Resolving Nondeterminism with Randomness

Thomas A. Henzinger
Institute of Science and Technology Austria, Klosterneuburg, Austria

Aditya Prakash
University of Warwick, UK
LIS, Aix-Marseille Université, France

K. S. Thejaswini
Institute of Science and Technology Austria, Klosterneuburg, Austria

Abstract

We define and study classes of ω-regular automata for which the nondeterminism can be resolved by a policy that uses a combination of memory and randomness on any input word, based solely on the prefix read so far. We examine two settings for providing the input word to an automaton. In the first setting, called *adversarial resolvability*, the input word is constructed letter-by-letter by an adversary, dependent on the resolver's previous decisions. In the second setting, called *stochastic resolvability*, the adversary pre-commits to an infinite word and reveals it letter-by-letter. In each setting, we require the existence of an almost-sure resolver, i.e., a policy that ensures that as long as the adversary provides a word in the language of the underlying nondeterministic automaton, the run constructed by the policy is accepting with probability 1.

The class of automata that are adversarially resolvable is the well-studied class of history-deterministic automata. The case of stochastically resolvable automata, on the other hand, defines a novel class. Restricting the class of resolvers in both settings to stochastic policies without memory introduces two additional new classes of automata. We show that the new automata classes offer interesting trade-offs between succinctness, expressivity, and computational complexity, providing a fine gradation between deterministic automata and nondeterministic automata.

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

Keywords and phrases ω-regular languages, History determinism, Stochastic strategies

Digital Object Identifier 10.4230/LIPIcs.MFCS.2025.57

Related Version *Full Version*: `https://arxiv.org/abs/2502.12872`

Funding This work is a part of project VAMOS that has received funding from the European Research Council (ERC), grant agreement No 101020093.

1 Introduction

The trade-offs between determinism and nondeterminism is a central theme in automata theory. For automata over infinite words, nondeterministic Büchi automata are strictly more expressive than their deterministic counterpart. In fact, they are as expressive as deterministic parity automata, and at the same time, are exponentially more succinct [27, 30]. Even though nondeterminism seems attractive because of these favourable qualities, it presents difficulties in contexts like reactive synthesis or runtime verification when the specifications are expressed as automata. The fundamental challenge here arises because any algorithm operating with nondeterministic automata needs to account for all possible future inputs.

Several attempts have evolved that involve modifying algorithms to avoid or minimise the blow-up using determinisation procedures [24, 25, 17]. Many recent attempts instead focus on classes of automata that aim to bridge this gap between determinism and nondeterminism, while avoiding exponential determinisation procedures. The latter of the attempts have lead to the introduction of several classes of automata like history-deterministic (HD) automata [20], good-for-MDP automata [19], and tight Büchi automata [22], to name a few.

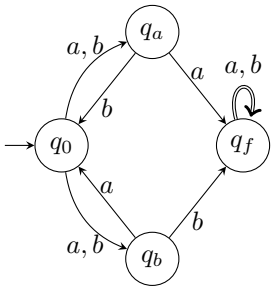


Figure 1 A reachability automaton that is not HD.

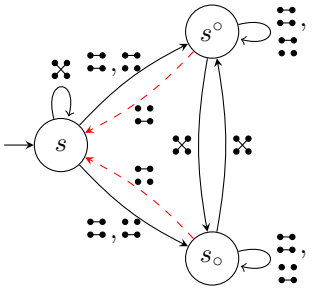


Figure 2 A HD coBüchi automaton where nondeterminism can be resolved randomly.

The notion of history determinism, in particular, which has inspired this work, has gained significant attention in recent years. History-deterministic automata are nondeterministic automata whose nondeterminism can be resolved on-the-fly, only based on the prefix of a word read so far. More precisely, history determinism of an automaton can be characterised by the following history-determinism game (HD game) on that automaton. The HD game is played between two players, Eve and Adam, on an arena of the automaton where Eve initially has a token at the start state. In each round of a play, Adam picks a letter, and Eve moves her token along a transition on the letter picked by Adam, thereby constructing a run on the word that Adam has picked so far. The game proceeds for an infinite duration, where Adam constructs an infinite word, and Eve produces a run on this word. Eve wins if she has a strategy that ensures that she constructs an accepting run whenever the infinite word constructed by Adam is in the language of the automaton. HD automata are defined as automata in which Eve wins the HD game.

Games whose winning objectives are given by HD automata can be solved as efficiently as if they were given by deterministic automata, and thus they were dubbed as good-for-games automata at the time of its introduction [20, Theorem 3.1]. This property makes history-deterministic automata relevant for the problems of reactive synthesis and model checking [11, Section 7]. Furthermore, they are exponentially more succinct than deterministic automata [23, Theorem 1], and history-deterministic coBüchi automata give rise to a canonical representation of coBüchi as well as ω-regular languages [1, 16].

Although history determinism is a useful concept, it is based on a highly adversarial setting for the player who resolves the nondeterminism. In the context of reactive synthesis, specifications written using history-deterministic automata can be used directly to synthesise reactive systems such that the synthesised system satisfy the specification against *all* possible adversarial environment actions. However, this might be an *overly restrictive* notion in settings where the synthesised systems react with an environment that is not privy to its current internal state. Furthermore, the strategies to resolve nondeterminism from such automata need to be explicitly maintained as controllers of the synthesised system, but HD automata require resolvers that use exponential memory [23].

For example, consider the infinite word reachability automaton in Figure 1, which accepts all infinite words over $\{a, b\}$. In order to resolve the nondeterministic decision in state q_0 and eventually reach state q_f, the next letter that is to be read must be guessed. If successive letters are chosen adversarially and based on the current state of the automaton, no resolver can successfully resolve the nondeterminism on all words. Therefore, this automaton is not HD. However, in a more lenient setting where the adversary pre-commits to an infinite word

(or equivalently, does not have access to the current internal state), a resolver that resolves the nondeterminism by choosing uniformly at random one of the two transitions from q_0 produces an accepting run almost surely (with probability 1).

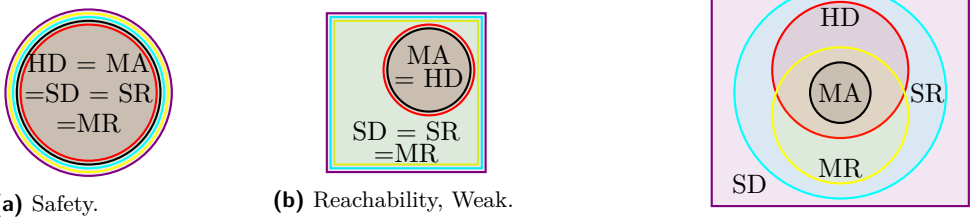
Allowing for stochastic resolvers can also simplify the resolvers for HD automata. Consider the coBüchi automaton in Figure 2 inspired from the work of Kuperberg and Skrzypczak [23, Theorem 1]. Accepting runs in this automaton contain only finitely many rejecting (red, dashed) transitions. The language of the automaton is over the alphabet {⁘, ⁖, ⁙, ⁚} and therefore, each word over this alphabet naturally constructs a graph, by recognising the right dots of a letter with the left dots of the next letter. For instance, the finite word ⁘⁖⁚⁖⁘ represents the graph ⁘⁙⁘, which has two distinct finite paths of length 4. The infinite word $(⁘⁖⁙⁖⁘)^\omega$ represents a graph with two infinite paths. However, the word $(⁘⁖⁙⁖)^\omega$ does not have any infinite path, as the letter ⁚ breaks each path infinitely often. It can be verified that the automaton in Figure 2 accepts an infinite word if and only if the corresponding graph has at least one infinite path. Indeed, the two nondeterministic choices correspond to verifying if the path starting from the "bottom dot" is infinite or if the path starting from the "top dot" is infinite, respectively. A correct guess ensures that rejecting transitions never occur in the run and a wrong guess brings the run back to the start vertex from which a guess needs to be made again. This automaton is HD. Consider the resolver that chooses the transition from state s that follows the longest unbroken path so far, in order to verify that it extends to an infinite path. If there is an infinite path then eventually it would be the longest unbroken path, and such a resolver would correctly resolve the nondeterminism to produce an accepting run. The simpler resolver that selects a transition from s uniformly at random also constructs an accepting run on any infinite word in the language *almost surely* (with probability 1), even if letters are chosen adversarially. This is because the run constructed by this resolver would almost surely eventually coincide with one of the longest unbroken paths.

In line with these examples, we introduce classes of automata inspired by history determinism, extending them in two ways. First, we consider resolvers that use both memory *and stochasticity*, and second, we study settings where the letter-giving adversary commits to a word in advance, or equivalently, is unaware of the current state.

New classes of automata. Intuitively, stochastic resolvers propose an outgoing transition using both memory and randomness. We study classes of automata for which there is an *almost-sure* resolver, a policy for resolving the nondeterminism (which can use memory or randomness) such that for all words in the language, the run produced is almost-surely accepting, that is, with probability 1.

We consider the existence of almost-sure resolvers in the following two settings for the letter-giving adversary. The first setting, called *adversarial resolvability*, is where the input words to the automaton are from an adversary that generates the input letter-by-letter based on the resolution of the nondeterminism in the past. The resolvers here represent the strategy of Eve in the HD game defined for history determinism, where we allow for a larger class of resolvers which allow for randomness. The second, and novel setting, called *stochastic resolvability*, is where the adversary commits to an entire input-word, but only reveals the input letter-by-letter to their opponent who is resolving the nondeterminism.

We call the class of automata that have an almost-sure resolver in the adversarial resolvability setting *adversarially resolvable automata*, and that have an almost-sure resolver in the stochastic resolvability setting as *stochastically resolvable automata*, or SR automata, for short. We further study the classes of automata where weaker resolvers are used. If the resolvers of nondeterminism are restricted to policies where only stochasticity is used, then



(a) Safety. **(b)** Reachability, Weak. **(c)** Büchi, co-Büchi, and Parity.

Figure 3 Landscape of the automata classes.

the resolving strategy is just a probability distribution among the outgoing transition for each state. We call such classes of resolvers *memoryless resolvers*, and the class of automata that are adversarially resolvable using memoryless almost-sure resolvers *memoryless-adversarially resolvable*, or MA for short. Likewise, we call the class of automata that are stochastically resolvable using memoryless almost-sure resolvers *memoryless-stochastically resolvable*, or MR for short. The class of adversarially resolvable automata, without any restrictions on resolvers, are equivalent to HD automata due to determinacy of ω-regular games.

SR automata form intermediate class of nondeterministic automata that can be used without determinisation during reactive synthesis or runtime verification in settings where the environment has no access to the system's internal states. Both the memoryless variants MA and MR, are also practically relevant subclasses of HD and SR, respectively, since they have memoryless resolvers which can serve as simple controllers for specifications represented by such automata (as opposed to the exponential memory required by their more general counterparts). Converting temporal logic specifications to HD and other automata is an ongoing research area [15], and our new classes like MR and SR provide a larger target for such conversions to aim for.

Our results. We introduce and then make comparisons between the three newly introduced classes of automata, and also with existing notions such as HD automata and semantically deterministic (SD) automata [2] (see Section 3). We study the succinctness, expressivity, and computational complexity of the class-membership problem for these automata classes.
In Section 3, we compare our novel automata classes with each other and also with HD and SD automata. We show separating examples or prove the equivalences between the classes (Theorem 6). A landscape of these automata can be found in Figure 3. In the same section, we show the *exponential succinctness of SR coBüchi to deterministic coBüchi automata, and of SR Büchi automata to HD Büchi automata (Theorem 8)*. For coBüchi automata, we show the surprising result that *every SR (and thus, every HD) coBüchi automaton can be converted into an MA automaton with the same number of states* (Theorem 7). As a corollary, although in HD coBüchi automata, Eve might need exponential memory resolvers to surely win the HD game [23, Theorem], we can efficiently transform it into an automaton where a memoryless random resolver suffices for Eve to win almost-surely in the resultant coBüchi automaton. Therefore, SR coBüchi automata are as succinct as HD coBüchi automata, while Büchi SR automata are exponentially succinct (Theorem 8).

In Section 4, we tackle the problem of expressivity. HD (and therefore also MA) parity automata that use priorities from $\{i, i+1, \ldots, j\}$ are as expressive as deterministic automata that use the same set of priorities [8]. We show that the same holds for SR and MR automata (Theorem 22), and thus, the *parity-index hierarchy is strict for SR and MR automata.*

Table 1 Complexity of checking membership of an automaton in a class.

	Safety	Reachability/Weak	Buchi	coBüchi	Parity
Checking MA (Theorem 23)	P	P	NP	NP	NP-complete
Checking HD	P [23, 10]	P [23, 10]	P [23]	P [4]	PSPACE [26], NP-hard [28]
Checking SR/ MR (Theorem 24)	P	PSPACE-comp	Open	Open	Open
Resolver checking for SR/MR (Theorem 24)	P	PSPACE-comp	undec.	undec.	undec.

In Section 5, we tackle the problems of deciding if an automata is MA and SR. The exact complexity of the problem of deciding if an automaton is HD is nearly a two-decade old open problem [20], with recent improvements giving the problem a NP-hardness lower bound [28] and a PSPACE upper bound [26, 2-Token Theorem]. We show that *checking if a given automaton is MA is* NP*-complete* (Theorem 23). The upper bound relies on showing that a slight modification of the 2-token game [4, 26], a game used to characterise HD automata, also characterises MA automata. With such a characterisation, we show that to check if an automata is MA, one first needs to guess a strategy in the modified two-token game. Checking correctness of this strategy reduces to solving a Markov decision process (MDP) with Muller objectives that are represented by a Zielonka DAG. We show that this can be computed in polynomial time (Theorem 27) known previously only known for a restricted subclass of Muller objectives [13]. We also consider problems related to checking whether an automaton is in the class MR and SR and summarise these results in Table 1.

2 Automata and resolution of nondeterminism

We use $\mathbb{N}$ to denote the set of natural numbers $\{0, 1, 2, \dots\}$. For two natural numbers i, j with $i < j$, we use $[i, j]$ to denote the set $\{i, i+1, \dots, j\}$ consisting of natural numbers that are at least i and at most j. For a natural number i, we use $[i]$ to denote the set $[0, i]$. We use $\mathbb{U}$ to denote the unit interval, i.e., $\mathbb{U} = \{x \mid 0 \leqslant x \leqslant 1\}$. For a finite set X, a probability distribution on X is a function $f : X \to \mathbb{U}$ that maps each element in X to a number in $\mathbb{U}$, such that $\sum_{a \in X} f(X) = 1$ and write $\mathsf{Distributions}(X)$ for the set of probability distributions.

Parity automata. An $[i, j]$-*nondeterministic parity automaton* $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$, $[i, j]$-parity automaton for short, consists of a finite directed graph with edges labelled by letters in Σ and *priorities* in $[i, j]$ for some $i, j \in \mathbb{N}$ with $i < j$. The set of *states* Q constitutes the vertices of this graph, and the set of *transitions* $\Delta \subseteq Q \times \Sigma \times [i, j] \times Q$ represents the labelled edges of the graph. Each automaton has a designated *initial state* $q_0 \in Q$. For states $p, q \in Q$ and a letter $a \in \Sigma$, we use $p \xrightarrow{a:c} q$ to denote a transition from p to q on the letter a that has the priority c. We assume that for any two states p, q and letter a, there is at most one transition of the form $p \xrightarrow{a:c} q$. We also assume our automata to be *complete*, i.e., for each state and letter, there is at least one transition from that state on that letter.

A *run* on an infinite word w in Σ^ω is an infinite path in the automaton, starting at the initial state and following transitions that correspond to the letters of w in sequence. A run is *accepting* if the highest priority occurring infinitely often amongst the transitions of that run is even, and a word w in Σ^ω is accepting if the automaton has an accepting run on w. The *language* of an automaton $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of words that it accepts. We

write that the automaton $\mathcal{A}$ *recognises* a language L if $\mathcal{L}(\mathcal{A}) = L$. A parity automaton $\mathcal{A}$ is said to be *deterministic* if for any given state in $\mathcal{A}$ and any given letter in Σ, there is exactly one transition from that state on that letter.

We will say that $[i, j]$ with $i = 0$ or 1 is the *parity index* of $\mathcal{A}$. A *Büchi* (resp. co-Büchi) automaton is a $[1, 2]$ (resp. $[0, 1]$) parity automaton. A *safety automaton* is a Büchi automaton where all transitions with priority 1 occur as self-loops on a sink state. Dually, a *reachability automaton* is a Büchi automaton $\mathcal{A}$ such that all transitions with priority 2 in $\mathcal{A}$ occur as self-loops on a sink state. A *weak automaton* is a Büchi automaton, in which there is no cycle that contains both a priority 2 transition and a priority 1 transition.

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

Probabilistic automata. A probabilistic parity automaton $\mathcal{P} = (Q, \Sigma, \Delta, \rho, q_0)$ – a natural extension of probabilistic Büchi automata defined by Baier, Größer, and Bertrand [6] – has the semantics of a parity automaton. Additionally, we assign a probability to each transition in Δ using the function $\rho : \Delta \to \mathbb{U}$, such that for each state and each letter, the sum of $\rho(\delta)$s for outgoing transitions δ from that state on that letter add up to 1. We write $\Delta_{q,a}$ to denote the set of outgoing transitions from the state q on the letter a.

Given a probabilistic automaton $\mathcal{P}$ as above, the behaviour of $\mathcal{P}$ on an input word w is formalised by an infinite Markov chain that captures all the possible runs of $\mathcal{P}$ on w. For the word $w = a_0a_1a_2\ldots$, consider the Markov chain M_w defined over the vertices $Q \times \mathbb{N}$. After "processing" the finite word $a_0a_1\ldots a_{i-1}$, the Markov chain will be at some state (q, i), where q is a state that can be reached from q_0 on the word $a_0a_1\ldots a_{i-1}$. The run of the Markov chain then moves from (q, i) to the state $(p, i + 1)$ with probability $\rho(\delta)$, where $\delta = q \xrightarrow{a_i:c_i} p$ is a transition in $\mathcal{A}$. The initial state of M_w is $(q_0, 0)$, and we say that a run in M_w is accepting if the run corresponding to ρ in $\mathcal{A}$ is accepting.

For an input word w and a probabilistic automaton $\mathcal{P}$, we define the probability $\texttt{Prob}_{\mathcal{P}}(w)$ to be the probability measure of accepting runs in M_w. We mostly deal with almost-sure semantics of probabilistic automata, and therefore, we refer to language of a probabilistic automaton $\mathcal{P}$ as $\mathcal{L}(\mathcal{P}) = \{w \in \Sigma^\omega \mid \texttt{Prob}_{\mathcal{P}}(w) = 1\}$.

Resolution of nondeterminism

We will deal with nondeterministic automata where the nondeterminism can be resolved using a combination of memory and randomness. We begin by recalling history-deterministic automata, which is characterised by the following history-determinism game.

▶ **Definition 1** (History-determinism game). *Given a nondeterministic parity automaton $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$, the* history-determinism (HD) game *on $\mathcal{A}$ is a two-player game between Eve and Adam that starts with Eve's token at q_0 and proceeds for infinitely many rounds. For each $i \in \mathbb{N}$, round i starts with Eve's token at a state q_i in Q, and proceeds as follows. (1) Adam selects a letter $a_i \in \Sigma$; (2) Eve selects a transition $q_i \xrightarrow{a_i:c_i} q_{i+1} \in \Delta$ along which she moves her token. Eve's token then is at q_{i+1} from where the round $(i + 1)$ is played. Thus, in the limit of a play of the HD game, Adam constructs a word letter-by-letter, and Eve constructs a run on her token transition-by-transition on that word. Eve wins such a play if Adam's word is not in $\mathcal{L}(\mathcal{A})$ or if the run on her token is accepting.*

An automaton is *history deterministic* (HD) if Eve has a winning strategy in the HD game on $\mathcal{A}$. If an automaton is HD then Eve has a finite memory winning strategy in its HD game [29, Theorem 3.12], that is, a mapping from the set of finite plays of the HD game

to the next transition that Eve chooses, where the mapping is only based on the current position and a finite memory structure. We will formalise this concept as *pure resolvers*, but we first define resolvers where we allow for stochasticity as well.

Resolver. For a nondeterministic parity automaton $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$, a *stochastic resolver*, or just a *resolver*, for $\mathcal{A}$ is given by $\mathcal{M} = (M, m_0, \mu, \mathsf{nextmove})$, where M is a finite set of *memory states*, m_0 is the initial memory state. The function $\mathsf{nextmove}$ assigns to every three-tuple of memory state m, state q of $\mathcal{A}$, and letter a in Σ, a probability distribution $\mathsf{nextmove}(m, q, a)$ in $\mathsf{Distributions}(\Delta_{q,a})$, where $\Delta_{q,a}$ is the set of outgoing transitions from state q on letter a. The function μ is the transition function and is given by $\mu : M \times \Delta \to M$.

Eve plays in the HD game on $\mathcal{A}$ using the resolver $\mathcal{M}$ as follows. At state q, when the memory state is m, suppose Adam chooses the letter a. Eve then selects an outgoing transition δ from q on a with the probability $(\mathsf{nextmove}(m, q, a)) \circ (\delta)$, and updates the memory state to $m' = \mu(m, \delta)$. We say that a resolver $\mathcal{M}$ for Eve constitutes a winning strategy in the HD game on $\mathcal{A}$ if she wins almost-surely (i.e., with probability 1) when she plays using $\mathcal{M}$ as described above. We say that a resolver $\mathcal{M}$ is *pure*, if for each memory state m, each state q, and letter a, the probability distribution $\mathsf{nextmove}(m, q, a)$ assigns probability 1 to some outgoing transition in $\Delta_{q,a}$ and probability 0 to every other transition.

In ω-regular games, which HD games are a subclass of, Eve has a strategy to win with positive probability if and only if she has a pure strategy to win surely [12] (see [7, Theorem 8.3] for a modern exposition). Thus, if Eve has a stochastic resolver to win the HD game on $\mathcal{A}$ almost surely, then she also has a pure resolver to win surely. We say that a resolver is *memoryless* if that resolver has one state.

An automaton $\mathcal{A}$ is *memoryless-adversarial resolvable* (MA) if there is a memoryless resolver using which Eve wins the HD game on $\mathcal{A}$ almost-surely, i.e., with probability 1.

If there is a memoryless resolver $\mathcal{M}$ for Eve using which she wins the HD game almost surely, then she wins almost surely using any memoryless resolver $\mathcal{M}'$ that assigns nonzero probabilities to the same transitions as $\mathcal{M}$.

Nonadversarial resolvability. We introduce a less-adversarial notion than adversarial resolvability discussed earlier. For a parity automaton $\mathcal{A}$ and a resolver $\mathcal{M}$ for $\mathcal{A}$, we say that $\mathcal{M}$ is a *almost-sure resolver* for $\mathcal{A}$ if for every word w in $L(\mathcal{A})$, the run constructed on w using $\mathcal{M}$ in the HD game on $\mathcal{A}$ is almost-surely accepting.

More concretely, consider the probabilistic automaton $\mathcal{P} = \mathcal{M} \circ \mathcal{A}$ that is obtained by *composing* the resolver $\mathcal{M}$ with $\mathcal{A}$. That is, the states of $\mathcal{P}$ are $Q \times M$, the initial state of $\mathcal{P}$ is $p_0 = (q_0, m_0)$. The automaton $\mathcal{P}$ has the transition $(q, m) \xrightarrow{a:c} (q', m')$ of probability p if $\delta = q \xrightarrow{a:c} q'$ is a transition in $\mathcal{A}$, $(\mathsf{nextmove}(m, q, a)) \circ \delta = p$ and $m' = \mu(m, \delta)$. Then, we write that $\mathcal{M}$ is an *almost-sure resolver* for $\mathcal{A}$ if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{P})$. For $\mathcal{M}$, $\mathcal{A}$, and $\mathcal{P}$ as above, we call $\mathcal{P}$ the *resolver-product* of $\mathcal{M}$ and $\mathcal{A}$.

An automaton $\mathcal{A}$ is *stochastically resolvable* (SR) if there is an almost-sure resolver for $\mathcal{A}$. An automaton $\mathcal{A}$ is *memoryless stochastically resolvable* (MR) if there is an almost-sure resolver for $\mathcal{A}$ that is memoryless.

▶ **Example 2.** Consider the example of the reachability automaton $\mathcal{A}$ in Figure 1, where the accepting transitions are double-arrowed. We will show that $\mathcal{A}$ is MR but not HD. This automaton accepts all infinite words over the alphabet $\{a, b\}$. At the only nondeterministic

state q_0 on reading a or b, if the resolver correctly guesses the next letter, then it will reach the accepting state. Consider a resolver that chooses the transitions to q_a and q_b with $\frac{1}{2}$ probability. For any fixed infinite word, this resolver produces an accepting run that will almost surely, eventually guess the letter one step ahead correctly. However, Adam has a winning strategy in the HD game on $\mathcal{A}$: whenever Eve's token in the HD game is at q_b, Adam chooses the letter a and chooses the letter b whenever Eve's token is at q_a. This adversarial choice of letters ensures that Eve's run is rejecting, and thus Adam wins the HD game on $\mathcal{A}$.

We next describe an equivalent game-based definition for SR and MR automata.

Stochastic resolvability game. The stochastic resolvability game (SR game) on an automaton $\mathcal{A}$ proceeds similarly to the HD game. In each round, Adam selects a letter and then Eve responds with a transition of $\mathcal{A}$ on that letter; thus, in the limit, Adam constructs an infinite word and Eve constructs a run on that word. Eve wins if her transitions form an accepting run whenever Adam's word is in the language. However, unlike the HD game, Adam does not observe Eve's run, and therefore his strategy must not depend on the position of Eve's token in the automaton. We define this game more formally as a partial-observation game in the full version of the paper, where we also prove the following result.

▶ **Lemma 3.** *For every parity automaton $\mathcal{A}$, a resolver $\mathcal{M}$ is an almost-sure resolver for $\mathcal{A}$ if and only if $\mathcal{M}$ is a finite-memory strategy that is almost-surely winning for Eve against all strategies of Adam in the SR game on $\mathcal{A}$.*

3 Comparisons between the different notions

In this section, we compare the notions of SR, MR, and MA with each other and other related notions of nondeterminism in the literature.

We start with the notion of semantic determinism. These were originally introduced as residual automata [23], but we follow the more recent works that call them semantically deterministic (SD) automata instead [3, 2]. A transition δ from p to q on a letter a in a parity automaton $\mathcal{A}$ is *language-preserving* if $\mathcal{L}(\mathcal{A}, q) = a^{-1}\mathcal{L}(\mathcal{A}, p)$. A parity automaton is *semantically deterministic* (SD) if all its transitions are language-preserving. The following observation concerning SD automata can be shown by an inductive argument on word length.

▶ **Observation 4.** *For every SD automaton $\mathcal{A}$, all states in $\mathcal{A}$ that can be reached from a state q upon reading a finite word u recognise the language $u^{-1}\mathcal{L}(\mathcal{A}, q)$.*

We observe that all SR automata are SD, up to removal of some transitions.

▶ **Lemma 5.** *Every SR parity automaton $\mathcal{A}$ has a language-equivalent SD subautomaton $\mathcal{B}$.*

Proof sketch. We fix an almost-sure resolver $\mathcal{M}$ for $\mathcal{A}$, and let $\mathcal{B}$ be the subautomaton consisting of transitions that $\mathcal{M}$ takes with nonzero probability. We show that every transition in $\mathcal{B}$ is language-preserving, and thus, $\mathcal{B}$ is SD and language equivalent to $\mathcal{A}$. ◀

A *pre-semantically deterministic* (pre-SD) automaton is an automaton that has a language-equivalent SD subautomaton. The following result gives a comprehensive comparison between the notions of nondeterminism we have discussed so far. The results of Theorem 6 are summarised by the Venn diagram in Figure 3.

▶ **Theorem 6.**

1. *For safety automata, the notions of pre-SD, SR, MA, MR, and HD are all equivalent.*
2. *For reachability and weak automata, the following statements hold.*
   a. *Pre-SD, SR, and MR are equivalent and are strictly larger classes than HD automata.*
   b. *HD and MA are equivalent notions.*
3. *For Büchi, coBüchi, and parity automata, the following statements hold.*
   a. *Pre-SD automata are a strictly larger class than SR automata.*
   b. *SR automata are a strictly larger class than HD automata.*
   c. *SR automata are a strictly larger class than MR automata.*
   d. *There are HD automata that are not MR, and there are MR automata that are not HD.*
   e. *Both HD and MR automata are strictly larger classes than MA automata.*

Even though the five notions of nondeterminism discussed for coBüchi automata are all different, we show that every SR coBüchi automaton (the second largest class after SD) can be efficiently converted to a language-equivalent MA coBüchi automaton (smallest class).

▶ **Theorem 7.** *There is a polynomial-time algorithm that converts SR coBüchi automata with n states into language-equivalent MA coBüchi automata with at most n states.*

SD coBüchi are exponentially more succinct than HD coBüchi automata [2, Theorem 14], and therefore, the above result does not hold for SD coBüchi automata. However, we show the succinctness of SR Büchi automata against HD Büchi automata and MA coBüchi automata against deterministic coBüchi automata.

▶ **Theorem 8.**

1. *There is a class of languages L_n such that, there are MR Büchi automata recognising L_n with $\mathcal{O}(n)$ states, and any HD Büchi automaton recognising L_n requires at least 2^n states.*
2. *There is a class of languages L'_n, such that there are MA coBüchi automata recognising L'_n with $\mathcal{O}(n)$ states and any deterministic coBüchi automaton recognising L'_n requires at least $\Omega(2^n/2n+1)$ states.*

We organise the proofs of Theorems 6–8 based on the acceptance conditions.

3.1 Safety automata

We showed in Lemma 5 that every SR automaton is SD. The next result shows that every safety automaton $\mathcal{S}$ is SD if and only if $\mathcal{S}$ is determinisable-by-pruning, that is, $\mathcal{S}$ contains a language-equivalent deterministic subautomaton.

▶ **Lemma 9** (Folklore). *Every SD safety automaton is determinisable-by-pruning.*

Observe that any determinisable-by-pruning automaton is MA, where Eve's strategy is to take transitions in a fixed language-equivalent deterministic subautomaton. Thus, Lemma 9 implies the following result.

▶ **Lemma 10.** *The notions of pre-SD, SR, MA, MR, and HD coincide on safety automata.*

3.2 Reachability and weak automata

We compare notions of nondeterminism on reachability and weak automata. Recall that Example 2 shows an MR reachability automaton that is not HD.

▶ **Lemma 11.** *There is an MR reachability automaton that is not HD.*

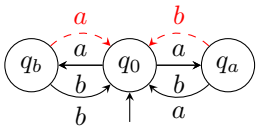


Figure 4 SD but not SR coBüchi automaton. Dashed arrows denote rejecting transitions.

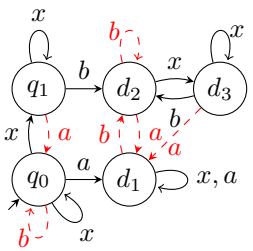


Figure 5 A HD coBüchi automaton that is not MR. Dashed arrows denote rejecting transitions.

We show that SD weak automata are MR, by proving that the resolver that selects transitions uniformly at random in an SD weak automaton $\mathcal{A}$ is an almost-sure resolver.

▶ **Lemma 12.** *Every SD weak automaton is MR.*

From Lemmas 5 and 11, we obtain that SR weak automata are also MR and that they strictly encompass HD automata. HD weak automata (and also MA automata) are determinisable-by-pruning (DBP) [8], and thus, we obtain the following result.

▶ **Lemma 13.** *The notions of DBP, HD and MA coincide on weak automata.*

3.3 CoBüchi automata

We now compare our notions of nondeterminism on coBüchi automata. We will show that no two notions amongst SD, SR, HD, MR, and MA are equivalent for coBüchi automata. We will then give a polynomial-time algorithm that converts SR coBüchi automata with n states into language-equivalent MA coBüchi automata with at most n states. This shows that SR and MR coBüchi automata are no more succinct than MA (and also HD) coBüchi automata.

Consider the automaton in Figure 4, which is an SD coBüchi automaton that recognises all words. This automaton is not SR, however, since Adam choosing a letter among $\{a, b\}$ with probability 1/2 in the SR game (see Lemma 3) ensures that Eve cannot almost-surely construct an accepting run.

▶ **Lemma 14.** *There is an SD coBüchi automaton that is not SR.*

In Section 3.2, we showed an MR reachability automaton that is not HD (Lemma 11). It follows that there is a MR coBüchi automaton that is not HD. We next show a HD coBüchi automaton that is not MR, and hence, also not MA.

▶ **Lemma 15.** *There is a HD coBüchi automaton that is not MR.*

Proof sketch. Consider the coBüchi automaton $\mathcal{C}$ shown in Figure 5. The automaton $\mathcal{C}$ has nondeterminism on the letter x in the initial state q_0. Informally, Eve, from the state q_0 in the HD game or the SR game, needs to "guess" whether the next sequence of letters till an a or b is seen forms a word in x^*a or in x^+b. The automaton $\mathcal{C}$ recognises the language $L = (x + a + b)^*((x)^\omega + (x^*a)^\omega + (x^+b)^\omega)$. $\mathcal{C}$ is HD: Eve's strategy to select the x-self loop on q_0 or to move to q_1 based on whether the last letter distinct from x was an a or b, respectively, is a winning strategy in the HD game on $\mathcal{A}$. To show that $\mathcal{C}$ is not MR, we first show that any memoryless resolver for $\mathcal{C}$ must take both x-outgoing transitions from q_0 with positive probability, and then show that no such memoryless resolver accepts the infinite word $x^2ax^3ax^4a\ldots$ with probability 1, as desired. ◀

We have thus shown so far that each of the five classes in Figure 3 are different for coBüchi automata. We now show that every SR coBüchi automaton can be converted into a language-equivalent MA coBüchi automaton without any additional states.

▶ **Theorem 7.** *There is a polynomial-time algorithm that converts SR coBüchi automata with n states into language-equivalent MA coBüchi automata with at most n states.*

We next describe a proof sketch of Theorem 7, throughout which we fix an SR coBüchi automaton $\mathcal{A}$. We first relabel the priorities on $\mathcal{A}$ to obtain $\mathcal{C}$ as follows. Consider the graph consisting of all states of $\mathcal{A}$ and 0 priority transitions of $\mathcal{A}$. For any 0 priority transition of $\mathcal{A}$ that is not in any strongly connected component (SCC) in this graph, we change that transition to have priority 1 in $\mathcal{C}$. Priorities of other transitions in $\mathcal{A}$ is preserved as is in $\mathcal{C}$. This relabelling of priorities does not change the acceptance of any run (Appendix B in full version), and thus, $\mathcal{C}$ is SR and language-equivalent to $\mathcal{A}$. We next introduce a few notions to describe a proof sketch of Theorem 7.

Safe-approximation. For the automaton $\mathcal{C}$, define its safe-approximation, denoted $\mathcal{C}_{\texttt{safe}}$, as the safety automaton with the same states as $\mathcal{C}$, plus an additional rejecting sink state. Transitions of priority 0 in $\mathcal{C}$ are preserved in $\mathcal{C}_{\texttt{safe}}$ with the same priority. Transitions of priority 1 are redirected to the rejecting sink state and have priority 1.

Weak-coreachability. We call two states p, q in $\mathcal{C}$ as coreachable, denoted by $p, q \in \mathsf{CR}(\mathcal{C})$, if there is a finite word u on which there are runs from the initial state of $\mathcal{C}$ to p and q. We denote the transitive closure of this relation as *weak-coreachability*, which we denote by $\mathsf{WCR}(\mathcal{C})$. Note that weak-coreachability is an equivalence relation.

SR covers. For two parity automata $\mathcal{B}$ and $\mathcal{B}'$, we say that $\mathcal{B}$ SR-covers $\mathcal{B}'$, denoted by $\mathcal{B} \succ_{SR} \mathcal{B}'$, if Eve has an almost-sure winning strategy in the modified SR game as follows. Eve, similar to the SR game on $\mathcal{B}$, constructs a run in $\mathcal{B}$, but Eve wins a play of the game if, in that play, Eve's constructed run in $\mathcal{B}$ is accepting whenever Adam's word is in $\mathcal{L}(\mathcal{B}')$.

SR self-coverage. We say that a coBüchi automaton $\mathcal{B}$ has *SR self-coverage* if for every state q there is another state p that is coreachable to q in $\mathcal{C}$ such that $(\mathcal{B}_{\texttt{safe}}, p)$ SR-covers $(\mathcal{B}_{\texttt{safe}}, q)$. The crux of Theorem 7 is in proving the following result.

▶ **Lemma 16.** *The coBüchi automaton $\mathcal{C}$ has SR self-coverage.*

SR-covers is a transitive relation, i.e., if $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ are nondeterministic parity automata such that $\mathcal{A}_1 \succ_{SR} \mathcal{A}_2$ and $\mathcal{A}_2 \succ_{SR} \mathcal{A}_3$, then $\mathcal{A}_1 \succ_{SR} \mathcal{A}_3$. The following result then follows from the definition of SR self-coverage and the fact that $\mathcal{C}$ has finitely many states.

▶ **Lemma 17.** *For every state q in $\mathcal{C}$, there is another state p weakly coreachable to q in $\mathcal{C}$ such that $(\mathcal{C}_{safe}, p)$ SR-covers $(\mathcal{C}_{safe}, q)$ and $(\mathcal{C}_{safe}, p)$ SR-covers $(\mathcal{C}_{safe}, p)$.*

Note that if $(\mathcal{C}_{\texttt{safe}}, p)$ SR-covers $(\mathcal{C}_{\texttt{safe}}, p)$ then $(\mathcal{C}_{\texttt{safe}}, p)$ is SR. Since SR automata are semantically deterministic (Lemma 5) and SD safety automata are determinisable-by-pruning (Lemma 9), we call such states p as *safe-deterministic*.

We build a MA automaton $\mathcal{H}$, whose states are the safe-deterministic states in $\mathcal{C}$. This construction is similar to the one used by Kuperberg and Skrzypczak for giving a polynomial time procedure to recognise HD coBüchi automata [23, Section E.7 in the full version]. We fix a uniform determinisation of transitions from each safe-deterministic state in $\mathcal{C}_{\texttt{safe}}$ and

we add these transitions in $\mathcal{H}$ with priority 0. If there are no outgoing transitions in $\mathcal{H}$ from the state p on letter a so far, then we add outgoing transitions from p on a as follows. Let q be a state in $\mathcal{C}$ such that there is a transition from p to q on a in $\mathcal{C}$. For each state r that is weakly coreachable to q in $\mathcal{C}$ and that is safe-deterministic, we add a transition from q to r in $\mathcal{H}$ with priority 1. This concludes our construction of $\mathcal{H}$.

We show that $\mathcal{H}$ is MA by showing that Eve wins the HD game on $\mathcal{H}$ almost-surely by picking transitions uniformly at random (Appendix B in full version). Both this fact and the language equivalence of $\mathcal{H}$ to $\mathcal{C}$ primarily relies on Lemma 17. Since a safety automaton is DBP if and only if that automaton is HD, and every HD safety automaton can be determinised in polynomial-time [10], the safe-deterministic states of $\mathcal{C}$ can be identified, and outgoing safe transitions from these states can be determinised in polynomial-time. Thus, the construction of $\mathcal{H}$ takes polynomial-time overall. This completes our proof sketch for Theorem 7.

Since every HD automata is SR, Theorem 7 together with the result of Kuperberg and Skrzypczak that HD coBüchi automata are exponentially more succinct than deterministic coBüchi automata [23, Theorem 1] implies that the same holds for MA coBüchi automata.

▶ **Corollary 18.** *There is a family $L_2, L_3, L_4, \ldots$ of languages such that for every $n \geqslant 2$, there is a MA automaton recognising L_n that has $2n+1$ states and any deterministic coBüchi automaton recognising L_n needs at least $\Omega(2^n/2n+1)$ states.*

3.4 Büchi automata

Similar to coBüchi automata, we show that for Büchi automata, no two notions amongst the notions of SD, SR, HD, MR, and MA coincide. We then later show that MR Büchi automata are exponentially more succinct than HD Büchi automata: recall that this is not the case for coBüchi automata (Theorem 7). We start by giving a SD Büchi automaton that is not SR.

▶ **Lemma 19.** *There is an SD Büchi automaton that is not SR.*

Consider the Büchi automaton $\mathcal{B}$ in Figure 7. This automaton $\mathcal{B}$ has nondeterminism on the initial state q_0, and it recognises the language $((x \cdot (a+b) \cdot y)^*(x \cdot (a+b) \cdot z))^\omega$.

It is easy to verify that $\mathcal{B}$ is SD. We describe a strategy for Adam in the SR game on $\mathcal{B}$ using which he wins almost surely, implying that $\mathcal{B}$ is not SR (Lemma 3). When Eve's token is at q_0 in the SR game, she needs to guess if the next letter after x is going to be a or b. If she guesses incorrectly, then her token moves to the left states – states l_1, l_2, and l_3, where she stays until a z is seen.

Adam's strategy in the SR game is as follows. Let Y be the regular expression $xay+xby$ and Z be the regular expression $xaz+xbz$. Adam picks a word from the set $YZY^2ZY^3ZY^4Z\ldots$ in the SR game, where from each occurrence of Y or Z, he picks one of the two words in the regular expression with half probability. We show that the probability that Eve's token takes an accepting transition on reading a word chosen randomly from Y^nZ is $\frac{1}{2^{n+1}}$. From the Borel-Cantelli lemma, we conclude that the probability that Eve's token takes infinitely many accepting transitions in the SR game is 0.

We showed, in Lemma 11, that there are MR reachability automaton that are not HD. Thus, the same holds for Büchi automata. We now show the opposite.

▶ **Lemma 20.** *There is a HD Büchi automaton that is not MR.*

Proof sketch. Consider the automaton in Figure 6, whose language is as follows. Let $\Sigma_\diamond = \{a, b, c, \diamond\}$ and $\Sigma = \{a, b, c\}$. Let L_1 and L_2 be languages of finite words over the alphabet $\Sigma_\diamond$ where $L_1 = {\Sigma_\diamond}^* c^+ \diamond$ and $L_2 = {\Sigma_\diamond}^* a \Sigma^* b^+ \diamond$. The automaton $\mathcal{A}$ accepts the language $[(L_1 + L_2)^*(L_1L_1 + L_2L_2)]^\omega$. We prove that this automaton is HD but not MR in the full version of the paper, with a proof similar to the proof of Lemma 15. ◀

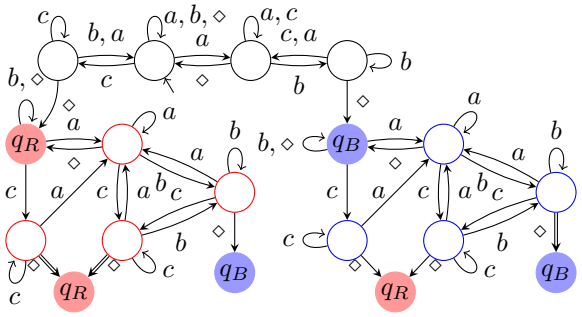


Figure 6 A HD Büchi automaton that is not MR. Accepting transitions are represented by double arrows. All red-filled (resp. blue-filled) states q_R (resp. q_B) are identified as the same state.

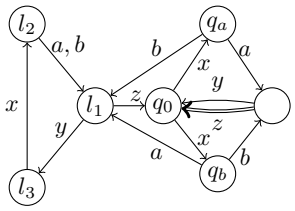


Figure 7 An SD Büchi automaton that is not SR. The accepting transitions are double-arrowed, and the initial state is q_0.

Next, we show that SR Büchi automata are exponentially more succinct than HD Büchi automata. To do this, we will use the language family from Abu Radi and Kupferman [2, Theorem 5] that shows exponential succinctness of SD Büchi over HD automata, and we show in the full version that the SD Büchi automata they construct are also MR.

▶ **Lemma 21.** *There is a family $L_2, L_3, L_4, \dots$ of languages such that for every $n \geqslant 2$, there is an MR automaton recognising L_n that has $3n + 3$ states and any HD Büchi automaton recognising L_n needs at least 2^n states.*

4 Expressivity

In Section 3, we compared the novel classes of nondeterministic automata we defined with each other and the existing notions of SD and HD, focusing on the questions of succinctness and when they coincide. In this section, we compare these notions in terms of expressivity. We show that, similar to history-determinism, stochastically resolvable $[i, j]$-parity automata are as expressive as deterministic $[i, j]$-parity automata.

▶ **Theorem 22.** *Stochastically resolvable $[i, j]$-parity automata recognise the same languages as deterministic $[i, j]$-parity automata.*

Since deterministic automata are also SR, one direction is clear. For the other direction, we show that any ω-regular language that is not recognised by any deterministic $[i, i + d]$-parity automaton cannot be recognised by any SR $[i, i + d]$-parity automaton. We reduce this to showing that the language $L_{[i+1,i+d+1]}$ of the $[i + 1, i + d + 1]$-parity condition – the set of infinite words over the alphabet $[i + 1, i + d + 1]$ in which the highest number occurring infinitely often is even – is not recognised by any SR $[i, i + d]$-parity automaton.

To prove that no SR $[i, i + d]$-parity automaton recognises $L_{[i+1,i+d+1]}$, consider any SR automaton $\mathcal{A}$ that recognises the language $L_{[i+1,i+d+1]}$ and that has an almost-sure resolver σ. We inductively construct words $u_0, u_1, \cdots, u_d$, such that from every state q, a run from q on the word u_k in automaton $\mathcal{A}$ constructed using σ contains a transition with priority at least $(i + k + 1)$ and of the same parity as $(i + k + 1)$ with positive probability. This part of the proof is nontrivial and requires careful analysis of probabilities. Once we have proved this result inductively, we obtain that $\mathcal{A}$ has at least as many priorities as in the interval $[i + 1, i + d + 1]$, and in particular, is not an $[i, i + d]$-parity automaton, as desired.

5 Complexity of recognition

We now turn our attention to the computational complexity for the problems of deciding if a given automaton is MR, SR, or MA, respectively. The exact complexity of deciding if a given automaton is HD is an open problem, with only recent results showing NP-hardness [28] and membership in PSPACE [26]. Unlike the complexity gap for deciding if a parity automaton is HD, we show that the problem for MA is NP-complete.

▶ **Theorem 23.** *The problem of deciding if a given parity automaton is MA is* NP*-complete.*

We also show undecidability for the problems of deciding if a given resolver is an almost-sure resolver for a given Büchi or coBüchi automata. For safety and weak automata, the complexities mentioned in the result below follow due to Theorem 6 and the complexities of recognising HD and SD automata.

▶ **Theorem 24.** *The problem of deciding if a given*

1. *safety automaton is SR or MR is in* P*,*
2. *reachability or weak automaton is SR or MR is* PSPACE*-complete, and*
3. *finite-memory resolver of a Büchi or coBüchi automaton is an almost-sure resolver of that automaton is undecidable.*

5.1 Memoryless adversarially resolvable automata

We first prove Theorem 23 and show NP-completeness for the problem of deciding if an automaton is MA. To prove the upper bound of NP (Lemma 25), we show that an automaton is MA if and only if there is a specific type of strategy in the so-called 2-token game on that automaton. The 2-token game is a two-player game that was introduced by Bagnol and Kuperberg [4], and it was shown by Lehtinen and Prakash that Eve wins the 2-token game on a parity automaton if and only if it is HD [26, 2-token theorem]. We use their result to show our characterisation of MA via the specific type of strategy in 2-token games in Lemma 26, and we then provide an algorithm to verify if a given strategy is indeed a winning strategy.

We then show the lower bound in Lemma 28 by showing that the problem is NP-hard, similar to the proof of NP-hardness for deciding history-determinism [28].

▶ **Lemma 25.** *Checking if a given parity automaton is MA is in* NP*.*

We describe a proof sketch for Lemma 25, where we will use the following notions.

2-token game. The 2-token game on an automaton is a game played between Adam and Eve with three tokens at the starting state of the automaton, one "owned" by Eve and two owned by Adam. The game proceeds in infinitely many rounds, where in each round, Adam selects a letter, Eve moves her token along a transition on that letter, and finally Adam moves each of his two tokens along transitions on the same letter. Therefore, in a play of this game, Eve builds a run and Adam two runs, all on the same word. A play is won by Adam if at least one of the runs on his tokens is accepting and Eve's run on her token is rejecting, and Eve wins otherwise. Given an automaton $\mathcal{A}$, we write $G2(\mathcal{A})$ to represent the 2-token game on $\mathcal{A}$, and we use $G2(\mathcal{B};\mathcal{A})$ to represent a modified 2-token game where Eve moves her token in the automaton $\mathcal{B}$, while Adam moves his two tokens in the automaton $\mathcal{A}$.

Muller condition. Muller objectives on graphs are specified by a finite set of colours C and a set of accepting subsets $\mathcal{F} \subseteq 2^C$. Each edge of the graph is labelled by a colour from C. An infinite path of this graph is accepting if the set of colours that appears infinitely often in this path is a member of $\mathcal{F}$. Muller objectives can be represented using Zielonka trees.

Zielonka tree [14] and Zielonka DAG [21]. For a Muller objective defined by colours C and accepting set $\mathcal{F}$, its Zielonka tree is a labelled rooted tree. We call vertices of this tree *nodes*. Each node ν of the Zielonka tree is labelled by a nonempty subset of C. The root is labelled by C. For a node ν_X labelled by X, its children are nodes ν_Y labelled by distinct maximal nonempty subsets Y of X such that $Y \in \mathcal{F}$ if and only if $X \notin \mathcal{F}$. If there are no such subsets Y, then ν_X has no children. A Zielonka DAG is a succinct representation of Zielonka tree, where nodes with the same labels are merged.

We show that we can characterise MA automata by the existence of a memoryless (random) strategy for Eve in the 2-token game.

▶ **Lemma 26.** *An automaton $\mathcal{A}$ is MA if and only if there is a subautomaton $\mathcal{B}$ of $\mathcal{A}$ such that the strategy of Eve where she picks available transitions uniformly at random is an almost-sure winning strategy in the game $G_2(\mathcal{B}; \mathcal{A})$.*

To check if an automaton is MA in NP, we guess a subautomaton $\mathcal{B}$ of $\mathcal{A}$, construct the 2-token game $G2(\mathcal{B}; \mathcal{A})$, and verify if Eve playing randomly is an almost-sure winning strategy. Subsequently, we construct a game where Eve's vertices in $G2(\mathcal{B}; \mathcal{A})$ are substituted with stochastic vertices, resulting in a Markov Decision Process (MDP) – a single-player complete-observation stochastic game where Adam is the only player.

If $\mathcal{A}$ is MA, Adam satisfies his objective with probability 0 in this MDP resulting out of the 2-token game. The winning condition of the 2-token game (and thus this MDP) for Adam can be represented by a Muller objective [29, Page 70], whose Zielonka DAG has size that is at most polynomial in the number of priorities of $\mathcal{A}$ [29, Page 72].

In Theorem 27, we show that it can be verified in polynomial time if Adam in an MDP has a strategy to satisfy a Muller objective with positive probability, where the objective is input as a Zielonka DAG. Therefore, we can verify if Eve wins $G2(\mathcal{B}; \mathcal{A})$ almost-surely in polynomial time. This concludes our proof of Lemma 25.

▶ **Theorem 27.** *There is an algorithm to decide if Adam positively (resp. almost-surely) wins MDPs $\mathcal{M}$ with Muller objectives represented by a Zielonka DAG $\mathcal{Z}$ in time $\mathcal{O}(|\mathcal{M}||\mathcal{Z}|)$, where $|\mathcal{M}|$ is the number of edges in $\mathcal{M}$ and $|\mathcal{Z}|$ is the number of edges in $\mathcal{Z}$.*

For MDPs with Muller objectives that are union closed, Chatterjee in 2007 gave a polynomial-time algorithm to decide if Adam has a strategy to win almost surely (or positively) [13, Section 4]. Since such conditions can be represented by a Zielonka DAG where all nodes other than the root are leaf nodes, Theorem 27 is a significant generalisation of their result.

We show the lower bound for the problem of recognising MA automata using the result of NP-hardness of recognising HD automata by Prakash [28], and show that the same reduction can be modified to show the NP-hardness of recognising MA automata.

▶ **Lemma 28.** *Checking if a given parity automaton is MA is* NP*-hard.*

5.2 Memoryless-stochastically resolvable automata and stochastically resolvable automata

We now discuss decision problems related to MR or SR automata. Our results for safety, reachability, and weak automata follows, due to Theorem 6, from known results for HD automata [9, Theorem 19] and semantically deterministic automata [3, Theorem 3], thus yielding the following proposition.

▶ **Proposition 29.** *Deciding if a safety automaton is SR is in* P*, and deciding if a reachability or weak automaton is SR is* PSPACE*-complete.*

Resolver-(co)Büchi: Given a finite memory resolver $\mathcal{R}$ for a (co)Büchi automaton $\mathcal{A}$, is $\mathcal{R}$ an almost-sure resolver for $\mathcal{A}$?

▶ **Lemma 30.** *The problems Resolver-coBuchi and Resolver-Buchi are undecidable.*

We prove undecidability of the problem of Resolver-coBuchi by reducing from the problem of *checking the emptiness of probabilistic Büchi automaton under the positive acceptance semantics* (word is accepted if there is an accepting run of non-zero probability) [5, Theorem 2]. We prove undecidability of Resolver-Buchi by reducing from the *zero-isolation problem* for finite probabilistic automata (if there are finite words for which probability of acceptance is positive and reaches arbitrarily close to 0), which is also undecidable [18, Theorem 4].

6 Discussion

We conclude with a brief discussion on a few related notions and pose some open problems.

In an effort to also reason about HD and SD automata, Abu Radi, Kupferman, and Leshkowitz [3] introduced and studied a class of automata they called almost-determinisable by pruning (almost-DBP). While their definition is related to the probability that a run on a randomly generated word in the language is accepting, we study the notion where a randomly generated run by a resolver on every word in the language is accepting. To further highlight the difference, almost-DBP Büchi auomtata are the same as SD Büchi automata, whereas SR Büchi automata are a different class than SD Büchi auomtata.

Good-for-MDP automata are automata that admit compositionality with MDPs that make them relevant for reinforcement learning and MDP model checking [19]. We remark that SR automata are good-for-MDP, since an almost-sure resolver for an automaton can be used as a strategy for syntactic satisfaction objective for the product of automaton with any MDP. However, the converse is not true since good-for-MDP Büchi automata recognise all ω-regular languages [19, Section 3.2] but SR Büchi automata are only as expressive as deterministic Büchi automata (Theorem 22).

We now discuss some open questions. For the class of MA automata, the exact probability distributions of the resolver does not matter. However, the same remains unknown for SR and MR automata.

▶ **Question.** *Are almost-sure resolvers in stochastically resolvable settings indifferent to the exact probability distribution?*

Despite the undecidability of resolver checking for Büchi or coBüchi automata, the complexity status of the class membership problem for SR and MR automata is open. The undecidability of Resolver-coBuchi (Lemma 30) is somewhat surprising since we also showed (Theorem 7) that every SR coBüchi automaton can be efficiently converted to an MA coBüchi automaton, where membership is in NP, providing hope for decidability.

▶ **Question.** *Is it decidable to check if a given (co)Büchi automaton is SR?*

Theorem 7 also implies that HD coBüchi automata have language-equivalent MA coBüchi automata with at most as many states, thus trading exponential memory required for HD coBüchi automata for only randomness. We ask whether the same holds for parity automata.

▶ **Question.** *Does every HD parity automaton have a language-equivalent MA parity automaton of the same size?*

References

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

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

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

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

5 Christel Baier, Nathalie Bertrand, and Marcus Größer. On Decision Problems for Probabilistic Büchi Automata. In *FoSSaCS*, volume 4962 of *Lecture Notes in Computer Science*, pages 287–301. Springer, 2008. doi:10.1007/978-3-540-78499-9_21.

6 Christel Baier, Marcus Grösser, and Nathalie Bertrand. Probabilistic ω-automata. *J. ACM*, 59(1), March 2012. doi:10.1145/2108242.2108243.

7 Mikołaj Bojanczyk and Wojciech Czerwinski. An automata toolbox. Technical report, University of Warsaw, 2018.

8 Udi Boker, Orna Kupferman, and Michal Skrzypczak. How Deterministic are Good-For-Games Automata? In *FSTTCS*, volume 93 of *LIPIcs*, pages 18:1–18:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.FSTTCS.2017.18.

9 Udi Boker and Karoliina Lehtinen. History Determinism vs. Good for Gameness in Quantitative Automata. In *Proc. of FSTTCS*, pages 35:1–35:20, 2021. doi:10.4230/LIPICS.FSTTCS.2021.38.

10 Udi Boker and Karoliina Lehtinen. Token Games and History-Deterministic Quantitative-Automata. *Logical Methods in Computer Science*, 19(4), 2023. doi:10.46298/LMCS-19(4:8)2023.

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

12 J Richard Büchi and Lawrence H Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

13 Krishnendu Chatterjee. Stochastic müller games are PSPACE-complete. In *FSTTCS*, volume 4855 of *Lecture Notes in Computer Science*, pages 436–448. Springer, 2007. doi:10.1007/978-3-540-77050-3_36.

14 Stefan Dziembowski, Marcin Jurdzinski, and Igor Walukiewicz. How Much Memory is Needed to Win Infinite Games? In *LICS*, pages 99–110. IEEE Computer Society, 1997. doi:10.1109/LICS.1997.614939.

15 Rüdiger Ehlers and Ayrat Khalimov. Fully generalized reactivity(1) synthesis. In *TACAS*, pages 83–102. Springer Nature Switzerland, 2024. doi:10.1007/978-3-031-57246-3_6.

16 Rüdiger Ehlers and Sven Schewe. Natural Colors of Infinite Words. In *FSTTCS*, volume 250 of *LIPIcs*, pages 36:1–36:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.FSTTCS.2022.36.

17 Javier Esparza, Jan Křetínský, and Salomon Sickert. From LTL to deterministic automata. *Formal Methods in System Design*, 49(3):219–271, 2016. doi:10.1007/s10703-016-0259-2.

18 Hugo Gimbert and Youssouf Oualhadj. Probabilistic automata on finite words: Decidable and undecidable problems. In *ICALP (2)*, volume 6199 of *Lecture Notes in Computer Science*, pages 527–538. Springer, 2010. doi:10.1007/978-3-642-14162-1_44.

19 Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Good-for-MDPs Automata for Probabilistic Analysis and Reinforcement Learning. In *TACAS (1)*, volume 12078 of *Lecture Notes in Computer Science*, pages 306–323. Springer, 2020. doi:10.1007/978-3-030-45190-5_17.

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

21 Paul Hunter and Anuj Dawar. Complexity Bounds for Regular Games. In *MFCS*, volume 3618 of *Lecture Notes in Computer Science*, pages 495–506. Springer, 2005. doi:10.1007/11549345_43.

22 Marek Jankola and Jan Strejcek. Tighter construction of tight büchi automata. In *FoSSaCS*, volume 14574 of *Lecture Notes in Computer Science*, pages 234–255. Springer, 2024. doi:10.1007/978-3-031-57228-9_12.

23 Denis Kuperberg and Michal Skrzypczak. On Determinisation of Good-for-Games Automata. In *ICALP (2)*, volume 9135 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 2015. doi:10.1007/978-3-662-47666-6_24.

24 Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. Safraless compositional synthesis. In *CAV*, volume 4144 of *Lecture Notes in Computer Science*, pages 31–44. Springer, 2006. doi:10.1007/11817963_6.

25 Orna Kupferman and Moshe Y. Vardi. Safraless decision procedures. In *FOCS*, pages 531–542. IEEE Computer Society, 2005. doi:10.1109/SFCS.2005.66.

26 Karoliina Lehtinen and Aditya Prakash. The 2-Token Theorem: Recognising History-Deterministic Parity Automata Efficiently. In *STOC*, 2025. doi:10.1145/3717823.3718310.

27 Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, 1966. doi:10.1016/S0019-9958(66)80013-X.

28 Aditya Prakash. Checking History-Determinism is NP-hard for Parity Automata. In *FoSSaCS (1)*, volume 14574 of *Lecture Notes in Computer Science*, pages 212–233. Springer, 2024. doi:10.1007/978-3-031-57228-9_11.

29 Aditya Prakash. *History-Deterministic Parity Automata: Games, Complexity, and the 2-Token Theorem*. PhD thesis, University of Warwick, 2025. doi:10.48550/arXiv.2501.12302.

30 Shmuel Safra. On the complexity of omega-automata. In *FOCS*, pages 319–327. IEEE Computer Society, 1988. doi:10.1109/SFCS.1988.21948.