Using the Past for Resolving the Future

Orna Kupferman
School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel
orna@cs.huji.ac.il

December 19, 2022

Abstract

Nondeterminism models an ability to see the future: an automaton with an infinite look-ahead can successfully resolve its nondeterministic choices. An automaton is *history deterministic* (HD) if it can successfully resolve its nondeterministic choices in a way that only depends on the past. Formally, an HD automaton has a strategy that maps each finite word to the transition to be taken after the word is read, and following this strategy results in accepting all the words in the language of the automaton. Beyond being theoretically interesting and intriguing, HD automata can replace deterministic automata in several applications, most notably reactive synthesis, and they attract a lot of interest in the research community. The survey describes the development of HD ω-regular automata, relates history determinism to other types of bounded nondeterminism, studies the determinization of HD automata and their succinctness with respect to deterministic ones, and discusses variants, extensions, and open problems around HD automata.

1 Introduction

Automata are among the most studied computation models in theoretical computer science. Their simple structure has made them a basic formalism for the study of fundamental notions. One such notion is *nondeterminism*, introduced by Rabin and Scott in 1959: *"A nondeterministic automaton has, at each stage of its operation, several choices of possible actions. This versatility enables us to construct very powerful automata using only a small number of internal states"* [49]. Nondeterminism allows a computing machine to examine several possible actions simultaneously, and some fundamental questions around it (most notably, P vs. NP) are still open. In the setting of automata on finite words, nondeterminism enables the definition of exponentially more succinct automata, but it does not add to the expressive power of deterministic automata (called "ordinary" in [49]): *"One might imagine at first sight that these new machines are more general than the ordinary ones, but this is not the case".*

In 1962, Büchi introduced *automata on infinite words* [15]. Acceptance in such automata is determined according to the set of states that are visited infinitely often along the run. In particular, in *Büchi* automata (NBW and DBW, for nondeterministic and deterministic Büchi word automata, respectively), the acceptance condition is a subset α of states, and a run is accepting iff it visits α infinitely often. The transition to infinite words significantly extends the combinatorial richness of automata. In particular, in 1969 Landweber proved that NBWs are strictly more expressive than DBWs [38]. That is, there exists a language of infinite words that is recognizable by an NBW but cannot be recognized by a DBW. Today, the

gap between deterministic and nondeterministic Büchi word automata is well understood: While NBWs can recognize all ω-regular languages, an ω-regular language L of infinite words can be recognized by a DBW iff there exists a regular language R of finite words such that L contains exactly all words that have infinitely many prefixes in R [38].

Using NBWs, Büchi solved the decidability problem for *monadic second order logic with one successor* ($S1S$). Given an $S1S$ formula φ, Büchi constructed an NBW $\mathcal{A}_\varphi$ that accepts exactly all the models of φ, and thus reduced the satisfiability of φ to the nonemptiness of $\mathcal{A}_\varphi$. The computer-science community has become further interested in automata on infinite objects thanks to their applications in reasoning about reactive systems. By translating specifications to automata, questions about *specification*, *verification*, and *synthesis* are reduced to questions about automata [55, 33]. In particular, while the translation from S1S formulas to automata is nonelementary, it is only exponential for specification formalisms like *linear temporal logic* [45]. In some applications, such as verification, algorithms can be based on nondeterministic automata, whereas in other applications, such as synthesis and control, algorithms are based on deterministic automata. There, the advantages of nondeterminism are lost, and the algorithms involve a complicated determinization construction [50] or acrobatics for circumventing determinization [37, 21].

To see the difficulty of using nondeterministic automata in synthesis, let us review the synthesis problem and its automata-based solution [23]. Consider a language L of infinite words over an alphabet $2^{I \cup O}$, where I and O are sets of input and output signals, respectively. The synthesis problem for the specification L is to build a reactive system that receives from its environment assignments to the signals in I (that is, letters in 2^I), responds with assignments to the signals in O (that is, letters in 2^O), and does so in such a way that the generated computation (an infinite word over the alphabet $2^{I \cup O}$) is in L [46]. Algorithms for solving the problem are based on taking a deterministic automaton $\mathcal{D}$ for L and conducting a *two-player game* on top of it. The game is played between a player that models the system and a player that models the environment. The positions of the game are the states of $\mathcal{D}$ and it starts in the initial state. In each turn of the game, the environment first chooses the 2^I component of the next letter, the system responds with the 2^O component, and $\mathcal{D}$ moves to the corresponding successor state. Together, the players generate an infinite word in $(2^{I \cup O})^\omega$ along with the run of $\mathcal{D}$ on it. The system wins if this run is accepting. It can be shown that the system has a winning strategy, namely a strategy to respond so that it wins against every environment, iff the language L can be synthesized. Now, if one replaces $\mathcal{D}$ with a nondeterministic automaton $\mathcal{A}$ for L, the system should also choose in each turn a transition to proceed with. Then, it might be that L is synthesizable and still the system has no winning strategy. Indeed, the transition that the system chooses should work for all possible futures of the game, whereas possibly each nondeterministic choice of $\mathcal{A}$ works for a strict subset of the possible futures.

Let us go back to the 1960s, when the solution of the decidability problem for $S1S$ has lead to increasing efforts to solve also the decidability problem for SnS, namely *monadic second order logic with multiple successors*. While $S1S$ formulas describe linear structures, and thus correspond to infinite words, SnS formulas describe branching structures, and correspond to infinite trees. Accordingly, researchers started to study *automata on infinite trees*, which define languages of infinite trees. In particular, they searched for translations of SnS formulas to nondeterministic Büchi tree automata (NBTs), aiming to reduce satisfiability to their nonemptiness.

In 1969, Rabin solved the decidability problem for SnS [47]. The solution involved an introduction of a new type of acceptance condition for automata on infinite objects, namely the *Rabin* acceptance condition. The condition is more complex than the Büchi acceptance condition and involves two types of constraints

on the set of states that are visited infinitely often in the run. Given an SnS formula φ, Rabin constructed a nondeterministic Rabin tree automaton (NRT) $\mathcal{A}_\varphi$ that accepts exactly all the models of φ, and thus reduced the satisfiability of φ to the nonemptiness of $\mathcal{A}_\varphi$. Then, in 1970, Rabin proved that in fact SnS cannot be translated to NBTs [48]. Specifically, NBTs can recognize only the weak fragment of SnS – one in which the sets we quantify over are finite. Thus, while nondeterministic Büchi and Rabin word automata have the same expressive power, NRTs are strictly more expressive than NBTs.

Proving that NBWs are strictly more expressive than DBWs, Landweber showed that the language $L_1 = (0+1)^* \cdot 1^\omega$ (only finitely many 0's) is in NBW $\setminus$ DBW. The proof is simple and can be stated in a few lines or using a two-state *expressiveness refuter* [36]. Much harder is the proof that NRTs are strictly more expressive than NBTs. In [48], Rabin had to use a complicated construction and a complicated inductive argument. Interestingly, the language of trees that Rabin used in his proof is the *derived language* of L_1. That is, the set of all trees all whose paths have only finitely many 0's.

In 1996, it turned out that Rabin's choice of L_1 was not arbitrary: for every language L of infinite words, let $L\triangle$ denote the derived language of L, namely the language of trees all whose paths are in L. In [34, 35], Kupferman, Safra, and Vardi proved that for every language L of infinite words, we have that $L \in$ NBW$\setminus$DBW iff $L\triangle \in$ NRT$\setminus$NBT. The difficult part in the proof is to show that if $L\triangle$ can be recognized by an NBT, then L can be recognized by a DBW. Intuitively, since the branches of a tree in $L\triangle$ may contain any word in L, the nondeterministic choices that an NBT performs when it recognizes $L\triangle$ have to accommodate all possible futures, which makes the usefulness of nondeterminism questionable. The results in [34, 35] where generalized in [44] to acceptance conditions that are stronger than the Büchi condition. Niwinski and Walukiewicz showed that if $L\triangle$ can be recognized by a nondeterministic tree automaton with some acceptance condition γ (for example, γ may be *parity* with index 5), then L can be recognized by a deterministic word automaton with acceptance condition γ. The difficulty in defining a nondeterministic tree automaton for a derived language are similar to the difficulty of the system player in the synthesis game, when played on a nondeterministic automaton: both have to resolve their nondeterministic choices in a way that only depends on the past and still accommodates all futures.

In [26], Henzinger and Piterman introduced *history deterministic* (HD) automata, which capture this difficulty in a very clean way. Essentially, a nondeterministic automaton is HD if it has a strategy to resolve its nondeterministic choices that only depends on the past. The notion used in [26] is *good for games* (GFG) automata, as they address the difficulty described above, of playing games on top of a nondeterministic automaton. As it turns out, the property of being good for games varies in different settings and HD is good for applications beyond games (see more in Section 5). Therefore, following [14], we use the term *history determinism*, introduced by Colcombet in the setting of quantitative automata with cost functions [19].

Formally, a nondeterministic automaton $\mathcal{A}$ over an alphabet Σ is HD if there is a strategy f that maps each finite word $u \in \Sigma^*$ to the transition to be taken after u is read; and following f results in accepting all the words in the language of $\mathcal{A}$. Note that a state q of $\mathcal{A}$ may be reachable via different words, and f may suggest different transitions from q after different words are read. Still, the choices of f only depend on the past, namely on the word read so far, and have to address all possible futures, namely all possible suffixes. As formalized in [8], the strategy f of an HD automaton for L is similar to a run of a tree automaton for $L\triangle$ on a tree that includes all words in L.[1] As their original "good for games" name suggests, HD

[1] Note that such a tree exists only when L is *fusion closed*; in the general case, the relation between a tree automaton for $L\triangle$ and an HD automaton for L is formalized by a tree that includes all the words in Σ^ω [8].

automata can be used in the reduction of synthesis to game solving. Indeed, if one tries to replace the deterministic automaton $\mathcal{D}$ discussed above by an HD automaton $\mathcal{A}$, the system should still choose in each turn a transition to proceed with, but now it is guaranteed that there is a transition that would work for all possible futures.

Obviously, there exist HD automata: deterministic ones, or nondeterministic ones that are *determinizable by pruning* (DBP); that is, ones that just add transitions on top of a deterministic automaton. In fact, the HD automata constructed in [26] are DBP.[2] Beyond the theoretical interest in DBP automata, they are used for modeling online algorithms: by relating the "unbounded look ahead" of optimal offline algorithms with nondeterminism, and relating the "no look ahead" of online algorithms with determinism, it is possible to reduce questions about the competitive ratio of online algorithms and the memory they require to questions about DBPness [5, 6]. As it turns out, HD automata on infinite words need not be DBP, and they constitute an interesting and intriguing class of automata, many of whose properties are still unknown.

This survey presents selected results about HD automata. We focus in Büchi auotmata, and their dual *co-Büchi* automata, denoted DCW and NCW, for the deterministic and nondeterministic classes. Section 3 studies determinization by pruning. It shows that HD nondeterministic automaton on finite words are always DBP. Moreover, a deterministic equivalent automaton that is embodied in every HD automaton can be found in polynomial time. On the other hand, once we move to automata on infinite words, HD-NBWs and HD-NCWs need not be DBP, and deciding their DBPness is NP-complete. Section 4 studies determinization of HD-NBWs and HD-NCWs. It shows that their determinization is simpler than that of NBWs and NCWs, and that HD affects Büchi and co-Büchi automata in a different and surprising way: Recall that nondeterminism is more significant for Büchi than for co-Büchi automata: NBWs are strictly more expressive than DBWs [38], and determinization of NBWs is very complicated and involves, beyond using a richer acceptance condition, also a $2^{O(n \log n)}$ blow up [50]. NCWs, on the other hand, are as expressive as DCWs, and their determinization only involves a $2^{O(n)}$ blow up [40]. One could then expect that HD nondeterminism would also be more significant for Büchi than co-Büchi automata. As we show in Section 4, this is not the case: while HD-NCWs are exponentially more succinct than DCWs, every HD-NBWs can be determinized to a DBW with a quadratic blow-up, and in fact no matching lower bound is known. The section also relates determinization and complementation of HD automata. Finally, Section 5 discusses variants, extensions, and open problems.

2 Preliminaries

For a finite nonempty alphabet Σ, an infinite *word* $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of letters from Σ. A *language* $L \subseteq \Sigma^\omega$ is a set of infinite words. For $i, j \geq 0$, we use $w[1, i]$ to denote the (possibly empty) *prefix* $\sigma_1 \cdot \sigma_2 \cdots \sigma_i$ of w, use $w[i+1, j]$ to denote the (possibly empty) *infix* $\sigma_{i+1} \cdot \sigma_{i+2} \cdots \sigma_j$ of w, and use $w[i+1, \infty]$ to denote its *suffix* $\sigma_{i+1} \cdot \sigma_{i+2} \cdots$. We sometimes refer also to languages of finite words, namely subsets of Σ^*. We denote the empty word by ϵ.

A *nondeterministic automaton* over infinite words is $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where Σ is an alphabet, Q is a finite set of *states*, $q_0 \in Q$ is an *initial state*, $\delta : Q \times \Sigma \to 2^Q \setminus \emptyset$ is a *transition function*, and α is an *acceptance condition*, to be defined below. For states q and s and a letter $\sigma \in \Sigma$, we say that s is a

[2] As explained in [26], the fact that the HD automata constructed there are DBP does not contradict their usefulness in practice, as their transition relation is simpler than the one of the embodied deterministic automaton and it can be defined symbolically.

σ-successor of q if $s \in \delta(q, \sigma)$. Note that $\mathcal{A}$ is *total*, in the sense that it has at least one successor for each state and letter. If $|\delta(q, \sigma)| = 1$ for every state $q \in Q$ and letter $\sigma \in \Sigma$, then $\mathcal{A}$ is *deterministic*.

When $\mathcal{A}$ runs on an input word, it starts in the initial state and proceeds according to the transition function. Formally, a *run* of $\mathcal{A}$ on $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of states $r = r_0, r_1, r_2, \ldots \in Q^\omega$, such that $r_0 = q_0$, and for all $i \geq 0$, we have that $r_{i+1} \in \delta(r_i, \sigma_{i+1})$. We extend δ to sets of states and finite words in the expected way. Thus, $\delta(S, u)$ is the set of states that $\mathcal{A}$ may reach when it reads the word $u \in \Sigma^*$ from some state in $S \in 2^Q$. Formally, $\delta : 2^Q \times \Sigma^* \to 2^Q$ is such that for every $S \in 2^Q$, finite word $u \in \Sigma^*$, and letter $\sigma \in \Sigma$, we have that $\delta(S, \epsilon) = S$, $\delta(S, \sigma) = \bigcup_{s \in S} \delta(s, \sigma)$, and $\delta(S, u \cdot \sigma) = \delta(\delta(S, u), \sigma)$. The transition function δ induces a transition relation $\Delta \subseteq Q \times \Sigma \times Q$, where for every two states $q, s \in Q$ and letter $\sigma \in \Sigma$, we have that $\langle q, \sigma, s \rangle \in \Delta$ iff $s \in \delta(q, \sigma)$. For a state $q \in Q$ of $\mathcal{A}$, we define $\mathcal{A}^q$ to be the automaton obtained from $\mathcal{A}$ by setting the initial state to be q. Thus, $\mathcal{A}^q = \langle \Sigma, Q, q, \delta, \alpha \rangle$.

The acceptance condition α determines which runs are "good". We consider here the *Büchi* and *co-Büchi* acceptance conditions, in both a *state-based* and a *transition-based* setting. In the traditional state-based setting, we have that $\alpha \subseteq Q$ is a subset of states. For a run r, let $\mathit{inf}(r) \subseteq Q$ be the set of states that r visits infinitely often. Thus, $\mathit{inf}(r) = \{q \in Q : q = r_i \text{ for infinitely many } i\text{'s}\}$. A run r of a Büchi automaton is *accepting* iff it visits states in α infinitely often, thus $\mathit{inf}(r) \cap \alpha \neq \emptyset$. Dually, a run r of a co-Büchi automaton is accepting iff it visits states in α only finitely often, thus $\mathit{inf}(r) \cap \alpha = \emptyset$. In the transition-based setting, we have that $\alpha \subseteq \Delta$ is a set of transitions, $\mathit{inf}(r)$ is defined as the set of transitions that are traversed infinitely often in r, and the definition of acceptance is similar. Thus, in Büchi automata, a run is accepting if it traverses infinitely many transitions in α, and in co-Büchi automata, a run is accepting if it traverses only finitely many transitions in α. In both the state-based and transition-based settings, a run that is not accepting is *rejecting*. As $\mathcal{A}$ is nondeterministic, it may have several runs on a word w. The word w is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on w. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. Two automata are *equivalent* if their languages are equivalent.

Consider a nondeterministic automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, We say that $\mathcal{A}$ is *semantically deterministic*, if its nondeterministic choices lead to states with the same language. Formally, for every state $q \in Q$ and letter $\sigma \in \Sigma$, all the σ-successors of q have the same language.

Then, we say that $\mathcal{A}$ is *history deterministic* (HD, for short) if there is a strategy $f : \Sigma^* \to Q$ that resolves the nondeterminism in $\mathcal{A}$ in a way that only depends on the past and leads to the acceptance of all words in $L(\mathcal{A})$. Formally, the following hold:

- The strategy f is compatible with δ. That is, for all $u \in \Sigma^*$ and $\sigma \in \Sigma$, we have that $f(u \cdot \sigma) \in \delta(f(u), \sigma)$.
- Following f guarantees the acceptance of all the words in $L(\mathcal{A})$. That is, for all words $\sigma_1 \cdot \sigma_2 \cdot \sigma_3 \cdots \in L(\mathcal{A})$, the sequence $f(\epsilon)$, $f(\sigma_1)$, $f(\sigma_1 \cdot \sigma_2)$, $f(\sigma_1 \cdot \sigma_2 \cdot \sigma_3), \ldots$ satisfies the acceptance condition α.

Finally, $\mathcal{A}$ is *determinizable by pruning* (DBP, for short) if it *embodies* an equivalent deterministic automaton, thus it can be determinized to an equivalent automaton by removing some of its transitions.

It is easy to see that every DBP automaton is HD. Indeed, a witness strategy f can follow the unpruned transitions. Also, every HD automaton can be pruned in polynomial time to a semantically deterministic automaton [32]. Indeed, the fact the automaton is HD implies that for every state $q \in Q$, we can prune transitions to σ-successors of q whose language does not contain the language of another σ-successor of q. Indeed, these transitions are never taken by an HD strategy. Since language-containment for HD automata can be checked in polynomial time, such a pruning can be done in polynomial time.

We denote the different classes of automata by three-letter acronyms in $\{\text{D, N}\} \times \{\text{F, B, C}\} \times \{\text{W}\}$. The first letter stands for the branching mode of the automaton (deterministic or nondeterministic); the second for the acceptance condition type (finite, Büchi, or co-Büchi); and the third indicates that we consider automata on words. For example, NBWs are nondeterministic Büchi word automata. When the acceptance condition is transition-based, we add "t" before the acronym, and when the automata are HD, we indicate it too. For example, HD-tNCWs are HD NCWs with a transition-based acceptance condition.

3 Determinization by Pruning

The fact nondeterminism leads to exponential succinctness implies that not all nondeterministic automata are DBP. In this section we study DBPness for HD automata. Recall that the strategy f that witnesses the HDness of an automaton $\mathcal{A}$ directs runs of $\mathcal{A}$ how to resolve nondeterministic choices based on the prefix of the word read so far, and may proceed with different nondeterministic choices in different visits to the same state. We can view the question of DBPness in HD automata as the question of whether the HD strategy really needs to make these different choices, namely whether the choices depend on the past. Indeed, an HD automaton is DBP if the past does not really play a role in the resolving of nondeterminism and the same choice can be taken whenever nondeterminism has to be resolved. As we shall see, the answer is positive for automata on finite words and negative for Büchi and co-Büchi automata.

3.1 The case of finite words

In this section we prove that for automata on finite words, all HD-NFWs are DBP. For this, we describe a sufficient condition for NFWs to be DBP, and argue that all HD-NFWs satisfy the condition. The condition is a simplification of a fixed-point characterization of NFWs that are DBPs introduced in [5], where it is used in order to decide whether a given NFW is DBP.

Consider an NFW $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$. For a relation $H \subseteq Q \times Q$, a set $S \subseteq Q$, and a states $q' \in Q$, we write $H(S, q')$ to indicate that $H(q, q')$ for all $q \in S$. We inductively define a sequence $H_0, H_1, \ldots \subseteq Q \times Q$ of relations as follows.

- $H_0 = (\alpha \times \alpha) \cup ((Q \setminus \alpha) \times Q)$. That is, $H_0 = \{\{\langle q, q' \rangle : \text{if } q \in \alpha, \text{ then } q' \in \alpha\}$.
- For $i \geq 0$, we have $H_{i+1} = H_i \cap \{\langle q, q' \rangle : \text{for all } \sigma \in \Sigma \text{ there is } v' \in \delta(q', \sigma) \text{ such that } H_i(\delta(q, \sigma), v')\}$.

Intuitively, $H_i(q, q')$ means that $\mathcal{A}$ can be prunned to a DFW $\mathcal{A}'$ such that all the words of length at most i accepted from q in $\mathcal{A}$ are also accepted from q' in $\mathcal{A}'$.

Since $H_0 \subseteq Q \times Q$ and $H_0 \supseteq H_1 \supseteq H_2 \supseteq \ldots$, the sequence of relations eventually reaches a fixed-point, which we denote by H. Intuitively, $H(q, q')$ if there is a DFW embodied in $\mathcal{A}^{q'}$ that accepts all words in $L(\mathcal{A}^q)$.

The relation H induces an NFW $\mathcal{A}^H = \langle \Sigma, Q, Q_0^H, \delta^H, \alpha \rangle$ embodied in $\mathcal{A}$, where

- $Q_0^H = \{v : v \in Q_0 \text{ and } H(Q_0, v)\}$.
- For all $q \in Q$ and $\sigma \in \Sigma$, we have that $\delta^H(q, \sigma) = \{v : v \in \delta(q, \sigma) \text{ and } H(\delta(q, \sigma), v)\}$.

Note that the set Q_0^H may be empty, and that for some state $q \in Q$ and letter $\sigma \in \Sigma$, it may be that $\delta^H(q, \sigma) = \emptyset$. We prove that nonemptiness of Q_0^H is a sufficient condition for $\mathcal{A}$ to be DBP.

Lemma 3.1. *Consider an NFW $\mathcal{A}$, its relation H, and the induced NFW $\mathcal{A}^H$. If $Q_0^H \neq \emptyset$, then $\mathcal{A}$ is DBP.*

Proof: Assume that Q_0^H is not empty. We prove that every DFW that is embodied in $\mathcal{A}^H$ is equivalent to $\mathcal{A}$. Let $\mathcal{A}' = \langle \Sigma, Q, q_0', \delta', \alpha \rangle$ be such a DFW. Thus, $q_0' \in Q_0^H$ and for all states $q \in Q$ and letters $\sigma \in \Sigma$, we have $\delta'(q,\sigma) \in \delta^H(q,\sigma)$. Note that if $\delta^H(q,\sigma) = \emptyset$, then $\delta'(q,\sigma)$ is not defined. As we shall prove, however, the fact Q_0^H is not empty implies that $\delta^H(q,\sigma) \neq \emptyset$ for all letters $\sigma \in \Sigma$ states $q \in Q$ that are reachable in $\mathcal{A}'$.

We prove that $L(\mathcal{A}') = L(\mathcal{A})$. Since $\mathcal{A}'$ is embodied in $\mathcal{A}^H$, which in turn is embodied in $\mathcal{A}$, it is clear that $L(\mathcal{A}') \subseteq L(\mathcal{A})$. In order to prove that $L(\mathcal{A}) \subseteq L(\mathcal{A}')$, consider a word $w = w_1 w_2 \dots w_n \in L(\mathcal{A})$. We prove that $\mathcal{A}'$ does not get stuck on w and that for every run $r = r_0 r_1 \dots r_n$ of $\mathcal{A}$ on w, the run $s = s_0 s_1 \dots s_n$ of $\mathcal{A}'$ on w is such that for all $0 \leq j \leq n$, we have that $H(r_j, s_j)$. Since $H \subseteq H_0$, the latter implies that membership of r_n in α implies membership of s_n in α. Thus, if there is an accepting run of $\mathcal{A}$ on w, then the run of $\mathcal{A}'$ on w is also accepting.

The proof proceeds by an induction on j. For $j = 0$, the definition of $\mathcal{A}'$ implies that $s_0 = q_0' \in Q_0^H$. Therefore, by the definition of Q_0^H, we have that $H(Q_0, s_0)$. In particular, as $r_0 \in Q_0$, we have that $H(r_0, s_0)$.

For the induction step, assume that the induction hypothesis holds for $j \geq 0$, thus $H(r_j, s_j)$. We prove that the run s does not get stuck in the j-th transition, and that the state s_{j+1} satisfies $H(r_{j+1}, s_{j+1})$. By the induction hypothesis, we have that $H(r_j, s_j)$. Hence, by the definition of H, for every letter $\sigma \in \Sigma$, there exists a state $v \in \delta(s_j, \sigma)$ such that $H(\delta(r_j, \sigma), v)$. Hence, if $\delta(r_j, \sigma)$ is not empty, so is $\delta^H(s_j, \sigma)$. In particular, as $\delta(r_j, w_{j+1})$ includes r_{j+1}, we have that $\delta^H(s_j, w_{j+1}) \neq \emptyset$, and so $\mathcal{A}'$ does not get stuck on w in the j-th transition. In addition, by the definition of δ^H, the fact $s_{j+1} \in \delta^H(s_j, w_{j+1})$ implies that $H(\delta(r_j, w_{j+1}), s_{j+1})$. Since $r_{j+1} \in \delta(r_j, w_{j+1})$, it follows that $H(r_{j+1}, s_{j+1})$, and we are done. □

We continue and prove that all HD-NFWs are DBP. Given an HD-NFW $\mathcal{A}$ as above, and a function $f : \Sigma^* \to Q$ that witnesses its HDness, consider the relation $G_f \subseteq Q \times Q$ where for all $q, q' \in Q$, we have that $G(q, q')$ iff there is a word $w \in \Sigma^*$ such that $q \in \delta(Q_0, w)$ and $f(w) = q'$. Intuitively, $G_f(q, q')$ if there is a word $w \in \Sigma^*$ such that the HD strategy f is guaranteed to accept from q' all suffixes that extend w to a word in $L(\mathcal{A})$ and are accepted from q.

Lemma 3.2. *If $\mathcal{A}$ is an HD-NFW and f witnesses its HDness, then $G_f \subseteq H$ and $f(\epsilon) \in Q_0^H$.*

Proof: Consider an HD-NFW $\mathcal{A}$ and a function $f : \Sigma^* \to Q$ that witnesses its HDness. We first prove that $G_f \subseteq H_i$ for all $i \geq 0$, thus $G_f \subseteq H$. The proof proceeds by an induction on i.

For the induction base, consider two states $q, q' \in Q$ such that $G_f(q, q')$. Let $w \in \Sigma^*$ be such that $q \in \delta(Q_0, w)$ and $f(w) = q'$. By the definition of G_f, such a word w exists. Assume that $q \in \alpha$. Then, $w \in L(\mathcal{A})$, and so, as f witnesses the the HDness of $\mathcal{A}$, we have that $f(w) \in \alpha$ too. Thus, $q \in \alpha$ implies that $q' \in \alpha$, and so $H_0(q, q')$, and we are done.

For the induction step, consider again two states $q, q' \in Q$ such that $G_f(q, q')$, and let $w \in \Sigma^*$ be such that $q \in \delta(Q_0, w)$ and $f(w) = q'$. First, by the induction hypothesis, we have that $H_i(q, q')$. Now, by definition, $H_{i+1}(q, q')$ iff $H_i(q, q')$ and for all letters $\sigma \in \Sigma$ there is $v' \in \delta(q', \sigma)$ such that $H_i(\delta(q, \sigma), v')$. For a letter $\sigma \in \Sigma$, let $v' = f(w \cdot \sigma)$. Note that by the definition of HD witness functions, the state v' is in $\delta(q', \sigma)$. Consider a state $v \in \delta(q, \sigma)$. Note that $v \in \delta(Q_0, w \cdot \sigma)$, and so $G_f(v, v')$. Therefore, by the induction hypothesis, we have $H_i(v, v')$. It follows that $H_i(\delta(q, \sigma), v')$. Since the above holds for all letters $\sigma \in \Sigma$, it follows that $H_{i+1}(q, q')$, and are done.

It is left to prove that $f(\epsilon) \in Q_0^H$. Recall that for every state $q \in Q$, we have that $q \in Q_0^H$ iff $q \in Q_0$ and $H(Q_0, q)$. Clearly, $f(\epsilon) \in Q_0$. Also, by the definition of G_f, as $Q_0 = \delta(Q_0, \epsilon)$, we have that $G_f(Q_0, f(\epsilon))$. Hence, as $G_f \subseteq H$, we have that $H(Q_0, f(\epsilon))$, and we are done. □

Lemmas 3.2 and 3.1 together imply that a function that witnesses the HDness of an HD-NFW $\mathcal{A}$ also witnesses the nonemptiness of Q_0^H, and so we can conclude with the following.

Corollary 3.3. *Every HD-NFW is DBP.*

Remark 3.1. A language $L \subseteq \Sigma^\omega$ is a *safety language* if it states that something "bad" never happens. Formally, for every infinite word $w \in \Sigma^\omega$, if $w \notin L$, then w has a prefix $x \in \Sigma^*$ such that $x \cdot y \notin L$ for all $y \in \Sigma^\omega$. Safety languages play an important role in verification and synthesis, as many natural specifications are safety. It is not hard to prove that safety languages can be recognized by *looping automata*, namely Büchi automata in which all states are in α (or dually, co-Büchi automata in which no state is in α) [54]. It is also not hard to see that the considerations in our proof above apply also to looping automata. Thus, all HD nondeterministic looping automata are DBP. In fact, by [41, 11], the above applies also to *weak automata*, which are a stronger special case of Büchi and co-Büchi, in which every strongly connected component of the graph induced by the automaton is either contained in α or disjoint from α. □

3.2 The case of infinite words

We continue to automata on infinite words and show that here, the past does play a role in resolving nondeterminism. Thus, HD Büchi and co-Büchi automata need not be DBP. The result was first proven, by examples, in [8]. In [31], the authors study DBPness for general NCWs and proved that deciding whether a given NCW is DBP is NP-hard. In [4], the authors noted that the NCW used in the proof is actually HD, thus deciding DPNess is NP-hard already for HD-NCWs, and proved a similar results also for Büchi automata. Clearly, in order for a problem to be NP-hard, the answer has to be non-trivial. Thus, there are HD-NCWs and HD-NBWs that are not DBP, and deciding whether a given HD-NCW or HD-NBW is DBP is NP-hard. Here, we describe a variant of the example in [8] for the Büchi case, and then describe the NP-hardness proof for the co-Büchi case. We consider both state-based and transition-based acceptance.

Theorem 3.4. *[8] There are HD-tNBWs and HD-NBWs that are not DBP.*

Proof: Consider the tNBW $\mathcal{A}$ appearing in Figure 1. We prove that $\mathcal{A}$ is HD and is not DBP.

Note that $\mathcal{A}$ gets stuck (and rejects) when it reads words that are not in $(a0 + a1)^\omega$. We claim that $L(\mathcal{A}) = L$, for

$$L = \{w \in (a0 + a1)^\omega : w \text{ has infinitely many infixes of the form } a0a0 \text{ or } a1a1\}.$$

In order to see that $L(\mathcal{A}) \subseteq L$, note that if a word in $(a0 + a1)^\omega$ is not in L, and thus it has only finitely many infixes of the form $a0a0$ or $a1a1$, then it has a suffix $(a0a1)^\omega$. Also, when a run of $\mathcal{A}$ traverses an α-transition when reading such a suffix, then after taking the α-transition, it keeps looping at the q_3, q_5, q_4, q_6 cycle and never traverses an α-transition again. In order to see that $L \subseteq L(\mathcal{A})$, note that after reading a prefix in $(a0 + a1)^*$, a run of $\mathcal{A}$ is in state s, q_3, or q_4. If the run is in state s or q_4 and reads $a0a0$, then it can traverse an α-transition and return to s, and if it is in state q_3 and reads $a0a0$, then it reaches the state s. Also, reading $a1$ from s, a run can return to s. Thus, reading $a0a0$ infinitely often enables a run to traverse α-transitions infinitely often, and similarly for $a1a1$.

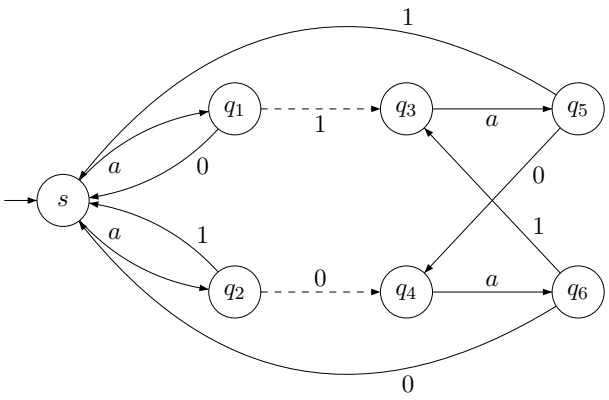

Figure 1: A HD-tNBW that is not DBP. Dashed transitions are in α.

We continue and prove that $\mathcal{A}$ is HD. We do so by describing a strategy f that witnesses its HDness. Note that there is one nondeterministic transition in $\mathcal{A}$: reading a in state s, a run can proceed to q_1 or q_2. We define f so that whenever $\mathcal{A}$ is in state s and reads a, it directs the run to proceed as follows. If the run has just started or s was reached from q_6 or q_2, then the run continues to q_1; if s was reached from q_5 or q_1, then the run continues to q_2. First, note that the above strategy can be described by means of a function with domain $\{a, 0, 1\}^*$. For example, $f(a) = q_1$, $f(a0a) = q_2$, and $f(a1a1a) = q_2$. In addition, the strategy guarantees that all words in the language are accepted. Indeed, reading $a0a0$ either leads to a traversal of an α-transition or leads to s, where the next $a0a0$ or $a1a1$ leads to a traversal of an α-transition, and similarly for $a1a1$.

It is left to prove that $\mathcal{A}$ is not DBP. Recall that there are two ways to make $\mathcal{A}$ deterministic by pruning: either prune the a transition from s to q_1 or the a-transition from s to q_2. We show that both ways result in a tDBW whose language is strictly contained in that of $\mathcal{A}$. First, if we prune the transition from s to q_1, then the obtained tDBW rejects the word $(a1)^\omega$, which is in $L(\mathcal{A})$. Indeed, the singe run on it is $(s, q_2)^\omega$, which is rejecting. Dually, if we prune the transition from s to q_2, then the single run of the obtained tDBW on the word $(a0)^\omega$, which is in $L(\mathcal{A})$, is $(s, q_1)^\omega$, which is rejecting.

Thus, $\mathcal{A}$ is an HD-tNBW that is not DBP. We continue and obtain from $\mathcal{A}$ an HD-NBW $\mathcal{A}'$ that is not DBP. For this, we replace the letters 0 and 1 by the words 0# and 1#, respectively, thus consider the language $L' = \{w \in (a0\# + a1\#)^\omega : w \text{ has infinitely many infixes of the form } a0\#a0\# \text{ or } a1\#a1\#\}$. We obtain $\mathcal{A}'$ from $\mathcal{A}$ by adding an intermediate state inside each 0-transition (and similarly for 1-transitions). The state is reached with 0 and continues with # to the destination of the original transition. The new state is accepting iff the transition that induces it is accepting. By applying the same considerations detailed above for $\mathcal{A}$, it is easy to see that $\mathcal{A}'$ accepts exactly all words in $(a0\# + a1\#)^\omega$ that have infinitely many infixes of the form $a0\#a0\#$ or $a1\#a1\#$, is HD, and is not DBP. □

Theorem 3.5. *[31, 4] Deciding whether a given HD-tNCW or HD-NCW is DBP is NP-complete.*

Proof: Since pruning transitions can only decrease the language of an automaton and checking the containment of the language of a tNCW (or NCW) in the language of a tDCW (or a DCW) can be checked in polynomial time, membership in NP is easy.

For the lower bound, we describe a reduction from the *Hamiltonian-cycle* problem: Given a connected directed graph graph $G = \langle V, E \rangle$, the problem is to decide whether G contains a cycle that visits every vertex

in V exactly once. We start with automata with state-based acceptance. Consider a graph $G = \langle V, E\rangle$. For simplicity, we assume that $V = \{1, 2, \ldots, |V|\}$. Given G, the reduction outputs an NCW $\mathcal{A}_G$ over the alphabet $V \cup \{\#\}$ that is obtained from G as follows (see example in Figure 2). The automaton $\mathcal{A}_G$ accepts only words in $(V \cdot \#)^\omega$. Each vertex $i \in V$ contributes three states to $\mathcal{A}_G$, denoted v_i, s_i, and u_i. From states of the form v_i, the NCW reads only letters in V, and from states of the form s_i and u_i, it reads only the letter $\#$. When in state v_i, the subword $i \cdot \#$ leads back to v_i via s_i, and subwords $j \cdot \#$, for $j \in V \setminus \{i\}$, nondeterministically lead, via u_i, to states v_k, for successors k of i in G. For example, in the graph G and its NCW appearing in Figure 2, there are two $\#$-transitions from state u_1, leading to v_2 and v_4 – or the successors 2 and 4 of the vertex 1 in G. Accordingly, reading $1\#$, a run from v_1 returns to v_1, and reading $2\#, 3\#$, or $4\#$, a run from v_1 can reach v_2 or v_4.

The co-Büchi condition α includes all states of the form u_i, and thus requires a run to eventually get stuck at some $(v_i, s_i)^\omega$ cycle. Accordingly, $L(\mathcal{A}_G) = (V \cdot \{\#\})^* \cdot \bigcup_{i \in V} (i \cdot \#)^\omega$. Indeed, no matter which state of the form v_k is reached after reading some prefix in $(V \cdot \{\#\})^*$, the fact G is connected guarantees that for every $i \in V$, the state v_i can be reached from v_k by reading a prefix of the $(i \cdot \#)^\omega$ suffix after at most $|V|$ visits in α, and then the run can stay forever in the $(v_i, s_i)^\omega$ cycle. Also, if w is accepted by $\mathcal{A}_G$, then the accepting run on it eventually loops in some $(v_i, s_i)^\omega$ cycle, which is possible only if w is in $(V \cdot \#)^\omega$ and has an $(i \cdot \#)^\omega$ suffix.

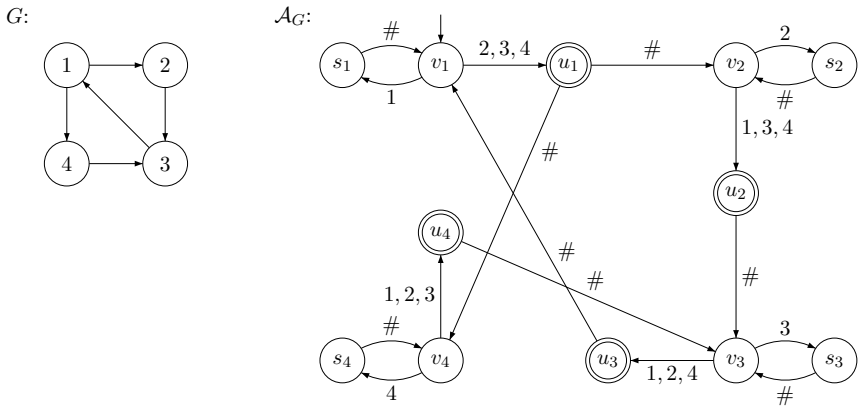

Figure 2: The reduction from the Hamiltonian-cycle problem.

In order to see that $\mathcal{A}_G$ is DBP iff there is a Hamiltonian cycle in G, note that the only nondeterminism in $\mathcal{A}_G$ is in states of the form u_k, where the letter $\#$ forces each deterministic pruning of $\mathcal{A}_G$ to proceed from the state v_k upon reading a subword $(V \setminus \{i\}) \cdot \#$, to the same state v_j, for some successor vertex j of k . DBPing $\mathcal{A}_G$ from Figure 2, for example, leaves only one $\#$-transition from u_1, forcing $2\#, 3\#$, or $4\#$ to all reach v_2 from v_1, or all reach v_4.

Now, if there is a Hamiltonian cycle in G, then $\mathcal{A}_G$ can be pruned to a DCW by leaving, from each state u_k, the $\#$-transition to the successor of vertex k in the cycle. Indeed, as the Hamiltonian cycle visits all vertices of G, reading a suffix $(i \cdot \#)^\omega$ of a word in $L(\mathcal{A}_G)$, the DCW can reach the $(v_i, s_i)^\omega$ cycle and stay there forever. For the other direction, a DCW that is obtained by pruning $\mathcal{A}_G$ and recognizes $L(\mathcal{A}_G)$ must induce a Hamiltonian cycle, as otherwise some vertices are not reachable in the DCW, making its language a strict subset of $L(\mathcal{A}_G)$.

Finally, it is not hard to see that $\mathcal{A}_G$ is HD for every graph G. Indeed, an HD strategy can decide to which successor of v_k to proceed with a subword in $(V \setminus \{k\}) \cdot \#$ by following a (not necessarily Hamiltonian)

cycle that traverses all the vertices of the graph G. Since when we read $(V \setminus \{k\}) \cdot \#$ we move to a state v_j for a successor vertex j of k, then by following the cycle when we read an $(i \cdot \#)^\omega$ suffix of a word in $L(\mathcal{A}_G)$, we eventually reach the state v_i, where we stay in the $(v_i, s_i)^\omega$ cycle, and accept. Thus, the Hamiltonian-cycle problem is reduced to DBPness of an HD-NCW, and we are done.

As for HD-NCWs, the reduction is similar, except that we define $\mathcal{A}_G$ to be an HD-tNCW, for example by defining α as the set of transitions that leave states of the form u_i. □

4 Determinization of HD Automata

Recall that HD automata are as expressive as deterministic ones. For the case of finite words, this follows immediately from the fact HD automata are DBP. For the case of infinite words, where HD automata need not be DBP, the result is more complicated. As discussed in Section 1, the expressive power of HD automata was first studied in the setting of derivable tree languages [34, 44]. Then, [32] studied also the *succinctness* of HD automata with respect to deterministic ones, namely the blow-up involved in determinizing a given HD automaton. In this section we study determinization and succinctness of HD automata. As we shall see, the answers for Büchi and co-Büchi automata are different, and in a surprising way: while nondeterminism is in general more significant for Büchi than for co-Büchi automata, HD nondeterminism is more significant for co-Büchi than for Büchi. Specifically, while HD-NBWs can be determinized with a quadratic blow-up, determinization of HD-NCWs may involve an exponential blow up [32].

4.1 Subset-construction-based determinization

We first show that both Büchi and co-Büchi HD automata can be determinized with a construction that is similar to the subset construction used for determinization of NFWs [49]. As noted above, for the Büchi case, this is a significant improvement over determinization of general NBWs [50]. For the co-Büchi case, determinization of general NCWs involves a "break-point construction", which augments the subset construction by a set that keeps track of visits to states in α, and involves a 3^n blow up [40], which is tight [10]. Thus, also in the co-Büchi case, deterninization of HD automata is simpler than determinization of general automata.

Theorem 4.1. *[35, 32] HD-NBWs (HD-NCWs) are as expressive as DBWs (respectively, DCWs). Given an HD-NBW (HD-NCW) with n states, we can construct an equivalent DBW (respectively, DCW) with 2^n states. Similar results hold for automata with transition-based acceptance.*

Proof: We start with Büchi automata. Consider an HD-NBW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. We assume that $\mathcal{A}$ is semantically deterministic, thus its nondeterministic choices lead to states with the same language. As detailed in Section 2, every HD automaton can be pruned in polynomial time to a semantically deterministic automaton.

We define the DBW $\mathcal{A}' = \langle \Sigma, 2^Q, \{q_0\}, \delta', \alpha' \rangle$, where $\alpha' = \{S \in 2^Q : S \subseteq \alpha\}$, and the transition function δ' is defined for every state $S \in 2^Q$ and letter $\sigma \in \Sigma$ as follows. If $\delta(S, \sigma) \cap \alpha = \emptyset$, then $\delta'(S, \sigma) = \delta(S, \sigma)$. Otherwise, namely if $\delta(S, \sigma) \cap \alpha \neq \emptyset$, then $\delta'(S, \sigma) = \delta(S, \sigma) \cap \alpha$. Thus, we proceed as in the standard subset construction, except that whenever a constructed set contains a state in α, we leave in the set only states in α.

The key observation about the correctness of the construction is that when $\mathcal{A}$ is semantically deterministic, then for all reachable states S of $\mathcal{A}'$, and all states $q, q' \in S$, we have that $\mathcal{A}^q$ and $\mathcal{A}^{q'}$ are equivalent.

Indeed, if $\mathcal{A}$ is semantically deterministic, then for every two states $q, q' \in Q$, letter $\sigma \in \Sigma$, and states $s \in \delta(q, \sigma)$ and $s' \in \delta(q', \sigma)$, if q and q' are equivalent, then so are s and s'. Also, by the definition of δ', every reachable state S of $\mathcal{A}'$ contains only states in α or only states not in α. As we formally prove below, these properties guarantee that indeed $L(\mathcal{A}') = L(\mathcal{A})$.

We first prove that $L(\mathcal{A}') \subseteq L(\mathcal{A})$. Let $r_{\mathcal{A}'} = S_0, S_1, S_2, \ldots$ be an accepting run of $\mathcal{A}'$ on a word $w = \sigma_1 \cdot \sigma_2 \cdots$. We construct an accepting run of $\mathcal{A}$ on w. Since $r_{\mathcal{A}'}$ is accepting, there are infinitely many positions $j_1, j_2, \ldots$ with $S_{j_i} \in \alpha'$. Let $j_0 = 0$, and consider the DAG $G = \langle V, E \rangle$, where

- $V \subseteq Q \times \mathbb{N}$ is the union $\bigcup_{i \geq 0}(S_{j_i} \times \{i\})$.
- $E \subseteq \bigcup_{i \geq 0}(S_{j_i} \times \{i\}) \times (S_{j_{i+1}} \times \{i+1\})$ is such that for all $i \geq 0$, it holds that $E(\langle s', i \rangle, \langle s, i+1 \rangle)$ iff there is a finite run from s' to s over $w[j_i + 1, j_{i+1}]$. Then, we label this edge by the run from s' to s.

By the definition of $\mathcal{A}'$, for every $j \geq 0$ and state $s_{j+1} \in S_{j+1}$, there is a state $s_j \in S_j$ such that $s_{j+1} \in \delta(s_j, \sigma_j)$. Thus, it follows by induction that for every $i \geq 0$ and state $s_{i+1} \in S_{j_{i+1}}$, there is a state $s_i \in S_{j_i}$ such that there is a finite run from s_i to s_{i+1} on $w[j_i + 1, j_{i+1}]$. Thus, the DAG G has infinitely many reachable vertices from the vertex $\langle q_0, 0 \rangle$. Also, as the nondeterminism degree of $\mathcal{A}$ is finite, so is the branching degree of G. Thus, by König's Lemma, G includes an infinite path, and the labels along the edges of this path define a run of $\mathcal{A}$ on w. Since for all $i \geq 1$, the state S_{j_i} is in α', and so all the states in S_{j_i} are in α, this run is accepting, and we are done.

For the other direction, assume that $w = \sigma_1 \cdot \sigma_2 \cdots \in L(\mathcal{A})$, and let $r = r_0, r_1, \ldots$ be an accepting run of $\mathcal{A}$ on w. Let $r' = S_0, S_1, S_2 \ldots$ be the run of $\mathcal{A}'$ on w, and assume, by way of contradiction, that r' is not accepting, thus there is a position $j \geq 0$ such that $S_l \notin \alpha'$, for all $l \geq j$. Consider a state S of $\mathcal{A}'$ and a letter $\sigma \in \Sigma$. By the definition of $\mathcal{A}'$, if $S' = \delta'(S, \sigma)$ and $S' \notin \alpha'$, then all the σ-successors of a state $s \in S$ are in not in α. Applying the above observation iteratively, we get that all the runs of a state $s_j \in S_j$ on the suffix $w[j+1, \infty]$ never visit an α state. Thus, for all $s_j \in S_j$, we have that $\mathcal{A}^{s_j}$ does not accept $w[j+1]$. We claim that r_j is equivalent (in $\mathcal{A}$) to all the states in S_j, which is a contradiction, as $\mathcal{A}^{r_j}$ does accept $s[j+1, \infty]$.

Consider states $q \in Q$, a letter $\sigma \in \Sigma$, and a state $q' \in \delta(q, \sigma)$. Since $\mathcal{A}$ is semantically deterministic, the definition of $\mathcal{A}'$ implies that if q is equivalent (in $\mathcal{A}$) to all the states in some set $S \in 2^Q$, then q' is equivalent (in $\mathcal{A}$) to all the states in $\delta'(S, \sigma)$. Now, since $r_0 = q_0$ and $S_0 = \{q_0\}$, an iterative application of the above observation implies that indeed r_j is equivalent to all the states in S_j, and we are done.

We continue to the co-Büchi automata, where the construction is similar, except that in $\mathcal{A}'$, we try to proceed to states that are not in α. Formally, $\mathcal{A}' = \langle \Sigma, 2^Q, \{q_0\}, \delta', \alpha' \rangle$, where $\alpha' = \{S \in 2^Q : S \subseteq \alpha\}$ is as in the Büchi case, and the transition function δ' is defined for every state $S \in 2^Q$ and letter $\sigma \in \Sigma$ as follows. If $\delta(S, \sigma) \subseteq \alpha$, then $\delta'(S, \sigma) = \delta(S, \sigma)$. Otherwise, namely if $\delta(S, \sigma) \cap (Q \setminus \alpha) \neq \emptyset$, then $\delta'(S, \sigma) = \delta(S, \sigma) \setminus \alpha$. Thus, whenever a constructed set contains a state not in α, we leave in the set only states not in α. The proof is based on the semantic determinism of $\mathcal{A}$ and follows the same considerations as in the Büchi case.

Finally, for automata with transition-based acceptance, the constructions are also similar, except that we restrict the sets according to the membership of transitions in α. For example, in the case of HD-tNBW, consider a state $S \in 2^Q$ and a letter $\sigma \in \Sigma$. If all the σ-transitions in $\mathcal{A}$ from states in S are not in α, then $\mathcal{A}'$ proceeds from S to the set of all the σ-successors of S in $\mathcal{A}$, and the transition is not in α'. If some σ-transitions from S are in α, then $\mathcal{A}'$ proceeds only with these transitions, and the transition is in

α. Formally, for $S \in 2^Q$ and $\sigma \in \sigma$, let

$$S' = \{s' : \text{ there is } s \in S \text{ such that } \langle s, \sigma, s' \rangle \in \alpha\}.$$

Now, if $S' = \emptyset$, then $\delta'(S, \sigma) = \delta(S, \sigma)$ and $\langle S, \sigma, \delta(S, \sigma) \rangle \notin \alpha'$, and if $S' \neq \emptyset$, then $\delta'(S, \sigma) = S'$ and $\langle S, \sigma, S' \rangle \in \alpha'$. □

4.2 Determinization via complementation

For a language $L \subseteq \Sigma^\omega$, the complement of L, denoted $comp(L)$, is the set of infinite words not in L, thus $comp(L) = \Sigma^\omega \setminus L$. In this section, we show that determinization of HD automata cannot induce an exponential blowup for both a language and its complement. This is different from the situation for general nondeterministic automata, where a blowup may occur for both languages. For example, consider the family of languages of finite words $L_k = (a + b)^* \cdot a \cdot (a + b)^{k-1}$. While for all $k \geq 1$, both L_k and $\{a, b\}^* \setminus L$ can be recognized by nondeterministic automata with $k + 1$ states, a deterministic automaton for L_k must have at least 2^k states.

The above property of HD automata was proved in [8] for HD automata with the Rabin acceptance condition.[3] Here we give a variant of the proof there, focusing on Büchi and co-Büchi automata.

Theorem 4.2. *Consider a language $L \subseteq \Sigma^\omega$. If there is an HD-NBW for L with n states, and an HD-NCW for $comp(L)$ with m states, then there is a DBW for L with nm states.*

Proof: Let $\mathcal{A}_1 = \langle \Sigma, Q_1, q_1^0, \delta_1, \alpha_1 \rangle$ be an HD-NBW for L, and $\mathcal{A}_2 = \langle \Sigma, Q_2, q_2^0, \delta_2, \alpha_2 \rangle$ be an HD-NCW for $comp(L)$. Consider the nondeterministic automaton $\mathcal{A}$ obtained by taking the product of $\mathcal{A}_1$ with $\mathcal{A}_2$. Thus, $\mathcal{A} = \langle \Sigma, Q_1 \times Q_2, \langle q_1^0, q_2^0 \rangle, \delta, \alpha \rangle$, where for every state $\langle q_1, q_2 \rangle \in Q_1 \times Q_2$ and letter $\sigma \in \Sigma$, we have that $\delta(\langle q_1, q_2 \rangle, \sigma) = \delta_1(q_1, \sigma) \times \delta_2(q_2, \sigma)$. It is easy to see that if we define $\mathcal{A}$ as an NBW with $\alpha = \alpha_1 \times Q_2$, we get that $L(\mathcal{A}) = L(\mathcal{A}_1) = L$, and if we define $\mathcal{A}$ as an NCW with $\alpha = Q_1 \times \alpha_2$, we get that $L(\mathcal{A}) = L(\mathcal{A}_2) = comp(L)$. Also, as every word in Σ^ω is either in L or in $comp(L)$, if we define $\mathcal{A}$ with a Rabin condition $\alpha = \{\langle \alpha_1 \times Q_2, \emptyset \rangle, \langle Q_1 \times Q_2, Q_1 \times \alpha_2 \rangle\}$ with two pairs (for readers not familiar with the Rabin acceptance condition, a run satisfies α iff its projection on Q_1 satisfies the Büchi condition α_1 or its projection on Q_2 satisfies the co-Büchi condition α_2), we get that $L(\mathcal{A}) = \Sigma^\omega$. We argue that in all three cases, $\mathcal{A}$ is DBP. Since the number of states in $\mathcal{A}$ is $|Q_1 \times Q_2|$, the theorem follows.

Consider the following game between Player $\exists$ and Player $\forall$. The game is played on $\mathcal{A}$ and starts from position $\langle q_1^0, q_2^0 \rangle$. When the game is in position $\langle q_1, q_2 \rangle \in Q_1 \times Q_2$, Player $\forall$ chooses a letter $\sigma \in \Sigma$, and Player $\exists$ chooses a successor position $\langle q_1', q_2' \rangle \in \delta(\langle q_1, q_2 \rangle, \sigma)$. The outcome of a play is an infinite run $r = \langle q_1^0, q_2^0 \rangle, \langle q_1^1, q_2^1 \rangle, \langle q_1^2, q_2^2 \rangle, \langle q_1^3, q_2^3 \rangle, \ldots$ of $\mathcal{A}$. Note that r combines a run $r_1 = q_1^0, q_1^1, q_1^2, q_1^3, \ldots$ of $\mathcal{A}_1$ with a run $r_2 = q_2^0, q_2^1, q_2^2, q_2^3, \ldots$ of $\mathcal{A}_2$, both on the word w obtained by concatenating the letters chosen by Player $\forall$.

The winning condition for Player $\exists$ is that either r_1 satisfies α_1 or r_2 satisfies α_2. This winning condition can be specified by a Rabin winning condition with two pairs: $\{\langle \alpha_1 \times Q_2, \emptyset \rangle, \langle Q_1 \times Q_2, Q_1 \times \alpha_2 \rangle\}$. It is easy to see that following the HD strategies of both automata is a winning strategy for Player $\exists$. Indeed, this strategy guarantees that if the word w is in L, the run r_1 is accepting in $\mathcal{A}_1$ and thus satisfies the Büchi condition α_1, and if $w \in comp(L)$, then the run r_2 is accepting in $\mathcal{A}_2$ and thus satisfies the co-Büchi

[3] We have not defined the Rabin acceptance condition in Section 2. The condition consists of a set of pairs of sets of states [47]. Thus, when the automaton is defined with respect to a set of states Q, it is of the form $\{\langle G_1, B_1 \rangle, \langle G_2, B_2 \rangle, \ldots, \langle G_k, B_k \rangle\}$, with $G_i, B_i \subseteq Q$, and a run r is accepting if for some $1 \leq i \leq k$, we have that $inf(r) \cap G_i \neq \emptyset$ and $inf(r) \cap B_i = \emptyset$.

condition α_2. Since every word is either in L or in $comp(L)$, the winning condition for Player $\exists$ is always satisfied.

It is known that Rabin games admit memoryless strategies [30, 29]. Hence, Player $\exists$ actually has a memoryless winning strategy in the game. Such a strategy maps each position $\langle q_1, q_2\rangle \in Q_1 \times Q_2$ and letter $\sigma \in \Sigma$ to a position $\langle q_1', q_2'\rangle$, and induces the required pruning of $\mathcal{A}$ into a deterministic automaton $\mathcal{A}'$. Specifically, $\mathcal{A}'$ with the Rabin condition $\{\langle \alpha_1 \times Q_2, \emptyset\rangle, \langle Q_1 \times Q_2, Q_1 \times \alpha_2\rangle\}$ accepts all the words in Σ^ω, then $\mathcal{A}'$ with the Büchi condition $\alpha_1 \times Q_2$ is a DBW for L, and $\mathcal{A}'$ with a co-Büchi condition $Q_1 \times \alpha_2$ is a DCW for $comp(L)$. □

In Section 4.3, we use Theorem 4.2 in order to obtain both upper and lower bounds on the succinctness of HD automata with respect to deterministic ones.

4.3 Succinctness

By Theorem 4.2, an upper bound on the complementation of HD automata implies an upper bound also on their determinization. Specifically, if $f : \mathbb{N} \to \mathbb{N}$ is a function such that complementing an HD automaton $\mathcal{A}$ with n states results in an HD automaton with at most $f(n)$ states, then determinization of $\mathcal{A}$ results in an automaton with at most $n \cdot f(n)$ states. In [32], the authors describe a linear complementation construction for HD-NBWs. Hence, HD-NBWs are at most quadratically more succinct than DBWs, and the same holds for HD-tNBWs. For the co-Büchi acceptance condition, Kuperberg and Skrzypczak proved that HDness can lead to a significant succinctness. Their proof makes use of Theorem 4.2 in the following way. Consider a language $L \subseteq \Sigma^\omega$, and assume that $\mathcal{A}$ is an HD automaton for L. By Theorem 4.2, an HD automaton for $comp(L)$ can serve as a "memory structure" that generates a strategy for $\mathcal{A}$: taking its product with $\mathcal{A}$, we obtain a deterministic automaton that inherits its acceptance condition from $\mathcal{A}$. Since every deterministic automaton is an HD automaton, and deterministic automata can be complemented by dualization, it follows that every deterministic automaton for L can also serve as a memory structure for an HD automaton for L. As we shall see now, this property is useful in the proof of the exponential succinctness of HD-NCWs with respect to DCWs. We state the theorem in the transition-based setting. Similar considerations hold in the state-based setting.

Theorem 4.3. [32] *There is an infinite family of languages $L_1, L_2, L_3, \ldots$ such that for every $n \geq 1$, the following holds.*

1. *There is an HD-tNCW with $2n$ states that recognizes L_n.*
2. *Every tDCW that recognizes L_n needs at least $\frac{2^n}{2n}$ states.*

Proof: For $n \geq 1$, let $[n] = \{0, 1, \ldots, 2n-1\}$. We define the language L_n over the alphabet $\Sigma = \{I, Z, X, H\}$. Each letter in Σ is a (possibly partial) function $\sigma : [n] \to [n]$, as described in Figure 3.

The functions I, X, and Z are one-one and onto: for every $x \in [n]$, we have that $I(x) = x$, $Z(x) = (x+1) \bmod 2n-1$, and X agrees with I, except for $x \in \{0,1\}$, where $X(0) = 1$ and $X(1) = 0$. The function H is partial; it agrees with I, except for $x = 0$, where $H(0)$ is undefined. Thus, the letters induce permutations on $[n]$, with H inducing a permutation only on $[n] \setminus \{0\}$.

We view a finite word w as the partial function $w : [n] \to [n]$ obtained by composing its letters. Thus, if $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_l$, then for all $i \in [n]$, we have that $w(i) = \sigma_l(\cdots \sigma_2(\sigma_1(i)))$. It is convenient to associate with each word $w \in \Sigma^*$ a grid of dimensions $(|w|+1) \times 2n$, and lines that start in "floors" in $[n]$ and

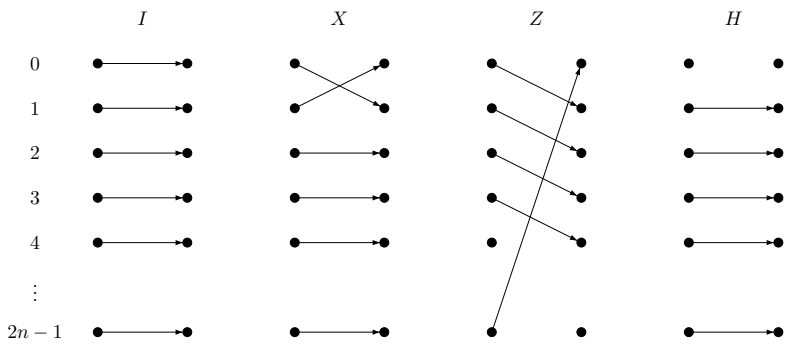

Figure 3: The permutations induced by the letters I, X, Z, and H.

traverse the floors along the grid according to the permutations induced by the letters in w. As $H(0)$ is undefined, a line that reaches floor 0 before H is read has a "hole" in the corresponding position in the grid. Figure 4 describes the grid associated with the word $IXHZZXHZ$ when $n = 2$ and $n = 3$.

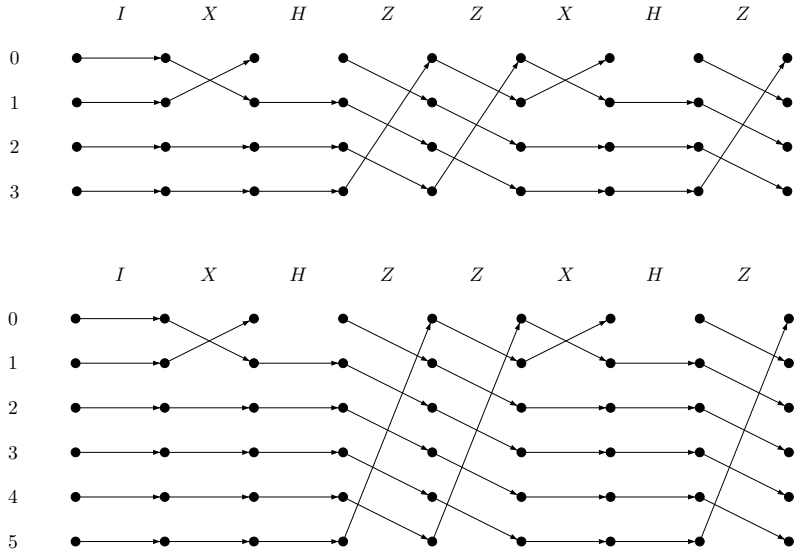

Figure 4: The grid induced by the word $IXHZZXHZ$ when $n = 2$ (top) and $n = 3$ (bottom).

An infinite word $u \in \Sigma^\omega$ corresponds to an infinite sequence of compositions of its letters, and thus the horizontal dimension of the grid associated with it is infinite. We define L_n as the set of words in Σ^ω whose grid contains an infinite line; that is, a line that has only finitely many holes. For example, back to Figure 4, it is not hard to see that the infinite word $u = w^\omega$, for $w = IXHZZXHZ$ is in L_2. Indeed, when $n = 2$, we have that $w(0) = 0$ and $w(2) = 2$, and so the lines starting at floors 0 and 2 are never cut. On the other hand, $u \notin L_3$. Indeed, when $n = 3$, we have that $w(0) = 4$, $w(2) = 5$, $w(3) = 0$, $w(4) = 2$, whereas $w(1)$ and $w(5)$ are undefined. Accordingly, $w^5(i)$ is undefined for all $i \in [3]$, implying that lines from all floors are cut whenever w^5 is read. Therefore, the grid of u contains no infinite line, and so $u \notin L_3$.

We first prove that there is an HD-tNCW with $2n$ states that recognizes L_n. It is easy to see that L_n

can be recognized by a tNCW with $2n$ states. Indeed, a tNCW can simply guesses a line to follow, and to initiate its guess whenever the line it follows is cut. Specifically (see $\mathcal{A}_2$ in Figure 5), the tNCW $\mathcal{A}_n$ has state space $\{q_0, \ldots, q_{2n-1}\}$ and for all $i \in [n]$, it visits q_i when the line it follows is in floor i. The initial state of $\mathcal{A}_n$ is arbitrarily set to q_0, and the transition function updates the floor according to the letter it reads. For example, when $\mathcal{A}_n$ is in state q_i and reads I, it stays in q_i, when it reads X, if stays in q_i for $i \in \{2, \ldots, 2n-1\}$, moves to q_1 from q_0, and moves to q_0 from q_1. Nondeterminism is required when $\mathcal{A}_n$ reads the letter H in state q_0, thus when it follows a line that is in floor 0 and the line is cut. Then, $\mathcal{A}_n$ guesses a new floor to follow. Since all floors are candidates for hosting an infinite line, $\mathcal{A}_n$ can nondeterministically move from q_0 with H to all states. Since the input word is in the language if it contains an infinite line, thus if it is possible to eventually follow a line that is never cut, we want an accepting run to take only finitely many H-transitions from the state q_0, thus α is the set of these transitions.

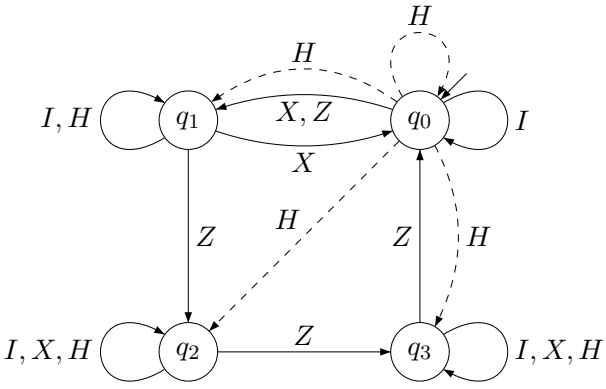

Figure 5: The HD-tNCW $\mathcal{A}_2$ that recognizes L_2.

It is less easy to see that the tNCW $\mathcal{A}_n$ is in fact HD. In order to see this, consider the following HD strategy $g : \Sigma^* \to Q$. First, $g(\epsilon) = q_0$, thus all runs start in state q_0, which is the only initial state of $\mathcal{A}_n$. Whenever a run is in state q_0 after reading a prefix u and it reads H, it proceeds to the state q_i such that the line that is now in floor i is the longest among all lines in the graph. Formally, for all words $u \in \Sigma^*$ and floors $i \in [n]$, let $\mathit{seniority}(u, i)$ be the length of the longest suffix u' of u such that there is $j \in [n]$ with $u'(j) = i$. Then, if $u \in \Sigma^*$ is such that $g(u) = q_0$, then $g(u \cdot H) = q_i$, for the minimal $i \in [n]$ that maximizes $\mathit{seniority}(u, i)$. Note that the choice of the minimal i is arbitrary, and it is required in order to decide between lines with the same seniority. Note also that it is possible to implement the HD strategy g by maintaining the order of seniority among the different floors during the run, thus it indeed depends only on the history of the run. Finally, as an infinite line would eventually obtain the maximal seniority, it is guaranteed that following the strategy g leads to accepting all words in the language: in all of then, the run that follows g eventually follows an infinite line.

It is left to prove that a tDCW for L_n needs at least $2^n - 1$ states. The full proof, in [32], is based on the following arguments: The first argument refers to the ability to generate with finite words over $\{I, Z, X\}$ every permutation on $[n]$. The proof in [32] focuses on *pair-based* permutations, namely permutations that map each floor $i \in [n]$ to floor $2\lfloor \frac{i}{2} \rfloor$ or $2\lfloor \frac{i}{2} \rfloor + 1$. That is, for every $j \in \{0, \ldots, n-1\}$, the permutation map $\{2j, 2j+1\}$ onto itself. Note that there are 2^n pair-based permutations, and that each pair-based permutation can be generated by a word in $((I + X) \cdot ZZ))^n$. Indeed, the word that generates a pair-based permutation π is $\sigma_0 \cdot ZZ \cdot \sigma_{n-2} \cdot ZZ \cdot \sigma_{n-1} \cdot ZZ \cdot \sigma_2 \cdot ZZ \cdot \sigma_1 \cdot ZZ$, where for all $j \in \{0, \ldots, n-1\}$, we have

that $\sigma_j = I$ if π does not switch $2j$ and $2j+1$ and $\sigma_j = X$ if π switches $2j$ and $2j+1$. For example (see figure 6), when $n = 3$, the word $IZZXZZXZZ$ induces the permutation $\langle 013254 \rangle$, thus it switches $\{2, 3\}$ and $\{4, 5\}$. It is easy to see that in a similar manner, we can generate with finite words over $\{I, Z, X\}$ every permutation on $[n]$, and not only pair-based ones.

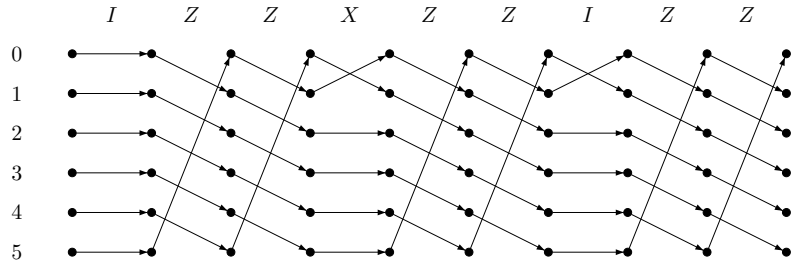

Figure 6: The word $IZZXZZXZZ$ generates the pair-based permutation $\langle 013254 \rangle$.

The second argument is that when we discuss potential tDCWs for L_n, we can restrict attention to tDCWs that are obtained by taking the product of $\mathcal{A}_n$ with a deterministic memory structure. The argument proceeds as follows. Given a tDCW with $f(n)$ states for L_n, we can dualize it and obtain a tDBW for $comp(L_n)$, which is also an HD-tNBW for $comp(L_n)$. Then, as specified in the proof of Theorem 4.2, we can take the product of this HD-tNBW with the HD-tNCW $\mathcal{A}_n$ described above for L_n and obtain a tDCW with $2n \cdot f(n)$ states for L whose acceptance condition is induced by $\mathcal{A}_n$. Thus, its α-transitions are H-transitions that leave states of the form $\langle q_0, s \rangle$ for a state s of the claimed tDCW. Note that in this process we start with a tDCW with $f(n)$ states for L_n and obtain an equivalent tDCW with $2n \cdot f(n)$ states, which may seem a bad idea in the context of proving a lower bound. The information, however, that we gain about the structure of the obtained tDCW makes this "$2n$ penalty" worthwhile.

The third, and most complicated argument is that a tDCW that attempts to recognize L_n, is obtained by taking the product of $\mathcal{A}_n$ with a deterministic memory structure, and has less than 2^n states, must err. Essentially, the argument proceeds as follows. Consider a tDCW $\mathcal{D}_n$ as above. Since $\mathcal{D}_n$ is obtained by taking the product of $\mathcal{A}_n$ with some memory structure, it accepts only words in the language, and so the error we point to is that it rejects a word in L_n. The word along with the run rejecting it are constructed as follows. The run starts from some state of the form $\langle q_0, s_0 \rangle$. Since $\mathcal{D}_n$ has less than 2^n states, there are two different pair-based permutations π_1 and π'_1 that lead from $\langle q_0, s_0 \rangle$ to the same state $\langle q_{i_1}, s_1 \rangle$. Since $\pi_1 \neq \pi'_1$, there is $j \in \{0, \ldots, n-1\}$ such that $\pi_1(2j) \neq \pi'_1(2j)$. Thus, $\{\pi_1(2j), \pi'_1(2j)\} = \{2j, 2j+1\}$. This implies that the size of the set $F = \{\pi_1(1), \pi_1(3), \ldots, \pi_1(2n-1), \pi'_1(2j)\}$ is $n+1$. Therefore, there is a permutation τ_1 that behaves as follows: First, it maps i_1 (that is, the floor that $\mathcal{D}_n$ now follows, as its $\mathcal{A}_n$ component is in state q_{i_1}) to 0. In addition, if $i_1 \in F$, then the other n elements in F are mapped to $\{1, 3, \ldots, 2n-1\}$, and if $i_1 \notin F$, then n elements in F are mapped to $\{1, 3, \ldots, 2n-1\}$, and one element is mapped arbitrarily. Note that since $\mathcal{D}_n$ is based on $\mathcal{A}_n$, its run on τ_1 from the state $\langle q_{i_1}, s_1 \rangle$, reaches a state of the form $\langle q_0, s'_1 \rangle$. Indeed, the operation of $\mathcal{A}_n$ on τ_1 is deterministic and it follows the states that correspond to the floors visited along the execution to the permutation τ_1, which maps i_1 to 0. Thus, when $\mathcal{D}_n$ reads either $\pi_1 \cdot \tau_1 \cdot H$ or $\pi'_1 \cdot \tau_1 \cdot H$, the transition taken when the last letter H is read, is an α-transition. On the other hand, the grid induced by at least one of these words, includes a line that is not cut and reaches one of the floors in $\{1, 3, \ldots, 2n-1\}$. Let $\langle q_{i'_1}, s''_1 \rangle$ be the H-successor of $\langle q_0, s'_1 \rangle$ in $\mathcal{D}_n$. We can now continue the generation of the run by considering two different pair-based permutations π_2 and π'_2

that lead from $\langle q_{i'_1}, s''_1 \rangle$ to the same state $\langle q_{i_2}, s_2 \rangle$ and a permutation τ_2 whose composition with π_2 and π'_2 maps i_2 to 0 and guarantees that the line to at least one floor in $\{1, 3, \ldots, 2n-1\}$ is not cut. Continuing in this manner, it can be shown that while the runs on all the words in $(\pi_1 + \pi'_1) \cdot \tau_1 \cdot H \cdot (\pi_2 + \pi'_2) \cdot \tau_2 \cdot H \cdots$ are rejecting, at least one of these words is in L_n. □

5 Variants, Extensions, and Open Problems

Since their introduction, history-deterministic automata have attracted a lot of interest in the research community. Indeed, beyond their practical usefulness, history determinism is theoretically interesting and intriguing, relevant to computation models beyond nondeterministic automata on infinite words, and many natural questions around it are still open. This survey focuses on some key results about HD nondeterministic Büchi and co-Büchi automata on infinite words. Due to the lack of space, several beautiful results, like the linear complementation and quadratic determinization of HD-NBWs [32], algorithms for deciding HDness [32, 7], relations to other types of bounded nondeterminism [4, 11], and results on HD automata with richer acceptance conditions [11, 9, 17] are not included.

History determinism has been studied also for *alternating*, *pushdown*, and *quantitative* automata. We briefly describe these models here. For an excellent recent survey, see [14]. An *alternating automaton* has both nondeterministic and universal transitions. While a nondeterministic transition stands for an existential choice, thus a run may choose a successor state to proceed to, in a universal transition the run should proceed to all successors [18]. Thus, universality actually involves no choices that have to be resolved, and one could have defined HD alternating automata as ones in which the nondeterministic choices can be resolved in a way that only depends on the past. One of the main features, however, of alternating automata, is the *duality* between the nondeterministic and universal choices. In particular, an alternating automaton $\mathcal{A}$ can be complemented (that is, turned into an automaton $\tilde{\mathcal{A}}$ for the complementing language) by dualizing its transition function (that it, making all nondeterministic branches universal, and all universal branches nondeterministic) and acceptance condition (that is, switching between Büchi and co-Büchi). With this duality in mind, Colcombet defined HD alternating automata with cost functions as automata that have two strategies – one for resolving nondeterministic choices in $\mathcal{A}$ and one for resolving nondeterministic choices in $\tilde{\mathcal{A}}$ [20]. HD alternating automata for ω-regular languages are further studied in [12].

History-deterministic *ω-pushdown automata* were studied in [39, 24]. Their definition is similar to the definition of HD ω-regular automata, except that now, the nondeterminism that the strategy resolves corresponds to the different choices in the transition function of pushdown automata; thus the strategy maps the history to both the next state of the automaton and the operations on the stack.

Quantitative automata define weighted languages, namely mappings from words to values. In these automata, nondeterminism essentially amounts to letting the automaton choose a run that leads to the best value. For example, if the value of a run is the limit average of values of transitions taken along the run, and the setting corresponds to a maximization question, then the value of a word is the supremum value of all the runs on it. Recall that a strategy of an HD Boolean automaton has to generate a run that accepts all words in the language of the automaton. In the quantitative setting, we want the strategy to generate a run that attains the supremum value (or, in a minimization setting, the infimum value) [13]. The quantitative setting calls for variants in which we seek strategies that approximate the optimal value or attain values above or below some threshold. In fact, the very first work of Colcombet on HD automata

studied strategies that approximate *regular cost functions* [19], and the analysis of the competitive-ratio of on-line algorithms in [5, 6] studied approximating HD quantitative automata.

An orthogonal extension of HD automata is motivated by their use in the synthesis problem. Recall that the problem can be reduced to solving a game on top of an HD automaton for the specification. HD automata are defined for general alphabets, whereas in the synthesis problem, the specification is over an alphabet $2^{I \cup O}$, for sets I and O of input and output signals, respectively. In [22], the authors introduced (I/O)-*aware HD automata*, which distinguish between nondeterminism due to I and O: both should be resolved in a way that depends only on the past; but while nondeterminism in I is hostile, and all I-futures should be accepted, nondeterminism in O is cooperative, and a single O-future may be accepted. It is shown in [22] that (I/O)-aware HD automata can be used for synthesis, and that they are unboundedly more succinct than deterministic and even HD automata.

Related variants of HD automata have to do with their applications. As discussed in Section 1, HD automata are *good for trees*, in the sense that an HD word automaton for L can be expanded to a tree automaton for $L_{\triangle}$, and are *good for games*, in the sense that synthesis can be reduced to playing a game on top of an HD automaton for the specification. As it turns out, these "goodness" properties need not characterize history determinism in all settings. For example, in the quantitative setting, an automaton may be good for games without being HD [13]. Moreover, even in the Boolean setting, HD may imply, yet not be characterized by, other useful properties. For example, every HD automaton is *good for MDPs*, thus its product with Markov decision processes maintains the probability of acceptance, and can therefore replace deterministic automata when reasoning about stochastic behaviors [25, 53].

Some basic problems around history determinism are still open. Most notable is the succinctness of HD-NBWs with respect to DBWs. Recall that, by [32], HD-NBWs can be determinized with a quadratic blow up. Yet, while we know that not all HD-NBWs are DBP, there is no matching quadratic lower bound, and in fact we still do not have even an example of a language L such that an HD-NBW for L is strictly smaller than a DBW for L. In particular, while the HD-tNBW $\mathcal{A}$ appearing in Figure 1 is not DBP, the tDBW obtained by merging the states q_1 and q_2 of $\mathcal{A}$ recognizes $L(\mathcal{A})$ and is smaller than $\mathcal{A}$.

Additional open problems refer to decision problems around history determinism. One such question is the complexity of deciding whether a given language is *HD-helpful*, namely whether an HD automaton for it is smaller than a deterministic automaton for it. Note that the definition is parameterized by the acceptance condition. For example, as discussed above, possibly there are no languages that are HD-Büchi-helpful (that is, languages L such that an HD-NBW for L is strictly smaller than a DBW for L), and the same for Büchi automata with transition-based acceptance. For co-Büchi automata, we do know that HD-co-Büchi-helpful languages exist. We also know that there are languages that are HD-co-Büchi-helpful only in automata with transition-based acceptance, but no clean characterization of tight complexity of the corresponding decision problems is known.

The problem of HD-helpfulness is related to the fundamental problem of *minimization*: generation of an equivalent automaton with a minimal number of states. For automata on finite words, the picture is well understood: For nondeterministic automata, minimization is PSPACE-complete [28], whereas for deterministic automata, a minimization algorithm, based on the Myhill-Nerode right congruence [42, 43], generates in polynomial time a canonical minimal deterministic automaton [27]. Essentially, the canonical automaton, a.k.a. the *quotient automaton*, is obtained by merging equivalent states. For automata on infinite words, merging of equivalent states fails, and minimization of DBWs (and hence, also DCWs, as the two dualize each other) is NP-complete [51]. In [1], Abu-Radi and Kupferman described a poly-

nomial minimization algorithm for HD-tNCW. Considering HD-tNCWs rather than DCWs involves two modifications: considering HD rather than deterministic automata, and considering transition-based rather than state-based acceptance. A natural question that arises is whether both modifications are crucial for efficiency. In [52], Schewe proved that his NP-hardness result of DCW minimization can be generalized to HD-NCWs. This suggests that the consideration of transition-based acceptance is crucial, and makes the study of tDBWs and tDCWs minimization, which is still open, very interesting. Moreover, for the richer acceptance condition of Rabin, Casares proved that minimization is NP-hard for HD automata with transition-based acceptance [16], and for automata with state-based acceptance, [3] shows that minimization is NP-hard already for automata that recognize fragments of ω-regular languages, in particular for automata that recognize liveness languages. The minimization algorithm of [1] also implies canonicity for HD-tNCWs: all minimal automata have isomorphic safe components (namely components obtained by restricting the transitions to these not in α) and once we saturate the automata with α-transitions, we get full isomorphism. This is in contrast with DCW, where no canonicity exists [2].

References

[1] B. Abu Radi and O. Kupferman. Minimizing GFG transition-based automata. In *Proc. 46th Int. Colloq. on Automata, Languages, and Programming*, volume 132 of *LIPIcs*, pages 100:1–100:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

[2] B. Abu Radi and O. Kupferman. Canonicity in GFG and transition-based automata. In *Proc. 11th International Symposium on Games, Automata, Logics and Formal Verification*. Electronic Proceedings in Theoretical Computer Science, 2020.

[3] B. Abu Radi and O. Kupferman. Minimization of automata for liveness languages. In *20th Int. Symp. on Automated Technology for Verification and Analysis*, volume 13505 of *Lecture Notes in Computer Science*, pages 191–207. Springer, 2022.

[4] B. Abu Radi, O. Kupferman, and O. Leshkowitz. A hierarchy of nondeterminism. In *46th Int. Symp. on Mathematical Foundations of Computer Science*, volume 202 of *LIPIcs*, pages 85:1–85:21, 2021.

[5] B. Aminof, O. Kupferman, and R. Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 6(2), 2010.

[6] B. Aminof, O. Kupferman, and R. Lampert. Formal analysis of online algorithms. In *9th Int. Symp. on Automated Technology for Verification and Analysis*, volume 6996 of *Lecture Notes in Computer Science*, pages 213–227. Springer, 2011.

[7] M. Bagnol and D. Kuperberg. Büchi good-for-games automata are efficiently recognizable. In *Proc. 38th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 122 of *LIPIcs*, pages 16:1–16:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[8] U. Boker, D. Kuperberg, O. Kupferman, and M. Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In *Proc. 40th Int. Colloq. on Automata, Languages, and Programming*, volume 7966 of *Lecture Notes in Computer Science*, pages 89–100, 2013.

[9] U. Boker, D. Kuperberg, K. Lehtinen, and M. Skrzypczak. On the succinctness of alternating parity good-for-games automata. In *Proc. 40th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 182 of *LIPIcs*, pages 41:1–41:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[10] U. Boker, O. Kupferman, and A. Rosenberg. Alternation removal in Büchi automata. In *Proc. 37th Int. Colloq. on Automata, Languages, and Programming*, volume 6199, pages 76–87, 2010.

[11] U. Boker, O. Kupferman, and M. Skrzypczak. How deterministic are Good-For-Games automata? In *Proc. 37th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 93 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:14, 2017.

[12] U. Boker and K. Lehtinen. Good for games automata: From nondeterminism to alternation. In *Proc. 30th Int. Conf. on Concurrency Theory*, volume 140 of *LIPIcs*, pages 19:1–19:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[13] U. Boker and K. Lehtinen. History determinism vs. good for gameness in quantitative automata. In *Proc. 41st Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 213 of *LIPIcs*, pages 38:1–38:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[14] U. Boker and K. Lehtinen. When a little nondeterminism goes a long way: an introduction to history-determinism. *ACM SIGLOG News*, 10(1):177–196, 2023.

[15] J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12. Stanford University Press, 1962.

[16] A. Casares. On the minimisation of transition-based rabin automata and the chromatic memory requirements of muller conditions. In *Proc. 30th Annual Conf. of the European Association for Computer Science Logic*, volume 216 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[17] A. Casares, T. Colcombet, and K. Lehtinen. On the size of good-for-games rabin automata and its link with the memory in muller games. In *Proc. 49th Int. Colloq. on Automata, Languages, and Programming*, volume 229 of *LIPIcs*, pages 117:1–117:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[18] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, 1981.

[19] T. Colcombet. The theory of stabilisation monoids and regular cost functions. In *Proc. 36th Int. Colloq. on Automata, Languages, and Programming*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009.

[20] T. Colcombet. Fonctions régulières de coût. Habilitation thesis, Université Paris Diderot – Paris 7, in French, 2013.

[21] J. Esparza, J. Kretínský, and S. Sickert. One theorem to rule them all: A unified translation of LTL into ω-automata. In *Proc. 33rd IEEE Symp. on Logic in Computer Science*, pages 384–393, 2018.

[22] R. Faran and O. Kupferman. On (I/O)-aware good-for-games automata. In *18th Int. Symp. on Automated Technology for Verification and Analysis*, volume 12302 of *Lecture Notes in Computer Science*, pages 161–178. Springer, 2020.

[23] E. Grädel, W. Thomas, and T. Wilke. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[24] S. Guha, I. Jecker, K. Lehtinen, and M. Zimmermann. A bit of nondeterminism makes pushdown automata expressive and succinct. In *46th Int. Symp. on Mathematical Foundations of Computer Science*, volume 202 of *LIPIcs*, pages 53:1–53:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[25] E.M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In *Proc. 26th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 12078 of *Lecture Notes in Computer Science*, pages 306–323. Springer, 2020.

[26] T.A. Henzinger and N. Piterman. Solving games without determinization. In *Proc. 15th Annual Conf. of the European Association for Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 394–410. Springer, 2006.

[27] J.E. Hopcroft. An $n \log n$ algorithm for minimizing the states in a finite automaton. In Z. Kohavi, editor, *The Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

[28] T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.

[29] C.S. Jutla. Determinization and memoryless winning strategies. *Information and Computation*, 133(2):117–134, 1997.

[30] N. Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. In *Proc. 7th IEEE Symp. on Logic in Computer Science*, pages 382–393. IEEE Computer Society, 1992.

[31] D. Kuperberg and A. Majumdar. Width of non-deterministic automata. In *Proc. 35th Symp. on Theoretical Aspects of Computer Science*, volume 96 of *LIPIcs*, pages 47:1–47:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[32] D. Kuperberg and M. Skrzypczak. On determinisation of good-for-games automata. In *Proc. 42nd Int. Colloq. on Automata, Languages, and Programming*, pages 299–310, 2015.

[33] O. Kupferman. Automata theory and model checking. In *Handbook of Model Checking*, pages 107–151. Springer, 2018.

[34] O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. In *Proc. 11th IEEE Symp. on Logic in Computer Science*, pages 322–333, 1996.

[35] O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. *Ann. Pure Appl. Logic*, 138(1-3):126–146, 2006.

[36] O. Kupferman and S. Sickert. Certifying inexpressibility. In *Proc. 24th Int. Conf. on Foundations of Software Science and Computation Structures*, volume 12650 of *Lecture Notes in Computer Science*, pages 385–405. Springer, 2021.

[37] O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, 2005.

[38] L.H. Landweber. Decision problems for ω–automata. *Mathematical Systems Theory*, 3:376–384, 1969.

[39] K. Lehtinen and M. Zimmermann. Good-for-games ω-pushdown automata. *Logical Methods in Computer Science*, 18(1), 2022.

[40] S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

[41] G. Morgenstern. Expressiveness results at the bottom of the ω-regular hierarchy. M.Sc. Thesis, The Hebrew University, 2003.

[42] J. Myhill. Finite automata and the representation of events. Technical Report WADD TR-57-624, pages 112–137, Wright Patterson AFB, Ohio, 1957.

[43] A. Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958.

[44] D. Niwinski and I. Walukiewicz. Relating hierarchies of word and tree automata. In *Proc. 15th Symp. on Theoretical Aspects of Computer Science*, volume 1373 of *Lecture Notes in Computer Science*. Springer, 1998.

[45] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

[46] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 179–190, 1989.

[47] M.O. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.

[48] M.O. Rabin. Weakly definable relations and special automata. In *Proc. Symp. Math. Logic and Foundations of Set Theory*, pages 1–23. North Holland, 1970.

[49] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.

[50] S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.

[51] S. Schewe. Beyond Hyper-Minimisation—Minimising DBAs and DPAs is NP-Complete. In *Proc. 30th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 400–411, 2010.

[52] S. Schewe. Minimising good-for-games automata is NP-complete. In *Proc. 40th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 182 of *LIPIcs*, pages 56:1–56:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[53] S. Schewe, Q. Tang, and T. Zhanabekova. Deciding what is good-for-MDPs. *CoRR*, abs/2202.07629, 2022.

[54] A.P. Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6:495–511, 1994.

[55] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.