WARWICK
THE UNIVERSITY OF WARWICK

Manuscript version: Author's Accepted Manuscript
The version presented in WRAP is the author's accepted manuscript and may differ from the published version or Version of Record.

Persistent WRAP URL:
http://wrap.warwick.ac.uk/174245

How to cite:
Please refer to published version for the most recent bibliographic citation information. If a published version is known of, the repository item page linked to above, will contain details on accessing it.

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

Publisher's statement:
Please refer to the repository item page, publisher's statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk.

On History-Deterministic One-Counter Nets

Aditya Prakash[1] and K. S. Thejaswini[1]

Department of Computer Science, University of Warwick, Coventry, UK
{aditya.prakash,thejaswini.raghavan.1}@warwick.ac.uk

Abstract. We consider the model of history-deterministic one-counter nets (OCNs). History-determinism is a property of transition systems that allows for a limited kind of non-determinism which can be resolved 'on-the-fly'. Token games, which have been used to characterise history-determinism over various models, also characterise history-determinism over OCNs. By reducing 1-token games to simulation games, we are able to show that checking for history-determinism of OCNs is decidable. Moreover, we prove that this problem is **PSPACE**-complete for a unary encoding of transitions, and **EXPSPACE**-complete for a binary encoding and undecidable for one-counter automata (OCA), which are OCNs that can test for zeroes.

We then study the language properties of history-deterministic OCNs. We show that the resolvers of non-determinism for history-deterministic OCNs are eventually periodic. As a consequence, for a given history-deterministic OCN, we construct a language equivalent deterministic OCA. We also show the decidability of comparing languages of history-deterministic OCNs, such as language inclusion and language universality.

Keywords: History-determinism · Token games · One-counter nets · One-counter automaton.

1 Introduction

While deterministic automata are algorithmically efficient for problems such as synthesis or for solving games, they are often much less succinct, or less expressive than their non-deterministic counterparts. As such, many intermediate models between determinism and non-determinism have been studied [1,2,3,4,5], with history-determinism being one such well-studied notion over the recent years. History-deterministic automata over infinite words with parity acceptance condition was introduced by Henzinger and Piterman as a tool to solve verification games, although dubbed good-for-games in their work [6]. Such automata are known to be exponentially more succinct than their deterministic counterpart [7], and are known to form a robust class of automata that is both algorithmically and conceptually interesting [6,8,9,7,10,11,12,13,14].

The notion of history-determinism emerged independently in the setting of cost automata that can capture all regular cost functions as opposed to their deterministic version [15]. Recently, history-determinism has been studied in quantitative settings [16,17], as well as infinite-state systems such as pushdown automata [18,19], Parikh automata [20], and timed automata [21,22], where they are often more succinct and expressive than their deterministic counter part.

One-counter nets are finite-state systems along with a counter that stores a non-negative integer value which can never be explicitly tested for zero. They correspond to 1-dimensional VASS, Petri nets with exactly one unbounded place, and are a subclass of one-counter automata which do not have zero tests, and hence are also a subclass of pushdown automata. They are one of the simplest infinite-state systems, and hence many problems pertaining to one-counter nets are easier than models that subsume them.

The structure of the resolvers that resolve non-determinism on-the-fly are crucial to understand history-determinism in various models. While for automata over infinite words with parity conditions, these resolvers take the shape of deterministic parity automata [6], the situation for resolvers in history-deterministic infinite-state systems is not as well understood. Indeed, the computability of such a resolver for a given history-deterministic pushdown automaton is left as an open problem in the works of Guha, Jecker, Lehtinen and Zimmermann [18]. For history-deterministic Parikh automata, it is still an open problem if the resolver can be given by a deterministic Parikh transducer [20]. Moreover, many other problems such as deciding history-determinism or even language inclusion among history-deterministic automata are undecidable for pushdown automata and Parikh automata [18,19,20]. We consider history-determinism over one-counter nets, where we are able to answer positively to all of the above questions.

To answer several of these questions, we use results and techniques from the simulation problem over one-counter nets [23,24]. This is not surprising, since simulation of various models has close ties with history-determinism [6,21].

Our Contribution We study history-deterministic OCNs and establish them as a class of infinite-state systems where many problems pertaining to history-determinism are decidable. This is unlike many other classes of history-deterministic infinite-state systems that have been studied so far.

Firstly, we show that checking for history-determinism of a given one-counter net is **PSPACE**-complete when the transitions are encoded in unary, and is **EXPSPACE**-complete for a more succinct encoding (Theorem 4, Theorem 26). We achieve the upper bound by giving a novel reduction from the one-token game [11] to the simulation problem over OCNs. One-token games characterise history-determinism over OCNs, and thus our reduction further extends the link between history-determinism and simulation. This decidability result is in contrast to one-counter automata (OCA), where checking for history-determinism becomes undecidable by just adding zero-tests to OCNs (Theorem 27).

Secondly, we show that resolvers for non-determinism in history-deterministic OCNs can be expressed as an eventually periodic set. Using this, we are able to determinise history-deterministic OCNs to give a language equivalent deterministic OCA.

Finally, we show the problems of language inclusion and language universality for history-deterministic OCNs to be in **PSPACE** and **P** respectively. This is in unlike non-deterministic OCNs, where these problems are known to be undecidable and Ackermann-complete respectively. Even for the class of deterministic OCA—which we show history-deterministic OCNs can be converted to—the inclusion problem is known to be undecidable.

Good-for-Gameness A notion closely related to history-determinism (HD) is that of good-for-gameness. An automaton is said to be *good-for-games* (GFG) if its composition with a game whose acceptance condition is given by the language of the automaton yields an equivalent game. For parity automata over infinite words, these two notions are known to be equivalent [6,25], but they do not coincide on all models [16]. For the purposes of our paper, we deal with history-deterministic OCNs, as in our setting the notion of history-determinism is equivalent to good-for-gameness when composition with infinitely branching games is considered [26]. We note however, that this is not true when compositionality is restricted to only finitely branching games [26].

2 Preliminaries

We use $\mathbb{N}$ to denote the set of positive integers and $\mathbb{N}_0$ to denote non-negative integers. An *alphabet*, denoted by Σ, is any finite non-empty set of *letters*, and the set of all finite words over Σ is denoted by Σ^*. The empty word over Σ is denoted by ϵ, and we use Σ_ϵ to denote the set $\Sigma \cup \{\epsilon\}$. A *language* $\mathcal{L}$ over Σ is a subset of Σ^*.

Labelled Transition System A *labelled transition system* (LTS) is a tuple $\mathcal{S}$ consisting of $\mathcal{S} = (Q, \Sigma, \rightarrow, q_0, F)$. In this paper, we assume that Q is a (countable) set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, Σ is a finite alphabet, $\rightarrow \subseteq Q \times \Sigma_\epsilon \times Q$ is the set of transitions.

If a transition (q_1, a, q_2) belongs to $\rightarrow$, we instead represent it as $q_1 \xrightarrow{a} q_2$ as well. On a finite word w, a ρ is said to be a *run* of the labelled transition system $\mathcal{S}$ if it is a finite alternating sequence of states and letters of Σ: $\rho = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots q_{k-1} \xrightarrow{a_k} q_k$, where each i, $q_i \xrightarrow{a_i} q_{i+1} \in \rightarrow$ and $a_i \in \Sigma_\epsilon$ such that $w = a_0 \cdot a_1 \dots a_k$. A run ρ described above is accepting if the state $q_k \in F$.

An LTS that has no ϵ-transitions is said to be a *realtime LTS*. For an LTS $\mathcal{S} = (Q, \Sigma, \rightarrow, q_0, F)$ being realtime, we have $\rightarrow \subseteq Q \times \Sigma \times Q$. Unless mentioned otherwise, we mostly deal with realtime LTS for the sake of a simpler presentation. An LTS $\mathcal{S} = (Q, \Sigma, \rightarrow, q_0, F)$ is *deterministic* if $\rightarrow$ is a function from $Q \times \Sigma$ to Q and not just a relation.

Two player games Throughout the paper, we will be using two player games on countably sized arenas, between the players Adam and Eve, denoted by $\forall$ and $\exists$ respectively. The winning condition will be a reachability condition for one of the players, often $\forall$. These can be interpreted as a Gale-Stewart games [27] and we know that such games are determined, that is they have a winner, which is either ∀ or ∃. Moreover, each of the players have a positional strategy, where their current strategy depends on their positions in the current arena. We say that two games are *equivalent*, if they have the same winner.

One-Counter Automata A *one-counter automaton* (OCA) $\mathcal{A}$ is given by a tuple $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, Σ is a finite alphabet, and Δ is the set of transitions, given as a relation $\Delta \subseteq Q \times \{\text{zero}, \neg\text{zero}\} \times \Sigma \times \{-1, 0, 1\} \times Q$.

Here, the symbols zero and ¬zero are used to distinguish between transitions that can happen when the counter value is 0, and when the counter value is positive respectively. One can think of the counter as a stack, where the stack has a distinguished bottom-of-the-stack symbol, which cannot be popped. The configurations in the automaton are given by pairs (q, m), where q denotes the current state, and $m \in \mathbb{N}_0$ denotes the counter value. We use $\mathcal{C}(\mathcal{A})$ to denote the set of configurations of $\mathcal{A}$.

A one-counter automaton generates an infinite-state LTS over the set of configurations $Q \times \mathbb{N}$, such that the transitions are as defined below. For each configuration (q, m), upon reading $a \in \Sigma_\epsilon$,

– if $m > 0$, takes a transition of the form $(q, \neg\text{zero}, a, d, q')$, where $d \in \{-1, 0, 1\}$ to $(q', m + d)$;
– if $m = 0$, takes a transition of the form $(q, \text{zero}, a, d, q')$, where $d \in \{0, 1\}$ to $(q', m + d)$.

For two configurations $c, c' \in \mathcal{C}(A) = Q \times \mathbb{N}_0$, we use the notation $c \xrightarrow{a,d} c'$ to denote the fact that c' can be reached from c upon taking some transition $\delta \in \Delta$ upon reading a, with a change of counter value d. We shall also say that $c \xrightarrow{a,d} c'$ is a transition in $\mathcal{A}$, as $c \xrightarrow{a,d} c'$ is a transition in the infinite LTS of $\mathcal{A}$. We thus view $\mathcal{A}$ as both an automaton and a LTS (generated by $\mathcal{A}$), and switch between these two notions interchangeably. A run of $\mathcal{A}$ over a word w is a finite sequence of alternating configurations and transitions : $\rho = c_0 \xrightarrow{a_0,d_0} c_1 \cdots c_n \xrightarrow{a_n,d_n} c_{n+1}$ such that $a_0 a_1 \cdots a_n = w$, and $c_0 = (q_0, 0)$. The run ρ is an *accepting run* if its last configuration $c_{n+1} = (q_{n+1}, k_{n+1})$ is accepting, i.e. $q_{n+1} \in F$. We say a word w is an *accepting word* in $\mathcal{A}$ if it has an accepting run in $\mathcal{A}$. Finally, we define the language of $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$ to be the set of all accepting words in $\mathcal{A}$. We say that $\mathcal{A}$ is a *deterministic one-counter automaton*, if Δ is a (partial) function from $Q \times \{\text{zero}, \neg\text{zero}\} \times \Sigma$ to $\{-1, 0, 1\} \times Q$.

One-counter nets The model of *one-counter nets* (OCNs) can be interpreted as a restriction added to one-counter automaton that do not have the ability to test for zero. Alternatively, one can view this as a finite-state automaton that has access to a stack which can store only one symbol and no bottom-of-the-stack element. Any feasible run cannot pop an empty stack. More formally, a one-counter net $\mathcal{N}$ is a tuple $(Q, \Sigma, \Delta, q_0, F)$ where Q is the set of finite states, Σ is a finite alphabet, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is the set of final or accepting states. The set $\Delta \subseteq Q \times \Sigma \times \{-1, 0, 1\} \times Q$ are the transitions in the net $\mathcal{N}$.

The configurations of an OCN are similar to that of an OCA. It consists of a pair $(q, n) \in Q \times \mathbb{N}_0$. We shall use the notation $\mathcal{C}(\mathcal{N}) = Q \times \mathbb{N}_0$ to denote the set of configurations of $\mathcal{N}$. From a configuration (q, n), we reach a configuration $(p, n + d)$ in one step, if there is a transition $\delta = (q, a, d, p)$, for some $a \in \Sigma$ and $d \in \{-1, 0, +1\}$ and $n + d \geq 0$. We can define a run on an OCN, an accepting run and an accepting word similar to an OCA. We say an OCN $\mathcal{N}$ is *complete* if for every configuration $c \in \mathcal{C}(\mathcal{N})$ and every letter $a \in \Sigma$, there exists a transition $c \xrightarrow{a,d} c'$.

Remark 1. For the most of the paper we talk about one-counter nets (automata) with unary transitions, i.e. transitions that increment or decrement the counter by at most 1. However, they are as expressive as succinct models where the one-counter net has a *binary encoding*, i.e. when the transitions allow the counter to be incremented or decremented by positive integers represented in binary. This can be observed, for instance, by giving a construction similar to that of Valiant's for deterministic pushdown automata ([28], Section 1.7).

History-Deterministic One-Counter Nets We define history-determinism in the setting of one-counter net. Informally, an OCN $\mathcal{N}$ is *history-deterministic*, if the non-deterministic choices required to accept a word w which is in $\mathcal{L}(\mathcal{N})$ can be made on-the-fly. These choices depend only on the word read so far, and do not require the knowledge of the future of the word to construct an accepting run for a word in $\mathcal{L}(\mathcal{N})$ (hence the term history-determinism). Formally, we say an OCN $\mathcal{N}$ is history-deterministic, if $\exists$ wins the letter game on $\mathcal{N}$ defined below.

Definition 2 (Letter game for OCN). *Given an OCN $\mathcal{N} = (Q, \Sigma, \Delta, q_0, F)$, the letter game on $\mathcal{N}$ is defined between the players $\forall$ and $\exists$ as follows: the positions of the game are $\mathcal{C}(\mathcal{N}) \times \Sigma^*$, with the initial position $((q_0, 0), \epsilon)$. At round i of the play, where the position is (c_i, w_i):*

- *$\forall$ selects $a_i \in \Sigma$*
- *$\exists$ selects a transition δ which can be taken at the configuration c_i on reading a_i, i.e. $c_i \xrightarrow{a_i, d_i} c_{i+1}$*

If $\exists$ is unable to choose a transition (i.e. there is no a_i transition at the configuration c_i in the LTS generated by the net $\mathcal{N}$), and $w_{i+1} = w_i a_i$ is the prefix of an accepting word, $\exists$ loses immediately. The player $\forall$ wins immediately when the word w_{i+1} is accepting but the configuration c_{i+1} is not at an accepting state, and the game terminates. The game continues from (c_{i+1}, w_{i+1}) otherwise. Player $\exists$ wins any infinite play.

We say a strategy for $\exists$ in the letter game of $\mathcal{N}$ is a *resolver* for $\mathcal{N}$, if it is a winning strategy for $\exists$ in the letter game.

Our characterization of history-deterministic one-counter nets by the above letter game is slightly different from the one presented in the work of Guha, Jecker, Lehtinen, and Zimmermann [18] for pushdown automata. In their work, they define history-determinism as having a consistent strategy based on the transitions taken so far. It is easy to argue that these two definitions are equivalent.

The letter game can be formulated as a reachability game over countably many vertices, where the player ∀ is trying to reach a position of the form $(c, w) \in \mathcal{C}(\mathcal{N}) \times \Sigma^*$, where c is at a rejecting state, while w is accepting. As such games are determined [27], the notion of history-determinism formulated as ∃ winning the letter game is well-defined.

Letter games have been used extensively to characterise history-determinism for other models as well, such as parity automata [6] and for various kinds of quantitative and timed automata on both finite and infinite words [12,16,21].

To aid our understanding of history-determinism as well as the above definition, we provide an example of a game where ∃ wins the letter game on this automaton but the strategy is based on her counter configuration.

Example 3. Consider the language

$$\mathcal{L} = \left\{ a^n \$ b^{n_1} \$ b^{n_2} \$ \ldots \$ b^{n_k} \$ \mid \sum_{i=1}^{k} n_i \leq n \text{ or } n_k = 2, \sum_{i=1}^{k-1} n_i = n - 1 \right\}$$

which can be accepted by a history-deterministic OCN as shown in Figure 1. The initial state is indicated with an arrow pointing to it, and the final states are double-circled. Missing transitions are assumed to go to a rejecting sink state. In the corresponding letter game, ∀ plays the letter a several times, say n-many times followed by a \$. The corresponding transitions so far are deterministic. Later, ∀ reads some series of bs and \$s, such that the word continues to be in the language. Note that the non-determinism occurs in only one state, which is marked with an X, upon reading the letter b. A winning strategy of ∃ which proves that this net is history-deterministic is the following: she takes the 'down' transition if the counter value is strictly larger than 1, but the 'right' transition on b otherwise. This non-determinism can't be resolved by removing transitions, because removing either of the 'down' b-transition or the 'right' b-transition changes the language accepted. We note that an equivalent deterministic OCN exists nevertheless, where on reading a b after any \$ does not change the value of the counter, but reduces the counter by two for the second b after a \$ and reduces the counter by 1 for any b after that, until a \$ is seen again.

3 Deciding History-Determinism

The main result of this section is that deciding history-determinism for a given OCN is decidable and is **PSPACE**-complete as stated in the theorem below.

Theorem 4. *Given a one-counter net $\mathcal{N}$, checking if $\mathcal{N}$ is history-deterministic is* **PSPACE***-complete.*

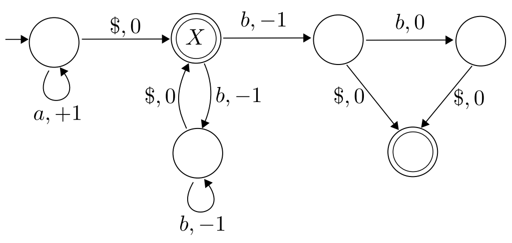

Fig. 1. A history-deterministic OCN accepting $\mathcal{L}$

The rest of this section is dedicated to the proof of the above statement.

The proof of showing the upper bound proceeds by a series of polynomial time reductions as below.

$$\begin{array}{c}\text{Deciding history-determinism}\\ \Updownarrow \\ \text{Deciding if } \exists \text{ wins letter game}\\ \Updownarrow \\ \text{Deciding if } \exists \text{ wins 1-token game}\\ \Downarrow \\ \text{Deciding if } \exists \text{ wins simulation game}\end{array}$$

We shall define these games rigorously and prove these reductions in Subsection 3.1. Finally, since the winner of the simulation game over one-counter nets is in **PSPACE** [24], this gives us the upper bound.

For the lower bound, we reduce from the problem of emptiness checking for alternating finite-state automata over a unary alphabet to deciding if $\exists$ wins the letter game.

3.1 Token Games

Deciding history-determinism efficiently for finite-state parity automata over infinite words has been a major area of study over the recent years. Bagnol and Kupergerg [11], gave a polynomial time procedure for deciding history-determinism when the finite automata accepts with a Büchi condition. Their underlying technique is a two-player game, called G_2 or 2-token games, which they proved to be equivalent to the letter game when the automaton is Büchi. Boker, Kuperberg, Lehtinen, and Skrzypczak [12] extended this to show that the game G_2 is equivalent to the letter game when the automaton is co-Büchi as well. Deciding the winner in G_2 for an automaton of a fixed parity index takes polynomial time [12], and hence deciding history-determinism for the cases of when the parity automata accepts words based on Büchi or co-Büchi condition is polynomial.

It is conjectured that winning G_2 is equivalent to the letter game for higher parity indices as well, and this is known as the G_2 conjecture [12]. Token games have also been instrumental in deciding history-determinism for quantitative automata, in the works of Boker and Lehtinen [17]. In their paper, they show that for finite words on a finite-state boolean automaton, history-determinism is characterised by G_1. This was later adapted to labelled transition systems with safety acceptance condition, in the works of Henzinger, Lehtinen, and Totzke [21]. Thus, the 1-token games also characterise history-determinism for OCNs over finite words. We include a proof nonetheless, for the sake of completeness.

In a play of the letter game, $\forall$ picks the letters while $\exists$ picks the transitions, and the winning condition for $\exists$ is to produce an accepting run for any word that is in the language. Token games work similarly, but they impose more constraints on $\forall$. This is done by asking him to also display a valid run during the game with the help of some number of tokens. Here, we concentrate on the 1-token game G_1. The player $\forall$ wins the game G_1 if and only if he produces an accepting run, whilst $\exists$ produces a rejecting run. We make this more formal in the definition below.

Definition 5 (One token game G_1). *Let $\mathcal{N} = (Q, \Sigma, \Delta, q_0, F)$ be a one-counter net. The positions of the game G_1 on $\mathcal{N}$ are a pair of configurations, $\mathcal{C}(\mathcal{N}) \times \mathcal{C}(\mathcal{N})$, where the first configuration in the pair denotes the position of $\exists$'s token, and the second $\forall$'s token. The game starts with the initial position $(c_0^\exists, c_0^\forall) = ((q_0, 0), (q_0, 0))$. At the i^{th} iteration of the play, where the position is $(c_i^\exists, c_i^\forall)$:*

1. *$\forall$ selects $a \in \Sigma$*
2. *$\exists$ selects a transition for her token, $c_i^\exists \xrightarrow{a,d} c_{i+1}^\exists$*
3. *$\forall$ selects a transition for his token, $c_i^\forall \xrightarrow{a,d'} c_{i+1}^\forall$*

If $\exists$ is unable to choose a transition for her token whereas $\forall$ can choose a transition and extend the run on his token to an accepting run, then the game terminates and $\exists$ loses the game. However, irrespective of $\exists$'s ability to extend her run, if $\forall$ is unable to choose a transition for his token, then the game again terminates but $\forall$ loses the game.

If both the players can extend their runs by picking a transition then and if $\forall$'s state in $c_{i+1}^\forall$ is accepting, but $\exists$'s state in $c_{i+1}^\exists$ is rejecting then again the game terminates and $\exists$ loses the game. Else, the game goes to (c_{i+1}, c'_{i+1}) for another round of the play. We add that $\exists$ wins any infinite play.

Letter games can be seen as a version of token games where $\forall$ plays with infinitely many tokens. We show in the following lemma that one-token games—even with this limited power of $\forall$—can capture letter games.

Lemma 6. *For an OCN $\mathcal{N}$, if $\exists$ wins the game G_1 on $\mathcal{N}$, then $\exists$ has a winning strategy in the letter game.*

To prove the above lemma, we need to understand better the structure of the resolvers for OCNs. Consider the definition given below of *residual transitions.*

Intuitively, these are transitions such that if there was an accepting word from a configuration with the first letter as a, then upon taking a residual transition on a, there is still an extension of the run on the word from the new configuration that is accepting. More formally, we say that a transition $(q,k) \xrightarrow{a,d} (q',k')$ is *residual* if $\mathcal{L}(q',k') = a^{-1}\mathcal{L}(q,k)$, where $\mathcal{L}(q,k)$ (and $\mathcal{L}(q',k')$) is the set of words that are accepted in $\mathcal{N}$ when the initial configuration is (q,k) ((q',k')), instead of $(q_0,0)$. The proposition below shows any winning strategy of $\exists$ can be characterised by these residual transitions.

Proposition 7. *For an OCN $\mathcal{N}$, an $\exists$ strategy σ in the letter game is winning for $\exists$ if and only if σ takes only residual transitions.*

Note that in the letter game, each player winning the game has a positional winning strategy, as it is a reachability game. Suppose that $\exists$ wins the letter game, then $\exists$ has a winning strategy which can be given by a (partial) function $\sigma : (Q \times \mathbb{N}) \times \Sigma^* \times \Sigma \to \Delta$. Using Proposition 7, we can show that $\exists$'s strategy only depends on the configuration, and is independent of the word read so far.

Proposition 8. *If $\exists$ wins the letter game, then $\exists$ has a winning strategy σ that only depends on the current configuration of the play, i.e σ is a partial function $\sigma : (Q \times \mathbb{N}) \times \Sigma \to \Delta$*

Having shown that G_1 is equivalent to the letter game, we show that deciding the winner in the game G_1 is in **PSPACE**. This implies deciding history-determinism is also decidable, and in **PSPACE**. We do so by reducing G_1 to the simulation problem between two one-counter nets, which is known to be **PSPACE**-complete ([24], Theorem 7).

Given two OCNs $\mathcal{N}$ and $\mathcal{N}'$ at configurations (q,n) and (q',n'), we say $\mathcal{N}'$ simulates $\mathcal{N}$ (or $\mathcal{N}$ is simulated by $\mathcal{N}'$) from their corresponding configurations if for any sequence of transitions from (q,n), there is also a sequence of transitions from (q',n') which is built 'on-the-fly'. This alternation between existential and universal quantifiers in the above statement renders this definition perfect to be captured by the following game between the players $\forall$ and $\exists$.

Definition 9 (Simulation Game). *Given two OCNs $\mathcal{N} = (Q,\Sigma,\Delta,q_I,F)$ and $\mathcal{N}' = (Q',q'_0,\Sigma,\Delta',q'_I,F')$ and two configurations $c = (p,k)$ and $c' = (p',k')$ in $\mathcal{C}(\mathcal{N})$ and $\mathcal{C}(\mathcal{N}')$ respectively where $k,k' \in \mathbb{N}$. The* simulation game *between the OCNs $\mathcal{N}$ and $\mathcal{N}'$ at a position (c,c'), denoted by $\mathcal{G}((\mathcal{N},c) \hookrightarrow (\mathcal{N}',c'))$, is a two player game between $\forall$ and $\exists$, with positions in $\mathcal{C}(\mathcal{N}) \times \mathcal{C}(\mathcal{N}')$ where the initial position is $(c_0,c'_0) = (c,c')$. At round i of the play, where the position is (c_i,c'_i):*

- *$\forall$ selects a letter $a \in \Sigma$, and a transition $c_i \xrightarrow{a,d} c_{i+1}$ in $\mathcal{N}$*
- *$\exists$ selects an a-transition $c'_i \xrightarrow{a,d'} c'_{i+1}$ in $\mathcal{N}'$*

If $\forall$ is unable to choose a transition, then $\forall$ loses the game immediately. If $\exists$ is unable to choose a transition but $\forall$ can select a transition and extend the run in $\mathcal{N}$ to an accepting run, then $\exists$ loses the game.

Otherwise, if $\forall$*'s state in* c_{i+1} *is accepting but* $\exists$*'s state in* c'_{i+1} *is rejecting, then* $\exists$ *loses the game, and the game terminates. Else, the game goes to* (c_{i+1}, c'_{i+1}) *for another round of the play. The player* $\exists$ *wins any infinite play.*

If $\exists$ wins the above game, we say $(\mathcal{N}', (p', k'))$ simulates $(\mathcal{N}, (p, k))$, and we denote it by $(\mathcal{N}, (p, k)) \hookrightarrow (\mathcal{N}', (p', k'))$. Furthermore, we say $\mathcal{N}'$ simulates $\mathcal{N}$ or $\mathcal{N} \hookrightarrow \mathcal{N}'$ if $(\mathcal{N}, (q_I, 0)) \hookrightarrow (\mathcal{N}', (q'_I, 0))$.

As the simulation game is a reachability game over a countably sized arena, it is determined, and the winning player has a positional strategy. Thus, if $\exists$ wins the above simulation game $\mathcal{G}((\mathcal{N}, (p, k)) \hookrightarrow (\mathcal{N}', (p', k')))$, then $\exists$ has a positional winning strategy $\sigma_\exists : \mathcal{C}(\mathcal{N}) \times \mathcal{C}(\mathcal{N}') \times \Sigma \to \Delta'$.

Remark 10. In the literature over one-counter nets [29,24,30], the winning condition for the players on the simulation game is expressed differently, via the inability of the players to choose transitions, rather than accepting states. The player $\forall$ ($\exists$) loses the game if $\forall$ ($\exists$) is unable to choose a transition. It can however, be shown that the two versions of the simulation games are log-space reducible to each other.

Note the similarities (and differences) in G_1 and the simulation game. In both, the winning condition for $\forall$ would like $\forall$'s run to be accepting, while $\exists$'s to be rejecting. In G_1 however, $\exists$ is picking the transition first, while in the simulation game, $\forall$ is picking the transition first.

With some modifications to the structure of the underlying net in G_1, we can ensure that the simulation game between the modified net and the original net captures G_1. The intuition is that, in the simulation game, the net which is simulated is modified so that $\forall$ is forced to delay choosing his transition. This is formalized in the proof of the following lemma, and explained with a diagram in Figure 2.

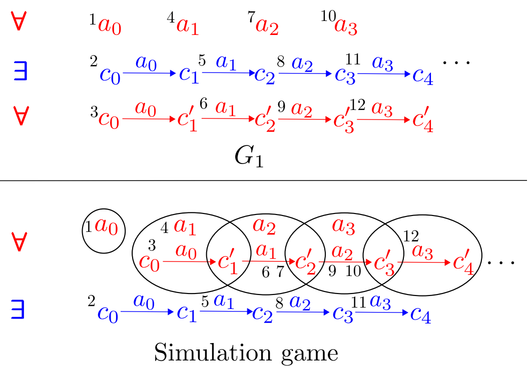

Fig. 2. An illustration of a play of G_1, seen as a play of the simulation game

Lemma 11. *Given a one-counter net $\mathcal{N}$, there are one-counter nets $\mathcal{M}$ and $\mathcal{M}'$, which have size at most polynomial in size of $\mathcal{N}$ such that $\exists$ wins G_1 on $\mathcal{N}$ if and only if $\exists$ wins $\mathcal{M} \hookrightarrow \mathcal{M}'$.*

Proof. (Sketch) For each run in $\mathcal{N}$, we have a run in $\mathcal{M}$ that lags behind one transition. The one-counter net $\mathcal{M}'$ on the other hand is relatively similar to $\mathcal{N}$. We impose this "one-transition lag" in $\mathcal{M}$ by construction where each transition chosen by $\forall$ in $\mathcal{M}$ corresponds to a letter along with a transition of $\mathcal{N}$. But this transition of $\mathcal{N}$ is over the letter that $\forall$ had chosen last turn. The alternation produced between $\forall$ and $\exists$ in a play of the simulation game between $\mathcal{M}$ and $\mathcal{M}'$ of the nets constructed corresponds exactly the alternation produced between $\forall$ and $\exists$ in G_1 over $\mathcal{N}$. Figure 2 captures the intuition behind this construction discussed.

The net $\mathcal{M}'$ is linear in the size of $\mathcal{N}$ whereas $\mathcal{M}$ has size approximately $\mathcal{N} \times |\Sigma|$, where $|\Sigma|$ is the size of the alphabet. This factor of $|\Sigma|$ arises due to remembering the previous letter read in the state space to create this lag for $\forall$'s decisions.

Finally, we see that the following theorem from the work of Hofman, Lasota, Mayr, and Totzke [24] shows that the winner of a simulation game can be solved in **PSPACE**. We recall their results to fit our notation below.

Theorem 12 ([24], Theorem 7). *Given two one-counter nets $\mathcal{N}$ and $\mathcal{N}'$, with configurations (p,k) and (p',k') in $\mathcal{C}(\mathcal{N})$ and $\mathcal{C}(\mathcal{N}')$ respectively, with k and k' represented in binary, deciding whether $(\mathcal{N}',(p',k'))$ simulates $(\mathcal{N},(p,k))$ is in* **PSPACE**. *Moreover, the set of (k,k') for which $(\mathcal{N},(p,k)) \hookrightarrow (\mathcal{N}',(p',k'))$ is semilinear, and can be computed in* **EXPSPACE**.

We get the following lemma as a corollary of Lemmas 6 and 11 and Theorem 12.

Lemma 13. *Given a one-counter net $\mathcal{N}$, we can decide in* **PSPACE** *if $\mathcal{N}$ is history-deterministic.*

3.2 Lower Bounds

Although solving the simulation game turns out to be **PSPACE**-complete itself from the work of Srba [29], this lower bound result does not work for our reduction to simulation games. The reduction we give from G_1 to simulation games produces only a restricted class of simulation games which solve G_1.

Nevertheless, we show that deciding history-determinism is still **PSPACE**-hard, showing that even this restriction of the simulation problem is enough to induce **PSPACE**-hardness.

Lemma 14. *Given a one-counter net $\mathcal{N}$, it is* **PSPACE***-hard to decide if $\mathcal{N}$ is history-deterministic.*

Proof (Sketch). We reduce from the problem of checking non-emptiness of an alternating finite-state automaton over a unary alphabet. This problem was proven to be **PSPACE** complete by Holzer [31], with its proof simplified by Jančar and Sawa [32]. The intuition behind the reduction is to recreate a run of the alternating automaton in the letter game of a constructed OCN. In the letter game, a "fair" play of $\forall$ corresponds to a branch of a run-tree in the automaton, with $\exists$ resolving universal transitions and $\forall$ resolving existential ones. The player $\forall$ can ensure that he wins the letter game if and only if the alternating automaton has some word that he can demonstrate is in the language. If $\forall$ plays "unfairly", then there are gadgets to ensure that $\exists$ automatically wins.

4 Languages and History-Determinism in OCNs

We dedicate this section to tackling different questions about languages accepted by history-deterministic one-counter nets and decision problems on such languages.

4.1 Languages Accepted by History-Deterministic OCNs

While in history-deterministic models we are able to resolve the non-determinism on-the-fly, it is not well-understood how these resolvers might look like in general. In fact, Guha, Jecker, Lehtinen, and Zimmermann showed that there are history-deterministic pushdown automata whose resolvers cannot be given by a pushdown automata [18], and whether such a resolver can be computed is an open problem.

In this sub-section, our goal is to understand better the languages of history-deterministic OCNs. As a first-step towards this goal, we already have some intuition from the previous section on the eventually periodic nature of the transitions that are residual (as a corollary of Lemma 11 and Theorem 12). Here, we solidify this intuition by defining what it means to have *semilinear-strategy property* for a resolver and to then show that all nets have this property. For the case of history-deterministic nets, using this semi-linearity of the resolvers, we show the existence of a language-equivalent deterministic OCA.

We first show a sufficient characterisation which we call the *semilinear-strategy property*, for if a given history-deterministic one-counter net can be determinised.

We say a transition $\delta = (p, a, d, p')$ in an one-counter net $\mathcal{N}$ is a good transition at (p, k), if $((p, k), (p, k))$ is in the winning region of G_1, and the transition $\delta = (p, k) \xrightarrow{a,d} (p', k+d)$ is a winning move for $\exists$ in G_1 when $\forall$ chooses the letter a. We also write this sometimes as $(p, k) \xrightarrow{a,d} (p', k + d)$ is a good transition in $\mathcal{N}$. The following lemma can be seen as a weakening of Proposition 7.

Lemma 15. *Let $\mathcal{N} = (Q, \Sigma, \Delta, q_0, F)$ be a history-deterministic one-counter net. An $\exists$ strategy σ in the letter game is winning for $\exists$ if and only if the strategy σ only takes good transitions $\delta = (p, k) \xrightarrow{a,d} (p', k')$.*

Given a one-counter net $\mathcal{N}$, we say $\mathcal{N}$ satisfies *semilinear-strategy property* if for each transition $\delta = (q, a, d, q')$, the set of $k \in \mathbb{N}$ such that δ is a good transition at (q, k) is semilinear. That is, for each transition $\delta = (q, a, d, q') \in \Delta$, we have that the following set is eventually periodic

$$\mathcal{S}_\delta = \left\{ k : (q, k) \xrightarrow{a,d} (q', k') \text{ is a good transition at } (q, k) \right\}.$$

Lemma 16. *If a history-deterministic OCN $\mathcal{N} = (Q, \Sigma, q_0, \Delta, F)$ satisfies the semilinear-strategy property, then there is a language-equivalent deterministic OCA $\mathcal{D}$.*

Proof (Sketch). We assume the history-deterministic OCN $\mathcal{N}$ is such that it satisfies semilinear-strategy property. We shall first construct a non-deterministic one-counter automata $\mathcal{B}$, which can be determinised easily by removing a minimal set of transitions to get rid of non-determinism while still preserving the language. The non-deterministic one-counter automata $\mathcal{B}$ would essentially be designed so that the transitions in $\mathcal{B}$ correspond to the good transitions in $\mathcal{N}$, from any configuration. The eventual periodicity of the sets S_δ allows us to express this as a one-counter automaton, rather than as a labelled transition system with countably many states.

Intuitively, the automaton $\mathcal{B}$ is constructed such that the state space of the automaton stores in its memory the period and the initial block of the semilinear sets. The idea is that this automaton's runs would be in bijection with those runs that take only good transitions in the OCN $\mathcal{N}$. We know that such a run exists in $\mathcal{N}$ by Lemma 15, as $\mathcal{N}$ is history-deterministic. However, the counter values in $\mathcal{B}$ are 'scaled down' to only remember how many periods have passed, while counter value 0 indicates that the counter value in the original run would have been at most I. The exact value of the counter in a run of $\mathcal{N}$ can be inferred as a function of the state space and the counter value of $\mathcal{B}$.

Having shown that every history-deterministic one-counter net that satisfies semilinear-strategy property has a language equivalent DOCA, we proceed to show that every one-counter net satisfies semilinear-strategy property. We first display an example which solidifies an intuition of the above statement.

Example 17. Consider the net $\mathcal{N}_7$, as shown in Figure 3, where all states labelled q_F are accepting. This automaton is not history-deterministic. However, if the counter value at q_1 is not a multiple of 7, then $\exists$ can resolve the non-determinism from q_1. Observe that the automaton accepts words of the form $a^n\$b^k\$ \cdot (\heartsuit, \clubsuit)$ such that $k \leq n$. Consider the following play of $\forall$ in the letter game from q_0: For $7n$ steps he reads a, after which he reads a \$. So far, all transitions are deterministic. After that, assume he again reads, $7n$ many times, the letter b. This ensures that the transition ends at the state q_1 with counter value 0. If he reads \$ here, this is the only position where $\exists$ has a choice. Note that she has to choose between transitions leading to $q_\heartsuit$ and $q_\clubsuit$. However, since both the suffixes $\heartsuit$ and $\clubsuit$ are accepting and only one of $\heartsuit$ or $\clubsuit$ is accepting from either states, $\forall$ can ensure $\exists$ loses no matter what she picks. However, if $\forall$ had read a

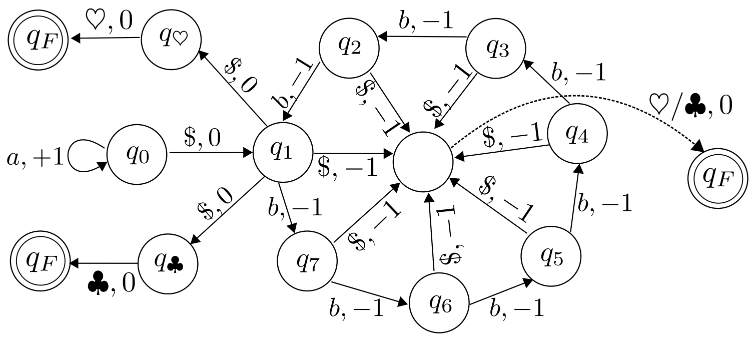

Fig. 3. The one-counter net $\mathcal{N}_7$ from Example 17

number of 'b's that was not a multiple of 7, the play of an accepting word would end at $q_\$$ which is accepting.

Lemma 18. *Every one-counter net $\mathcal{N}$ satisfies semilinear-strategy property.*

The proof of the above lemma follows from Theorem 12 on using a construction similar to the proof of Lemma 11 along wit. As an easy corollary of the above two lemmas, we get the following theorem.

Theorem 19. *Every history-deterministic OCN can be determinised to produce an equivalent deterministic OCA.*

An easy analysis of our proof combined with the results on the representation of simulation preorder ([24], Lemma 28) shows a doubly exponential upper bound on the size of the equivalent deterministic OCA constructed from the proof of the theorem above.

Remark 20. On the topic of expressivity of history-determinism, we conclude this subsection with a remark that history-deterministic OCNs are strictly less expressive than non-deterministic OCNs. This can be demonstrated with the language $\mathcal{L} = \{a^i\$b^j\$b^k \mid j \leq i \text{ or } k \leq i\}$. It is routine to verify that such a language is not accepted by any history-deterministic OCN, but this language can be accepted by a non-deterministic OCN. Note that history-determinism itself is not the limiting factor in accepting this language, as this language is accepted by a history-deterministic pushdown automaton [18].

4.2 Complexity of comparing languages of history-deterministic OCNs

Comparisons between languages of non-deterministic OCNs are undecidable [23], and even the restricted question of universality, is Ackermann-complete [33]. In this section, we show that for history-deterministic nets, these problems are no longer undecidable and have a significantly lower complexity when compared to non-deterministic nets.

Although we show that history-deterministic OCNs can be converted to a deterministic automaton, this determinisation does not help us answer these questions. This is because for deterministic OCAs, the problem of inclusion is undecidable [28]. Even though equality and universality for a deterministic OCA is **NL**-complete [34], the resulting deterministic OCA we get from determinisation of history-deterministic OCNs could be much larger than our input net, leading to a larger complexity.

Nevertheless, we show that checking for language inclusion, and hence checking language equivalence between two history-deterministic one-counter nets is in **PSPACE**. This is done by giving a polynomial-time reduction to the problem of deciding history-determinism, which we showed to be in **PSPACE** in Lemma 13. Moreover, combining our techniques with results of Kucera [35] gives us decidability in **P** for checking language universality of HD-OCNs.

Lemma 21. *Deciding language inclusion and language equivalence between two history-deterministic one-counter nets is in* **PSPACE***.*

We can show that the problem of checking language inclusion between two history-deterministic OCNs reduces to checking if a larger OCN (linear in the sum of the size of the two OCNs) is history-deterministic. Since language equivalence is essentially checking language inclusion both ways, we have the above results.

Lemma 22. *Deciding language universality for a given history-deterministic one-counter net is in* **P***.*

The problem of universality reduces to checking if the input net $\mathcal{M}$ simulates a finite state automata. This problem was shown to be **P** by Kucera ([35], Lemma 2), showing that universality is in **P**.

We therefore have the following theorem.

Theorem 23. *For nets $\mathcal{H}$ and $\mathcal{H}'$ that are history-deterministic, the problem of checking if $\mathcal{L}(\mathcal{H}) \subseteq \mathcal{L}(\mathcal{H}')$ as well as checking if $\mathcal{L}(\mathcal{H}) = \mathcal{L}(\mathcal{H}')$ can be done in* **PSPACE***. If $\mathcal{H}$ is instead a deterministic finite-state automaton, this problem can be solved in* **P***.*

We summarise known results and complexity of relevant results for comparison with other automata models in Table 1.

5 Extensions and Variations of OCN

We revisit the question of deciding history-determinism in this section for one-counter nets and its variants. In the first subsection, we tackle the question of how the complexity changes if the nets are encoded succinctly. We show that as expected, this increases the complexity of the problem from **PSPACE**-complete to **EXPSPACE**-complete. We then answer affirmatively to the question of whether adding zero-tests add too much power to one-counter nets by showing that the problem of deciding history-determinism becomes undecidable.

	$\mathcal{L} \subseteq \mathcal{L}'$	$\mathcal{L} = \mathcal{L}'$	$\mathcal{L} = \Sigma^*$
DOCN	**NL**-complete [33]	**NL**-complete [33]	**NL**-complete [33]
HOCN	In **PSPACE**	In **PSPACE**	In **P**
OCN	Undecidable [28]	Undecidable [23]	Ackermann-complete [33]
DOCA	Undecidable [28]	**NL**-complete [36]	**NL**-complete [36]

Table 1. Complexities for the problems of deciding language inclusion, equivalence and universality over deterministic OCN, HD-OCN, non-deterministic OCN and deterministic OCA.

5.1 Succinct Encoding of Counters

We consider a succinct representation of the input nets or a succinct one-counter net, where the transitions can allow for increments or decrements by integers (potentially greater than 1) that are represented in binary. Unsurprisingly, we show that checking for history-determinism becomes **EXPSPACE**-complete in this case. The upper bound follows from the previous proof of the **PSPACE** upper bound from Lemma 13 of deciding history-determinism for one-counter nets, where counter values are in unary. Any succinct one-counter net can be converted with only an exponential blow-up into another language equivalent net with unary encoding, preserving history-determinism thereby giving us an **EXPSPACE** upper bound.

Proposition 24. *Given a succinct one-counter net $\mathcal{N}$,deciding if $\mathcal{N}$ is history-deterministic is in* **EXPSPACE**.

However, much more work is needed to show a matching lower bound, which we do by giving a reduction from reachability games on succinct one-counter nets (SOCN). Intuitively, these games are played on the configuration graphs of a succinct OCN whose alphabet is a singleton. The states of this SOCN are partitioned among two players, denoted by $\wedge$ and $\vee$ who are responsible for choosing the transition from that state. The goal of $\vee$ is to take the play to a designated winning state with value 0. This problem of deciding the winner in the SOCN-reachability game was shown to be **EXPSPACE**-complete by Hunter [37] and later, several of its variants were also shown to have the same complexity [30]. A polynomial reduction from checking for history-determinism in a SOCN gives us **EXPSPACE**-hardness.

Lemma 25. *Given a SOCN $\mathcal{N}$, deciding if $\mathcal{N}$ is history-deterministic is* **EXPSPACE**-*hard.*

Proof (Sketch). Given an instance of a SOCN-reachability game on $\mathcal{N}$, We construct a SOCN $\mathcal{M}$ such that $\vee$ wins in the SOCN-reachability game on $\mathcal{N}$ if and only if $\forall$ wins in the letter game on $\mathcal{M}$.

The high-level idea of the construction is such that in a play of the letter game on $\mathcal{M}$, the players $\forall$ and $\exists$ create a transcript of a run of $\mathcal{N}$. This is done by $\forall$ ensuring that picking the letters in $\mathcal{M}$ corresponds to picking a transition out of $\vee$ states in $\mathcal{N}$. Since $\exists$ resolves the non-determinism in the letter game on $\mathcal{M}$, her choice of transitions correspond to transitions out of a $\wedge$ state in the SOCN-reachability game.

However, there are some subtleties in the construction as we need to ensure a few important aspects while constructing $\mathcal{M}$. Firstly, any sequence of letters chosen by $\forall$ in $\mathcal{M}$'s letter game so far must correspond to a run in $\mathcal{N}$ and secondly, the interplay between $\exists$'s and $\forall$'s choices in the letter game of $\mathcal{M}$ must correspond to the choices of the player $\wedge$ and $\vee$ respectively in the SOCN-reachability game of $\mathcal{N}$. These are the main challenges while constructing such an OCN $\mathcal{N}$ and they are resolved by the use of a few gadgets.

We conclude this subsection by combining Proposition 24 and Lemma 25 to obtain the following theorem.

Theorem 26. *Given a SOCN $\mathcal{N}$, deciding if $\mathcal{N}$ is history-deterministic is* **EXPSPACE***-complete.*

5.2 Deciding History-Determinsm for OCA

We show that, given a one-counter automaton $\mathcal{A}$, deciding if $\mathcal{A}$ is history-deterministic is undecidable. It was shown by Guha, Jecker, Lehtinen, and Zimmermann [18] that deciding if a given non-deterministic pushdown automaton is history-deterministic is undecidable. This extends their result to OCAs, which follows via a reduction from checking for language inclusion for deterministic one-counter automata (DOCA), which is known to be undecidable [28].

Theorem 27. *Given an OCA $\mathcal{A}$, deciding if $\mathcal{A}$ is history-deterministic is undecidable.*

Proof (Sketch). Consider the following problem :

DOCA Inclusion: Given two DOCAs $\mathcal{A}$ and $\mathcal{B}$, is $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$?

The above problem was shown to be undecidable in Section 5.1 of Valiant's thesis [28]. We show that the problem of deciding if a given one-counter automaton is history-deterministic is also undecidable, by giving a reduction from the DOCA inclusion problem to checking for history-determinism of a given OCA.

6 Discussion

We showed several decision problems related to history-determinism to be decidable over OCNs. This is unlike other classes of infinite-state systems that subsume them, where either some or all of these problems are undecidable.

We note that we only deal with realtime nets with no ϵ-transitions, but our results hold without too much modification when ϵ-transitions are present, as weak simulation over OCNs can be decided in **PSPACE** (and **EXPSPACE** for a succinct encoding), and the weak simulation pre-order is semilinear as well [24].

We also showed that testing the counter for zero made checking for history-determinism undecidable. Along these lines, one could ask about models like reversal bounded one-counter automata [38], or automata with bounded number of zero-tests, to gauge the frontier between decidability and undecidability on these systems.

Although not obvious from the main part of the paper, we are confident that our results could easily be extended to safety acceptance conditions. One could also ask, for instance, to look at reachability or Büchi and co-Büchi acceptance conditions and understand how history-determinism works in these models.

There are several questions about the expressivity of history-deterministic OCNs which we believe need further study. Overloading the notation and assuming DOCN, DOCA, OCN, HD-OCN and HD-OCA to denote the class of languages that are accepted by the corresponding models, we have shown that

$$\text{DOCN} \subseteq \text{HD-OCN} \subseteq \text{OCN} \cap \text{DOCA}.$$

An interesting problem would be to prove or disprove if any of these inclusions are strict. In fact, we don't have an example of a language that is accepted by a history-deterministic OCN which is not accepted by a deterministic OCN.

One could ask similar questions about expressivity of history-determinism in OCAs, i.e. if HD-OCA = DOCA. Although deciding history-determinism is undecidable, it might be possible for one to show that the language accepted by a history-deterministic OCA is as expressive as deterministic OCA. We remark that the 1-token game G_1 characterises history-determinisation for OCAs as well. Moreover, we can again show with similar techniques that if history-deterministic OCAs satisfy the semilinear-strategy property, then these languages can also be expressed by a deterministic OCA. The key part that we need to prove for determinisation of history-deterministic OCA would be the semilinear-strategy property. It would be interesting to see how such a proof would look like, given that checking for history-determinism is undecidable for OCAs.

Acknowledgements We would like to thank Dmitry Chistikov for listening to our conjectures and pointing us to important references. We are also grateful for his comments on our introduction. We are thankful to Neha Rino for carefully proofreading our paper, and suggesting improvements in our presentation. We also thank Sougata Bose, Piotrek Hofman, Filip Mazowiecki, David Purser, and Patrick Totzke for their insightful remarks on our draft, and for telling us about weak simulation. We are grateful to Shaull Almagor and Asaf Yeshurun for a fun talk about OCNs. Finally, we thank Marcin Jurdziński for his support, and for bringing us homemade rhubarb crumble.

References

1. Thomas Colcombet. Unambiguity in automata theory. In Jeffrey O. Shallit and Alexander Okhotin, editors, *Descriptional Complexity of Formal Systems - 17th In-*

ternational Workshop, DCFS 2015, Waterloo, ON, Canada, June 25-27, 2015. Proceedings, volume 9118 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015.

2. Denis Kuperberg and Anirban Majumdar. Width of non-deterministic automata. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 47:1–47:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
3. Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In Armin Biere and David Parker, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part I*, volume 12078 of *Lecture Notes in Computer Science*, pages 306–323. Springer, 2020.
4. Bader Abu Radi, Orna Kupferman, and Ofer Leshkowitz. A hierarchy of nondeterminism. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*, volume 202 of *LIPIcs*, pages 85:1–85:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
5. Emile Hazard and Denis Kuperberg. Explorable automata, May 2022. working paper or preprint.
6. Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, volume 4207 of *Lecture Notes in Computer Science*, pages 395–410. Springer, 2006.
7. Denis Kuperberg and Michal Skrzypczak. On determinisation of good-for-games automata. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 2015.
8. Thomas Colcombet. Forms of Determinism for Automata (Invited Talk). In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, volume 14 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1–23, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
9. Udi Boker, Denis Kuperberg, Orna Kupferman, and Michał Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In *Proceedings of the 40th International Conference on Automata, Languages, and Programming - Volume Part II*, ICALP'13, page 89–100, Berlin, Heidelberg, 2013. Springer-Verlag.
10. Udi Boker, Orna Kupferman, and Michal Skrzypczak. How Deterministic are Good-For-Games Automata‽. In Satya Lokam and R. Ramanujam, editors, *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2017)*, volume 93 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

11. Marc Bagnol and Denis Kuperberg. Büchi Good-for-Games Automata Are Efficiently Recognizable. In Sumit Ganguly and Paritosh Pandya, editors, *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018)*, volume 122 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
12. Udi Boker, Denis Kuperberg, Karoliina Lehtinen, and Michał Skrzypczak. On the Succinctness of Alternating Parity Good-For-Games Automata. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020)*, volume 182 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 41:1–41:13, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
13. Antonio Casares, Thomas Colcombet, and Karoliina Lehtinen. On the size of good-for-games rabin automata and its link with the memory in muller games. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 117:1–117:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
14. Bader Abu Radi and Orna Kupferman. Minimization and canonization of GFG transition-based automata. *Log. Methods Comput. Sci.*, 18(3), 2022.
15. Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009.
16. Udi Boker and Karoliina Lehtinen. History determinism vs. good for gameness in quantitative automata. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, pages 38:1–38:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
17. Udi Boker and Karoliina Lehtinen. Token games and history-deterministic quantitative automata. In Patricia Bouyer and Lutz Schröder, editors, *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*, volume 13242 of *Lecture Notes in Computer Science*, pages 120–139. Springer, 2022.
18. Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. A Bit of Nondeterminism Makes Pushdown Automata Expressive and Succinct. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 53:1–53:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
19. Karoliina Lehtinen and Martin Zimmermann. Good-for-games ω-pushdown automata. *Log. Methods Comput. Sci.*, 18(1), 2022.
20. Enzo Erlich, Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. History-deterministic parikh automata. *CoRR*, abs/2209.07745, 2022.

21. Thomas A. Henzinger, Karoliina Lehtinen, and Patrick Totzke. History-deterministic timed automata. In Bartek Klin, Slawomir Lasota, and Anca Muscholl, editors, *33rd International Conference on Concurrency Theory, CONCUR 2022, September 12-16, 2022, Warsaw, Poland*, volume 243 of *LIPIcs*, pages 14:1–14:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
22. Sougata Bose, Thomas A. Henzinger, Karoliina Lehtinen, Sven Schewe, and Patrick Totzke. History-deterministic timed automata are not determinizable. In Anthony W. Lin, Georg Zetzsche, and Igor Potapov, editors, *Reachability Problems - 16th International Conference, RP 2022, Kaiserslautern, Germany, October 17-21, 2022, Proceedings*, volume 13608 of *Lecture Notes in Computer Science*, pages 67–76. Springer, 2022.
23. Piotr Hofman, Richard Mayr, and Patrick Totzke. Decidability of weak simulation on one-counter nets. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 203–212. IEEE Computer Society, 2013.
24. Piotr Hofman, Slawomir Lasota, Richard Mayr, and Patrick Totzke. Simulation problems over one-counter nets. *Log. Methods Comput. Sci.*, 12(1), 2016.
25. Udi Boker and Karoliina Lehtinen. Good for games automata: From nondeterminism to alternation. In Wan J. Fokkink and Rob van Glabbeek, editors, *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*, volume 140 of *LIPIcs*, pages 19:1–19:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
26. Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. Parikh automata over infinite words, 2022.
27. David Gale and Frank M Stewart. Infinite games with perfect information. *Contributions to the Theory of Games*, 2(245-266):2–16, 1953.
28. Leslie G. Valiant. *Decision procedures for families of deterministic pushdown automata*. PhD thesis, University of Warwick, Coventry, UK, 1973.
29. Jirí Srba. Visibly pushdown automata: From language equivalence to simulation and bisimulation. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, volume 4207 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2006.
30. Petr Jancar, Petr Osicka, and Zdenek Sawa. EXPSPACE-hardness of behavioural equivalences of succinct one-counter nets. *CoRR*, abs/1801.01073, 2018.
31. Markus Holzer. On emptiness and counting for alternating finite automata. In Jürgen Dassow, Grzegorz Rozenberg, and Arto Salomaa, editors, *Developments in Language Theory II, At the Crossroads of Mathematics, Computer Science and Biology, Magdeburg, Germany, 17-21 July 1995*, pages 88–97. World Scientific, Singapore, 1995.
32. Petr Jancar and Zdenek Sawa. A note on emptiness for alternating finite automata with a one-letter alphabet. *Inf. Process. Lett.*, 104(5):164–167, 2007.
33. Piotr Hofman and Patrick Totzke. Trace inclusion for one-counter nets revisited. In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings*, volume 8762 of *Lecture Notes in Computer Science*, pages 151–162. Springer, 2014.
34. Stanislav Böhm, Stefan Göller, and Petr Jancar. Equivalence of deterministic one-counter automata is NL-complete. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 131–140. ACM, 2013.

35. Antonín Kucera. On simulation-checking with sequential systems. In Jifeng He and Masahiko Sato, editors, *Advances in Computing Science - ASIAN 2000, 6th Asian Computing Science Conference, Penang, Malaysia, November 25-27, 2000, Proceedings*, volume 1961 of *Lecture Notes in Computer Science*, pages 133–148. Springer, 2000.
36. Stanislav Böhm and Stefan Göller. Language equivalence of deterministic real-time one-counter automata is nl-complete. In Filip Murlak and Piotr Sankowski, editors, *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 2011.
37. Paul Hunter. Reachability in succinct one-counter games. In Mikolaj Bojanczyk, Slawomir Lasota, and Igor Potapov, editors, *Reachability Problems - 9th International Workshop, RP 2015, Warsaw, Poland, September 21-23, 2015, Proceedings*, volume 9328 of *Lecture Notes in Computer Science*, pages 37–49. Springer, 2015.
38. Oscar H. Ibarra. Automata with reversal-bounded counters: A survey. In Helmut Jürgensen, Juhani Karhumäki, and Alexander Okhotin, editors, *Descriptional Complexity of Formal Systems - 16th International Workshop, DCFS 2014, Turku, Finland, August 5-8, 2014. Proceedings*, volume 8614 of *Lecture Notes in Computer Science*, pages 5–22. Springer, 2014.