

Deterministic and game separability for regular languages of infinite trees

Lorenzo Clemente Michał Skrzypczak

May 5, 2021

Abstract

We show that it is decidable whether two regular languages of infinite trees are separable by a deterministic language, resp., a game language. We consider two variants of separability, depending on whether the set of priorities of the separator is fixed, or not. In each case, we show that separability can be decided in EXPTIME, and that separating automata of exponential size suffice. We obtain our results by reducing to infinite duration games with ω -regular winning conditions and applying the finite-memory determinacy theorem of Büchi and Landweber.

1 Introduction

One of the most intriguing and motivating problems in the field of automata theory is the *membership problem*. For two fixed classes of languages \mathcal{C} (*input class*) and \mathcal{D} (*output class*), the $(\mathcal{C}, \mathcal{D})$ -membership problem asks, given a representation of a language in \mathcal{C} , whether this language belongs to \mathcal{D} . Among the first results of this type is the famous theorem by Schützenberger [44] and McNaughton-Papert [32], characterising, among all regular languages of finite words, the subclass of languages that can be defined in first-order logic.

In this paper we consider the class \mathcal{C} of regular languages of infinite trees. While there are many equivalent automata models for this class—e.g., Muller, Rabin, and Street automata [27]—*parity automata* are without doubt the most established such model [24]. The most important descriptonal complexity measure of a parity automaton is the set of priorities $C \subseteq \mathbb{N}$ it is allowed to use, which is called its *index*. Not only a larger index allows the automaton to recognise more languages [34], but the computational complexity of known procedures for the emptiness problem crucially depends on the index (the current best bound is quasi-polynomial [8]). The most famous open problem in the area of regular languages of infinite trees is the *nondeterministic index membership problem*, which is the $(\mathcal{C}, \mathcal{D})$ -membership problem for \mathcal{D} the class of languages recognised by some nondeterministic parity automaton of a fixed index C (c.f. [17]). In many cases, the solution of the membership problem relies either on algebraic representations or determinisation, however algebraic structures for regular languages of infinite trees are of limited availability (c.f. [2]) and deterministic automata do not capture all regular languages. While on infinite words this problem was essentially solved by Wagner already at the end of the '70s [47], its solution for infinite trees seems still far away.

Known decidability results abound if we restrict either the input class \mathcal{C} or the output class \mathcal{D} . Results of the first kind are known for \mathcal{C} being the class of deterministic [37] and, more generally, game automata [25, Theorem 1.2]. Results of the second kind (i.e., when the input class \mathcal{C} is the full class of regular languages) exist for the output class \mathcal{D} being the lower levels of the index hierarchy [31, 48] and of the Borel hierarchy [4], the class of deterministic languages [35], and Boolean combinations of open sets [6]. Other variants of the index membership problem are known to be decidable, including the early result of Urbański showing that it is decidable whether a given deterministic parity tree automaton is equivalent to some nondeterministic Büchi one [46], the weak alternating index problems for the class of deterministic automata [33] and Büchi automata [16, 45], and deciding whether a given parity automaton is equivalent to some nondeterministic co-Büchi automaton [16].

Another problem closely related to membership is separability. The $(\mathcal{C}, \mathcal{D})$ -*separability* problem asks, given a pair of languages L, M in \mathcal{C} , whether there exists a language S in \mathcal{D} (called a *separator*) s.t. $L \subseteq S$ and¹ $S \perp M$. Intuitively, a separator S provides a certificate of disjointness, yielding information on the structure of L, M up to some chosen granularity. The separability problem is a generalisation of the membership problem if the class \mathcal{C} is closed under complement, since we can always take M to be the complement of L , in which case the only candidate for the separator is L itself. There are many elegant results in computer science, formal logic, and mathematics showing that separators always exist. Instances include Lusin’s separation theorem in topology (two disjoint analytic sets are always separable by a Borel set; c.f. [28, Theorem 14.7]), a folklore result in computability theory (two disjoint co-recursively enumerable sets are separable by a recursive set), Craig’s theorems in logic (jointly contradictory first-order formulas can be separated by a formula containing only symbols in the shared vocabulary [18]) and model theory (two disjoint projective classes are separable by an elementary class [18]); in formal language theory, a generalisation of a theorem suggested by Tarski and proved by Rabin [41, Theorem 29] states that two disjoint Büchi languages of infinite trees are separable by a weak language (c.f. [42]).

In this work we study the $(\mathcal{C}, \mathcal{D})$ -separability problems where \mathcal{C} is the full class of regular languages of infinite trees, and \mathcal{D} is one of four kinds of subclasses thereof, depending on whether the automaton is deterministic or game, and depending on whether we fix a finite index $C \subseteq \mathbb{N}$ or we leave it unrestricted $C = \mathbb{N}$. Our main result is that all four kinds of the separability problems above are decidable and in EXPTIME. Moreover, we show that if a separator exists, then there is one of exponential size.

Theorem 1.1. *The deterministic and game separability problems can be solved in EXPTIME, both for a fixed finite index $C \subseteq \mathbb{N}$, and an unrestricted one $C = \mathbb{N}$. Moreover, separators with exponentially many states and polynomially many priorities suffice.*

Our work is permeated by the observation that the separability problem for two languages L, M can be phrased in terms of a game of infinite duration with an ω -regular winning condition. In such a *separability game* there are two players, Separator trying to prove that L, M are separable, and Input with the

¹We write $S \perp M$ for $S \cap M = \emptyset$.

opposite objective. In the simple case of $(\mathcal{C}, \mathcal{D})$ -separability where \mathcal{C} is the class of regular languages of ω -words and \mathcal{D} the subclass induced by deterministic parity automata of finite index C , the i -th round of the game is as follows:

- Separator plays a priority $c_i \in C$.
- Input plays a letter a_i from the finite alphabet Σ .

The resulting infinite play $(c_0, a_0)(c_1, a_1) \cdots$ is won by Separator if 1) $a_0 a_1 \cdots \in L$ implies $c_0 c_1 \cdots$ is accepting and 2) $a_0 a_1 \cdots \notin L$ implies $c_0 c_1 \cdots$ is rejecting. Since the winning condition is ω -regular, by the result of Büchi and Landweber [7] we can decide who wins the game and moreover finite-memory strategies for Separator suffice. Thanks to a correspondence between such strategies and deterministic separators, Separator wins such a game iff there exists a deterministic automaton with priorities in C separating L, M . This provides both decidability of the separability problem and an upper-bound on the size of separators. We design analogous games with ω -regular winning conditions for the more involved case of infinite trees for the separability problems mentioned above and apply [7].

The separability problems we consider have been open so far and generalise the corresponding membership problems. A solution for deterministic separability can easily be derived from [36], however our techniques based on games are novel and provide a unified view on all problems. When instantiated to the specific case of membership, our decidability results generalise the deterministic case (for both fixed and unconstrained index) [36, 35] and the game membership case for unconstrained index [25, Theorem 7.12]. We believe the game approach is much more direct than the combinatorial and pattern-based techniques used in the previous solutions, c.f. [25, Section 7, pp. 29–37]. The game membership problem for a fixed index C has been open so far.

We are not aware of computation complexity results for separability problems over regular languages of infinite trees, neither of an analysis of the size of separators. Regarding deterministic membership, EXPTIME-completeness is known [36, Corollary 11], as well as EXPTIME upper [35, end of page 12] and lower bounds [48, Theorem 4.1] (c.f., also [31]) for computing the optimal deterministic index. Devising non-trivial complexity lower bounds for the separability problem is left for future work, as well as extending our approach to other classes of separators.

Related works. Over finite words, variants of the $(\mathcal{C}, \mathcal{D})$ -separability problem have been studied for classes \mathcal{C} both more general than the regular languages, such as the context free languages [22, 49] and higher-order languages [14] (later extended to safe schemes over finite trees [1]), and for classes \mathcal{D} more restrictive than the regular languages, such as in [39, 40]. The separability and membership problems have also been studied for several classes of infinite-state systems, such as vector addition systems [11, 10, 23], well-structured transition systems [21], one-counter automata [20], and timed automata [13, 12]. Recent developments on efficient algorithms solving parity games are based on the ability to find a simple separator, yielding both upper bounds on the problem, and lower bounds for a wide family of algorithms [5, 19, Chapter 3]. Finally, it is worth mentioning that games have already been successfully used to provide several characterisation results, such as in [17, 16, 15, 3, 45, 9].

Outline. In Section 2 we introduce automata and other mathematical preliminaries. In Sections 3 to 6 we present the game-theoretic characterisations of the separability problems we consider. We believe this is the most interesting aspect of this work. Section 7 is devoted to an analysis of the computational complexity of our decision methods leading to the proof of the announced Theorem 1.1.

2 Preliminaries

A nonempty finite set Σ of *letters* $a \in \Sigma$ is called an *alphabet*. A (Σ -labelled) *tree* is a function $t: \{\mathbf{L}, \mathbf{R}\}^* \rightarrow \Sigma$ assigning to each *node* $u \in \{\mathbf{L}, \mathbf{R}\}^*$ a *label* $t(u) \in \Sigma$. The *root* of a tree is denoted ϵ . The set of all Σ -labelled trees is denoted Tr_Σ . The symbols \mathbf{L}, \mathbf{R} are called *directions* and a *branch* is an infinite sequence thereof $d_0 d_1 \cdots \in \{\mathbf{L}, \mathbf{R}\}^\omega$. A tree t is uniquely defined by the set of its *paths* $\text{Path}(t) = \{(a_0, d_0)(a_1, d_1) \cdots \in (\Sigma \times \{\mathbf{L}, \mathbf{R}\})^\omega \mid \forall i. a_i = t(d_0 d_1 \cdots d_{i-1})\}$, which is extended to languages pointwise as $\text{Path}(L) = \{\text{Path}(t) \mid t \in L\}$.

2.1 Automata

Fix a nonempty finite set of *priorities* $C \subseteq \mathbb{N}$. A (*top-down, nondeterministic, parity, tree*) *automaton* is a tuple $\mathcal{A} = (\Sigma, Q, q_0, \Omega, \Delta)$, where Σ is a finite alphabet, Q is a finite set of *states*, amongst which $q_0 \in Q$ is the *initial state*, $\Omega: Q \rightarrow C$ assigns a priority to every state, and $\Delta \subseteq Q \times \Sigma \times Q \times Q$ is a set of *transitions*. The priority function Ω is extended to a transition $\delta = (q, _, _, _)$ as $\Omega(\delta) := \Omega(q)$, pointwise to an infinite sequence of states $\Omega(q_0 q_1 \cdots) := \Omega(q_0) \Omega(q_1) \cdots \in C^\omega$ and transitions $\Omega(\delta_0 \delta_1 \cdots) = \Omega(\delta_0) \Omega(\delta_1) \cdots \in C^\omega$. An infinite sequence of priorities $c_0 c_1 \cdots \in C^\omega$ is *accepting* if the maximal priority occurring infinitely often is even. Similarly, an infinite sequence of states $\rho = q_0 q_1 \cdots \in Q^\omega$ or of transitions $\rho = \delta_0 \delta_1 \cdots \in \Delta^\omega$ is *accepting* whenever $\Omega(\rho)$ is accepting. We write $\Delta(q, a) = \{(q, a, q_L, q_R) \in \Delta\}$ for the set of transitions from a state $q \in Q$ over a letter $a \in \Sigma$, and $\Delta(a) = \bigcup \{\Delta(q, a) \mid q \in Q\}$ for all transitions over a . We extend the notation above to an infinite path $b = (a_0, d_0)(a_1, d_1) \cdots \in (\Sigma \times \{\mathbf{L}, \mathbf{R}\})^\omega$ by writing $\Delta(b)$ for the set of infinite sequences of transitions $\vec{\delta} = \delta_0 \delta_1 \cdots \in \Delta^\omega$ of the form $\delta_i = (q_i, a_i, q_{L,i}, q_{R,i})$ for every i , which are *conform to* b in the sense that q_0 is the initial state of the automaton and $q_{i+1} = q_{d_i, i}$.

A *run* of an automaton \mathcal{A} as above over a tree $t \in \text{Tr}_\Sigma$ is a Q -labelled tree $\rho \in \text{Tr}_Q$ s.t. $\rho(\epsilon) = q_0$ is the initial state and for every node in the tree $u \in \{\mathbf{L}, \mathbf{R}\}^*$ the quadruple $(\rho(u), t(u), \rho(u\mathbf{L}), \rho(u\mathbf{R}))$ belongs to Δ . Such a run is *accepting* if for every branch $d_0 d_1 \cdots \in \{\mathbf{L}, \mathbf{R}\}^\omega$ the sequence of states $(\rho(d_0 \cdots d_{i-1}))_{i \in \omega}$ is accepting. The set of all trees $t \in \text{Tr}_\Sigma$ s.t. \mathcal{A} has an accepting run over t is denoted $L(\mathcal{A})$ and is called the *language* recognised by \mathcal{A} . The corresponding *path language* is $L^{\text{path}}(\mathcal{A}) := \text{Path}(L(\mathcal{A})) \subseteq (\Sigma \times \{\mathbf{L}, \mathbf{R}\})^\omega$. If $q \in Q$ is a state of an automaton \mathcal{A} then by \mathcal{A}_q we denote the same automaton as \mathcal{A} but with the initial state q_0 changed to q . Thus, $L(\mathcal{A}_q)$ is the set of trees over which \mathcal{A} has an accepting run ρ starting at $\rho(\epsilon) = q$. In the rest of the paper we assume that all states q in an automaton are *productive* in the sense that $L(\mathcal{A}_q) \neq \emptyset$.

2.2 Deterministic and game automata

We say that \mathcal{A} is a *game automaton* if, for every $q \in Q$ and $a \in \Sigma$, either we have a *conjunctive transition* $\Delta(q, a) = \{(q, a, q_L, q_R)\}$ or two *disjunctive transitions* $\Delta(q, a) = \{(q, a, q_L, \top), (q, a, \top, q_R)\}$ (c.f. [25, Definition 3.2]), where $\top \neq q_0$ represents a distinguished state in Q accepting every tree (i.e., $L(\mathcal{A}_\top) = \text{Tr}_\Sigma$) and $q_L, q_R \neq \top$. An automaton \mathcal{A} is *deterministic* if it is a game automaton with only conjunctive transitions and in this case for every tree $t \in \text{Tr}_\Sigma$ there exists a unique run ρ of \mathcal{A} over t . A tree language L is *deterministic*, resp., *game*, if it can be recognised by some deterministic, resp., game automaton. Game automata can be complemented with very low complexity by just increasing every priority by one and by swapping conjunctive and disjunctive transitions.

Lemma 2.1. *If \mathcal{A} is a game parity tree automaton, then $\text{Tr}_\Sigma \setminus L(\mathcal{A})$ can be recognised by a game parity tree automaton with the same number of states and priorities.*

Proof. Let $\mathcal{A} = (\Sigma, Q, q_0, \Omega, \Delta)$ be a game automaton. Its complement is the game automaton \mathcal{A}^c obtained by swapping conjunctive transitions with disjunctive ones, and vice versa, and by increasing priorities by one. Formally, $\mathcal{A}^c = (\Sigma, Q, q_0, \Omega^c, \Delta^c)$ where $\Omega^c(q) = \Omega(q) + 1$ and the set of transitions Δ^c is obtained by dualising Δ as follows: for every $q \in Q$ and $a \in \Sigma$, if $\Delta(q, a) = \{(q, a, q_L, q_R)\}$ is conjunctive then $\Delta^c(q, a) = \{(q, a, q_L, \top), (q, a, \top, q_R)\}$ is disjunctive, and symmetrically in the other case. It is standard to check that $L(\mathcal{A}^c) = \text{Tr}_\Sigma \setminus L(\mathcal{A})$. \square

2.3 Determinisation over ω -words

A *nondeterministic ω -word parity automaton* is a tuple $\mathcal{A} = (\Sigma, Q, q_0, \Omega, \Delta)$ where Σ is a finite input alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, $\Omega: Q \rightarrow C$ assigns to each state a priority in C , and $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation. The notions of runs and the accepted language $L(\mathcal{A}) \subseteq \Sigma^\omega$ are standard [27]. We recall that nondeterministic ω -word parity automata can be determinised with an exponential complexity in the number of states and a polynomial complexity in the number of priorities. We will use this fact in later proofs.

Lemma 2.2 (c.f. [43]). *A nondeterministic ω -word parity automaton \mathcal{A} with n states and k priorities can be converted to an equivalent deterministic parity automaton with $n' = 2 \cdot (n \cdot (k+1))^{n \cdot (k+1)} \cdot (n \cdot (k+1))!$ states and $k' = 2 \cdot n \cdot (k+1)$ priorities.*

In order to prove Lemma 2.2, we first prove the following result allowing us to convert nondeterministic parity to nondeterministic Büchi automata.

Lemma 2.3. *A nondeterministic ω -word parity automaton \mathcal{A} with n states and k priorities can be converted to an equivalent nondeterministic ω -word Büchi automaton with $n \cdot (k+1)$ states.*

Proof. Let \mathcal{A} have $n = |P|$ states and $k = |C|$ priorities. Automaton \mathcal{B} has states of the form $Q = P \cup P \times C$. In the first phase \mathcal{B} just simulates \mathcal{A} , until it goes to a state of the form (p, c) by nondeterministically guessing an even priority $c \in C$ and checking that c is visited infinitely often and no larger priority is visited in the rest of the run. \square

Lemma 2.2 follows from Lemma 2.3 and the following result allowing us to convert from nondeterministic Büchi to deterministic parity automata.

Lemma 2.4 ([38, Theorem 3.10]). *A nondeterministic ω -word Büchi automaton \mathcal{A} with n states can be converted to an equivalent deterministic ω -word parity automaton \mathcal{B} with $2 \cdot n^n \cdot n!$ states and $2 \cdot n$ priorities.*

2.4 Games

In this section we formalise the framework of games used throughout the paper. These are variants of two-player zero-sum perfect information games on graphs of infinite duration where some intermediate positions are hidden. The default names of the two players are PI and PII, however in most games it will be more convenient to work with some more meaningful names. If $P \in \{\text{PI}, \text{PII}\}$ is a player then the other player is called the *opponent* of P . To specify a game we need to define an arena and a winning condition. An *arena* of a game consists of: a nonempty set V of *positions*, an *initial position* $v_0 \in V$, a *position update function* η , a finite sequence $((P^{(0)}, X^{(0)}), \dots, (P^{(n)}, X^{(n)}))$ of possible *decisions* that players can make during a round, and *restrictions* O_v , one for each position $v \in V$. Each *decision* $(P^{(k)}, X^{(k)})$ is left in the hands of one of the players $P^{(k)} \in \{\text{PI}, \text{PII}\}$ and is taken from some fixed nonempty finite set $X^{(k)}$ of *possible choices*. The product of all the sets of possible choices $O := X_v^{(0)} \times \dots \times X_v^{(n)}$ is called the set of *round outcomes*. The *position update function* is of the type $\eta: V \times O \rightarrow V$. An arena can additionally *restrict* some decisions of some players depending on the current position $v \in V$ of the game and some previous decisions in this round using the *restrictions*, i.e., nonempty subsets $O_v \subseteq O$, indexed by the positions $v \in V$.

At the i -th round starting in a position $v_i \in V$ players declare their choices $x_i^{(k)} \in X^{(k)}$ in the order specified by the arena and according to the imposed restrictions. More formally, they inductively define a vector $o_i = (x_i^{(0)}, \dots, x_i^{(n)})$, keeping the invariant that

$$(x_i^{(0)}, \dots, x_i^{(k-1)}) \in \text{Proj}_{0, \dots, k-1}(O_v), \quad (1)$$

i.e., the constructed vector belongs to the projection of O_v onto the coordinates $0, \dots, k-1$. This guarantees that the successive player $P^{(k)}$ has always at least one choice $x^{(k)} \in X^{(k)}$ satisfying the invariant. Once the whole vector $o_i = (x_i^{(0)}, \dots, x_i^{(n)})$ is constructed, the i -th round is finished. The next position of the game is defined by $v_{i+1} := \eta(v_i, o_i)$. A play of the game is the sequence of round outcomes $o_0 o_1 \dots \in O^\omega$ (the visited positions are implicit, however can easily be computed using η).

A *winning condition* of a game specifies which infinite plays $o_0 o_1 \dots \in O^\omega$ are considered winning for one of the players P ; with the remaining plays losing for P and winning for the opponent of P . Formally, a winning condition is just a subset of O^ω .

Some of the considered games are *positionless*, i.e., there is only a single position $V = \{v_0\}$. An arena is called *finite* if V is finite.

A *strategy* of a player P for a game is a tuple $\mathcal{M} = (M, \ell_0, \bar{v}, \tau)$ where: M is a set of *memory states*, $\ell_0 \in M$ is an *initial memory state*, $\bar{o} = (\bar{x}, \dots, \bar{z})$ is a vector of *decision functions*, one function for each decision of the player P ,

and τ is a *memory update function* that maps a position of the game $v \in V$, a memory state $\ell \in M$, and a vector of choices of the opponent (x', \dots, z') into the next memory value $\ell' = \tau(v, \ell, x', \dots, z') \in M$. The domain of a decision function \bar{y} for a decision $(P, X^{(k)})$ allowing the player P to choose $x^{(k)} \in X^{(k)}$ is the product of V , M , and all the possible previous sets of possible options of the opponent in a round. The range of \bar{y} is Y , with the restriction that the player P needs to preserve the invariant as in (1).

A strategy σ is of *finite memory* if M is a finite set. Notice that each finite memory strategy for a finite game is a finite object that can be effectively represented. A strategy is *positional* if $M = \{\ell_0\}$. In the case of a positional strategy there is a unique memory value, the function τ is trivial, and we ignore the M argument of the decision functions. Similarly, if the given arena is positionless then we ignore the V argument of the functions above.

Fix a strategy \mathcal{M} of a player P . Such a strategy determines the way in which the player P should make her choices. Consider the i -th round of the game, starting in a position $v_i \in V$ and with a memory state $\ell_i \in M$ (for $i = 0$ the memory state ℓ_0 is the initial memory value). The consecutive choices of P in this round are given by the decision functions in \bar{v} applied to v , ℓ_i , and the previous choices of the opponent. Once the round is finished with opponent's choices (x'_i, \dots, z'_i) , we take $\ell_{i+1} := \tau(v_i, \ell_i, x'_i, \dots, z'_i)$.

A play $o_0 o_1 \dots \in O^\omega$ that is obtained according to the policy above is said to be *conform* to the strategy \mathcal{M} . Notice that if \mathcal{M} and \mathcal{M}' are two strategies of the two players then there exists a unique play $o_0 o_1 \dots \in O^\omega$ that is conform to both of them—this play can be defined inductively in the standard way.

A strategy \mathcal{M} of a player P is said to be *winning* if all the plays conform to \mathcal{M} are winning for P . Because of the observation above, at most one of the players has a winning strategy. We say that a position $v \in V$ is *winning* for P if P has a winning strategy in the game with the initial position v_0 set to v . We say that a game is *determined* if exactly one player has a winning strategy.

Games on graphs. To formally prove the results of finite-memory determinacy of the games involved in our work, we show how to reduce them to the standard framework of *games on graphs*.

An arena of a game on graph is specified by a directed graph (V, E) without dead-ends (the elements of V are called *positions*), an *initial position* $v_0 \in V$, an *ownership partition* $V = V_{\text{PI}} \cup V_{\text{PII}}$ into two disjoint sets, and a labelling function $\rho: E \rightarrow \Sigma$. A *play* of such a game is constructed inductively by the players, starting from the initial position v_0 . At the i -th round, with the current position $v_i \in V_P$, the player P chooses an edge $(v_i, v_{i+1}) \in E$, defining the consecutive position v_{i+1} .

A winning condition of PI in such a game is a language $W \subseteq \Sigma^\omega$. A play as above is winning for PI if the ω -word $\rho(v_0, v_1)\rho(v_1, v_2)\dots$ belongs to L .

We will now show how to reduce a game defined according to our definition into a game on graph. Consider a game with an arena consisting of a set of positions V ; an initial position $v_0 \in V$; a position update function η ; decisions $((P^{(0)}, X^{(0)}), \dots, (P^{(n-1)}, X^{(n-1)}))$; and restrictions $(O_v)_{v \in V}$. Let O be the set of round outcomes. Consider a graph with the set of positions

$$V' := \bigcup_{v \in V} \left(\{v\} \times \bigcup_{k=0, \dots, n} \text{Proj}_{0, \dots, k-1}(O_v) \right),$$

where $\text{Proj}_{0,\dots,-1}(O_v)$ is the singleton $\{()\}$ consisting of the empty tuple $()$. The initial position is $v'_0 := (v_0, ())$.

Let the set of edges $E' := E'_0 \cup E'_1$ consists of the following two types of edges. The first is defined as

$$E'_0 := \left((v, (x^{(0)}, \dots, x^{(k-1)})), (v, (x^{(0)}, \dots, x^{(k-1)}, x^{(k)})) \right),$$

where $c \in V$ and $(x^{(0)}, \dots, x^{(k-1)}, x^{(k)}) \in \text{Proj}_{0,\dots,k}(O_v)$. The second is defined as

$$E'_1 := ((v, o), (v', ())),$$

where $v \in V$, $o \in O_v$, and $\eta(v, o) = v'$. Let $\Sigma = O \cup \{\epsilon\}$ and the labelling ρ be defined as $\rho(e) := \epsilon$ for $e \in E'_0$; and $\rho((v, o), (v', ())) := o$ for $((v, o), (v', ())) \in E'_1$.

Given a winning condition $W \subseteq O^\omega$, we define the new winning condition $W' \subseteq \Sigma^\omega$ by skipping the symbols ϵ (notice that the shape of the arena ensures that every n -th edge is labelled by an element of O).

Claim 2.5. *Each strategy (understood in the standard sense) of a player P in the corresponding game on graph can be translated into a strategy of the shape $\mathcal{M} = (M, \ell_0, \bar{v}, \tau)$ in the original game. Moreover, this translation preserves the size of the memory and maps a winning strategy into a winning strategy.*

Determinacy. We rely on two important known results of determinacy of the considered games, obtained directly from the known results via Claim 2.5.

Theorem 2.6 ([7, Theorem 1']). *Consider a game arena with the set of round outcomes O . Assume that the set of winning plays of PI is an ω -regular language over the alphabet O . Then one of the players has a finite memory winning strategy in this game. Moreover, such a strategy can be effectively computed based on a representation of a finite arena and the winning condition.*

We say that a winning condition $W \subseteq O^\omega$ is a *Rabin* condition over O if

$$W = (\text{inf}(E_0) \cap \text{fin}(F_0)) \cup \dots \cup (\text{inf}(E_n) \cap \text{fin}(F_n))$$

s.t. for $k = 0, \dots, n$ we have $E_k, F_k \subseteq O$ and

$$\begin{aligned} \text{inf}(E_k) &:= \{(o_i)_{i \in \omega} \in O^\omega \mid o_i \in E_k \text{ for infinitely many } i \in \omega\} \\ \text{fin}(F_k) &:= \{(o_i)_{i \in \omega} \in O^\omega \mid o_i \in F_k \text{ for only finitely many } i \in \omega\}. \end{aligned}$$

Notice that the family of Rabin conditions is closed under union.

If $C \subseteq \mathbb{N}$ is a finite set of priorities then the set of parity accepting sequences $(c_i)_{i \in \omega} \subseteq C^\omega$ can be written as a Rabin condition with $E_k = \{n \in C \mid n \geq 2k\}$ and $F_k = \{n \in C \mid n \geq 2n+1\}$ for $0 \leq k \leq \frac{\max C}{2}$. Therefore, the parity condition is a special case of Rabin condition. Similarly, the complement of a parity condition is also a Rabin condition.

Theorem 2.7 ([29, Lemma 9]; c.f. also [26, Theorem 4] and [30, Theorem 7.12]). *Consider a game arena with the set of round outcomes O . Assume that the set of winning plays of PI is a Rabin condition over O . Then PI has a uniform positional strategy \mathcal{M} in that game, i.e., a positional strategy such that for every position $v \in V$, if v is winning for PI then \mathcal{M} is a winning strategy from v .*

Complexity. Finally, we recall that games on graphs with parity winning conditions can be solved in quasi-polynomial time.

Lemma 2.8 ([8, Theorem 2.9]). *A parity game with n positions and $k = |C|$ priorities can be solved in deterministic time $O(n^{\log k+6})$.*

This will be used in our complexity analysis in Section 7. In fact, already a naïve parity game algorithm with complexity $O(n^k)$ (i.e., polynomial in n and exponential in k) would suffice for our purposes since we will instantiate it on game graphs of exponential size and polynomially many priorities.

2.5 Acceptance games

We present a game-theoretic view on accepting runs for automata based on the framework from Section 2.4. This will serve both as an example of the kind of games that we consider throughout paper, and as a technical tool in the proofs from Sections 5 and 6.

Let $t \in \text{Tr}_\Sigma$ be a tree. The *acceptance game* $G^{\text{acc}}(\mathcal{A}, t)$ is played in rounds by two players, Automaton and Pathfinder. The goal of Automaton is to show that $t \in L(\mathcal{A})$; Pathfinder has the complementary objective $t \notin L(\mathcal{A})$.

Acceptance game $G^{\text{acc}}(\mathcal{A}, t)$

At the i -th round starting at a *position* $v_i = (u_i, q_i) \in V := \{\text{L}, \text{R}\}^* \times Q$:

[A: δ] Automaton plays a transition $\delta_i = (q_i, t(u_i), q_{\text{L},i}, q_{\text{R},i}) \in \Delta(q_i, t(u_i))$.

[P: d] Pathfinder plays a direction $d_i \in \{\text{L}, \text{R}\}$.

The next position is $v_{i+1} := (u_i d_i, q_{d_i,i})$.

The initial position is $v_0 := (\epsilon, q_0)$. Automaton *wins* the resulting infinite play $\pi = (\delta_0, d_0)(\delta_1, d_1) \cdots$ if the sequence of transitions $\delta_0 \delta_1 \cdots$ is accepting.

The following proposition is folklore.

Proposition 2.9. *Let $t \in \text{Tr}_\Sigma$ and \mathcal{A} be an automaton over the alphabet Σ . Automaton wins the acceptance game $G^{\text{acc}}(\mathcal{A}, t)$ if, and only if, $t \in L(\mathcal{A})$.*

2.6 Disjointness games

Let \mathcal{A} and \mathcal{B} be two nondeterministic automata. We recall a standard game used to characterise whether $L(\mathcal{A}) \perp L(\mathcal{B})$. This will be crucial in the correctness proofs throughout Sections 3 to 6. The *disjointness game* $G^{\text{dis}}(\mathcal{A}, \mathcal{B})$ is played by two players, Automaton and Pathfinder. Automaton's aim is to incrementally build a tree accepted by both \mathcal{A} and \mathcal{B} , witnessing $L(\mathcal{A}) \cap L(\mathcal{B}) \neq \emptyset$, while Pathfinder has the opposite objective.² The set of positions of the game is $Q^{\mathcal{A}} \times Q^{\mathcal{B}}$, and the initial position is $(q_0^{\mathcal{A}}, q_0^{\mathcal{B}})$.

²The disjointness game could equivalently be phrased as a nonemptiness game for the product automaton $\mathcal{A} \times \mathcal{B}$ recognising $L(\mathcal{A}) \cap L(\mathcal{B})$. However, in our technical development it will be more direct to use the disjointness game.

Disjointness game $G^{\text{dis}}(\mathcal{A}, \mathcal{B})$

At the i -th round starting at a position $(q_i^{\mathcal{A}}, q_i^{\mathcal{B}})$:

[A: a] Automaton plays a letter $a_i \in \Sigma$.

[A: $\delta^{\mathcal{A}}$] Automaton plays a transition $\delta_i^{\mathcal{A}} = (q_i^{\mathcal{A}}, a_i, q_{\text{L},i}^{\mathcal{A}}, q_{\text{R},i}^{\mathcal{A}}) \in \Delta^{\mathcal{A}}(q_i^{\mathcal{A}}, a_i)$.

[A: $\delta^{\mathcal{B}}$] Automaton plays a transition $\delta_i^{\mathcal{B}} = (q_i^{\mathcal{B}}, a_i, q_{\text{L},i}^{\mathcal{B}}, q_{\text{R},i}^{\mathcal{B}}) \in \Delta^{\mathcal{B}}(q_i^{\mathcal{B}}, a_i)$.

[P: d] Pathfinder plays a direction $d_i \in \{\text{L}, \text{R}\}$.

The next position is $(q_{d_i,i}^{\mathcal{A}}, q_{d_i,i}^{\mathcal{B}})$.

Let the resulting infinite play be $\pi = (a_0, \delta_0^{\mathcal{A}}, \delta_0^{\mathcal{B}}, d_0)(a_1, \delta_1^{\mathcal{A}}, \delta_1^{\mathcal{B}}, d_1) \cdots$. Such a play induces an infinite path $b = (a_0, d_0)(a_1, d_1) \cdots$ and two sequences of transitions $\vec{\delta}^{\mathcal{A}} := \delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \cdots$ and $\vec{\delta}^{\mathcal{B}} := \delta_0^{\mathcal{B}} \delta_1^{\mathcal{B}} \cdots$. The rules of the game guarantee that $\vec{\delta}^{\mathcal{A}} \in \Delta^{\mathcal{A}}(b)$ and $\vec{\delta}^{\mathcal{B}} \in \Delta^{\mathcal{B}}(b)$. Automaton wins the play π if both sequences $\vec{\delta}^{\mathcal{A}}$ and $\vec{\delta}^{\mathcal{B}}$ are accepting.

In the rest of the paper it will be more useful to consider Pathfinder's point of view. Since her winning condition can be presented as Rabin condition (see Section 2.4), whenever she wins, she has a memoryless (i.e., $M = \{\ell_0\}$) winning strategy. Such a memoryless strategy for Pathfinder in the disjointness game can be represented by a function $\mathcal{P}: (\bigcup_{a \in \Sigma} \Delta^{\mathcal{A}}(a) \times \Delta^{\mathcal{B}}(a)) \rightarrow \{\text{L}, \text{R}\}$, which we call a *pathfinder*.

Lemma 2.10. *If $\text{L}(\mathcal{A}) \perp \text{L}(\mathcal{B})$ then there is a pathfinder \mathcal{P} which is winning for Pathfinder in the disjointness game $G^{\text{dis}}(\mathcal{A}, \mathcal{B})$.*

Proof. The winning condition for Pathfinder is a disjunction of two properties: Either $\vec{\delta}^{\mathcal{A}}$ or $\vec{\delta}^{\mathcal{B}}$ is rejecting. Both these properties are complements of parity conditions. Therefore, the winning condition of Pathfinder is a Rabin condition. Since memoryless winning strategies suffice for games with Rabin winning conditions, it follows that if Pathfinder wins then she has a memoryless winning strategy.

In general, such a positional strategy is of the form $(\{\ell_0\}, \ell_0, \vec{d}, \tau)$, with τ constantly equal ℓ_0 and $\vec{d}: (q^{\mathcal{A}}, q^{\mathcal{B}}, a, \delta^{\mathcal{A}}, \delta^{\mathcal{B}}) \mapsto d$, for $q^{\mathcal{A}} \in Q^{\mathcal{A}}$, $q^{\mathcal{B}} \in Q^{\mathcal{B}}$, $a \in \Sigma$, $\delta^{\mathcal{A}} \in \Delta^{\mathcal{A}}(q^{\mathcal{A}}, a)$, $\delta^{\mathcal{B}} \in \Delta^{\mathcal{B}}(q^{\mathcal{B}}, a)$, and $d \in \{\text{L}, \text{R}\}$. Due to the redundancy within the arguments of the decision function \vec{d} , we can represent such a strategy by a function $\mathcal{P}: (\bigcup_{a \in \Sigma} \Delta^{\mathcal{A}}(a) \times \Delta^{\mathcal{B}}(a)) \rightarrow \{\text{L}, \text{R}\}$, i.e., a pathfinder. \square

Corollary 2.11 below follows directly from the construction of \mathcal{P} and the fact that the strategy \mathcal{M} used to obtain it is winning.

Corollary 2.11. *Assume that $\text{L}(\mathcal{A}) \perp \text{L}(\mathcal{B})$ and let \mathcal{P} be a pathfinder as above. Let $b = (a_0, d_0)(a_1, d_1) \cdots \in (\Sigma \times \{\text{L}, \text{R}\})^\omega$ be a path and $\vec{\delta}^{\mathcal{A}} = \delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \cdots \in \Delta^{\mathcal{A}}(b)$, $\vec{\delta}^{\mathcal{B}} = \delta_0^{\mathcal{B}} \delta_1^{\mathcal{B}} \cdots \in \Delta^{\mathcal{B}}(b)$ be two sequences of transitions of these automata that are conform to b . If for every $i \in \omega$ we have $\mathcal{P}(\delta_i^{\mathcal{A}}, \delta_i^{\mathcal{B}}) = d_i$ then at least one of the sequences $\vec{\delta}^{\mathcal{A}}$ and $\vec{\delta}^{\mathcal{B}}$ is rejecting.*

The construction from Lemma 2.10 above has a specific property when one of the involved automata (e.g., \mathcal{A}) is a game automaton. Since we assume that

every state is productive, positions of the form $(\top, q^{\mathcal{B}})$ are losing for **Pathfinder** in $G^{\text{dis}}(\mathcal{A}, \mathcal{B})$. Therefore, without loss of generality we can assume that the pathfinder \mathcal{P} satisfies the following observation.

Remark 2.12. *Consider a transition $\delta^{\mathcal{A}} = (q^{\mathcal{A}}, a, q_{\top}^{\mathcal{A}}, \top)$ (resp., $\delta^{\mathcal{A}} = (q^{\mathcal{A}}, a, \top, q_{\text{R}}^{\mathcal{A}})$) in a game automaton \mathcal{A} . Then, $\mathcal{P}(\delta^{\mathcal{A}}, _)$ is constantly equal to L (resp., R).*

2.7 Deterministic separability over ω -words

In this section, we provide full details for the separability problem for ω -words sketched in the introduction. This will also serve as an introduction to the more challenging separability problems over infinite trees considered in the rest of the paper. Let \mathcal{A}, \mathcal{B} be two nondeterministic parity automata over ω -words and let $C \subseteq \mathbb{N}$ be a set of priorities. Consider the following C -deterministic-separability game $G_{\omega}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$.

C -deterministic-separability game over ω -words $G_{\omega}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$

At the i -th round:

[S: c] Separator plays a priority $c_i \in C$.

[I: a] Input plays a letter $a_i \in \Sigma$.

Separator wins an infinite play $\pi = (c_0, a_0)(c_1, a_1) \cdots \in (C \times \Sigma)^{\omega}$ if the following two conditions are both satisfied (i.e., if $\pi \in W := \mathbf{W}_{\mathcal{A}} \cap \mathbf{W}_{\mathcal{B}} \subseteq (C \times \Sigma)^{\omega}$):

- $\pi \in \mathbf{W}_{\mathcal{A}}$: If $a_0 a_1 \cdots \in L(\mathcal{A})$, then $c_0 c_1 \cdots$ is accepting.
- $\pi \in \mathbf{W}_{\mathcal{B}}$: If $a_0 a_1 \cdots \in L(\mathcal{B})$, then $c_0 c_1 \cdots$ is rejecting.

Lemma 2.13. *Separator wins $G_{\omega}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$ if, and only if, $L(\mathcal{A}), L(\mathcal{B})$ can be separated by a deterministic parity automaton with priorities in C .*

A strategy for **Separator** in $G_{\omega}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$ is of the form $\mathcal{M} = (M, \ell_0, \bar{c}: M \rightarrow C, \tau: M \times \Sigma \rightarrow M)$. The rest of this section is devoted to the proof of this lemma.

Soundness. Assume that **Separator** wins $G_{\omega}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$, let \mathcal{M} be his finite-memory winning strategy as above. Consider a candidate separating automaton $\mathcal{S} := (\Sigma, M, \ell_0, \Omega, \Delta)$ having the same set of states M as \mathcal{M} 's memory states, and the same initial state ℓ_0 , where state ℓ 's priority $\Omega(\ell) := \bar{c}(\ell)$ is provided directly by the decision function \bar{c} , and the set of transitions is defined according to the memory update function τ as

$$\Delta = \{(\ell, a, \tau(\ell, a)) \mid \ell \in M, a \in \Sigma\}.$$

Clearly \mathcal{S} is a C -deterministic automaton over ω -words. Moreover, $\mathbf{W}_{\mathcal{A}}$ guarantees that $L(\mathcal{A}) \subseteq L(\mathcal{S})$, while $\mathbf{W}_{\mathcal{B}}$ guarantees that $L(\mathcal{B}) \perp L(\mathcal{S})$. Therefore, \mathcal{S} is the required separator. \square

Completeness. Let $\mathcal{S} = (\Sigma, Q, q_0, \Omega, \Delta)$ be a separating automaton. We build a finite-state winning strategy for **Separator** $\mathcal{M} = (Q, q_0, \bar{c}, \tau)$ over the same set of states Q s.t. $\bar{c}(q) = \Omega(q)$ is q 's priority in \mathcal{S} , and $\tau(q, a) = q'$ for the unique q' s.t. $(q, a, q') \in \Delta$. It is immediate to show that \mathcal{M} is winning from the fact that \mathcal{S} is a separator. \square

3 Separability by deterministic automata with priorities in C

In this section we present a game-theoretic characterisation of separability over infinite trees by deterministic automata with a fixed finite set of priorities $C \subseteq \mathbb{N}$. Let \mathcal{A}, \mathcal{B} be two nondeterministic automata over infinite trees. We extend the game over ω -words from the introduction (and formally defined in Section 2.7) with two additional actions: a *selector* for Separator and a direction for Input.

C -deterministic-separability game $G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$

At the i -th round:

[S: c] Separator plays a priority $c_i \in C$.

[I: a] Input plays a letter $a_i \in \Sigma$.

[S: f] Separator plays a *selector* $f_i \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{B}}(a_i)}$.

[I: d] Input plays a direction $d_i \in \{\mathbf{L}, \mathbf{R}\}$.

Intuitively, a selector encodes a direction for each (relevant) transition of \mathcal{B} and this is used for the correctness of the separator. (In Section 3.1 we consider a simpler variant without selectors and we discuss which separability problem it captures.) Let the resulting infinite play be $\pi = (c_0, a_0, f_0, d_0)(c_1, a_1, f_1, d_1) \cdots$, with the induced infinite path $b := (a_0, d_0)(a_1, d_1) \cdots$. Separator wins the play π if the following two conditions are satisfied:

- $\pi \in \mathbf{W}_{\mathcal{A}}$: If there exists an accepting sequence of transitions $\vec{\delta}^{\mathcal{A}} = \delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \cdots \in \Delta^{\mathcal{A}}(b)$, then $c_0 c_1 \cdots$ is accepting.
- $\pi \in \mathbf{W}_{\mathcal{B}}$: If there exists an accepting sequence of transitions $\vec{\delta}^{\mathcal{B}} = \delta_0^{\mathcal{B}} \delta_1^{\mathcal{B}} \cdots \in \Delta^{\mathcal{B}}(b)$ s.t. for every $i \in \omega$ we have $f_i(\delta_i^{\mathcal{B}}) = d_i$, then $c_0 c_1 \cdots$ is rejecting.

The following lemma states that the separability game correctly characterises the deterministic separability problem.

Lemma 3.1. *Separator wins $G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$ if, and only if, $L(\mathcal{A}), L(\mathcal{B})$ can be separated by a deterministic parity tree automaton with priorities in C .*

We present a full proof in order to show the rôle of Separator's selectors.

Soundness. Assume that Separator wins the separability game $G := G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$ by a finite-memory winning strategy $\mathcal{M} = (M, \ell_0, (\vec{c}, \vec{f}), \tau)$. Strategy \mathcal{M} has two decision functions: \vec{c} assigns to each $\ell \in M$ a priority $\vec{c}(\ell) \in C$, and \vec{f} assigns to each $\ell \in M$ and $a \in \Sigma$ a selector $\vec{f}(\ell, a) \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{B}}(a)}$. Moreover, the type of the memory update function is $\tau: M \times \Sigma \times \{\mathbf{L}, \mathbf{R}\} \rightarrow M$. Consider a deterministic parity tree automaton $\mathcal{S} := (\Sigma, M, \ell_0, \Omega^{\mathcal{S}}, \Delta^{\mathcal{S}})$ which has the same set of states M and initial state ℓ_0 as \mathcal{M} , priorities are induced by the decision function \vec{c} of \mathcal{M} as $\Omega^{\mathcal{S}}(\ell) := \vec{c}(\ell)$, and transitions are of the form $\Delta^{\mathcal{S}} = \{(\ell, a, \tau(\ell, a, \mathbf{L}), \tau(\ell, a, \mathbf{R})) \mid \ell \in M, a \in \Sigma\}$.

We show that \mathcal{S} separates $L(\mathcal{A}), L(\mathcal{B})$. We first show $L(\mathcal{A}) \subseteq L(\mathcal{S})$. Let $t \in L(\mathcal{A})$ be a tree that is accepted by the automaton \mathcal{A} , as witnessed by

an accepting run ρ^A . Let ρ^S be the unique run of \mathcal{S} over t . Consider any branch $d_0 d_1 \cdots \in \{\mathbf{L}, \mathbf{R}\}^\omega$. We need to show that the sequence of priorities $(\Omega^S(\rho^S(d_0 \cdots d_{i-1})))_{i \in \omega}$ is accepting. Consider a play π of G where at the i -th round **Separator** plays according to the strategy \mathcal{M} with current memory state $\ell_i \in M$ and **Input** plays according to the letters from t and directions $d_0 d_1 \cdots$ fixed above:

- [S: c] **Separator** plays the priority $c_i := \bar{c}(\ell_i) \in C$.
- [I: a] **Input** plays the letter $a_i := t(u_i) \in \Sigma$, where $u_i := d_0 \cdots d_{i-1}$.
- [S: f] **Separator** plays the selector $f_i := \bar{f}(\ell_i, a_i) \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^B(a_i)}$ (the selector is irrelevant in this part of the proof).
- [I: d] **Input** plays the direction $d_i \in \{\mathbf{L}, \mathbf{R}\}$ as fixed above.

The next memory state is $\ell_{i+1} := \tau(\ell_i, a_i, d_i)$. Let the resulting infinite play be $\pi = (c_0, a_0, f_0, d_0)(c_1, a_1, f_1, d_1) \cdots$. By the construction of \mathcal{S} we know that $\ell_i = \rho^S(u_i)$ and therefore $c_i = \Omega^S(\rho^S(u_i))$. Since $t \in L(\mathcal{A})$, there exists an accepting sequence of transitions $\bar{\delta}^A = \delta_0^A \delta_1^A \cdots \in \Delta^A(b)$ along the path $b = (a_0, d_0)(a_1, d_1) \cdots$. Since **Separator** is winning, $\pi \in \mathbf{W}_{\mathcal{A}}$ and thus the sequence $c_0 c_1 \cdots$ is accepting, as required.

We now argue that $L(\mathcal{S})$ and $L(\mathcal{B})$ are disjoint. Towards reaching a contradiction, assume that $t \in L(\mathcal{S}) \cap L(\mathcal{B})$ belongs to their intersection. Let ρ^S be the unique run of \mathcal{S} over t , and let ρ^B be an accepting run of \mathcal{B} over t . Consider a play $\pi = (c_0, a_0, f_0, d_0)(c_1, a_1, f_1, d_1) \cdots$ of G where the i -th round is played as above except that **Input** plays the direction $d_i := f_i(\delta_i^B)$, obtained by applying the selector f_i to the transition $\delta_i^B := (\rho^B(u_i), t(u_i), \rho^B(u_i \mathbf{L}), \rho^B(u_i \mathbf{R}))$ determined according to the run ρ^B . By the choice of directions d_i 's, the sequence of transitions $\bar{\delta}^B = \delta_0^B \delta_1^B \cdots \in (\Delta^B)^\omega$ satisfies $f_i(\delta_i^B) = d_i$ for every $i \in \omega$. Since the run ρ^B is accepting, $\bar{\delta}^B$ is accepting. Since **Separator** is winning, $\pi \in \mathbf{W}_{\mathcal{B}}$ and thus the sequence of priorities $c_0 c_1 \cdots$ is rejecting. However, this is a contradiction, because for each $i \in \omega$ we have $\ell_i = \rho^S(u_i)$ and $c_i = \Omega^S(\ell_i)$ and we assumed that the run ρ^S is accepting. \square

Completeness. Assume that $\mathcal{S} = (\Sigma, Q^S, q_0^S, \Delta^S, \Omega^S)$ is a deterministic automaton with priorities in C separating $L(\mathcal{A})$, $L(\mathcal{B})$, and we show that **Separator** wins the separability game G . Since \mathcal{S} is a separator, we have that $L(\mathcal{S}) \perp L(\mathcal{B})$, and by Lemma 2.10 there exists a pathfinder \mathcal{P} . Consider the following strategy of **Separator**, with memory structure Q^S and initial memory state q_0^S . At the i -th round of G , starting with a memory state q_i^S ,

- [S: c] **Separator** plays the priority $c_i := \Omega^S(q_i^S) \in C$.
- [I: a] **Input** plays an arbitrary letter $a_i \in \Sigma$.
- [S: f] **Separator** plays the selector $f_i := \mathcal{P}(\delta_i^S, -) \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^S(q_i^S, a_i)} = \{\delta_i^S\}$.
- [I: d] **Input** plays an arbitrary direction $d_i \in \{\mathbf{L}, \mathbf{R}\}$.

The next memory state is $q_{i+1}^S := q_{d_i, i}^S$, where $\delta_i^S = (q_i^S, a_i, q_{\mathbf{L}, i}^S, q_{\mathbf{R}, i}^S)$. This concludes the description of the i -th round of G . Let the resulting infinite

play be $\pi = (c_0, a_0, f_0, d_0)(c_1, a_1, f_1, d_1) \cdots$, with induced infinite path $b := (a_0, d_0)(a_1, d_1) \cdots$. Let $\vec{\delta}^{\mathcal{S}} := \delta_0^{\mathcal{S}} \delta_1^{\mathcal{S}} \cdots$ be the sequence of transitions used to define the selectors f_i . Clearly $\vec{\delta}^{\mathcal{S}} \in \Delta^{\mathcal{S}}(b)$.

First, we argue that $\pi \in \mathbf{W}_{\mathcal{A}}$ holds. Let $\vec{\delta}^{\mathcal{A}} = \delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \cdots \in \Delta^{\mathcal{A}}(b)$ be an accepting sequence of transitions of the automaton \mathcal{A} . Since each state of \mathcal{A} is productive, one can construct a tree $t \in \mathbf{L}(\mathcal{A})$ s.t. $b \in \text{Path}(t)$. Since $\mathbf{L}(\mathcal{A}) \subseteq \mathbf{L}(\mathcal{S})$ by the assumption, $t \in \mathbf{L}(\mathcal{S})$ as well, and since \mathcal{S} is deterministic, the unique run of \mathcal{S} over t is accepting. By the definition of Separator's strategy, the sequence of priorities along the branch $d_0 d_1 \cdots$ of this accepting run is precisely $c_0 c_1 \cdots$, which thus must be accepting, as required.

Regarding $\mathbf{W}_{\mathcal{B}}$, let $\vec{\delta}^{\mathcal{B}} := \delta_0^{\mathcal{B}} \delta_1^{\mathcal{B}} \cdots \in \Delta^{\mathcal{B}}(b)$ be an accepting sequence of transitions over the path b conform to the selectors f_i , i.e., for every $i \in \omega$ we have $f_i(\delta_i^{\mathcal{B}}) = d_i$. By the definition of f_i , for every $i \in \omega$ we have $d_i = \mathcal{P}(\delta_i^{\mathcal{S}}, \delta_i^{\mathcal{B}})$. Thus, the assumptions of Corollary 2.11 are satisfied and at least one of the sequences $\vec{\delta}^{\mathcal{S}}, \vec{\delta}^{\mathcal{B}}$ must be rejecting. Since we assumed that $\vec{\delta}^{\mathcal{B}}$ is accepting, it means that $\vec{\delta}^{\mathcal{S}}$ is rejecting, and so is $c_0 c_1 \cdots$ since $c_i = \Omega^{\mathcal{S}}(\delta_i^{\mathcal{S}})$. \square

3.1 A variant of the separability game for trees

A first attempt at generalising the case of ω -words to infinite trees is to let Input play a *direction* $d_i \in \{\mathbf{L}, \mathbf{R}\}$ after she plays a letter a_i , and not considering selectors for Separator. This yields the following simpler variant of the game considered at the beginning of this section. At round i of the separability game,

[S: c] Separator plays a priority $c_i \in C$.

[I: a] Input plays a letter $a_i \in \Sigma$.

[I: d] Input plays a direction $d_i \in \{\mathbf{L}, \mathbf{R}\}$.

Separator wins the corresponding infinite play $\pi = (c_0, a_0, d_0)(c_1, a_1, d_1) \cdots \in (C \times \Sigma \times \{\mathbf{L}, \mathbf{R}\})^\omega$ if the induced infinite path $b = (a_0, d_0)(a_1, d_1) \cdots \in (\Sigma \times \{\mathbf{L}, \mathbf{R}\})^\omega$ satisfies the following two conditions:

- $\pi \in \mathbf{W}_{\mathcal{A}}$: If there exists an accepting sequence of transitions $\vec{\delta}^{\mathcal{A}} = \delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \cdots \in \Delta^{\mathcal{A}}(b)$, then $c_0 c_1 \cdots$ is accepting.
- $\pi \in \mathbf{W}_{\mathcal{B}}$: If there exists an accepting sequence of transitions $\vec{\delta}^{\mathcal{B}} = \delta_0^{\mathcal{B}} \delta_1^{\mathcal{B}} \cdots \in \Delta^{\mathcal{B}}(b)$, then $c_0 c_1 \cdots$ is rejecting.

It turns out that the winning condition above is not strong enough in order to characterise deterministic separability over languages of infinite trees. A deterministic automaton \mathcal{S} is *universally rejecting* on a set of trees L if, for every $t \in L$, *all* branches in the corresponding run in \mathcal{S} are rejecting. The following lemma states that the game in this section characterises separability by deterministic automata \mathcal{S} which are universally rejecting on $\mathbf{L}(\mathcal{B})$.

Lemma 3.2. *Separator wins the game above if, and only if, $\mathbf{L}(\mathcal{A}), \mathbf{L}(\mathcal{B})$ can be separated by a deterministic separator \mathcal{S} with priorities from C which is universally rejecting on $\mathbf{L}(\mathcal{B})$.*

Proof sketch. The proof is analogous to that of Lemma 3.1. \square

4 Separability by deterministic automata

In this section we present a game-theoretic characterisation of the deterministic separability problem. Notice that here we do not fix in advance a finite set of priorities C . The deterministic-separability game $G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$ below is a variant of the game with fixed priorities C from Section 3.

Deterministic-separability game $G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$

At the i -th round:

- [I: a] Input plays a letter $a_i \in \Sigma$.
- [S: f] Separator plays a selector $f_i \in \{\text{L}, \text{R}\}^{\Delta^{\mathcal{B}}(a_i)}$.
- [I: d] Input plays a direction $d_i \in \{\text{L}, \text{R}\}$.

Separator wins the resulting infinite play $\pi = (a_0, f_0, d_0)(a_1, f_1, d_1) \cdots$, with induced infinite path $b := (a_0, d_0)(a_1, d_1) \cdots$, if at least one of the two conditions below fails:

- $\pi \in \mathbf{W}_{\mathcal{A}}$: There exists an accepting sequence of transitions $\vec{\delta}^{\mathcal{A}} = \delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \cdots \in \Delta^{\mathcal{A}}(b)$.
- $\pi \in \mathbf{W}_{\mathcal{B}}$: There exists an accepting sequence of transitions $\vec{\delta}^{\mathcal{B}} = \delta_0^{\mathcal{B}} \delta_1^{\mathcal{B}} \cdots \in \Delta^{\mathcal{B}}(b)$ s.t. for every $i \in \omega$ we have $f_i(\delta_i^{\mathcal{B}}) = d_i$.

Before we prove the equivalence between the game and the existence of a separator, we define a separator candidate, namely the path-closure of $L(\mathcal{A})$. This is important since it will turn out that if a separator exists, then the path-closure is itself a separator. Given a language of trees L , its *path-closure*, denoted $\forall\text{Path}(L)$, is the set of all trees t s.t. for every path $b \in \text{Path}(t)$ there exists some tree $t' \in L$ s.t. $b \in \text{Path}(t')$ as well.

The following lemma states formally some basic facts justifying that $\forall\text{Path}(_)$ is indeed a closure operator.

Lemma 4.1.

1. *The path-closure operator is monotonic (w.r.t. set inclusion):*
If $L \subseteq M$, then $\forall\text{Path}(L) \subseteq \forall\text{Path}(M)$.
2. *The path-closure operator is non-decreasing (w.r.t. set inclusion):*
For any language L , $L \subseteq \forall\text{Path}(L)$.
3. *The path-closure $\forall\text{Path}(L)$ of L is the smallest (w.r.t. set inclusion) language of infinite trees M s.t. a) M contains L : $L \subseteq M$, and b) M is path-closed: $\forall\text{Path}(M) \subseteq M$.*

Proof. The first two properties are clear. For the third property, $M := \forall\text{Path}(L)$ itself satisfies a) since the path-closure operator is non-increasing, and b) since the path-closure operator is idempotent $\forall\text{Path}(\forall\text{Path}(L)) = \forall\text{Path}(L)$. Now let M be an arbitrary language s.t. a) $L \subseteq M$, and b) $\forall\text{Path}(M) \subseteq M$. We immediately have

$$L \stackrel{(1)}{\subseteq} \forall\text{Path}(L) \stackrel{(2)}{\subseteq} \forall\text{Path}(M) \stackrel{(3)}{\subseteq} M,$$

where (1) follows from the fact that the path-closure operator is non-decreasing, (2) from the fact that it is monotone, and (3) from the fact that M is path-closed.

Now consider any deterministic tree automaton \mathcal{B} . Observe that since \mathcal{B} is deterministic, we clearly get $\forall\text{Path}(\mathbf{L}(\mathcal{B})) = \mathbf{L}(\mathcal{B})$. Now assume that $\mathbf{L}(\mathcal{A}) \subseteq \mathbf{L}(\mathcal{B})$. By monotonicity of $\forall\text{Path}(_)$ we obtain that $\forall\text{Path}(\mathbf{L}(\mathcal{A})) \subseteq \forall\text{Path}(\mathbf{L}(\mathcal{B})) = \mathbf{L}(\mathcal{B})$. \square

The path-closure operator is directly connected with deterministic automata.

Lemma 4.2 (c.f. [36, Proposition 1]). *Given a nondeterministic automaton \mathcal{A} one can construct a deterministic automaton $\mathcal{A}^{\text{path}}$ recognising the path closure of $\mathbf{L}(\mathcal{A})$, i.e., $\mathbf{L}(\mathcal{A}^{\text{path}}) = \forall\text{Path}(\mathbf{L}(\mathcal{A}))$. Moreover, $\mathbf{L}(\mathcal{A}^{\text{path}})$ is the smallest deterministic language containing $\mathbf{L}(\mathcal{A})$.*

Proof. Fix a nondeterministic automaton $\mathcal{A} = (\Sigma, Q, q_0, \Omega, \Delta)$. Our aim is to construct a deterministic automaton $\mathcal{A}^{\text{path}}$ recognising the path closure of $\mathbf{L}(\mathcal{A})$, i.e., $\mathbf{L}(\mathcal{A}^{\text{path}}) = \forall\text{Path}(\mathbf{L}(\mathcal{A}))$. Let $\mathcal{D} = (\Sigma \times \{\mathbf{L}, \mathbf{R}\}, Q^{\mathcal{D}}, q_0^{\mathcal{D}}, \Delta^{\mathcal{D}}, \Omega^{\mathcal{D}})$ be a deterministic parity automaton over ω -words over the alphabet $\Sigma \times \{\mathbf{L}, \mathbf{R}\}$ that recognises the set of paths b s.t. there exists an accepting sequence of transitions in $\Delta^{\mathcal{A}}(b)$. It is easy to see how a nondeterministic such automaton can be obtained, and it can be determinised thanks to Lemma 2.2. Consider a deterministic parity tree automaton $\mathcal{A}^{\text{path}} := (\Sigma, Q^{\mathcal{D}}, q_0^{\mathcal{D}}, \Omega^{\mathcal{D}}, \Delta')$ which has the same set of states, initial state, and priority mapping as \mathcal{D} , and transitions are of the form

$$\Delta' = \{(q, a, \Delta^{\mathcal{D}}(q, (a, \mathbf{L})), \Delta^{\mathcal{D}}(q, (a, \mathbf{R})) \mid q \in Q^{\mathcal{D}}, a \in \Sigma\}.$$

Now, we claim that the following conditions are equivalent, for a tree $t \in \text{Tr}_{\Sigma}$:

1. $t \in \forall\text{Path}(\mathbf{L}(\mathcal{A}))$,
2. for every path $b \in \text{Path}(t)$ there exists a tree $t' \in \mathbf{L}(\mathcal{A})$ s.t. $b \in \text{Path}(t')$,
3. for every path $b \in \text{Path}(t)$ there exists an accepting sequence of transitions in $\Delta^{\mathcal{A}}(b)$,
4. for every path $b \in \text{Path}(t)$ the automaton \mathcal{D} accepts b ,
5. $t \in \mathbf{L}(\mathcal{A}^{\text{path}})$.

Indeed, the only nontrivial implication is “3 \Rightarrow 2”, however, since every state of \mathcal{A} is productive, one can easily construct the tree t' by extending the considered sequence of transitions of \mathcal{A} to the subtrees outside the path b . We can thus conclude $\forall\text{Path}(\mathbf{L}(\mathcal{A})) = \mathbf{L}(\mathcal{A}^{\text{path}})$, as required. \square

The following lemma binds together the game $G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$, separability, and path-closures.

Lemma 4.3. *The following three conditions are equivalent:*

1. *Separator wins the deterministic-separability game $G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$.*
2. *The automaton $\mathcal{A}^{\text{path}}$ is a deterministic separator for $\mathbf{L}(\mathcal{A})$, $\mathbf{L}(\mathcal{B})$.*
3. *There exists a deterministic separator for $\mathbf{L}(\mathcal{A})$, $\mathbf{L}(\mathcal{B})$.*

Proof. We begin by proving “1 \Rightarrow 2”. Assume that **Separator** wins the separability game $G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$ by a finite-memory winning strategy $\mathcal{M} = (M, \ell_0, \bar{f}, \tau)$ which has one decision function \bar{f} assigning to each $\ell \in M$ and $a \in \Sigma$ a selector $\bar{f}(\ell, a) \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{B}}(a)}$. Moreover, the type of the memory update function is $\tau: M \times \Sigma \times \{\mathbf{L}, \mathbf{R}\} \rightarrow M$.

Let $\mathcal{S} := \mathcal{A}^{\text{path}}$ be the deterministic automaton recognising the path closure of $L(\mathcal{A})$. We show that \mathcal{S} separates $L(\mathcal{A})$, $L(\mathcal{B})$.

The condition $L(\mathcal{A}) \subseteq L(\mathcal{S})$ follows immediately from the fact that $L(\mathcal{S}) = \forall\text{Path}(L(\mathcal{A}))$ and by Item 1 of Lemma 4.1 we know that $\forall\text{Path}(L(\mathcal{A})) \supseteq L(\mathcal{A})$.

We now argue that $L(\mathcal{S})$ and $L(\mathcal{B})$ are disjoint. Towards reaching a contradiction, assume that $t \in L(\mathcal{S}) \cap L(\mathcal{B})$ belongs to their intersection. Let $\rho^{\mathcal{S}}$ be the unique run of \mathcal{S} over t , and let $\rho^{\mathcal{B}}$ be an accepting run of \mathcal{B} over t . Consider a play π of $G := G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$ where at the i -th round, **Separator** plays according to the strategy \mathcal{M} with current memory state $\ell_i \in M$ and **Input** plays as follows:

[I: a] **Input** plays the letter $a_i := t(u_i) \in \Sigma$, where $u_i := d_0 \cdots d_{i-1}$.

[S: f] **Separator** plays the selector $f_i := \bar{f}(\ell_i, a_i) \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{B}}(a_i)}$.

[I: d] **Input** plays the direction $d_i := f_i(\delta_i^{\mathcal{B}}) \in \{\mathbf{L}, \mathbf{R}\}$, where $\delta_i^{\mathcal{B}}$ is the respective transition of $\rho^{\mathcal{B}}$, i.e., $\delta_i^{\mathcal{B}} := (\rho^{\mathcal{B}}(u_i), t(u_i), \rho^{\mathcal{B}}(u_i\mathbf{L}), \rho^{\mathcal{B}}(u_i\mathbf{R}))$.

The next memory state is $\ell_{i+1} := \tau(\ell_i, a_i, d_i)$. Let π be the obtained play. By the choice of directions d_i we know that the sequence of transitions $\vec{\delta}^{\mathcal{B}} = \delta_0^{\mathcal{B}}\delta_1^{\mathcal{B}} \cdots \in (\Delta^{\mathcal{B}})^{\omega}$ satisfies $f_i(\delta_i^{\mathcal{B}}) = d_i$ for every $i \in \omega$. Moreover, as the run $\rho^{\mathcal{B}}$ is accepting, we know that $\vec{\delta}^{\mathcal{B}}$ is accepting. Therefore, $\mathbf{W}_{\mathcal{B}}$ holds for this play. It means that $\mathbf{W}_{\mathcal{A}}$ must fail, meaning that the infinite path $b := (a_0, d_0)(a_1, d_1) \cdots$ does not belong to $\text{Path}(L(\mathcal{A}))$. However, this is a contradiction with the assumption that the run $\rho^{\mathcal{S}}$ of \mathcal{S} over t is accepting.

The implication “2 \Rightarrow 3” is trivial.

Finally, we prove “3 \Rightarrow 1”. Assume that $\mathcal{S} = (\Sigma, Q, q_0, \Delta, \Omega)$ is a deterministic automaton separating $L(\mathcal{A})$, $L(\mathcal{B})$, and we show that **Separator** wins the separability game $G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$. Since \mathcal{S} is a separator, we have that $L(\mathcal{S}) \perp L(\mathcal{B})$, and thus **Pathfinder** wins the disjointness game $G^{\text{dis}}(\mathcal{S}, \mathcal{B})$, see Lemma 2.10. Let \mathcal{P} be a pathfinder as in Section 2.6.

Consider the following strategy of **Separator**, with the memory structure $Q^{\mathcal{S}}$ and the initial memory state $q_0^{\mathcal{S}}$. At the i -th round of $G := G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$, starting with a memory state $q_i^{\mathcal{S}}$, **Separator** plays as follows:

[I: a] **Input** plays an arbitrary letter $a_i \in \Sigma$.

[S: f] **Separator** plays the selector $f_i := \mathcal{P}(\delta_i^{\mathcal{S}}, _) \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{B}}(a_i)}$, where $\Delta^{\mathcal{S}}(q_i^{\mathcal{S}}, a_i) = \{\delta_i^{\mathcal{S}}\}$.

[I: d] **Input** plays an arbitrary direction $d_i \in \{\mathbf{L}, \mathbf{R}\}$.

The next memory state is $q_{i+1}^{\mathcal{S}} := q_{d,i}^{\mathcal{S}}$, where $\delta_i^{\mathcal{S}} = (q_i^{\mathcal{S}}, a_i, q_{\mathbf{L},i}^{\mathcal{S}}, q_{\mathbf{R},i}^{\mathcal{S}})$.

Let the resulting infinite play be π , with the induced infinite path $b := (a_0, d_0)(a_1, d_1) \cdots$. Let $\vec{\delta}^{\mathcal{S}} := \delta_0^{\mathcal{S}}\delta_1^{\mathcal{S}} \cdots$ be the sequence of transitions used to define the selectors f_i .

Assume for the sake of contradiction that both $\mathbf{W}_{\mathcal{A}}$ and $\mathbf{W}_{\mathcal{B}}$ hold, as witnessed by accepting sequences of transitions $\vec{\delta}^{\mathcal{A}} = \delta_0^{\mathcal{A}}\delta_1^{\mathcal{A}} \cdots \in \Delta^{\mathcal{A}}(b)$ and

$\vec{\delta}^{\mathcal{B}} = \delta_0^{\mathcal{B}} \delta_1^{\mathcal{B}} \dots \in \Delta^{\mathcal{B}}(b)$. The sequence $\vec{\delta}^{\mathcal{A}}$ implies that there exists a tree $t \in \text{Tr}_{\Sigma}$ s.t. b is a path of t and $t \in L(\mathcal{A})$. Since $L(\mathcal{A}) \subseteq L(\mathcal{S})$, the run of the automaton \mathcal{S} must be accepting on t and therefore the sequence of transitions $\vec{\delta}^{\mathcal{S}}$ is accepting. Moreover, the choice of the selectors f_i means that for every $i \in \omega$ we have $d_i = \mathcal{P}(\delta_i^{\mathcal{S}}, \delta_i^{\mathcal{B}})$. Thus, the assumptions of Corollary 2.11 are satisfied and at least one of the sequences $\vec{\delta}_i^{\mathcal{S}}, \vec{\delta}_i^{\mathcal{B}}$ must be rejecting. A contradiction, because we assumed that $\vec{\delta}^{\mathcal{B}}$ is accepting and we know that $\vec{\delta}^{\mathcal{S}}$ is also accepting. \square

5 Separability by game automata

In this section we provide a game-theoretic characterisation for the game automata separability problem. Fix two automata \mathcal{A} and \mathcal{B} and consider the following separability game $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$. The new ingredient is that **Separator** can choose a *mode*—a symbol from the set $\{\vee, \wedge\}$. It has two uses. First, in the construction of the separating game automaton, the mode dictates whether there will be a conjunctive or a disjunctive transition. Second, depending on the chosen mode, **Separator** will have to play a selector for the automaton \mathcal{A} or \mathcal{B} , which will guarantee that the constructed automaton is a separator.

Game-separability game $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$

At the i -th round:

[I: a] Input plays a letter $a_i \in \Sigma$.

[S: m] Separator plays a mode $m_i \in \{\vee, \wedge\}$.

[S: f] Separator plays either

1. a selector $f_i \in \{\text{L}, \text{R}\}^{\Delta^{\mathcal{A}}(a_i)}$ for \mathcal{A} if $m_i = \vee$ or
2. a selector $f_i \in \{\text{L}, \text{R}\}^{\Delta^{\mathcal{B}}(a_i)}$ for \mathcal{B} if $m_i = \wedge$.

[I: d] Input plays a direction $d_i \in \{\text{L}, \text{R}\}$.

Separator wins an infinite play $\pi = (a_0, m_0, f_0, d_0)(a_1, m_1, f_1, d_1) \dots$ inducing a path $b = (a_0, d_0)(a_1, d_1) \dots$ whenever at least one of the two conditions below fail:

- $\pi \in \mathbf{W}_{\mathcal{A}}$: There exists an accepting sequence of transitions $\vec{\delta}^{\mathcal{A}} = \delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \dots \in \Delta^{\mathcal{A}}(b)$ s.t. for all $i \in \mathbb{N}$ we have $(m_i = \vee) \Rightarrow f_i(\delta_i^{\mathcal{A}}) = d_i$.
- $\pi \in \mathbf{W}_{\mathcal{B}}$: There exists an accepting sequence of transitions $\vec{\delta}^{\mathcal{B}} = \delta_0^{\mathcal{B}} \delta_1^{\mathcal{B}} \dots \in \Delta^{\mathcal{B}}(b)$ s.t. for all $i \in \mathbb{N}$ we have $(m_i = \wedge) \Rightarrow f_i(\delta_i^{\mathcal{B}}) = d_i$.

Lemma 5.1. *Separator wins the separability game $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$ if, and only if, there exists a game automaton \mathcal{S} separating $L(\mathcal{A}), L(\mathcal{B})$.*

In the proof of this lemma we will build separating automata with a more general acceptance condition than the parity condition, which will simplify the technical details. A *generalised game automaton* $\mathcal{A} = (\Sigma, Q, q_0, \Delta, \mathcal{D})$ is just like a game automaton except that the priority mapping Ω is replaced by a deterministic ω -word parity automaton \mathcal{D} over alphabet $\Sigma \times \{\text{L}, \text{R}\}$. A run

$\rho \in \text{Tr}_Q$ of such an automaton over a tree $t \in \text{Tr}_\Sigma$ is *accepting* if for every path $b = (a_0, d_0)(a_1, d_1) \cdots \in \text{Path}(t)$ either $\rho(d_0 \cdots d_{i-1}) = \top$ for some $i \in \omega$, or $b \in L(\mathcal{D})$. The acceptance game $G^{\text{acc}}(\mathcal{A}, t)$ can easily be adapted to the case of a generalised game automaton \mathcal{A} by only modifying the winning condition.

Lemma 5.2. *A generalised game automaton \mathcal{A} with a generalised acceptance condition recognised by a deterministic parity automaton \mathcal{D} can be transformed into an equivalent (ordinary) game automaton \mathcal{B} of size polynomial in \mathcal{A} and \mathcal{D} .*

Proof. Consider the game automaton \mathcal{B} defined as the following product of \mathcal{A} and \mathcal{D} :

$$\mathcal{B} := (\Sigma, (Q^{\mathcal{A}} \setminus \{\top\}) \times Q^{\mathcal{D}} \cup \{\top\}, (q_0^{\mathcal{A}}, q_0^{\mathcal{D}}), \Delta^{\mathcal{B}}, \Omega^{\mathcal{B}}),$$

where $\Omega^{\mathcal{B}}$ is just inherited from \mathcal{D} , i.e., $\Omega^{\mathcal{B}}(q^{\mathcal{A}}, q^{\mathcal{D}}) := \Omega^{\mathcal{D}}(q^{\mathcal{D}})$. Moreover, for each conjunctive \mathcal{A} -transition $(q^{\mathcal{A}}, a, q_{\text{L}}^{\mathcal{A}}, q_{\text{R}}^{\mathcal{A}}) \in \Delta^{\mathcal{A}}$, $\Delta^{\mathcal{B}}$ contains the transition

$$((q^{\mathcal{A}}, q^{\mathcal{D}}), a, (q_{\text{L}}^{\mathcal{A}}, q_{\text{L}}^{\mathcal{D}}), (q_{\text{R}}^{\mathcal{A}}, q_{\text{R}}^{\mathcal{D}})),$$

where for $d \in \{\text{L}, \text{R}\}$ we have $\Delta^{\mathcal{D}}(q^{\mathcal{D}}, (a, d)) = q_d^{\mathcal{D}}$. Similarly, for each disjunctive \mathcal{A} -transition $(q^{\mathcal{A}}, a, q_{\text{L}}^{\mathcal{A}}, \top) \in \Delta^{\mathcal{A}}$ (resp., $(q^{\mathcal{A}}, a, \top, q_{\text{R}}^{\mathcal{A}}) \in \Delta^{\mathcal{A}}$), $\Delta^{\mathcal{B}}$ contains the transition $((q^{\mathcal{A}}, q^{\mathcal{D}}), a, (q_{\text{L}}^{\mathcal{A}}, q_{\text{L}}^{\mathcal{D}}), \top)$ (resp., $((q^{\mathcal{A}}, q^{\mathcal{D}}), a, \top, (q_{\text{R}}^{\mathcal{A}}, q_{\text{R}}^{\mathcal{D}}))$), where for $d \in \{\text{L}, \text{R}\}$ we have $\Delta^{\mathcal{D}}(q^{\mathcal{D}}, (a, d)) = q_d^{\mathcal{D}}$.

Notice that for every tree $t \in \text{Tr}_\Sigma$ there is a bijection between the runs $\rho^{\mathcal{A}}$ of \mathcal{A} over t and runs $\rho^{\mathcal{B}}$ of \mathcal{B} over t : given a run $\rho^{\mathcal{B}}$ we can just project it onto the first coordinate to obtain $\rho^{\mathcal{A}}$, and the run $\rho^{\mathcal{B}}$ is obtained in a top-down deterministic way from $\rho^{\mathcal{A}}$ by running the automaton \mathcal{D} deterministically on all the paths. Therefore, it is enough to argue that if $\rho^{\mathcal{A}}$ and $\rho^{\mathcal{B}}$ are two such runs then $\rho^{\mathcal{A}}$ is accepting if and only if $\rho^{\mathcal{B}}$ is. Consider a branch $d_0 d_1 \cdots \in \{\text{L}, \text{R}\}^\omega$ and let $u_i := d_0 \cdots d_{i-1}$ for $i \in \omega$. Without loss of generality assume that $\rho^{\mathcal{A}}(u_i) \neq \top$ for every $i \in \omega$ (otherwise both runs are accepting on this branch). For each $i \in \omega$ let $(q_i^{\mathcal{A}}, q_i^{\mathcal{D}}) := \rho^{\mathcal{B}}(u_i)$ and notice that by the choice of the runs $\rho^{\mathcal{A}}$ and $\rho^{\mathcal{B}}$ we know that $\rho^{\mathcal{A}}(u_i) = q_i^{\mathcal{A}}$. Now let $b := (t(u_0), d_0)(t(u_1), d_1) \cdots$ be the path used to define the generalised acceptance condition of \mathcal{A} on the considered branch. By the construction of the automaton \mathcal{B} , we know that the sequence of states $q_0^{\mathcal{D}} q_1^{\mathcal{D}} \cdots$ is the run of \mathcal{D} on b . Therefore, $\rho^{\mathcal{A}}$ satisfies the generalised acceptance condition on the path b if and only if $\rho^{\mathcal{B}}$ satisfies the parity condition on the branch $d_0 d_1 \cdots$. \square

We now prove Lemma 5.1.

Soundness. Assume that **Separator** wins the game-separability game above $G := G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$ and we show that there exists a game automaton \mathcal{S} separating $L(\mathcal{A})$ from $L(\mathcal{B})$. Let $\mathcal{M} = (M, \ell_0, (\overline{m}, \overline{f}), \tau)$ be a finite-memory winning strategy of **Separator** in G .

Before we move to the construction of the separating automaton, we first define its generalised acceptance condition. Let $L_{\mathcal{A}}$ (resp., $L_{\mathcal{B}}$) be the set of those paths $b = (a_0, d_0)(a_1, d_1) \cdots \in (\Sigma \times \{\text{L}, \text{R}\})^\omega$ s.t. the unique play π of G in which **Input** plays consecutive letters and directions from b and **Separator** uses her winning strategy \mathcal{M} , satisfies the condition $\mathbf{W}_{\mathcal{A}}$ (resp., $\mathbf{W}_{\mathcal{B}}$). Since the strategy \mathcal{M} is winning for **Separator**, the languages $L_{\mathcal{A}}$ and $L_{\mathcal{B}}$ are disjoint. Moreover, since the strategy \mathcal{M} is finite memory and both $\mathbf{W}_{\mathcal{A}}$, $\mathbf{W}_{\mathcal{B}}$ are ω -regular, so

are the languages $L_{\mathcal{A}}$ and $L_{\mathcal{B}}$. Let \mathcal{D} be any deterministic automaton over ω -words that separates $L_{\mathcal{A}}$ from $L_{\mathcal{B}}$ (the simplest case is to take \mathcal{D} recognising the language $L_{\mathcal{A}}$). We build a separating automaton as a generalised game automaton

$$\mathcal{S} := \alpha(\mathcal{M}, \mathcal{D}) := (\Sigma, M \cup \{\top\}, \ell_0, \Delta^{\mathcal{S}}, \mathcal{D}), \text{ where}$$

$$\Delta^{\mathcal{S}}(\ell, a) := \begin{cases} \{(\ell, a, \ell_{\mathbf{L}}, \top), (\ell, a, \top, \ell_{\mathbf{R}})\} & \text{if } \overline{m}(\ell, a) = \vee, \\ \{(\ell, a, \ell_{\mathbf{L}}, \ell_{\mathbf{R}})\} & \text{if } \overline{m}(\ell, a) = \wedge, \end{cases}$$

for every $\ell \in M$ and $a \in \Sigma$, where for $d \in \{\mathbf{L}, \mathbf{R}\}$ we have $\ell_d := \tau(\ell, a, d)$. We now show that \mathcal{S} separates $L(\mathcal{A})$ from $L(\mathcal{B})$. In order to show $L(\mathcal{A}) \subseteq L(\mathcal{S})$, let $t \in L(\mathcal{A})$ as witnessed by an accepting run $\rho^{\mathcal{A}}$. We show that **Automaton** wins the acceptance game $G_{\mathcal{S}} := G^{\text{acc}}(\mathcal{S}, t)$. To show this we play in parallel the separability game G and the acceptance game $G_{\mathcal{S}}$, maintaining the following invariant: At the i -th round, the current finite path of the input tree t is $(a_0, d_0) \cdots (a_{i-1}, d_{i-1})$, **Separator's** winning strategy \mathcal{M} in the separability game G is in memory state ℓ_i , the current state of the separating automaton \mathcal{S} in the acceptance game $G_{\mathcal{S}}$ is also ℓ_i , and $\rho^{\mathcal{A}}(d_0 \cdots d_{i-1}) = q_i^{\mathcal{A}}$. The i -th round is then played as follows:

$G.[\mathbf{I}: a]$ Input plays the letter $a_i := t(u_i)$ for $u_i := d_0 \cdots d_{i-1}$.

$G.[\mathbf{S}: m]$ Separator plays the mode $m_i := \overline{m}(\ell_i, a_i) \in \{\vee, \wedge\}$.

$G.[\mathbf{S}: f]$ Separator plays either

1. a selector $f_i := \overline{f}(\ell_i, a_i) \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{A}}(a_i)}$ for \mathcal{A} if $m_i = \vee$ or
2. a selector $f_i := \overline{f}(\ell_i, a_i) \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{B}}(a_i)}$ for \mathcal{B} if $m_i = \wedge$.

$G_{\mathcal{S}}.[\mathbf{A}: \delta]$ Automaton plays the transition $\delta_i^{\mathcal{S}} \in \Delta^{\mathcal{S}}(\ell_i, a_i)$, defined as follows. Let $\delta_i^{\mathcal{A}} := (\rho^{\mathcal{A}}(u_i), t(u_i), \rho^{\mathcal{A}}(u_i \mathbf{L}), \rho^{\mathcal{A}}(u_i \mathbf{R}))$ be the \mathcal{A} -transition used in u_i by the run $\rho^{\mathcal{A}}$. We distinguish two cases.

1. In the first case, assume that **Separator** played $m_i = \vee$ and $f_i \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{A}}(a_i)}$. It means that $\Delta^{\mathcal{S}}((\ell_i, q_i), a_i)$ contains two disjunctive transitions, $\delta_{\mathbf{L}, i}^{\mathcal{S}} := (\ell_i, a_i, \ell_{\mathbf{L}, i}, \top)$ and $\delta_{\mathbf{R}, i}^{\mathcal{S}} := (\ell_i, a_i, \top, \ell_{\mathbf{R}, i})$. Let us put $\delta_i^{\mathcal{S}} := \delta_{f_i(\delta_i^{\mathcal{A}}), i}^{\mathcal{S}}$, i.e., the transition that sends a non- \top state in the direction given by $f_i(\delta_i^{\mathcal{A}})$.
2. In the second case, **Separator** played $m_i = \wedge$ and $f_i \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{B}}(a_i)}$. It means that $\Delta^{\mathcal{S}}(\ell_i, a_i)$ contains one conjunctive transition $\delta_i^{\mathcal{S}} := (\ell_i, a_i, \ell_{\mathbf{L}, i}, \ell_{\mathbf{R}, i})$.

$G_{\mathcal{S}}.[\mathbf{I}: d]$ Input plays an arbitrary direction $d_i \in \{\mathbf{L}, \mathbf{R}\}$.

$G.[\mathbf{I}: d]$ Input plays the direction $d_i \in \{\mathbf{L}, \mathbf{R}\}$.

If $m_i = \vee$ and $d_i \neq f_i(\delta_i^{\mathcal{A}})$ then the next position of the acceptance game $G_{\mathcal{S}}$ is $(u_i d_i, \top)$, which is a winning position for **Automaton**. Therefore, w.l.o.g. we assume that:

$$\forall i \in \omega. (m_i = \vee) \Rightarrow f_i(\delta_i^{\mathcal{A}}) = d_i. \quad (2)$$

Moreover, the new state of \mathcal{S} in $G_{\mathcal{S}}$ is $\ell_{i+1} := \tau(\ell_i, a_i, d_i)$. Similarly, the new memory state of \mathcal{M} in G is ℓ_{i+1} . This concludes the description of the i -th round of both games. Clearly the invariant is preserved. We argue that **Automaton** wins the resulting infinite play $(\delta_0^{\mathcal{S}}, d_0)(\delta_1^{\mathcal{S}}, d_1) \cdots$ of the acceptance game $G_{\mathcal{S}}$. Consider the infinite play $\pi = (a_0, m_0, f_0, d_0)(a_1, m_1, f_1, d_1) \cdots$ of the separability game G . Since the run $\rho^{\mathcal{A}}$ is accepting, the infinite sequence of \mathcal{A} -transitions $\delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \cdots$ is accepting. Thus, (2) implies that $\pi \in \mathbf{W}_{\mathcal{A}}$. Therefore, the infinite path $b := (a_0, d_0)(a_1, d_1) \cdots$ belongs to $L_{\mathcal{A}} \subseteq L(\mathcal{D})$ and thus the corresponding infinite play $(\delta_0^{\mathcal{S}}, d_0)(\delta_1^{\mathcal{S}}, d_1) \cdots$ of the acceptance game $G_{\mathcal{S}}$ is winning for **Automaton**, as required. This concludes the argument establishing $L(\mathcal{A}) \subseteq L(\mathcal{S})$.

It remains to show that $L(\mathcal{S}) \perp L(\mathcal{B})$, which is the same as $L(\mathcal{B}) \subseteq L(\mathcal{S}^c)$ for the complement game automaton. This follows directly from the construction above via the duality of the game G . Indeed, consider the generalised game automaton \mathcal{S} as defined in Section 5. Let \mathcal{S}^c be the complementary game automaton $(\Sigma, M \cup \{\top\}, \ell_0, \Delta^{\mathcal{S}^c}, \mathcal{D}^c)$, which recognises the complement language of \mathcal{S} . (Here, $\Delta^{\mathcal{S}^c}$ is the dualisation of $\Delta^{\mathcal{S}}$ as in the proof of Lemma 2.1, and \mathcal{D}^c is the complementary automaton to \mathcal{D} —its priorities are increased by 1.) We first observe that if $\mathcal{M} = (M, \ell_0, (\overline{m}, \overline{f}), \tau)$ is a strategy of **Separator** in $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$, then $\mathcal{M}^c := (M, \ell_0, (\overline{m}^c, \overline{f}), \tau)$ with \overline{m}^c returning the opposite mode than \overline{m} is a strategy of **Separator** in $G_{\text{game}}^{\text{sep}}(\mathcal{B}, \mathcal{A})$. By the symmetry of the winning condition, \mathcal{M} is winning if and only if \mathcal{M}^c is winning. The following claim follows directly from the definition of α .

Claim 5.3. *If \mathcal{M} is a winning strategy of **Separator** in $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$ then*

$$\mathcal{S}^c = \alpha(\mathcal{M}, \mathcal{D})^c = \alpha(\mathcal{M}^c, \mathcal{D}^c).$$

Therefore, by applying the argument that $L(\mathcal{A}) \subseteq L(\mathcal{S})$ to $\mathcal{M}^c, \mathcal{D}^c$ for the game $G_{\text{game}}^{\text{sep}}(\mathcal{B}, \mathcal{A})$, we obtain $L(\mathcal{B}) \subseteq L(\mathcal{S}^c)$, as required. \square

Completeness. Assume that there exists a game automaton \mathcal{S} that separates $L(\mathcal{A})$ from $L(\mathcal{B})$. We need to show that **Separator** wins the separability game $G := G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$. Let $\mathcal{R} := \mathcal{S}^c$ be the syntactic dual of the game automaton \mathcal{S} as in Lemma 2.1. Thus, the automata \mathcal{S} and \mathcal{R} share the same set of states. Also, their transitions are related: the conjunctive transitions of \mathcal{S} correspond to disjunctive transitions of \mathcal{R} and vice versa. By slightly rephrasing the separation condition, we have $L(\mathcal{A}) \perp L(\mathcal{R})$ and $L(\mathcal{B}) \perp L(\mathcal{S})$. This means that **Pathfinder** wins both disjointness games $G^{\text{dis}}(\mathcal{R}, \mathcal{A})$ and $G^{\text{dis}}(\mathcal{S}, \mathcal{B})$. Thus, we can apply Lemma 2.10 to obtain pathfinders $\mathcal{P}_{\mathcal{A}}: (\bigcup_{a \in \Sigma} \Delta^{\mathcal{R}}(a) \times \Delta^{\mathcal{A}}(a)) \rightarrow \{\mathbf{L}, \mathbf{R}\}$ and $\mathcal{P}_{\mathcal{B}}: (\bigcup_{a \in \Sigma} \Delta^{\mathcal{S}}(a) \times \Delta^{\mathcal{B}}(a)) \rightarrow \{\mathbf{L}, \mathbf{R}\}$.

We will now provide a strategy of **Separator** in G . The constructed strategy uses as its memory states the set of states of \mathcal{S} that are distinct than \top . Let the initial memory state be q_0 . Assume that the current memory state is q_i and consider the i -th round of the game.

[I: a] Input plays an arbitrary letter $a_i \in \Sigma$.

[S: m] **Separator** plays the mode $m_i \in \{\vee, \wedge\}$ defined as follows. We consider the following two cases for the mode of the transitions $\Delta^{\mathcal{S}}(q_i, a_i)$.

1. If $\Delta^{\mathcal{S}}(q_i, a_i) = \{\delta_i^{\mathcal{S}}\}$ is a single conjunctive transition $\delta_i^{\mathcal{S}} = (q_i, a_i, q_{\mathcal{L},i}, q_{\mathcal{R},i})$ then we put $m_i := \wedge$ and $f_i := \mathcal{P}_{\mathcal{B}}(\delta_i^{\mathcal{S}}, -)$ is a selector for \mathcal{B} .
2. Otherwise, $\Delta^{\mathcal{S}}(q_i, a_i)$ is a pair of disjunctive transitions which means that $\Delta^{\mathcal{R}}(q_i, a_i)$ is a single conjunctive transition $\delta_i^{\mathcal{R}} = (q_i, a_i, q_{\mathcal{L},i}, q_{\mathcal{R},i})$. In this case we put $m_i := \vee$ and $f_i := \mathcal{P}_{\mathcal{A}}(\delta_i^{\mathcal{R}}, -)$ is a selector for \mathcal{A} .

[S: f] Separator plays the selector f_i defined above (notice that f_i is either a selector for \mathcal{A} or for \mathcal{B} , according to m_i).

[I: d] Input plays an arbitrary direction $d_i \in \{\mathcal{L}, \mathcal{R}\}$.

The next memory state of our strategy is the state $q_{d_i,i}$ taken from one of the transitions $\delta_i^{\mathcal{S}}$ or $\delta_i^{\mathcal{R}}$, see above. We now argue that Separator wins the corresponding infinite play $\pi = (a_0, m_0, f_0, d_0)(a_1, m_1, f_1, d_1) \cdots$. Let $b = (a_0, d_0)(a_1, d_1) \cdots$ be the corresponding path. Consider a number $i \in \omega$. By the construction of the strategy above, we have two cases:

1. If $m_i = \wedge$, then a conjunctive transition $\delta_i^{\mathcal{S}} = (q_i, a_i, q_{\mathcal{L},i}, q_{\mathcal{R},i})$ of \mathcal{S} was used to determine f_i . In this case, define $\delta_i^{\mathcal{R}}$ as the following disjunctive transition of \mathcal{R} : if $d_i = \mathcal{L}$ then $\delta_i^{\mathcal{R}} := (q_i, a_i, q_{\mathcal{L},i}, \top)$, otherwise $d_i = \mathcal{R}$ and $\delta_i^{\mathcal{R}} := (q_i, a_i, \top, q_{\mathcal{R},i})$.
2. If $m_i = \vee$, then a conjunctive transition $\delta_i^{\mathcal{R}} = (q_i, a_i, q_{\mathcal{L},i}, q_{\mathcal{R},i})$ of \mathcal{R} was used to determine f_i . In this case, define $\delta_i^{\mathcal{S}}$ as the following disjunctive transition of \mathcal{S} : if $d_i = \mathcal{L}$ then $\delta_i^{\mathcal{S}} := (q_i, a_i, q_{\mathcal{L},i}, \top)$, otherwise $d_i = \mathcal{R}$ and $\delta_i^{\mathcal{S}} := (q_i, a_i, \top, q_{\mathcal{R},i})$.

The definitions above provide two sequences of transitions $\vec{\delta}^{\mathcal{S}} := \delta_0^{\mathcal{S}} \delta_1^{\mathcal{S}} \cdots \in \Delta^{\mathcal{S}}(b)$, $\vec{\delta}^{\mathcal{R}} := \delta_0^{\mathcal{R}} \delta_1^{\mathcal{R}} \cdots \in \Delta^{\mathcal{R}}(b)$. Since for every $i \in \omega$ the transitions $\delta_i^{\mathcal{S}}$ and $\delta_i^{\mathcal{R}}$ are from the same state $q_i \neq \top$, $\vec{\delta}^{\mathcal{S}}$ is accepting in \mathcal{S} if, and only if, $\vec{\delta}^{\mathcal{R}}$ is rejecting in \mathcal{R} . Assume that $\vec{\delta}^{\mathcal{S}}$ is accepting (the other case is analogous). We will show that $\mathbf{W}_{\mathcal{B}}$ is violated (if $\vec{\delta}^{\mathcal{R}}$ is accepting then $\mathbf{W}_{\mathcal{A}}$ is violated). Assume for the sake of contradiction that $\mathbf{W}_{\mathcal{B}}$ holds, as witnessed by a sequence of \mathcal{B} -transitions $\vec{\delta}^{\mathcal{B}} = \delta_0^{\mathcal{B}} \delta_1^{\mathcal{B}} \cdots \in \Delta^{\mathcal{B}}(b)$. By Remark 2.12 we obtain that whenever $m_i = \vee$ and $\delta_i^{\mathcal{S}}$ is a disjunctive transition of \mathcal{S} then $\mathcal{P}_{\mathcal{B}}(\delta_i^{\mathcal{S}}, -)$ is constantly equal to d_i . By the assumption on $\vec{\delta}^{\mathcal{B}}$ from $\mathbf{W}_{\mathcal{B}}$ we know that whenever $m_i = \wedge$ then $f_i(\delta_i^{\mathcal{B}}) = d_i$. However, if $m_i = \wedge$ then $f_i(\delta_i^{\mathcal{B}}) = \mathcal{P}_{\mathcal{B}}(\delta_i^{\mathcal{S}}, \delta_i^{\mathcal{B}})$. Therefore, in both cases we know that $\mathcal{P}_{\mathcal{B}}(\delta_i^{\mathcal{S}}, \delta_i^{\mathcal{B}}) = d_i$. This means that the assumptions of Corollary 2.11 are met and at least one of the sequences $\vec{\delta}^{\mathcal{S}}$, $\vec{\delta}^{\mathcal{B}}$ is rejecting—a contradiction, since we assumed both these sequences to be accepting. \square

6 Separability by game automata with priorities in C

In this section we present our last game-theoretic characterisation, namely game automata separability for a fixed finite set $C \subseteq \mathbb{N}$ of priorities. Fix two automata $\mathcal{A} = (\Sigma, Q^{\mathcal{A}}, q_0^{\mathcal{A}}, \Omega^{\mathcal{A}}, \Delta^{\mathcal{A}})$ and $\mathcal{B} = (\Sigma, Q^{\mathcal{B}}, q_0^{\mathcal{B}}, \Omega^{\mathcal{B}}, \Delta^{\mathcal{B}})$ over the same alphabet Σ . The game is a variation of $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$ from Section 5 where Separator additionally plays priorities from C .

C -game-automata separability game $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$

At the i -th round:

[S: c] Separator plays a priority $c_i \in C$.

[I: a] Input plays a letter $a_i \in \Sigma$.

[S: m] Separator plays a mode $m_i \in \{\vee, \wedge\}$.

[S: f] Separator plays either

1. a selector $f_i \in \{\text{L}, \text{R}\}^{\Delta^{\mathcal{A}}(a_i)}$ for \mathcal{A} if $m_i = \vee$, or
2. a selector $f_i \in \{\text{L}, \text{R}\}^{\Delta^{\mathcal{B}}(a_i)}$ for \mathcal{B} if $m_i = \wedge$.

[I: d] Input plays a direction $d_i \in \{\text{L}, \text{R}\}$.

Separator wins an infinite play $\pi = (c_0, a_0, m_0, f_0, d_0)(c_1, a_1, m_1, f_1, d_1) \cdots$ inducing a path $b = (a_0, d_0)(a_1, d_1) \cdots$ whenever both conditions below hold:

- $\pi \in \mathbf{W}_{\mathcal{A}}$: If there exists an accepting sequence of transitions $\vec{\delta}^{\mathcal{A}} = \delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \cdots \in \Delta^{\mathcal{A}}(b)$ s.t. for all $i \in \omega$ we have $(m_i = \vee) \Rightarrow f_i(\delta_i^{\mathcal{A}}) = d_i$, then $c_0 c_1 \cdots$ is accepting.
- $\pi \in \mathbf{W}_{\mathcal{B}}$: If there exists an accepting sequence of transitions $\vec{\delta}^{\mathcal{B}} = \delta_0^{\mathcal{B}} \delta_1^{\mathcal{B}} \cdots \in \Delta^{\mathcal{B}}(b)$ s.t. for all $i \in \omega$ we have $(m_i = \wedge) \Rightarrow f_i(\delta_i^{\mathcal{B}}) = d_i$, then $c_0 c_1 \cdots$ is rejecting.

Lemma 6.1. *Separator wins $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$ if, and only if, there exists a game automaton \mathcal{S} with priorities in C separating $\text{L}(\mathcal{A})$, $\text{L}(\mathcal{B})$.*

The proof of this lemma can be seen as a simplified variant of the proof of Lemma 5.1, except for the acceptance condition of the separator which is given by the priorities c_i 's as in the proof of Lemma 3.1. A strategy for Separator in $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$ is a tuple

$$\mathcal{M} = (M, \ell_0, (\bar{c}, \bar{m}, \bar{f}), \tau) \quad (3)$$

where M is a set of memory states, $\ell_0 \in M$ is the initial memory state, $\bar{c}, \bar{m}, \bar{f}$ are decision functions, and $\tau : M \times \Sigma \times \{\text{L}, \text{R}\} \rightarrow M$ is the memory update function. More precisely, $\bar{c} : M \rightarrow C$ outputs a priority $\bar{c}(\ell)$ in position ℓ , $\bar{m} : M \times \Sigma \rightarrow \{\wedge, \vee\}$ outputs a mode $\bar{m}(\ell, a)$ in position ℓ when Input plays $a \in \Sigma$, and $\bar{f} : M \times \Sigma \rightarrow \bigcup_{a \in \Sigma} (\{\text{L}, \text{R}\}^{\Delta^{\mathcal{A}}(a)} \cup \{\text{L}, \text{R}\}^{\Delta^{\mathcal{B}}(a)})$ outputs a selector $\bar{f}(\ell, a)$ in similar circumstances. With this notation we can define a correspondence α from finite-memory winning strategies for Separator in $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$ to game automata separating \mathcal{A} , \mathcal{B} with priorities in C . More precisely, we map an arbitrary finite-memory strategy \mathcal{M} to a game automaton

$$\mathcal{S} := \alpha(\mathcal{M}) := (\Sigma, M, \ell_0, \Delta^{\mathcal{S}}, \Omega^{\mathcal{S}}) \quad (4)$$

which has the same set of states M and initial state ℓ_0 as \mathcal{M} , priorities are induced by the decision function \bar{c} of \mathcal{M} as

$$\Omega^{\mathcal{S}}(\ell) := \bar{c}(\ell),$$

and transitions are of the form

$$\Delta^{\mathcal{S}}(\ell, a) = \begin{cases} \{(\ell, a, \ell_{\mathbf{L}}, \top), (\ell, a, \top, \ell_{\mathbf{R}})\} & \text{if } \overline{m}(\ell, a) = \vee, \\ \{(\ell, a, \ell_{\mathbf{L}}, \ell_{\mathbf{R}})\} & \text{if } \overline{m}(\ell, a) = \wedge, \end{cases}$$

where $\ell_{\mathbf{L}} := \tau(\ell, a, \mathbf{L})$ and $\ell_{\mathbf{R}} := \tau(\ell, a, \mathbf{R})$. Notice how the acceptance condition of \mathcal{S} is simply inherited from the winning strategy \mathcal{M} . This should be contrasted with Section 5 where the set of priorities C is not fixed beforehand, and thus the acceptance condition of \mathcal{S} is defined with the help of the winning condition for **Separator** in the corresponding separability game. The decision function \overline{f} is not involved in the definition of $\alpha(\mathcal{M})$, however it is used to show that if \mathcal{M} is winning, then $\alpha(\mathcal{M})$ is in fact a separator.

In the following, let $G := G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$ be the C -game-separability game. The proof below is very similar to the one of Lemma 5.1, with some adaptations to take care of the additional priorities c_i 's selected by **Separator**.

Soundness. Assume that **Separator** wins the separability game G . By Theorem 2.6, there exists a finite-memory winning strategy \mathcal{M} as in (3). Let $\mathcal{S} = \alpha(\mathcal{M})$ be the game automaton corresponding to \mathcal{M} . We show that \mathcal{S} separates $\mathbf{L}(\mathcal{A})$ from $\mathbf{L}(\mathcal{B})$, i.e., $\mathbf{L}(\mathcal{A}) \subseteq \mathbf{L}(\mathcal{S})$ and $\mathbf{L}(\mathcal{S}) \perp \mathbf{L}(\mathcal{B})$. First, we show that $\mathbf{L}(\mathcal{A}) \subseteq \mathbf{L}(\mathcal{S})$. To this end, assume that $t \in \mathbf{L}(\mathcal{A})$ and let $\rho^{\mathcal{A}}$ be an accepting run of \mathcal{A} over t witnessing this. We show that **Automaton** wins the acceptance game $G_{\mathcal{S}} := G^{\text{acc}}(\mathcal{S}, t)$. To show this we play in parallel the separability game G and the acceptance game $G_{\mathcal{S}}$. As we play both games in lock-steps, we maintain the following invariant: At every round i , the current finite path of the input tree t is $(a_0, d_0) \cdots (a_{i-1}, d_{i-1})$, **Separator**'s winning strategy \mathcal{M} in the separability game G is in memory state ℓ_i , the current state of the separating automaton \mathcal{S} in the acceptance game $G_{\mathcal{S}}$ is ℓ_i as well, and $\rho^{\mathcal{A}}(d_0 \cdots d_{i-1}) = q_i^{\mathcal{A}}$. Let now be at round i and assume that the invariant holds. We play the separability and acceptance games as follows.

G .[S: c] **Separator** plays the priority $c_i := \overline{c}(\ell_i) \in C$.

G .[I: a] **Input** plays the letter $a_i := t(u_i)$ for $u_i := d_0 \cdots d_{i-1}$.

G .[S: m] **Separator** plays the mode $m_i := \overline{m}(\ell_i, a_i) \in \{\vee, \wedge\}$.

G .[S: f] **Separator** plays either

1. the selector $f_i := \overline{f}(\ell_i, a_i) \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{A}}(a_i)}$ for \mathcal{A} if $m_i = \vee$ or
2. the selector $f_i := \overline{f}(\ell_i, a_i) \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{B}}(a_i)}$ for \mathcal{B} if $m_i = \wedge$.

$G_{\mathcal{S}}$.[A: δ] **Automaton** plays the transition $\delta_i^{\mathcal{S}} \in \Delta^{\mathcal{S}}(\ell_i, a_i)$, defined as follows. Let $\delta_i^{\mathcal{A}} := (\rho^{\mathcal{A}}(u_i), t(u_i), \rho^{\mathcal{A}}(u_i \mathbf{L}), \rho^{\mathcal{A}}(u_i \mathbf{R}))$ be the \mathcal{A} -transition used in u_i by the run $\rho^{\mathcal{A}}$. We distinguish two cases.

1. In the first case, assume that **Separator** played $m_i = \vee$ and $f_i \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{A}}(a_i)}$. It means that $\Delta^{\mathcal{S}}((\ell_i, q_i), a_i)$ contains two disjunctive transitions, $\delta_{\mathbf{L}, i}^{\mathcal{S}} := (\ell_i, a_i, \ell_{\mathbf{L}, i}, \top)$ and $\delta_{\mathbf{R}, i}^{\mathcal{S}} := (\ell_i, a_i, \top, \ell_{\mathbf{R}, i})$. Let us put $\delta_i^{\mathcal{S}} := \delta_{f_i(\delta_i^{\mathcal{A}}), i}^{\mathcal{S}}$, i.e., the transition that sends a non- \top state in the direction given by $f_i(\delta_i^{\mathcal{A}})$.

2. In the second case, **Separator** played $m_i = \wedge$ and $f_i \in \{\mathbf{L}, \mathbf{R}\}^{\Delta^{\mathcal{B}}(a_i)}$. It means that $\Delta^{\mathcal{S}}(\ell_i, a_i)$ contains one conjunctive transition $\delta_i^{\mathcal{S}} := (\ell_i, a_i, \ell_{\mathbf{L},i}, \ell_{\mathbf{R},i})$.

$G_{\mathcal{S}}.[! : d]$ Input plays an arbitrary direction $d_i \in \{\mathbf{L}, \mathbf{R}\}$.

$G.[! : d]$ Input plays the direction $d_i \in \{\mathbf{L}, \mathbf{R}\}$.

Notice that if $m_i = \vee$ and $d_i \neq f^{\vee}(\delta_i^{\mathcal{A}})$ then the next position of the acceptance game $G_{\mathcal{S}}$ is $(u_i d_i, \top)$, which is a winning position for **Automaton**. Therefore, without loss of generality we can assume that

$$\forall i \in \omega. (m_i = \vee) \Rightarrow f^{\vee}(\delta_i^{\mathcal{A}}) = d_i. \quad (5)$$

Moreover, the new state of \mathcal{S} in $G_{\mathcal{S}}$ is ℓ_{i+1} for $\ell_{i+1} := \tau(\ell_i, a_i, d_i)$. Similarly, the new memory state of \mathcal{M} in G is ℓ_{i+1} . This concludes the description of round i of both games. We argue that **Automaton** wins the resulting infinite play $(\delta_0^{\mathcal{S}}, d_0)(\delta_1^{\mathcal{S}}, d_1) \cdots$ of the acceptance game $G_{\mathcal{S}}$. Consider the infinite play $\pi = (c_0, a_0, m_0, f_0, d_0)(c_1, a_1, m_1, f_1, d_1) \cdots$ of the separability game G . Let $b := (a_0, d_0)(a_1, d_1) \cdots$ be the induced path. Since we used the winning strategy of **Separator**, this play satisfies $\mathbf{W}_{\mathcal{A}}$. Since the run $\rho^{\mathcal{A}}$ is accepting, the infinite sequence of \mathcal{A} 's transitions $\delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \cdots$ is accepting. Additionally, (5) holds. Therefore, the sequence of priorities $c_0 c_1 \cdots$ must be accepting by $\mathbf{W}_{\mathcal{A}}$. However, by the definition of the automaton \mathcal{S} , we know that $\Omega^{\mathcal{S}}(\ell_i) = c_i$, which means that **Automaton** wins the considered play of the acceptance game $G_{\mathcal{S}}$.

It remains to prove that $L(\mathcal{S}) \perp L(\mathcal{B})$. The latter is equivalent to $L(\mathcal{B}) \subseteq L(\mathcal{S}^c)$, where the dual game automaton \mathcal{S}^c recognises the complement language of \mathcal{S} . By the construction, \mathcal{S}^c has the same states as \mathcal{S} , and transitions are defined by exchanging the conjunctive and disjunctive ones. Moreover, the priorities in \mathcal{S}^c can be chosen as $\Omega^{\mathcal{S}^c}(\ell) = \Omega^{\mathcal{S}}(\ell) + 1$, and thus a sequence of priorities $\Omega^{\mathcal{S}}(\ell_0) \Omega^{\mathcal{S}}(\ell_1) \cdots$ in \mathcal{S} is rejecting if, and only if, the corresponding sequence $\Omega^{\mathcal{S}^c}(\ell_0) \Omega^{\mathcal{S}^c}(\ell_1) \cdots$ in \mathcal{S}^c is accepting. With these observations in hand, we can conclude by repeating the argument in the first part of the proof above with \mathcal{A} replaced by \mathcal{B} , \mathcal{S} replaced by \mathcal{S}^c , and condition $\mathbf{W}_{\mathcal{A}}$ replaced by $\mathbf{W}_{\mathcal{B}}$. \square

Completeness. Assume that \mathcal{S} is a game automaton with priorities in C separating $L(\mathcal{A})$ from $L(\mathcal{B})$, and we show that **Separator** wins the separability game $G := G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$. Let $\mathcal{R} := \mathcal{S}^c$ be the dual game automaton recognising the complement language $L(\mathcal{R}) = \text{Tr}_{\Sigma} \setminus L(\mathcal{S})$.

Since \mathcal{S} is a separator, we have that $L(\mathcal{S}) \perp L(\mathcal{B})$ and $L(\mathcal{R}) \perp L(\mathcal{A})$, which means that **Pathfinder** wins both disjointness games $G^{\text{dis}}(\mathcal{S}, \mathcal{B})$ and $G^{\text{dis}}(\mathcal{R}, \mathcal{A})$. Let

$$\begin{aligned} \mathcal{P}_{\mathcal{B}} &: \left(\bigcup_{a \in \Sigma} \Delta^{\mathcal{S}}(a) \times \Delta^{\mathcal{B}}(a) \right) \rightarrow \{\mathbf{L}, \mathbf{R}\}, \\ \mathcal{P}_{\mathcal{A}} &: \left(\bigcup_{a \in \Sigma} \Delta^{\mathcal{R}}(a) \times \Delta^{\mathcal{A}}(a) \right) \rightarrow \{\mathbf{L}, \mathbf{R}\}, \end{aligned}$$

be two pathfinders witnessing this.

We will now provide a strategy of **Separator** in G . The constructed strategy uses as its memory states the set of states of \mathcal{S} that are distinct than \top . Let

the initial memory state be q_0 . Assume that the current memory state is q_i and consider the i -th round of the game.

[S: m] Separator plays the priority $c_i := \Omega^S(\ell_i) \in \{\vee, \wedge\}$.

[I: a] Input plays an arbitrary letter $a_i \in \Sigma$.

[S: m] Separator plays the mode $m_i \in \{\vee, \wedge\}$ defined as follows. We consider the following two cases for the mode of the transitions $\Delta^S(q_i, a_i)$.

1. If $\Delta^S(q_i, a_i) = \{\delta_i^S\}$ is a single conjunctive transition $\delta_i^S = (q_i, a_i, q_{L,i}, q_{R,i})$ then we put $m_i := \wedge$ and $f_i := \mathcal{P}_{\mathcal{B}}(\delta_i^S, _)$ is a selector for \mathcal{B} .
2. Otherwise, $\Delta^S(q_i, a_i)$ is a pair of disjunctive transitions which means that $\Delta^{\mathcal{R}}(q_i, a_i)$ is a single conjunctive transition $\delta_i^{\mathcal{R}} = (q_i, a_i, q_{L,i}, q_{R,i})$. In this case we put $m_i := \vee$ and $f_i := \mathcal{P}_{\mathcal{A}}(\delta_i^{\mathcal{R}}, _)$ is a selector for \mathcal{A} .

[S: f] Separator plays the selector f_i defined above (notice that f_i is either a selector for \mathcal{A} or for \mathcal{B} , according to m_i).

[I: d] Input plays an arbitrary direction $d_i \in \{\mathsf{L}, \mathsf{R}\}$.

The next memory state of our strategy is the state $q_{d_i, i}$ taken from one of the transitions δ_i^S or $\delta_i^{\mathcal{R}}$, see above.

We now argue that Separator wins the corresponding infinite play $\pi = (a_0, m_0, f_0, d_0)(a_1, m_1, f_1, d_1) \cdots$. Let $b = (a_0, d_0)(a_1, d_1) \cdots$ be the corresponding path.

We begin by showing that $\pi \in \mathbf{W}_{\mathcal{B}}$. Let

$$\bar{\delta}^{\mathcal{B}} = \delta_0^{\mathcal{B}} \delta_1^{\mathcal{B}} \cdots \in \Delta^{\mathcal{B}}(b) \quad (6)$$

be an infinite accepting sequence of transitions over the branch b conform to π_0 , where $\delta_i^{\mathcal{B}}$ has the form $\delta_i^{\mathcal{B}} = (q_i^{\mathcal{B}}, a_i, q_{L,i}^{\mathcal{B}}, q_{R,i}^{\mathcal{B}})$. We need to show that $c_0 c_1 \cdots$ is rejecting.

Consider a number $i \in \omega$. By the construction of the strategy of Separator above, we know that there are two cases:

1. If $m_i = \wedge$, then a conjunctive transition $\delta_i^S = (q_i, a_i, q_{L,i}, q_{R,i})$ of \mathcal{S} was used to determine f_i . In this case, define $\delta_i^{\mathcal{R}}$ as the following disjunctive transition of \mathcal{R} : If $d_i = \mathsf{L}$, then $\delta_i^{\mathcal{R}} := (q_i, a_i, q_{L,i}, \top)$, otherwise $d_i = \mathsf{R}$ and $\delta_i^{\mathcal{R}} := (q_i, a_i, \top, q_{R,i})$.
2. If $m_i = \vee$, then a conjunctive transition $\delta_i^{\mathcal{R}} = (q_i, a_i, q_{L,i}, q_{R,i})$ of \mathcal{R} was used to determine f_i . In this case, define δ_i^S as the following disjunctive transition of \mathcal{S} : If $d_i = \mathsf{L}$, then $\delta_i^S := (q_i, a_i, q_{L,i}, \top)$, otherwise $d_i = \mathsf{R}$ and $\delta_i^S := (q_i, a_i, \top, q_{R,i})$.

The definitions above provide two sequences of transitions: $\bar{\delta}^{\mathcal{S}} := \delta_0^{\mathcal{S}} \delta_1^{\mathcal{S}} \cdots \in (\Delta^{\mathcal{S}})^{\omega}$ and $\bar{\delta}^{\mathcal{R}} := \delta_0^{\mathcal{R}} \delta_1^{\mathcal{R}} \cdots \in (\Delta^{\mathcal{R}})^{\omega}$. Notice that the construction guarantees that $\bar{\delta}^{\mathcal{S}} \in \Delta^{\mathcal{S}}(b)$ and $\bar{\delta}^{\mathcal{R}} \in \Delta^{\mathcal{R}}(b)$.

By Remark 2.12 we obtain that if $m_i = \vee$ and δ_i^S is a disjunctive transition of \mathcal{S} , then $\mathcal{P}_{\mathcal{B}}(\delta_i^S, _)$ is constantly equal d_i (the direction in which δ_i^S sends the state different than \top). By the assumption on $\bar{\delta}^{\mathcal{B}}$ from $\mathbf{W}_{\mathcal{B}}$ we know that if

$m_i = \wedge$, then $f_i(\delta_i^{\mathcal{B}}) = d_i$. However, if $m_i = \vee$, then $f_i(\delta_i^{\mathcal{B}}) = \mathcal{P}_{\mathcal{B}}(\delta_i^{\mathcal{S}}, \delta_i^{\mathcal{B}})$. Therefore, in both cases we know that $\mathcal{P}_{\mathcal{B}}(\delta_i^{\mathcal{S}}, \delta_i^{\mathcal{B}}) = d_i$.

This means that the assumptions of Corollary 2.11 are met and at least one of the sequences $\vec{\delta}^{\mathcal{S}}, \vec{\delta}^{\mathcal{B}}$ is rejecting. Since we assumed that $\vec{\delta}^{\mathcal{B}}$ is accepting, $\vec{\delta}^{\mathcal{S}}$ must be rejecting. But the priorities $c_0 c_1 \dots$ are just the priorities of the transitions $\delta_i^{\mathcal{S}}$, so $c_0 c_1 \dots$ is rejecting.

The case of $\mathbf{W}_{\mathcal{A}}$ is entirely dual: we consider a sequence of transitions $\vec{\delta}^{\mathcal{A}} = \delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \dots \in \Delta^{\mathcal{A}}(b)$ that is accepting and use Corollary 2.11 for $\mathcal{P}_{\mathcal{A}}$ to show that $\vec{\delta}^{\mathcal{R}}$ must be rejecting, which implies that $c_0 c_1 \dots$ is accepting. \square

7 Complexity

In this section we perform a detailed analysis of the complexity of solving the separability problems from Sections 3 to 6 and the complexity of separators, thus proving Theorem 1.1 announced in the introduction:

Theorem 1.1. *The deterministic and game separability problems can be solved in EXPTIME, both for a fixed finite index $C \subseteq \mathbb{N}$, and an unrestricted one $C = \mathbb{N}$. Moreover, separators with exponentially many states and polynomially many priorities suffice.*

In each case it will be a matter of constructing a deterministic parity automaton \mathcal{W} over ω -words recognising the set of winning plays and then solving a suitable parity game. In the following, let $M = \{\vee, \wedge\}$ be the set of alternation modes, and let $D = \{\mathbf{L}, \mathbf{R}\}$ be the set of directions.

7.1 Separability by deterministic automata

In this section we perform a complexity analysis for Section 4. Let $\mathcal{A} = (\Sigma, Q^{\mathcal{A}}, q_0^{\mathcal{A}}, \Omega^{\mathcal{A}}, \Delta^{\mathcal{A}})$ and recall that $\mathbf{W}_{\mathcal{A}}$ is the set of plays $\pi = (a_0, f_0, d_0)(a_1, f_1, d_1) \dots$ with branch $b = (a_0, d_0)(a_1, d_1) \dots$ s.t. there is an accepting sequence of transitions $\vec{\delta}^{\mathcal{A}} \in \Delta^{\mathcal{A}}(b)$. The language $\mathbf{W}_{\mathcal{A}}$ can be recognised by a nondeterministic ω -word parity automaton $\mathcal{W}_{\mathcal{A}}$ over the alphabet

$$\Sigma' = \Sigma \times \left(\bigcup_{a \in \Sigma} D^{\Delta^{\mathcal{B}}(a)} \right) \times D. \quad (7)$$

which reads π , nondeterministically guesses the sequence of transitions $\vec{\delta}^{\mathcal{A}}$, and verifies that it is accepting. (Notice that Σ' has size exponential in the size of \mathcal{A} .) More precisely, we can take $\mathcal{W}_{\mathcal{A}} = (\Sigma', Q^{\mathcal{A}}, q_0^{\mathcal{A}}, \Omega^{\mathcal{A}}, \Delta^{\mathcal{W}_{\mathcal{A}}})$ to have the same states $Q^{\mathcal{A}}$, initial state $q_0^{\mathcal{A}}$, and priority function $\Omega^{\mathcal{A}}$ as \mathcal{A} , and set of transitions

$$\Delta^{\mathcal{W}_{\mathcal{A}}} = \{(q, a', q_d) \mid \text{for some } a' = (a, f, d) \in \Sigma' \text{ and } \delta^{\mathcal{A}} = (q, a, q_{\mathbf{L}}, q_{\mathbf{R}}) \in \Delta^{\mathcal{A}}\}.$$

It is immediate to verify that $\mathbf{W}_{\mathcal{A}} = \mathbf{L}(\mathcal{W}_{\mathcal{A}})$. Let $\mathcal{B} = (\Sigma, Q^{\mathcal{B}}, q_0^{\mathcal{B}}, \Omega^{\mathcal{B}}, \Delta^{\mathcal{B}})$ and recall that $\mathbf{W}_{\mathcal{B}}$ is the set of plays π with branch b as above s.t. there is an accepting sequence of transitions $\vec{\delta}^{\mathcal{B}} = \delta_0^{\mathcal{B}} \delta_1^{\mathcal{B}} \dots \in \Delta^{\mathcal{A}}(b)$ s.t., for all $i \in \mathbb{N}$, $f_i(\delta_i^{\mathcal{A}}) = d_i$. As above, the language $\mathbf{W}_{\mathcal{B}} = \mathbf{L}(\mathcal{W}_{\mathcal{B}})$ can be recognised by a nondeterministic ω -word parity automaton $\mathcal{W}_{\mathcal{B}} = (\Sigma', Q^{\mathcal{B}}, q_0^{\mathcal{B}}, \Omega^{\mathcal{B}}, \Delta^{\mathcal{W}_{\mathcal{B}}})$ where

$$\Delta^{\mathcal{W}_{\mathcal{B}}} = \left\{ (q, a', q_d) \mid \begin{array}{l} \text{for some } a' = (a, f, d) \in \Sigma' \text{ and } \delta^{\mathcal{B}} = (q, a, q_{\mathbf{L}}, q_{\mathbf{R}}) \in \Delta^{\mathcal{B}} \\ \text{s.t. } f(\delta^{\mathcal{B}}) = d. \end{array} \right\}$$

Putting the two constructions above together, **Input**'s winning condition $\mathbf{W}_{\text{Input}} = \mathbf{W}_{\mathcal{A}} \cap \mathbf{W}_{\mathcal{B}}$ can be recognised by a nondeterministic ω -word parity automaton of size polynomial in \mathcal{A}, \mathcal{B} , and thus by Lemma 2.2 by a deterministic ω -word parity automaton $\mathcal{W}_{\text{Input}}$ of exponential size and polynomially many priorities. By applying Lemma 2.8 and the characterisation of Lemma 4.3 we can thus solve the deterministic separability problem in EXPTIME. Thanks to the implication “3 \Rightarrow 2” of Lemma 4.3, if a deterministic separator exists, then the path closure automaton $\mathcal{A}^{\text{path}}$ is a deterministic separator. By inspecting the construction of $\mathcal{A}^{\text{path}}$, one can see that it has number of states exponential in that of \mathcal{A} , and the same set of priorities as \mathcal{A} . This discussion is summarised in the following result.

Theorem 7.1. *The deterministic separability problem can be solved in EXPTIME. Moreover, when a deterministic separator exists, there is one with exponentially many states and polynomially many priorities.*

7.2 Separability by deterministic automata with priorities in C

In this section we perform a complexity analysis for Section 3. We build a nondeterministic automaton $\mathcal{W}_{\mathcal{A}} = (\Sigma'', Q, q_0, \Omega, \Delta)$ recognising the set of plays $L(\mathcal{W}_{\mathcal{A}})$ not satisfying $\mathbf{W}_{\mathcal{A}}$. Automaton $\mathcal{W}_{\mathcal{A}}$ is over the alphabet

$$\Sigma'' = C \times \Sigma', \quad (8)$$

with Σ' from (7). Intuitively, $\mathcal{W}_{\mathcal{A}}$ accepts an infinite play $\pi = (c_0, a_0, f_0, d_0)(c_1, a_1, f_1, d_1) \cdots$ with path $b = (a_0, d_0)(a_1, d_1) \cdots$ whenever there exists an accepting sequence of transitions $\vec{\delta}^{\mathcal{A}} = \delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \cdots \in \Delta^{\mathcal{A}}(b)$ and $c_0 c_1 \cdots$ is rejecting. In order to achieve this, $\mathcal{W}_{\mathcal{A}}$ guesses an accepting sequence of transitions from \mathcal{A} (as in Section 7.1) and also guesses an odd priority $c \in C$ and verifies that it occurs infinitely often, and that no larger priority occurs infinitely often. This can be achieved by a set of states Q of size polynomial in \mathcal{A} . Note that the input alphabet Σ'' has exponential size in \mathcal{A}, \mathcal{B} (due to the selectors f_i 's), and thus $\mathcal{W}_{\mathcal{A}}$ will have exponentially many transitions. A very similar construction yields a nondeterministic parity ω -word automaton $\mathcal{W}_{\mathcal{B}}$ over the same action alphabet Σ'' from (8) recognising the set of plays $L(\mathcal{W}_{\mathcal{B}})$ not in $\mathbf{W}_{\mathcal{B}}$ with polynomially many states and exponentially many transitions. It follows that the complement of $\mathbf{W}_{\mathcal{A}} \cap \mathbf{W}_{\mathcal{B}}$ can be recognised by a nondeterministic parity ω -word automaton \mathcal{W} of the same complexity. By Lemma 2.2 we can further convert \mathcal{W} to an equivalent deterministic parity automaton \mathcal{W}' with exponentially many states and polynomially many priorities (w.r.t. the number of states of \mathcal{A}, \mathcal{B}). By Lemma 2.8 we can thus solve $G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$ in EXPTIME, and by the characterisation in Lemma 3.1 we can solve the C -deterministic separability problem within the same complexity.

Based on the size of the winning condition \mathcal{W}' and the strong connection between winning strategies for **Separator** and deterministic separators in the “soundness” direction of the proof of Lemma 3.1, we can also provide an upper bound on the size of a separating deterministic automaton, when it exists. More precisely, if **Separator** wins the C -deterministic-separability game $G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$, then she has a positional winning strategy in the corresponding graph game of exponential size from Lemma 2.8. This means that **Separator** has a winning

strategy \mathcal{M} of exponential memory in $G_{\text{det}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$. This strategy is then translated to a separating deterministic automaton \mathcal{S} with exponentially many states and priorities in C . Putting these considerations together gives the following complexity result.

Theorem 7.2. *The C -deterministic separability problem is in EXPTIME. Moreover, deterministic separators of exponential size suffice.*

7.3 Separability by game automata

In this section we perform a complexity analysis for Section 5. Let $\mathcal{A} = (\Sigma, P, p_0, \Omega, \Delta^{\mathcal{A}})$ and recall that $\mathbf{W}_{\mathcal{A}}$ is the set of plays of the form $\pi = (a_0, m_0, f_0, d_0)(a_1, m_1, f_1, d_1) \cdots$ with branch $b = (a_0, d_0)(a_1, d_1) \cdots$ s.t. there is an accepting sequence of transitions $\vec{\delta}^{\mathcal{A}} = \delta_0^{\mathcal{A}} \delta_1^{\mathcal{A}} \cdots \in \Delta^{\mathcal{A}}(b)$ s.t., for all $i \in \mathbb{N}$, (\dagger) if $m_i = \vee$ then $f_i(\delta_i^{\mathcal{A}}) = d_i$. The language $\mathbf{W}_{\mathcal{A}}$ can be recognised by a nondeterministic ω -word parity automaton $\mathcal{W}_{\mathcal{A}}$ over the alphabet

$$\Sigma' = \Sigma \times M \times \left(\bigcup_{a \in \Sigma} D^{\Delta^{\mathcal{A}}(a)} \cup D^{\Delta^{\mathcal{B}}(a)} \right) \times D \quad (9)$$

which reads π , nondeterministically guesses the sequence of transitions $\vec{\delta}^{\mathcal{A}}$, and verifies that it is accepting and that (\dagger) defined above holds. (Notice that Σ' has size exponential in the size of \mathcal{A} .) More precisely, we can take $\mathcal{W}_{\mathcal{A}} = (\Sigma', P, p_0, \Omega, \Delta)$ to have the same states P , initial state p_0 , and priority function Ω as \mathcal{A} , and set of transitions

$$\begin{aligned} \Delta = \{ (q, a', p_d) \mid \text{for some } a' = (a, m, f, d) \in \Sigma' \text{ and } \delta^{\mathcal{A}} = (q, a, p_L, p_R) \in \Delta^{\mathcal{A}} \\ \text{s.t. } (m = \vee) \Rightarrow f(\delta^{\mathcal{A}}) = d \}. \end{aligned}$$

It is immediate to verify that $\mathbf{W}_{\mathcal{A}} = L(\mathcal{W}_{\mathcal{A}})$. With an analogous construction starting from \mathcal{B} we can build a nondeterministic ω -word parity automaton $\mathcal{W}_{\mathcal{B}}$ recognising the set of plays in $\mathbf{W}_{\mathcal{B}} = L(\mathcal{W}_{\mathcal{B}})$. Putting the two together, Input's winning condition $\mathbf{W}_{\text{Input}} = \mathbf{W}_{\mathcal{A}} \cap \mathbf{W}_{\mathcal{B}}$ can be recognised by a nondeterministic ω -word parity automaton of polynomially many states and exponentially many transitions w.r.t. \mathcal{A}, \mathcal{B} , and thus by Lemma 2.2 by a deterministic ω -word parity automaton $\mathcal{W}_{\text{Input}}$ of exponential size and polynomially many priorities. By Lemma 2.8 we can solve such a game in EXPTIME, and thanks to the characterisation from Lemma 5.1, we can solve the game separability problem in EXPTIME.

In fact, we can also provide an upper bound on the number of states and priorities of a separating game automaton (when it exists). Since parity games are memoryless determined and the graph game has exponential size, if Separator wins $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B})$ then she has a winning strategy \mathcal{M} of exponential memory. This means that the separating automaton $\alpha(\mathcal{M}, \mathcal{D})$ with generalised acceptance condition \mathcal{D} has exponential size (ignoring the size of \mathcal{D} for a moment).

We now argue about the size of a suitable deterministic automaton \mathcal{D} for the generalised acceptance condition. First of all, the winning condition $\mathbf{W}_{\mathcal{A}} \subseteq (\Sigma')^{\omega}$ is recognised by the nondeterministic parity automaton $\mathcal{W}_{\mathcal{A}}$ above with the same number of states as \mathcal{A} and exponentially many transitions (since Σ' has exponential size). As suggested in the ‘‘soundness’’ direction of the proof of Lemma 5.1, we take \mathcal{D} to be a deterministic automaton recognising the language

$L_{\mathcal{A}} \subseteq (\Sigma \times D)^\omega$ containing all paths $b = (a_0, d_0)(a_1, d_1) \cdots$ s.t. there exists a play $\pi \in \mathbf{W}_{\mathcal{A}}$ conform to b and **Separator**'s strategy \mathcal{M} . The automaton \mathcal{D} can be obtained as a product construction of $\mathcal{W}_{\mathcal{A}}$ (polynomial) above and **Separator**'s strategy $\mathcal{M} = (L, \ell_0, (\bar{c}, \bar{m}, \bar{f}), \tau)$ (exponential), a projection operation from alphabet Σ' to alphabet $\Sigma \times D$, and then a determinisation operation. More precisely, let

$$\mathcal{D}_0 = (\Sigma \times D, P \times L, (p_0, \ell_0), \Omega_0, \Delta_0)$$

be a nondeterministic parity automaton over alphabet $\Sigma \times D$ where Δ_0 and Ω_0 are defined as follows: $((p, \ell), (a, d), (p', \ell')) \in \Delta_0$ iff $(p, (a, \bar{m}(\ell, a), \bar{f}(\ell, a), d), p') \in \Delta$ and $\tau(\ell, a, d) = \ell'$; $\Omega_0(p, -) = \Omega(p)$. Since \mathcal{M} is winning and by the definition of $\mathcal{W}_{\mathcal{A}}$ we have $L(\mathcal{D})_0 = L_{\mathcal{A}}$. However \mathcal{D}_0 is nondeterministic and a direct determinisation seems to produce a doubly exponential blow-up (since L has exponential size). However, the L -component of the state is in fact a deterministic finite automaton (with no acceptance condition), and since the determinisation operation commutes with products with deterministic finite automata, \mathcal{D}_0 can be determinised into an equivalent deterministic parity automaton \mathcal{D} of exponential size and polynomially many priorities, as required. By Lemma 5.2 applied to the generalised automaton $\alpha(\mathcal{M}, \mathcal{D})$ we can build a game parity automaton \mathcal{S} equivalent to $\alpha(\mathcal{M}, \mathcal{D})$ (and thus separating $L(\mathcal{A}), L(\mathcal{B})$) of exponential size and polynomially many priorities. This discussion is summarised in the following result.

Theorem 7.3. *The game separability problem for can be solved in EXPTIME. Moreover, if a separating game automaton exists, then there is one with exponentially many states and polynomially many priorities.*

7.4 Separability by game automata with priorities in C

In this section we perform a complexity analysis for Section 6. As in Section 7.2 one can build a nondeterministic parity automaton $\mathcal{W}_{\mathcal{A}} = (\Sigma'', Q, q_0, \Omega, \Delta)$ over alphabet $\Sigma'' = C \times \Sigma'$ (where Σ' has been defined in (9)) recognising the set of plays $L(\mathcal{W}_{\mathcal{A}})$ *not* satisfying $\mathbf{W}_{\mathcal{A}}$ with polynomially many states Q and priorities and exponentially many transitions Δ (due to the exponential alphabet Σ''). In the same way, we can build a nondeterministic parity ω -word automaton $\mathcal{W}_{\mathcal{B}}$ recognising the complement of the winning condition $\mathbf{W}_{\mathcal{B}}$, and thus the complement of $\mathbf{W}_{\mathcal{A}} \cap \mathbf{W}_{\mathcal{B}}$ can be recognised by a nondeterministic parity ω -word automaton \mathcal{W} with polynomially many states and priorities and exponentially many transitions. By Lemma 2.2 we can further convert \mathcal{W} to an equivalent deterministic parity automaton \mathcal{W}' with exponentially many states and polynomially many priorities (w.r.t. the number of states of \mathcal{A}, \mathcal{B}). By Lemma 2.8 we can thus solve $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$ in EXPTIME, and by the characterisation in Lemma 6.1 we can solve the C -game separability problem within the same complexity.

Based on the size of the winning condition \mathcal{W}' and the strong connection between winning strategies for **Separator** and separating automata in the “soundness” direction of the proof of Lemma 6.1, we can also provide an upper bound on the size of a separating game automaton, when it exists. More precisely, if **Separator** wins the C -game-separability game $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$, then she has a positional winning strategy in the corresponding graph game of exponential size from Lemma 2.8. This means that **Separator** has a winning strategy \mathcal{M}

of exponential memory in $G_{\text{game}}^{\text{sep}}(\mathcal{A}, \mathcal{B}, C)$. This strategy is then translated to a separating game automaton $\alpha(\mathcal{M})$ with exponentially many states and priorities in C . Putting these considerations together gives the following complexity result.

Theorem 7.4. *The C -game separability problem can be solved in EXPTIME. Moreover, if a separating game automaton exists, then there exists one of exponential size.*

Altogether, Theorems 7.1 to 7.4 prove Theorem 1.1 announced in the introduction.

References

- [1] David Barozzini, Lorenzo Clemente, Thomas Colcombet, and Paweł Parys. Cost Automata, Safe Schemes, and Downward Closures. In *Proc. of ICALP'20*, LIPIcs, pages 109:1–109:18, 2020. URL: <https://drops.dagstuhl.de/opus/volltexte/2020/12516>, doi:10.4230/LIPIcs.ICALP.2020.109.
- [2] Achim Blumensath. Recognisability for algebras of infinite trees. *Theoretical Computer Science*, 412(29):3463–3486, 2011.
- [3] Mikołaj Bojańczyk. Star height via games. In *LICS*, pages 214–219, 2015.
- [4] Mikołaj Bojańczyk, Filippo Cavallari, Thomas Place, and Michał Skrzypczak. Regular tree languages in low levels of the Wadge Hierarchy. *Log. Meth. Comput. Sci.*, Volume 15, Issue 3, 2019. URL: <https://lmcs.episciences.org/5743>.
- [5] Mikołaj Bojańczyk and Wojciech Czerwiński. An automata toolbox, Feb 2018. URL: <https://www.mimuw.edu.pl/~bojan/paper/automata-toolbox-book>.
- [6] Mikołaj Bojańczyk and Thomas Place. Regular languages of infinite trees that are boolean combinations of open sets. In *Proc. of ICALP'12*, ICALP'12, pages 104–115, Berlin, Heidelberg, 2012. Springer-Verlag. URL: https://doi.org/10.1007/978-3-642-31585-5_13, doi:10.1007/978-3-642-31585-5_13.
- [7] J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969. URL: <http://www.jstor.org/stable/1994916>.
- [8] Cristian S. Calude, Sanjay Jain, Bakhadyr Khossainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proc. of STOC'17*, 2017.
- [9] Filippo Cavallari, Henryk Michalewski, and Michał Skrzypczak. A characterisation of Π_2^0 regular tree languages. In *Proc. of MFCS'17*, 2017.

- [10] Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. Regular separability of parikh automata. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *Proc. of ICALP'17*, volume 80, pages 117:1–117:13, 2017. URL: <http://drops.dagstuhl.de/opus/volltexte/2017/7497>, doi:10.4230/LIPIcs.ICALP.2017.117.
- [11] Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. Separability of Reachability Sets of Vector Addition Systems. In *Proc. of STACS'17*, volume 66 of *LIPICs*, pages 24:1–24:14, 2017. URL: <http://drops.dagstuhl.de/opus/volltexte/2017/7009>, doi:10.4230/LIPIcs.STACS.2017.24.
- [12] Lorenzo Clemente, Sławomir Lasota, and Radosław Piórkowski. Determinisability of One-Clock Timed Automata. In *Proc. of CONCUR'20*, volume 171 of *LIPICs*, pages 42:1–42:17, 2020. URL: <https://drops.dagstuhl.de/opus/volltexte/2020/12854>, doi:10.4230/LIPIcs.CONCUR.2020.42.
- [13] Lorenzo Clemente, Sławomir Lasota, and Radosław Piórkowski. Timed Games and Deterministic Separability. In *Proc. of ICALP'20*, volume 168 of *LIPICs*, pages 121:1–121:16, 2020. URL: <https://drops.dagstuhl.de/opus/volltexte/2020/12528>, doi:10.4230/LIPIcs.ICALP.2020.121.
- [14] Lorenzo Clemente, Paweł Parys, Sylvain Salvati, and Igor Walukiewicz. The diagonal problem for higher-order recursion schemes is decidable. In *Proc. of LICS'16*, 2016. URL: <http://doi.acm.org/10.1145/2933575.2934527>, doi:10.1145/2933575.2934527.
- [15] Thomas Colcombet. Fonctions régulières de coût. Habilitation thesis, Université Paris Diderot—Paris 7, 2013.
- [16] Thomas Colcombet, Denis Kuperberg, Christof Löding, and Michael Vanden Boom. Deciding the weak definability of Büchi definable tree languages. In *Proc. of CSL'13*, *LIPICs*, pages 215–230, 2013. URL: <http://drops.dagstuhl.de/opus/volltexte/2013/4199>, doi:<http://dx.doi.org/10.4230/LIPIcs.CSL.2013.215>.
- [17] Thomas Colcombet and Christof Löding. The Non-deterministic Mostowski Hierarchy and Distance-Parity Automata. In *Proc. of ICALP'08*, pages 398–409, 2008. URL: http://dx.doi.org/10.1007/978-3-540-70583-3_33, doi:10.1007/978-3-540-70583-3_33.
- [18] William Craig. Three uses of the herbrand-gentzen theorem in relating model theory and proof theory. *The Journal of Symbolic Logic*, 22(3):269–285, 1957. URL: <http://www.jstor.org/stable/2963594>.
- [19] Wojciech Czerwiński, Laure Daviaud, Nathanaël Fijalkow, Marcin Jurdziński, Ranko Lazić, and Paweł Parys. Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games. In

- Proc. of SODA '19*, pages 2333–2349, USA, 2019. Society for Industrial and Applied Mathematics.
- [20] Wojciech Czerwiński and Sławomir Lasota. Regular Separability of One Counter Automata. *Logical Methods in Computer Science*, Volume 15, Issue 2, June 2019. URL: <https://lmcs.episciences.org/5563>.
- [21] Wojciech Czerwiński, Sławomir Lasota, Roland Meyer, Sebastian Muskalla, K. Narayan Kumar, and Prakash Saivasan. Regular Separability of Well-Structured Transition Systems. In *Proc. of CONCUR'18*, LIPIcs, pages 35:1–35:18, 2018. URL: <http://drops.dagstuhl.de/opus/volltexte/2018/9573>, doi:10.4230/LIPIcs.CONCUR.2018.35.
- [22] Wojciech Czerwiński, Wim Martens, Lorijn van Rooijen, and Marc Zeitoun. A note on decidable separability by piecewise testable languages. In *Proc. of FCT'15*, 2015. URL: http://dx.doi.org/10.1007/978-3-319-22177-9_14, doi:10.1007/978-3-319-22177-9_14.
- [23] Wojciech Czerwiński and Georg Zetsche. An approach to regular separability in vector addition systems. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 341–354. ACM, 2020. URL: <https://doi.org/10.1145/3373718.3394776>, doi:10.1145/3373718.3394776.
- [24] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. of SFCS'91*, pages 368–377. IEEE Computer Society, 1991. URL: <http://dx.doi.org/10.1109/SFCS.1991.185392>, doi:10.1109/SFCS.1991.185392.
- [25] Alessandro Facchini, Filip Murlak, and Michał Skrzypczak. Index problems for game automata. *ACM Trans. Comput. Logic*, 17(4):24:1–24:38, November 2016. URL: <http://doi.acm.org/10.1145/2946800>, doi:10.1145/2946800.
- [26] Erich Grädel. Positional Determinacy of Infinite Games. In *Proc. of STACS'04*, volume 2996 of LNCS, pages 2–18. Springer, 2004. URL: <http://www.logic.rwth-aachen.de/pub/graedel/Gr-stacs04.ps>.
- [27] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata Logics, and Infinite Games - A Guide to Current Research*. Springer, 2002.
- [28] Alexander Kechris. *Classical Descriptive Set Theory*. Springer-Verlag, 1995.
- [29] Nils Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Annals of Pure and Applied Logic*, 69(2–3):243–268, 1994. URL: <http://www.sciencedirect.com/science/article/pii/0168007294900868>, doi:http://dx.doi.org/10.1016/0168-0072(94)90086-8.

- [30] Eryk Kopczyński. *Half-positional determinacy of infinite games*. PhD thesis, University of Warsaw, 2008.
- [31] Ralf Küsters and Thomas Wilke. Deciding the first level of the μ -calculus alternation hierarchy. In *Proc. of FSTTCS'02*, pages 241–252, Berlin, 2002.
- [32] Robert McNaughton and Seymour Papert. *Counter-free automata*. M.I.T. Press research monographs. M.I.T. Press, 1971.
- [33] Filip Murlak. Weak index versus Borel rank. In *Proc. of STACS'08*, volume 1 of *LIPICs*, pages 573–584, 2008. URL: <http://drops.dagstuhl.de/opus/volltexte/2008/1318>, doi:10.4230/LIPICs.STACS.2008.1318.
- [34] Damian Niwiński. On fixed-point clones (extended abstract). In *Proc. of ICALP'86*, pages 464–473, 1986. URL: <http://dl.acm.org/citation.cfm?id=646240.683678>.
- [35] Damian Niwiński and Igor Walukiewicz. Relating hierarchies of word and tree automata. In *Proc. of STACS'98*, pages 320–331, 1998.
- [36] Damian Niwiński and Igor Walukiewicz. A gap property of deterministic tree languages. *Theoretical Computer Science*, 303(1):215–231, 2003. URL: <http://www.sciencedirect.com/science/article/pii/S0304397502004528>, doi:[https://doi.org/10.1016/S0304-3975\(02\)00452-8](https://doi.org/10.1016/S0304-3975(02)00452-8).
- [37] Damian Niwiński and Igor Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Electronic Notes in Theoretical Computer Science*, 123:195–208, 2005. Proceedings of the 11th Workshop on Logic, Language, Information and Computation (WoLLIC 2004). URL: <http://www.sciencedirect.com/science/article/pii/S1571066105000551>, doi:<https://doi.org/10.1016/j.entcs.2004.05.015>.
- [38] Nir Piterman. From nondeterministic buchi and streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3):1–21, 2007.
- [39] Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. *Log. Methods Comput. Sci.*, 12(1), 2016. URL: [https://doi.org/10.2168/LMCS-12\(1:5\)2016](https://doi.org/10.2168/LMCS-12(1:5)2016), doi:10.2168/LMCS-12(1:5)2016.
- [40] Thomas Place and Marc Zeitoun. Adding successor: A transfer theorem for separation and covering. *ACM Trans. Comput. Logic*, 21(2), 2019. URL: <https://doi.org/10.1145/3356339>, doi:10.1145/3356339.
- [41] Michael O. Rabin. Weakly definable relations and special automata. In *Mathematical Logic and Foundations of Set Theory*, volume 59 of *Studies in Logic and the Foundations of Mathematics*, pages 1–23. Elsevier, 1970. URL: <http://www.sciencedirect.com/science/article/pii/S0049237X08719293>, doi:[https://doi.org/10.1016/S0049-237X\(08\)71929-3](https://doi.org/10.1016/S0049-237X(08)71929-3).

- [42] Luigi Santocanale and André Arnold. Ambiguous classes in mu-calculi hierarchies. *Theoretical Computer Science*, 333(1):265–296, 2005. Foundations of Software Science and Computation Structures. URL: <http://www.sciencedirect.com/science/article/pii/S0304397504007145>, doi:<https://doi.org/10.1016/j.tcs.2004.10.024>.
- [43] Sven Schewe and Thomas Varghese. Determinising parity automata. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Proc. of MFCS'14*, pages 486–498, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
- [44] Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.
- [45] Michał Skrzypczak and Igor Walukiewicz. Deciding the Topological Complexity of Büchi Languages. In *Proc. of ICALP'16*, volume 55 of *LIPICs*, pages 99:1–99:13, 2016. URL: <http://drops.dagstuhl.de/opus/volltexte/2016/6234>, doi:[10.4230/LIPICs.ICALP.2016.99](https://doi.org/10.4230/LIPICs.ICALP.2016.99).
- [46] Tomasz Fryderyk Urbanski. On Deciding if Deterministic Rabin Language Is in Büchi Class. In *Proc. of ICALP'00*, volume 1853 of *LNCS*, pages 663–674. 2000. URL: http://dx.doi.org/10.1007/3-540-45022-X_56, doi:[10.1007/3-540-45022-X_56](https://doi.org/10.1007/3-540-45022-X_56).
- [47] Klaus Wagner. On omega-regular sets. *Information and Control*, 43(2):123–177, 1979. URL: <https://www.sciencedirect.com/science/article/pii/S0019995879906533>, doi:[https://doi.org/10.1016/S0019-9958\(79\)90653-3](https://doi.org/10.1016/S0019-9958(79)90653-3).
- [48] Igor Walukiewicz. Deciding low levels of tree-automata hierarchy. *ENTCS*, 67:61–75, 2002. URL: <http://www.sciencedirect.com/science/article/pii/S1571066104805413>, doi:[https://doi.org/10.1016/S1571-0661\(04\)80541-3](https://doi.org/10.1016/S1571-0661(04)80541-3).
- [49] Georg Zetsche. An approach to computing downward closures. In *Proc. of ICALP'15*, volume 9135 of *LNCS*, pages 440–451, 2015. URL: http://dx.doi.org/10.1007/978-3-662-47666-6_35, doi:[10.1007/978-3-662-47666-6_35](https://doi.org/10.1007/978-3-662-47666-6_35).