm{Tr}_A \mid a(s,t) \in K\big\}$. Is easy to see that wa has a profile $Z \subseteq \mathrm{Tr}_A \times \mathrm{Tr}_A$ if and only if $Z = a^{-1}K$. Thus, it remains to check that $a^{-1}K$ is of one of the forms allowed by Definition 7.1.

For a set $U \subseteq X \times Y$, we define the *lower section of U by $x \in X$* as $U_x = \{y \in Y \mid (x,y) \in U\}$, and the *upper section of U by $y \in Y$* as $U^y = \{x \in X \mid (x,y) \in U\}$.

Since $wa\mathtt{L}$ and $wa\mathtt{R}$ have profiles $K_\mathtt{L}$ and $K_\mathtt{R}$, respectively, it follows easily that

— each lower section of $a^{-1}K$ is either $\emptyset$, or Tr_A, or $K_\mathtt{R}$; and
— each upper section of $a^{-1}K$ is either $\emptyset$, or Tr_A, or $K_\mathtt{L}$.

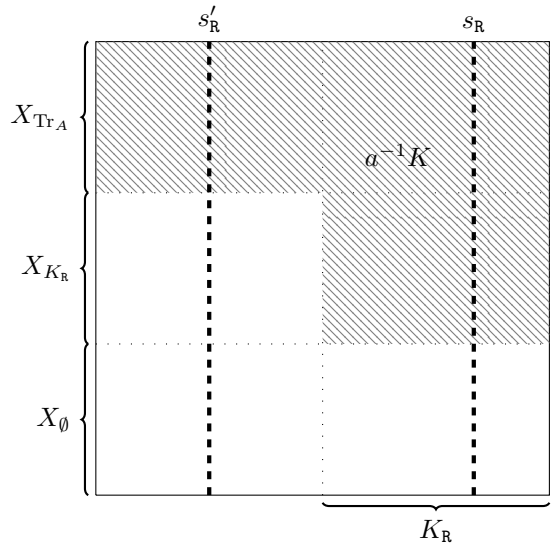

Fig. 5. An illustration of the set $a^{-1}K$ split into a union of products.

The following three sets form a partition of Tr_A:

$$\begin{aligned} X_{\mathrm{Tr}_A} &= \left\{ s_L \;\middle|\; (a^{-1}K)_{s_{\mathtt{L}}} = \mathrm{Tr}_A \right\}, \\ X_{K_{\mathtt{R}}} &= \left\{ s_L \;\middle|\; (a^{-1}K)_{s_{\mathtt{L}}} = K_{\mathtt{R}} \right\} \quad \text{(if } K_{\mathtt{R}} \text{ is trivial, take } X_{K_{\mathtt{R}}} = \emptyset\text{)}, \\ X_{\emptyset} &= \left\{ s_L \;\middle|\; (a^{-1}K)_{s_{\mathtt{L}}} = \emptyset \right\}, \end{aligned}$$

and $a^{-1}K = X_{\mathrm{Tr}_A} \times \mathrm{Tr}_A \cup X_{K_{\mathtt{R}}} \times K_{\mathtt{R}} \cup X_{\emptyset} \times \emptyset$.

First, assume that $K_{\mathtt{R}}$ is trivial. Then $a^{-1}K = X_{\mathrm{Tr}_A} \times \mathrm{Tr}_A$ is a binary profile—either $\emptyset$, or $\mathrm{Tr}_A \times \mathrm{Tr}_A$, or $Z_{\mathtt{L}} \times \mathrm{Tr}_A$, depending on X_{Tr_A}.

Now, assume that $X_{\mathtt{R}}$ is non-trivial and fix $s_{\mathtt{R}} \in K_{\mathtt{R}}$ and $s'_{\mathtt{R}} \in \mathrm{Tr}_A - K_{\mathtt{R}}$. It follows that

$$\begin{aligned} (a^{-1}K)^{s_{\mathtt{R}}} &= X_{\mathrm{Tr}_A} \cup X_{K_{\mathtt{R}}}, \\ (a^{-1}K)^{s'_{\mathtt{R}}} &= X_{\mathrm{Tr}_A}. \end{aligned}$$

Hence, by the initial observation on upper sections, X_{Tr_A} is Tr_A, $K_{\mathtt{L}}$, or $\emptyset$, and similarly for $X_{\mathrm{Tr}_A} \cup X_{K_{\mathtt{R}}}$. We distinguish three cases (see Fig. 5).

(1) If $X_{\mathrm{Tr}_A} = \emptyset$ then $a^{-1}K = K_{\mathtt{R}} \times X_{K_{\mathtt{R}}}$ is a binary profile.
(2) If $X_{\emptyset} = \emptyset$ then $a^{-1}K = \mathrm{Tr}_A \times X_{\mathrm{Tr}_A} \cup K_{\mathtt{R}} \times \mathrm{Tr}_A$ is a binary profile (either trivial or of the form $Z_{\mathtt{R}} \times \mathrm{Tr}_A \;\cup\; \mathrm{Tr}_A \times Z_{\mathtt{L}}$).
(3) Finally, assume that both X_{Tr_A} and $X_{\emptyset}$ are non-empty. In that case both upper sections $(a^{-1}K)^{s_{\mathtt{R}}}$ and $(a^{-1}K)^{s'_{\mathtt{R}}}$ are non-trivial and therefore are both equal to $K_{\mathtt{L}}$. It means that $a^{-1}K = \mathrm{Tr}_A \times \left(X_{\mathrm{Tr}_A} \cup X_{K_{\mathtt{R}}}\right)$ is a binary profile (either trivial or of the form $\mathrm{Tr}_A \times Z_{\mathtt{L}}$).

This concludes the proof. □

Let us now examine the connections between profiles, and states and transitions of game automata. Let $\mathcal{B}$ be a total game automaton and let q_I be a state of $\mathcal{B}$. For a trace w, let $\rho_w = \rho(\mathcal{B}, t_w, q_I)$ be the run over the tree t_w associated with w.

If w is an unlabelled trace, define $p_{\mathcal{B},q_I}(w) = L(\mathcal{B}, q)$, if $\rho_w(w) = q \in Q^{\mathcal{B}}$; if $\rho_w(w) \notin Q^{\mathcal{B}}$, set $p_{\mathcal{B},q_I}(w) = *$.

If w is a labelled trace, set $p_{\mathcal{B},q_I}(w) = *$ if $\rho_w(w) \notin Q^{\mathcal{B}}$; otherwise, let $\rho_w(w) = q$ and $b_w = \delta_B(q, a)$, where a is the last symbol of w, and set $p_{\mathcal{B},q_I}(w) = L(\mathcal{B}, b_w)$ where $L(\mathcal{B}, b)$ is the *profile of the transition* b *in* $\mathcal{B}$, defined as

$$\begin{aligned}
\emptyset \text{ for } b = \bot;\\
\mathrm{Tr}_A \times \mathrm{Tr}_A \text{ for } b = \top;\\
L(\mathcal{B}, q_{\mathtt{L}}) \times \mathrm{Tr}_A \text{ for } b = (q_{\mathtt{L}}, \mathtt{L});\\
\mathrm{Tr}_A \times L(\mathcal{B}, q_{\mathtt{R}}) \text{ for } b = (q_{\mathtt{R}}, \mathtt{R});\\
L(\mathcal{B}, q_{\mathtt{L}}) \times \mathrm{Tr}_A \cup \mathrm{Tr}_A \times L(\mathcal{B}, q_{\mathtt{R}}) \text{ for } b = (q_{\mathtt{L}}, \mathtt{L}) \vee (q_{\mathtt{R}}, \mathtt{R});\\
L(\mathcal{B}, q_{\mathtt{L}}) \times L(\mathcal{B}, q_{\mathtt{R}}) \text{ for } b = (q_{\mathtt{L}}, \mathtt{L}) \wedge (q_{\mathtt{R}}, \mathtt{R}).
\end{aligned}$$

The following is an easy consequence of Fact 3.

LEMMA 7.4. *For each total game automaton* $\mathcal{B}$ *and state* $q_I \in Q^{\mathcal{B}}$*, for each trace* w*,*

$$p_{\mathcal{B},q_I}(w) = p_{L(\mathcal{B},q_I)}(w).$$

PROOF. Let $M = L(\mathcal{B}, q_I)$.

First consider the case of an unlabelled trace w. Let $\rho = \rho(\mathcal{B}, t_w, q_I)$ be the run. If $\rho(w) = *$ then by the definition $p_{\mathcal{B},q_I}(w) = *$. For every realization t of w the position w is not accessible in the game $\mathbf{G}_\rho(\mathcal{B}, t, q_I)$ so $t^{-1}(M)$ is either $\emptyset$ or Tr_A and by the definition w has profile $*$ in M.

Now let $\rho(w) = q \in Q^{\mathcal{B}}$. In that case $p_{\mathcal{B},q_I}(w) = L(\mathcal{B}, q)$. Let t be any realization of w. Observe that either:

(1) Player P has a winning σ strategy in $\mathbf{G}_\rho(\mathcal{B}, t, q_I)$ such that $w \notin \sigma$. Then $t^{-1}(M)$ is either $\emptyset$ or Tr_A depending on P.
(2) Every winning strategy σ of P in $\mathbf{G}_\rho(\mathcal{B}, t, q_I)$ contains w. In that case $t^{-1}(M) = L(\mathcal{B}, q)$ since the following conditions are equivalent:
— a composition $t[w := s]$ belongs to M,
— there exists a winning strategy for $\exists$ in the game $\mathbf{G}_\rho(\mathcal{B}, t[w := s], q_I)$,
— $\exists$ can win $\mathbf{G}_\rho(\mathcal{B}, t[w := s], q_I)$ from w,
— $\exists$ has a winning strategy in the game $\mathbf{G}_\rho(\mathcal{B}, s, q)$,
— $s \in L(\mathcal{B}, q)$.

Recall that there exists a tree t_0 that realizes w and resolves $\mathcal{B}$ from q_I—we plug subtrees in the side holes of t_w accordingly to the states assigned by ρ. By Fact 3 we obtain that $t_0^{-1}M = L(\mathcal{B}, q)$ so t_0 is a witness that w has profile $L(\mathcal{B}, q_I)$.

For the case when w is a labelled trace we use Lemma 7.3—since every unlabelled trace has a profile, we know that every labelled trace also has a profile. It is then easy to verify that the respective equality holds. □

COROLLARY 7.5. *Languages recognized by game automata are locally game.*

Being locally game is necessary but not sufficient to be recognizable by a game automaton.

PROPOSITION 7.6. *There exists a regular tree language* L *such that* L *is locally game but* L *cannot be recognized by a game automaton.*

PROOF. Consider the alphabet $A = \{a, b\}$. Let $t \in \mathrm{Tr}_A$ be a tree. Let us denote $\mathrm{Cut}(a, t)$ as the subtree of t containing those nodes that are accessible by only letters a from the root of t. A total tree $t \in \mathrm{Tr}_A$ is called *thin* if $\mathrm{Cut}(a, t)$ has only countably many infinite branches. Let Thin be the language of all thin trees. This language is regular by the equivalence of the following conditions for each tree $t \in \mathrm{Tr}_A$:

(1) t is not thin,
(2) there exists an embedding of the full binary tree $\{\mathtt{L},\mathtt{R}\}^*$ into $\mathrm{Cut}(a,t)$.

Note that every trace w has a profile Z_w in Thin:

— if w contains a letter b then $Z_w = *$,
— otherwise either w is labelled and therefore $Z_w = \mathrm{Thin} \times \mathrm{Thin}$, or w is unlabelled and $Z_w = \mathrm{Thin}$.

This means that Thin is locally game.

Assume that Thin is recognized by a game automaton $\mathcal{B}$. In that case all transitions of $\mathcal{B}$ have profile $\mathrm{Thin} \times \mathrm{Thin}$ (see Lemma 7.4 in Section 7), so $\mathcal{B}$ is a deterministic automaton. However, a standard argument shows that Thin is not recognizable by any deterministic automaton. □

In what follows, for a given locally game language M we construct a game automaton $\mathcal{G}_M$ that locally computes the profiles and globally reflects the infinitary aspects of M. We show that M is recognized by a game automaton if and only if it is recognized by $\mathcal{G}_M$.

We say that a DFA $\mathcal{A} = \langle A, Q, q_I, \delta, F\rangle$ computes a *partial* function $f\colon A^* \rightharpoonup X$ if $\mathcal{A}$ recognizes $\mathrm{dom}(f)$ and it comes equipped with a function $\tau^{\mathcal{A}}\colon F \to \mathrm{rg}(f)$, such that $\tau^{\mathcal{A}}(\delta(q_I, w)) = f(w)$ for each $w \in \mathrm{dom}(f)$, where $\delta(q, v)$ is the state of $\mathcal{A}$ after reading word v from state q.

The following lemma shows that for each regular tree language M one can effectively construct a DFA $\mathcal{A}$ that computes (a finite representation of) the profile in M of given trace w. In particular, it is decidable whether M is locally game, and the set $\mathrm{Profiles}_M$ of all possible profiles of traces in M is finite and can be computed from M.

LEMMA 7.7. *Let M be a regular tree language over an alphabet A. There exists a finite automaton that reads a word w over $A \cup \{\mathtt{L},\mathtt{R}\}$ and outputs:*

— $\mathrm{NotTrace}$ *if w is not a trace;*
— $\mathrm{NoProfile}$ *if w is a trace but w has no profile in M; and*
— *a finite representation of $p_M(w)$ if w is a trace and has a profile in M.*

A proof could easily be obtained by the composition method [Shelah 1975]. However, to make the paper self-contained, we give a direct reasoning. The crucial observation is that if a tree t' is put in a hole of a tree t, then the only thing that matters for the acceptance of t is *the type of* t'. For the sake of this proof let us fix a regular tree language M recognized by a non-deterministic tree automaton $\mathcal{B}$ from an initial state $q_I \in Q$.

The type of a total tree $t \in \mathrm{Tr}_A$ is defined as follows:

$$\mathrm{tp}(t) = \{q \in Q :\ t \in L(\mathcal{B}, q)\} \subseteq Q.$$

The set of types of all total trees is finite and effective, we denote it by $\mathrm{Tp} \subseteq \mathcal{P}(Q)$. For a set $T \subseteq \mathrm{Tp}$, by $L(T)$ we denote the language of all total trees t such that $\mathrm{tp}(t) \in T$.

FACT 8. *Let $t_{\mathtt{L}}, t_{\mathtt{R}}, t'_{\mathtt{L}}, t'_{\mathtt{R}} \in \mathrm{Tr}_A$, let q be a state of $\mathcal{B}$, and t be a tree with two holes. If*

$$(\mathrm{tp}(t_{\mathtt{L}}), \mathrm{tp}(t_{\mathtt{R}})) = (\mathrm{tp}(t'_{\mathtt{L}}), \mathrm{tp}(t'_{\mathtt{R}}))$$

then

$$t(t_{\mathtt{L}}, t_{\mathtt{R}}) \in L(\mathcal{B}, q) \iff t(t'_{\mathtt{L}}, t'_{\mathtt{R}}) \in L(\mathcal{B}, q).$$

In particular, the type $\mathrm{tp}\,(t(t_{\mathtt{L}}, t_{\mathtt{R}}))$ does not depend on the choice of representatives $t_{\mathtt{L}}, t_{\mathtt{R}}$.

By the fact above, we can write $t(\tau_{\mathtt{L}}, \tau_{\mathtt{R}})$ for the type of $t(t_{\mathtt{L}}, t_{\mathtt{R}})$ for any $t_{\mathtt{L}}, t_{\mathtt{R}}$ with $(\mathrm{tp}(t_{\mathtt{L}}), \mathrm{tp}(t_{\mathtt{R}})) = (\tau_{\mathtt{L}}, \tau_{\mathtt{R}})$.

Our aim is to construct a finite automaton $\mathcal{A}$ that reads a finite word $w \in (A \cup \{\mathtt{L}, \mathtt{R}\})^*$, checks that w is a trace, and computes a representation of $p_M(w)$, provided that w has profile.

First let us fix

$$\begin{aligned} Q_1 &= \mathcal{P}(\mathrm{Tp}), \\ Q_2 &= \left\{ S \subseteq \mathrm{Tp}^2 \;\middle|\; L(S) \text{ is a profile} \right\} \cup \{\emptyset, \mathrm{Tp}^2\}, \\ Q_E &= \{\mathrm{NotTrace}, \mathrm{NoProfile}\}, \\ Q^{\mathcal{A}} &= Q_1 \cup Q_2 \cup Q_E, \\ q_I^{\mathcal{A}} &= \{T \subseteq \mathrm{Tp} : q_I \in T\} \in Q_1. \end{aligned}$$

Our aim is to define the transition function $\delta^{\mathcal{A}}$ in such a way that Lemma 7.8, given below, is satisfied. First, for every $T \in Q_1$, $S \in Q_2$, $U \in Q^{\mathcal{A}}$, $a \in A$, $d \in \{\mathtt{L}, \mathtt{R}\}$, and $l \in A \cup \{\mathtt{L}, \mathtt{R}\}$ we put

— $\delta^{\mathcal{A}}(T, d) = \mathrm{NotTrace}$,
— $\delta^{\mathcal{A}}(S, a) = \mathrm{NotTrace}$,
— $\delta^{\mathcal{A}}(U, l) = U$.

Second, assume that the current state is $T \in Q_1$ and a letter a is given. We define the successive state $\delta^{\mathcal{A}}(T, a) \in Q_2 \cup \{\mathrm{NoProfile}\}$. Let us define the following set of pairs of types

$$S = \{(\tau_{\mathtt{L}}, \tau_{\mathtt{R}}) : a(\tau_{\mathtt{L}}, \tau_{\mathtt{R}}) \in T\}. \tag{4}$$

Note that, given S, we can decide if $L(S)$ is a profile and, if it is, we define $\delta^{\mathcal{A}}(T, a) = S$. Otherwise we put $\delta^{\mathcal{A}}(T, a) = \mathrm{NoProfile}$.

Third, assume that the current state is $S \in Q_2$ and a direction d is given. We define the successive state $\delta^{\mathcal{A}}(S, d) \in Q_1$. By the symmetry assume that $d = \mathtt{L}$. Consider the following cases:

— if $S = T_{\mathtt{L}} \times \mathrm{Tp} \cup \mathrm{Tp} \times T_{\mathtt{R}}$ for some $T_{\mathtt{L}}, T_{\mathtt{R}} \subseteq \mathrm{Tp}$ then $\delta^{\mathcal{A}}(S, d) = T_{\mathtt{L}}$,
— otherwise, $\delta^{\mathcal{A}}(S, d) = \pi_1(S)$—the projection of S onto the first coordinate.

In the case $d = \mathtt{R}$ we consider $T_{\mathtt{R}}$ instead of $T_{\mathtt{L}}$ and the projection onto the second coordinate of S.

LEMMA 7.8. *Let w be a word and U be the state of $\mathcal{A}$ after reading w. The following conditions hold*

(i) if $U \in (Q_1 \cup Q_2) - \{\emptyset, \mathrm{Tp}, \mathrm{Tp}^2\}$ then w is a trace and $L(U)$ is the profile of w in M,
(ii) if $U \in \{\emptyset, \mathrm{Tp}, \mathrm{Tp}^2\}$ then w is a trace and has profile $*$ in M,
(iii) if $U = \mathrm{NotTrace}$ then w is not a trace,
(iv) if $U = \mathrm{NoProfile}$ then w is a trace but has no profile in M.

PROOF. The first three items follow easily from the definition of profile.

What remains is to show that if $U = \mathrm{NoProfile}$ then the trace w has no profile in M.

Assume the contrary and consider a minimal counterexample. Notice that, by definition, such minimal counterexample is a trace of the form wa for some letter a. Assume that wa has profile Z in M. Let T be the state of $\mathcal{A}$ after reading w and let S be the set computed according to equation (4). If $T \in \{\emptyset, \mathrm{Tp}\}$ then w has profile $*$ in M by Item (ii). Then wa has also profile $*$ in M and the state of $\mathcal{A}$ after reading wa belongs

to $\{\emptyset, \mathrm{Tp}^2\}$. Assume that $T \notin \{\emptyset, \mathrm{Tp}\}$. By Item (i) we obtain that w has profile $L(T)$ in M.

Let t_0 be a realization of w such that $t_0^{-1}(M) = L(T)$. Then, by the definition of t^{-1}, we obtain that

$$(t_0[w := a])^{-1}(M) = a^{-1}(L(T)) = Z. \tag{5}$$

It is enough to show that $L(S) = Z$ and thus $S \in Q_2$ and $S \neq \mathrm{NoProfile}$.

To see this, note that the following conditions are all equivalent for a pair of trees $t_{\mathtt{L}}, t_{\mathtt{R}} \in \mathrm{Tr}_A$:

(a) $(t_{\mathtt{L}}, t_{\mathtt{R}}) \in Z$,
(b) $a(t_{\mathtt{L}}, t_{\mathtt{R}}) \in L(T)$,
(c) $a(\mathrm{tp}(t_{\mathtt{L}}), \mathrm{tp}(t_{\mathtt{R}})) \in T$;
(d) $(\mathrm{tp}(t_{\mathtt{L}}), \mathrm{tp}(t_{\mathtt{R}})) \in S$,
(e) $t_{\mathtt{L}}, t_{\mathtt{R}} \in L(S)$.

Indeed, the equivalence between conditions (a) and (b) is given by equation (5), the equivalence between conditions (b) and (c) follows by the definition of $a(\tau_{\mathtt{L}}, \tau_{\mathtt{R}})$, the equivalence between conditions (c) and (d) is a consequence of equation (4), and finally the equivalence between conditions (d) and (e) is by the definition of $L(S)$. □

The infinitary aspects of M are captured by the notion of *correct* infinite traces. An infinite trace is an infinite word π over $A \cup \{\mathtt{L}, \mathtt{R}\}$ with letters from A on even positions and directions from $\{\mathtt{L}, \mathtt{R}\}$ on odd positions. Just like a finite trace, π can be seen as a tree t_π consisting of a single infinite branch which has only side holes. A tree t realizes π if it is obtained by plugging total trees in the side holes of t_π.

Assume that M is locally game and let $p_M(w)$ be the profile of w in M. We say that t *resolves* M *up to* π if t realizes π and for each labelled trace w that is a prefix of π, if wL is a prefix of π then

— $t\restriction_{w\mathtt{R}} \notin Z_{\mathtt{R}}$ if $p_M(w) = (Z_{\mathtt{L}} \times \mathrm{Tr}_A) \cup (\mathrm{Tr}_A \times Z_{\mathtt{R}})$,
— $t\restriction_{w\mathtt{R}} \in Z_{\mathtt{R}}$ if $p_M(w) = Z_{\mathtt{L}} \times Z_{\mathtt{R}}$,

and symmetrically if $w\mathtt{R}$ is a prefix of π. An infinite trace π is M*-correct* if some tree $t \in M$ resolves M up to π.

There is an automata-theoretic counterpart of the notion of M-correct infinite traces. Consider a game automaton $\mathcal{C}$, an initial state q_I, and an infinite trace π. Notice that π corresponds to a play of the game $\rho = \rho(\mathcal{C}, t_\pi, q_I)$ associated with $\mathcal{C}$. We say that $\mathcal{C}$ *accepts* π *from* q_I if either $\rho(v) = *$ for some $v \in \mathrm{dom}(t_\pi)$ and $\rho(w) \neq \bot$ for all $w \in \mathrm{dom}(t_\pi)$; or $\exists$ wins the play corresponding to π in ρ.

LEMMA 7.9. *A game automaton $\mathcal{C}$ accepts π from q_I if and only if π is $L(\mathcal{C}, q_I)$-correct.*

PROOF. Let use denote $L(\mathcal{C}, q_I)$ as M'. Let $\pi \in (A \times \{\mathtt{L}, \mathtt{R}\})^\omega$ be an infinite trace. First assume that π is M'-correct. Let $t' \in M'$ be a tree witnessing it. Recall, that t' is obtained by putting some total trees in the holes of t_π. Since $t' \in M'$ so there exists a winning strategy σ for $\exists$ in $\mathbf{G}_\rho(\mathcal{C}, t', q_I^D)$. Since, whenever $\exists$ could make a choice to leave the branch π, the respective subtree in t' is losing for her. Let $\rho = \rho(\mathcal{C}, t', q_I^D)$. If there exists $v \in \mathrm{dom}(t_\pi)$ such that $\rho(v) = *$, then there cannot be any $w \prec \mathrm{dom}(t_\pi)$ with $\rho(w) = \bot$ (otherwise the strategy σ would not be winning). In the opposite case the whole branch corresponding to π must be contained in the strategy σ. In both cases $\mathcal{C}$ accepts π from $q_I^{\mathcal{C}}$.

Now assume that $\mathcal{C}$ accepts π from $q_I^{\mathcal{C}}$. Let t' be some realization of π. Consider σ to be the strategy of $\exists$ in $\mathbf{G}_\rho(\mathcal{C}, t', q_I^{\mathcal{C}})$ defined as follows:

— in all the nodes along π follow this branch,
— whenever $\forall$ selects to go off the branch π, use some winning strategy in the respective subtree (it exists by the construction).

By the definition, σ is a winning strategy for $\exists$ in the game $\mathbf{G}_\rho(\mathcal{C}, t', q_I^{\mathcal{C}})$. Therefore, $t' \in M'$ so t' is a witness that π is an M'-correct branch. □

LEMMA 7.10. *For each regular tree language M one can effectively construct a deterministic parity automaton $\mathcal{D}$ recognizing the set of M-correct infinite traces.*

PROOF. Let $\mathcal{B}$ be a non-deterministic automaton recognizing the given regular tree language M from a state g_I. We show how to express in monadic second-order logic over ω the fact, that a given ω-word π is an M-correct branch. By the results of Büchi and McNaughton such a formula φ can be effectively translated into a deterministic parity ω-word automaton $\mathcal{D}$.

As in the proof of Lemma 7.4, we make use of compositional tools. By $\mathrm{Tp} \subseteq \mathcal{P}(Q^{\mathcal{B}})$ we denote the set of all types of total trees with respect to the automaton $\mathcal{B}$.

Intuitively, the formula φ guesses the types of the total subtrees that need to be put in the side holes of t_π to obtain a tree t that resolves M up to π. Basing on these guessed types φ can verify that $t \in M$.

Recall that an infinite trace π is defined as a word in the language $(A \cdot \{\mathtt{L}, \mathtt{R}\})^\omega$. A witness for the existence of t will be an infinite word over the alphabet $A \cup (\{\mathtt{L}, \mathtt{R}\} \times \mathrm{Tp})$ denoted $\widehat{\pi}$ and called *enrichment of π by types*. We require that every even position of $\widehat{\pi}$ belongs to A and every odd position belongs to $\{\mathtt{L}, \mathtt{R}\} \times \mathrm{Tp}$.

$$\widehat{\pi} = a_0 \cdot (d_0, \tau_0) \cdot a_1 \cdot (d_1, \tau_1) \cdot \ldots$$

If $w = a_0 \cdot d_0 \cdot a_1 \cdot d_1 \cdot \ldots \cdot d_{n-1} \cdot a_n$ is a labelled trace that is a prefix of π we say that d_n is the *final direction of* w and τ_n is the *final type of* w.

Let the formula $\psi_{\mathtt{R}}$ express that for every trace $w \prec \pi$ with final direction d, final type τ, and such that S is the state of $\mathcal{A}$ (see Lemma 7.4) after reading w, the following conditions are satisfied:

— if $S = S_{\mathtt{L}} \times \mathrm{Tp} \cup \mathrm{Tp} \times S_{\mathtt{R}}$ then $\tau \notin S_{\bar{d}}$,
— otherwise $\tau \in \pi_i(S)$ where $i = 1, 2$ for $d = \mathtt{L}, \mathtt{R}$ respectively.

Note that every $\widehat{\pi}$ that is an enrichment of π by types induces a total tree $t_{\widehat{\pi}}$ over the alphabet A where in the n-th side hole of π we put some total tree of type τ_n. Note also that $\widehat{\pi} \models \psi_{\mathtt{R}}$ if and only if $t_{\widehat{\pi}}$ resolves M up to π. In particular the exact subtrees we put into side holes of t_π are irrelevant, we only need to take care of their types.

What remains is to express in MSO logic on $\widehat{\pi}$ that $t_{\widehat{\pi}} \in M$. For this we say that there exists an infinite word ρ coding a run the non-deterministic automaton $\mathcal{B}$ on t_π. Formally a *word coding a run* ρ is defined as a word over the alphabet $Q^B \cup \{*\} \cup D^B$ where D^B is the set of all deterministic transitions appearing in the transitions of $\mathcal{B}$. The elements of $Q^B \cup \{*\}$ are supposed to appear on even positions of ρ and elements of D^B are supposed to appear on odd positions of ρ:

$$\rho = q_0 \cdot b_0 \cdot q_1 \cdot b_1 \cdot \ldots$$

Let the formula ψ_M express the following facts about the combination of words $\widehat{\pi} \otimes \rho$ in the language

$$\big[A \times \big(Q^B \cup \{*\}\big) \cdot \big((\{\mathtt{L}, \mathtt{R}\} \times \mathrm{Tp}) \times D^B\big)\big]^\omega$$

— the state q_0 equals q_I,
— for every n the transition b_n is one of the deterministic transitions appearing in $\delta^B(q_n, a_n)$,

— for every n the state assigned to $\bar{d_n}$ by b_n (if any) belongs to τ_n,
— for every n the state assigned to d_n by b_n (if any) equals q_{n+1}, if there is no such state then $q_{n+1} = *$,
— either from some point on $b_n = *$ or the parity condition is satisfied by the sequence of states $q_0, q_1, \ldots$.

Note, that $\widehat{\pi} \otimes \rho \models \psi_M$ if and only if ρ encodes an accepting run of $\mathcal{B}$ on t_π that assigns to the n'th hole of t_π a state belonging to τ_n. Therefore, $\widehat{\pi} \otimes \rho \models \psi_M$ if and only if the run encoded by ρ can be extended to an accepting run of $\mathcal{B}$ on $t_{\widehat{\pi}}$.

Let φ express for a given infinite trace π that there exists an enrichment of π by types τ_n and an encoding of run ρ such that $\widehat{\pi} \models \psi_{\mathtt{R}}$ and $\widehat{\pi} \otimes \rho \models \psi_M$. Note that $\pi \models \varphi$ if and only if there exists a tree $t = t_{\widehat{\pi}} \in M$ that realizes π and that resolves M up to π. □

We define $\mathcal{G}_M$ as a product of $\mathcal{A}$ and $\mathcal{D}$, with priorities inherited from $\mathcal{D}$ and the types of transitions ($\vee$, $\wedge$, etc.) determined by the type of profile computed by $\mathcal{A}$. More precisely, for $a \in A$, $(p, q) \in Q^{\mathcal{A}} \times Q^{\mathcal{D}}$, $\tau = \tau^{\mathcal{A}}(\delta^{\mathcal{A}}(p, a))$, define $\delta\big((p, q), a\big)$ as

$$\begin{array}{rl} \top & \text{if} \quad \tau \in \{*, \mathrm{Tr}_A \times \mathrm{Tr}_A\}; \\ \bot & \text{if} \quad \tau = \emptyset; \\ \beta(p, q, a, \mathtt{L}) & \text{if} \quad \tau = Z_{\mathtt{L}} \times \mathrm{Tr}_A; \\ \beta(p, q, a, \mathtt{R}) & \text{if} \quad \tau = \mathrm{Tr}_A \times Z_{\mathtt{R}}; \\ \beta(p, q, a, \mathtt{L}) \vee \beta(p, q, a, \mathtt{R}) & \text{if} \quad \tau = Z_{\mathtt{L}} \times \mathrm{Tr}_A \cup \mathrm{Tr}_A \times Z_{\mathtt{R}}; \\ \beta(p, q, a, \mathtt{L}) \wedge \beta(p, q, a, \mathtt{R}) & \text{if} \quad \tau = Z_{\mathtt{L}} \times Z_{\mathtt{R}}; \end{array}$$

where $\beta(p, q, a, d)$ is defined as $\big((\delta^{\mathcal{A}}(p, ad), \delta^{\mathcal{D}}(q, ad)), d\big)$. Let $q_M = (q_I^{\mathcal{A}}, q_I^{\mathcal{D}})$.

THEOREM 7.11. *A regular language M is recognized by a game automaton iff M is locally game and $L(\mathcal{G}_M, q_M) = M$.*

PROOF. Assume that $M = L(\mathcal{B}, q_I^{\mathcal{B}})$ for some game automaton $\mathcal{B}$ and $q_I^{\mathcal{B}} \in Q^{\mathcal{B}}$. By Corollary 7.5, M is locally game. Fix $t \in \mathrm{Tr}_A$ and let $\rho_M = \rho(\mathcal{G}_M, t, q_M)$ and $\rho_{\mathcal{B}} = \rho(\mathcal{B}, t, q_I^{\mathcal{B}})$. By Lemma 7.4, $p_M(w)$ determines the profiles of the corresponding transitions in $\rho_{\mathcal{B}}$ and ρ_M. Hence, the games associated to these runs are isomorphic if the priorities are ignored. Let π be an infinite trace in t. By the construction, $\mathcal{G}_M$ accepts π from q_M iff π is M-correct. By Lemma 7.9, π is M-correct iff $\mathcal{B}$ accepts π from $q_I^{\mathcal{B}}$. It follows that $\rho_{\mathcal{B}}$ is accepting iff ρ_M is accepting. □

As an immediate corollary we obtain the following.

THEOREM 7.12. *Given an alternating automaton $\mathcal{A}$ and a state q_I, it is decidable whether $L(\mathcal{A}, q_I)$ is recognized by a game automaton. If so, some game automaton recognizing $L(\mathcal{A}, q_I)$ can be effectively constructed from $\mathcal{A}$ and q_I.*

REFERENCES

André Arnold. 1999. The mu-calculus alternation-depth hierarchy is strict on binary trees. *ITA* 33, 4/5 (1999), 329–340.

André Arnold and Damian Niwiński. 2001. *Rudiments of mu-calculus*. Elsevier.

André Arnold and Damian Niwiński. 2007. Continuous Separation of Game Languages. *Fundamenta Informaticae* 81, 1-3 (2007), 19–28.

André Arnold and Luigi Santocanale. 2005. Ambiguous classes in μ-calculi hierarchies. *TCS* 333, 1–2 (2005), 265–296.

Mikołaj Bojańczyk and Thomas Place. 2012. Regular Languages of Infinite Trees That Are Boolean Combinations of Open Sets. In *ICALP*. 104–115.

Julian Bradfield. 1998. Simplifying the modal mu-calculus alternation hierarchy. In *STACS*. 39–49.

Julius Richard Büchi. 1962. On a Decision Method in Restricted Second-Order Arithmetic. In *Proc. 1960 Int. Congr. for Logic, Methodology and Philosophy of Science*. 1–11.

Thomas Colcombet, Denis Kuperberg, Christof Löding, and Michael Vanden Boom. 2013. Deciding the weak definability of Büchi definable tree languages. In *CSL*. 215–230.

Thomas Colcombet and Christof Löding. 2008. The Non-deterministic Mostowski Hierarchy and Distance-Parity Automata. In *ICALP (2)*. 398–409.

Jacques Duparc, Alessandro Facchini, and Filip Murlak. 2011. Definable Operations On Weakly Recognizable Sets of Trees. In *FSTTCS*. 363–374.

Jacques Duparc and Filip Murlak. 2007. On the Topological Complexity of Weakly Recognizable Tree Languages. In *Fundamentals of Computation Theory, 16th International Symposium, FCT 2007, Budapest, Hungary, August 27-30, 2007, Proceedings*. 261–273.

Allen Emerson and Charanjit Jutla. 1991. Tree Automata, mu-Calculus and Determinacy. In *FOCS'91*. 368–377.

Alessandro Facchini, Filip Murlak, and Michał Skrzypczak. 2013. Rabin-Mostowski Index Problem: A Step beyond Deterministic Automata. In *LICS*. 499–508.

Alessandro Facchini, Filip Murlak, and Michał Skrzypczak. 2015. On the weak index problem for game automata. In *WoLLIC*. 93–108.

Alexander Kechris. 1995. *Classical descriptive set theory*. Springer-Verlag, New York.

Denis Kuperberg. 2012. *Etude de classes de fonctions de coût régulières*. Ph.D. Dissertation. Université Paris Diderot.

Ralf Küsters and Thomas Wilke. 2002. Deciding the First Level of the μ-Calculus Alternation Hierarchy. In *FSTTCS*. 241–252.

Andrzej W. Mostowski. 1984. Regular expressions for infinite trees and a standard form of automata. In *Symposium on Computation Theory*. 157–168.

Andrzej W. Mostowski. 1991a. *Games with forbidden positions*. Technical Report. University of Gdańsk.

Andrzej W. Mostowski. 1991b. Hierarchies of Weak Automata and Weak Monadic Formulas. *Theor. Comput. Sci.* 83, 2 (1991), 323–335.

David E. Muller, Ahmed Saoudi, and Paul E. Schupp. 1986. Alternating Automata. The Weak Monadic Theory of the Tree, and its Complexity. In *ICALP (Lecture Notes in Computer Science)*, Laurent Kott (Ed.), Vol. 226. 275–283.

Filip Murlak. 2005. On Deciding Topological Classes of Deterministic Tree Languages. In *CSL 2005*. 428–441.

Filip Murlak. 2008a. *Effective topological hierarchies of recognizable tree languages*. Ph.D. Dissertation. University of Warsaw.

Filip Murlak. 2008b. Weak index versus Borel rank. In *STACS 2008 (LIPIcs)*, Vol. 1. 573–584.

Damian Niwiński and Igor Walukiewicz. 1998. Relating Hierarchies of Word and Tree Automata. In *STACS*. 320–331.

Damian Niwiński and Igor Walukiewicz. 2003. A gap property of deterministic tree languages. *Theor. Comput. Sci.* 1, 303 (2003), 215–231.

Damian Niwiński and Igor Walukiewicz. 2005. Deciding Nondeterministic Hierarchy of Deterministic Tree Automata. *Electr. Notes Theor. Comput. Sci.* 123 (2005), 195–208.

Martin Otto. 1999. Eliminating Recursion in the μ-Calculus. In *STACS*. 531–540.

Michael O. Rabin. 1969. Decidability of second-order theories and automata on infinite trees. *Trans. of the American Math. Soc.* 141 (1969), 1–35.

Michael O. Rabin. 1970. Weakly definable relations and special automata. In *Proceedings of the Symposium on Mathematical Logic and Foundations of Set Theory*. North-Holland, 1–23.

Saharon Shelah. 1975. The Monadic Theory of Order. *The Annals of Mathematics* 102, 3 (1975), 379–419.

Jerzy Skurczyński. 1993. The Borel hierarchy is infinite in the class of regular sets of trees. *Theoretical Computer Science* 112, 2 (1993), 413–418.

Tomasz Fryderyk Urbański. 2000. On Deciding if Deterministic Rabin Language Is in Büchi Class. In *ICALP*. 663–674.

Michael Vanden Boom. 2012. *Weak Cost Automata over Infinite Trees*. Ph.D. Dissertation. University of Oxford.

William Wadge. 1983. *Reducibility and determinateness in the Baire space*. Ph.D. Dissertation. University of California, Berkeley.

Igor Walukiewicz. 2002. Deciding low levels of tree-automata hierarchy. *Electr. Notes Theor. Comput. Sci.* 67 (2002), 61–75.

Received June 2015; revised XXX; accepted XXX