History-Deterministic Parikh Automata

Erlich, Enzo; Guha, Shibashis; Jecker, Ismaël; Lehtinen, Karoliina; Zimmermann, Martin

Published in:
34th International Conference on Concurrency Theory, CONCUR 2023

DOI (link to publication from Publisher):
10.4230/LIPICS.CONCUR.2023.31

Publication date:
2023

Document Version
Publisher's PDF, also known as Version of record

History-Deterministic Parikh Automata

Enzo Erlich
ENS Rennes, France

Shibashis Guha
Tata Institute of Fundamental Research, Mumbai, India

Ismaël Jecker
University of Warsaw, Poland

Karoliina Lehtinen
CNRS, Aix-Marseille University, LIS, France

Martin Zimmermann
Aalborg University, Denmark

Abstract

Parikh automata extend finite automata by counters that can be tested for membership in a semilinear set, but only at the end of a run. Thereby, they preserve many of the desirable properties of finite automata. Deterministic Parikh automata are strictly weaker than nondeterministic ones, but enjoy better closure and algorithmic properties.

This state of affairs motivates the study of intermediate forms of nondeterminism. Here, we investigate history-deterministic Parikh automata, i.e., automata whose nondeterminism can be resolved on the fly. This restricted form of nondeterminism is well-suited for applications which classically call for determinism, e.g., solving games and composition.

We show that history-deterministic Parikh automata are strictly more expressive than deterministic ones, incomparable to unambiguous ones, and enjoy almost all of the closure properties of deterministic automata.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Parikh automata, History-determinism, Reversal-bounded Counter Machines

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2023.31

Related Version *Full Version*: `https://arxiv.org/abs/2209.07745` [13]

Funding *Shibashis Guha*: Supported by Indian Science and Engineering Research Board (SERB) grant SRG/2021/000466.
Ismaël Jecker: Supported by the ERC grant 950398 (INFSYS).
Martin Zimmermann: Supported by DIREC – Digital Research Centre Denmark.

1 Introduction

Some of the most profound (and challenging) questions of theoretical computer science are concerned with the different properties of deterministic and nondeterministic computation, the P vs. NP problem being arguably the most important and surely the most well-known one. However, even in the more modest setting of automata theory, there is a tradeoff between deterministic and nondeterministic automata with far-reaching consequences for, e.g., the automated verification of finite-state systems. In the automata-based approach to model checking, for example, one captures a finite-state system $\mathcal{S}$ and a specification φ by automata $\mathcal{A}_\mathcal{S}$ and $\mathcal{A}_\varphi$ and then checks whether $L(\mathcal{A}_\mathcal{S}) \subseteq L(\mathcal{A}_\varphi)$ holds, i.e., whether every execution of $\mathcal{S}$ satisfies the specification φ. To do so, one tests $L(\mathcal{A}_\mathcal{S}) \cap \overline{L(\mathcal{A}_\varphi)}$ for emptiness. Hence, one is interested in expressive automata models that have good closure and algorithmic properties. Nondeterminism yields conciseness (think DFA's vs. NFA's) or

even more expressiveness (think pushdown automata) while deterministic automata often have better algorithmic properties and better closure properties (again, think, e.g., pushdown automata).

Limited forms of nondeterminism constitute an appealing middle ground as they often combine the best of both worlds, e.g., increased expressiveness in comparison to deterministic automata and better algorithmic and closure properties than nondeterministic ones. A classical, and well-studied, example are unambiguous automata, i.e., nondeterministic automata that have at most one accepting run for every input. For example, unambiguous finite automata can be exponentially smaller than deterministic ones while unambiguous pushdown automata are more expressive than deterministic ones [25].

Another restricted class of nondeterministic automata is that of residual automata [12], automata where every state accepts a residual language of the automaton's language. For every regular language there exists a residual automaton. While there exist residual automata that can be exponentially smaller than DFA, there also exist languages for which NFA can be exponentially smaller than residual automata [12].

More recently, another notion of limited nondeterminism has received considerable attention: history-deterministic automata [9, 24][1] are nondeterministic automata whose nondeterminism can be resolved based on the run constructed thus far, but independently of the remainder of the input. This property makes history-deterministic automata suitable for the composition with games, trees, and other automata, applications which classically require deterministic automata. History-determinism has been studied in the context of regular [1, 24, 28], pushdown [22, 29], quantitative [3, 9], and timed automata [23]. For automata that can be determinized, history-determinism offers the potential for succinctness (e.g., co-Büchi automata [28]) while for automata that cannot be determinized, it even offers the potential for increased expressiveness (e.g., pushdown automata [22, 29]). In the quantitative setting, the exact power of history-determinism depends largely on the type of quantitative automata under consideration. So far, it has been studied for quantitative automata in which runs accumulate weights into a value using a value function such as `Sum, LimInf, Average`, and that assign to a word the supremum among the values of its runs. For these automata, history-determinism turns out to have interesting applications for quantitative synthesis [2]. Here, we continue this line of work by investigating history-deterministic Parikh automata, a mildly quantitative form of automata.

Parikh automata, introduced by Klaedtke and Rueß [27], consist of finite automata, augmented with counters that can only be incremented. A Parikh automaton only accepts a word if the final counter-configuration is within a semilinear set specified in the automaton. As the counters do not interfere with the control flow of the automaton, that is, counter values do not affect whether transitions are enabled, they allow for mildly quantitative computations without the full power of vector addition systems or other more powerful models.

For example the language of words over the alphabet $\{0, 1\}$ having a prefix with strictly more 1's than 0's is accepted by a Parikh automaton that starts by counting the number of 0's and 1's and after some prefix nondeterministically stops counting during the processing of the input. It accepts if the counter counting the 1's is, at the end of the run, indeed larger than the counter counting the 0's. Note that the nondeterministic choice can be made based on the word processed so far, i.e., as soon as a prefix with more 1's than 0's is encountered, the counting is stopped. Hence, the automaton described above is in fact history-deterministic.

[1] There is a closely related notion, good-for-gameness, which is often, but not always equivalent [2] (despite frequently being used interchangeably in the past).

Klaedtke and Rueß [27] showed Parikh automata to be expressively equivalent to a quantitative version of existential weak MSO that allows for reasoning about set cardinalities. Their expressiveness also coincides with that of reversal-bounded counter machines [27], in which counters can go from decrementing to incrementing only a bounded number of times, but in which counters affect control flow [26]. The weakly unambiguous restriction of Parikh automata, that is, those that have at most one accepting run, on the other hand, coincide with unambiguous reversal-bounded counter machines [4]. Parikh automata are also expressively equivalent to weighted finite automata over the groups $(\mathbb{Z}^k, +, 0)$ [11, 31] for $k \geqslant 1$. This shows that Parikh automata accept a natural class of quantitative specifications.

Despite their expressiveness, Parikh automata retain some decidability: nonemptiness, in particular, is NP-complete [14]. For weakly unambiguous Parikh automata, inclusion [7] and regular separability [8] are decidable as well. Figueira and Libkin [14] also argued that this model is well-suited for querying graph databases, while mitigating some of the complexity issues related with more expressive query languages. Further, they have been used in the model checking of transducer properties [16].

As Parikh automata have been established as a robust and useful model, many variants thereof exist: pushdown (visibly [10] and otherwise [32]), two-way with [10] and without stack [15], unambiguous [6], and weakly unambiguous [4] Parikh automata, to name a few.

Our contribution. We introduce history-deterministic Parikh automata (HDPA) and study their expressiveness, their closure properties, and their algorithmic properties.

Our main result shows that history-deterministic Parikh automata are more expressive than deterministic ones (DPA), but less expressive than nondeterministic ones (PA). Furthermore, we show that they are of incomparable expressiveness to both classes of unambiguous Parikh automata found in the literature, but equivalent to history-deterministic reversal-bounded counter machines, another class of history-deterministic automata that is studied here for the first time. These results show that history-deterministic Parikh automata indeed constitute a novel class of languages capturing quantitative features.

Secondly, we show that history-deterministic Parikh automata satisfy almost the same closure properties as deterministic ones, the only difference being non-closure under complementation. This result has to be contrasted with unambiguous Parikh automata being closed under complement [6]. Thus, history-determinism is a too strong form of nondeterminism to preserve closure under complementation, a phenomenon that has already been observed in the case of pushdown automata [22, 29].

Finally, we study the algorithmic properties of history-deterministic Parikh automata. Most importantly, safety model checking for HDPA is decidable, as it is for PA. The problem asks, given a system and a set of *bad* prefixes specified by an automaton, whether the system has an execution that has a bad prefix. This allows, for example, to check properties of an arbiter of some shared resource like "the accumulated waiting time between requests and responses of client 1 is always at most twice the accumulated waiting time for client 2 and vice versa". Note that this property is not ω-regular.

Non-emptiness and finiteness are also both decidable for HDPA (as they are for nondeterministic automata), but universality, inclusion, equivalence, and regularity are not. This is in stark contrast to unambiguous Parikh automata (and therefore also deterministic ones), for which *all* of these problems are decidable. Finally, we show that it is undecidable whether a Parikh automaton is history-deterministic and whether it is equivalent to a history-deterministic one.

Note that we consider only automata over finite words here, but many of our results can straightforwardly be transferred to Parikh automata over infinite words, introduced independently by Guha et al. [21] and Grobler et al. [18, 19].

All proofs omitted due to space restrictions can be found in [13].

2 Definitions

An alphabet is a finite nonempty set Σ of letters. As usual, ε denotes the empty word, Σ^* denotes the set of finite words over Σ, Σ^+ denotes the set of finite nonempty words over Σ, and Σ^ω denotes the set of infinite words over Σ. The length of a finite word w is denoted by $|w|$ and, for notational convenience, we define $|w| = \infty$ for all infinite words w. Finally, $|w|_a$ denotes the number of occurrences of the letter a in a finite word w.

Semilinear Sets. We denote the set of nonnegative integers by $\mathbb{N}$. Given vectors $\vec{v} = (v_0, \ldots, v_{d-1}) \in \mathbb{N}^d$ and $\vec{v}\,' = (v'_0, \ldots, v'_{d'-1}) \in \mathbb{N}^{d'}$, we define their concatenation $\vec{v} \cdot \vec{v}\,' = (v_0, \ldots, v_{d-1}, v'_0, \ldots, v'_{d'-1}) \in \mathbb{N}^{d+d'}$. We lift the concatenation of vectors to sets $D \subseteq \mathbb{N}^d$ and $D' \subseteq \mathbb{N}^{d'}$ via $D \cdot D' = \{\vec{v} \cdot \vec{v}\,' \mid \vec{v} \in D \text{ and } \vec{v}\,' \in D'\}$.

Let $d \geqslant 1$. A set $C \subseteq \mathbb{N}^d$ is linear if there are vectors $\vec{v}_0, \ldots, \vec{v}_k \in \mathbb{N}^d$ such that

$$C = \left\{ \vec{v}_0 + \sum\nolimits_{i=1}^{k} c_i \vec{v}_i \;\middle|\; c_i \in \mathbb{N} \text{ for } i = 1, \ldots, k \right\}.$$

Furthermore, a subset of $\mathbb{N}^d$ is semilinear if it is a finite union of linear sets.

▶ **Example 1.** The sets $\{(n, n) \mid n \in \mathbb{N}\} = \{(0,0) + c(1,1) \mid c \in \mathbb{N}\}$ and $\{(n, 2n) \mid n \in \mathbb{N}\} = \{(0,0) + c(1,2) \mid c \in \mathbb{N}\}$ are linear, so their union is semilinear. Further, the set $\{(n, n') \mid n < n'\} = \{(0,1) + c_1(1,1) + c_2(0,1) \mid c_1, c_2 \in \mathbb{N}\}$ is linear and thus also semilinear.

▶ **Proposition 2** ([17]). *If $C, C' \subseteq \mathbb{N}^d$ are semilinear, then so are $C \cup C'$, $C \cap C'$, $\mathbb{N}^d \setminus C$, as well as $\mathbb{N}^{d'} \cdot C$ and $C \cdot \mathbb{N}^{d'}$ for every $d' \geqslant 1$.*

Finite Automata. A (nondeterministic) finite automaton (NFA) $\mathcal{A} = (Q, \Sigma, q_I, \Delta, F)$ over Σ consists of the finite set Q of states containing the initial state q_I, the alphabet Σ, the transition relation $\Delta \subseteq Q \times \Sigma \times Q$, and the set $F \subseteq Q$ of accepting states. The NFA is deterministic (i.e., a DFA) if for every state $q \in Q$ and every letter $a \in \Sigma$, there is at most one $q' \in Q$ such that (q, a, q') is a transition of $\mathcal{A}$.

A run of $\mathcal{A}$ is a (possibly empty) sequence $(q_0, a_0, q_1)(q_1, a_1, q_2) \cdots (q_{n-1}, a_{n-1}, q_n)$ of transitions with $q_0 = q_I$. It processes the word $a_0 a_1 \cdots a_{n-1} \in \Sigma^*$. The run is accepting if it is either empty and the initial state is accepting or if it is nonempty and q_n is accepting. The language $L(\mathcal{A})$ of $\mathcal{A}$ contains all finite words $w \in \Sigma^*$ such that $\mathcal{A}$ has an accepting run processing w.

Parikh Automata. Let Σ be an alphabet, $d \geqslant 1$, and D a finite subset of $\mathbb{N}^d$. Furthermore, let $w = (a_0, \vec{v}_0) \cdots (a_{n-1}, \vec{v}_{n-1})$ be a word over $\Sigma \times D$. The Σ-projection of w is $p_\Sigma(w) = a_0 \cdots a_{n-1} \in \Sigma^*$ and its *extended Parikh image* is $\Phi_e(w) = \sum_{j=0}^{n-1} \vec{v}_j \in \mathbb{N}^d$ with the convention $\Phi_e(\varepsilon) = \vec{0}$, where $\vec{0}$ is the d-dimensional zero vector.

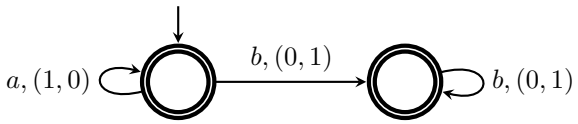

Figure 1 The automaton for Example 3.

A Parikh automaton (PA) is a pair $(\mathcal{A}, C)$ such that $\mathcal{A}$ is an NFA over $\Sigma \times D$ for some input alphabet Σ and some finite $D \subseteq \mathbb{N}^d$ for some $d \geqslant 1$, and $C \subseteq \mathbb{N}^d$ is semilinear. The language of $(\mathcal{A}, C)$ consists of the Σ-projections of words $w \in L(\mathcal{A})$ whose extended Parikh image is in C, i.e.,

$$L(\mathcal{A}, C) = \{p_\Sigma(w) \mid w \in L(\mathcal{A}) \text{ with } \Phi_e(w) \in C\}.$$

The Parikh automaton $(\mathcal{A}, C)$ is deterministic if for every state q of $\mathcal{A}$ and every $a \in \Sigma$, there is at most one pair $(q', \vec{v}) \in Q \times D$ such that $(q, (a, \vec{v}), q')$ is a transition of $\mathcal{A}$. Note that this definition does *not* coincide with $\mathcal{A}$ being a DFA: As mentioned above, $\mathcal{A}$ accepts words over $\Sigma \times D$ while $(\mathcal{A}, C)$ accepts words over Σ. Therefore, determinism is defined with respect to Σ only.

Note that the above definition of $L(\mathcal{A}, C)$ coincides with the following alternative definition via accepting runs: A run ρ of $(\mathcal{A}, C)$ is a run

$$\rho = (q_0, (a_0, \vec{v}_0), q_1)(q_1, (a_1, \vec{v}_1), q_2) \cdots (q_{n-1}, (a_{n-1}, \vec{v}_{n-1}), q_n)$$

of $\mathcal{A}$. We say that ρ *processes* the word $a_0 a_1 \cdots a_{n-1} \in \Sigma^*$, i.e., the $\vec{v}_j$ are ignored, and that ρ's extended Parikh image is $\sum_{j=0}^{n-1} \vec{v}_j$. The run is accepting if it is either empty and both the initial state of $\mathcal{A}$ is accepting and the zero vector (the extended Parikh image of the empty run) is in C, or if it is nonempty, q_n is accepting, and ρ's extended Parikh image is in C. Finally, $(\mathcal{A}, C)$ accepts $w \in \Sigma^*$ if it has an accepting run processing w.

▶ **Example 3.** Consider the deterministic PA $(\mathcal{A}, C)$ with $\mathcal{A}$ in Figure 1 and $C = \{(n, n) \mid n \in \mathbb{N}\} \cup \{(n, 2n) \mid n \in \mathbb{N}\}$ (cf. Example 1). It accepts the language $\{a^n b^n \mid n \in \mathbb{N}\} \cup \{a^n b^{2n} \mid n \in \mathbb{N}\}$.

3 History-deterministic Parikh Automata

In this section, we introduce history-deterministic Parikh automata and give examples.

Let $(\mathcal{A}, C)$ be a PA with $\mathcal{A} = (Q, \Sigma \times D, q_I, \Delta, F)$. For a function $r \colon \Sigma^+ \to \Delta$ we define its iteration $r^* \colon \Sigma^* \to \Delta^*$ via $r^*(\varepsilon) = \varepsilon$ and $r^*(a_0 \cdots a_n) = r^*(a_0 \cdots a_{n-1}) \cdot r(a_0 \cdots a_n)$. We say that r is a resolver for $(\mathcal{A}, C)$ if, for every $w \in L(\mathcal{A}, C)$, $r^*(w)$ is an accepting run of $(\mathcal{A}, C)$ processing w. Further, we say that $(\mathcal{A}, C)$ is history-deterministic (i.e., an HDPA) if it has a resolver.

▶ **Example 4.** Fix $\Sigma = \{0, 1\}$ and say that a word $w \in \Sigma^*$ is non-Dyck if $|w|_0 < |w|_1$. We consider the language $N \subseteq \Sigma^+$ of words that have a non-Dyck prefix. It is accepted by the PA $(\mathcal{A}, C)$ where $\mathcal{A}$ is depicted in Figure 2 and $C = \{(n, n') \mid n < n'\}$ (cf. Example 1). Intuitively, in the initial state q_c, the automaton counts the number of 0's and 1's occurring in some prefix, nondeterministically decides to stop counting by moving to q_n (this is the only nondeterminism in $\mathcal{A}$), and accepts if there are more 1's than 0's in the prefix.

The nondeterministic choice can be made only based on the prefix processed so far, i.e., as soon as the first non-Dyck prefix is encountered, the resolver proceeds to state q_n, thereby ending the prefix. Formally, the function

$$wb \mapsto \begin{cases} (q_c, (b, (1-b, b)), q_c) & \text{if } wb \text{ has no non-Dyck prefix,} \\ (q_c, (b, (1-b, b)), q_n) & \text{if } wb \text{ is non-Dyck, but } w \text{ has no non-Dyck prefix,} \\ (q_n, (b, (0, 0)), q_n) & \text{if } w \text{ has a non-Dyck prefix,} \end{cases}$$

is a resolver for $(\mathcal{A}, C)$.

▶ Remark 5. As a resolver resolves nondeterminism and a DPA has no nondeterminism to resolve, every DPA is history-deterministic.

4 Expressiveness

In this section, we study the expressiveness of HDPA by comparing them to related automata models, e.g., deterministic and nondeterministic Parikh automata, unambiguous Parikh automata (capturing another restricted notion of nondeterminism), and reversal-bounded counter machines (which are known to be related to Parikh automata). Overall, we obtain the relations shown in Figure 3, where the additional classes of languages and the separating languages will be introduced throughout this section.

We begin by stating and proving a pumping lemma for HDPA. The basic property used here, just as for the pumping lemmata for PA and DPA [5], is that shuffling around cycles of a run does not change whether it is accepting or not, as acceptance only depends on the last state of the run being accepting and the vectors (and their multiplicity) that appear on the run, but not the order of their appearance.

▶ **Lemma 6.** *Let $(\mathcal{A}, C)$ be an HDPA with $L(\mathcal{A}, C) \subseteq \Sigma^*$. Then, there exist $p, \ell \in \mathbb{N}$ such that every $w \in \Sigma^*$ with $|w| > \ell$ can be written as $w = uvxvz$ such that*

- *$0 < |v| \leqslant p$, $|x| > p$, and $|uvxv| \leqslant \ell$, and*
- *for all $z' \in \Sigma^*$: if $uvxvz' \in L(\mathcal{A}, C)$, then also $uv^2xz' \in L(\mathcal{A}, C)$ and $uxv^2z' \in L(\mathcal{A}, C)$.*

Proof. Fix some resolver r for $(\mathcal{A}, C)$. Note that the definition of a resolver only requires $r^*(w)$ to be a run processing w for those $w \in L(\mathcal{A}, C)$. Here, we assume without loss of generality that $r^*(w)$ is a run processing w for each $w \in \Sigma^*$. This can be achieved by completing $\mathcal{A}$ (by adding a nonaccepting sink state and transitions to the sink where necessary) and redefining r where necessary (which is only the case for inputs that cannot be extended to a word in $L(\mathcal{A}, C)$).

A cycle is a nonempty finite run infix

$$(q_0, a_0, q_1)(q_1, a_1, q_2) \cdots (q_{n-1}, a_{n-1}, q_n)(q_n, a_n, q_0)$$

starting and ending in the same state and such that the q_j are pairwise different. Now, let p be the number of states of $\mathcal{A}$ and let m be the number of cycles of $\mathcal{A}$. Note that every run infix containing at least p transitions contains a cycle.

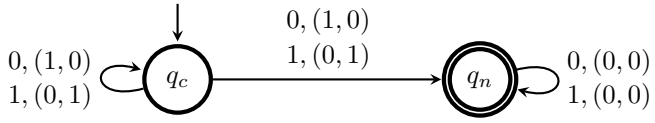

Figure 2 The automaton for Example 4.

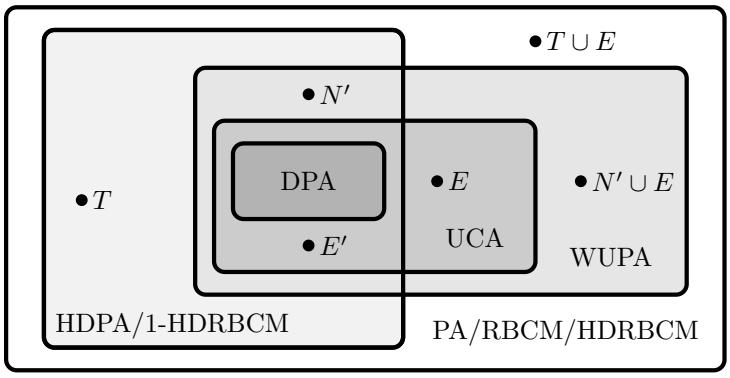

Figure 3 The classes of languages accepted by different models of Parikh automata.

We define $\ell = (p+1)(2m+1)$, consider a word $w \in \Sigma^*$ with $|w| > \ell$, and let $\rho = r^*(w)$ be the run of $\mathcal{A}$ induced by r which processes w. We split ρ into $\rho_0\rho_1 \cdots \rho_{2m}\rho'$ such that each ρ_j contains $p+1$ transitions. Then, each ρ_j contains a cycle and there are j_0, j_1 with $j_1 > j_0 + 1$ such that ρ_{j_0} and ρ_{j_1} contain the same cycle. Now, let

- ρ_v be the cycle in ρ_{j_0} and ρ_{j_1},
- ρ_u be the prefix of ρ before the first occurrence of ρ_v in ρ_{j_0}, and
- ρ_x be the infix of ρ between the first occurrences of ρ_v in ρ_{j_0} and ρ_{j_1}.

Furthermore, let $u, v, x \in \Sigma^*$ be the inputs processed by ρ_u, ρ_v, and ρ_x respectively. Then, we indeed have $0 < |v| \leqslant p$ (as we consider simple cycles), $|x| > p$ (as $j_1 > j_0 + 1$), and $|uvxv| \leqslant \ell$.

Note that $\rho_u\rho_v\rho_x\rho_v$, $\rho_u\rho_v^2\rho_x$, and $\rho_u\rho_x\rho_v^2$ are all runs of $\mathcal{A}$ which process $uvxv$, uv^2x, and uxv^2 respectively. Furthermore, all three runs end in the same state and their extended Parikh images are equal, as we only shuffled pieces around.

Now, consider some z' such that $uvxvz' \in L(\mathcal{A}, C)$. Then $r^*(uvxvz')$ is an accepting run, and of the form $\rho_u\rho_v\rho_x\rho_v\rho_{z'}$ for some $\rho_{z'}$ processing z'. Now, $\rho_u\rho_v^2\rho_x\rho_{z'}$, and $\rho_u\rho_x\rho_v^2\rho_{z'}$ are accepting runs of $(\mathcal{A}, C)$ (although not necessarily induced by r) processing uv^2xz' and uxv^2z', respectively. Thus, $uv^2xz' \in L(\mathcal{A}, C)$ and $uxv^2z' \in L(\mathcal{A}, C)$. ◀

It is instructive to compare our pumping lemma for HDPA to those for PA and DPA [5]:

- The pumping lemma for PA states that every *long* word accepted by a PA can be decomposed into $uvxvz$ as above such that both uv^2xz and uxv^2z are accepted as well. This statement is weaker than ours, as it only applies to the two words obtained by moving a v while our pumping lemma applies to any suffix z'. This is possible, as the runs of an HDPA on words of the form $uvxvz'$ (for fixed $uvxv$), induced by a resolver, all coincide on their prefixes processing $uvxv$. This is not necessarily the case in PA.
- The pumping lemma for DPA states that every *long* word (not necessarily accepted by the automaton) can be decomposed into $uvxvz$ as above such that $uvxv$, uv^2x, and uxv^2 are all equivalent with respect to the Myhill-Nerode equivalence. This statement is stronger than ours, as Myhill-Nerode equivalence is concerned both with the language of the automaton and its complement. But similarly to our pumping lemma, the one for DPA applies to all possible suffixes z'.

Now, we apply the pumping lemma to compare the expressiveness of HDPA, DPA, and PA.

▶ **Theorem 7.** *HDPA are more expressive than DPA, but less expressive than PA.*

Proof. First, we consider the separation between DPA and HDPA. The language N from Example 4, which is accepted by an HDPA, is known to be not accepted by any DPA: DPA are closed under complementation [27] while the complement of N is not even accepted by any PA [6].

To show that PA are more expressive than HDPA, consider the language $E = \{a, b\}^* \cdot \{a^n b^n \mid n > 0\}$, which can easily be seen to be accepted by a PA. We show that E is not accepted by any HDPA[2] via an application of the pumping lemma.

To this end, assume there is some HDPA $(\mathcal{A}, C)$ accepting E, and let p, ℓ as in the pumping lemma. We pick $w = (a^{p+1}b^{p+1})^\ell$, which we decompose as $uvxvz$ with the properties guaranteed by the pumping lemma. In particular, we have $|v| \leqslant p$, therefore $v \in a^*b^* + b^*a^*$. We consider two cases depending on the last letter of v. In each one, we show the existence of a word z' such that the word $uvxvz'$ is in the language E, yet either uv^2xz' or uxv^2z' is not. This yields the desired contradiction to the pumping lemma.

1. First, assume that the last letter of v is an a. Since $|x| > p$ and x appears between two copies of v in $(a^{p+1}b^{p+1})^\ell$, the infix xv contains at least one full b-block: we have $xv = x'b^{p+1}a^k$ with $x' \in \{a, b\}^*$ and $0 < k \leqslant p + 1$. We set $z' = a^{p+1-k}b^{p+1}$. Hence, $uvxvz' = uvx'b^{p+1}a^{p+1}b^{p+1} \in E$. We show that $uv^2xz' \notin E$ by differentiating two cases:
   a. If $v = a^i$ for some i, which must satisfy $0 < i \leqslant k$, then uv^2xz' is not in E as it ends with $b^{p+1}a^{p+1-i}b^{p+1}$.
   b. Otherwise, we must have $v = b^ia^k$ with $0 < i < p$. Then, uv^2xz' is not in E as it ends with $b^{p+1-i}a^{p+1-k}b^{p+1}$.
2. Otherwise, the last letter of v is a b. Since $|x| > p$ and x appears between two copies of v in $(a^{p+1}b^{p+1})^\ell$, the infix xv contains at least one full a-block: we have $xv = x'a^{p+1}b^k$ with $x' \in \{a, b\}^*$ and $0 < k \leqslant p + 1$. This time we set $z' = b^{p+1-k}$. Thus, $uvxvz' = uvx'a^{p+1}b^{p+1} \in E$, and we differentiate two cases to show that $uxv^2z' \notin E$:
   a. If $v = b^i$ for some i, which must satisfy $0 < i \leqslant p$, then uxv^2z' ends with b^{p+i+1}. However, each of its a-blocks has length $p + 1$, as moving $v = b^i$ with $i \leqslant p$ does not merge any a-blocks. Hence, uxv^2z' is not in E.
   b. Otherwise, we must have $v = a^ib^k$ with $0 < i < p$. Then, uxv^2z' is not in E as it ends with $v^2z' = a^ib^ka^ib^{p+1}$. ◀

4.1 History-determinism vs. Unambiguity

After having placed history-deterministic Parikh automata strictly between deterministic and nondeterministic ones, we now compare them to unambiguous Parikh automata, another class of automata whose expressiveness lies strictly between that of DPA and PA. In the literature, there are two (nonequivalent) forms of unambiguous Parikh automata. We consider both of them here.

Cadilhac et al. studied unambiguity in Parikh automata in the guise of unambiguous constrained automata (UCA) [6]. Constrained automata are a related model and effectively equivalent to PA. Intuitively, an UCA $(\mathcal{A}, C)$ over an alphabet Σ consists of an unambiguous ε-NFA $\mathcal{A}$ over Σ, say with d transitions, and a semilinear set $C \subseteq \mathbb{N}^d$, i.e., C has one dimension for each transition in $\mathcal{A}$. It accepts a word $w \in \Sigma^*$ if $\mathcal{A}$ has an accepting run processing w (due to unambiguity this run must be unique) such that the Parikh image of the run (recording the number of times each transition occurs in the run) is in C.

[2] Note that the related language $\{a, b\}^* \cdot \{a^n \# a^n \mid n \in \mathbb{N}\}$ is not accepted by any DPA [5].

On the other hand, Bostan et al. introduced so-called weakly-unambiguous Parikh automata (WUPA) [4]. Intuitively, a WUPA $(\mathcal{A}, C)$ over Σ is a classical PA as introduced here where every input over Σ has at most one accepting run (in the sense of the definition in Section 2). Bostan et al. discuss the different definitions of unambiguity and in particular show that every UCA is a WUPA, but that WUPA are strictly more expressive [4]. Here, we compare the expressiveness of HDPA to that of UCA and WUPA.

The language E from the proof of Theorem 7 is accepted by an UCA [6] and a WUPA [4], but not by any HDPA (see Theorem 7). On the other hand, the language

$$T = \{c^{n_0} d c^{n_1} d \cdots c^{n_k} d \mid k \geq 1, n_0 = 1, \text{ and } n_{j+1} \neq 2n_j \text{ for some } 0 \leq j < k\}$$

is not accepted by any WUPA, and hence also not by any UCA [4], but there is an HDPA accepting it. Hence, these two languages show that these three classes of automata have incomparable expressiveness.

▶ **Theorem 8.** *The expressiveness of HDPA is neither comparable with that of UCA nor with that of WUPA.*

Finally, we show that all intersections between the different classes introduced above are nonempty.

▶ **Theorem 9.**

1. *There is a language that is accepted by an HDPA and by an UCA, but not by any DPA.*
2. *There is a language that is accepted by an HDPA and by a WUPA, but not by any UCA.*
3. *There is a language that is accepted by a PA, but not by any HDPA nor by any WUPA.*
4. *There is a language that is accepted by a WUPA, but not by any HDPA nor by any UCA.*

Here, we give proof sketches, full proofs can be found in [13].

Recall that the language $E = \{a, b\}^* \cdot \{a^n b^n \mid n > 0\}$ from Theorem 7 is accepted by an UCA, but not by any HDPA. However, a slight modification allows it to be accepted by both types of automata, but not by a deterministic automaton: We show that

$$E' = \{c^m \{a, b\}^{m-1} b a^n b^n \mid m, n > 0\}$$

has the desired property. Intuitively, the prefix c^m allows us to resolve the nondeterminism on-the-fly, but nondeterminism is still required.

The second separation relies on a similar trick: The language

$$N' = \{c^n w \mid w = a_0 \cdots a_k \in \{0, 1\}^*, |w| \geq n > 0, \text{ and } a_0 \cdots a_{n-1} \text{ is non-Dyck}\},$$

which is a variation of the language N of words that have a non-Dyck prefix, was shown to be accepted by a WUPA, but not by any UCA [4]. It is straightforward to show that it is also accepted by an HDPA.

The last two results follow easily from closure properties: The union $T \cup E$ is neither accepted by any HDPA nor by any WUPA, as both models are closed under intersection[3], i.e., $(T \cup E) \cap \{a, b\}^* = E$ and $(T \cup E) \cap \{c, d\}^* = T$ yield the desired separation. A similar argument works for $N' \cup E$, which is accepted by some WUPA (as WUPA are closed under disjoint unions) but not by any HDPA nor by any UCA, as both classes are closed under intersection.

[3] For WUPA, this was shown by Bostan et al. [4], for HDPA this is shown in the next section.

4.2 History-deterministic Reversal-bounded Counter Machines

There is one more automaton model that is closely related to Parikh automata, i.e., reversal-bounded counter machines, originally introduced by Ibarra [26]. These are, in their most general form, two-way automata with multiple counters that can be incremented, decremented, and tested for zero, but there is a constant bound on the number of reversals of the reading head *and* on the number of switches between increments and decrements (on each counter). It is known that Parikh automata and nondeterministic reversal-bounded counter machines are equivalent [27], while deterministic reversal-bounded counter machines are strictly more expressive than deterministic Parikh automata [5]. Here, we compare history-deterministic reversal-bounded counter machines and history-deterministic Parikh automata (and, for technical reasons, also history-deterministic Parikh automata with ε-transitions).

We begin by introducing counter machines and then their reversal-bounded variant.

A (two-way) counter machine is a tuple $\mathcal{M} = (k, Q, \Sigma, \rhd, \lhd, q_I, \Delta, F)$ where $k \in \mathbb{N}$ is the number of counters, Q is the finite set of states, Σ is the alphabet, $\rhd, \lhd \notin \Sigma$ are the left and right endmarkers respectively, $q_I \in Q$ is the initial state,

$$\Delta \subseteq (Q \times \Sigma_{\rhd\lhd} \times \{0,1\}^k) \times (Q \times \{-1,0,1\} \times \{-1,0,1\}^k)$$

is the transition relation, and $F \subseteq Q$ is the set of accepting states. Here, we use the shorthand $\Sigma_{\rhd\lhd} = \Sigma \cup \{\rhd, \lhd\}$. Intuitively, a transition $((q, a, \vec{g}), (q', m, \vec{v}))$ is enabled if the current state is q, the current letter on the tape is a, and for each $0 \leq j \leq k-1$, the j-th entry in the guard $\vec{g}$ is nonzero if and only if the current value of counter j is nonzero. Taking this transition updates the state to q', moves the head in direction m, and adds the j-th entry of $\vec{v}$ to counter j.

We require that all transitions $((q, a, \vec{g}), (q', m, \vec{v})) \in \Delta$ satisfy the following properties:

- If $a = \rhd$, then $m \geq 0$: the head never leaves the tape to the left.
- If $a = \lhd$, then $m \leq 0$: the head never leaves the tape to the right.
- $\vec{g}$ and $\vec{v}$ are *compatible*, i.e. if the j-th entry of $\vec{g}$ is zero, then the j-th entry of $\vec{v}$ is nonnegative: a zero counter is not decremented.

For the sake of brevity, we refer the formal definition of the semantics to the full version [13].

In the following, we just need the definition of configurations: A configuration of $\mathcal{M}$ on an input $w \in \Sigma^*$ is of the form $(q, \rhd w \lhd, h, \vec{c})$ where $q \in Q$ is the current state, $\rhd w \lhd$ is the content of the tape (which does not change during a run), $0 \leq h \leq |w| + 1$ is the current position of the reading head, and $\vec{c} \in \mathbb{N}^k$ is the vector of current counter values.

We say that a two-way counter machine is reversal-bounded, if there is a $b \in \mathbb{N}$ such that on each run, the reading head reverses its direction at most b times *and* each counter switches between incrementing and decrementing at most b times. We write RBCM for reversal-bounded counter machines and 1-RBCM for RBCM that do not make a reversal of the reading head (i.e., they are one-way). Their deterministic variants are denoted by DRBCM and 1-DRBCM, respectively.

Ibarra [26] has shown that every RBCM can be effectively turned into an equivalent 1-RBCM and that every RBCM can be effectively turned into an equivalent one where the number of reversals of each counter is bounded by 1. The latter construction preserves determinism and one-wayness. Hence, in the following, we assume that during each run of an RBCM, each counter reverses at most once.

In terms of expressiveness, Klaedtke and Rueß [27] showed that RBCM are equivalent to Parikh automata while Cadilhac et al. [5] showed that 1-DRBCM are strictly more expressive than DPA.

In the following, we determine the relation between history-deterministic RBCM and HDPA. To this end, we first have to define the notion of history-determinism for RBCM, which is slightly technical due to the two-wayness of these machines.

Let $\mathcal{M} = (k, Q, \Sigma, \rhd, \lhd, q_I, \Delta, F)$ be an RBCM. Given a sequence $\tau_0 \cdots \tau_j$ of transitions inducing a run ρ, let $\mathrm{pos}(\tau_0 \cdots \tau_j)$ be the position of the reading head at the end of ρ, so in particular $\mathrm{pos}(\varepsilon) = 0$. Hence, $(\rhd w \lhd)_{\mathrm{pos}(\tau_0\tau_1\cdots\tau_j)}$ is the letter the reading head is currently pointing to.

A resolver for $\mathcal{M}$ is a function $r\colon \Delta^* \times \Sigma_{\rhd\lhd} \to \Delta$ such that if w is accepted by $\mathcal{M}$, there is a sequence of transitions $\tau_0\tau_1 \cdots \tau_{n-1}$ such that

- $\tau_{j+1} = r(\tau_0\tau_1 \cdots \tau_j, (\rhd w \lhd)_{\mathrm{pos}(\tau_0\tau_1\cdots\tau_j)})$ for all $0 \leq j < n-1$, and
- the run of $\mathcal{M}$ on w induced by the sequence of transitions $\tau_0\tau_1 \cdots \tau_{n-1}$ is accepting.

An RBCM $\mathcal{M}$ is history-deterministic (an HDRBCM) if there exists a resolver for $\mathcal{M}$. One-way HDRBCM are denoted by 1-HDRBCM.

Now, we are able to state the main theorem of this subsection: History-deterministic two-way RBCM are as expressive as RBCM and PA while history-deterministic one-way RBCM are as expressive as history-deterministic PA.

▶ **Theorem 10.**

1. *HDRBCM are as expressive as RBCM, and therefore as expressive as PA.*
2. *1-HDRBCM are as expressive as HDPA.*

The proof of the first equivalence is very general and not restricted to RBCM: A two-way automaton over finite inputs can first read the whole input and then resolve nondeterministic choices based on the whole word. Spelt out more concisely: two-wayness makes history-determinism as powerful as general nondeterminism.

For the other equivalence, both directions are nontrivial: We show how to simulate a PA using an RBCM while preserving history-determinism, and how to simulate a 1-RBCM by a PA, again while preserving history-determinism. Due to the existence of transitions that do not move the reading head in a 1-RBCM, this simulation takes a detour via PA with ε-transitions.

Finally, let us remark that HDPA (or equivalently 1-HDRBCM) and deterministic RBCM have incomparable expressiveness. Indeed, the language E, which is not accepted by any HDPA (see Theorem 7), can easily be accepted by a deterministic RBCM while the language N (see Example 4) is accepted by an HDPA, but not by any deterministic RBCM [6]. The reason is that these machines are closed under complement, but the complement of N is not accepted by any PA as shown in [6], and therefore also not by any RBCM.

5 Closure Properties

In this subsection, we study the closure properties of history-deterministic Parikh automata, i.e., we consider Boolean operations, concatenation and Kleene star, (inverse) homomorphic image, and commutative closure. Let us begin by recalling the last three notions.

Fix some ordered alphabet $\Sigma = (a_0 < a_1 < \cdots < a_{d-1})$. The Parikh image of a word $w \in \Sigma^*$ is the vector $\Phi(w) = (|w|_{a_0}, |w|_{a_1}, \ldots, |w|_{a_{d-1}})$ and the Parikh image of a language $L \subseteq \Sigma^*$ is $\Phi(L) = \{\Phi(w) \mid w \in L\}$. The commutative closure of L is $\{w \in \Sigma^* \mid \Phi(w) \in \Phi(L)\}$.

Now, fix some alphabets Σ and Γ and a homomorphism $h\colon \Sigma^* \to \Gamma^*$. The homomorphic image of a language $L \subseteq \Sigma^*$ is $h(L) = \{h(w) \mid w \in L\} \subseteq \Gamma^*$. Similarly, the inverse homomorphic image of a language $L \subseteq \Gamma^*$ is $h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$.

Table 1 Closure properties of history-deterministic Parikh automata (in grey) and comparison to other types of Parikh automata (results for other types are from [5, 6, 27]).

	$\cup$	$\cap$	$\overline{\ }$	$\cdot$	*	h	h^{-1}	c
DPA	Y	Y	Y	N	N	N	Y	Y
HDPA	Y	Y	N	N	N	N	Y	Y
UCA	Y	Y	Y	N	N	N	?	Y
PA	Y	Y	N	Y	N	Y	Y	Y

▶ **Theorem 11.** *HDPA are closed under union, intersection, inverse homomorphic images, and commutative closure, but not under complement, concatenation, Kleene star, and homomorphic image.*

Proof Sketch. Closure under union and intersection is shown by a product construction, while closure under inverse homomorphic images is shown by lifting the construction for finite automata (just as for DPA and PA). Finally closure under commutative closure follows from previous work: Cadilhac et al. proved that the commutative closure of any PA (and therefore that of any HDPA) is accepted by some DPA, and therefore also by some HDPA.

The negative results follow from a combination of expressiveness results proven in Section 4 and nonexpressiveness results in the literature [5, 6]:

- Complement: In the first part of the proof of Theorem 7, we show that the language N is accepted by an HDPA, but its complement is known to not be accepted by any PA [6].
- Concatenation: The language E is the concatenation of the languages $\{a, b\}^*$ and $\{a^n b^n \mid n \in \mathbb{N}\}$, which are both accepted by a DPA, but itself is not accepted by any HDPA (see the proof of Theorem 7).
- Kleene star: There is a DPA (and therefore also an HDPA) such that the Kleene star of its language is not accepted by any PA [5], and therefore also by no HDPA.
- Homomorphic image: There is a DPA (and therefore also an HDPA) and a homomorphism such that the homomorphic image of the DPA's language is not accepted by any PA [5], and therefore also by no HDPA. ◀

Table 1 compares the closure properties of HDPA with those of DPA, UCA, and PA. We do not compare to RBCM, as only deterministic ones differ from Parikh automata and results on these are incomplete: However, Ibarra proved closure under union, intersection, and complement [26].

6 Decision Problems

Next, we study various decision problems for history-deterministic PA. First, let us mention that nonemptiness and finiteness are decidable for HDPA, as these problems are decidable for PA [27, 5]. In the following, we consider universality, inclusion, equivalence, regularity, and model checking. We start with the universality problem.

▶ **Theorem 12.** *The following problem is undecidable: Given an HDPA $(\mathcal{A}, C)$ over Σ, is $L(\mathcal{A}, C) = \Sigma^*$?*

Proof Sketch. The result is shown by turning (deterministic) Minsky machines into HDPA such that the machine does not terminate if and only if the automaton is universal. As nontermination of Minsky machines is undecidable [30], the same is true for HDPA universality.

Intuitively, the automaton processes sequences of instructions of the Minsky machine and checks whether they are prefixes of the unique run of the machine or not, employing the counters of the automaton to simulate the counter of the Minsky machine. Finally, history-determinism can be employed to find a position in the sequence of instructions where it differs from the unique run of the machine. ◀

The next results follow more or less immediately from the undecidability of universality.

▶ **Theorem 13.** *The following problems are undecidable:*
1. *Given two HDPA* $(\mathcal{A}_0, C_0)$ *and* $(\mathcal{A}_1, C_1)$*, is* $L(\mathcal{A}_0, C_0) \subseteq L(\mathcal{A}_1, C_1)$*?*
2. *Given two HDPA* $(\mathcal{A}_0, C_0)$ *and* $(\mathcal{A}_1, C_1)$*, is* $L(\mathcal{A}_0, C_0) = L(\mathcal{A}_1, C_1)$*?*
3. *Given an HDPA* $(\mathcal{A}, C)$*, is* $L(\mathcal{A}, C)$ *regular?*
4. *Given an HDPA* $(\mathcal{A}, C)$*, is* $L(\mathcal{A}, C)$ *context-free?*

Proof Sketch. The first two items follow directly from the undecidability of universality (cf. Theorem 12), so let us consider the latter two. They are both shown by a variation of the construction proving Theorem 12: Given a Minsky machine we construct an HDPA such that the machine terminates if and only if the automaton accepts a regular language (respectively, a context-free language). ◀

Table 2 compares the decidability of standard problems for HDPA with those of DPA, UCA, and PA.

Finally, we consider the problems of deciding whether a Parikh automaton is history-deterministic and whether it is equivalent to some HDPA. Both of our proofs follow arguments developed for similar results for history-deterministic pushdown automata [29].

▶ **Theorem 14.** *The following problems are undecidable:*
1. *Given a PA* $(\mathcal{A}, C)$*, is it history-deterministic?*
2. *Given a PA* $(\mathcal{A}, C)$*, is it equivalent to some HDPA?*

Finally, let us introduce the model-checking problem (for safety properties): A transition system $\mathcal{T} = (V, v_I, E, \lambda)$ consists of a finite set V of vertices containing the initial state $v_I \in V$, a transition relation $E \subseteq V \times V$, and a labeling function $\lambda \colon V \to \Sigma$ for some alphabet Σ. A (finite and initial) path in $\mathcal{T}$ is a sequence $v_0 v_1 \cdots v_n \in V^+$ such that $v_0 = v_I$ and $(v_i, v_{i+1}) \in E$ for all $0 \leq i < n$. Infinite (initial) paths are defined analogously. The trace of a path $v_0 v_1 \cdots v_n$ is $\lambda(v_0)\lambda(v_1)\cdots\lambda(v_n) \in \Sigma^+$. We denote the set of traces of paths of $\mathcal{T}$ by $\mathsf{tr}(\mathcal{T})$.

The model-checking problem for HDPA asks, given an HDPA $\mathcal{A}$ and a transition system $\mathcal{T}$, whether $\mathsf{tr}(\mathcal{T}) \cap L(\mathcal{A}) = \emptyset$? Note that the automaton specifies the set of *bad prefixes*, i.e., $\mathcal{T}$ satisfies the specification encoded by $\mathcal{A}$ if no trace of $\mathcal{T}$ is in $L(\mathcal{A})$.

Table 2 Decision problems for history-deterministic Parikh automata (in grey) and comparison to other types of Parikh automata (results are from [5, 6, 27]).

	$\neq \emptyset$?	finite?	$= \Sigma^*$?	$\subseteq$?	$=$?	regular?	MC
DPA	Y	Y	Y	Y	Y	Y	Y
HDPA	Y	Y	N	N	N	N	Y
UCA	Y	Y	Y	Y	Y	Y	Y
PA	Y	Y	N	N	N	N	Y

As the model-checking problem for PA is decidable, so is the model-checking problem for HDPA, which follows from the fact that a transition system $\mathcal{T}$ can be turned into an NFA and hence into a PA $\mathcal{A}_\mathcal{T}$ with $L(\mathcal{A}_\mathcal{T}) = \mathsf{tr}(\mathcal{T})$. Then, closure under intersection and decidability of nonemptiness yields the desired result.

▶ **Theorem 15.** *The model-checking problem for HDPA is decidable.*

Let us conclude by mentioning that the dual problem, i.e., given a transition system $\mathcal{T}$ and an HDPA $\mathcal{A}$, does every infinite path of $\mathcal{T}$ have a prefix whose trace is in $L(\mathcal{A})$, is undecidable. This follows from recent results on Parikh automata over infinite words [21], i.e., that model checking for Parikh automata with reachability conditions is undecidable. Such automata are syntactically equal to Parikh automata over finite words and an (infinite) run is accepting if it has a prefix ending in an accepting state whose extended Parikh image is in the semilinear set of the automaton.

7 Conclusion

In this work, we have introduced and studied history-deterministic Parikh automata. We have shown that their expressiveness is strictly between that of deterministic and nondeterministic PA, incomparable to that of unambiguous PA, but equivalent to history-deterministic 1-RBCM. Furthermore, we showed that they have almost the same closure properties as DPA (complementation being the notable difference), and enjoy some of the desirable algorithmic properties of DPA.

An interesting direction for further research concerns the complexity of resolving nondeterminism in history-deterministic Parikh automata. It is straightforward to show that every HDPA has a positional resolver (i.e., one whose decision is only based on the last state of the run constructed thus far and on the extended Parikh image induced by this run) and that HDPA that have finite-state resolvers (say, implemented by a Mealy machine) can be determinized by taking the product of the HDPA and the Mealy machine. In fact, both proofs are simple adaptions of the corresponding ones for history-deterministic pushdown automata [22, 29]. A more interesting question is whether there is a notion of Parikh transducer such that every HDPA has a resolver implemented by such a transducer. Note that the analogous result for history-deterministic pushdown automata fails: not every history-deterministic pushdown automaton has a pushdown resolver [22, 29].

Good-for-gameness is another notion of restricted nondeterminism that is very tightly related to history-determinism. In fact, both terms were used interchangeably until very recently, when it was shown that they do not always coincide [2]. Formally, an automaton $\mathcal{A}$ is good-for-games if every two-player zero-sum game with winning condition $L(\mathcal{A})$ has the same winner as the game where the player who wins if the outcome is in $L(\mathcal{A})$ additionally has to construct a witnessing run of $\mathcal{A}$ during the play. This definition comes in two forms, depending on whether one considers only finitely branching (weak compositionality) or all games (compositionality).

Recently, the difference between being history-deterministic and both types of compositionality has been studied in detail for pushdown automata [20]. These results are very general and can easily be transferred to PA and 1-RBCM. They show that for PA, being history-deterministic, compositionality, and weak compositionality all coincide, while for 1-RBCM, being history-deterministic and compositionality coincide, but not weak compositionality.

The reason for this difference can be traced back to the fact that 1-RBCM may contain transitions that do not move the reading head (which are essentially ε-transitions), but that have side-effects beyond state changes, i.e., the counters are updated. This means that an

unbounded number of configurations can be reached by processing a single letter, which implies that the game composed of an arena and a 1-RBCM may have infinite branching. So, while HDPA and 1-HDRBCM are expressively equivalent, they, perhaps surprisingly, behave differently when it comes to compositionality. We plan to investigate these differences in future work.

References

1 Marc Bagnol and Denis Kuperberg. Büchi good-for-games automata are efficiently recognizable. In Sumit Ganguly and Paritosh Pandya, editors, *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.

2 Udi Boker and Karoliina Lehtinen. History determinism vs. good for gameness in quantitative automata. In Mikołaj Bojańczy and Chandra Chekuri, editors, *FSTTCS 2021*, volume 213 of *LIPIcs*, pages 38:1–38:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSTTCS.2021.38.

3 Udi Boker and Karoliina Lehtinen. Token games and history-deterministic quantitative automata. In Patricia Bouyer and Lutz Schröder, editors, *FOSSACS 2022*, volume 13242 of *LNCS*, pages 120–139. Springer, 2022. doi:10.1007/978-3-030-99253-8_7.

4 Alin Bostan, Arnaud Carayol, Florent Koechlin, and Cyril Nicaud. Weakly-unambiguous Parikh automata and their link to holonomic series. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *ICALP 2020*, volume 168 of *LIPIcs*, pages 114:1–114:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.114.

5 Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. On the expressiveness of Parikh automata and related models. In Rudolf Freund, Markus Holzer, Carlo Mereghetti, Friedrich Otto, and Beatrice Palano, editors, *NCMA 2011*, volume 282 of *books@ocg.at*, pages 103–119. Austrian Computer Society, 2011.

6 Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. Unambiguous constrained automata. *Int. J. Found. Comput. Sci.*, 24(7):1099–1116, 2013. doi:10.1142/S0129054113400339.

7 Giusi Castiglione and Paolo Massazza. On a class of languages with holonomic generating functions. *Theor. Comput. Sci.*, 658:74–84, 2017. doi:10.1016/j.tcs.2016.07.022.

8 Lorenzo Clemente, Wojciech Czerwinski, Slawomir Lasota, and Charles Paperman. Regular Separability of Parikh Automata. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *ICALP 2017*, volume 80 of *LIPIcs*, pages 117:1–117:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.117.

9 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, *ICALP 2009, (Part II)*, volume 5556 of *LNCS*, pages 139–150. Springer, 2009. doi:10.1007/978-3-642-02930-1_12.

10 Luc Dartois, Emmanuel Filiot, and Jean-Marc Talbot. Two-way Parikh automata with a visibly pushdown stack. In Mikolaj Bojanczyk and Alex Simpson, editors, *FOSSACS 2019*, volume 11425 of *LNCS*, pages 189–206. Springer, 2019. doi:10.1007/978-3-030-17127-8_11.

11 Jürgen Dassow and Victor Mitrana. Finite automata over free groups. *Int. J. Algebra Comput.*, 10(6):725–738, 2000. doi:10.1142/S0218196700000315.

12 François Denis, Aurélien Lemay, and Alain Terlutte. Residual finite state automata. In Afonso Ferreira and Horst Reichel, editors, *STACS 2001*, volume 2010 of *LNCS*, pages 144–157. Springer, 2001.

13 Enzo Erlich, Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. History-deterministic parikh automata. *arXiv*, 2209.07745, 2022. arXiv:2209.07745.

14 Diego Figueira and Leonid Libkin. Path logics for querying graphs: Combining expressiveness and efficiency. In *LICS 2015*, pages 329–340. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.39.

15 Emmanuel Filiot, Shibashis Guha, and Nicolas Mazzocchi. Two-way Parikh automata. In Arkadev Chattopadhyay and Paul Gastin, editors, *FSTTCS 2019*, volume 150 of *LIPIcs*, pages 40:1–40:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

16 Emmanuel Filiot, Nicolas Mazzocchi, and Jean-François Raskin. A pattern logic for automata with outputs. *Int. J. Found. Comput. Sci.*, 31(6):711–748, 2020.

17 Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966. doi:pjm/1102994974.

18 Mario Grobler, Leif Sabellek, and Sebastian Siebertz. Parikh automata on infinite words. *arXiv*, 2301.08969, 2023. doi:10.48550/arXiv.2301.08969.

19 Mario Grobler and Sebastian Siebertz. Büchi-like characterizations for parikh-recognizable omega-languages. *arxiv*, 2302.04087, 2023. doi:10.48550/arXiv.2302.04087.

20 Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. A bit of nondeterminism makes pushdown automata expressive and succinct. *arXiv*, 2105.02611, 2021. doi:10.48550/arXiv.2105.02611.

21 Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. Parikh automata over infinite words. *arXiv*, 2207.07694, 2022. doi:10.48550/arXiv.2207.07694.

22 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, *MFCS 2021*, volume 202 of *LIPIcs*, pages 53:1–53:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.MFCS.2021.53.

23 Thomas A. Henzinger, Karoliina Lehtinen, and Patrick Totzke. History-deterministic timed automata. In *CONCUR 2022*, 2022. To appear.

24 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In Zoltán Ésik, editor, *CSL 2006*, volume 4207 of *LNCS*, pages 395–410. Springer, 2006. doi:10.1007/11874683_26.

25 John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

26 Oscar H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *J. ACM*, 25(1):116–133, 1978. doi:10.1145/322047.322058.

27 Felix Klaedtke and Harald Rueß. Monadic second-order logics with cardinalities. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *ICALP 2003*, volume 2719 of *LNCS*, pages 681–696. Springer, 2003. doi:10.1007/3-540-45061-0_54.

28 Denis Kuperberg and Michal Skrzypczak. On determinisation of good-for-games automata. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP 2015 (Part II)*, volume 9135 of *LNCS*, pages 299–310. Springer, 2015. doi:10.1007/978-3-662-47666-6_24.

29 Karoliina Lehtinen and Martin Zimmermann. Good-for-games ω-pushdown automata. *LMCS*, 18(1), 2022. doi:10.46298/lmcs-18(1:3)2022.

30 Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

31 Victor Mitrana and Ralf Stiebe. Extended finite automata over groups. *Discret. Appl. Math.*, 108(3):287–300, 2001. doi:10.1016/S0166-218X(00)00200-6.

32 Karianto Wong. Parikh automata with pushdown stack, 2004. Diploma thesis, RWTH Aachen University. URL: https://old.automata.rwth-aachen.de/download/papers/karianto/ka04.pdf.