Logical Methods in Computer Science
Volume 21, Issue 4, 2025, pp. 13:1–13:31
https://lmcs.episciences.org/

Submitted Dec. 27, 2023
Published Oct. 28, 2025

A HIERARCHY OF NONDETERMINISM *

BADER ABU RADI, ORNA KUPFERMAN, AND OFER LESHKOWITZ

School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel

Abstract. We study three levels in a hierarchy of nondeterminism: A nondeterministic automaton $\mathcal{A}$ is *determinizable by pruning* (DBP) if we can obtain a deterministic automaton equivalent to $\mathcal{A}$ by removing some of its transitions. Then, $\mathcal{A}$ is *history deterministic* (HD) if its nondeterministic choices can be resolved in a way that only depends on the past. Finally, $\mathcal{A}$ is *semantically deterministic* (SD) if different nondeterministic choices in $\mathcal{A}$ lead to equivalent states. Some applications of automata in formal methods require deterministic automata, yet in fact can use automata with some level of nondeterminism. For example, DBP automata are useful in the analysis of online algorithms, and HD automata are useful in synthesis and control. For automata on finite words, the three levels in the hierarchy coincide. We study the hierarchy for Büchi, co-Büchi, and weak automata on infinite words. We show that the hierarchy is strict, study the expressive power of the different levels in it, as well as the complexity of deciding the membership of a language in a given level. Finally, we describe a probability-based analysis of the hierarchy, which relates the level of nondeterminism with the probability that a random run on a word in the language is accepting. We relate the latter to nondeterministic automata that can be used when reasoning about probabilistic systems.

1. Introduction

Nondeterminism is a fundamental notion in theoretical computer science. It allows a computing machine to examine several possible actions simultaneously. For automata on finite words, nondeterminism does not increase the expressive power, yet it leads to an exponential succinctness [RS59].

A prime application of automata theory is specification, verification, and synthesis of reactive systems [VW94, Kup18]. Since we care about the on-going behavior of nonterminating systems, the automata run on infinite words. Acceptance in such automata is determined according to the set of states that are visited infinitely often along the run. In *Büchi* automata [Büc62], the acceptance condition is a subset α of states, and a run is accepting iff it visits α infinitely often. Dually, in *co-Büchi* automata, a run is accepting iff it visits α only finitely often. We also consider *weak* automata, which are a special case of both Büchi and co-Büchi automata in which no cycle contains both states in α and states not in α. We use three-letter acronyms in $\{\mathrm{D}, \mathrm{N}\} \times \{\mathrm{F}, \mathrm{B}, \mathrm{C}, \mathrm{W}\} \times \{\mathrm{W}\}$ to describe the different classes of automata. The first letter stands for the branching mode of the

Key words and phrases: Automata, Nondeterminism.

* A preliminary version of this paper appears in the Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science, 2021.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.46298/LMCS-21(4:13)2025

automaton (deterministic or nondeterministic); the second for the acceptance condition type (finite, Büchi, co-Büchi or weak); and the third indicates that we consider automata on words.

For automata on infinite words, nondeterminism may increase the expressive power and also leads to an exponential succinctness. For example, NBWs are strictly more expressive than DBWs [Lan69], whereas NCWs are as expressive as DCWs [MH84]. In some applications of the automata-theoretic approach, such as model checking, algorithms can be based on nondeterministic automata, whereas in other applications, such as synthesis and control, they cannot. There, the advantages of nondeterminism are lost, and algorithms involve a complicated determinization construction [Saf88] or acrobatics for circumventing determinization [KV05b]. Essentially, the inherent difficulty of using nondeterminism in synthesis and control lies in the fact that each guess of the nondeterministic automaton should accommodate all possible futures.

A study of nondeterministic automata that can resolve their nondeterministic choices in a way that only depends on the past started in [KSV06], where the setting is modeled by means of tree automata for derived languages. It then continued by means of *history deterministic* (HD) automata [HP06]. [1] A nondeterministic automaton $\mathcal{A}$ over an alphabet Σ is HD if there is a strategy g that maps each finite word $u \in \Sigma^*$ to the transition to be taken after u is read; and following g 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 g may suggest different transitions from q after different words are read. Still, g depends only on the past, namely on the word read so far. Obviously, there exist HD automata: deterministic ones, or nondeterministic ones that are *determinizable by pruning* (DBP); that is, ones that embody an equivalent deterministic automaton. In fact, the HD automata constructed in [HP06] are DBP. [2] Beyond the theoretical interest in DBP automata, they are used for modelling 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 [AKL10, AK15].

For automata on finite words, HD-NFWs are always DBP [KSV06, Mor03]. For automata on infinite words, HD-NBWs and HD-NCWs are as expressive as DBWs and DCWs, respectively [KSV06, NW98], but they need not be DBP [BKKS13]. Moreover, the best known determinization construction for HD-NBWs is quadratic, and determinization of HD-NCWs has a tight exponential blow-up [KS15]. Thus, HD automata on infinite words are (possibly even exponentially) more succinct than deterministic ones.

Further research studies characterization, typeness, complementation, and further constructions and decision procedures for HD automata [KS15, BKS17, BK18], as well as an extension of the HD setting to pushdown ω-automata [LZ20] and to alternating automata [BL19, BKLS20].

[1] The notion used in [HP06] is *good for games* (GFG) automata, as they address the difficulty 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. Therefore, we use the term *history determinism*, introduced by Colcombet in the setting of quantitative automata with cost functions [Col09].

[2] As explained in [HP06], 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.

A nondeterministic automaton is *semantically deterministic* (SD, for short) if its nondeterministic choices lead to states with the same language. Thus, for every state q of the automaton and letter $\sigma \in \Sigma$, all the σ-successors of q have the same language. Beyond the fact that semantically determinism is a natural relaxation of determinism, and thus deserves consideration, SD automata naturally arise in the setting of HD automata. Indeed, if different σ-successors of a state in an HD automaton have different languages, then we can prune the transitions to states whose language is contained in the language of another σ-successor. Moreover, since containment for HD automata can be checked in polynomial time, such a pruning can be done in polynomial time [HKR02, KS15]. Accordingly (see more in Section 2.3), we can assume that all HD automata are SD [KS15].

Hence, we obtain the following hierarchy, from deterministic to nondeterministic automata, where each level is a special case of the levels to its right.

For automata on finite words, as NFWs are as expressive as DFWs, all levels of the hierarchy coincide in their expressive power. In fact, the three internal levels coincide already in the syntactic sense: every SD-NFW is DBP. Also, given an NFW, deciding whether it is SD, HD or DBP, can each be done in polynomial time [AKL10].

For Büchi and co-Büchi automata, the picture is less clear, and is the subject of our research. Before we describe our results, let us mention that an orthogonal level of nondeterminism is that of *unambiguous* automata, namely automata that have a single accepting run on each word in their languages. An unambiguous NFW for a non-empty language is SD iff it is deterministic, and a DBP-NFW need not be unambiguous. It is known, however, that an HD unambiguous NCW, or NBW, is DBP [BKS17].

We study the following aspects and questions about the hierarchy.

Strictness. For each nondeterminism level we study for which acceptance condition it is a strict super class of its preceding level. Recall that not all HD-NBWs and HD-NCWs are DBP [BKKS13], and examples for this include also SD automata. On the other hand, all HD-NWWs (in fact, all HD-NXWs whose language can be recognized by a DWW) are DBP [BKS17]. We show that SD-NXWs need not be HD for all $\mathrm{X} \in \{\mathrm{B}, \mathrm{C}, \mathrm{W}\}$. Of special interest is our result on weak automata, whose properties typically agree with these of automata on finite words. Here, while all SD-NFWs are HD, this is not the case for SD-NWWs.

Expressive power. It is known that for all $\mathrm{X} \in \{\mathrm{B}, \mathrm{C}, \mathrm{W}\}$, HD-NXWs are as expressive as DXWs. We extend this result to semantic determinism and show that while SD-NXWs need not be HD, they are not more expressive, thus SD-NXWs are as expressive as DXWs. Since an SD-NXW need not be HD, this extends the known frontier of nondeterministic Büchi and weak automata that are not more expressive than their deterministic counterpart.

Deciding the nondeterminism level of an automaton. It is already known that deciding the HDness of a given NXW, for $X \in \{B, C, W\}$, can be done in polynomial time [AKL10, KS15, BK18]. On the other hand, deciding whether a given NCW is DBP is NP-complete [KM18]. We complete the picture in three directions. First, we show that NP-completeness of deciding DBPness applies also to NBWs. Second, we show that in both cases, hardness applies even when the given automaton is HD. Thus, while it took the community some time to get convinced that not all HD automata are DBP, in fact it is NP-complete to decide whether a given HD-NBW or HD-NCW is DBP. Third, we study also the problem of deciding whether a given NXW is SD, and show that it is PSPACE-complete. Note that our results imply that the nondeterminism hierarchy is not monotone with respect to complexity: deciding DBPness, which is closest to determinism, is NP-complete, then HDness can be checked in polynomial time, and finally SDness is PSPACE-complete. Also, as PSPACE-hardness of checking SDness applies already to NWWs, we get another, even more surprising, difference between weak automata and automata on finite words. Indeed, for NFWs, all the three levels of nondeterminism coincide and SDness can be checked in polynomial time.

Approximated determinization by pruning. We say that a nondeterministic automaton $\mathcal{A}$ is *almost-DBP* if $\mathcal{A}$ embodies a deterministic automaton $\mathcal{A}'$ such that the probability of a random word to be in $L(\mathcal{A}) \setminus L(\mathcal{A}')$ is 0. Clearly, if $\mathcal{A}$ is DBP, then it is almost-DBP. Recall that DBPness is associated with the ability of an online algorithm to perform as good as an offline one. A typical analysis of the performance of an online algorithm compares its performance with that of an offline algorithm. The notion of almost-DBPness captures cases where the online algorithm performs, with probability 1, as good as the offline algorithm. From the point of view of formal language theory, almost-DBPness captures the ability to approximate from below the language of automata that need not be DBP. We study the almost-DBPness of HD and SD automata. Our results imply that the nondeterminism hierarchy "almost collapses": We show that for Büchi (and hence also weak) automata, semantic determinism implies almost-DBPness, thus every SD-NBW is almost-DBP. Then, for co-Büchi automata, semantic determinism is not enough, and we need HDness. Thus, there is an SD-NCW that is not almost-DBP, yet all HD-NCWs are almost-DBP.

Reasoning about Markov decision processes. Another application in which nondeterminism is problematic is reasoning about probabilistic systems. Technically, such a reasoning involves the product of a Markov decision processes that models the system with an automaton for the desired behavior. When the automaton is nondeterministic, the product need not reflect the probability in which the system satisfies the behavior. A nondeterministic automaton is *good-for-MDPs* (GFM) if its product with Markov decision processes maintains the probability of acceptance. Thus, GFM automata can replace deterministic automata when reasoning about probabilistic systems [HPS$^+$20, STZ22]. We study the relation between semantic determinism and GFMness. We show that all GFM automata are almost-DBP. On the other hand, semantic determinism implies GFMness only in Büchi automata (that is, all SD-NBWs are GFM), and a GFM automaton need not be SD, even for weak automata.

2. Preliminaries

2.1. **Automata.** 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 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*.

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, $\mathcal{A}^q = \langle \Sigma, Q, q, \delta, \alpha \rangle$. Two states $q, s \in Q$ are *equivalent*, denoted $q \sim_{\mathcal{A}} s$, if $L(\mathcal{A}^q) = L(\mathcal{A}^s)$.

The acceptance condition α determines which runs are "good". We consider here the *Büchi* and *co-Büchi* acceptance conditions, where $\alpha \subseteq Q$ is a subset of states. We use the terms *α-states* and *$\bar{\alpha}$-states* to refer to states in α and in $Q \setminus \alpha$, respectively. For a run r, let $inf(r) \subseteq Q$ be the set of states that r traverses infinitely often. Thus, $inf(r) = \{q \in Q : q = r_i \text{ for infinitely many } i\text{'s}\}$. A run r of a Büchi automaton is *accepting* iff it visits states in α infinitely often, thus $inf(r) \cap \alpha \neq \emptyset$. Dually, a run r of a co-Büchi automaton is accepting iff it visits states in α only finitely often, thus $inf(r) \cap \alpha = \emptyset$. A run that is not accepting is *rejecting*. Note that 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 directed graph $G = \langle V, E \rangle$. A *strongly connected set* in G (SCS, for short) is a set $C \subseteq V$ such that for every two vertices $v, v' \in C$, there is a path from v to v'. A SCS is *maximal* if it is maximal w.r.t containment, that is, for every non-empty set $C' \subseteq V \setminus C$, it holds that $C \cup C'$ is not a SCS. The *maximal strongly connected sets* are also termed *strongly connected components* (SCCs, for short). The *SCC graph of G* is the graph defined over the SCCs of G, where there is an edge from an SCC C to another SCC C' iff there are two vertices $v \in C$ and $v' \in C'$ with $\langle v, v' \rangle \in E$. A SCC is *ergodic* iff it has no outgoing edges in the SCC graph.

An automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ induces a directed graph $G_{\mathcal{A}} = \langle Q, E \rangle$, where $\langle q, q' \rangle \in E$ iff there is a letter $\sigma \in \Sigma$ such that $\langle q, \sigma, q' \rangle \in \Delta$. The SCSs and SCCs of $\mathcal{A}$ are those of $G_{\mathcal{A}}$.

The *α-free SCCs* of $\mathcal{A}$ are the SCCs of $\mathcal{A}$ that do not contain states from α.

A Büchi automaton $\mathcal{A}$ is *weak* [MSS88] if for each SCC C in $G_{\mathcal{A}}$, either $C \subseteq \alpha$ (in which case we say that C is an accepting SCC) or $C \cap \alpha = \emptyset$ (in which case we say that C is a rejecting SCC). Note that a weak automaton can be viewed as both a Büchi and a co-Büchi automaton, as a run of $\mathcal{A}$ visits α infinitely often, iff it gets trapped in an accepting SCC, iff it visits states in $Q \setminus \alpha$ only finitely often.

We denote the different classes of word automata by three-letter acronyms in $\{\text{D, N}\} \times \{\text{F, B, C, W}\} \times \{\text{W}\}$. The first letter stands for the branching mode of the automaton (deterministic or nondeterministic), and the second for the acceptance condition type (finite, Büchi, co-Büchi or weak). For example, NBWs are nondeterministic Büchi word automata.

2.2. **Probability.** Consider the probability space $(\Sigma^\omega, \mathbb{P})$ where each word $w = \sigma_1 \cdot \sigma_2 \cdot \sigma_3 \cdots \in \Sigma^\omega$ is drawn by taking the σ_i's to be independent and identically distributed Unif(Σ). Thus. for all positions $i \geq 1$ and letters $\sigma \in \Sigma$, the probability that σ_i is σ is $\frac{1}{|\Sigma|}$. Let $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ be a deterministic automaton, and let $G_{\mathcal{A}} = \langle Q, E \rangle$ be its induced directed graph. A *random walk on* $\mathcal{A}$, is a random walk on $G_{\mathcal{A}}$ with the probability matrix $P(q,p) = \frac{|\{\sigma \in \Sigma : \langle q,\sigma,p \rangle \in \Delta\}|}{|\Sigma|}$. It is not hard to see that $\mathbb{P}(L(\mathcal{A}))$ is precisely the probability that a random walk on $\mathcal{A}$ is an accepting run. Note that with probability 1, a random walk on $\mathcal{A}$ reaches an ergodic SCC $C \subseteq Q$, where it visits all states infinitely often. It follows that $\mathbb{P}(L(\mathcal{A}))$ equals the probability that a random walk on $\mathcal{A}$ reaches an ergodic accepting SCC.

2.3. **Automata with Some Nondeterminism.** Consider a nondeterministic automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. The automaton $\mathcal{A}$ is *determinizable by pruning* (DBP) if we can remove some of the transitions of $\mathcal{A}$ and get a deterministic automaton $\mathcal{A}'$ that recognizes $L(\mathcal{A})$. We then say that $\mathcal{A}'$ is a *deterministic pruning* of $\mathcal{A}$.

The automaton $\mathcal{A}$ is *history deterministic* (*HD*, for short) if its nondeterminism can be resolved based on the past, thus on the prefix of the input word read so far. Formally, $\mathcal{A}$ is *HD* if there exists a *strategy* $f : \Sigma^* \to Q$ such that the following hold:

(1) The strategy f is consistent with the transition function. That is, $f(\epsilon) = q_0$, and for every finite word $u \in \Sigma^*$ and letter $\sigma \in \Sigma$, we have that $\langle f(u), \sigma, f(u \cdot \sigma) \rangle \in \Delta$.
(2) Following f causes $\mathcal{A}$ to accept all the words in its language. That is, for every infinite word $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$, if $w \in L(\mathcal{A})$, then the run $f(w[1,0]), f(w[1,1]), f(w[1,2]), \ldots$, which we denote by $f(w)$, is an accepting run of $\mathcal{A}$ on w.

We say that the strategy f *witnesses* $\mathcal{A}$'s HDness. Note that a strategy f need not use all the states or transitions of $\mathcal{A}$. In particular, a state q of $\mathcal{A}$ may be such that $\mathcal{A}^q$ is not HD. We say that a state q of $\mathcal{A}$ is *HD* if $\mathcal{A}^q$ is HD.

The automaton $\mathcal{A}$ is *semantically deterministic* (SD, for short) if different nondeterministic choices in $\mathcal{A}$ lead to equivalent states. Thus, for every state $q \in Q$ and letter $\sigma \in \Sigma$, all the σ-successors of q are equivalent: for every two states $s, s' \in \delta(q, \sigma)$, we have that $s \sim_{\mathcal{A}} s'$.

Note that every deterministic automaton $\mathcal{A}$ is DBP (with $\mathcal{A}$ being the deterministic pruning of itself). Also, every nondeterministic DBP automaton is HD. Indeed, the deterministic pruning of $\mathcal{A}$ induces a witness strategy. Finally, if a state q of an HD automaton has a σ-successor s that does not accept all the words w such that $\sigma \cdot w$ is in $L(\mathcal{A}^q)$, then the σ-transition from q to s can be removed. Moreover, by [HKR02, KS15, BK18], the detection

of such transitions can be done in polynomial time. Hence, we can assume that every HD automaton, and hence also every DBP automaton, is SD.[3]

For $X \in \{B, C, W\}$, we use SD-NXW, HD-NXW, and DBP-NXW to refer to the different classes of NXWs. For example, SD-NBWs are nondeterministic Büchi automata that are semantically-deterministic.

3. The Syntactic and Semantic Hierarchies

In this section we study syntactic and semantic hierarchies induced by the different levels of nondeterminism. We start with the syntactic hierarchy. For two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ of automata, we use $\mathcal{C}_1 \preceq \mathcal{C}_2$ to indicate that every automaton in $\mathcal{C}_1$ is also in $\mathcal{C}_2$. Accordingly, $\mathcal{C}_1 \prec \mathcal{C}_2$ if $\mathcal{C}_1 \preceq \mathcal{C}_2$ yet there are automata in $\mathcal{C}_2$ that are not in $\mathcal{C}_1$. For example, clearly, DFW $\prec$ NFW. We first show that the nondeterminism hierarchy is strict, except for all HD-NWWs being DBP. The latter is not surprising, as all HD-NFWs are DBP, and weak automata share many properties with automata on finite words. On the other hand, unlike the case of finite words, we show that not all SD-NWWs are HD. In fact the result holds already for NWWs that accept co-safety languages, namely all whose states except for an accepting sink are rejecting.

Theorem 3.1. [Syntactic Hierarchy] *For* $X \in \{B, C, W\}$*, we have that* DXW $\prec$ DBP-NXW $\preceq$ HD-NXW $\prec$ SD-NXW $\prec$ NXW*. For* $X \in \{B, C\}$*, the second inequality is strict.*

Proof. By definition, each class is a special case of the one to its right. We prove strictness. It is easy to see that the first and last strict inequalities hold. For the first strict inequality, for all $X \in \{B, C, W\}$, consider a nonempty DXW $\mathcal{A}$, and obtain an NXW $\mathcal{B}$ from $\mathcal{A}$ by duplicating some state and the transitions to it. Then, $\mathcal{B}$ is a DBP-NXW that is not a DXW. For the last inequality, consider an SD-NXW $\mathcal{A}$, and let σ be a letter such that $\mathcal{A}$ accepts some word that starts with σ. Consider the NXW $\mathcal{C}$ obtained from $\mathcal{A}$ by adding a σ-transition from $\mathcal{A}$'s initial state to a new rejecting sink. Then, as at least one σ-successor of the initial state of $\mathcal{A}$ is not empty, we have that $\mathcal{C}$ is an NXW that is not an SD-NXW.

The relation between DBPness and HDness has already been studied. It is shown in [BKKS13] that HD-NXW need not be DBP for $X \in \{B, C\}$, and shown in [BKS17] that HD-NWW are DBP.

It is left to relate HDness and SDness. Consider the NWW $\mathcal{W}$ in Figure 1. It is not hard to check that $\mathcal{W}$ is weak, and note that it is SD, as all its states recognize the language $\{a, b\}^\omega$. Yet $\mathcal{W}$ is not HD, as every strategy has a word with which it does not reach q_{acc} – a word that forces each visit in q_a and q_b to be followed by a visit in q_0.

Hence, HD-NWW $\prec$ SD-NWW. As weak automata are a special case of Büchi and co-Büchi, strictness for them follows. □

We continue to the semantic hierarchy, where we study the expressive power of the different classes. Now, for two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ of automata, we say that $\mathcal{C}_1$ is less expressive than $\mathcal{C}_2$, denoted $\mathcal{C}_1 \leq \mathcal{C}_2$, if every automaton in $\mathcal{C}_1$ has an equivalent automaton in $\mathcal{C}_2$. Accordingly, $\mathcal{C}_1 = \mathcal{C}_2$ if $\mathcal{C}_1 \leq \mathcal{C}_2$ and $\mathcal{C}_2 \leq \mathcal{C}_1$, and $\mathcal{C}_1 < \mathcal{C}_2$ if $\mathcal{C}_1 \leq \mathcal{C}_2$ yet $\mathcal{C}_2 \neq \mathcal{C}_1$. Since NCW=DCW, we expect the hierarchy to be strict only in the cases of Büchi and weak automata. As we now show, however, semantically deterministic automata are not more expressive than deterministic ones also in the case of Büchi and weak automata.

[3] Using the terminology of [KS15], we can assume that the *residual languages* of all the σ-successors of each state in an HD automaton are equivalent.

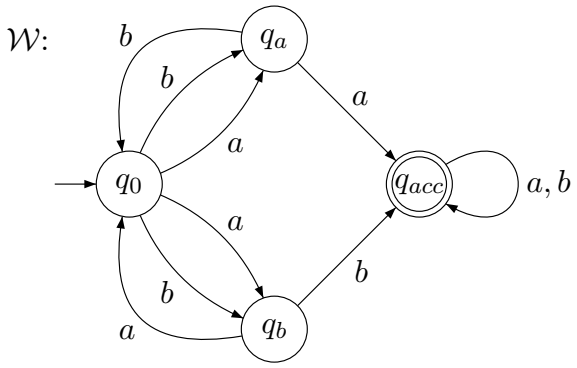

Figure 1: An SD-NWW that is not HD.

Theorem 3.2. [Semantic Hierarchy] *For $X \in \{B, W\}$, we have that* DXW = DBP-NXW = HD-NXW = SD-NXW < NXW.

Proof. In [KSV06, KS15], the authors suggest variants of the subset construction that determinize HD-NBWs. As we argue below, the construction in [KS15] is correct also when applied to SD-NBWs. Moreover, it preserves weakness. Thus, DBW=SD-NBW and DWW=SD-NWW. Also, the last inequality follows from the fact DBW<NBW and DWW<NWW [Lan69].

Given an NBW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, the DBW $\mathcal{A}' = \langle \Sigma, Q', q'_0, \delta', \alpha' \rangle$ generated in [KS15][4] is such that $Q' = 2^Q$, $q'_0 = \{q_0\}$, $\alpha' = \{S \in 2^Q : S \subseteq \alpha\}$, and the transition function δ' is defined for every subset $S \in 2^Q$ 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 the standard subset construction, except that whenever a constructed set contains a state in α, we leave in the set only states in α. Accordingly, every reachable state $S \in Q'$ contains only α-states of $\mathcal{A}$ or only $\bar{\alpha}$-states of $\mathcal{A}$. Note that as $\mathcal{A}$ is SD, then for every two states $q, q' \in Q$, letter $\sigma \in \Sigma$, and transitions $\langle q, \sigma, s \rangle, \langle q', \sigma, s' \rangle \in \Delta$, if $q \sim_{\mathcal{A}} q'$, then $s \sim_{\mathcal{A}} s'$. Consequently, every reachable state S of $\mathcal{A}'$ consists of $\sim_{\mathcal{A}}$-equivalent states. As we formally prove in Appendix A, these properties guarantee that indeed $L(\mathcal{A}') = L(\mathcal{A})$ and that weakness of $\mathcal{A}$ is maintained in $\mathcal{A}'$. □

4. Deciding the Nondeterminism Level of an Automaton

In this section we study the complexity of the problem of deciding the nondeterminism level of a given automaton. Note that we refer here to the syntactic class (e.g., deciding whether a given NBW is HD) and not to the semantic one (e.g., deciding whether a given NBW has an equivalent HD-NBW). Indeed, by Theorem 3.2, the latter boils down to deciding whether the language of a given NXW, for $\mathrm{X} \in \{\mathrm{B}, \mathrm{C}, \mathrm{W}\}$, can be recognized by a DXW, which is well known: the answer is always "yes" for an NCW, and the problem is PSPACE-complete for NBWs and NWWs [KV05a].[5]

[4]The construction in [KS15] assumes automata with transition-based acceptance, and (regardless of this) is slightly different: when α is visited, $\mathcal{A}'$ continues with a single state from the set of successors. However, the key point remains the same: $\mathcal{A}$ being SD enables $\mathcal{A}'$ to maintain only subsets of states, rather than Safra trees, which makes determinization much easier.

[5]The proof in [KV05a] is for NBWs, yet the arguments there apply also for weak automata.

Our results are summarized in Table 1. The entries there describe the complexity of deciding whether a given NXW belongs to a certain nondeterministic level (for example, the upper left cell describes the complexity of deciding whether an NBW is DBP). In fact, the results described are valid already when the given NXW is known to be one level above the questioned one with only two exceptions – deciding the DBPness of an HD-NCW, whose NP-hardness is proved in Theorem 4.6, and deciding whether a given NWW is DBP, which is PTIME in general, and is $O(1)$ when the given NWW is HD, in which case the answer is always "yes".

	DBP	**HD**	**SD**
NBW	NP-complete Theorem 4.2	PTIME [BK18]	PSPACE-complete Theorem 4.1
NCW	NP-complete [KM18]	PTIME [KS15]	PSPACE-complete Theorem 4.1
NWW	PTIME [KS15, BK18]	PTIME [KS15, BK18]	PSPACE-complete Theorem 4.1

Table 1: The complexity of deciding the nondeterminism level of an NXW, for $\mathrm{X} \in \{\mathrm{B}, \mathrm{C}, \mathrm{W}\}$. The results are valid also in the case the given NXW is one level to the right in the nondeterminism hierarchy, with the exception of deciding the DBPness of an HD-NCW, whose NP-hardness is proved in Theorem 4.6, and deciding the DBPness of an HD-NWW, where the answer is always positive [BKS17].

We start with the complexity of deciding semantic determinism.

Theorem 4.1. *Deciding whether an NXW is semantically deterministic is PSPACE-complete, for* $\mathrm{X} \in \{\mathrm{B}, \mathrm{C}, \mathrm{W}\}$.

Proof. Membership in PSPACE is easy, as we check SDness by polynomially many checks of language equivalence. Formally, given an NXW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, a PSPACE algorithm goes over all states $q \in Q$, letters σ, and σ-successors s and s' of q, and checks that $s \sim_{\mathcal{A}} s'$. Since language equivalence can be checked in PSPACE [SVW87] and there are polynomially many checks to perform, we are done.

Proving PSPACE-hardness, we do a reduction from polynomial-space Turing machines. Given a Turing machine T with $|T|$ many transitions and space complexity $s : \mathbb{N} \to \mathbb{N}$, we construct in time polynomial in $|T|$ and $s(0)$, an NWW $\mathcal{A}$ of size linear in T and $s(0)$, such that $\mathcal{A}$ is SD iff T accepts the empty tape[6]. Clearly, this implies a lower bound also for NBWs and NCWs. Let $n_0 = s(0)$. Thus, each configuration in the computation of T on the empty tape uses at most n_0 cells.

[6]This is sufficient, as one can define a generic reduction from every language L in PSPACE as follows. Let T_L be a Turing machine that decides L in polynomial space $f(n)$. On input w for the reduction, the reduction considers the machine T_w that on every input, first erases the tape, writes w on its tape, and then runs as T_L on w. Then, the reduction outputs an automaton $\mathcal{A}$, such that T_w accepts the empty tape iff $\mathcal{A}$ is SD. Note that the space complexity of T_w is $s(n) = \max(n, f(|w|))$, and that w is in L iff T_w accepts the empty tape. Since $\mathcal{A}$ is constructed in time polynomial in $s(0) = f(|w|)$ and $|T_w| = \mathrm{poly}(|w|)$, it follows that the reduction is polynomial in $|w|$.

We assume that T halts from all configurations (that is, not just from these reachable from an initial configuration of T); indeed, by adding to T a step-counter that uses polynomial-space, one can transform a polynomial-space Turing machine that need not halt from all configurations to one that does halt.

We also assume, without loss of generality, that once T reaches a final (accepting or rejecting) state, it erases the tape, moves with its reading head to the leftmost cell, and moves to the initial state. Thus, all computations of T are infinite and after visiting a final configuration for the first time, they consists of repeating the same finite computation on the empty tape that uses n_0 tape cells.

We define $\mathcal{A}$ so that it accepts a word w iff

- (C1) w is not a suffix of an encoding of a legal computation of T that uses at most n_0 cells, or
- (C2) w includes an encoding of an accepting configuration of T that uses at most n_0 cells.

It is not hard to see that if T accepts the empty tape, then $\mathcal{A}$ is universal (that is, accepts all words). Indeed, each word w is either not a suffix of an encoding of a legal computation of T that uses at most n_0 cells, in which case w is accepted thanks to C1, or w is such a suffix, in which case, since we assume that T halts from any configuration, the encoded computation eventually reaches a final configuration and, by the definition of T, continues by an encoding of the computation of T on the empty tape. Thus, w eventually includes the encoding of the computation of T on the empty tape. In particular, since T accepts the empty tape, w includes an encoding of an accepting configuration that uses at most n_0 tape cells, and so w is accepted thanks to C2.

Also, if T rejects the empty tape, then $\mathcal{A}$ rejects the word w_ε that encodes the computation of T on the empty tape. Indeed, C1 is not satisfied, and since by definition of T, the infinite computation of T on the empty tape is just the same finite rejecting computation of T on the empty tape repeated, then w_ε clearly does not include an encoding of an accepting configuration, and so C2 is not satisfied too.

In order to define $\mathcal{A}$ so that it is SD iff T accepts the empty tape, we define all its states to be universal iff T accepts the empty tape. Intuitively, we do it by letting $\mathcal{A}$ guess and check the existence of an infix that witnesses satisfaction of C1 or C2, and also let it, at each point of its operation, go back to the initial state, where it can guess again. Recall that $\mathcal{A}$ is universal when T accepts the empty tape. That is, for every infinite word w, all the suffixes of w satisfy C1 or C2. Thus, $\mathcal{A}$ making a bad guess on w does not prevent it from later branching into an accepting run.

We now describe the operation of $\mathcal{A}$ in more detail (see Figure 2). In its initial state, $\mathcal{A}$ guesses which of C1 and C2 is satisfied. In case $\mathcal{A}$ guesses that C1 is satisfied, it guesses the place in which w includes a violation of the encoding. As we detail in Appendix B, this amounts to guessing whether the violation is in the encoding of a single configuration encoded in w, or whether the violation is of the transition function of T. For the second type of violations, $\mathcal{A}$ may guess, in each step, that the next three letters encode a position in a configuration and the letter to come n_0 letters later, namely at the same position in the successive configuration, is different from the one that should appear in a legal encoding of two successive configurations. If a violation is detected, $\mathcal{A}$ moves to an accepting sink. Otherwise, $\mathcal{A}$ returns to the initial state and w gets another chance to be accepted.

In case $\mathcal{A}$ guesses that C2 is satisfied, it essentially waits to see a letter that encodes the accepting state of T, and when such a letter arrives it moves to the accepting sink.

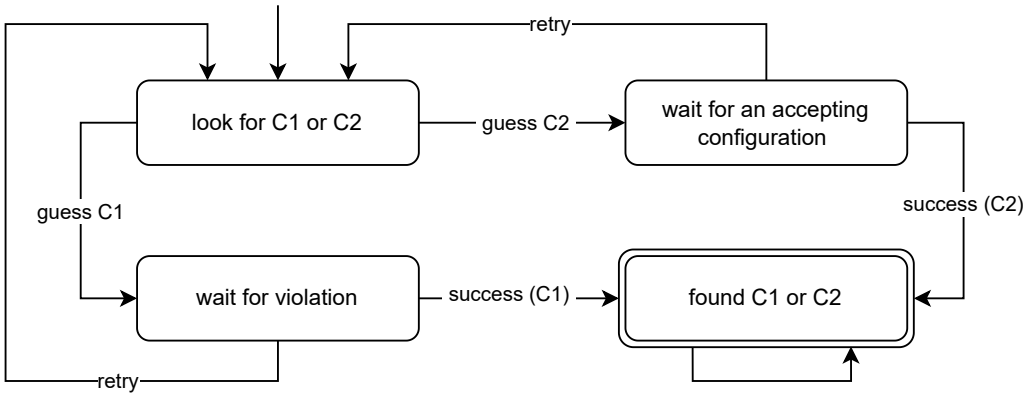

Figure 2: The structure of the NWW constructed in Theorem 4.1.

Also, whenever $\mathcal{A}$ waits to witness some behavior, namely, waits for a position where there is a violation of the encoding, or waits for an encoding of an accepting configuration of T that uses n_0 cells, it may nondeterministically, upon reading the next letter, return to the initial state. It is not hard to see that $\mathcal{A}$ can be defined in size linear in $|T|$ and n_0. As the only accepting states of $\mathcal{A}$ is the accepting sink, it is clearly weak, and in fact describes a co-safety language.

We prove that T accepts the empty tape iff $\mathcal{A}$ is SD. First, if T rejects the empty tape, then $\mathcal{A}$ is not SD. To see this, consider the word w_ε that encodes the computation of T on the empty tape, and let w'_ε be a word that is obtained from w_ε by making a single violation in the first letter. Note that $w'_\varepsilon \in L(\mathcal{A})$ since it has a violation. Note also that any proper suffix of w'_ε is also a suffix of w_ε, and hence is not in $L(\mathcal{A})$ as it encodes a suffix of a computation of T that uses at most n_0 tape cells and does not include an encoding of an accepting configuration of T. Consequently, the word w'_ε can be accepted by $\mathcal{A}$ only by guessing a violation that is caused by the first letter. In particular, if we guess incorrectly that there is a violation on the second letter of w'_ε, then that guess fails, and after it we cannot branch to an accepting run. This shows that $\mathcal{A}$ is not SD.

For the other direction, we show that if T accepts the empty tape, then all the states of $\mathcal{A}$ are universal. In particular, all the states of $\mathcal{A}$ are equivalent, which clearly implies that $\mathcal{A}$ is SD. First, as argued above, if T accepts the empty tape, then $\mathcal{A}$ is universal. In addition, by the definition of $\mathcal{A}$, for every infinite word w and for all states q of $\mathcal{A}$ that are not the accepting sink, there is a path from q to the initial state that is labeled by a prefix of w. Thus, the language of all states is universal, and we are done.

Thus, we conclude that T accepts the empty tape iff $\mathcal{A}$ is SD. In Appendix B, we give the full technical details of the construction of $\mathcal{A}$. □

Note that the considerations in the proof of Theorem 4.1 can be used in order to prove that the universality problem for SD-NBWs is PSPACE-hard [AK23].

Theorem 4.2. *The problem of deciding whether a given NBW or HD-NBW is DBP is NP-complete.*

Proof. For membership in NP, observe we can check that a witness deterministic pruning $\mathcal{A}'$ is equivalent to $\mathcal{A}$ by checking whether $L(\mathcal{A}) \subseteq L(\mathcal{A}')$. Since $\mathcal{A}'$ is deterministic, the latter can be checked in polynomial time. For NP-hardness, we describe a parsimonious

polynomial time reduction from SAT. That is, given a CNF formula φ, we construct an HD-NBW $\mathcal{A}_\varphi$ such that there is a bijection between assignments to the variables of φ and DBWs embodied in $\mathcal{A}_\varphi$, and an assignment satisfies φ iff its corresponding embodied DBW is equivalent to $\mathcal{A}_\varphi$. In particular, φ is satisfiable iff $\mathcal{A}_\varphi$ is DBP.

Consider a SAT instance φ over the variable set $X = \{x_1, \ldots, x_n\}$ and with $m \geq 1$ clauses $C = \{c_1, \ldots, c_m\}$. For $n \geq 1$, let $[n] = \{1, 2, \ldots, n\}$. For a variable $x_k \in X$, let $C^0_k \subseteq C$ be the set of clauses in which x_k appears negatively, and let $C^1_k \subseteq C$ be the set of clauses in which x_k appears positively. For example, if $c_1 = x_1 \vee \neg x_2 \vee x_3$, then c_1 is in C^1_1, C^0_2, and C^1_3. Assume that all clauses depend on at least two different variables (that is, no clause is a tautology or forces an assignment to a single variable). Let $\Sigma_{n,m} = X \cup C$, and let

$$R_{n,m} = (X \cdot C)^* \cdot \{x_1 \cdot c_j \cdot x_2 \cdot c_j \cdots x_n \cdot c_j : j \in [m]\} \subseteq \Sigma^*_{n,m}.$$

We construct an HD-NBW $\mathcal{A}_\varphi$ that recognizes $L_{n,m} = (R_{n,m})^\omega$, and is DBP iff φ is satisfiable.

Let $D_{n,m}$ be a DFW that recognizes $R_{n,m}$ with $O(n \cdot m)$ states, a single accepting state p, and an initial state q_0 that is visited only once in all runs. For example, we can define $D_{n,m} = \langle \Sigma_{n,m}, Q_{n,m}, q_0, \delta_{n,m}, \{p\} \rangle$ as follows: from q_0, the DFW expects to read only words in $(X \cdot C)^*$ – upon a violation of this pattern, it goes to a rejecting sink. Now, if the pattern is respected, then with $X \setminus \{x_1\}$, the DFW goes to two states where it loops with $C \cdot (X \setminus \{x_1\})$ and, upon reading x_1 from all states that expect to see letters in X, it branches with each c_j, for all $j \in [m]$, to a path where it hopes to detect an $x_2 \cdot c_j \cdots x_n \cdot c_j$ suffix. If the detection is completed successfully, it goes to the accepting state p. Otherwise, it returns to the two-state loop.

Now, we define $\mathcal{A}_\varphi = \langle \Sigma_{n,m}, Q_\varphi, p, \delta_\varphi, \{q_0\} \rangle$, where $Q_\varphi = Q_{n,m} \cup \{q^i_k : (i,k) \in \{0,1\} \times [n]\}$. The idea behind $\mathcal{A}_\varphi$ is as follows. From state p (that is, the accepting state of $D_{n,m}$, which is now the initial state of $\mathcal{A}_\varphi$), the NBW $\mathcal{A}_\varphi$ expects to read a letter in X. When it reads x_k, for $1 \leq k \leq n$, it nondeterministically branches to the states q^0_k and q^1_k. Intuitively, when it branches to q^0_k, it guesses that the clause that comes next is one that is satisfied when $x_k = 0$, namely a clause in C^0_k. Likewise, when it branches to q^1_k, it guesses that the clause that comes next is one that is satisfied when $x_k = 1$, namely a clause in C^1_k. When the guess is successful, $\mathcal{A}_\varphi$ moves to the α-state q_0. When the guess is not successful, it returns to p. Implementing the above intuition, transitions from the states $Q_{n,m} \setminus \{p\}$ are inherited from $D_{n,m}$, and transitions from the states in $\{q^i_k : (i,k) \in \{0,1\} \times [n]\} \cup \{p\}$ are defined as follows (see also Figure 3).

- For all $k \in [n]$, we have that $\delta_\varphi(p, x_k) = \{q^0_k, q^1_k\}$.
- For all $k \in [n]$, $i \in \{0,1\}$, and $j \in [m]$, if $c_j \in C^i_k$, then $\delta_\varphi(q^i_k, c_j) = \{q_0\}$. Otherwise, $\delta_\varphi(q^i_k, c_j) = \{p\}$. For example, if $c_1 = x_1 \vee \neg x_2 \vee x_3$, then $\delta_\varphi(q^0_2, c_1) = \{q_0\}$ and $\delta_\varphi(q^1_2, c_1) = \{p\}$.

Note that p is the only nondeterministic state of $\mathcal{A}_\varphi$ and that for every deterministic pruning of $\mathcal{A}_\varphi$, all the words in $(X \cdot C)^\omega$ have an infinite run in the pruned automaton. This run, however, may eventually loop in $\{p\} \cup \{q^0_k, q^1_k : k \in [n]\}$. Note also, that for readability purposes, the automaton $\mathcal{A}_\varphi$ is not total. Specifically, the states of $\mathcal{A}_\varphi$ are partitioned into states that expect to see letters in X and states that expect to see letters in C. In particular, all infinite paths in $\mathcal{A}_\varphi$ are labeled by words in $(X \cdot C)^\omega$. Thus, when defining an HD strategy g for $\mathcal{A}_\varphi$, we only need to define g on prefixes in $(X \cdot C)^* \cup (X \cdot C)^* \cdot X$.

In the following propositions, we prove that $\mathcal{A}_\varphi$ is an HD NBW recognizing $L_{n,m}$, and that $\mathcal{A}_\varphi$ is DBP iff φ is satisfiable. □

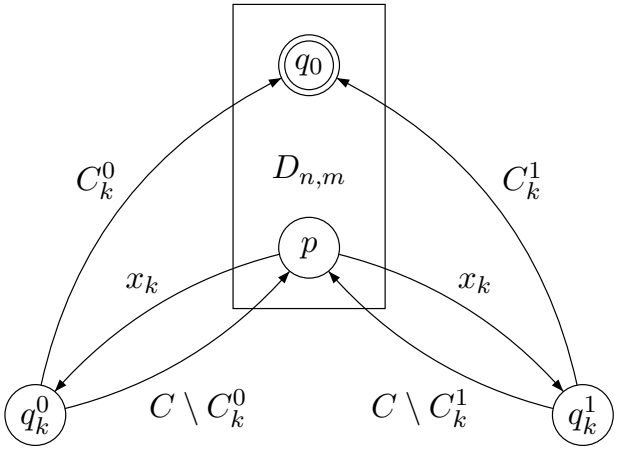

Figure 3: The transitions to and from the states q_k^0 and q_k^1 in $\mathcal{A}_\varphi$.

Proposition 4.3. $L(\mathcal{A}_\varphi) \subseteq L_{n,m}$.

Proof. As already mentioned, all infinite paths of $\mathcal{A}_\varphi$, accepting or rejecting, are labeled by words in $(X \cdot C)^\omega$. Further, any accepting run of $\mathcal{A}_\varphi$ has infinitely many sub-runs that are accepting finite runs of $D_{n,m}$. Let $\infty R_{n,m}$ be the language of infinite words that include infinitely many disjoint finite subwords in $R_{n,m}$. Since $L_{n,m} = (R_{n,m})^\omega = (X \cdot C)^\omega \cap (\infty R_{n,m})$, it follows that $L(\mathcal{A}_\varphi) \subseteq L_{n,m}$. □

Proposition 4.4. *There exists a strategy $g : \Sigma^* \to Q_\varphi$ for $\mathcal{A}_\varphi$ that accepts all words in $L_{n,m}$. Formally, for all $w \in L_{n,m}$, the run $g(w) = g(w[1,0]), g(w[1,1]), g(w[1,2]), \ldots$, is an accepting run of $\mathcal{A}_\varphi$ on w.*

Proof. The definition of $L_{n,m}$ is such that when reading a prefix that ends with a subword of the form $x_1 \cdot c_j$, for some $j \in [m]$, then we can guess that the word continues with $x_2 \cdot c_j \cdot x_3 \cdot c_j \cdots x_n \cdot c_j$; thus that c_j is the clause that is going to repeat. Therefore, when we are at state p after reading a word that ended with $x_1 \cdot c_j$, and we read x_2, it is a good HD strategy to move to a state q_2^i such that the assignment $x_2 = i$ satisfies c_j (if such $i \in \{0,1\}$ exists; otherwise the strategy can choose arbitrary between q_2^0 and q_2^1), and if the run gets back to p, the strategy continues with assignments that hope to satisfy c_j, until the run gets to q_0 or another occurrence of x_1 is detected. Note that while it is not guaranteed that for all $k \in [n]$ there is $i \in \{0,1\}$ such that the assignment $x_k = i$ satisfies c_j, it is guaranteed that such an i exists for at least two different k's (we assume that all clauses depend on at least two variables). Thus, even though we a priori miss an opportunity to satisfy c_j with an assignment to x_1, it is guaranteed that there is another $2 \leq k \leq n$ such that c_j can be satisfied by x_k.

We define g inductively as follows. Recall that $\mathcal{A}_\varphi$ is nondeterministic only in the state p, and so in all other states, the strategy g follows the only possible transition. First, for all $k \in [n]$, we define $g(x_k) = q_k^0$. Let $v \in (X \cdot C)^* \cdot X$, be such that g has already been defined on v and let $j \in [m]$. Since $v \notin (X \cdot C)^*$, we have that $g(v) \neq p$ and so $g(v \cdot c_j)$ is uniquely defined. We continue and define g on $u = v \cdot c_j \cdot x_k$, for all $k \in [n]$. If $g(v \cdot c_j) \neq p$, then $g(u)$ is uniquely defined. Otherwise, $g(v \cdot c_j) = p$ and we define $g(u)$ as follows,

- If $k = 1$, then we define $g(u) = q_1^0$.

- If $k > 1$ and x_k participates in c_j, then we define $g(u) = q_k^i$, where $i \in \{0, 1\}$ is such that $c_j \in C_k^i$. Note that since we assume that c_j is not a tautology, then i is uniquely defined.
- If $k > 1$ and x_k does not participate in c_j, then the value of c_j is not affected by the assignment to x_k, and in that case we define $g(u) = q_k^0$.

The reason for the distinction between the cases $k = 1$ and $k > 1$ is that when we see a finite word that ended with $c_j \cdot x_1$, then there is no special reason to hope that the next letter is going to be c_j. This is in contrast, for example, to the case we have seen a word that ends with $c_{j'} \cdot x_1 \cdot c_j \cdot x_2$, where it is worthwhile to guess we are about to see c_j as the next letter.

By the definition of g, it is consistent with Δ_φ. In Appendix C we formally prove that g is a winning HD strategy for $\mathcal{A}_\varphi$. Namely, that for all $w \in L_{n,m}$, the run $g(w)$ on w, generated by g is accepting. □

We now examine the relation between prunings of $\mathcal{A}_\varphi$ and assignments to φ, and conclude the proof by proving the following:

Proposition 4.5. *The formula φ is satisfiable iff the HD-NBW $\mathcal{A}_\varphi$ is DBP.*

Proof. Consider an assignment $i_1, \ldots, i_n \in \{0, 1\}$, for X. I.e., $x_k = i_k$ for all $k \in [n]$. A possible memoryless HD strategy is to always move from p to $q_k^{i_k}$ when reading x_k. This describes a one to one correspondence between assignments for X and prunings of $\mathcal{A}_\varphi$. Assume that the assignment $i_k \in \{0, 1\}$, for $k \in [n]$, satisfies φ. Then, the corresponding pruning recognizes $L_{n,m}$. Indeed, instead of trying to satisfy the last read clause c_j, we may ignore this extra information, and rely on the fact that one of the assignments $x_k = i_k$ is going to satisfy c_j. In other words, the satisfiability of φ allows us to ignore the history and still accept all words in $L_{n,m}$, which makes $\mathcal{A}_\varphi$ DBP. On the other hand, if an assignment does not satisfy some clause c_j, then the corresponding pruning fails to accept the word $(x_1 \cdot c_j \cdots x_n \cdot c_j)^\omega$, which shows that if φ is not satisfiable then $\mathcal{A}_\varphi$ is not DBP. In Appendix C we formally prove that there is a one to one correspondence between prunings of $\mathcal{A}_\varphi$ and assignments to φ, and that an assignment satisfies φ iff the corresponding pruning recognizes $L_{n,m}$, implying Proposition 4.5. □

We continue to co-Büchi automata. In [KM18], the authors prove that deciding the DBPness of a given NCW is NP-complete. Below we extend their proof to HD-NCWs.

Theorem 4.6. *The problem of deciding whether a given HD-NCW is DBP is NP-complete.*

Proof. The upper bound follows by Lemma 18 in [KM18] where the authors prove that checking if an automaton is DBP is in NP already for general NCWs. For the lower bound, we analyze the reduction in [KM18] and prove that the NCW constructed there is HD. The reduction is from the *Hamiltonian-cycle* problem: given a connected graph $G = \langle [n], E \rangle$, the reduction outputs an NCW $\mathcal{A}_G$ over the alphabet $[n]$ that is obtained from G by adding self loops to all vertices, labelling the loop at a vertex i by the letter i, and labelling the edges from vertex i to all its neighbors in G by every letter $j \neq i$. Then, the co-Büchi condition requires a run to eventually get stuck at a self-loop[7]. Accordingly, $L(\mathcal{A}_G) = [n]^* \cdot \bigcup_{i \in [n]} i^\omega$.

It is not hard to see that $\mathcal{A}_G$ is HD. Indeed, an HD strategy can decide to which neighbor of i to proceed with a letter $j \neq i$ by following a cycle c that traverses all the vertices of the graph G. Since when we read $j \neq i$ at vertex i we move to a neighbor state, then by

[7] The exact reduction is more complicated and involves an additional letter # that forces each deterministic pruning of $\mathcal{A}_G$ to proceed to the same neighbor of i upon reading a letter $j \neq i$ from the vertex i.

following the cycle c upon reading i^ω, we eventually reach the vertex i and get stuck at the i-labeled loop. Thus, the Hamiltonian-cycle problem is reduced to DBPness of an HD-NCW, and we are done. □

5. Approximated Determinization by Pruning

Consider a nondeterministic automaton $\mathcal{A}$. We say that $\mathcal{A}$ is *almost-DBP* if it can be pruned to a deterministic automaton that retains almost all the words in $L(\mathcal{A})$.[8] Formally, an NXW $\mathcal{A}$ is almost-DBP if there is a DXW $\mathcal{A}'$ embodied in $\mathcal{A}$ such that $\mathbb{P}(L(\mathcal{A}) \setminus L(\mathcal{A}')) = 0$. Thus, while $\mathcal{A}'$ need not accept all the words accepted by $\mathcal{A}$, it rejects only a negligible set of words in $L(\mathcal{A})$. Note that if $\mathcal{A}$ is DBP or $\mathbb{P}(L(\mathcal{A})) = 0$, then $\mathcal{A}$ is almost-DBP. Indeed, if $\mathcal{A}$ is DBP, then the DXW $\mathcal{A}'$ embodies in $\mathcal{A}$ with $L(\mathcal{A}') = L(\mathcal{A})$ is such that $L(\mathcal{A}) \setminus L(\mathcal{A}') = \emptyset$, and if $\mathbb{P}(L(\mathcal{A})) = 0$, then for every DBW $\mathcal{A}'$ embodied in $\mathcal{A}$, we have that $\mathbb{P}(L(\mathcal{A}) \setminus L(\mathcal{A}')) = 0$.

In this section we study approximated determinization by pruning. We first show that, unsurprisingly, almost-DBPness is not a trivial property:

Theorem 5.1. *There is an NWW (and hence, also an NBW and an NCW) that is not almost-DBP.*

Proof. Consider the NWW $\mathcal{A}_1$ in Figure 4. It is not hard to see that $L(\mathcal{A}_1) = \{a, b\}^\omega$, and so $\mathbb{P}(L(\mathcal{A}_1)) = 1$. Moreover, every deterministic pruning of $\mathcal{A}_1$ is such that q_{rej} is reachable from all states, which implies that $\{q_{rej}\}$ is the only ergodic SCC of any pruning. Since $\{q_{rej}\}$ is α-free, it follows that every deterministic pruning of $\mathcal{A}_1$ recognizes a language of measure zero, and hence $\mathcal{A}_1$ is not almost-DBP. As an example, consider the deterministic pruning $\mathcal{A}'_1$ described on the right hand side of Figure 4. The only ergodic SCC of $\mathcal{A}'_1$ is α-free, and as such $\mathbb{P}(L(\mathcal{A}'_1)) = 0$. □

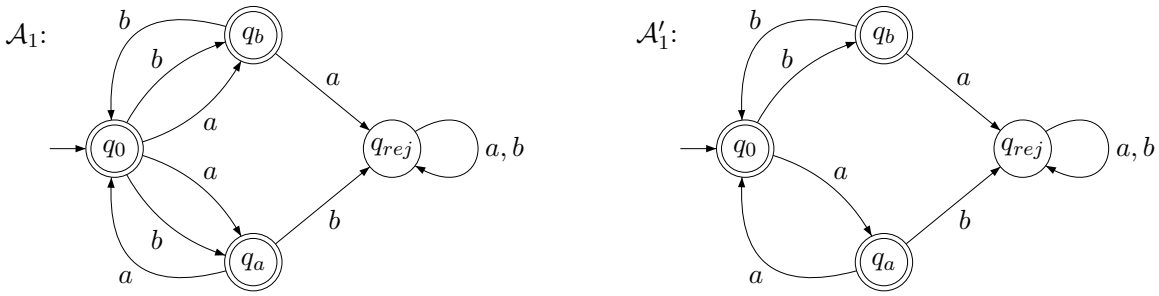

Figure 4: An NWW that is not almost-DBP.

We continue and study the almost-DBPness of HD and SD automata. We show that while for Büchi (and hence also weak) automata, semantic determinism implies almost-DBPness, thus every SD-NBW is almost-DBP, for co-Büchi automata semantic determinism is not enough, and we need HDness. Thus, there is an SD-NCW that is not almost-DBP, yet all HD-NCWs are almost-DBP.

[8] Not to be confused with the definition of almost-DBPness in [AKL10], where it is used to describe automata that are deterministic in their transition function yet may have a set of initial states.

We start with the positive result about Büchi automata. Consider an NBW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. We define a *simple stochastic Büchi game* $\mathcal{G}_\mathcal{A}$ as follows.[9] The game is played between Random and Eve. The set of positions of Random are Q, and the set of positions of Eve are $Q \times \Sigma$. The game starts from position q_0. A round in the game starts at some position $q \in Q$ and proceeds as follows.

(1) Random picks a letter $\sigma \in \Sigma$ uniformly, and the game moves to position (q, σ).
(2) Eve picks a transition $(q, \sigma, p) \in \Delta$, and the game moves to position p.

A probabilistic strategy for Eve is $f : (Q \times \Sigma)^+ \to [0,1]^Q$, where for all histories $x \in (Q \times \Sigma)^*$ and positions of Eve $(q, \sigma) \in Q \times \Sigma$, the function $d = f(x \cdot (q, \sigma)) : Q \to [0,1]$, is a distribution on Q such that $d(p) \neq 0$ implies that $p \in \delta(q, \sigma)$. As usual, we say that a strategy f is *memoryless*, if it depends only on the current position, thus for all histories $x, y \in (Q \times \Sigma)^*$ and positions of Eve $(q, \sigma) \in Q \times \Sigma$, it holds that $f(x \cdot (q, \sigma)) = f(y \cdot (q, \sigma))$. A strategy for Eve is *pure* if for all histories $x \in (Q \times \Sigma)^*$ and positions of Eve $(q, \sigma) \in Q \times \Sigma$, there is a position $p \in \delta(q, \sigma)$ such that $f(x \cdot (q, \sigma))(p) = 1$. When Eve plays according to a strategy f, the outcome of the game can be viewed as a run $r_f = q_f^0, q_f^1, q_f^2, \ldots$ in $\mathcal{A}$, over a random word $w_f \in \Sigma^\omega$. (The word that is generated in a play is independent of the strategy of Eve, but we use the notion w_f to emphasize that we are considering the word that is generated in a play where Eve plays according to f).

Let $Q_{rej} = \{q \in Q : \mathbb{P}(L(\mathcal{A}^q)) = 0\}$. The outcome r_f of the game is *winning* for Eve iff r_f is accepting, or r_f visits Q_{rej}. Note also that for all positions $q \in Q_{rej}$ and $p \in Q$, if p is reachable from q, then $p \in Q_{rej}$. Hence, the winning condition can be defined by the Büchi objective $\alpha \cup Q_{rej}$. Note that r_f is winning for Eve iff $\inf(r_f) \subseteq Q_{rej}$ or $\inf(r_f) \cap \alpha \neq \emptyset$. We say that f is an *almost-sure winning* strategy, if r_f is winning for Eve with probability 1, and Eve *almost-sure wins* in $\mathcal{G}_\mathcal{A}$ if she has an almost-sure winning strategy.

Theorem 5.2. *All SD-NBWs are almost-DBP.*

Proof. Let $\mathcal{A}$ be an SD-NBW, and consider the simple stochastic Büchi game $\mathcal{G}_\mathcal{A}$. We first show that Eve has a probabilistic strategy to win $\mathcal{G}_\mathcal{A}$ with probability 1, even without assuming that $\mathcal{A}$ is semantically deterministic. Consider the probabilistic strategy g where from (q, σ), Eve picks one of the σ-successors of q uniformly at random. Note that this strategy is memoryless, and hence the outcome of the game can be thought as a random walk in $\mathcal{A}$ that starts at q_0 and gives positive probabilities to all transitions. Thus, with probability 1, the run r_g is going to reach an ergodic SCC of $\mathcal{A}$ and visit all its states. If the run r_g reaches an α-free ergodic SCC, then $\inf(r_g) \subseteq Q_{rej}$, and hence r_g is then winning for Eve. Otherwise, r_g reaches a non α-free ergodic SCC, and with probability 1, it visits all the states in that SCC. Thus, with probability 1, we have $\inf(r_g) \cap \alpha \neq 0$, and r_g is winning for Eve. Overall, Eve wins $\mathcal{G}_\mathcal{A}$ with probability 1 when playing according to g.

For a strategy f for Eve, we say that r_f *is correct* if $w_f \in L(\mathcal{A})$ implies that r_f is accepting. Note that $w_f \notin L(\mathcal{A})$ always implies that r_f is rejecting. We show that if $\mathcal{A}$ is SD, then for every almost-sure winning strategy f for Eve, we have that r_f is correct with probability 1. Consider an almost-sure winning strategy f for Eve. Since f is almost-sure winning for Eve, we have that with probability 1, either r_f is an accepting run or $\inf(r_f) \subseteq Q_{rej}$. Thus, it is sufficient to prove that if $\inf(r_f) \subseteq Q_{rej}$, then $w_f \in L(\mathcal{A})$ with probability 0, since otherwise, almost surely r_f is accepting, and in particular is correct.

[9] In [CJH03] these games are called simple $1\frac{1}{2}$ -player games with Büchi winning objectives and almost-sure winning criterion.

By semantic determinism, for all $i \geq 0$, it holds that $w_f \in L(\mathcal{A})$ iff $w_f[i+1,\infty] \in L(\mathcal{A}^{q_f^i})$. Moreover, the suffix $w_f[i+1,\infty]$ is independent of q_f^i, and hence for all $q \in Q$ and $i \geq 0$, the event $w_f[i+1,\infty] \in L(\mathcal{A}^q)$ is independent of q_f^i. Thus, for all $q \in Q$ and $i \geq 0$, it holds that $\mathbb{P}(w_f \in L(\mathcal{A}) \mid q_f^i = q) = \mathbb{P}(w_f[i+1,\infty] \in L(\mathcal{A}^q)) = \mathbb{P}(L(\mathcal{A}^q))$. Hence, by the definition of Q_{rej}, and by the fact that Q_{rej} is finite, we have that $\mathbb{P}(w_f \in L(\mathcal{A}) \mid q_f^i \in Q_{rej}) = 0$ for all $i \geq 0$, and so $\mathbb{P}(w_f \in L(\mathcal{A}) \mid r_f \text{ visits } Q_{rej}) = 0$. Overall, we showed that if f is winning for Eve and $\mathcal{A}$ is SD, then $\mathbb{P}(r_f \text{ is correct}) = 1$.

Hence, by pure memoryless determinacy of simple stochastic parity games [CJH03], we may consider a pure memoryless winning strategy f for Eve in $\mathcal{G}_\mathcal{A}$, and by the above we know that a random walk in the corresponding pruned DBW is correct with probability 1. Formally, since f is pure memoryless, it induces a pruning of $\mathcal{A}$. We denote this pruning by $\mathcal{A}^f$, and think of r_f as a random walk in $\mathcal{A}^f$. As we saw, since f is a winning strategy and $\mathcal{A}$ is SD, we have that r_f is correct with probability 1. Note that $\mathbb{P}(L(\mathcal{A}) \setminus L(\mathcal{A}^f))$, is precisely the probability that a random word w_f is in $L(\mathcal{A})$ but not accepted by $\mathcal{A}^f$. Namely, the probability that r_f is not correct. Hence, $\mathbb{P}(L(\mathcal{A}) \setminus L(\mathcal{A}^f)) = 0$, and $\mathcal{A}$ is almost-DBP. □

We continue to co-Büchi automata and show that unlike the case of Büchi, here semantic determinism does not imply almost-DBPness. Note that the NWW in Figure 4 is not SD.

Theorem 5.3. *There is an SD-NCW that is not almost-DBP.*

Proof. Consider the NCW $\mathcal{A}_2$ in Figure 5. It is not hard to see that $L(\mathcal{A}_2) = \{a,b\}^\omega$, and hence $\mathbb{P}(L(\mathcal{A}_2)) = 1$. In fact all the states q of $\mathcal{A}_2$ have $L(\mathcal{A}_2^q) = \{a,b\}^\omega$, and so it is semantically deterministic.

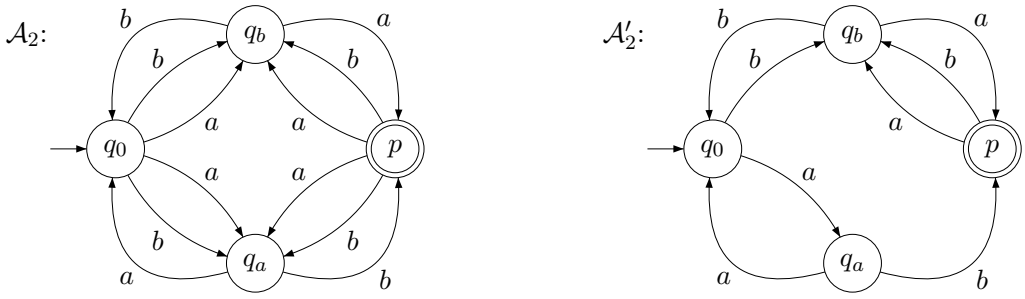

Figure 5: An SD-NCW that is not almost-DBP.

Moreover, every deterministic pruning of $\mathcal{A}_2$ is strongly connected and not α-free. It follows that any deterministic pruning of $\mathcal{A}_2$ recognizes a language of measure zero, and hence $\mathcal{A}_2$ is not almost-DBP. As an example, consider the deterministic pruning $\mathcal{A}'_2$ described on the right hand side of Figure 5. It is easy to see that $\mathcal{A}'_2$ is strongly connected and not α-free, and as such, $\mathbb{P}(L(\mathcal{A}'_2)) = 0$. Thus, $\mathcal{A}_2$ is not almost-DBP. □

Consider a language $L \subseteq \Sigma^\omega$ of infinite words. We say that a finite word $x \in \Sigma^*$ is a *good prefix for* L if $x \cdot \Sigma^\omega \subseteq L$. Then, L is a *co-safety* language if every word in L has a good prefix [AS87]. Let *co-safe*$(L) = \{x \cdot w \in \Sigma^\omega : x \text{ is a good prefix of } L\}$. Clearly, *co-safe*$(L) \subseteq L$. The other direction is not necessarily true. For example, if $L \subseteq \{a,b\}^\omega$ is the set of all words with infinitely many a's, then *co-safe*$(L) = \emptyset$. In fact, *co-safe*$(L) = L$ iff

L is a co-safety language. As we show now, when L is NCW-recognizable, we can relate L and *co-safe*(L) as follows.

Lemma 5.4. *If L is NCW-recognizable, then $\mathbb{P}(L \setminus \textit{co-safe}(L)) = 0$.*

Proof. Consider an NCW-recognizable language L. Since NCW=DCW, there is a DCW $\mathcal{D}$ that recognizes L. Assume without loss of generality that $\mathcal{D}$ has a single state q with $L(\mathcal{D}^q) = \Sigma^\omega$, in particular, $C = \{q\}$ is the only ergodic α-free SCC of $\mathcal{D}$. Then, for every word $w \in \Sigma^\omega$, we have that $w \in \textit{co-safe}(L)$ iff the run r of $\mathcal{D}$ on w reaches C. Hence, the probability that $w \in L \setminus \textit{co-safe}(L)$ equals the probability that $\inf(r)$ is α-free but is not an ergodic SCC of $\mathcal{D}$. Since the later happens with probability 0, we have that $\mathbb{P}(L \setminus \textit{co-safe}(L)) = 0$. □

By Lemma 5.4, pruning an NCW in a way that would make it recognize *co-safe*$(L(\mathcal{A}))$ results in a DCW that approximates $\mathcal{A}$, and thus witnesses that $\mathcal{A}$ is almost-DBP. We now show that for HD-NCWs, such a pruning is possible, and conclude that HD-NCWs are almost-DBP.

Theorem 5.5. *All HD-NCWs are almost-DBP.*

Proof. Let $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ be an HD-NCW. Consider the NCW $\mathcal{A}' = \langle \Sigma, Q, q_0, \delta, \alpha' \rangle$, where $\alpha' = \alpha \cup \{q \in Q : L(\mathcal{A}^q) \neq \Sigma^\omega\}$. We prove that $\mathcal{A}'$ is an HD-NCW with $L(\mathcal{A}') =$ *co-safe*$(L(\mathcal{A}))$. Consider a word $w \in L(\mathcal{A}')$, and let $r = q_0, q_1, q_2, \ldots$ be an accepting run of $\mathcal{A}'$ on w. There exists a prefix $x \in \Sigma^*$ of w such that r reaches some $q \notin \alpha'$ when reading x. Hence $L(\mathcal{A}^q) = \Sigma^\omega$, and so $x \cdot \Sigma^\omega \subseteq L(\mathcal{A})$. That is, x is a good prefix and $w \in \textit{co-safe}(L(\mathcal{A}))$. Thus, $L(\mathcal{A}') \subseteq \textit{co-safe}(L(\mathcal{A}))$.

In order to see that $\mathcal{A}'$ is HD and that *co-safe*$(L(\mathcal{A})) \subseteq L(\mathcal{A}')$, consider an HD strategy f of $\mathcal{A}$. We claim that f is also an HD strategy for $\mathcal{A}'$, and for all words $w \in \textit{co-safe}(L(\mathcal{A}))$, the run r that f generates on w eventually visits only states $q \notin \alpha'$. Since *co-safe*$(L(\mathcal{A})) \subseteq L(\mathcal{A})$, we know that $\inf(r) \cap \alpha = \emptyset$. It is left to prove that r visits only finitely many states $q \in Q$ with $L(\mathcal{A}^q) \neq \Sigma^\omega$.

Since $w \in \textit{co-safe}(L(\mathcal{A}))$, it has a good prefix $x \in \Sigma^*$. Since f is an HD strategy and x is a good prefix, it must be that $L(\mathcal{A}^{f(x)}) = \Sigma^\omega$. Also, since all the extensions of x are also good prefixes, the above holds also for all states that f visits after $f(x)$. Hence, the run r visits states q with $L(\mathcal{A}^q) \neq \Sigma^\omega$ only during the finite prefix x. Thus, $\inf(r) \cap \alpha' = \emptyset$, and we are done.

So, $\mathcal{A}'$ is an HD-NCW with $L(\mathcal{A}') =$ *co-safe*$(L(\mathcal{A}))$. Since *co-safe*$(L(\mathcal{A}))$ is co-safe, it is DWW-recognizable [Sis94]. By [BKS17], HD-NCWs whose language is DWW-realizable are DBP. Let δ' be the restriction of δ to a deterministic transition function such that $\mathcal{D}' = \langle \Sigma, Q, q_0, \delta', \alpha' \rangle$ is a DCW with $L(\mathcal{D}') = L(\mathcal{A}') =$ *co-safe*$(L(\mathcal{A}))$. Consider now the DCW $\mathcal{D} = \langle \Sigma, Q, q_0, \delta', \alpha \rangle$ that is obtained from $\mathcal{D}'$ by replacing α' with α.

It is clear that $\mathcal{D}$ is a pruning of $\mathcal{A}$. Note that, $\alpha \subseteq \alpha'$, and hence *co-safe*$(L(\mathcal{A})) = L(\mathcal{D}') \subseteq L(\mathcal{D})$. That is, $\mathcal{D}$ is a pruning of $\mathcal{A}$ that approximates $L(\mathcal{A})$ up to a negligible set, and $\mathcal{A}$ is almost-DBP. □

6. Nondeterminism and Reasoning about MDPs

A probabilistic open system can be modeled by a *Markov decision process* (MDP, for short). As we define formally below, an MDP describes the behavior of the system when

it interacts with its environment. The behavior is probabilistic: each input provided by the environment induces a probability on the reaction of the system. Reasoning about the behavior of an MDP can proceed by taking its product with a deterministic automaton that describes a desired behavior for the MDP. Indeed, this product contains information on the probability that the interaction results in a behavior accepted by the automaton. When the automaton is nondeterministic, this information is lost, and thus reasoning about MDPs involves determinization. An automaton is said to be *good-for-MDPs* (GFM), if its product with MDPs maintains the probability of acceptance, and can therefore replace deterministic automata when reasoning about stochastic behaviors [HPS$^+$20, STZ22]. In Section 5, we related semantic determinism with almost-DBPness. In this section we relate semantic determinism with GFMness. The terminology used in this section is based on [HPS$^+$20].

6.1. **Good-for-MDP automata.** We first define MDPs and good-for-MDP automata. A Σ-labeled *Markov decision process* (MDP) is $\mathcal{M} = \langle S, s_0, (A_s)_{s\in S}, \Pr_{\mathcal{M}}, \Sigma, \tau\rangle$, where S is a finite set of states, $s_0 \in S$ is an initial state, and for every $s \in S$, the set A_s is a finite set of actions that are available in s. Let $A = \bigcup_{s\in S} A_s$ be the set of all actions. Then, $\Pr_{\mathcal{M}} : S \times A \times S \nrightarrow [0, 1]$ is a (partial) stochastic transition function: for every two states $s, s' \in S$ and action $c \in A_s$, we have that $\Pr_{\mathcal{M}}(s, c, s')$ is the probability of moving from s to s' when action c is taken. Accordingly, for every $s \in S$ and $c \in A_s$, we have $\Sigma_{s'\in S}\Pr_{\mathcal{M}}(s, c, s') = 1$. Finally, Σ is a finite alphabet, and $\tau : S \to \Sigma$ is a labeling function on the states.

Consider a nondeterministic automaton $\mathcal{N}$ with alphabet Σ, and a Σ-labeled MDP $\mathcal{M}$, and consider the following game between a player and nature. The positions of the game are pairs $\langle s, q\rangle$, where $s \in S$ is a state of $\mathcal{M}$, and $q \in Q$ is a state of $\mathcal{N}$. The game starts by the player picking an action $c \in A_{s_0}$ available from the initial state of $\mathcal{M}$, and the game moves to $\langle s, q_0\rangle$ with probability $\Pr_{\mathcal{M}}(s_0, c, s)$. Then, at each round from position $\langle s, q\rangle$, the player chooses an action $c \in A_s$ and a state $q' \in \delta(q, \tau(s))$, and then nature chooses a state $s' \in S$ with probability $\Pr_{\mathcal{M}}(s, c, s')$, and the game moves to $\langle s', q'\rangle$. The objective of the player is to on the one hand maximize the probability of generating a word in $L(\mathcal{N})$ and on the other hand, to generate a rejecting run on words in $L(\mathcal{N})$ with zero probability. When the automaton $\mathcal{N}$ is deterministic, the player has to only pick actions, as the state $q' \in \delta(q, \tau(s))$ is unique. When the automaton is nondeterministic, the player may fail to generate an accepting run of $\mathcal{N}$ even when the actions he provides induce a word in $L(\mathcal{N})$. Indeed, each nondeterministic choice may lead to accepting only a subset of the words in $L(\mathcal{N})$. Intuitively, $\mathcal{N}$ is good-for-MDP, if for every MDP $\mathcal{M}$, the nondeterminism in $\mathcal{N}$ does not prevent the player from maximizing the probability of generating a word in $L(\mathcal{N})$ while generating a rejecting run on words in $L(\mathcal{N})$ with zero probability. We now formalize this intuition.

A *run* of $\mathcal{M}$ is an infinite sequence of states $r = \langle s_0, s_1, s_2, \ldots\rangle \in S^\omega$, such that for all $i \geq 0$ there exists $c \in A_{s_i}$ with $\Pr_{\mathcal{M}}(s_i, c, s_{i+1}) > 0$. A finite run is a prefix of such a sequence. Let $\Omega(\mathcal{M}) \subseteq S^\omega$ denote the set of runs of $\mathcal{M}$ from the initial state s_0, and let $\Pi(\mathcal{M}) \subseteq S^*$ denote the set of finite such runs. For a run $r \in \Omega(\mathcal{M})$, we define the corresponding labeled run as $\tau(r) = \tau(s_1), \tau(s_2), \tau(s_3), \ldots \in \Sigma^\omega$. Note that the label of the first state s_0 of a run is not part of $\tau(r)$.

Consider the set A of actions. Let $\mathrm{dist}(A)$ be the set of distributions over A. A distribution $d \in \mathrm{dist}(A)$ is a *point distribution* if there exists $c \in A$ such that $d(c) = 1$. A *strategy* for an MDP $\mathcal{M}$ is a function $\mu : \Pi(\mathcal{M}) \to \mathrm{dist}(A)$, which suggests to the player a

distribution on the available actions given the history of the game so far. The strategy should suggest an available action, thus $\mu(s_0, s_1, s_2, \ldots, s_k)(c) > 0$ implies that $c \in A_{s_k}$. Let $\Pi(\mathcal{M}, \mu)$ denote the subset of $\Pi(\mathcal{M})$ that includes exactly all finite runs that agree with the strategy μ. Formally, we have that $\langle s_0 \rangle \in \Pi(\mathcal{M}, \mu)$, and for all $p' = \langle s_0, s_1, \ldots, s_k \rangle \in \Pi(\mathcal{M}, \mu)$, and $s_{k+1} \in S$, we have that $p = p' \cdot s_{k+1} \in \Pi(\mathcal{M}, \mu)$ iff there exists $c_{k+1} \in A_{s_k}$ such that $\mu(p')(c_{k+1}) > 0$ and $\Pr_{\mathcal{M}}(s_k, c_{k+1}, s_{k+1}) > 0$. Thus, a run in $\Pi(\mathcal{M}, \mu)$ can only be extended using actions that have a positive probability according to μ. Let $\Omega(\mathcal{M}, \mu)$ denote the subset of $\Omega(\mathcal{M})$ that includes exactly all runs r such that for every $p \in \Pi(\mathcal{M})$ that is a prefix of r, it holds that $p \in \Pi(\mathcal{M}, \mu)$. Let $\mathcal{F}(\mathcal{M})$ be the set of all strategies.

We say that a strategy μ is *pure* if for all $p \in \Pi(\mathcal{M})$, the distribution $\mu(p)$ is a point distribution, and is *mixed* otherwise. We say that μ has *finite-memory*, if there exists a finite set M of memories, with a distinguished initial memory $m_0 \in M$, and a memory update function $\eta : S \times M \to M$ such that μ only depends on the current memory and MDP state. Formally, η is extended to finite sequences in S^* as follows: $\eta^* : S^* \to M$ is such that for every finite sequence $p \in S^*$ and MDP state $s \in S$, we have that $\eta^*(\epsilon) = m_0$ and $\eta^*(p \cdot s) = \eta(s, \eta^*(p))$. Then, for all $s \in S$ and $p \cdot s, p' \cdot s \in \Pi(\mathcal{M}, \mu)$, if $\eta^*(p \cdot s) = \eta^*(p' \cdot s)$ then $\mu(p \cdot s) = \mu(p' \cdot s)$. Equivalently, μ can be thought as a function $\mu : S \times M \to \text{dist}(A) \times M$, that for a given the current MDP state s and memory m, outputs a distribution on the available actions from s and the next memory state to move to from m.

Indeed, we show that every finite memory strategy μ with memory set M can be represented as a strategy of the form $\mu' : S \times M \to \text{dist}(A) \times M$. We say that $\langle s, m \rangle \in S \times M$ is *feasible* by μ if there is a finite run $p_{m,s}$ such that $p_{m,s} \cdot s \in \Pi(\mathcal{M}, \mu)$ and $\eta^*(p_{m,s} \cdot s) = m$. In other words, there is a positive probability that when playing according to μ we reach the memory state m when being in state s in the MDP. We can now define $\mu' : S \times M \nrightarrow \text{dist}(A) \times M$ as a partial function on all feasible $\langle s, m \rangle \in S \times M$ by $\mu'(s, m) = \langle \mu(p_{m,s} \cdot s), \eta(s, m) \rangle$. Observe that $\mu(p_{m,s} \cdot s)$ is independent of the choice of witness $p_{m,s}$. That is, for all $p \cdot s \in \Pi(\mathcal{M}, \mu)$ with $\eta^*(p \cdot s) = m$ we have that $\mu(p \cdot s) = \mu(p_{m,s} \cdot s)$. Thus, if $\mu'(s, m) = \langle d, m' \rangle$, then $\mu(p \cdot s) = d$ for all $p \cdot s \in \Pi(\mathcal{M}, \mu)$ such that $\eta^*(p \cdot s) = m$.

We say that μ is *memoryless* if it has a finite memory M with $|M| = 1$. Equivalently, if for all $p, p' \in \Pi(\mathcal{M}, \mu)$ and $s \in S$ such that $p \cdot s, p' \cdot s \in \Pi(\mathcal{M}, \mu)$, it holds that $\mu(p \cdot s) = \mu(p' \cdot s)$. Namely, if μ only depends on the current state, and not on the history beforehand.

A *Markov Chain* (MC, for short) is an MDP with a dummy set of actions, $A_s = \{*\}$ for all $s \in S$. Formally, an MC is $\mathcal{C} = \langle S, s_0, \Pr_{\mathcal{C}}, \Sigma, \tau \rangle$, where S is the set of states, $s_0 \in S$ is the initial state, $\Pr_{\mathcal{C}} : S \times S \to [0, 1]$ is a stochastic transition function: for every $s \in S$ it holds that $\sum_{s' \in S} \Pr_{\mathcal{C}}(s, s') = 1$. A *random process* of $\mathcal{C}$, is an infinite sequence of random variables $S_0, S_1, S_2, \ldots$ with values in S, such that $S_0 = s_0$ with probability 1, and for all $s_1, s_2, \ldots, s_{i+1} \in S$, $i \geq 0$, it holds that $\mathbb{P}_{\mathcal{C}}(S_{i+1} = s_{i+1} | S_0 = s_0, \ldots, S_i = s_i) = \Pr_{\mathcal{C}}(s_i, s_{i+1})$.

Given a strategy μ, we can obtain from $\mathcal{M}$ and μ the MC $\mathcal{C}(\mathcal{M}, \mu) = \left\langle \Pi(\mathcal{M}, \mu), s_0, \Pr^{\mu}_{\mathcal{M}} \right\rangle$ in which the choice of actions is resolved according to μ. Formally, for all $p \in \Pi(\mathcal{M}, \mu)$ and $s_1, s_2 \in S$ such that both $p_1 = p \cdot s_1$ and $p_2 = p_1 \cdot s_2$ belong to $\Pi(\mathcal{M}, \mu)$, then $\Pr^{\mu}_{\mathcal{M}}(p_1, p_2) = \sum_{c \in A_{s_1}} \mu(p_1)(c) \cdot \Pr_{\mathcal{M}}(s_1, c, s_2)$. It is clear that when μ is memoryless, that is, μ only depends on the last position in p_1, then $\Pr^{\mu}_{\mathcal{M}}$ only depends on s_1 and s_2, and hence the projection of the chain $\mathcal{C}(\mathcal{M}, \mu)$ onto the last position in a path is a finite MC. Formally, let $\iota : \Pi(\mathcal{M}, \mu) \to S$ be the function that maps a finite path to its last state and let $d_s \in \text{dist}(\mathcal{A})$ be the unique distribution such that $\mu(p) = d_s$ for all $p \in \Pi(\mathcal{M}, \mu)$ with $\iota(p) = s$. Note that if no such $p \in \Pi(\mathcal{M}, \mu)$ with $\iota(p) = s$ exists, then we arbitrarily take d_s to be some distribution in A_s. Let $\Pr'^{\mu}_{\mathcal{M}} : S \times S \to [0, 1]$ be the stochastic transition function that is defined by

$\Pr'^{\mu}_{\mathcal{M}}(s,s') = \sum_{c\in A_s} d_s(c)\cdot \Pr_{\mathcal{M}}(s,c,s')$. Then, if $p_0,p_1,p_2,\ldots \in (\Pi(\mathcal{M},\mu))^\omega$ is an infinite sequence of increasing paths that are sampled according to $\mathcal{C}(\mathcal{M},\mu)$, then it is not hard to see that the sequence $\iota(p_0),\iota(p_1),\iota(p_2),\ldots \in S^\omega$ satisfies $\mathbb{P}(\iota(p_{i+1}) = s' | \iota(p_i) = s) = \Pr'^{\mu}_{\mathcal{M}}(s,s')$ for all $s',s\in S$. In other words, the sequence $\iota(p_0),\iota(p_1),\iota(p_2),\ldots$ is sampled according to the finite markov chain $\left\langle S, s_0, \Pr'^{\mu}_{\mathcal{M}}\right\rangle$.

Let $\mathcal{M} = \langle S, s_0, (A_s)_{s\in S}, \Pr_{\mathcal{M}}, \Sigma, \tau\rangle$ be an MDP and $\mathcal{N} = \langle \Sigma, Q, q_0, \delta, \alpha\rangle$ be a nondeterministic automaton. When the player takes actions in $\mathcal{M}$ the infinite word $\tau(r)\in\Sigma^\omega$ is generated. Assume that, simultaneously, the player also generates a run of $\mathcal{N}$. This process can be modeled by an MDP that is a product of $\mathcal{M}$ with $\mathcal{N}$, and in each step the player takes an action of $\mathcal{M}$ and chooses a successor state in $\mathcal{N}$ that extends the current run on the new letter drawn by $\mathcal{M}$. Formally, we define the MDP $\mathcal{M}\times\mathcal{N} = \langle S', \langle s_0,\bot\rangle, A', \Pr_{\mathcal{M}\times\mathcal{N}}, \Sigma, \gamma\rangle$ that is defined as follows. The set of states of $\mathcal{M}\times\mathcal{N}$ is $S' = (S\times Q)\cup\{\langle s_0,\bot\rangle\}$, for $\bot\notin Q$. The set of actions $A' = A\,\langle s_0,\bot\rangle \cup \bigcup_{\langle s,q\rangle\in S\times Q}(A_{\langle s,q\rangle})$ is defined as follows. For all $\langle s,q\rangle \in S\times Q$, the set of actions $A_{\langle s,q\rangle}$ available from $\langle s,q\rangle$ consists of all pairs $\langle c,q'\rangle \in A_s\times\delta(q,\tau(s))$. In addition, the actions available from $\langle s_0,\bot\rangle$ are $A_{\langle s_0,\bot\rangle} = A_{s_0}\times\{q_0\}$. Then, for all $\langle s,*\rangle\in S'$ and $\langle c,q'\rangle\in A_{\langle s,*\rangle}$, we have $\Pr_{\mathcal{M}\times\mathcal{N}}(\langle s,*\rangle,\langle c,q'\rangle,\langle s',q'\rangle) = \Pr_{\mathcal{M}}(s,c,s')$ and $\gamma(s,*) = \tau(s)$. Given a finite or infinite run $r\in\Pi(\mathcal{M}\times\mathcal{N})\cup\Omega(\mathcal{M}\times\mathcal{N})$, $r = \langle\langle s_0,\bot\rangle,\langle s_1,q_0\rangle,\langle s_2,q_1\rangle,\ldots\rangle$, we define $\pi_{\mathcal{N}}(r) = \langle q_0,q_1,q_2,\ldots\rangle$, and $\pi_{\mathcal{M}}(r) = \langle s_0,s_1,s_2,\ldots\rangle$ to be the $\mathcal{N}$-run and $\mathcal{M}$-run that correspond to r respectively. Conversely, for $r_{\mathcal{M}} = \langle s_0,s_1,s_2,\ldots\rangle \in S^*\cup S^\omega$ and $r_{\mathcal{N}} = \langle q_0,q_1,q_2,\ldots\rangle \in (Q_{\mathcal{N}})^*\cup(Q_{\mathcal{N}})^\omega$ of appropriate lengths, let $r_{\mathcal{M}}\oplus r_{\mathcal{N}} = \langle\langle s_0,\bot\rangle,\langle s_1,q_0\rangle,\langle s_2,q_1\rangle,\ldots\rangle$. Notice that for all $r_{\mathcal{M}}\oplus r_{\mathcal{N}}\in\Pi(\mathcal{M}\times\mathcal{N})\cup\Omega(\mathcal{M}\times\mathcal{N})$, we have $\pi_{\mathcal{N}}(r_{\mathcal{M}}\oplus r_{\mathcal{N}}) = r_{\mathcal{N}}$ and $\pi_{\mathcal{M}}(r_{\mathcal{M}}\oplus r_{\mathcal{N}}) = r_{\mathcal{M}}$.

We can now formally define the notion an automaton being *good-for-MDP*. We use the formalism of [HPS+20]. The *semantic satisfaction probability* of a strategy $\mu\in\mathcal{F}(\mathcal{M})$ with respect to $\mathcal{N}$, denoted $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}(\mu)$, is defined as:

$$\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}(\mu) = \Pr^{\mu}_{\mathcal{M}}(r\in\Omega(\mathcal{M},\mu) : \tau(r)\in L(\mathcal{N})).$$

That is, $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}(\mu)$ is the probability that when the player plays with μ, the word $\tau(r)$ generated by $\mathcal{M}$ is in $L(\mathcal{N})$. Then, the *semantic satisfaction probability* of an MDP $\mathcal{M}$ with respect to $\mathcal{N}$, denoted $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}$, is defined by

$$\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}} = \sup_{\mu\in\mathcal{F}(\mathcal{M})} \mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}(\mu).$$

That is, $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}$ is the least upper bound on how good the player can do with respect to the objective of generating a word in $L(\mathcal{N})$.

Similarly, the *syntactic satisfaction probability* of $\mu\in\mathcal{F}(\mathcal{M}\times\mathcal{N})$ and $\mathcal{M}$, denoted $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu)$ and $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}$ respectively, are defined by

$$\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu) = \Pr^{\mu}_{\mathcal{M}\times\mathcal{N}}(r\in\Omega(\mathcal{M}\times\mathcal{N},\mu) : \pi_{\mathcal{N}}(r) \text{ is accepting}),$$

and

$$\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}} = \sup_{\mu\in\mathcal{F}(\mathcal{M}\times\mathcal{N})} \mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu).$$

Maximizing the syntactic satisfaction can be expressed as a simple stochastic parity game that is played on top of the MDP $\mathcal{M}\times\mathcal{N}$, where the winning objective is inherited from $\mathcal{N}$. In [CH03], the authors prove that simple stochastic parity games admit optimal pure memoryless strategies. Thus, the supremum in the definition of $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}$ is attained by a pure

memoryless strategy. Note that $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}$ is evaluated with respect to the MDP $\mathcal{M} \times \mathcal{N}$, and thus by pure memoryless we mean that there is a strategy $\mu^*_{\mathcal{N}} : S \times Q \cup \{\langle s_0, \bot\rangle\} \to A \times Q$ such that $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu^*_{\mathcal{N}}) = \mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}$.

Similarly, the semantic satisfaction can be expressed as a simple stochastic parity game that augments $\mathcal{M} \times \mathcal{N}$ with a deterministic automaton $\mathcal{D}$ for $L(\mathcal{N})$. Thus, the supremum in the definition of $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}$ is attained by a pure finite-memory strategy. Indeed, the considerations are as in the syntactic case, except that here, the current state of the deterministic automaton $\mathcal{D}$ is used as a memory for the player.

In general, $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}} \leq \mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}$. In the syntactic case, the player not only aims to generate a word in $L(\mathcal{N})$, but also to simultaneously generate an accepting run of $\mathcal{N}$ on it. Therefore, any strategy for the syntactic case can be used for the semantic case by ignoring the component of $\mathcal{N}$, resulting in a potentially higher semantic satisfaction. Equality, however, is not guaranteed. Indeed, the same way not all automata are history deterministic, it may not be possible to resolve the nondeterminism in $\mathcal{N}$ in a way that maximizes the probability of generating a word in $L(\mathcal{N})$ (thus reaching a maximal semantic satisfaction) while generating rejecting runs on words in $L(\mathcal{N})$ with zero probability (and thus reaching an equal syntactic satisfaction). When the equality holds for all MDP $\mathcal{M}$, we say that $\mathcal{N}$ is *good-for-MDP* (GFM for short). Note that when $\mathcal{N}$ is GFM, it holds that for all MDP $\mathcal{M} = \langle S, s_0, (A_s)_{s\in S}, \Pr_{\mathcal{M}}, \Sigma, \tau\rangle$, there exists a pure-memroyless strategy for $\mathcal{M} \times \mathcal{N}$, $\mu^*_{\mathcal{N}} : S \times Q \cup \{\langle s_0, \bot\rangle\} \to A \times Q$, such that $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}} = \mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu^*_{\mathcal{N}})$ and $\mu^*_{\mathcal{N}}$ generates a rejecting run on words in $L(\mathcal{N})$ with zero probability. Note that $\mu^*_{\mathcal{N}}$ can be viewed as a pure finite-memory strategy for $\mathcal{M}$ for which $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}} = \mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}(\mu^*_{\mathcal{N}})$.

6.2. **The nondeterminism hierarchy and GFMness.** In this section we locate GFMness in the nondeterminism hierarchy. For the semantic hierarchy, it is proved in [HPS+20] that GFM-NBWs are as expressive as NBWs. More specifically, every NBW can be translated into an equivalent GFM-NBW. Unlike the case of co-Büchi automata, where nondeterministic and deterministic automata have the same expressive power, making the expressiveness equivalence of GFM-NCWs and NCW straightforward, NBWs are more expressive than DBWs, making the result about GFM-NBWs interesting. Another aspect that has been studied is deciding GFMness [STZ22]. In [STZ22], the authors prove an exponential-time upper bound as well as a polynomial-space lower bound for the problem of deciding whether an automaton is GFM-NBW.

We study the syntactic hierarchy, we first point out that every HD automaton is GFM [KMBK14]. Then, by relating GFMness and almost-DBPness, we show that SD-NBWs are GFM, yet SD-NCWs need not be GFM. Thus, with respect to the syntactic hierarchy, GFM-NBWs are between SD and nondeterministic Büchi automata, while GFM-NCWs are incomparable to SD-NCWs.

Theorem 6.1 [KMBK14]**.** *Every HD automaton is GFM.*

Proof. The theorem is proved in [KMBK14]. We give a short proof for completeness. Consider an HD automaton $\mathcal{N}$. Let f be an HD strategy for $\mathcal{N}$. Then, every strategy $\mu \in \mathcal{F}(\mathcal{M})$ can be augmented with f to obtain a strategy $\mu \times f \in \mathcal{F}(\mathcal{M} \times \mathcal{N})$ such that $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}(\mu) = \mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}\times\mathcal{N}}(\mu \times f) = \mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu \times f)$. Thus, $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}} \leq \mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}$, and so $\mathcal{N}$ is GFM. □

Theorem 6.2. *Every GFM automaton is almost-DBP.*

Proof. Consider an ω-regular automaton $\mathcal{N}$ over an alphabet Σ, and consider the MDP $\mathcal{M} = \langle \Sigma, \sigma_0, (A_\sigma)_{\sigma\in\Sigma}, \Pr_{\mathcal{M}}, \mathsf{id}\rangle$, where $\sigma_0 \in \Sigma$ is an arbitrary initial state, $A_\sigma = \{*\}$ is a dummy set of actions for each letter $\sigma \in \Sigma$, and $\Pr_{\mathcal{M}}(\sigma, *, \sigma') = |\Sigma|^{-1}$, for all $\sigma, \sigma' \in \Sigma$. By [CJH03], there is a pure memoryless strategy $\mu^* : S \times Q_{\mathcal{N}} \cup \{\langle s_0, \bot\rangle\} \to A \times Q_{\mathcal{N}}$ that attains $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}$. Observe that since $\Pr_{\mathcal{M}}(\sigma, *, \cdot) = (\sigma' \mapsto |\Sigma|^{-1})$ is the uniform distribution on Σ for all $\sigma \in \Sigma$, then $\mathcal{M}$ is essentially an MC that generates a random word $w = \sigma_1, \sigma_2, \sigma_3, \ldots \in \Sigma^\omega$, such that all its letters are independent and distributed uniformly. Thus, the membership $w \in L(\mathcal{N})$, where w is the word generated by $\mathcal{C}(\mathcal{M}, \mu')$ (or by $\mathcal{C}(\mathcal{M} \times \mathcal{N}, \mu')$) for $\mu' \in \mathcal{F}(\mathcal{M})$ (respectively $\mu' \in \mathcal{F}(\mathcal{M} \times \mathcal{N})$), is independent of the strategy μ'. In particular $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}$ equals the probability that a random word w is in $L(\mathcal{N})$.

Note that it must be that $\mu^*(\sigma_0, \bot) = \langle *, q_0\rangle$; i.e., there is no choice at the initial state of $\mathcal{M} \times \mathcal{N}$. Also, as $A_\sigma = \{*\}$, for all $\sigma \in \Sigma$, we can think of μ^* as a function $\mu^* : \Sigma \times Q_{\mathcal{N}} \to Q_{\mathcal{N}}$, with $\mu^*(\sigma, q) \in \delta_{\mathcal{N}}(q, \sigma)$ for all $\langle \sigma, q\rangle \in \Sigma \times Q_{\mathcal{N}}$. Thus, μ^* induces a pruning $\mathcal{N}_{\mu^*}$ of $\mathcal{N}$, and $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu^*)$ equals to the probability that a random word is accepted by $\mathcal{N}_{\mu^*}$.

Now, since μ^* is an optimal strategy for $\mathcal{M} \times \mathcal{N}$ and $\mathcal{N}$ is GFM, it follows that $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu^*) = \mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}} = \mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}$. Hence, the probability that a random word $w \in \Sigma^\omega$ is accepted by $\mathcal{N}_{\mu^*}$ equals the probability that $w \in L(\mathcal{N})$. That is, $\mathcal{N}_{\mu^*}$ is a deterministic pruning of $\mathcal{N}$ that loses only a set of words of measure zero and hence $\mathcal{N}$ is almost-DBP. □

Recall that in Theorem 5.3, we argued that there is an SD-NCW that is not almost-DBP. By Theorem 6.2, we can conclude that the SD-NCW described there, is also not GFM. Thus, while HDness implies GFMness for all acceptance conditions [KMBK14], for co-Büchi automata, SDness need not imply GFMness. In the rest of this section we prove that for Büchi automata, SDness does imply GFMness, thus all SD-NBWs are GFM. In particular, Theorem 6.2 suggests an alternative proof of Theorem 5.2, which states that all SD-NBWs are almost-DBP.

Proposition 6.3. *If $\mathcal{N}$ is an SD-NBW and $\mu \in \mathcal{F}(\mathcal{M})$ is a finite memory strategy, then there exists $\mu' \in \mathcal{F}(\mathcal{M} \times \mathcal{N})$ such that* $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}(\mu) = \mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu')$.

Proof. Let $\mu : S \times M \to \mathrm{dist}(A) \times M$, be a finite memory strategy in $\mathcal{F}(\mathcal{M})$, and let $\mathcal{C} = \mathcal{C}(\mathcal{M}, \mu)$ be the Markov chain induced when taking actions in $\mathcal{M}$ according to μ. Note that since μ uses M as a memory structure, we may think of $\mathcal{C}$ as a Markov chain with positions $S \times M$, and a stochastic transition function $\Pr_{\mathcal{C}} : (S \times M)^2 \to [0, 1]$ that is defined by $\Pr_{\mathcal{C}}(\langle s, m\rangle, \langle s', m'\rangle) = \sum_{c\in A_s} d(c) \cdot \Pr_{\mathcal{M}}(s, c, s')$, when $\mu(s, m) = \langle d, m'\rangle$, and $\Pr_{\mathcal{C}}(\langle s, m\rangle, \langle s', m'\rangle) = 0$, otherwise. Let $r = \langle\langle s_0, m_0\rangle, \langle s_1, m_1\rangle, \langle s_2, m_2\rangle, \ldots\rangle \in (S \times M)^\omega$ be a random process sampled according to $\mathcal{C}$. Let $r_{\mathcal{N}} = \langle q_0, q_1, q_2, \ldots\rangle \in (Q_{\mathcal{N}})^\omega$ be a random run of $\mathcal{N}$ over the word $\tau(r)$ that is sampled when playing in $\mathcal{M}$ with the strategy μ. I.e., given $\langle s_0, m_0\rangle, \langle s_1, m_1\rangle, \ldots, \langle s_k, m_k\rangle$, for $k \geq 1$, and $q_0, q_1, \ldots, q_{k-1}$, we set q_k to be one of the $\tau(s_k)$-successors of q_{k-1} at random. Notice that the composition $r \oplus r_{\mathcal{N}} = \langle\langle s_0, m_0, q_0\rangle, \langle s_1, m_1, q_1\rangle, \ldots\rangle$ is a process of a finite Markov chain $\mathcal{C}_{\mathcal{N}}$ with a set of positions $S \times M \times Q_{\mathcal{N}}$. Indeed, the stochastic transition function of $\mathcal{C}_{\mathcal{N}}$ can be expressed as follows:

$$\Pr_{\mathcal{C}_{\mathcal{N}}}(\langle s, m, q\rangle, \langle s', m', q'\rangle) = \begin{cases} |\delta_{\mathcal{N}}(q, \tau(s'))|^{-1} \cdot \Pr_{\mathcal{C}}(\langle s, m\rangle, \langle s', m'\rangle) & : q' \in \delta_{\mathcal{N}}(q, \tau(s')) \\ 0 & : \text{otherwise} \end{cases}$$

Since $\mathcal{C}_{\mathcal{N}}$ is a finite Markov chain, it holds that with probability 1, the random process $r \oplus r_{\mathcal{N}}$ of $\mathcal{C}_{\mathcal{N}}$ eventually enters an ergodic component and traverses all its states. We say that

an ergodic component $C \subseteq S \times M \times Q_{\mathcal{N}}$ of $\mathcal{C}_{\mathcal{N}}$ is $\alpha_{\mathcal{N}}$-*accepting* if the projection of C onto $Q_{\mathcal{N}}$ intersects $\alpha_{\mathcal{N}}$. I.e., C is $\alpha_{\mathcal{N}}$-accepting if there exist $s \in S$, $m \in M$ and $q \in \alpha_{\mathcal{N}}$ such that $\langle s, m, q\rangle \in C$. Thus, if $r \oplus r_{\mathcal{N}}$ eventually enters an $\alpha_{\mathcal{N}}$-accepting ergodic component C, then with probability 1 the process traverses all the positions in C infinitely often, and in particular with probability 1 $r_{\mathcal{N}}$ is an accepting run of $\mathcal{N}$ over the generated word $\tau(r)$. Since with probability 1 the process of $\mathcal{C}_{\mathcal{N}}$ eventually enters an ergodic component, it follows that the probability that $\tau(r) \in L(\mathcal{N})$ but $r_{\mathcal{N}}$ is rejecting equals the probability that $r \oplus r_{\mathcal{N}}$ enters an ergodic component that is not $\alpha_{\mathcal{N}}$-accepting and $\tau(r) \in L(\mathcal{N})$. We prove that if $r \oplus r_{\mathcal{N}}$ enters an ergodic component C that is not $\alpha_{\mathcal{N}}$-accepting then $\tau(r) \notin L(\mathcal{N})$. Note that we claim that the implication C not $\alpha_{\mathcal{N}}$-accepting implies that $\tau(r) \notin L(\mathcal{N})$, is surely, and not only almost-surely. Consider an ergodic component C that is not $\alpha_{\mathcal{N}}$-accepting, and assume that $r \oplus r_{\mathcal{N}}$ is such that $\langle s_k, m_k, q_k\rangle \in C$ for some $k \geq 0$. Assume by way of contradiction that $\tau(r) \in L(\mathcal{N})$. Since $\mathcal{N}$ is SD, it is possible to extend the finite run $q_0, q_1, \ldots q_k$ of $\mathcal{N}$ over $\tau(s_1), \tau(s_2), \ldots, \tau(s_k)$ into an accepting run of $\mathcal{N}$ over $\tau(r)$. In particular there exist $j > k$ and $q'_{k+1}, q'_{k+2}, \ldots, q'_j \in (Q_{\mathcal{N}})^*$, such that $q'_j \in \alpha_{\mathcal{N}}$ and $q_0, \ldots, q_k, q'_{k+1}, \ldots, q'_j$ is a finite run of $\mathcal{N}$ over $\tau(s_1), \ldots, \tau(s_j)$. By definition of the stochastic transition function of $\mathcal{C}_{\mathcal{N}}$, it follows that there is a positive probability to move to $\langle s_{k+1}, m_{k+1}, q'_{k+1}\rangle$ from $\langle s_k, m_k, q_k\rangle$. Similarly, it follows by induction that for all $k+1 < i \leq j$ there is a positive probability to move to $\langle s_i, m_i, q'_i\rangle$ from $\langle s_{i-1}, m_{i-1}, q'_{i-1}\rangle$. In particular $\langle s_j, m_j, q'_j\rangle \in C$, in contradiction to the assumption that C is not $\alpha_{\mathcal{N}}$-accepting.

Let $\mu' \in \mathcal{F}(\mathcal{M} \times \mathcal{N})$ be the strategy for $\mathcal{M} \times \mathcal{N}$ that resolves the actions of $\mathcal{M}$ according to μ and in addition generates a random run of $\mathcal{N}$ over $\tau(r)$. It is not hard to see that $\mathcal{C}_{\mathcal{N}}$ induces the same distribution over infinite paths in $(S \times Q_{\mathcal{N}})^\omega$ as the Markov chain $\mathcal{C}(\mathcal{M} \times \mathcal{N}, \mu')$. We proved that there is a zero probability that $\tau(r) \in L(\mathcal{N})$ and $r_{\mathcal{N}}$ is rejecting. Thus, $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu') = \mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}(\mu)$. □

We can now conclude with the desired connection between SDness and GFMness. We first examine whether SDness implies GFMness, and show that the answer depends on the acceptance condition.

Theorem 6.4. *All SD-NBWs are GFM, yet there is an SD-NCW that is not GFM.*

Proof. The co-Büchi case follows from Theorems 5.3 and 6.2. For Büchi automata, consider an SD-NBW $\mathcal{N} = \langle \Sigma, Q_{\mathcal{N}}, q_0, \delta_{\mathcal{N}}, \alpha_{\mathcal{N}}\rangle$, and a Σ-labeled MDP $\mathcal{M}$. Let μ be a finite memory strategy such that $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}(\mu) = \mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}$ given by [CJH03]. Then by Proposition 6.3 it holds that there exists $\mu' \in \mathcal{F}(\mathcal{M} \times \mathcal{N})$ such that $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu') = \mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}(\mu)$, and we get $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}} = \mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu') \leq \mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}$. Since $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}} \leq \mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}$ holds for all $\mathcal{N}$ and $\mathcal{M}$, we have the equality $\mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}} = \mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}$. □

We continue and check whether GFMness implies SDness, and show that the answer is negative even for weak automata.

Theorem 6.5. *There is a GFM-NWW that is not SD.*

Proof. Consider the NBW $\mathcal{N}$ described in Figure 6. Note that $\mathcal{N}$ is weak, as all cycles are self-loops. Also, $\mathcal{N}$ is not SD, as $a \cdot b^\omega \in L(\mathcal{N}^{q_0}) \setminus L(\mathcal{N}^{q_{acc}})$. Also, it is easy to see that $L(\mathcal{N}) = \{w : w \text{ has finitely many } a\text{'s}\}$. Indeed, $\mathcal{N}$ moves to the state q_{acc} when it guesses that there are no more a's in the suffix to be read.

We show that $\mathcal{N}$ is GFM. Consider an MDP $\mathcal{M}$, and let $\mu : S \times M \to dist(A) \times M$ be a finite memory strategy such that $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}(\mu) = \mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}$. Consider the MC $\mathcal{C} = \mathcal{C}(\mathcal{M}, \mu)$,

and note that since μ uses M as a memory structure, we may think of $\mathcal{C}$ as a finite MC with positions $S \times M$, and a stochastic transition function $\Pr_{\mathcal{C}} : (S \times M)^2 \to [0,1]$ that is defined $\Pr_{\mathcal{C}}(\langle s, m\rangle, \langle s', m'\rangle) = \sum_{c \in A_s} d(c) \cdot \Pr_{\mathcal{M}}(s, c, s')$ when $\mu(s, m) = \langle d, m'\rangle$, and otherwise $\Pr_{\mathcal{C}}(\langle s, m\rangle, \langle s', m'\rangle) = 0$.

To conclude the GFMness of $\mathcal{N}$, we show that there is a strategy $\mu' \in \mathcal{F}(\mathcal{M} \times \mathcal{N})$ such that $\mathrm{PSem}^{\mathcal{N}}_{\mathcal{M}}(\mu) = \mathrm{PSyn}^{\mathcal{N}}_{\mathcal{M}}(\mu')$. Recall that a strategy for $\mathcal{M} \times \mathcal{N}$ needs to take actions in $\mathcal{M}$ and in $\mathcal{N}$ simultaneously. In the first it needs to choose an action $c \in A_s$ where $s \in S$ is the current state in $\mathcal{M}$, while in the latter it should choose a $\tau(s)$-successor of the current state q in $\mathcal{N}$. The strategy μ' resolves the first according to μ, and hence ignores the $\mathcal{N}$ component. Then, for the $\mathcal{N}$ component, μ' generates a run $r_{\mathcal{N}}$ of $\mathcal{N}$ on $w_{\mathcal{C}}$ as follows. The run $r_{\mathcal{N}}$ waits at q_0 (possibly forever), and moves to q_{acc} only when an ergodic SCC that does not contain an a-labeled state is reached in $\mathcal{C}$; in particular, if the ergodic SCC that is reached in $\mathcal{C}$ contains an a-labeled state, then $r_{\mathcal{N}}$ does not leave q_0. In other words, the strategy μ' moves from q_0 to q_{acc} only when it is certain that there are no more a's in the suffix to be read.

As the MC $\mathcal{C}$ is finite, then with probability 1 an ergodic component of $\mathcal{C}$ is reached and all of its states are traversed infinitely often. Hence, with probability 1, we have one of the following outcomes. Either the ergodic component that is reached does not contain an a-labeled state, or it contains an a-labeled state. In the former case, the run $r_{\mathcal{N}}$ is accepting. Indeed, according to μ', the run $r_{\mathcal{N}}$ moves to q_{acc} once the ergodic component that has no a-labeled states is reached, and thus $r_{\mathcal{N}}$ eventually gets stuck at q_{acc} while reading a suffix of $w_{\mathcal{C}}$ that has no a's. In the latter case, $r_{\mathcal{N}}$ is stuck in q_0 and is thus rejecting, and with probability 1 $w_{\mathcal{C}}$ has infinitely many a's. Hence, in the latter case there is a zero probability that $w_{\mathcal{C}} \in L(\mathcal{N})$ and $r_{\mathcal{N}}$ is rejecting, and we are done. □

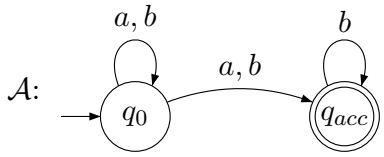

Figure 6: A GFM-NWW that is not SD. Missing transitions lead to a rejecting sink.

References

[AK15] G. Avni and O. Kupferman. Stochastization of weighted automata. In *40th Int. Symp. on Mathematical Foundations of Computer Science*, volume 9234 of *Lecture Notes in Computer Science*, pages 89–102. Springer, 2015.

[AK23] B. Abu Radi and O. Kupferman. On semantically-deterministic automata. In *Proc. 51st Int. Colloq. on Automata, Languages, and Programming*, volume 261 of *LIPIcs*, pages 109:1–109:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2023.

[AKL10] B. Aminof, O. Kupferman, and R. Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 6(2), 2010.

[AS87] B. Alpern and F.B. Schneider. Recognizing safety and liveness. *Distributed computing*, 2:117–126, 1987.

[BK18] 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.

[BKKS13] 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.

[BKLS20] 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.

[BKS17] 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.

[BL19] 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.

[Büc62] 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.

[CH03] H. Chockler and J.Y. Halpern. Responsibility and blame: a structural-model approach. In *Proc. 19th Int. Joint Conf. on Artificial Intelligence*, pages 147–153, 2003.

[CJH03] K. Chatterjee, M. Jurdziński, and T.A. Henzinger. Simple stochastic parity games. In *Proc. 12th Annual Conf. of the European Association for Computer Science Logic*, volume 2803, pages 100–113. Springer, 2003.

[Col09] 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.

[HKR02] T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. *Information and Computation*, 173(1):64–81, 2002.

[HP06] 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.

[HPS+20] 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.

[KM18] 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.

[KMBK14] Joachim Klein, David Müller, Christel Baier, and Sascha Klüppelholz. Are good-for-games automata good for probabilistic model checking? In *Language and Automata Theory and Applications*, pages 453–465. Springer International Publishing, 2014.

[KS15] 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.

[KSV06] O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. *Ann. Pure Appl. Logic*, 138(1-3):126–146, 2006.

[Kup18] O. Kupferman. Automata theory and model checking. In *Handbook of Model Checking*, pages 107–151. Springer, 2018.

[KV05a] O. Kupferman and M.Y. Vardi. From linear time to branching time. *ACM Transactions on Computational Logic*, 6(2):273–294, 2005.

[KV05b] O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, 2005.

[Lan69] L.H. Landweber. Decision problems for ω–automata. *Mathematical Systems Theory*, 3:376–384, 1969.

[LZ20] K. Lehtinen and M. Zimmermann. Good-for-games ω-pushdown automata. In *Proc. 35th IEEE Symp. on Logic in Computer Science*, 2020.

[MH84] S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

[Mor03] G. Morgenstern. Expressiveness results at the bottom of the ω-regular hierarchy. M.Sc. Thesis, The Hebrew University, 2003.

[MSS88] D.E. Muller, A. Saoudi, and P. E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *Proc. 3rd IEEE Symp. on Logic in Computer Science*, pages 422–427, 1988.

[NW98] 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.

[RS59] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.

[Saf88] S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.

[Sis94] A.P. Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6:495–511, 1994.

[STZ22] S. Schewe, Q. Tang, and T. Zhanabekova. Deciding what is good-for-MDPs. *CoRR*, abs/2202.07629, 2022. URL: `https://arxiv.org/abs/2202.07629`, `arXiv:2202.07629`.

[SVW87] A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.

[VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

Appendix A. Determinization of SD-NBWs

In this section we prove formally that the DBW $\mathcal{A}'$ is equivalent to the SD-NBW $\mathcal{A}$, and that if $\mathcal{A}$ is weak, then so is $\mathcal{A}'$. The following two propositions follow immediately from the definitions:

Proposition A.1. *Consider states $q \in Q$ and $S \in Q'$, a letter $\sigma \in \Sigma$, and transitions $\langle q, \sigma, q' \rangle$ and $\langle S, \sigma, S' \rangle$ of $\mathcal{A}$ and $\mathcal{A}'$, respectively. If q is $\mathcal{A}$-equivalent to the states in S, then q' is $\mathcal{A}$-equivalent to the states in S'.*

Proposition A.2. *Consider a state S of $\mathcal{A}'$ and a letter $\sigma \in \Sigma$. If $\langle S, \sigma, S' \rangle \in \Delta'$ and $S' \notin \alpha'$, then all the σ-successors of a state $s \in S$ are in $S' \setminus \alpha$.*

We can now prove the correctness of the construction:

Proposition A.3. *The automata $\mathcal{A}$ and $\mathcal{A}'$ are equivalent.*

Proof. 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 $S_{j_i} \in \alpha'$.

We also define $j_0 = 0$.

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 $\langle s_j, \sigma_j, s_{j+1} \rangle \in \Delta$. 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 $S_0, S_1, S_2 \ldots$ be the run of $\mathcal{A}'$ on w, and assume, by way of contradiction, that there is a position $j \geq 0$ such that $S_j, S_{j+1}, \ldots$ is an α-free run on the suffix $w[j+1, \infty]$. Then, an iterative application of Proposition A.2 implies that all the runs of a state $s_j \in S_j$ on $w[j+1, \infty]$ are α-free in $\mathcal{A}$. Also, an iterative application of Proposition A.1 implies that $r_j \sim_{\mathcal{A}} s_j$, and since r is an accepting run of $\mathcal{A}$, it holds that $\mathcal{A}^{s_j}$ has an accepting run on $w[j+1, \infty]$, and we have reached a contradiction.

□

It is left to prove that weakness of $\mathcal{A}$ is preserved in $\mathcal{A}'$.

Proposition A.4. *If $\mathcal{A}$ is an NWW, then $\mathcal{A}'$ is a DWW.*

Proof. Assume by way of contradiction that there are reachable states $S \in \alpha'$ and $S' \notin \alpha'$, and an infinite run $r_{\mathcal{A}'} = S_0, S_1, S_2, \ldots$ that visits both S and S' infinitely often. Recall that a reachable state in Q' contains only α-states of $\mathcal{A}$ or only $\bar{\alpha}$-states of $\mathcal{A}$. Hence, S' contains only $\bar{\alpha}$-states of $\mathcal{A}$.

As in the proof of Proposition A.3, the run $r_{\mathcal{A}'}$ induces an infinite run $r_{\mathcal{A}} = s_0, s_1, s_2, \ldots$, where for all positions $j \geq 0$, it holds that $s_j \in S_j$. Since the run $r_{\mathcal{A}'}$ visits S infinitely often, then $r_{\mathcal{A}}$ visits infinitely many α-states. Likewise, since $r_{\mathcal{A}'}$ visits S' infinitely often, then $r_{\mathcal{A}}$ also visits infinitely many $\bar{\alpha}$-states. This contradicts the weakness of $\mathcal{A}$, and we are done. □

Appendix B. Details of the Reduction in Theorem 4.1

We describe the technical details of the construction of $\mathcal{A}$. Let $T = \langle \Gamma, Q, \rightarrow, q_0, q_{acc}, q_{rej} \rangle$, where Γ is the working alphabet, Q is the set of states, $\rightarrow \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$ is the transition relation (we use $(q, a) \rightarrow (q', b, \Delta)$ to indicate that when T is in state q and it reads the input a in the current tape cell, it moves to state q', writes b in the current tape cell, and its reading head moves one cell to the left/right, according to Δ), q_0 is the initial state, q_{acc} is the accepting state, and q_{rej} is the rejecting one.

The transitions function $\rightarrow$ is defined also for the final states q_{acc} and q_{rej}: when a computation of T reaches them, it erases the tape, goes to the leftmost cell in the tape, and moves to the initial state q_0. Recall that $s : \mathbb{N} \rightarrow \mathbb{N}$ is the polynomial space function of T. Thus, when T runs on the empty tape, it uses at most $n_0 = s(0)$ cells.

We use strings of the form $\#\gamma_1\gamma_2 \ldots (q, \gamma_i) \ldots \gamma_{n_0}$ to encode a configuration of T on a word of length at most n_0. That is, a configuration starts with $\#$, and all its other letters are in Γ, except for one letter in $Q \times \Gamma$. The meaning of such a configuration is that the j'th cell in T, for $1 \leq j \leq n_0$, is labeled γ_j, the reading head points at cell i, and T is in state q. For example, the initial configuration of T is $\#(q_0, b)b \ldots b$ (with $n_0 - 1$ occurrences of b's) where b stands for an empty cell. We can now encode a computation of T by a sequence of configurations.

Let $\Sigma = \{\#\} \cup \Gamma \cup (Q \times \Gamma)$ and let $\#\sigma_1 \ldots \sigma_{n_0} \# \sigma'_1 \ldots \sigma'_{n_0}$ be two successive configurations of T. We also set σ_0, σ'_0, and σ_{n_0+1} to $\#$. For each triple $\langle \sigma_{i-1}, \sigma_i, \sigma_{i+1} \rangle$ with $1 \leq i \leq n_0$,

we know, by the transition relation of T, what σ'_i should be. In addition, the letter # should repeat exactly every $n_0 + 1$ letters. Let $next(\sigma_{i-1}, \sigma_i, \sigma_{i+1})$ denote our expectation for σ'_i. That is,

- $next(\sigma_{i-1}, \sigma_i, \sigma_{i+1}) = \sigma_i$ if $\sigma_i = \#$, or if non of σ_{i-1}, σ_i and σ_{i+1} are in $Q \times \Gamma$.
- $next(\sigma_{i-1}, \sigma_i, \sigma_{i+1}) = \#$ if $|\{\sigma_{i-1}, \sigma_i, \sigma_{i+1}\} \cap Q \times \Gamma| \geq 2$, or if $|\{\sigma_{i-1}, \sigma_i, \sigma_{i+1}\} \cap \{\#\}| \geq 2$ [10].
- $next((q, \gamma_{i-1}), \gamma_i, \gamma_{i+1}) = next((q, \gamma_{i-1}), \gamma_i, \#) =$
$$\begin{cases} \gamma_i & \text{If } (q, \gamma_{i-1}) \to (q', \gamma'_{i-1}, L) \\ (q', \gamma_i) & \text{If } (q, \gamma_{i-1}) \to (q', \gamma'_{i-1}, R) \end{cases}$$
- $next(\gamma_{i-1}, \gamma_i, (q, \gamma_{i+1})) = next(\#, \gamma_i, (q, \gamma_{i+1})) =$
$$\begin{cases} \gamma_i & \text{If } (q, \gamma_{i+1}) \to (q', \gamma'_{i+1}, R) \\ (q', \gamma_i) & \text{If } (q, \gamma_{i+1}) \to (q', \gamma'_i, L) \end{cases}$$
- $next(\#, (q, \gamma_i), \gamma_{i+1}) = (q', \gamma')$ where $(q, \gamma_i) \to (q', \gamma'_i, L)$ [11].
- $next(\gamma_{i-1}, (q, \gamma_i), \gamma_{i+1}) = next(\gamma_{i-1}, (q, \gamma_i), \#) = \gamma'_i$ where $(q, \gamma_i) \to (q', \gamma'_i, \Delta)$.

Consistency with *next* now gives us a necessary condition for a word to encode a legal computation that uses n_0 tape cells.

In order to accept words that satisfy C1, namely detect a violation of the encoding, we distinguish between two types of violations: a violation of a single encoded configuration, and a violation of *next*. In order to detect a violation of *next*, the NWW $\mathcal{A}$ use its nondeterminism and guesses a triple $\langle \sigma_{i-1}, \sigma_i, \sigma_{i+1} \rangle \in \Sigma^3$ and guesses a position in the word, where it checks whether the three letters to be read starting this position are σ_{i-1}, σ_i, and σ_{i+1}, and checks whether $next(\sigma_{i-1}, \sigma_i, \sigma_{i+1})$ is not the letter to come $n_0 + 1$ letters later. Once $\mathcal{A}$ sees such a violation, it goes to an accepting sink. If *next* is respected, or if the guessed triple and position is not successful, then $\mathcal{A}$ returns to its initial state. Also, at any point that $\mathcal{A}$ still waits to guess a position of a triple, it can guess to return back to the initial state.

In order to detect a violation of a single encoded configuration, the NWW $\mathcal{A}$ accepts words that include in them a subword x of length $n_0 + 1$, which is either #-free, or is of the form $\# \cdot y$, for a finite word y that does not encode a legal configuration that uses n_0 cells. Specifically, y is not of the form $\Gamma^i \cdot (Q \times \Gamma) \cdot \Gamma^{n_0-(i+1)}$, for some $0 \leq i \leq n_0 - 1$. Indeed, such words either include a long subword with no #, or have some block of size n_0 that follows a #, that does not encode a legal configuration of T when it uses n_0 cells. If $\mathcal{A}$ succeeds to detect such a violation, it goes to an accepting sink, otherwise, $\mathcal{A}$ returns to its initial state.

In order to accept words that satisfy C2, namely detect an encoding of an accepting configuration of T, the NWW $\mathcal{A}$ guesses a position where it checks if the next $n_0 + 1$ letters are of the form $\# \cdot x \cdot (q_{acc}, \gamma) \cdot y$, for $x, y \in \Gamma^*$ and $\gamma \in \Gamma$. If the guess succeeded, then $\mathcal{A}$ goes to the accepting sink, and otherwise it goes back to the initial state. At any point that $\mathcal{A}$ waits to guess a position where the accepting configuration begins, it can guess to return back to the initial state.

[10] These case describes an illegal encoding of a configuration, and there is no correct expectation of the letter that comes in the same position in the next configuration. Thus, the choice $next(\sigma_{i-1}, \sigma_i, \sigma_{i+1}) = \#$ is arbitrary.

[11] We assume that the reading head of T stays in place when it is above the leftmost cell and the transition function directs it to move left.

Appendix C. Correctness and full details of the reduction in Theorem 4.2

We first prove that the HD strategy g defined in Proposition 4.4 satisfies two essential properties. Then, in Lemma C.3, we show that these properties imply that g is a winning HD strategy for $\mathcal{A}_\varphi$.

Lemma C.1. *For all $u \in (X \cdot C)^*$ and $v \in R_{n,m}$, if $g(u) = p$, then there is a prefix $y \in (X \cdot C)^*$ of v such that $g(u \cdot y) = q_0$.*

Proof. Let $j \in [m]$ and $2 \leq k \leq n$ be such that v ends with the word $x_k \cdot c_j \cdot x_{k+1} \cdot c_j \cdots x_n \cdot c_j$, and k is the minimal index that is greater than 1 for which x_k participates in c_j. Since we assume that each of the clauses of φ depends on at least two variables, such $k > 1$ exists. Let $i \in \{0, 1\}$ be minimal with $c_j \in C^i_k$, and let $z \in (X \cdot C)^*$ be a prefix of v such that $v = z \cdot x_k \cdot c_j \cdots x_n \cdot c_j$. If there is a prefix $y \in (X \cdot C)^*$ of z such that $g(u \cdot y) = q_0$, then we are done. Otherwise, $g(u \cdot z) = p$. By definition of g and the choice of k, we know that $g(u \cdot z \cdot x_k) = q^i_k$, where the assignment $x_k = i$ satisfies c_j. Thus, if we take $y = z \cdot x_k \cdot c_j$, then $g(u \cdot y) = q_0$, and y is a prefix of v. □

Lemma C.2. *For all $u \in (X \cdot C)^*$ and $v \in R_{n,m}$, if $g(u) = q_0$, there is a prefix $z \in (X \cdot C)^*$ of v such that $g(u \cdot z) = p$.*

Proof. This follows immediately from the fact that $D_{n,m}$ is a DFW that recognizes $R_{n,m}$ and p is the only accepting state of $D_{n,m}$. Thus, we may take z to be the minimal prefix of v that is in $R_{n,m}$. □

Recall that an HD strategy $g : \Sigma^* \to Q$ has to agree with the transitions of $\mathcal{A}_\varphi$. That is, for all $w \in (X \cdot C)^*$, $x_k \in X$, and $c_j \in C$, it holds that $(g(w), x_k, g(w \cdot x_k))$ and $(g(w \cdot x_k), c_j, g(x \cdot x_k \cdot c_j))$ are in Δ_φ. In addition, if g satisfies the conditions in Lemmas C.1 and C.2, we say that g *supports a (p, q_0)-circle*.

Lemma C.3. *If $g : \Sigma^* \to Q$ is consistent with Δ_φ and supports a (p, q_0)-circle, then for all words $w \in L_{n,m}$, the run $g(w)$ is accepting.*

Proof. Consider a word $w \in L_{n,m} = (R_{n,m})^\omega$. Observe that if $w' \in (X \cdot C)^\omega$ is a suffix of w, then $w' \in L_{n,m}$, and hence has a prefix in $R_{n,m}$. Thus, if g supports a (p, q_0)-circle, there exist $y_1, z_1 \in (X \cdot C)^*$, such that $y_1 \cdot z_1$ is a prefix of w, $g(y_1) = q_0$, and $g(y_1 \cdot z_1) = p$. Let $w' \in (X \cdot C)^\omega$ be the suffix of w with $w = y_1 \cdot z_1 \cdot w'$. By the above, $w' \in L_{n,m}$, and we can now apply again the assumption on g to obtain $y_2, z_2 \in (X \cdot C)^*$ such that $y_2 \cdot z_2$ is a prefix of w', $g(y_1 \cdot z_1 \cdot y_2) = q_0$, and $g(y_1 \cdot z_1 \cdot y_2 \cdot z_2) = p$. By iteratively applying this argument, we construct $\{y_i, z_i : i \geq 1\} \subseteq (X \cdot C)^*$, such that $w^i = y_1 \cdot z_1 \cdot y_2 \cdot z_2 \cdots y_{i-1} \cdot z_{i-1} \cdot y_i$ is a prefix of w, and $g(w^i) = q_0$, for all $i \geq 1$. We conclude that $q_0 \in \inf(g(w))$, and hence $g(w)$ is accepting. □

It is easy to see that there is a correspondence between assignments to the variables in X and deterministic prunings of $\mathcal{A}_\varphi$. Indeed, a pruning of p amounts to choosing, for each $k \in [n]$, a value $i_k \in \{0, 1\}$: the assignment $x_k = i_k$ corresponds to keeping the transition $\left\langle p, x_k, q^{i_k}_k \right\rangle$ and removing the transition $\left\langle p, x_k, q^{\neg i_k}_k \right\rangle$. For an assignment $a : X \to \{0, 1\}$, we denote by $\mathcal{A}^a_\varphi$ the deterministic pruning of $\mathcal{A}_\varphi$ that is associated with a. We prove that a satisfies φ iff $\mathcal{A}^a_\varphi$ is equivalent to $\mathcal{A}_\varphi$. Thus, the number of deterministic prunings of $\mathcal{A}_\varphi$ that result in a DBW equivalent to $\mathcal{A}_\varphi$, equals to the number of assignments that satisfy φ. In particular, φ is satisfiable iff $\mathcal{A}_\varphi$ is DBP.

Proposition C.4. *For every assignment $a : X \to \{0,1\}$, we have that $L(\mathcal{A}^a_\varphi) = L(\mathcal{A}_\varphi)$ iff φ is satisfied by a.*

Proof. Assume first that φ is not satisfied by a. We prove that $L_{n,m} \neq L(\mathcal{A}^a_\varphi)$. Let $j \in [m]$ be such that c_j is not satisfied by a. I.e, for all $k \in [n]$ the assignment $x_k = i_k$ does not satisfy c_j. Since q^i_k is reachable in $\mathcal{A}^a_\varphi$ iff $i = i_k$, and all c_j-labeled transitions from $\{q^{i_k}_k : k \in [n]\}$ are to p, it follows that the run of $\mathcal{A}^a_\varphi$ on $\{x_1 \cdot c_j \cdot x_2 \cdot c_j \cdots x_n \cdot c_j\}^\omega$ never visits q_0, and hence is rejecting. Thus, $(x_1 \cdot c_j \cdot x_2 \cdot c_j \cdots x_n \cdot c_j)^\omega \in L_{n,m} \setminus L(\mathcal{A}^a_\varphi)$.

For the other direction, we assume that a satisfies φ and prove that $L(\mathcal{A}^a_\varphi) = L_{n,m}$. Let $g^a : \Sigma^* \to Q$ be the memoryless strategy that correspond to the pruning $\mathcal{A}^a_\varphi$. By Lemma C.3, it is sufficient proving that g^a supports a (p, q_0)-circle. Note that every strategy for $L_{n,m}$ satisfies Lemma C.2. Indeed, the proof only uses the fact that $D_{n,m}$ is a DFW that recognizes $R_{n,m}$ with a single accepting state p. Thus, we only need to prove that g^a satisfies Lemma C.1. That is, for all $u \in (X \cdot C)^*$ and $v \in R_{n,m}$, if $g^a(u) = p$, then there is a prefix $y \in (X \cdot C)^*$ of v, such that $g^a(u \cdot y) = q_0$. Consider such words u and v, and let $j \in [m]$ be such that c_j is the last letter of v.

Let $k \in [n]$ be the minimal index for which $c_j \in C^{i_k}_k$, and let $z \in (X \cdot C)^*$ be a prefix of v such that $v = z \cdot x_k \cdot c_j \cdot x_{k+1} \cdot c_j \cdots x_n \cdot c_j$. If there exists a prefix $y \in (X \cdot C)^*$ of z such that $g^a(u \cdot y) = q_0$, then we are done. Otherwise, the finite run of $\mathcal{A}^a_\varphi$ on z from p, returns back to p, and hence $g^a(u \cdot z) = p$. Now $g^a(u \cdot z \cdot x_k) = q^{i_k}_k$, and since $x_k = i_k$ satisfies c_j we have $g^a(u \cdot z \cdot x_k \cdot c_j) = q_0$. Thus, we may take $y = z \cdot x_k \cdot c_j$ which is a prefix of v, and we are done. □