Lookahead Games and Efficient Determinisation of History-Deterministic Büchi Automata

Rohan Acharya
University of Warwick, Coventry, UK

Marcin Jurdziński
University of Warwick, Coventry, UK

Aditya Prakash
University of Warwick, Coventry, UK

Abstract

Our main technical contribution is a polynomial-time determinisation procedure for history-deterministic Büchi automata, which settles an open question of Kuperberg and Skrzypczak, 2015. A key conceptual contribution is the lookahead game, which is a variant of Bagnol and Kuperberg's token game, in which Adam is given a fixed lookahead. We prove that the lookahead game is equivalent to the 1-token game. This allows us to show that the 1-token game characterises history-determinism for semantically-deterministic Büchi automata, which paves the way to our polynomial-time determinisation procedure.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects

Keywords and phrases History determinism, Good-for-games, Automata over infinite words, Games

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.124

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: https://arxiv.org/abs/2404.17530 [2]

Funding We acknowledge the Centre for Discrete Mathematics and Its Applications (DIMAP) at the University of Warwick for partial support.
Rohan Acharya: Undergraduate Research Support Scheme and the Department of Computer Science, University of Warwick.
Aditya Prakash: Chancellors' International Scholarship from the University of Warwick.

Acknowledgements We thank Udi Boker, Denis Kuperberg, and Karoliina Lehtinen for several insightful exchanges. We are grateful to the reviewers for their feedback and suggestions on how to improve the paper.

1 Introduction

History-deterministic (HD) automata are non-deterministic automata in which the non-determinism can be resolved "on the fly", based only on the prefix of the word read so far [6, 15]. This concept can be formalised using the history-determinism game (HD game), in which two players Adam and Eve make alternating moves choosing letters and transitions, thus constructing a word and a run of the automaton on it, respectively. Eve wins if the run is accepting or if the word is not in the language, and hence Eve's winning strategy will successfully resolve non-determinism by constructing an accepting run on the fly, for all words in the language. An automaton is then defined to be history-deterministic if Eve has a winning strategy in the game.

Henzinger and Piterman [11] introduced HD automata because of their potential to speed up key algorithmic tasks in verification and synthesis, such as language containment and strategy synthesis. In language containment, we ask whether all executions of an implementation $\mathcal{A}$ satisfy a specification $\mathcal{H}$. If $\mathcal{H}$ is non-deterministic then the problem is **PSPACE**-hard, but if $\mathcal{H}$ is HD then it is more tractable, because it amounts to checking that $\mathcal{H}$ simulates $\mathcal{A}$. This can be done in polynomial time if the parity index of $\mathcal{H}$ is fixed [8, Theorem 3] and in quasi-polynomial time otherwise [18, Theorem 20]. Henzinger and Piterman had originally dubbed HD automata as good-for-games automata in their work because games whose winning conditions are represented by an HD automaton can be solved efficiently without automaton determinisation [11, Theorem 3.1], a well-known computational bottleneck in synthesis.

1.1 Related work

Key questions studied for HD parity automata include recognising them, their succinctness relative to deterministic automata, minimisation, and determinisation.

Recognising History-Deterministic Automata via Games. Kuperberg and Skrzypczak [14] gave a polynomial time algorithm to recognise HD co-Büchi automata, and Bagnol and Kuperberg [3] gave a polynomial time algorithm to recognise HD Büchi automata. These algorithms have been conceptually unified by Boker, Kuperberg, Lehtinen, and Skrzypczak [5] to be based on the 2-token game introduced by Bagnol and Kuperberg [3], leading to the 2-token conjecture.

▶ **Conjecture 1** (The 2-token conjecture [3, 5])**.** *A parity automaton is HD if and only if Eve wins the 2-token game on it.*

Proving the 2-token conjecture would imply that recognising HD parity automata of fixed parity index can be done in polynomial time. In contrast, the best upper bound currently known for the problem is **EXPTIME**, dating back to Henzinger and Piterman [11]. In the general case, when the parity index is not fixed, a lower bound of **NP**-hardness has been achieved only very recently [18].

Bagnol and Kuperberg [3] introduced the k-token game as a tool to characterise the conceptually more complex HD game. Like in the HD game, in the k-token game for $k \geq 1$, two players Adam and Even make alternating moves choosing letters and transitions, but in addition, in every round, after Eve chooses a transition, Adam also chooses k transitions. As a result, Adam constructs a word and k runs, Eve constructs a run, and Eve wins if her run is accepting or all of Adam's k runs are rejecting. A key insight in Bagnol and Kuperberg's work [3] is that the 2-token game is equivalent to the k-token game for all $k \geq 2$.

▶ **Theorem 2** ([3])**.** *Eve wins the 2-token game on a parity automaton if and only if for all $k \geq 2$, Eve wins the k-token game on it.*

Bagnol and Kuperberg's proof that the 2-token game characterises history-determinism for Büchi automata exploits this insight, by showing that if Adam wins the HD game on a Büchi automaton then he can win the k-token game for some k that is doubly-exponential in the size of the automaton, and hence also the 2-token game.

Boker et al. [5] have used an analogous, but more involved, argument to show that the 2-token game also characterises history-determinism for co-Büchi automata, combining Theorem 2 with the algorithm of Kuperberg and Skrzypczak [14] to recognise HD co-Büchi automata efficiently, which was based on the so-called Joker game. The Joker game is similar to the 1-token game but, additionally, Adam has the power to (finitely many times) "play Joker" by choosing a transition from Eve's token instead of a transition from his token, and Eve wins if her run is accepting, or Adam's run is rejecting, or Adam has played Joker infinitely many times.

Kuperberg and Skrzypczak's algorithm uses Joker games in their polynomial time algorithm to recognise HD co-Büchi automata, but to date, it was not known if Joker games characterise history-determinism on co-Büchi or Büchi automata, or on parity automata in general.

Succinctness and minimisation of HD automata. Kuperberg and Skrzypczak [14] proved that HD co-Büchi automata are exponentially more succinct than deterministic co-Büchi automata [14], which is tight [15, Theorem 4.1]. Abu Radi and Kupferman [1] showed that transition-based HD co-Büchi automata can be minimised in polynomial time and that they have canonicity. In contrast, minimisation of state-based HD Büchi or HD co-Büchi automata is **NP**-complete [22, Theorem 1]. Minimisation of transition-based Büchi automata is easily seen to be in **NP**, but the exact complexity is open.

Determinisation of HD Büchi automata. Kuperberg and Skrzypczak [14] also proved that every HD Büchi automaton with n states has an equivalent deterministic Büchi automaton with at most n^2 states. However, it is not known if HD Büchi automata are strictly more succinct than deterministic Büchi automata.

The determinisation procedure of Kuperberg and Skrzypczak for an HD Büchi automaton $\mathcal{H}$ involves carefully analysing the simulation game between $\mathcal{H}$ and an equivalent deterministic Büchi automaton of exponential size. At a high level, the procedure iteratively modifies the simulation game and the automaton $\mathcal{H}$, eventually yielding an equivalent game of quadratic size, from which a deterministic Büchi automaton of quadratic size can be extracted, but the procedure itself runs in exponential time.

Kuperberg and Skrzypczak also gave a non-deterministic polynomial-time procedure for determinisation of HD Büchi automata, which guesses a deterministic Büchi automaton of quadratic size and then checks for language equivalence [14, Theorem 10]. They left the exact complexity of determinisation for HD Büchi automata open, in particular, the question of whether HD Büchi automata can be determinised in polynomial time.

1.2 Our Contributions

- We introduce the k-lookahead game, a variant of the 1-token game, in which Adam's transition on his token is delayed by k steps, thus giving him a lookahead of k. We prove that the 1-token game is equivalent to the k-lookahead game.

  ▶ **Theorem A.** *For every parity automaton $\mathcal{A}$, Eve wins the 1-lookahead game on $\mathcal{A}$ if and only if she wins the k-lookahead game on $\mathcal{A}$.*

  The 1-token game is syntactically equivalent to the 1-lookahead game. Theorem A thus demonstrates that the 1-token game is already quite powerful, and it is analogous to Theorem 2 of Bagnol and Kuperberg.
- With Theorem A as a key tool, we show that the 1-token game characterises history-determinism on semantically-deterministic Büchi automata. These are automata in which, for every state, all transitions labelled by the same letter lead to language-equivalent states [20].

  ▶ **Theorem B.** *A semantically-deterministic Büchi automaton is history-determinsitic if and only if Eve wins the 1-token game on it.*

  A consequence of Theorem B is that Joker games characterise history-determinism on Büchi automata (Theorem 19). Since Joker games have smaller arenas than 2-token games, this leads to a more efficient algorithm for recognising HD Büchi automata (Lemma 20).

- We give a parity automaton with priorities 1, 2, and 3 on which Eve wins the Joker game but that is not HD (Theorem 23). This implies that the Joker game does not characterise history-determinism for parity automata and that Theorem B does not extend to parity automata.
- We give a polynomial time determinisation procedure for HD Büchi automata, thus resolving an open question of Kuperberg and Skrzypczak [14].

  ▶ **Theorem C.** *There is a polynomial-time procedure that converts every HD Büchi automaton with n states into an equivalent deterministic Büchi automaton with n^2 states.*

  Our determinisation procedure is inspired by that of Kuperberg and Skrzypczak [14], but rather than working with the simulation game between the automaton and a deterministic automaton of exponential size, thanks to Theorem B, we can work with the 1-token game instead. This results in an algorithm that is conceptually simpler and that runs in polynomial time.
- We also give a technique to reduce game-based characterisations of history-determinism to universal automata (automata that accept all words). Hence to prove the 1-token game characterisation of history-determinism for semantically-deterministic (SD) Büchi automata, it suffices to prove it for universal SD Büchi automata (Theorem 11). Likewise, to prove the 2-token conjecture for parity automata, it suffices to prove it for universal parity automata (Theorem 13).

2 Preliminaries

We let $\mathbb{N} = \{0, 1, 2, \cdots\}$ to be the set of natural numbers. For two natural numbers i, j such that $i < j$, we write $[i, j]$ to denote the set of integers $\{i, i+1, \ldots, j\}$, and $[j]$ to denote $[0, j]$. 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 be the unique word of length 0. A language $\mathcal{L} \in \Sigma^\omega$ is a set of infinite words. For a finite word $u \in \Sigma^*$ and a language $\mathcal{L}$, we define $u^{-1}\mathcal{L}$ to be $\{w \mid uw \in \mathcal{L}\}$.

2.1 Games

Game arenas. An *arena* is a directed graph $G = (V, E)$ with vertices partitioned into $V_\forall$ 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 vertices in $V_\exists$ are owned by Eve and those in $V_\forall$ are owned by Adam.

A *play* on this arena is an infinite path starting at v_0 and it is formed as follows. A play starts with a token at v_0 and it proceeds for infinitely many 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 $\rho \in L$, and Adam wins otherwise. A *strategy* for Eve in such a game $\mathcal{G}$ is a function from the set of plays that end at an Eve's vertex to an outgoing edge from that vertex. Eve's strategy is said to be winning if any play produced while she plays according to this strategy is winning for her. 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 deal with ω-regular games, which are known to be determined [17, 10], i.e., each game has a winner. Two games are *equivalent* if they have the same winner.

Parity games. An $[i,j]$-*parity game* $\mathcal{G}$ is played over a finite game arena $G = (V, E)$, with the edges of G labelled by a priority function $\chi : E \to [i,j]$ for some $i, j \in \mathbb{N}$ with $i < j$, and $i = 0$ or $i = 1$. A play ρ in the arena of $\mathcal{G}$ is winning for Eve if and only if the highest edge priority that occurs infinitely often is even. It is well known that parity games can be solved in polynomial time when the interval $[i,j]$ is fixed, and in quasi-polynomial time otherwise [7, 13, 16].

2.2 Automata

Parity automata. An $[i,j]$-*non-deterministic parity automaton* $\mathcal{A} = (Q, \Sigma, q_0, \Delta)$ consists of a finite directed graph with edges labelled by letters in Σ and *priorities* in $[i,j]$ for some $i, j \in \mathbb{N}$ with $i < j$. These edges are called *transitions*, which are elements of the set $\Delta \subseteq Q \times \Sigma \times [i,j] \times Q$, and the vertices of this graph are called *states*, which are elements of the set Q. Each automaton has a designated *initial state* $q_0 \in Q$. For states $p, q \in Q$ and a letter $a \in \Sigma$, 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 write that such a run is *accepting* if it *satisfies the parity condition*, i.e., the highest priority occurring infinitely often amongst the transitions of the run is even, 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 $\mathcal{L}(\mathcal{A})$, is the set of words that it accepts. We write that the automaton $\mathcal{A}$ *recognises* a language $\mathcal{L}$ if $\mathcal{L}(\mathcal{A}) = \mathcal{L}$. A language $\mathcal{L} \subseteq \Sigma^\omega$ is ω-*regular* if it is recognised by some parity automaton. A parity automaton $\mathcal{A}$ is *deterministic* if for any given state in $\mathcal{A}$ and any given letter in Σ, there is at most one outgoing transition from that state on that letter.

We write that $[i,j]$, with $i = 0$ or 1, is the parity index of $\mathcal{A}$. A *Büchi* (resp. co-Büchi) automaton is a $[1,2]$ (resp. $[0,1]$) parity automaton. A *safety automaton* is one where all transitions have priority 0.

We write $(\mathcal{A}, q)$ to denote the automaton $\mathcal{A}$ with q as its initial state, and $\mathcal{L}(\mathcal{A}, q)$ to denote the language it recognises. Two states p and q in $\mathcal{A}$ are *equivalent* if $\mathcal{L}(\mathcal{A}, p) = \mathcal{L}(\mathcal{A}, q)$.

History-determinism. The (HD game) is a two 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 HD if Eve has a winning strategy in the history-determinism game.

▶ **Definition 3** (History-determinism game). *Given a parity automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta)$, the* history-determinism game *of $\mathcal{A}$ is defined between the players Adam and Eve as follows, with positions in Q. The game starts at q_0 and proceeds for infinitely many rounds. For each $i \in \mathbb{N}$, round i starts at a position $q_i \in Q$, and proceeds as follows:*

1. *Adam selects a letter $a_i \in \Sigma$;*
2. *Eve selects a transition $q_i \xrightarrow{a_i:c_i} q_{i+1} \in \Delta$.*

The new position is q_{i+1} from where round $(i+1)$ is played. Thus, the play of a history-determinism 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 $\mathcal{L}(\mathcal{A})$, then Eve's run is accepting.

If Eve wins the history-determinism game on $\mathcal{A}$, then we say that $\mathcal{A}$ is *history-deterministic*.

Semantic-determinism. Let $\mathcal{A}$ be a parity automaton. A transition δ from p to q on a letter a in $\mathcal{A}$ is called *language-preserving* if $\mathcal{L}(\mathcal{A}, q) = a^{-1}\mathcal{L}(\mathcal{A}, p)$. We say that a parity automaton is *semantically-deterministic*, SD for short, if all transitions in it are language-preserving. The following lemma can be shown by a simple inductive argument on the length of words.

▶ **Lemma 4.** *If a parity automaton is SD then all states in the automaton that can be reached from a fixed state q upon reading a finite word u accept the language $u^{-1}\mathcal{L}(\mathcal{A}, q)$.*

SD automata were introduced by Kuperberg and Skrzypczak as residual automata [14]. We follow Abu Radi, Kupferman, and Leshkowitz [21] by calling them SD automata instead.

2.3 Games on Automata

Simulation and simulation-like games (such as token games [3]) are fundamental amongst the techniques we use in this paper. We define these games below.

Simulation and stepahead simulation.

▶ **Definition 5** (Simulation game). *Given two parity automata $\mathcal{A} = (P, \Sigma, p_0, \Delta_A)$ and $\mathcal{B} = (Q, \Sigma, q_0, \Delta_B)$, the* simulation game *between $\mathcal{B}$ and $\mathcal{A}$ is a two player game played between Eve and Adam as follows, with positions in $P \times Q$. The game starts at (p_0, q_0), and proceeds for infinitely many rounds. For each $i \geq 0$, round i starts at position (p_i, q_i) and proceeds as follows:*

1. *Adam selects a letter $a_i \in \Sigma$;*
2. *Adam selects a transition $p_i \xrightarrow{a_i:c'} p_{i+1}$ in $\mathcal{A}$;*
3. *Eve selects a transition $q_i \xrightarrow{a_i:c} q_{i+1}$ in $\mathcal{B}$.*

At the end of a play of the simulation game, the letters selected by Adam in sequence form a word, while the sequence of his selected transitions and the sequence of Eve's selected transitions form runs on that word in $\mathcal{A}$ and $\mathcal{B}$, respectively. We say Eve wins the game if her run in $\mathcal{B}$ is accepting or Adam's run in $\mathcal{A}$ is rejecting. If Eve has a strategy to win the simulation game, then we say that $\mathcal{B}$ simulates *$\mathcal{A}$.*

The *stepahead simulation* game between $\mathcal{B}$ and $\mathcal{A}$ is defined similarly to the simulation game, except the orders of move in each round are changed as follows: Adam selects a letter first, then Eve selects a transition on $\mathcal{B}$, and then Adam selects a transition on $\mathcal{A}$. The winning condition is identical, which is that Eve's run on $\mathcal{B}$ is accepting if Adam's run on $\mathcal{A}$ is accepting. If Eve wins the stepahead simulation game between $\mathcal{B}$ and $\mathcal{A}$, then we say that $\mathcal{B}$ *step-ahead simulates* $\mathcal{A}$.

Token games. k-token games are similar to stepahead simulation games, but are played on a single automaton, and Adam constructs k runs instead of one. The winning objective of Eve requires her to construct an accepting run if one of k Adam's runs is accepting.

▶ **Definition 6** (k-token game). *The* k-token game *on a non-deterministic parity automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta)$ is defined between the 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 round i 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.*

1. *Adam selects a letter $a_i \in \Sigma$.*
2. *Eve selects a transition $q_i \xrightarrow{a_i:c} q_{i+1} \in \Delta$.*
3. *Adam selects k transitions $p_i^1 \xrightarrow{a_i:c_1'} p_{i+1}^1, p_i^2 \xrightarrow{a_i:c_2'} p_{i+1}^2, \cdots p_i^k \xrightarrow{a_i:c_k'} p_{i+1}^k \in \Delta$.*

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

Thus, in a play of the k-token game on $\mathcal{A}$, 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.

Observe that the stepahead-simulation game between $\mathcal{A}$ and itself is equivalent to the 1-token game on $\mathcal{A}$.

Joker games. *Joker games* are defined similar to 1-token games, but additionally in each round, Adam can choose to play *Joker* and choose a transition from Eve's position instead of a transition from his position. The winning condition for Eve is the following: If Adam's sequence of transitions satisfies the parity conditions and Adam has played finitely many Jokers, then Eve's run is accepting as well.

▶ **Definition 7** (Joker games). *The* Joker game *on a non-deterministic parity automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta)$ is defined between the players Adam and Eve as follows, with positions in $Q \times Q$. The game starts at (q_0, q_0) and proceeds in ω many rounds. For each $i \in \mathbb{N}$, the round i starts at a position $(q_i, p_i) \in Q \times Q$, and proceeds as follows.*

1. *Adam selects a letter $a_i \in \Sigma$.*
2. *Eve selects a transition $q_i \xrightarrow{a_i:c_i} q_{i+1} \in \Delta$*
3. *Adam either selects a transitions $p_i \xrightarrow{a_i:c'_i} p_{i+1}$, or plays Joker and selects a transition $q_i \xrightarrow{a_i:c'_i} p_{i+1}$.*

The new position is (q_{i+1}, p_{i+1}), from where round $(i + 1)$ begins.

Eve wins such a play if the following holds: if Adam plays finitely many Jokers and his sequence of transitions satisfies the parity condition, then Eve's run is accepting.

The following observations are easy to see.

▶ **Lemma 8.** *If $\mathcal{A}$ is an HD parity automaton, and if $\mathcal{B}$ is a parity automaton such that $\mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}(\mathcal{A})$, then we have:*

1. *Eve wins the Joker game on $\mathcal{A}$ and the k-token game on $\mathcal{A}$, for all $k \geq 1$;*
2. *$\mathcal{A}$ simulates and step-ahead simulates $\mathcal{B}$.*

Proof. Fix a winning strategy σ for Eve in the HD game of $\mathcal{A}$. Consider the strategy for Eve in the above games, in which she follows σ based on the letters Adam chooses, ignoring the rest of his moves. Then Eve constructs an accepting run whenever the word constructed by Adam is accepting. In particular, if Adam constructs an accepting run in the k-token game or the (stepahead) simulation game, then his word must be in $\mathcal{L}(\mathcal{A})$, implying Eve's run is accepting as well.

Similarly, in a play of the Joker game, suppose Adam plays finitely many Jokers and his sequence of transitions satisfies the parity condition. Let i be the last round where Adam played a Joker. Then there is an accepting run on Adam's word, which can be obtained by concatenating Eve's run until round $(i - 1)$ with Adam's run from round i. This implies that Adam's word is in $\mathcal{L}(\mathcal{A})$, once again implying that Eve's run is accepting. ◀

2.4 History-Deterministic Automata and Simulation

Let $\mathcal{L}$ be an ω-regular language and let $\mathcal{F}_\mathcal{L}$ be the set of automata whose recognized languages are subsets of $\mathcal{L}$, that is $\mathcal{F}_\mathcal{L} = \{\mathcal{A} \mid \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}\}$. For $\mathcal{A}, \mathcal{B} \in \mathcal{F}_\mathcal{L}$, we define $\mathcal{A} \preceq \mathcal{B}$ to hold if $\mathcal{B}$ simulates $\mathcal{A}$. This relation $\preceq$ is called the *simulation preorder* because it is reflexive and transitive. Lemma 8 implies that every HD automaton $\mathcal{H}$ that recognises $\mathcal{L}$ is a *greatest* element in $\mathcal{F}_\mathcal{L}$ with respect to the simulation preorder, that is, we have $\mathcal{A} \preceq \mathcal{H}$ for all $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{H})$. We show that the converse also holds.

▶ **Lemma 9.** *Let $\mathcal{L}$ be an ω-regular language and let $\mathcal{F}_{\mathcal{L}} = \{\mathcal{A} \mid \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}\}$. Then, an automaton $\mathcal{H}$ in $\mathcal{F}_{\mathcal{L}}$ is greatest w.r.t. the simulation preorder if and only if $\mathcal{H}$ recognises $\mathcal{L}$ and it is history-deterministic.*

Proof. We only need to prove the forward implication, since the backward implication follows from Lemma 8. Let $\mathcal{H} \in \mathcal{F}_{\mathcal{L}}$ be such that $\mathcal{A} \preceq \mathcal{H}$ for all automata $\mathcal{A}$ that satisfy $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}$. Fix $\mathcal{D}$ to be a deterministic parity automaton that recognises $\mathcal{L}$ (such a $\mathcal{D}$ always exists, see [9, Theorem 1.19 and 3.11]). Then, in particular, we have $\mathcal{D} \preceq \mathcal{H}$. Observe that this implies $\mathcal{L}(\mathcal{H}) \supseteq \mathcal{L}(\mathcal{D}) = \mathcal{L}$, and since $\mathcal{L}(\mathcal{H}) \subseteq \mathcal{L}$, we get that $\mathcal{L}(\mathcal{H}) = \mathcal{L}$.

We proceed to show that Eve wins the HD game on $\mathcal{H}$. Fix σ to be a winning strategy for Eve in the simulation game between $\mathcal{H}$ and $\mathcal{D}$. We use σ to construct a winning strategy in the HD game on $\mathcal{H}$ as follows. During the letter game on $\mathcal{H}$, Eve keeps a corresponding play of the simulation game between $\mathcal{H}$ and $\mathcal{D}$, where Adam is playing the same letters as the HD game and choosing the unique transitions available to him. Then Eve chooses transitions according to σ in the HD game and in the simulation game in her memory. This way, whenever Adam's word w in the HD game is in $\mathcal{L}(\mathcal{H}) = \mathcal{L}$, then the unique run in $\mathcal{D}$ on w is accepting, and hence, Eve's run in the HD game on $\mathcal{H}$ must be accepting as well. ◀

We make explicit a corollary of the above lemma.

▶ **Corollary 10** ([11, Theorem 4])**.** *If a nondeterministic parity automaton $\mathcal{A}$ simulates a language-equivalent history-deterministic automaton $\mathcal{H}$ then $\mathcal{A}$ is history-deterministic.*

Proof. Let $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{H}) = \mathcal{L}$. From Lemma 9, we know that $\mathcal{H}$ is a greatest element in $\mathcal{F}_{\mathcal{L}}$, and $\mathcal{H} \preceq \mathcal{A}$ implies that $\mathcal{A}$ is a greatest element in $\mathcal{F}_{\mathcal{L}}$ as well. It follows from Lemma 9 that $\mathcal{A}$ is history-deterministic. ◀

3 Sufficient to think about Universal Automata

In the next section, we will show that 1-token games characterise history-determinism on semantically-deterministic Büchi automata (Theorem B). In order to show this, we start by reducing this result to the restriction where our automata are universal (Theorem 11), i.e., recognise all words in the language. A very similar reduction also shows that proving the 2-token conjecture for universal parity is sufficient to conclude the 2-token conjecture for parity automata (Theorem 19).

▶ **Theorem 11.** *The following statements are equivalent:*

1. *For any semantically-deterministic Büchi automaton $\mathcal{A}$, Eve wins the 1-token game on $\mathcal{A}$ if and only if $\mathcal{A}$ is history-deterministic.*
2. *For any semantically-deterministic Büchi automaton $\mathcal{U}$ with $\mathcal{L}(\mathcal{U}) = \Sigma^\omega$, Eve wins the 1-token game on $\mathcal{U}$ if and only if $\mathcal{U}$ is history-deterministic.*

We shall use the following fact shown by Boker, Henzinger, Lehtinen, and Prakash to prove Theorem 11 [4].

▶ **Lemma 12** ([4])**.** *A non-deterministic parity automaton $\mathcal{A}$ is history-deterministic if and only if it simulates all deterministic safety automata $\mathcal{S}$ with $\mathcal{L}(\mathcal{S}) \subseteq \mathcal{L}(\mathcal{A})$.*

A proof of Lemma 12 can be found in the full version [2]. We now show Theorem 11.

Proof sketch for Theorem 11. It is clear that 1 implies 2. For the other direction, suppose 2 holds. Let $\mathcal{A}$ be a semantically-deterministic automaton that is not HD. We will show that Adam wins the 1-token game on $\mathcal{A}$.

From Lemma 12, we know that there is a deterministic safety automaton $\mathcal{S}$ such that $\mathcal{A}$ does not simulate $\mathcal{S}$ and $\mathcal{L}(\mathcal{S}) \subseteq \mathcal{L}(\mathcal{A})$. Consider the product safety automaton $\mathcal{P}$ of $\mathcal{S}$ and $\mathcal{A}$ which recognises the language $\mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{S})$. We then complete $\mathcal{P}$ by adding an accepting sink state f, and transitions to f from all states q on letters a such that q did not have an outgoing transition on a in $\mathcal{P}$. We call this automaton $\mathcal{U}$. It is clear that $\mathcal{L}(\mathcal{U}, p) = \Sigma^\omega$ for all states p in $\mathcal{U}$, and hence $\mathcal{U}$ is SD. We show that Adam wins the HD game on $\mathcal{U}$, by using his winning strategy in the simulation game between $\mathcal{A}$ and $\mathcal{S}$ (recall that $\mathcal{P}$ was constructed by taking product of $\mathcal{S}$ and $\mathcal{A}$). The hypothesis implies that Adam wins the 1-token game on $\mathcal{U}$. We then show that we can adapt a winning strategy for Adam on 1-token game of $\mathcal{U}$ to one for the 1-token game on $\mathcal{A}$, by simply "projecting" his strategy to the $\mathcal{A}$ component: note that since $\mathcal{S}$ is deterministic, in plays of the 1-token game on $\mathcal{U}$, Eve's and Adam's states have the same $\mathcal{S}$-component at the start of each round. ◀

An almost word-by-word identical proof to above also shows that the 2-token conjecture can be reduced to the case where the automata are universal.

▶ **Theorem 13.** *The following statements are equivalent:*

1. *For any non-deterministic parity automaton $\mathcal{A}$, Eve wins the 2-token game on $\mathcal{A}$ if and only if $\mathcal{A}$ is history-deterministic.*
2. *For any non-deterministic parity automaton $\mathcal{U}$ with $\mathcal{L}(\mathcal{U}) = \Sigma^\omega$, Eve wins the 2-token game on $\mathcal{U}$ if and only if $\mathcal{U}$ is history-deterministic.*

4 When 1-Token Game is Enough

In this section, we will show the following result.

▶ **Theorem B.** *A semantically-deterministic Büchi automaton is history-determinsitic if and only if Eve wins the* 1*-token game on it.*

Towards this, we first introduce k-lookahead games, which are variants of 1-token games where Adam is given a lookahead of k.

4.1 Lookahead Games

Let us briefly recall how a round of the 1-token game on a parity automaton $\mathcal{A}$ works. In each round, Adam selects a letter, then Eve selects a transition on that letter on her token, and then Adam selects a transition on that letter on his token. The winning condition for Eve is that at the end of the play, either Eve's run is accepting or Adam's run is rejecting. This is very close to the simulation game between $\mathcal{A}$ and itself, except that the order of the moves in which Eve and Adam select transitions has been reversed. One can, however, see the 1-token game as a simulation game, where Adam picks the transition for round i in round $(i + 1)$. Or equivalently, we can construct an automaton $\mathsf{Delay}(\mathcal{A})$ such that any non-determinism on $\mathcal{A}$ is "delayed" by one step, and then the 1-token game on $\mathcal{A}$ is equivalent to the simulation game between $\mathcal{A}$ and $\mathsf{Delay}(\mathcal{A})$. This insight was used by Prakash and Thejaswini to give an algorithm for deciding history-determinism of one-counter nets, by reducing the 1-token game to a simulation game [19]. Below we give a construction Delay on parity automata that delays the non-determinism by one-step, inspired by a similar construction for one-counter nets [19, Lemma 11].

▶ **Definition 14.** *For any non-deterministic parity automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta)$, the automaton* $\mathsf{Delay}(\mathcal{A})$ *is constructed so that it runs "one letter behind" $\mathcal{A}$, by storing a letter in its state space. More formally,* $\mathsf{Delay}(\mathcal{A}) = (Q', \Sigma, s, \Delta')$*, where $Q' = Q \times \Sigma \cup \{s\}$, and s is the initial state. The set of transitions Δ' is the union of the following sets of transitions.*
1. $\{(s \xrightarrow{a:0} (q_0, a)) \mid a \in \Sigma\}$.
2. $\{((p, a) \xrightarrow{a':c} (q, a')) \mid \ (p \xrightarrow{a:c} q) \in \Delta\}$.

Observe that $\mathsf{Delay}(\mathcal{A})$ accepts the same language as $\mathcal{A}$. The following lemma is easy to prove, since the expanded game arenas of the 1-token game on $\mathcal{A}$, and the simulation game between $\mathcal{A}$ and $\mathsf{Delay}(\mathcal{A})$ are equivalent, with identical winning conditions.

▶ **Lemma 15.** *For every non-deterministic parity automaton $\mathcal{A}$, Eve wins the* 1*-token game on $\mathcal{A}$ if and only if $\mathcal{A}$ simulates* $\mathsf{Delay}(\mathcal{A})$*.*

Furthermore, we can also show that Eve wins the 1-token game on $\mathsf{Delay}(\mathcal{A})$ if Eve wins the 1-token game on $\mathcal{A}$, by simply "delaying" her winning strategy in the 1-token game on $\mathcal{A}$.

▶ **Lemma 16.** *If Eve wins the* 1*-token game on an automaton $\mathcal{A}$, then Eve wins the* 1*-token game on* $\mathsf{Delay}(\mathcal{A})$*.*

An iterative application of Lemmas 15 and 16 gives us the following corollary.

▶ **Corollary 17.** *If Eve wins the* 1*-token game on a parity automaton $\mathcal{A}$, then $\mathcal{A}$ simulates* $\mathsf{Delay}^k(\mathcal{A})$ *for all $k \in \mathbb{N}$.*

Proof. Note that simulation relation is transitive, i.e., if $\mathcal{A}_0$ simulates $\mathcal{A}_1$ and $\mathcal{A}_1$ simulates $\mathcal{A}_2$, then $\mathcal{A}_0$ simulates $\mathcal{A}_2$. Suppose Eve wins the 1-token game on $\mathcal{A}$. From Lemma 16, induction gives us that Eve wins the 1-token game on $\mathsf{Delay}^k(\mathcal{A})$ for all $k \in \mathbb{N}$. From Lemma 15, we see that $\mathsf{Delay}^k(\mathcal{A})$ simulates $\mathsf{Delay}^{k+1}(\mathcal{A})$ for all $k \in \mathbb{N}$. Combining this with transitivity of simulation, we get that $\mathcal{A}$ simulates $\mathsf{Delay}^k(\mathcal{A})$ for all $k \in \mathbb{N}$. ◀

Call the simulation game between $\mathcal{A}$ and $\mathsf{Delay}^k(\mathcal{A})$ as the *k-lookahead game* on $\mathcal{A}$. Note that the 1-token game of $\mathcal{A}$ is then equivalent to the 1-lookahead game of $\mathcal{A}$. Corollary 17 can thus be restated as the following theorem.

▶ **Theorem A.** *For every parity automaton $\mathcal{A}$, Eve wins the* 1*-lookahead game on $\mathcal{A}$ if and only if she wins the k-lookahead game on $\mathcal{A}$.*

4.2 Games to Characterise History-Determinism

We now proceed to show Theorem B. From Theorem 11, we know that it suffices to only consider SD Büchi automata that are universal. The following lemma shows that every universal SD Büchi automaton is history-deterministic with sufficient lookahead.

▶ **Lemma 18.** *Let $\mathcal{U}$ be a semantically-deterministic Büchi automata such that $\mathcal{L}(\mathcal{U}) = \Sigma^\omega$. Then, there is a K such that* $\mathsf{Delay}^K(\mathcal{U})$ *is history-deterministic.*

Proof sketch. We let $K = 2^n$, where n is the number of states of $\mathcal{A}$. The crucial observation is that since $\mathcal{L}(\mathcal{U}, q) = \Sigma^\omega$ for any state q in $\mathcal{U}$, any finite word u that has length at least 2^n must have a run from q that passes through an accepting transition. Eve thus wins the history-determinism game on $\mathsf{Delay}^K(\mathcal{U})$ by exploiting the lookahead of K to take at least one accepting transition every K steps. The run of Eve's token then has infinitely many accepting transitions and hence is accepting, as desired. ◀

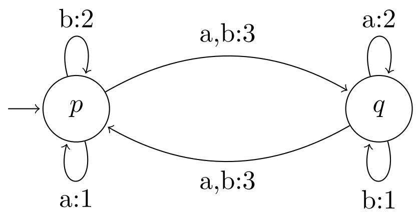

Figure 1 An [1,3]-automaton $\mathcal{A}$ that is not HD but on which Eve wins the Joker game.

We can now prove Theorem B.

Proof of Theorem B. The forward implication is clear by Lemma 8. For the backward direction, suppose that Eve wins the 1-token game on $\mathcal{A}$. Due to Theorem 11, we may assume that $\mathcal{A}$ is universal. Then, from Lemma 18, we know that there is a K such that $\mathsf{Delay}^K(\mathcal{A})$ is history-deterministic. If Eve wins the 1-token game on $\mathcal{A}$, then from Lemma 15, we know that $\mathcal{A}$ simulates $\mathsf{Delay}^K(\mathcal{A})$. But since $\mathsf{Delay}^K(\mathcal{A})$ is language equivalent to $\mathcal{A}$, we get from Corollary 10 that $\mathcal{A}$ is HD as well. ◀

Having shown that the 1-token game on a semantically-deterministic Büchi automaton $\mathcal{A}$ is equivalent to the HD game on $\mathcal{A}$, we are able to show that Joker games characterise history-determinism on Büchi automata. We also get an alternate proof of Bagnol and Kuperberg's result of 2-token games characterising history-determinism of Büchi automata as a corollary.

▶ **Theorem 19.** *For every Büchi automaton $\mathcal{A}$, the following statements are equivalent.*
1. *$\mathcal{A}$ is history-deterministic.*
2. *Eve wins the Joker game on $\mathcal{A}$.*
3. *Eve wins the 2-token game on $\mathcal{A}$.*

We prove Theorem 19 by reducing it to SD automata [2, Lemma 33], similar to Bagnol and Kuperberg [3, Lemma 16].

Joker games on a Büchi automaton have smaller arenas than 2-token games. As a result, we get a more efficient algorithm to recognise HD Büchi automata.

▶ **Lemma 20.** *Given a non-deterministic Büchi automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta)$, we can decide whether it is history-deterministic in time $\mathcal{O}(|\Sigma|^2|Q|^3|\Delta|)$.*

▶ Remark 21. While we have proven Theorem B by reducing it to universal automata for the sake of simplicity of arguments, one can also give a more direct proof. Such a direct proof involves arguing that if an automaton $\mathcal{A}$ is not history-deterministic, then there is a K exponential in the size of $\mathcal{A}$ such that $\mathcal{A}$ does not simulate $\mathsf{Delay}^K(\mathcal{A})$. It then follows from Lemma 15 that Eve loses the 1-token game on $\mathcal{A}$, as desired.

To argue that $\mathcal{A}$ does not simulate $\mathsf{Delay}^K(\mathcal{A})$, we reason based on the size of Adam's strategy in the history-determinism game on $\mathcal{A}$. Adam, in the simulation game between $\mathcal{A}$ and $\mathsf{Delay}^K(\mathcal{A})$, can pick letters according to his strategy in the history-determinism game on $\mathcal{A}$, thus ensuring Eve produces a rejecting run on her token. At the same time, Adam can exploit the lookahead in $\mathsf{Delay}^K(\mathcal{A})$ to construct an accepting run on his token, thus winning the simulation game between $\mathcal{A}$ and $\mathsf{Delay}^K(\mathcal{A})$.

We end this section by showing that unlike Büchi automata, Joker games do not characterise history-determinism on parity automata.

▶ **Example 22.** Consider the $[1, 3]$-automaton $\mathcal{A}$ shown in Figure 1. Note that $\mathcal{A}$ accepts all words in $\{a, b\}^\omega$, and is SD. It is easy to see that the automaton $\mathcal{A}$ is not history-deterministic, since Adam can win the history-determinism game on $\mathcal{A}$ by choosing the letter a when Eve's token is at p, and b when her token is at q. This forces Eve to never see an even transition, causing her run to be rejecting.

Eve wins the Joker game on $\mathcal{A}$, however. Consider the strategy of Eve where she switches states if her and Adam's tokens are at different states, and otherwise stays at the same state. For Adam to win, Adam must play only finitely many Jokers, and construct an accepting run, which requires him to eventually stay in the same state. But then, Eve's strategy will ensure that Eve's and Adam's run are eventually identical, thus ensuring Adam cannot win the Joker game of $\mathcal{A}$.

▶ **Theorem 23.** *Joker games do not characterise history-determinism on (semantically-deterministic) parity automata.*

5 Determinising HD Büchi Automata in Polynomial Time

In this section, we present a polynomial time determinisation procedure for HD Büchi automata with only a quadratic state-space blowup. Our procedure combines ideas from the exponential-time procedure given by Kuperberg and Skrzypczak [14] with our 1-token game characterisation of history-determinism on SD Büchi automata.

▶ **Theorem C.** *There is a polynomial-time procedure that converts every HD Büchi automaton with n states into an equivalent deterministic Büchi automaton with n^2 states.*

Let $\mathcal{H}$ be an HD Büchi automaton. Kuperberg and Skrzypczak's procedure relies on carefully analysing the simulation game between $\mathcal{H}$ and an equivalent deterministic Büchi automaton $\mathcal{D}$ whose states are the subsets of $\mathcal{H}$. Their procedure iteratively makes modifications to $\mathcal{H}$ and $\mathcal{D}$ based on the structure of Eve's winning strategies in this game or, more precisely, the progress measures [12] or the signatures [23] of the vertices in this game, which they call ranks [14]. The end result of their procedure is a relation between states of $\mathcal{H}$ and states of $\mathcal{D}$ that correspond to sets containing a singleton in $\mathcal{H}$. A product construction based on this relation allows them to construct an equivalent determinsitic Büchi automaton whose number of states is quadratic in the number of states of $\mathcal{A}$.

The reason Kuperberg and Skrzypczak work with the simulation game between $\mathcal{H}$ and $\mathcal{D}$ is that it characterises the history-determinism of $\mathcal{H}$. A non-deterministic automaton is HD if and only if it simulates an equivalent deterministic one, as shown by Hezinger and Piterman [11, Theorem 4]. Our result on 1-token games characterising history-determinism (Theorem B) allows us to instead work with the 1-token game on $\mathcal{H}$. This results in a conceptually simpler procedure that works in polynomial time.

We present our algorithm in two steps. First, we introduce HD Büchi automata that have a sprint self-simulation, and we give a polynomial time determinisation procedure for them. The procedure involves a quadratic state-space blowup. Then, we give a polynomial time iterative procedure to transform $\mathcal{H}$ into an equivalent HD Büchi automaton of the same size that has a sprint self-simulation. Overall, this gives us a polynomial time procedure to determinise HD Büchi automata with a quadratic state-space blowup.

5.1 Determinising Automata with Sprint Self-Simulation

In this subsection, we describe what automata with sprint self-simulation are and give a polynomial time determinisation procedure for such HD Büchi automata. The key concept here is the *sprint simulation* relation between two Büchi automata, characterised by the *sprint* *step-ahead simulation game*. This game is similar to the step-ahead simulation game, but the winning condition for Eve is that she must see an accepting transition before Adam does, i.e., she is in a sprint with Adam to see an accepting transition in the step-ahead simulation game first. Sprint step-ahead simulation (game) would be a more accurate phrase, but for brevity, we shorten it to just sprint simulation (game).

▶ **Definition 24.** *For two non-deterministic Büchi automata $\mathcal{A} = (Q, \Sigma, q_0, \Delta_A)$ and $\mathcal{B} = (P, \Sigma, p_0, \Delta_B)$, the* sprint simulation game *between $\mathcal{A}$ and $\mathcal{B}$ is played on the set of positions $Q \times P$ and it proceeds in rounds. In each round $i = 0, 1, 2, \dots,$ from position (q_i, p_i), the two players Adam and Eve make the following choices:*

1. *Adam chooses a letter $a_i \in \Sigma$;*
2. *Eve chooses a transition $\delta_i = (q_i \xrightarrow{a_i:c_i} q_{i+1}) \in \Delta_A$;*
3. *Adam chooses a transition $\delta'_i = (p_i \xrightarrow{a_i:c'_i} p_{i+1}) \in \Delta_B$.*

The new position is (q_{i+1}, p_{i+1}). At every round i, if transition δ_i is accepting then Eve wins the game, and otherwise, if the transition δ'_i is accepting then Adam wins the game. If neither δ_i nor δ'_i are accepting transitions, then the game continues for another round. Eve wins every infinite play.

Observe that if Eve wins the sprint simulation game then she can do so by a positional strategy, because the objective for Eve is a disjunction of a safety and a reachability objective, which can be seen as a $[0, 1]$-parity game. If Eve has a winning strategy in the above game, we say that $\mathcal{A}$ sprint simulates $\mathcal{B}$. The sprint simulation relation is transitive, i.e., if $\mathcal{A}$ sprint simulates $\mathcal{B}$ and $\mathcal{B}$ sprint simulates $\mathcal{C}$, then $\mathcal{A}$ sprint simulates $\mathcal{C}$ [2, Lemma 36].

▶ Remark 25. The sprint simulation relation is similar to the 'dependency' relation introduced by Kuperberg and Skrzypczak [14, Definition 30]. While the sprint simulation relation is between two Büchi automata, dependency relation is derived from the sprint simulation game between a Büchi automaton and an equivalent exponential-sized deterministic Büchi automaton.

We say that an HD Büchi automaton $\mathcal{H}$ has a *sprint self-simulation* if it is semantically-deterministic and for every state p in $\mathcal{H}$, there is a language-equivalent state q, such that $(\mathcal{H}, p)$ sprint simulates $(\mathcal{H}, q)$. When $\mathcal{H}$ is clear from the context, we will just say that p sprint simulates q. For the rest of this subsection, fix $\mathcal{H} = (Q, \Sigma, q_0, \Delta)$ to be an HD Büchi automaton that has a sprint self-simulation. The following lemma follows from transitivity of sprint simulation.

▶ **Lemma 26.** *For every state p in $\mathcal{H}$, there is a language-equivalent state q such that p sprint simulates q and q sprint simulates itself.*

Proof. Fix a state p in $\mathcal{H}$. Then, there is a language-equivalent state q_0 in $\mathcal{H}$ such that p sprint simulates q_0. If q_0 does not sprint simulates itself, there is another language-equivalent q_1 such that q_0 sprint simulates q_1. Repeating this argument, we get a sequence of states $q_0, q_1, q_2, \dots,$ such that q_i sprint simulates q_{i+1}. Since, there are finitely many states in $\mathcal{H}$, there are two natural numbers $i < j$ such that $q_i = q_j$. Due to transitivity of sprint simulation [2, Lemma 36], it follows that p sprint simulates q_i and q_i sprint simulates itself, as desired. ◀

Let us call a state q *sprint deterministic* if the automaton $\mathcal{H}$ can be determinised by deleting transitions to get a deterministic subautomaton $\mathcal{F}_q$ so that the following holds: for all finite words w that have a run in $\mathcal{H}$ starting at q going through an accepting transition, the unique run on w in $\mathcal{F}_q$ from q also sees an accepting transition. Thus, for states q that are sprint deterministic, there is a uniform strategy that achieves the objective of seeing an accepting transition as soon as possible on all words. We say that the automaton $\mathcal{F}_q$ as above is a witness for sprint determinism of q.

▶ **Lemma 27.** *A state q in $\mathcal{H}$ is sprint deterministic if and only if q sprint simulates itself. Moreover, there is a deterministic subautomaton $\mathcal{F}$ that can be computed in polynomial time and is a witness for all sprint deterministic states.*

Lemmas 26 and 27 above tell us that every state in $\mathcal{H}$ is either sprint deterministic, or it sprint simulates a state that is sprint deterministic. Fix a subautomaton $\mathcal{F}$ from Lemma 27, and a positional Eve strategy τ in the step-ahead simulation game from (p, q) for all pairs of states (p, q) such that p sprint simulates q.

The deterministic automaton $\mathcal{D}$ that is language-equivalent to $\mathcal{H}$ is then constructed to consist of pairs of states (p, q) such that p and q are language equivalent, p sprint simulates q, and q is sprint deterministic. Note that the pair of such states can be found in polynomial time, since checking for language containment on HD Büchi automata [18, Corollary 17] and deciding the winner of sprint simulation game can be done in polynomial time. Furthermore, for each state p in $\mathcal{H}$, we know from Lemmas 26 and 27 that there is a state q in $\mathcal{H}$ such that (p, q) is a state in $\mathcal{D}$. We let the initial state d_0 be (q_0, r_0) for some r_0 such that $(q_0, r_0) \in \mathcal{D}$.

At a state (q, p), the transitions in $\mathcal{D}$ from the second component p are chosen according to transitions from $\mathcal{F}$, while transitions from q are chosen via the positional Eve strategy τ. When an accepting transition $q \xrightarrow{a:2} q'$ is taken on the first component, we update the second component deterministically to be p' such that (q', p') is a state in $\mathcal{D}$. Or, equivalently, q' and p' are language equivalent, q' sprint simulates p', and p' is sprint deterministic. The priorities of transitions in $\mathcal{D}$ are the priorities of transitions of the first component.

We show the correctness of our construction using the definition of sprint simulation game and the fact that $\mathcal{H}$ is semantically-deterministic.

▶ **Lemma 28.** *The automaton $\mathcal{D}$ accepts the same language as $\mathcal{H}$.*

Proof. $\mathcal{L}(\mathcal{D}) \subseteq \mathcal{L}(\mathcal{H})$: If ρ is an accepting run of a word w in $\mathcal{D}$, then the projection of ρ on the first component is an accepting run in $\mathcal{H}$ as well.

$\mathcal{L}(\mathcal{H}) \subseteq \mathcal{L}(\mathcal{D})$: We show that for each state (p, q) in $\mathcal{D}$, if $w \in \mathcal{L}(\mathcal{H}, p)$ then the run from (p, q) on w in $\mathcal{D}$ sees an accepting transition eventually. We can then conclude by induction and the semantic determinism of $\mathcal{H}$ that runs of $\mathcal{D}$ on all accepting words in $\mathcal{L}(\mathcal{H})$ contain infinitely many accepting transitions each, and hence are accepting. To see this, let $w \in \mathcal{L}(\mathcal{H}, p)$ and let ρ_D be the run of $\mathcal{D}$ on w from (p, q). By construction of $\mathcal{D}$, we know that q is language-equivalent to p. Since q is sprint deterministic, the second component of ρ_D in $\mathcal{D}$ must contain an accepting transition on w eventually. But since p sprint simulates q, the run on the first component of ρ_D contains an accepting transition as well, as desired. ◀

5.2 Towards Automata with Sprint Self-Simulation

We now present a polynomial time algorithm to convert an HD Büchi automaton into an equivalent HD Büchi automaton that has a sprint self-simulation. Throughout this subsection, let $\mathcal{H} = (Q, \Sigma, q_0, \Delta)$ be an HD Büchi automaton.

We say that $\mathcal{H}$ is *good* if $\mathcal{H}$ is semantically-deterministic and Eve wins the Joker game from all states in $\mathcal{H}$. Every HD Büchi automaton $\mathcal{H}$ can be converted to an equivalent good HD Büchi automaton in polynomial time: we fix a winning strategy τ for Eve on the Joker game on $\mathcal{H}$, and consider the subautomaton $\mathcal{H}_N$ consisting of transitions that Eve takes according to τ [2, Lemma 38]. We thus assume without loss of generality that $\mathcal{H}$ is good.

To get an HD Büchi automaton equivalent to $\mathcal{H}$ and that has a sprint self-simulation, we iteratively make modifications to $\mathcal{H}$ based on the ranks of the 1-token game on $\mathcal{H}$. We first give a description of the 1-token game on a Büchi automaton as a $[0,2]$-parity game, and briefly recall the properties of ranks that we need on such games.

▶ **Definition 29.** *For a semantically-deterministic Büchi automaton $\mathcal{B} = (Q, \Sigma, q_0, \Delta)$, define the $[0,2]$-parity game $G_1(\mathcal{B}) = (V, E)$ as follows:*

- *The set of vertices V consists of the set $V = V_1 \cup V_2 \cup V_3$, where:*
  1. $V_1 = \{(p,q) \mid$ *p, q are states reachable from q_0 upon reading the same word w*$\}$
  2. $V_2 = \{(p,a,q) \mid (p,q) \in V_1\}$
  3. $V_3 = \{(p',q,a) \mid (p,a,q) \in V_2$ *and* $p \xrightarrow{a} p' \in \Delta\}$

  Eve's vertices are $V_\exists = V_2$, while Adam's vertices are $V_\forall = V_1 \cup V_3$
- *The set of edges E is the union of following sets:*
  1. $E_1 = \{(p,q) \to (p,a,q) \mid a \in \Sigma\}$ *(Adam chooses a letter)*
  2. $E_2 = \{(p,a,q) \to (p',q,a) \mid p \xrightarrow{a:c} p' \in \Delta\}$ *(Eve chooses a transition on her token)*
  3. $E_3 = \{(p',q,a) \to (p',q') \mid q \xrightarrow{a:c} q' \in \Delta\}$ *(Adam chooses a transition on his token)*
- *The priority function Ω is defined as follows. All elements in E_1 are assigned priority 0, while edges $(p,a,q) \to (p',q,a)$ in E_2 are assigned priority 2 if the transition $\delta = p \xrightarrow{a:c} p'$ in $\mathcal{B}$ is accepting (or equivalently, $c = 2$), and 0 otherwise. The edge $(p',q,a) \to (p',q')$ in E_3 is assigned priority 1 if the transition $q \xrightarrow{a:c} q'$ is accepting, and 0 otherwise.*

Observe that since $\mathcal{H}$ is SD, we have $\mathcal{L}(\mathcal{H}, p) = \mathcal{L}(\mathcal{H}, q)$ if (p,q) is a vertex in $G_1(\mathcal{H})$.

Ranks. We now define the ranks of a $[0,2]$-parity game $\mathcal{G}$. For each vertex v in $\mathcal{G}$, $\mathrm{rank}(v)$ is the largest number of 1's that Adam can guarantee Eve will see before seeing a 2 in the play (or only 0's) starting from v.

Observe that Eve wins such a parity game from every position if and only if the ranks of all the vertices are bounded. If this is the case, then there is a positional winning strategy τ, using which Eve can guarantee that she sees at most $\mathrm{rank}(v)$ many 1's before seeing a 2 (or seeing 0's forever) [23, Lemma 8] in every play. We will call such a strategy *optimal*. For $[0,2]$-parity games with n vertices and m edges, an optimal strategy can be computed in time $\mathcal{O}(mn)$ [12, Theorem 11].

The following property of ranks follows from their definition.

▶ **Lemma 30.** *Let $\mathcal{G}$ be a $[0,2]$-parity game, and let $v \xrightarrow{e} u$ be an edge in $\mathcal{G}$, such that either v belongs to Adam, or the edge e is prescribed by Eve's optimal strategy τ. Then the edge e has priority 2 or $\mathrm{rank}(v) \geq \mathrm{rank}(u)$. Furthermore, this inequality is strict if e has priority 1.*

Consider the 1-token game $G_1(\mathcal{H})$, and the ranks of its vertices. Note that for a vertex (q,p) in $G_1(\mathcal{H})$, we have $\mathrm{rank}(q,p) = 0$ if and only if q sprint simulates p. Define, for each state $q \in \mathcal{H}$, its optimal rank $\mathrm{opt}(q)$ to be the minimum rank of a vertex of the form (q,p) in $G_1(\mathcal{H})$. Note that if $\mathrm{opt}(q) = 0$ for all states $q \in \mathcal{H}$, then $\mathcal{H}$ has a sprint self-simulation. Thus, our iterative procedure focuses on reducing the optimal ranks for all states until they are all 0. We describe this procedure below.

Iterating towards a sprint self-simulation. Set $\mathcal{H}_0 = \mathcal{H}$. For each $i \geq 0$, perform the following three steps on $\mathcal{H}_i$ until $\mathcal{H}_{i+1} = \mathcal{H}_i$.

1. For all vertices $(p,q) \in G_1(\mathcal{H}_i)$, compute the optimal ranks $\mathrm{opt}_i(p)$ in $G_1(\mathcal{H}_i)$.
2. Obtain $\mathcal{H}'_i$ from $\mathcal{H}_i$ by removing all transitions $q \xrightarrow{a:1} q'$ with $\mathrm{opt}_i(q) < \mathrm{opt}_i(q')$.
3. Obtain $\mathcal{H}_{i+1}$ from $\mathcal{H}'_i$, by making all transitions $q \xrightarrow{a:1} q'$ with $\mathrm{opt}_i(q) > \mathrm{opt}_i(q')$ accepting.

We show that for each i, both $\mathcal{H}'_i$ and $\mathcal{H}_{i+1}$ are good HD Büchi automata that are equivalent to $\mathcal{H}_i$ [2, Lemmas 39 and 40]. By a simple induction, we get that each $\mathcal{H}_i$ for $i \geq 0$ is a good HD Büchi automaton equivalent to $\mathcal{H}$.

Note that in steps 2 and 3, we are either removing rejecting transitions or making rejecting transitions accepting, and hence this procedure terminates after at most $|\Delta|$ iterations. Let $\mathcal{H}^*$ be the automaton obtained after the procedure terminates. Since $\mathcal{H}^*$ is good, we know that it is SD. The next lemma thus shows that $\mathcal{H}^*$ has a sprint self-simulation.

▶ **Lemma 31.** *For all states p in $\mathcal{H}^*$, there is a language-equivalent state q in $\mathcal{H}^*$ such that p sprint simulates q.*

Proof. Assume, to the contrary, that there exists a state p such that $\mathrm{opt}^*(p) = \mathrm{rank}^*(p, q) > 0$. Fix an optimal winning strategy τ for Eve in $G_1(\mathcal{H}^*)$. Consider a finite play ρ of $G_1(\mathcal{H}^*)$ from (p, q) where Eve is playing according to τ, Adam chooses an accepting transition in his token at some point, while Eve is unable to. Then by monotonicity of ranks (Lemma 30), we know that rank^* strictly decreases across ρ at some point. Then, there must be a rejecting transition across which the quantity opt^* decreases as well. But since such transitions would have been made accepting in step 3 of the iteration, we get a contradiction. ◀

From the polynomial time determinisation construction for HD Büchi automata that have a sprint self-simulation presented in Section 5.1, we get a polynomial time determinisation procedure for HD Büchi automata. This concludes the proof of Theorem C.

6 Discussion

Our paper has shown two key results on HD Büchi automata: a 1-token game based characterisation of history-determinism for semantically-deterministic Büchi automata, and a polynomial time determinisation procedure. In the process of obtaining these results, we developed several novel techniques that we believe to be equally exciting and insightful. We finish by remarking some implications of our results and techniques, and natural future directions that our work points to.

Our first technique, presented in Section 3, reduces game based characterisations of history-determinism on parity automata to parity universal automata. But the history-determinism game on such an automaton is just a parity game, since Adam's word is always accepting. The 2-token conjecture thus reduces, by Theorem 13, to showing that this parity game is equivalent to the 2-token game. This seems easy enough to show at first glance, but it proves to be (unsurprisingly) difficult. This result also shows that the difficulty in proving or disproving the 2-token conjecture arises not from the language an automaton recognises, but rather from the structure of the automaton.

We also introduced lookahead games, and showed that k-lookahead games are equivalent to 1-token games for all $k \geq 1$ (Theorem A). This shows that the 1-token games are quite powerful, in the same sense that 2-token games are powerful due to them being equivalent to k-token games for all $k \geq 2$. While our 1-token game characterisation of history-determinism on SD Büchi automata does not extend to parity automata (Theorem 23), one can combine the two different approaches to give the 2-token game more power, both in the form of lookahead and more tokens. It would be interesting to consider such games to try extending the 2-token conjecture beyond Büchi and co-Büchi automata.

Our algorithm to determinise HD Büchi automata involves a quadratic blowup. However, we do not know whether this is tight. In fact, it is still open if HD Büchi automata are strictly more succinct than determinstic Büchi automata. Nevertheless, we are hopeful that our algorithm can offer some insights on how to make progress on this problem.

Let us end with a problem highlighted by Boker and Lehtinen in their recent survey [6, Section 6.3.2]. In all the existing game-based characterisations which are used to recognise HD automata efficiently, including ours (Theorem B), it is not clear how we can naturally convert a winning strategy for Eve from the 2-token game or the Joker game to a winning strategy in the HD game. Our algorithm to determinise HD Büchi automata, however, can be seen as one: starting with a winning strategy for Eve in the Joker game, we construct a strategy in the HD game that requires linear memory, thus obtaining a deterministic automaton of quadratic size. But the proof of correctness of our algorithm relies on Theorem B. Towards a more pure strategy-transfer argument, where ideally an algorithm for strategy transfer also proves a game-based characterisation of history-determinism, our algorithm comes tantalisingly close. Indeed, proving that the automaton constructed in Step 2 preserves the relevant invariants [2, Lemma 39] is the only place where we use the fact that we started with an HD automaton. We believe that trying to get rid of this assumption, in order to give an alternative strategy-transfer proof for the 1-token game characterisation of history-determinism on SD Büchi automata, could lead to crucial insights towards better understanding HD (Büchi) automata and token games.

References

1 Bader Abu Radi and Orna Kupferman. Minimization and Canonization of GFG Transition-Based Automata. *Logical Methods in Computer Science*, 18(3), 2022. doi:10.46298/lmcs-18(3:16)2022.

2 Rohan Acharya, Marcin Jurdziński, and Aditya Prakash. Lookahead Games and Efficient Determinisation of History-Deterministic Büchi Automata. *CoRR*, abs/2310.13498, 2024. doi:10.48550/arXiv.2310.13498.

3 Marc Bagnol and Denis Kuperberg. Büchi Good-for-Games Automata Are Efficiently Recognizable. In *Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018*, volume 122 of *LIPIcs*, pages 16:1–16:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.16.

4 Udi Boker, Thomas Henzinger, Karoliina Lehtinen, and Aditya Prakash. History-Determinism vs. Fair Simulation, To appear.

5 Udi Boker, Denis Kuperberg, Karoliina Lehtinen, and Michal Skrzypczak. On the Succinctness of Alternating Parity Good-for-Games Automata. *CoRR*, abs/2009.14437, 2020. arXiv:2009.14437.

6 Udi Boker and Karoliina Lehtinen. When a Little Nondeterminism Goes a Long Way: An Introduction to History-Determinism. *ACM SIGLOG News*, 10(1):24–51, 2023. doi:10.1145/3584676.3584682.

7 Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding Parity Games in Quasi-polynomial Time. *SIAM J. Comput.*, 51(2):17–152, 2022. doi:10.1137/17m1145288.

8 Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Generalized Parity Games. In *Foundations of Software Science and Computational Structures, 10th International Conference, FOSSACS 2007*, volume 4423 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2007. doi:10.1007/978-3-540-71389-0_12.

9 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. doi:10.1007/3-540-36387-4.

10 Yuri Gurevich and Leo Harrington. Trees, Automata, and Games. In *Symposium on Theory of Computing, STOC 1982*, pages 60–65. ACM, 1982. doi:10.1145/800070.802177.

11 Thomas A. Henzinger and Nir Piterman. Solving Games Without Determinization. In *Computer Science Logic, CSL 2006*, volume 4207 of *Lecture Notes in Computer Science*, pages 395–410. Springer, 2006. doi:10.1007/11874683_26.

12 Marcin Jurdzinski. Small Progress Measures for Solving Parity Games. In *STACS 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2000. doi:10.1007/3-540-46541-3_24.

13 Marcin Jurdzinski and Ranko Lazic. Succinct Progress Measures for Solving Parity Games. In *Symposium on Logic in Computer Science, LICS 2017*, pages 1–9. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005092.

14 Denis Kuperberg and Michal Skrzypczak. On Determinisation of Good-for-Games Automata. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015*, volume 9135 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 2015. doi:10.1007/978-3-662-47666-6_24.

15 Orna Kupferman. Using the Past for Resolving the Future. *Frontiers Comput. Sci.*, 4, 2022. doi:10.3389/fcomp.2022.1114625.

16 Karoliina Lehtinen, Pawel Parys, Sven Schewe, and Dominik Wojtczak. A Recursive Approach to Solving Parity Games in Quasipolynomial Time. *Log. Methods Comput. Sci.*, 18(1), 2022. doi:10.46298/lmcs-18(1:8)2022.

17 Donald A. Martin. Borel Determinacy. *Annals of Mathematics*, 102(2):363–371, 1975. URL: http://www.jstor.org/stable/1971035.

18 Aditya Prakash. Checking History-Determinism is NP-hard for Parity Automata. In *Foundations of Software Science and Computation Structures - 27th International Conference, FoSSaCS 2024*, volume 14574 of *Lecture Notes in Computer Science*, pages 212–233. Springer, 2024. doi:10.1007/978-3-031-57228-9_11.

19 Aditya Prakash and K. S. Thejaswini. On History-Deterministic One-Counter Nets. In *Foundations of Software Science and Computation Structures - 26th International Conference, FoSSaCS 2023*, volume 13992 of *Lecture Notes in Computer Science*, pages 218–239. Springer, 2023. doi:10.1007/978-3-031-30829-1_11.

20 Bader Abu Radi and Orna Kupferman. On Semantically-Deterministic Automata. In *International Colloquium on Automata, Languages, and Programming, ICALP 2023*, volume 261 of *LIPIcs*, pages 109:1–109:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.109.

21 Bader Abu Radi, Orna Kupferman, and Ofer Leshkowitz. A Hierarchy of Nondeterminism. In *Mathematical Foundations of Computer Science, MFCS 2021*, volume 202 of *LIPIcs*, pages 85:1–85:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.MFCS.2021.85.

22 Sven Schewe. Minimising Good-For-Games Automata Is NP-Complete. In *Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020*, volume 182 of *LIPIcs*, pages 56:1–56:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.FSTTCS.2020.56.

23 Igor Walukiewicz. Monadic Second-Order Logic on Tree-like Structures. *Theor. Comput. Sci.*, 275(1-2):311–346, 2002. doi:10.1016/S0304-3975(01)00185-2.