Checking History-Determinism is NP-hard for Parity Automata

Aditya Prakash

University of Warwick, UK
`aditya.prakash@warwick.ac.uk`

Abstract. We show that the problem of checking if a given nondeterministic parity automaton simulates another given nondeterministic parity automaton is NP-hard. We then adapt the techniques used for this result to show that the problem of checking history-determinism for a given parity automaton is NP-hard. This is an improvement from Kuperberg and Skrzypczak's previous lower bound of parity games from 2015. We also show that deciding if Eve wins the one-token game or the two-token game of a given parity automaton is NP-hard. Finally, we show that the problem of deciding if the language of a nondeterministic parity automaton is contained in the language of a history-deterministic parity automaton can be solved in quasi-polynomial time.

1 Introduction

Deciding language inclusion between two automata is a fundamental problem in verification, wherein we ask whether all executions of an implementation satisfy a given specification. Unfortunately, the problem of checking language inclusion is often computationally hard. For parity automata—which are the focus of this paper—it is **PSPACE**-complete, with **PSPACE**-hardness already occurring for finite state automata [40].

On the other hand, simulation is a fundamental behavioural relation between two automata [35,24], which is a finer relation than language inclusion and is easier to check. For parity automata, simulation can be decided in polynomial time if the parity indices are fixed; otherwise it is in **NP** [13]. Note that while simulation between two automata is sufficient to guarantee language inclusion, it is not necessary.

For history-deterministic automata, however, the relation of language inclusion is equivalent to simulation [9,8], thus making them suitable for verification. These are nondeterministic automata where the nondeterminism can be resolved 'on-the-fly', just based on the prefix of the word read so far. The definition we use here was introduced by Henzinger and Piterman in 2006, where they dubbed it 'good-for-games' automata, while the term 'history-determinism' was coined by Colcombet [15] in the context of regular cost automata.

History-deterministic parity automata are more succinct than their deterministic counterparts [29] whilst still maintaining tractability for the problems of

verification and synthesis on them [25,29,8]. Consequently, history-deterministic parity automata have been the subject of extensive research [29,5,3,38,2,30], and has garnered significant attention over the recent years beyond parity automata as well, extending to quantitative automata [6,7], infinite state systems [22,33,36,10,20], and timed automata [9].

Despite these recent research efforts, a significant gap remains in understanding the complexity of checking whether a given parity automaton is history-deterministic. While Henzinger and Piterman have shown an **EXPTIME** upper bound [25], the best lower bound known so far is by Kuperberg and Skrzypczak since 2015 [29], who showed that checking for history-determinism is at least as hard as finding the winner of a parity game [29]—a problem that can be solved in quasi-polynomial time and is in $\mathbf{NP} \cap \mathbf{coNP}$ (and even in $\mathbf{UP} \cap \mathbf{coUP}$ [27]).

Kuperberg and Skrzypczak also gave a polynomial-time algorithm to check for history-determinism of co-Büchi automata in their work [29]. This was followed by a polynomial time algorithm to check for history-determinism of Büchi automata in 2018 by Bagnol and Kuperberg [3], who showed that in order to check if a Büchi automaton is history-deterministic, it suffices to find the winner of the so-called 'two-token game' of the automaton. This connection between history-determinism and two-token games was extended in 2020 to co-Büchi automata by Boker, Kuperberg, Lehtinen, and Skrzypczak [4]. It is conjectured that the winner of the two-token game of a parity automaton characterises its history-determinism. While the two-token conjecture is open to date, showing this conjecture would imply that one can check history-determinism of a given parity automata with a fixed parity index in polynomial time.

Our contributions. We show that checking for simulation between two parity automata is **NP**-hard when the parity index is not fixed. Since simulation is known to be in **NP**, this establishes the problem to be **NP**-complete (Theorem 10).

An adaptation of our proof of Theorem 10 gives us that checking history-determinism for a parity automata is also **NP**-hard (Theorem 14), when the parity index is not fixed. This is an improvement on Kuperberg and Skrzypczak's result from 2015, which shows that checking history-determinism for parity automata is at least as hard as solving parity games [29]. We also show, using the same reduction, that checking whether Eve wins the 2-token game (of a given parity automaton) is **NP**-hard, while checking whether Eve wins the 1-token game is **NP**-complete (Theorem 14).

As remarked earlier, for history-deterministic parity automata, the relation of language inclusion is equivalent to simulation. This gives us an immediate **NP** upper bound for checking language inclusion of a nondeterministic parity automaton in an HD-parity automata, as was observed by Schewe [38]. We show that we can do better, by showing the problem to be decidable in quasi-polynomial time (Theorem 19).

Overview of the paper: one reduction for all. The central problem used in our reduction is of checking whether Eve wins a 2-D parity game, which is known to be **NP**-complete due to Chatterjee, Henzinger and Piterman [13]. In Section 3, we give a reduction from this problem to checking for simulation between two parity automata, thus establishing its **NP**-hardness (Theorem 10). We then show, in Section 4.1, that the problem of checking whether Eve wins a good 2-D parity games—a technical subclass of 2-D parity games—is also **NP**-hard. In Section 4.2, we show that modifying the reduction in proof of Theorem 10 to take as inputs good 2-D parity games yields **NP**-hardness for the problems of checking history-determinism (Lemma 13) and of checking if Eve wins the 1-token game or the 2-token game (Theorem 14). Finally, in Section 5, we give a quasi-polynomial algorithm to check whether the language of a nondeterministic parity automaton is contained in the language of a history-deterministic parity automaton (Theorem 19), by reducing to finding the winner in a parity game.

2   Preliminaries

We let $\mathbb{N} = \{0, 1, 2, \cdots\}$ to be the set of natural numbers, and ω to be the cardinality of $\mathbb{N}$. We will use $[i, j]$ to denote the set of integers in the interval $\{i, i + 1, \ldots, j\}$ for two natural numbers i, j with $i < j$, and $[j]$ for the interval $[0, j]$. An *alphabet* Σ is a finite set of *letters*. We use Σ^* and Σ^ω to denote the set of words of finite and countably infinite length over Σ respectively. We also let ε denote the unique word of length 0.

2.1   Game arenas

An *arena* is a directed graph $G = (V, E)$ with vertices partitioned as $V_\forall$ and $V_\exists$ between two players Adam and Eve respectively. Additionally, a vertex $v_0 \in V_\forall$ is designated as the initial vertex. We say that the set of vertices $V_\exists$ is owned by Eve while the set of vertices $V_\forall$ is owned by Adam. Additionally, we assume that the edges E don't have both its start and end vertex in $V_\exists$ or $V_\forall$.

Given an arena as above, a *play* of this arena is an infinite path starting at v_0, and is formed as follows. A play starts with a token at the start vertex v_0, and proceeds for countably infinite rounds. At each round, the player who owns the vertex on which the token is currently placed chooses an outgoing edge, and the token is moved along this edge to the next vertex for another round of play. This creates an infinite path in the arena, which we call a play of G.

A *game* $\mathcal{G}$ consists of an arena $G = (V, E)$ and a winning condition given by a language $L \subseteq E^\omega$. We say that Eve *wins a play* ρ in G if ρ is in L, and Adam wins otherwise. Eve is said to have a *winning strategy* if she has a strategy to ensure that any play that can be produced when she plays according to her strategy is winning for Eve. We say that Eve *wins the game* if she has a winning strategy. Winning strategies are defined for Adam analogously, and we say that Adam wins the game if he has a winning strategy. In this paper we will deal with ω-regular games, which are known to be determined [34,23], i.e. each game has a winner. Two games are *equivalent* if they have the same winner.

2.2   Parity conditions

Let $G = (V, E)$ be a (finite or infinite) directed graph equipped with a *priority function* $\chi : E \to \mathbb{N}$ that assigns each edge with a natural number, called its *priority*. We say that an infinite path ρ in G satisfies the χ-*parity condition* if the highest priority occurring infinitely often in the path is even. When clear from the context, we will drop 'parity condition' and instead say that ρ satisfies χ.

A parity condition is easily *dualised*. Given a priority function χ as above, consider the priority function $\chi' := \chi + 1$ that is obtained by increasing all the labels by 1. Then, an infinite path satisfies χ' if and only if it does not satisfy χ.

2.3   Parity games

A *parity game* $\mathcal{G}$ is played over a finite game arena $G = (V, E)$, with the edges of G labelled by a priority function $\chi : E \to \{0, 1, 2, \cdots, d\}$. A play ρ in the arena of $\mathcal{G}$ is winning for Eve if and only if ρ satisfies the χ-parity condition. Following the breakthrough of Calude, Jain, Khoussainov, Li, and Stephan, it is well-known that the winner of a parity game can be found in quasi-polynomial time: $n^{\mathcal{O}(\log d)}$ for an n-vertex parity game with priority in $[d]$ [11,28,32,21].

2.4   Muller conditions and Zielonka trees

A $(C, \mathcal{F})$-Muller conditions consists of a finite set of colours C, and a set $\mathcal{F}$ consisting of subsets of C. An infinite sequence in C^ω satisfies the $(C, \mathcal{F})$-Muller condition if the set of colours seen infinitely often along the sequence is in $\mathcal{F}$.

A *Muller game* $\mathcal{G}$ consists of an arena $G = (V, E)$, a colouring function $\pi : E \to C$ and a Muller condition $(C, \mathcal{F})$. An infinite play ρ in $\mathcal{G}$ is winning for Eve if the set of colours seen infinitely often along the play is in $\mathcal{F}$, and Eve wins the Muller game $\mathcal{G}$ if she has a winning strategy.

Every Muller game can be converted to an equivalent parity game, as shown by Gurevich and Harrington [23]. We will use the conversion of Dziembowski, Jurdziński, and Walukiewikz that involve Zielonka trees [17,12], which we define below.

Definition 1 (Zielonka tree). *Given a Muller condition $(C, \mathcal{F})$, the* Zielonka tree *of a Muller condition, denoted $Z_{C,\mathcal{F}}$, is a tree whose nodes are labelled by subsets of C, and is defined inductively. The root of the tree is labelled by C. For a node that is already constructed and labelled with the set X, its children are nodes labelled by distinct maximal non-empty subsets $X' \subsetneq X$ such that $X \in \mathcal{F} \Leftrightarrow X' \notin \mathcal{F}$. If there are no such X', then the node labelled X is a leaf of $Z_{C,\mathcal{F}}$ and has no attached children.*

Given a $(C, \mathcal{F})$-Muller condition, consider the language $L \subseteq C^\omega$ consisting of words w that satisfy the $(C, \mathcal{F})$-Muller condition. The language L is then said to be the language of the $(C, \mathcal{F})$-Muller condition, and can be recognised by a deterministic parity automaton, whose size depends on the size of the Zielonka tree [12].

Lemma 2 ([12]). *Let $(C, \mathcal{F})$ be a Muller condition with the Zielonka tree $Z_{C,\mathcal{F}}$ that has n leaves and height h. Then there is a deterministic parity automaton $\mathcal{D}_{C,\mathcal{F}}$ that can be constructed in polynomial time such that $\mathcal{D}_{C,\mathcal{F}}$ has n states and $(h + 1)$ priorities, and accepts the language of the the $(C, \mathcal{F})$-Muller condition.*

Consider a Muller game $\mathcal{G}$ on the arena $G = (V, E)$ with the colouring function $\pi : E \rightarrow C$ and the Muller condition $(C, \mathcal{F})$. We can then construct an equivalent parity game $\mathcal{G}'$ by taking the product of $\mathcal{G}$ with the automaton $\mathcal{D}_{C,\mathcal{F}}$ from Lemma 2. In more details, the set of vertices V' of $\mathcal{G}'$ consists of vertices of the form $v' = (v, q)$, where v is a vertex in $\mathcal{G}$ and q is a state in $\mathcal{D}_{C,\mathcal{F}}$. The owner of the vertex (v, q) is the owner of the vertex v, and the initial vertex is (ι, q_0), where ι is the initial vertex in $\mathcal{G}$ and q_0 is the initial state in $\mathcal{D}_{C,\mathcal{F}}$. We have the edge $e' = (v, q) \rightarrow (v', q')$ in $\mathcal{G}'$ if $e = v \rightarrow v'$ is an edge in $\mathcal{G}$ with the colour $\pi(e) = c$, and $\delta = q \xrightarrow{c} q'$ is a transition in $\mathcal{D}_{C,\mathcal{F}}$. The edge e' is assigned the priority $\Omega(\delta)$ in $\mathcal{G}'$, where Ω is the priority function of the automaton $\mathcal{D}_{C,\mathcal{F}}$. The game $\mathcal{G}'$ then is such that Eve wins $\mathcal{G}$ if and only if Eve wins $\mathcal{G}'$.

Lemma 3. *Let $\mathcal{G}$ be a Muller game on an arena consisting of m vertices with a Muller condition $(C, \mathcal{F})$ whose Zielonka tree $Z_{C,\mathcal{F}}$ has n leaves and height h. Then, $\mathcal{G}$ can be converted to an equivalent parity game $\mathcal{G}'$ which has mn many vertices and $h + 1$ priorities.*

2.5   2-dimensional parity game

Multi-dimensional parity games were introduced by Chatterjee, Henzinger and Piterman, where they called it generalised parity games [13]. For our purposes, it suffices to consider 2-dimensional (2-D) parity games, which is what we define now.

A *2-dimensional parity game* $\mathcal{G}$ is similar to a parity game, but we now have two priority functions $\pi_1 : E \rightarrow [0, d_1]$ and $\pi_2 : E \rightarrow [0, d_2]$ on E. Any infinite play in the game is winning for Eve if the following holds: *if the play satisfies π_1, then it satisfies π_2.*
We say that Adam wins the game otherwise. We call the problem of deciding whether Eve wins a 2-D parity game as 2-D PARITY GAME.

2-D PARITY GAME: Given a 2-D parity game $\mathcal{G}$, does Eve win $\mathcal{G}$?

If Eve has a strategy to win a 2-D parity game, then Eve has a positional winning strategy to do so, i.e. she can win by always choosing the same edge from each vertex in $V_\exists$, which is given by a function $\sigma : V_\exists \rightarrow E$. This can be inferred directly from seeing the 2-D parity game as a Rabin game, which are known to have positional strategies for Eve [18]. Furthermore, given a positional strategy σ for Eve in a 2-D parity game (or a Rabin game), one can check in polynomial time if σ is a winning strategy [18]. This gives us a nondeterministic polynomial time procedure to decide if Eve wins a given 2-D parity game. In 1988, Emerson and Jutla established **NP**-hardness for Rabin games [19], and their proof was extended by Chatterjee, Henzinger and Piterman in 2007 to show **NP**-hardness for 2-D parity games as well [13].

Theorem 4 ([13]). *The problem of deciding whether Eve wins a given 2-D parity game is* **NP***-complete.*

Remark 5. Chatterjee, Henzinger and Piterman give a slightly different and a more natural definition of 2-D parity games[13], where the winning condition for Eve requires every play to satisfy either of two given parity conditions. It is easy to see, however, that both definitions are log-space inter-reducible to each other, by dualising the first parity condition. Our definition, although less natural, makes the connection to simulation games and our reductions in Sections 3 and 4 more transparent.

2.6   Parity automata

A *nondeterministic parity automaton* $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \Omega)$ contains a finite directed graph with edges labelled by letters in Σ. These edges are called *transitions*, which are elements of the set $\Delta \subseteq Q \times \Sigma \times Q$, and the vertices of this graph are called *states*, which are elements of the set Q.

Each automaton has a designated *initial state* $q_0 \in Q$, and a priority function $\Omega : \Delta \to [i, j]$ which assigns each transition a *priority* in $[i, j]$, for $i < j$ two natural numbers. We say that $(j - i + 1)$ is the *number of priorities* or the *parity index* of $\mathcal{A}$. For states p, q and an alphabet $a \in \Sigma$, we use $p \xrightarrow{a:c} q$ to denote a transition from p to q on the letter a that has the priority c.

A *run* on an infinite word w in Σ^ω is an infinite path in the automaton, starting at the initial state and following transitions that correspond to the letters of w in sequence. We say that such a run is *accepting* if it satisfies the Ω-parity condition, and a word w in Σ^ω is accepting if the automaton has an accepting run on w. The *language* of an automaton $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of words that it accepts. We say that the automaton $\mathcal{A}$ *recognises* a language $\mathcal{L}$ if $L(\mathcal{A}) = \mathcal{L}$. A parity automaton $\mathcal{A}$ is said to be *deterministic* if for any given state in $\mathcal{A}$ and any given letter in Σ, there is at most one transition from the given state on the given letter. A *Büchi (resp. co-Büchi)* automaton is a parity automaton with priorities in $[1, 2]$ (resp. $[0, 1]$).

2.7   Simulation

We say a parity automaton $\mathcal{A}$ *simulates* another parity automaton $\mathcal{B}$ if for any (finite or infinite) run on $\mathcal{B}$, there is a corresponding run on $\mathcal{A}$ on the same word that can be constructed on-the-fly such that if the run in $\mathcal{B}$ is accepting, so is the corresponding run in $\mathcal{A}$. This is made more formal by the following *simulation game*.

Definition 6 (Simulation game). *Given nondeterministic parity automata* $\mathcal{A} = (Q, \Sigma, q_0, \Delta_A, \Omega_A)$ *and* $\mathcal{B} = (P, \Sigma, p_0, \Delta_B, \Omega_B)$, *the* simulation game *between* $\mathcal{A}$ *and* $\mathcal{B}$, *denoted* $Sim(\mathcal{A}, \mathcal{B})$, *is defined as a two player game between Adam and Eve as follows, with positions in* $P \times Q$. *A play of the simulation game starts at the position* (p_0, q_0), *and has* ω *many rounds. For each* $i \in \mathbb{N}$, *the* $(i + 1)^{th}$ *round starts at a position* $(p_i, q_i) \in P \times Q$, *and proceeds as follows:*

– *Adam selects a letter $a \in \Sigma$, and a transition $p_i \xrightarrow{a} p_{i+1}$ in $\mathcal{B}$.*
– *Eve selects a transition $q_i \xrightarrow{a} q_{i+1}$ on the same letter in $\mathcal{A}$.*

The new position is (p_{i+1}, q_{i+1}), for another round of the play.

The player Eve wins the above play if either her constructed run in $\mathcal{A}$ is accepting, or Adam's constructed run in $\mathcal{B}$ is rejecting. If Eve has a winning strategy in $Sim(\mathcal{A}, \mathcal{B})$, then we say that $\mathcal{A}$ simulates $\mathcal{B}$, *and denote it by $\mathcal{B} \lesssim \mathcal{A}$.*

We call the problem of checking whether a parity automaton simulates another as SIMULATION:

SIMULATION: Given two parity automata $\mathcal{A}$ and $\mathcal{B}$, does $\mathcal{A}$ simulate $\mathcal{B}$?

The simulation game $Sim(\mathcal{A}, \mathcal{B})$ can naturally be seen as a 2-D parity game, where the arena is the product of two automata with Adam selecting letters and transitions in $\mathcal{A}$ and Eve transitions in $\mathcal{B}$, and the priority functions χ_1 and χ_2 based on corresponding priorities of transitions in $\mathcal{A}$ and $\mathcal{B}$ respectively. Since 2-D PARITY GAME can be solved in **NP**, SIMULATION can be solved in **NP** as well.

2.8 History-determinism

A *history-deterministic* (HD) parity automaton is a nondeterministic parity automaton in which the nondeterminism can be resolved 'on-the-fly' just based on the prefix read so far, without knowing the rest of the word. The history-determinism of a parity automaton can be characterised by the letter game, which is a 2-player turn-based game between Adam and Eve, who take alternating turns to select a letter and a transition in the automaton (on that letter), respectively. After the game ends, the sequence of Adam's choices of letters is an infinite word, and the sequence of Eve's choices of transitions is a run on that word. Eve wins the game if her run is accepting or Adam's word is rejecting, and we say that an automaton is history-deterministic if Eve has a winning strategy in the history-determinism game.

Definition 7 (Letter game). *Given a parity automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \Omega)$, the* letter game *of $\mathcal{A}$ is defined between the two players $\forall$ and $\exists$ as follows, with positions in $Q \times \Sigma^*$. The game starts at (q_0, ε) and proceeds in ω many rounds. For each $i \in \mathbb{N}$, the $(i+1)^{th}$ round starts at a position $(q_i, w_i) \in Q \times \Sigma^i$, and proceeds as follows:*

– $\forall$ *selects a letter $a_i \in \Sigma$*
– $\exists$ *selects a transition $q_i \xrightarrow{a_i} q_{i+1} \in \Delta$*

The new position is (q_{i+1}, w_{i+1}), where $w_{i+1} = w_i a_i$.

Thus, the play of a letter game can be seen as Adam constructing a word letter-by-letter, and Eve constructing a run transition-by-transition on the same word. Eve wins such a play if the following holds: if Adam's word is in $L(\mathcal{A})$, then Eve's run is accepting.

The automaton $\mathcal{A}$ is said to be *history-deterministic* if Eve has a winning strategy on the letter game of $\mathcal{A}$. We are interested in the problem of checking whether a given parity automaton is history-deterministic, which we shall denote by HISTORY-DETERMINISTIC.

HISTORY-DETERMINISTIC: Given a parity automaton $\mathcal{A}$, is $\mathcal{A}$ history-deterministic?

2.9   Token games

Token games, or k-token games, is defined on an automaton and is similar to letter games, where Adam is constructing a word letter-by-letter, Eve is constructing a run transition-by-transition on the same word over ω many rounds, and additionally, Adam is constructing k runs transition-by-transition on this word as well. The winning objective of Eve requires her to construct an accepting run if one of k Adam's runs is accepting.

Definition 8 (k-token game). *Given a nondeterministic parity automaton $\mathcal{A} = (Q, \Sigma, , q_0, \Delta, \Omega)$, the k-token game of $\mathcal{A}$ is defined between the two players Adam and Eve as follows, with positions in $Q \times Q^k$. The game starts at $(q_0, (q_0)^k)$ and proceeds in ω many rounds. For each $i \in \mathbb{N}$, the $(i+1)^{th}$ round starts at a position $(q_i, (p_i^1, p_i^2, \cdots, p_i^k)) \in Q \times Q^k$, and proceeds as follows:*

- *Adam selects a letter $a_i \in \Sigma$*
- *Eve selects a transition $q_i \xrightarrow{a_i} q_{i+1} \in \Delta$*
- *Adam selects k transitions $p_i^1 \xrightarrow{a_i} p_{i+1}^1, p_i^2 \xrightarrow{a_i} p_{i+1}^2, \cdots p_i^k \xrightarrow{a_i} p_{i+1}^k$,*

The new position is $(q_{i+1}, (p_{i+1}^1, p_{i+1}^2, \cdots, p_{i+1}^k))$, from where the $(i+2)^{th}$ round begins.

Thus, in a play of the k-token game, Eve constructs a run and Adam k runs, all on the same word. Eve wins such a play if the following holds: if one of Adam's k runs is accepting, then Eve's run is accepting.

Bagnol and Kuperberg have shown that for any parity automaton $\mathcal{A}$, the 2-token game of $\mathcal{A}$, and the k-token game of $\mathcal{A}$ for any $k \geq 2$, are equivalent.

Lemma 9 ([3]). *Given a parity automaton $\mathcal{A}$, Eve wins 2-token game of $\mathcal{A}$ if and only if Eve wins the k-token game of $\mathcal{A}$ for all $k \geq 2$.*

If $\mathcal{A}$ is a nondeterministic Büchi or co-Büchi automaton, then Eve wins the 2-token game of $\mathcal{A}$ if and only if $\mathcal{A}$ is history-deterministic[3,4], and it is conjectured that this result extends to all parity automata.

TWO-TOKEN CONJECTURE: Given a nondeterministic parity automaton $\mathcal{A}$, Eve wins the 2-token game of $\mathcal{A}$ if and only if $\mathcal{A}$ is history-deterministic.

3 Simulation is NP-hard

In this section, we show that the problem of deciding if a parity automaton simulates another is **NP**-hard, by giving a reduction from the problem of deciding whether Eve wins a 2-D parity game, which was shown to be **NP**-complete by Chatterjee, Henzinger and Piterman [13]. Since a simulation game can be solved in **NP** (see Section 2.7), we obtain **NP**-completeness.

Theorem 10. *Given two parity automata $\mathcal{A}$ and $\mathcal{B}$, deciding if $\mathcal{A}$ simulates $\mathcal{B}$ is **NP**-complete.*

Since $\mathcal{A}$ simulates $\mathcal{B}$ if and only if Eve wins the simulation game, which is a 2-dimensional parity game (see Section 2.7), and deciding if Eve wins a 2-D parity game is in **NP** [13], we get that the problem of checking for simulation is in **NP**. Hence, we show that SIMULATION is **NP**-hard in the rest of this section, by giving a reduction from 2-D PARITY GAME.

Let $\mathcal{G}$ be a two-dimensional parity game played on the arena $G = (V, E)$, with two priority function χ_1 and χ_2. We recall that the winning condition for Eve in such a game requires a play to satisfy the χ_2-parity condition if the χ_1-parity condition is satisfied (see Section 2.5).

Overview of the reduction. We shall construct two parity automata $\mathcal{H}$ and $\mathcal{D}$ such that $\mathcal{H}$ simulates $\mathcal{D}$ if and only if Eve wins $\mathcal{G}$. The automata $\mathcal{H}$ and $\mathcal{D}$ are over the alphabet $E \cup \{\$\}$, where $\$$ is a letter added for padding. The automaton $\mathcal{D}$ is deterministic, while the automaton $\mathcal{H}$ has nondeterminism on the letter $\$$ and contains a copy of $\mathcal{D}$.

Adam, by his choice of letter in $Sim(\mathcal{H}, \mathcal{D})$, captures his moves from Adam vertices in $\mathcal{G}$. Similarly, Eve, by means of choosing her transition on $\$$ in $\mathcal{H}$, captures her moves from Eve vertices in $\mathcal{G}$. After each $\$$-round in $Sim(\mathcal{H}, \mathcal{D})$, we require Adam to 'replay' Eve's choice as the next letter. Otherwise, Eve can take a transition to the same state as Adam (recall that $\mathcal{H}$ contains a copy of $\mathcal{D}$), from where she wins the play in $Sim(\mathcal{H}, \mathcal{D})$ by copying Adam's transitions in each round from here on-wards. The priorities of $\mathcal{D}$ are based on χ_1, while the priorities of $\mathcal{H}$ are based on χ_2. This way $\mathcal{D}$ and $\mathcal{H}$ roughly accept words that correspond to plays in $\mathcal{G}$ satisfying χ_1 and χ_2 respectively.

We first present our reduction on an example 2-D parity game whose sub-game consists of vertices u, v, v', w, w' with edges between them as shown in Fig. 1. For the Adam vertex u, we have corresponding states u_D in $\mathcal{D}$ and u_H in $\mathcal{H}$. An Adam move from u in $\mathcal{G}$ corresponds to one round of $Sim(\mathcal{H}, \mathcal{D})$ from the position (u_D, u_H). In $\mathcal{G}$, Adam chooses an outgoing edge, say $e = (u, v)$ from u such that $\chi_1(u) = c_1$ and $\chi_2(u) = c_2$. This corresponds to Adam choosing the letter e in $Sim(\mathcal{H}, \mathcal{D})$. We then have the corresponding unique transitions $u_D \xrightarrow{e:c_1} v_\$$ in $\mathcal{D}$ and $u_H \xrightarrow{e:c_2} v_H$ in $\mathcal{H}$, and hence the simulation game goes to $(v_\$, v_H)$.

Fig. 1. A snippet of a game $\mathcal{G}$, and the corresponding automata $\mathcal{D}$ and $\mathcal{H}$ constructed in the reduction. The Adam vertices are represented by pentagons and Eve vertices by squares. The automaton $\mathcal{D}$ is deterministic, and $\mathcal{H}$ contains a copy of $\mathcal{D}$.

An Eve move from v in $\mathcal{G}$ corresponds to two rounds of the simulation game from $(v_\$, v_H)$. In $Sim(\mathcal{H}, \mathcal{D})$, Adam must select a letter $\$$ and the unique $\$$ transition $v_\$ \xrightarrow{\$:0} v_D$ on $\mathcal{D}$, since $\$$ is the only letter on which there is an outgoing transition from $v_\$$. Eve must now select a transition on $\$$ from v_H. Suppose she picks $v_H \xrightarrow{\$:0} (v_H, f)$ where $f = (v, w)$ is an outgoing edge from v in $\mathcal{G}$ with $\chi_1(f) = c_5$ and $\chi_2(f) = c_6$. This corresponds to Eve selecting the edge f from her vertex v in $\mathcal{G}$. The simulation game goes to the position $(v_D, (v_H, f))$. From here, Adam may select any outgoing edge from v as the letter. If he picks $f' = (v, w')$ and the transition $v_D \xrightarrow{f':c_7} w'_D$, then Eve can pick the transition $(v_H, f) \xrightarrow{f':c_8} w'_D$ and move to the same state as Adam: such transitions are
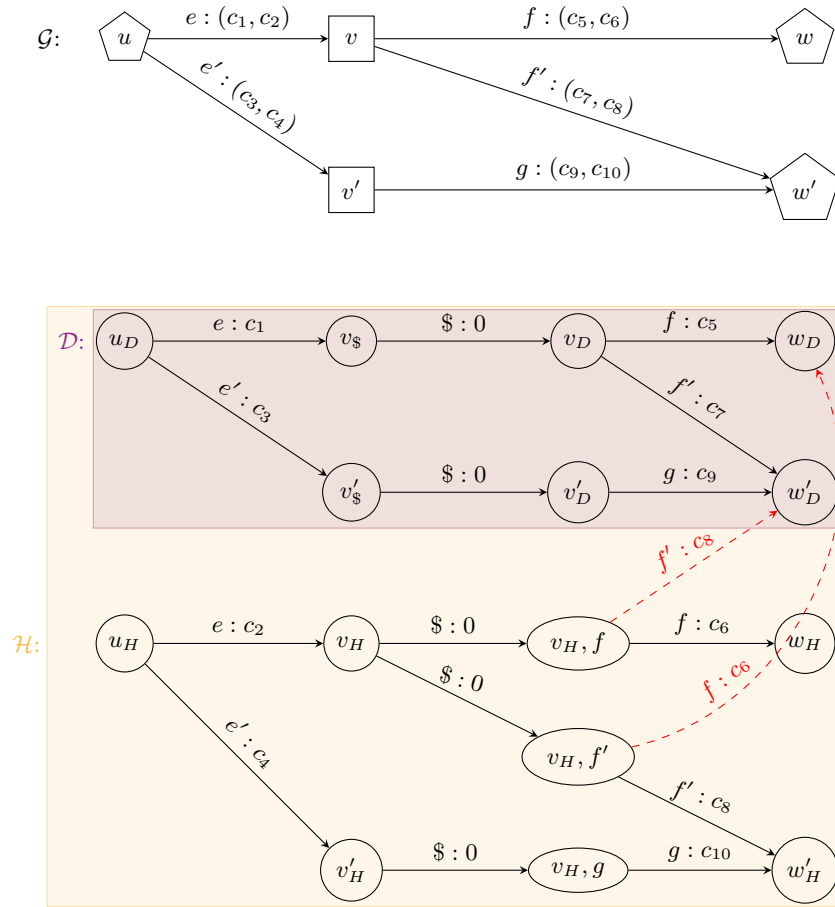
indicated by dashed edges in Fig. 1. From here, Eve can win $Sim(\mathcal{H}, \mathcal{D})$ by simply copying Adam's transitions. Otherwise, Adam picks the edge f as the letter, same as Eve's 'choice' in the previous round, resulting in the transition $v_D \xrightarrow{f:c_5} w_D$ in $\mathcal{D}$ and $(v_H, f) \xrightarrow{f:c_6} w_H$ in $\mathcal{H}$, and the simulation game goes to the position (w_D, w_H), from where the game continues similarly.

The reduction. We now give formal descriptions of the two parity automata $\mathcal{D}$ and $\mathcal{H}$ such that $\mathcal{H}$ simulates $\mathcal{D}$ if and only if Eve wins $\mathcal{G}$. We encourage the reader to refer to Fig. 1 while reading the construction of the automata described below.

Both automata $\mathcal{D}$ and $\mathcal{H}$ are over the alphabet $\Sigma = E \cup \{\$\}$. The automaton $\mathcal{D}$ is given by $\mathcal{D} = (P, \Sigma, p_0, \Delta_D, \Omega_D)$, where the set P consists of the following states:

- states u_D for each Adam vertex $u \in V_\forall$,
- states $v_\$$ and v_D for each Eve vertex $v \in V_\exists$.

The state $p_0 = \iota_D$ is the initial vertex, where ι is the initial vertex of the game $\mathcal{G}$. The set Δ_D consists of the following transitions with priorities as indicated:

- transitions $u_D \xrightarrow{e:\chi_1(e)} v_\$$ for every edge $e = (u, v)$ in $\mathcal{G}$ such that $u \in V_\forall$ is an Adam-vertex in $\mathcal{G}$,
- transitions $v_\$ \xrightarrow{\$:0} v_D$ for every $v \in V_\exists$ that is an Eve-vertex in $\mathcal{G}$,
- transitions $v_D \xrightarrow{f:\chi_1(f)} w_D$ for every edge $f = (v, w)$ in $\mathcal{G}$ such that $v \in V_\exists$ is an Eve vertex in $\mathcal{G}$

The automaton $\mathcal{H}$ is given by $\mathcal{H} = (Q, \Sigma, q_0, \Delta_H, \Omega_H)$, where the set Q consists of the following states:

- states u_H for each Adam vertex $u \in V_\forall$,
- states v_H for each Eve vertex $v \in V_\exists$,
- states (v_H, f) for each edge $f = (v, w)$ in $\mathcal{G}$ such that $v \in V_\exists$ is an Eve vertex,
- all states in P, the set of states of $\mathcal{D}$.

The vertex ι_H is the initial vertex. The set Δ_H consists of the following transitions with priorities as indicated:

- transitions $u_H \xrightarrow{e:\chi_2(e)} v_H$ for every edge $e = (u, v)$ in $\mathcal{G}$ such that $u \in V_\forall$ is an Adam-vertex in $\mathcal{G}$,
- transitions $v_H \xrightarrow{\$:0} (v_H, f)$ for every edge $f = (v, w)$ in $\mathcal{G}$ that is outgoing from an Eve-vertex $v \in V_\exists$,
- transitions $(v_H, f) \xrightarrow{f:\chi_2(f)} w_H$ for every edge $f = (v, w)$ in $\mathcal{G}$ outgoing from an Eve-vertex $v \in V_\exists$,
- transitions $(v_H, f) \xrightarrow{f':\chi_2(f')} w'_D$ for every edge $f' = (v, w') \neq f$ in $\mathcal{G}$ outgoing from an Eve-vertex $v \in V_\exists$,
- all transitions of $\mathcal{D}$.

Note that, by construction, $\mathcal{H}$ contains a copy of $\mathcal{D}$ as a sub-automaton.

Correctness of the reduction. We now show that Eve wins the simulation game $Sim(\mathcal{H}, \mathcal{D})$ if and only if Eve wins the game $\mathcal{G}$. Call any play of the simulation game *uncorrupted* if the following holds: whenever Eve's state in $\mathcal{H}$ is at (v_H, f) at the start of a round of $Sim(\mathcal{H}, \mathcal{D})$, Adam plays the letter f. If Adam plays a letter $f' \neq f$, then we call such a move *corrupted*. Any play consisting of a corrupted move is called a corrupted play.

It is clear that Eve wins any play of $Sim(\mathcal{H}, \mathcal{D})$ that is corrupted, since a corrupted move causes Eve's state in $\mathcal{H}$ and Adam's state in $\mathcal{D}$ to be the same in $Sim(\mathcal{H}, \mathcal{D})$. Then, both Eve's and Adam's runs are identical and determined by the choices of Adam's letters. In particular, Eve's run is accepting if Adam's run is.

Thus, it suffices to consider only uncorrupted plays. We first observe an invariant that is preserved throughout any uncorrupted play of $Sim(\mathcal{H}, \mathcal{D})$.

> *Invariant:* At the start of any round of the simulation game $Sim(\mathcal{H}, \mathcal{D})$ following an uncorrupted play:
> - Adam's state is at u_D for some $u \in V_\forall$ if and only if Eve's state is at u_H
> - Adam's state is at $v_\$$ for some $v \in V_\exists$ if and only if Eve's state is at v_H
> - Adam's state is at v_D for some $v \in V_\exists$ if and only if Eve's state is at (v_H, f) for some edge f that is outgoing from v.

This invariant is easy to observe from the construction, and can be shown by a routine inductive argument.

Note that if Adam constructs the word $w = e_0 \$ f_0 e_1 \$ f_1 \ldots$—which we denote by $(e_i \$ f_i)_{i \geq 0}$ for succinctness—in an uncorrupted play of $Sim(\mathcal{H}, \mathcal{D})$, then Eve's run on $\mathcal{H}$ is uniquely determined, since the letter f_i indicates how nondeterminism on $\mathcal{H}$ was resolved by Eve on the ith occurrence of $\$$ in $Sim(\mathcal{H}, \mathcal{D})$. Thus, any uncorrupted play in the simulation can be thought of as Adam selecting the e_i's and Eve selecting the f_i's, resulting in the word $w = (e_i \$ f_i)_{i \geq 0}$ being constructed in the simulation game. Note that then, by construction, $(e_i f_i)_{i \geq 0}$ is a play in $\mathcal{G}$. Conversely, if $(e_i f_i)_{i \geq 0}$ is a play in $\mathcal{G}$, then there is an uncorrupted play of $Sim(\mathcal{H}, \mathcal{D})$ whose word is $w = (e_i \$ f_i)_{i \geq 0}$.

Furthermore, observe that the transitions on a letter $e \in E$ in $\mathcal{D}$ and $\mathcal{H}$ in any uncorrupted play have the priorities $\chi_1(e)$ and $\chi_2(e)$ respectively, while transitions on $\$$ have priority 0. Thus, in a uncorrupted play of $Sim(\mathcal{H}, \mathcal{D})$ whose word is $(e_i \$ f_i)_{i \geq 0}$, the highest priorities occurring infinitely often in the run on $\mathcal{D}$ and $\mathcal{H}$ are the same as the highest χ_1-priority and χ_2-priority occurring infinitely often in the play $(e_i f_i)_{i \geq 0}$ respectively.

Thus, an uncorrupted play in $Sim(\mathcal{H}, \mathcal{D})$ whose word is $w = (e_i \$ f_i)_{i \geq 0}$ is winning for Eve if and only if the play $(e_i f_i)_{i \geq 0}$ in $\mathcal{G}$ is winning for Eve. Since Eve wins any corrupted play, the equivalence of the games $\mathcal{G}$ and $Sim(\mathcal{H}, \mathcal{D})$ follows easily now. If Eve has a winning strategy in $\mathcal{G}$, she can use her strategy to select transitions so that the word $w = (e_i \$ f_i)_{i \geq 0}$ that is constructed in any uncorrupted play ρ of $Sim(\mathcal{H}, \mathcal{D})$ corresponds to a winning play for her in $\mathcal{G}$,

and hence ρ is winning in $Sim(\mathcal{H}, \mathcal{D})$. If Adam ever makes a corrupted move, she wins trivially.

Conversely, if she has a winning strategy in $Sim(\mathcal{H}, \mathcal{D})$, then she can use her strategy to choose moves in $\mathcal{G}$ so that the play $(e_i f_i)_{i \geq 0}$ corresponds to a winning uncorrupted play of $Sim(\mathcal{H}, \mathcal{D})$ in which the word $(e_i \$ f_i)_{i \geq 0}$ is constructed, thus resulting in the play $(e_i f_i)_{i \geq 0}$ to also be winning for Eve.

4   Checking History-Determinism is NP-hard

In this section, we show that the problem of deciding whether a given nondeterministic parity automaton is history-deterministic is **NP**-hard, as is the problem of deciding whether Eve wins the 1-token game or the 2-token game of a given parity automaton. To show this, we reduce from deciding whether Eve wins a 2-D parity game with priority functions χ_1 and χ_2 that satisfies the following property: any play satisfying the χ_2-parity condition also satisfies the χ_1-parity condition. We call such games as 'good 2-D parity games'. We first show in Section 4.1 that deciding whether Eve wins a good 2-D parity game is **NP**-hard, and then use this to show **NP**-hardness for the problems mentioned above in Section 4.2.

4.1   Good 2-D parity games

Definition 11 (Good 2-D parity game). *A 2-D parity game $\mathcal{G}$ with the priority functions χ_1 and χ_2 is called* good *if any play in $\mathcal{G}$ that satisfies χ_2 also satisfies χ_1.*

We call the problem of deciding whether Eve wins a 2-D parity game as GOOD 2-D PARITY GAME. Chatterjee, Henzinger and Piterman's reduction from SAT to 2-D PARITY GAME [13] can also be seen as a reduction to GOOD 2-D PARITY GAME, as we show below.

Lemma 12. *Deciding whether Eve wins a good 2-D parity game is* **NP***-hard.*

Proof. We reduce from the problem of SAT [16]. Let ϕ be a Boolean formula over the variables $X = \{x_1, x_2, \cdots, x_M\}$ that is a disjunction of terms t_i for each $i \in [1, N]$, where each term t_i is a finite conjunction of *literals*—elements of the set $L = \{x_1, x_2, \cdots, x_M, \neg x_1, \neg x_2, \cdots, \neg x_M\}$. We shall construct a good 2-D parity game $\mathcal{G}_\phi$ such that Eve wins $\mathcal{G}_\phi$ if and only if ϕ has a satisfying assignment.

Let $T = \{t_1, t_2, \cdots, t_N\}$ be the set of all terms in ϕ. The game $\mathcal{G}_\phi$ has the set $T \cup L$ as its set of vertices. The elements of L are Adam vertices, while the elements of T are Eve vertices. We set the element x_1 in L to be the initial vertex. Each Adam vertex l in L has an outgoing edge $e = (l, t)$ to every term t in T, and every Eve vertex $t \in T$ has an outgoing edge $f = (t, l)$ to a literal l if l is a literal in t. Thus, each play in the game $\mathcal{G}_\phi$ can be seen as Adam and Eve choosing a term and a literal in that term in alternation respectively.

The game $\mathcal{G}_\phi$ has priority functions χ_1 and χ_2. To every edge $e = (l, t)$ that is outgoing from an Adam vertex, both priority functions χ_1 and χ_2 assign e the priority 0, i.e., $\chi_1(e) = \chi_2(e) = 0$. Every edge $e = (t, l)$ that is outgoing from an Eve vertex is assigned priorities as follows:

$$\chi_1(e) = \begin{cases} 2j + 2 & \text{if } l = x_j \\ 2j + 1 & \text{if } l = \neg x_j \end{cases} \qquad \chi_2(e) = \begin{cases} 2j & \text{if } l = x_j \\ 2j + 1 & \text{if } l = \neg x_j \end{cases}$$

This concludes our description of the game $\mathcal{G}_\phi$. We now show that $\mathcal{G}_\phi$ is a good 2-D parity game, which Eve wins if and only if ϕ is satisfiable.

$\mathcal{G}_\phi$ is a good 2-D parity game. Let ρ be a play in $\mathcal{G}_\phi$ that satisfies the χ_2 parity condition. If $2c$ is the largest χ_2-priority occurring infinitely often in ρ, then by construction, $2c + 2$ is the largest χ_1-priority occurring infinitely often in the ρ, which is also even. Thus, ρ satisfies the χ_1 parity condition.

If ϕ is satisfiable, then Eve wins $\mathcal{G}_\phi$. Let $f : \{x_1, x_2, \cdots, x_M\} \to \{\top, \bot\}$ be a satisfying assignment of ϕ. Let σ be a function which assigns, to each term t_i, a literal $l \in t_i$ that is assigned $\top$ in f. Consider the Eve-strategy $\sigma_\exists$ in $\mathcal{G}_\phi$ defined by $\sigma_\exists(t) = (t, \sigma(l))$. We claim that $\sigma_\exists$ is a winning strategy. Indeed, let ρ be a play in $\mathcal{G}_\phi$ following $\sigma_\exists$, and consider the largest i such that x_i or $\neg x_i$ appear infinitely often in ρ. Since $\sigma_\exists$ is obtained from a satisfying assignment, we know that either only x_i appears infinitely often, or only $\neg x_i$ appears infinitely often. In the former case, the highest χ_2 priority appearing infinitely often is $2i$, which is even, and hence ρ is winning for Eve. In the latter case, the highest χ_1 priority appearing infinitely often is $2i+1$, which is odd, and hence the χ_1-parity condition is not satisfied, implying ρ is winning for Eve.

If Eve wins $\mathcal{G}_\phi$, then ϕ is satisfiable. If Eve wins $\mathcal{G}_\phi$, then we know she can win using a positional strategy. Let $\sigma_\exists : T \to L$ be such a strategy, where Eve chooses the edge $(t, \sigma_\exists(t))$ at a vertex t. If there are no two terms t, t' such that $\sigma_\exists(t) = x_i$ and $\sigma_\exists(t') = \neg x_i$ for some x_i, then we are done since then $\sigma_\exists$ can naturally be extended to a satisfying assignment for ϕ. Otherwise, we claim that Adam wins the game $\mathcal{G}_\phi$: Adam can alternate between picking t and t', and then the highest χ_1 priority appearing infinitely often is $2i + 2$, while the highest χ_2 priority appearing infinitely often is $2i + 1$. This implies that the play is winning for Adam, which is a contradiction since $\sigma_\exists$ is a winning strategy for Eve.

4.2   NP-hardness of checking history-determinism

We now show that deciding the history-determinism, whether Eve wins the 1-token game, and whether Eve wins the 2-token game of a given parity automaton

is **NP**-hard (Theorem 14). Much of the work towards this has already been done in the reduction from 2-D PARITY GAME to SIMULATION given in Section 3. We show that the automaton $\mathcal{H}$ that is constructed when using this reduction from a good 2-D parity game $\mathcal{G}$ is such that Eve wins $\mathcal{G}$ if and only if $\mathcal{H}$ is history-deterministic. Since GOOD 2-D PARITY GAME is **NP**-hard (Lemma 12), we get that HISTORY-DETERMINISTIC is **NP**-hard as well.

Lemma 13. *Checking whether a given nondeterministic parity automaton is history-deterministic is* **NP***-hard.*

Proof. Let us consider a good 2-D parity game $\mathcal{G}$. Recall the construction of the automata $\mathcal{H}$ and $\mathcal{D}$ in Section 3, which is such that Eve wins $\mathcal{G}$ if and only if $\mathcal{H}$ simulates $\mathcal{D}$. We will show that if $\mathcal{G}$ is a good 2-D parity game, then the following statements are equivalent.

1. Eve wins $\mathcal{G}$.
2. $\mathcal{H}$ simulates $\mathcal{D}$.
3. $\mathcal{H}$ is history-deterministic.

The equivalence of 1 and 3 would then conclude the proof. The equivalence of 1 and 2 has already been shown in the proof of Theorem 10, and we now focus on showing that 2 and 3 are equivalent.

Towards this, let $\Sigma = E \cup \{\$\}$, and consider the languages $\mathcal{L}_j$ over Σ consisting of the words $(e_i\$f_i)_{i\geq 0}$ such that $(e_i f_i)_{i\geq 0}$ is a play in $\mathcal{G}$ that satisfies χ_j, for $j = 1, 2$. By construction, we know $L(\mathcal{D}) = \mathcal{L}_1$, and $L(\mathcal{H}) = \mathcal{L}_1 \cup \mathcal{L}_2$. Furthermore, since $\mathcal{G}$ is good, we know that $\mathcal{L}_1 \supseteq \mathcal{L}_2$ and hence $L(\mathcal{D}) = L(\mathcal{H})$. Observe that by construction, $\mathcal{D}$ is deterministic.

If $\mathcal{H}$ is history-deterministic, then since $L(\mathcal{D}) = L(\mathcal{H})$, Eve wins the simulation game between $\mathcal{H}$ and $\mathcal{D}$: she can use her strategy in the letter game of $\mathcal{H}$ to play in $Sim(\mathcal{H}, \mathcal{D})$, ignoring Adam's transitions in $\mathcal{D}$.

Conversely, if $\mathcal{H}$ simulates $\mathcal{D}$, then she can use her winning strategy in $Sim(\mathcal{H}, \mathcal{D})$ to win the letter game of $\mathcal{H}$ as follows. Eve, during the letter game of $\mathcal{H}$, will keep in her memory, a play of the game $Sim(\mathcal{H}, \mathcal{D})$. On each round in the letter game of $\mathcal{H}$, Adam gives a letter, and Eve, in the game $Sim(\mathcal{H}, \mathcal{D})$, lets Adam pick the same letter and the unique transition on that letter in $\mathcal{D}$. She then uses her strategy in $Sim(\mathcal{H}, \mathcal{D})$ to pick a transition in $\mathcal{H}$, and she plays the same transition in the letter game of $\mathcal{H}$. We claim that any resulting play of the letter game of $\mathcal{H}$ if Eve plays as above is winning for Eve. Indeed, if Adam constructs an accepting word in $\mathcal{H}$, then it is accepting in $\mathcal{D}$ as well. Hence, since $\mathcal{D}$ is deterministic, Adam's run on $\mathcal{D}$ in the simulation game between $\mathcal{H}$ and $\mathcal{D}$ that is stored in Eve's memory is accepting. Since Eve is playing according to a winning strategy in $Sim(\mathcal{H}, \mathcal{D})$, Eve's run in $\mathcal{H}$, which is the same in $Sim(\mathcal{H}, \mathcal{D})$ and the letter game of $\mathcal{H}$, is accepting as well. Hence, Eve wins the letter game of $\mathcal{H}$, and thus $\mathcal{H}$ is history-deterministic.

We also argue in the appendix that the automaton $\mathcal{H}$ in proof of Lemma 13 above is such that Eve wins the 1-token game of $\mathcal{H}$ *if and only if* Eve wins the

2-token game of $\mathcal{H}$ *if and only if* $\mathcal{H}$ is history-deterministic. This gives us that checking whether Eve wins the 1-token game or the 2-token game of a parity automaton is **NP**-hard. Since 1-token games can naturally be seen as a 2-D parity game, we get that deciding whether Eve wins the 1-token game of a given parity automaton is in **NP**, and hence the problem is **NP**-complete.

Theorem 14. *The following problems are* **NP**-*hard:*

1. *Given a parity automaton $\mathcal{A}$, is $\mathcal{A}$ history-deterministic?*
2. *Given a parity automaton $\mathcal{A}$, does Eve win the 2-token game of $\mathcal{A}$?*

Additionally, the following problem is **NP**-*complete: Given a parity automaton $\mathcal{A}$, does Eve win the 1-token game of $\mathcal{A}$?*

5   Language Containment

In this section, we consider the following problem:

> HD-automaton containment: Given two parity automata $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{B}$ is history-deterministic, is $L(\mathcal{A}) \subseteq L(\mathcal{B})$?

While the problem of checking language inclusion between two non-deterministic parity automata is **PSPACE**-complete (regardless of whether the parity index is fixed or not) [31,1], the same for deterministic parity automata is **NL**-complete [37, Theorem 1]. For history-deterministic parity automata with fixed parity indices, however, the problem of language inclusion reduces to checking for simulation (Lemma 15), which can be solved in polynomial time when the parity indices are fixed [13].

Lemma 15 ([38,9]).   *Given a nondeterministic parity automaton $\mathcal{A}$ and a history-deterministic parity automaton $\mathcal{B}$, the following are equivalent:*

1. *$\mathcal{B}$ simulates $\mathcal{A}$*
2. *$L(\mathcal{A}) \subseteq L(\mathcal{B})$*

Proof. $(1) \Rightarrow (2)$: Fix $\sigma_\exists$ to be a winning strategy for Eve in $Sim(\mathcal{B}, \mathcal{A})$. Let w be a word accepted by $\mathcal{A}$ via an accepting run ρ. Consider a play of $Sim(\mathcal{B}, \mathcal{A})$ where Adam constructs the run ρ on the word w, and Eve plays according to $\sigma_\exists$. Then, the run in $\mathcal{B}$ that Eve constructs must be accepting, and hence w is accepted by $\mathcal{B}$.

$(2) \Rightarrow (1)$: Let σ_B be a winning strategy for Eve in the letter game of $\mathcal{B}$. Consider the strategy for Eve in $Sim(\mathcal{B}, \mathcal{A})$ where Eve chooses the transitions on $\mathcal{B}$ according to σ_B, ignoring Adam's transitions in $\mathcal{A}$. If Adam constructs an accepting run in $\mathcal{A}$ on a word w in $Sim(\mathcal{B}, \mathcal{A})$, then $w \in L(\mathcal{A}) \subseteq L(\mathcal{B})$. Hence σ_B would have constructed an accepting run in $\mathcal{B}$ in $Sim(\mathcal{B}, \mathcal{A})$. It follows that Eve wins $Sim(\mathcal{B}, \mathcal{A})$, and hence $\mathcal{B}$ simulates $\mathcal{A}$.

Corollary 16. *Given a nondeterministic parity automaton $\mathcal{A}$ and a history-deterministic parity automaton $\mathcal{B}$ such that both $\mathcal{A}$ and $\mathcal{B}$ have priorities in $[d]$ for a fixed d, the problem of whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$ can be decided in polynomial time.*

We now focus on the problem HD-AUTOMATON CONTAINMENT when the parity index is not fixed. From Lemma 15, we know that this can be reduced to SIMULATION. Since SIMULATION is in **NP** [13], we get an immediate **NP**-upper bound for HD-AUTOMATON CONTAINMENT [38, Lemma 3]. We show that we can do better, in quasi-polynomial time, by giving a polynomial time reduction to finding the winner in a parity game[11,28].

Towards this, let us fix a nondeterministic parity automaton $\mathcal{A}$ and a history-deterministic parity automaton $\mathcal{B}$ over the alphabet Σ throughout the rest of this section, for which we want to decide if $L(\mathcal{A}) \subseteq L(\mathcal{B})$. Suppose that $\mathcal{A}$ has n_1 states and priorities in $[d_1]$, and $\mathcal{B}$ has n_2 states and priorities in $[d_2]$.

It is well known that every such parity automaton $\mathcal{A}$ can be converted efficiently to a language-equivalent nondeterministic Büchi automaton $\mathcal{A}'$ that has at most $(n_1 \cdot d_1)$ states [14,39]. Then, from Lemma 15, it suffices to check if Eve wins the game $Sim(\mathcal{B}, \mathcal{A}')$. Note that $Sim(\mathcal{B}, \mathcal{A}')$ is a 2-D parity game $\mathcal{G}$ with $(n_1 \cdot d_1 \cdot n_2 \cdot |\Sigma|)$-many vertices that has the priority functions $\chi_1 : V \to [1,2]$ and $\chi_2 : V \to [d_2]$, where V is the set of vertices of $\mathcal{G}$.

The game $\mathcal{G}$ can be viewed equivalently as a Muller game with the condition $(C, \mathcal{F})$, where $C = [1,2] \times [d_2]$ and $\mathcal{F}$ consists of sets $F \subseteq C$ such that if $\max(F|_1)$ is even, then $\max(F|_2)$ is even. Here, $F|_i$ for $i \in \{1, 2\}$ indicates the projection of F onto the ith component. Call the Zielonka tree (Definition 1) of this Muller condition as Z_{d_2}. We shall show that the size of Z_{d_2} is polynomial in d_2.

Lemma 17. *The Zielonka tree Z_{d_2} has $(\lceil \frac{d_2}{2} \rceil)$ many leaves and its height is d_2.*

The proof of the lemma, obtained via an inductive argument, is given in the appendix. Lemma 17 allows us to use Lemma 3 on $Sim(\mathcal{B}, \mathcal{A}')$ to obtain an equivalent Parity game $\mathcal{G}'$ with $(n_1 \cdot d_1 \cdot n_2 \cdot |\Sigma| \cdot \lceil \frac{d_2}{2} \rceil)$ vertices which has $d_2 + 1$ priorities, such that Eve wins $Sim(\mathcal{B}, \mathcal{A}')$ if and only if Eve wins $\mathcal{G}'$.

Lemma 18. *Given a nondeterministic parity automaton $\mathcal{A}$ with n_1 states and a history-deterministic parity automaton $\mathcal{B}$ with n_2 states whose priorities are in $[d_2]$ that are both over the alphabet Σ, the problem of deciding whether $L(\mathcal{A})$ is contained in $L(\mathcal{B})$ can be reduced in polynomial time to finding the winner of a parity game $\mathcal{G}$ which has $(n_1 \cdot d_1 \cdot n_2 \cdot |\Sigma| \cdot \lceil \frac{d_2}{2} \rceil)$ many vertices and $d_2 + 1$ priorities.*

Since parity games can be solved in quasi-polynomial time[11,28], Lemma 18 implies that the problem of language containment in a history-deterministic automaton can be solved in quasi-polynomial time as well.

Theorem 19. *Given a nondeterministic parity automaton $\mathcal{A}$ with n_1 states and priorities in $[d_1]$, and a history-deterministic parity automaton $\mathcal{B}$ with n_2 states*

whose priorities are in $[d_2]$, checking whether the language of $\mathcal{A}$ is contained in the language of $\mathcal{B}$ can be done in time

$$(n_1 \cdot d_1 \cdot n_2 \cdot d_2 \cdot |\Sigma|)^{\mathcal{O}(\log d_2)}.$$

6   Discussion

We have shown **NP**-hardness for the problem of checking for simulation between two parity automata (when their parity indices are not fixed). We have also established upper and lower bounds of several decision problems relating to history-deterministic parity automata. The most significant amongst these, in our view, is the **NP**-hardness for the problem of deciding if a given parity automaton is history-deterministic, which is an improvement from the previous lower bound of solving a parity game [29].

There still remains a significant gap between the lower bound of **NP**-hardness and upper bound of **EXPTIME** for checking history-determinism, however. Furthermore, note that even if one shows the two-token conjecture [3,4], this would only imply a **PSPACE**-upper bound (when the parity index is not fixed), since 2-token games can be seen as Emerson-Lei games [26]. Thus, a natural direction for future research is to try to show that the problem of checking for history-determinism is **PSPACE**-hard.

On the other hand, however, it is also plausible that checking whether Eve wins the 2-token game of a given parity automaton can be done in **NP**. A proof for this might show that if Eve wins a 2-token game, then she has a strategy that can be represented and verified polynomially. Such an approach, which would involve understanding the strategies for the players in the 2-token games better, could also yield crucial insights for proving or disproving the two-token conjecture (see Section 2.9).

Boker and Lehtinen showed in their recent survey that for a 'natural' class of automata T, checking history-determinism for T-automata is at least as hard as solving T-games [8]. Interestingly, the problem of checking history-determinism over T-automata also has the matching upper bound of solving T-games for all classes of automata T over finite words, and over infinite words with safety and reachability objectives on which the notion of history-determinism has been studied so far [7,22,36,9,20]. Our result of the problem of checking history-determinism being **NP**-hard for parity automata deviates from this trend (unless parity games are **NP**-hard, which would have the drastic and unlikely consequence of **NP** = **NP** ∩ **coNP**), and demonstrates the additional intricacy that parity conditions bring.

References

1. Abdulla, P.A., Chen, Y., Clemente, L., Holík, L., Hong, C., Mayr, R., Vojnar, T.: Advanced ramsey-based büchi automata inclusion testing. In: Katoen, J., König, B. (eds.) CONCUR 2011 - Concurrency Theory - 22nd International Conference, CONCUR 2011, Aachen, Germany, September 6-9, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6901, pp. 187–202. Springer (2011). `https://doi.org/10.1007/978-3-642-23217-6_13`, `https://doi.org/10.1007/978-3-642-23217-6_13`

2. Abu Radi, B., Kupferman, O.: Minimization and canonization of GFG transition-based automata. Log. Methods Comput. Sci. **18**(3) (2022). `https://doi.org/10.46298/lmcs-18(3:16)2022`, `https://doi.org/10.46298/lmcs-18(3:16)2022`

3. Bagnol, M., Kuperberg, D.: Büchi good-for-games automata are efficiently recognizable. In: Ganguly, S., Pandya, P.K. (eds.) 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India. LIPIcs, vol. 122, pp. 16:1–16:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). `https://doi.org/10.4230/LIPIcs.FSTTCS.2018.16`, `https://doi.org/10.4230/LIPIcs.FSTTCS.2018.16`

4. Boker, U., Kuperberg, D., Lehtinen, K., Skrzypczak, M.: On the succinctness of alternating parity good-for-games automata. CoRR **abs/2009.14437** (2020), `https://arxiv.org/abs/2009.14437`

5. Boker, U., Kupferman, O., Skrzypczak, M.: How deterministic are good-for-games automata? In: Lokam, S.V., Ramanujam, R. (eds.) 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India. LIPIcs, vol. 93, pp. 18:1–18:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). `https://doi.org/10.4230/LIPIcs.FSTTCS.2017.18`, `https://doi.org/10.4230/LIPIcs.FSTTCS.2017.18`

6. Boker, U., Lehtinen, K.: History determinism vs. good for gameness in quantitative automata. In: Bojanczyk, M., Chekuri, C. (eds.) 41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference. LIPIcs, vol. 213, pp. 38:1–38:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). `https://doi.org/10.4230/LIPIcs.FSTTCS.2021.38`, `https://doi.org/10.4230/LIPIcs.FSTTCS.2021.38`

7. Boker, U., Lehtinen, K.: Token games and history-deterministic quantitative automata. In: Bouyer, P., Schröder, L. (eds.) Foundations of Software Science and Computation Structures - 25th International Conference, FOSSACS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13242, pp. 120–139. Springer (2022). `https://doi.org/10.1007/978-3-030-99253-8_7`, `https://doi.org/10.1007/978-3-030-99253-8_7`

8. Boker, U., Lehtinen, K.: When a little nondeterminism goes a long way: An introduction to history-determinism. ACM SIGLOG News **10**(1), 24–51 (2023). `https://doi.org/10.1145/3584676.3584682`, `https://doi.org/10.1145/3584676.3584682`

9. Bose, S., Henzinger, T.A., Lehtinen, K., Schewe, S., Totzke, P.: History-deterministic timed automata. CoRR **abs/2304.03183** (2023). `https://doi.org/10.48550/arXiv.2304.03183`, `https://doi.org/10.48550/arXiv.2304.03183`

10. Bose, S., Purser, D., Totzke, P.: History-deterministic vector addition systems. In: Pérez, G.A., Raskin, J. (eds.) 34th International Conference on Concurrency Theory, CONCUR 2023, September 18-23, 2023, Antwerp, Belgium. LIPIcs, vol. 279, pp. 18:1–18:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023). `https://doi.org/10.4230/LIPIcs.CONCUR.2023.18`, `https://doi.org/10.4230/LIPIcs.CONCUR.2023.18`

11. Calude, C.S., Jain, S., Khoussainov, B., Li, W., Stephan, F.: Deciding parity games in quasi-polynomial time. SIAM J. Comput. **51**(2), 17–152 (2022). `https://doi.org/10.1137/17m1145288`, `https://doi.org/10.1137/17m1145288`

12. Casares, A., Colcombet, T., Fijalkow, N.: Optimal transformations of games and automata using muller conditions. In: Bansal, N., Merelli, E., Worrell, J. (eds.) 48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference). LIPIcs, vol. 198, pp. 123:1–123:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). `https://doi.org/10.4230/LIPIcs.ICALP.2021.123`, `https://doi.org/10.4230/LIPIcs.ICALP.2021.123`

13. Chatterjee, K., Henzinger, T.A., Piterman, N.: Generalized parity games. In: Seidl, H. (ed.) Foundations of Software Science and Computational Structures, 10th International Conference, FOSSACS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007, Braga, Portugal, March 24-April 1, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4423, pp. 153–167. Springer (2007). `https://doi.org/10.1007/978-3-540-71389-0_12`, `https://doi.org/10.1007/978-3-540-71389-0_12`

14. Choueka, Y.: Theories of automata on omega-tapes: A simplified approach. J. Comput. Syst. Sci. **8**(2), 117–141 (1974). `https://doi.org/10.1016/S0022-0000(74)80051-6`, `https://doi.org/10.1016/S0022-0000(74)80051-6`

15. Colcombet, T.: The theory of stabilisation monoids and regular cost functions. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S.E., Thomas, W. (eds.) Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II. Lecture Notes in Computer Science, vol. 5556, pp. 139–150. Springer (2009). `https://doi.org/10.1007/978-3-642-02930-1_12`, `https://doi.org/10.1007/978-3-642-02930-1_12`

16. Cook, S.A.: The complexity of theorem-proving procedures. In: Harrison, M.A., Banerji, R.B., Ullman, J.D. (eds.) Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA. pp. 151–158. ACM (1971). `https://doi.org/10.1145/800157.805047`, `https://doi.org/10.1145/800157.805047`

17. Dziembowski, S., Jurdzinski, M., Walukiewicz, I.: How much memory is needed to win infinite games? In: Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997. pp. 99–110. IEEE Computer Society (1997). `https://doi.org/10.1109/LICS.1997.614939`, `https://doi.org/10.1109/LICS.1997.614939`

18. Emerson, E.A.: Automata, tableaux and temporal logics (extended abstract). In: Parikh, R. (ed.) Logics of Programs, Conference, Brooklyn College, New York, NY, USA, June 17-19, 1985, Proceedings. Lecture Notes in Computer Science, vol. 193, pp. 79–88. Springer (1985). `https://doi.org/10.1007/3-540-15648-8_7`, `https://doi.org/10.1007/3-540-15648-8_7`

19. Emerson, E.A., Jutla, C.S.: The complexity of tree automata and logics of programs. SIAM J. Comput. **29**(1), 132–158 (1999). `https://doi.org/10.1137/S0097539793304741`, `https://doi.org/10.1137/S0097539793304741`

20. Erlich, E., Guha, S., Jecker, I., Lehtinen, K., Zimmermann, M.: History-deterministic parikh automata. In: Pérez, G.A., Raskin, J. (eds.) 34th International Conference on Concurrency Theory, CONCUR 2023, September 18-23, 2023, Antwerp, Belgium. LIPIcs, vol. 279, pp. 31:1–31:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023). `https://doi.org/10.4230/LIPIcs.CONCUR.2023.31`, `https://doi.org/10.4230/LIPIcs.CONCUR.2023.31`

21. Fearnley, J., Jain, S., de Keijzer, B., Schewe, S., Stephan, F., Wojtczak, D.: An ordered approach to solving parity games in quasi-polynomial time and quasi-linear space. Int. J. Softw. Tools Technol. Transf. **21**(3), 325–349 (2019). `https://doi.org/10.1007/s10009-019-00509-3`, `https://doi.org/10.1007/s10009-019-00509-3`

22. Guha, S., Jecker, I., Lehtinen, K., Zimmermann, M.: A bit of nondeterminism makes pushdown automata expressive and succinct. In: Bonchi, F., Puglisi, S.J. (eds.) 46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia. LIPIcs, vol. 202, pp. 53:1–53:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). `https://doi.org/10.4230/LIPIcs.MFCS.2021.53`, `https://doi.org/10.4230/LIPIcs.MFCS.2021.53`

23. Gurevich, Y., Harrington, L.: Trees, automata, and games. In: Lewis, H.R., Simons, B.B., Burkhard, W.A., Landweber, L.H. (eds.) Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA. pp. 60–65. ACM (1982). `https://doi.org/10.1145/800070.802177`, `https://doi.org/10.1145/800070.802177`

24. Henzinger, T.A., Kupferman, O., Rajamani, S.K.: Fair simulation. Inf. Comput. **173**(1), 64–81 (2002). `https://doi.org/10.1006/inco.2001.3085`, `https://doi.org/10.1006/inco.2001.3085`

25. Henzinger, T.A., Piterman, N.: Solving games without determinization. In: Ésik, Z. (ed.) Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4207, pp. 395–410. Springer (2006). `https://doi.org/10.1007/11874683_26`, `https://doi.org/10.1007/11874683_26`

26. Hunter, P., Dawar, A.: Complexity bounds for regular games. In: Jedrzejowicz, J., Szepietowski, A. (eds.) Mathematical Foundations of Computer Science 2005, 30th International Symposium, MFCS 2005, Gdansk, Poland, August 29 - September 2, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3618, pp. 495–506. Springer (2005). `https://doi.org/10.1007/11549345_43`, `https://doi.org/10.1007/11549345_43`

27. Jurdzinski, M.: Deciding the winner in parity games is in UP \cap co-up. Inf. Process. Lett. **68**(3), 119–124 (1998). `https://doi.org/10.1016/S0020-0190(98)00150-1`, `https://doi.org/10.1016/S0020-0190(98)00150-1`

28. Jurdzinski, M., Lazic, R.: Succinct progress measures for solving parity games. In: 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017. pp. 1–9. IEEE Computer Society (2017). `https://doi.org/10.1109/LICS.2017.8005092`, `https://doi.org/10.1109/LICS.2017.8005092`

29. Kuperberg, D., Skrzypczak, M.: On determinisation of good-for-games automata. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9135, pp. 299–310. Springer (2015). `https://doi.org/10.1007/978-3-662-47666-6_24`, `https://doi.org/10.1007/978-3-662-47666-6_24`

30. Kupferman, O.: Using the past for resolving the future. Frontiers Comput. Sci. **4** (2022). `https://doi.org/10.3389/fcomp.2022.1114625`, `https://doi.org/10.3389/fcomp.2022.1114625`

31. Kupferman, O., Vardi, M.Y.: Verification of fair transition systems. Chic. J. Theor. Comput. Sci. **1998** (1998), `http://cjtcs.cs.uchicago.edu/articles/1998/2/contents.html`

32. Lehtinen, K., Parys, P., Schewe, S., Wojtczak, D.: A recursive approach to solving parity games in quasipolynomial time. Log. Methods Comput. Sci. **18**(1) (2022). `https://doi.org/10.46298/lmcs-18(1:8)2022`, `https://doi.org/10.46298/lmcs-18(1:8)2022`

33. Lehtinen, K., Zimmermann, M.: Good-for-games ω-pushdown automata. Log. Methods Comput. Sci. **18**(1) (2022). `https://doi.org/10.46298/lmcs-18(1:3)2022`, `https://doi.org/10.46298/lmcs-18(1:3)2022`

34. Martin, D.A.: Borel determinacy. Annals of Mathematics **102**(2), 363–371 (1975), `http://www.jstor.org/stable/1971035`

35. Milner, R.: An algebraic definition of simulation between programs. In: Cooper, D.C. (ed.) Proceedings of the 2nd International Joint Conference on Artificial Intelligence. London, UK, September 1-3, 1971. pp. 481–489. William Kaufmann (1971), `http://ijcai.org/Proceedings/71/Papers/044.pdf`

36. Prakash, A., Thejaswini, K.S.: On history-deterministic one-counter nets. In: Kupferman, O., Sobocinski, P. (eds.) Foundations of Software Science and Computation Structures - 26th International Conference, FoSSaCS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22-27, 2023, Proceedings. Lecture Notes in Computer Science, vol. 13992, pp. 218–239. Springer (2023). `https://doi.org/10.1007/978-3-031-30829-1_11`, `https://doi.org/10.1007/978-3-031-30829-1_11`

37. Schewe, S.: Beyond hyper-minimisation—minimising dbas and dpas is np-complete. In: Lodaya, K., Mahajan, M. (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India. LIPIcs, vol. 8, pp. 400–411. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2010). `https://doi.org/10.4230/LIPIcs.FSTTCS.2010.400`, `https://doi.org/10.4230/LIPIcs.FSTTCS.2010.400`

38. Schewe, S.: Minimising Good-For-Games Automata Is NP-Complete. In: Saxena, N., Simon, S. (eds.) Proceedings of FSTTCS. Leibniz International Proceedings in Informatics (LIPIcs), vol. 182, pp. 56:1–56:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2020). `https://doi.org/10.4230/LIPIcs.FSTTCS.2020.56`, `https://drops.dagstuhl.de/opus/volltexte/2020/13297`

39. Seidl, H., Niwinski, D.: On distributive fixed-point expressions. RAIRO Theor. Informatics Appl. **33**(4/5), 427–446 (1999). `https://doi.org/10.1051/ita:1999101`, `https://doi.org/10.1051/ita:1999101`

40. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time: Preliminary report. In: Aho, A.V., Borodin, A., Constable, R.L., Floyd, R.W., Harrison, M.A., Karp, R.M., Strong, H.R. (eds.) Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin,

Texas, USA. pp. 1–9. ACM (1973). https://doi.org/10.1145/800125.804029, https://doi.org/10.1145/800125.804029

A   Appendix for Section 4

Lemma 20. *The following problems are* **NP***-hard:*

1. *Given a parity automaton $\mathcal{A}$, does Eve win the 1-token game of $\mathcal{A}$?*
2. *Given a parity automaton $\mathcal{A}$, does Eve win the 2-token game of $\mathcal{A}$?*

Proof. As in the proof of Lemma 13, consider a good 2-D parity game $\mathcal{G}$, and let $\mathcal{H}$ and $\mathcal{D}$ be automaton constructed in Section 3, which is such that Eve wins $\mathcal{G}$ if and only if $\mathcal{H}$ simulates $\mathcal{A}$. From Lemma 13, we also know that Eve wins $\mathcal{G}$ if and only if $\mathcal{H}$ is HD. We will show that the following holds:

1. if $\mathcal{H}$ is history-deterministic, then Eve wins the 1-token game and the 2-token game of $\mathcal{H}$,
2. if $\mathcal{H}$ is not history-deterministic, then Adam wins the 1-token game and the 2-token game of $\mathcal{H}$.

(1) is easy to see, since if $\mathcal{H}$ is history-deterministic, then Eve can use her letter game strategy to pick transitions on her token in the 1-token game or the 2-token game, ignoring Adam's tokens.

For (2), we show that Adam wins the 1-token game of $\mathcal{H}$ if $\mathcal{H}$ is not HD. This would imply that Adam wins the 2-token game of $\mathcal{H}$ as well, since he can use his strategy in 1-token game to win the 2-token game as follows: he copies Eve's transitions in his second token and follows his strategy in the 1-token game of $\mathcal{H}$ to choose the letters and transitions on his first token.

Since Adam wins the letter game of $\mathcal{H}$, we know that $\mathcal{H}$ does not simulate $\mathcal{D}$. Fix a winning strategy $\sigma_\forall$ of Adam in the simulation game. We now describe how Adam can win the 1-token game of $\mathcal{H}$ by using $\sigma_\forall$. At a high level, Adam will exploit the nondeterminism on \$ to ensure his token eventually moves to $\mathcal{D}$ and traces out an accepting run, while picking letters according to $\sigma_\forall$ and ensuring Eve's run on her token is rejecting.

In more details, Adam in the 1-token game will keep a play of the game $Sim(\mathcal{H}, \mathcal{D})$ in his memory in order to pick the letters and transitions in the 1-token game of $\mathcal{H}$. He picks letters in the 1-token game of $\mathcal{H}$ using $\sigma_\forall$, by viewing Eve's transitions in the 1-token game as transitions on $\mathcal{H}$ in $Sim(\mathcal{H}, \mathcal{D})$. Note that since $\mathcal{D}$ is deterministic, Adam's transitions in $\mathcal{D}$ in $Sim(\mathcal{H}, \mathcal{D})$ depends solely on Adam's choice of letters. For Adam's token in the 1-token game, he copies Eve's transitions till Eve hasn't had to resolve nondeterminism, i.e., there is a unique transition from Eve's state on Adam's letter in the corresponding round. Eventually, however, there must be a round where Eve would need to resolve nondeterminism: otherwise, since $L(\mathcal{D}) = L(\mathcal{H})$, Eve would construct an accepting run if the word is accepting, and in particular, if Adam's run on his token is accepting.

Thus, let Eve's token be at the state q when she needed to resolve the nondeterminism. Then, Adam's token is also at q, and by construction, we know that $q = v_H$ for some $v \in V_\exists$. In the corresponding simulation game $Sim(\mathcal{H}, \mathcal{D})$ in Adam's memory, his state is $v_\$$. Now, Adam gives the letter \$, and his state

in the simulation game is then updated to v_D, while Eve picks a transition $v_H \xrightarrow{\$:0} (v_H, f)$ where $f = (v, w)$ is an outgoing edge from v. Adam will then pick a transition $v_H \xrightarrow{\$:0} (v_H, f')$ for some edge $f' \neq f$ on his token.

In the next round, Adam must pick the letter f according to $\sigma_\forall$, or Adam's state and Eve's state in $Sim(\mathcal{H}, \mathcal{D})$ would both be the same and in $\mathcal{D}$, causing Eve to win, which is a contradiction since $\sigma_\forall$ is a winning strategy. This causes Adam's state in $\mathcal{D}$ in $Sim(\mathcal{H}, \mathcal{D})$ to be w_D. In the 1-token game, Eve's token goes to w_H and Adam's token goes to w_D—same as his state in $Sim(\mathcal{H}, \mathcal{D})$. From here, Adam can continue choosing letters according to $\sigma_\forall$, while his transitions are uniquely determined. Since $\sigma_\forall$ is a winning strategy, Eve's run on her token is rejecting, while Adam's run ρ in $\mathcal{D}$ in $Sim(\mathcal{H}, \mathcal{D})$ is accepting. Since the run of Adam's token eventually coincides with ρ, it is accepting as well, and hence Adam wins the 1-token game.

B   Appendix for Section 5

Lemma 17. *The Zielonka tree Z_{d_2} has $(\lceil \frac{d_2}{2} \rceil)$ many leaves and its height is d_2.*

Proof. We prove the lemma by induction on d_2. When d_2 is 0 or 1, the tree Z_{d_2} is as shown below in Fig. 2, and the induction hypothesis is clearly satisfied.
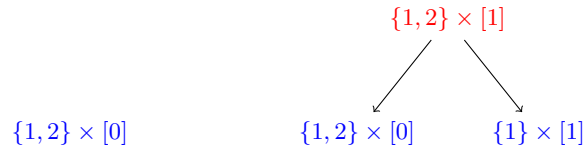

Fig. 2. Zielonka trees Z_0 and Z_1

Let $d_2 \geq 2$, and suppose that the lemma holds for all $d' < d_2$. We distinguish between the cases of when d_2 is even or odd.
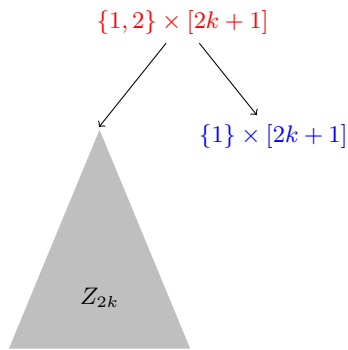
If $d_2 = 2k$ for some $k > 0$, then the root of the node is labelled by the set $[1, 2] \times [2k]$, which is in $\mathcal{F}$. Then, the only maximal set which is a proper subset of $[1, 2] \times [2k]$ and not in $\mathcal{F}$ is $[1, 2] \times [2k - 1]$. Thus, the child c of the root is labelled $[1, 2] \times [2k - 1]$ and c is then the root of the tree Z_{2k-1} itself (see Fig. 3). By induction hypothesis, the tree Z_{2k-1} has height $2k - 1$ and k leaves, and hence, the tree Z_{2k} has height $2k - 1 + 1 = 2k$ and k leaves, as desired.

If $d_2 = 2k + 1$ for some $k > 0$, then the root of the node is labelled by the set $[1, 2] \times [2k + 1]$, which is not in $\mathcal{F}$. Now, there are two maximal sets that are proper subsets of $[1, 2] \times [2k + 1]$ and are in $\mathcal{F}$: the set $\{1\} \times [2k + 1]$ which then has no proper subsets that are not in $\mathcal{F}$, and the set $\{1, 2\} \times [2k]$. Accordingly, the root of Z_{2k+1} has two children, one labelled by $\{1, 2\} \times [2k]$ that is then the the root of the tree Z_{2k}, and another labelled $\{1\} \times [2k + 1]$, as shown in Fig. 4.

$\{1, 2\} \times [2k]$
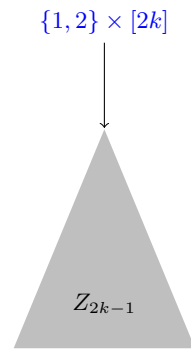
Fig. 3. Zielonka tree Z_{2k}

By induction hypothesis, the tree Z_{2k} has height $2k$ and k leaves, and hence, the tree Z_{2k+1} has height $2k + 1$ and $k + 1$ leaves, as desired.

$\{1, 2\} \times [2k + 1]$

$\{1\} \times [2k + 1]$

Z_{2k}

Fig. 4. Zielonka tree Z_{2k+1}