Word Automata with Limited Nondeterminism

Yong Li
Key Laboratory of System Software (Chinese Academy of Sciences),
Institute of Software, Chinese Academy of Sciences, Beijing, China

Soumyajit Paul
University of Liverpool, UK

Sven Schewe
University of Liverpool, UK

Qiyi Tang
University of Liverpool, UK

Abstract

We survey word automata with limited nondeterminism, a family of models lying between deterministic and fully nondeterministic automata. While determinism provides a simple algorithmic basis for verification, reactive synthesis, and probabilistic analysis, determinisation incurs large state blow-up, especially for ω-regular specifications. Limited nondeterminism offers a middle ground: it preserves some of the succinctness of nondeterministic automata while retaining enough structure for algorithmic use. We focus on three notions: unambiguous automata, in which each accepted word has at most one accepting run; good-for-games automata, whose nondeterministic choices can be resolved on the fly from the input prefix; and good-for-MDPs automata, which preserve optimal satisfaction probabilities when composed with MDPs. We compare these models in terms of expressiveness, succinctness, decision problems, minimisation, and applications to model checking, synthesis, reinforcement learning, and stochastic planning. Finally, we discuss how these threads converge: recent work has used good-for-games minimisation as a preprocessing step to reduce unambiguous and good-for-MDPs automata before composition, yielding more compact constructions for probabilistic analysis and planning. We present this as a recurring algorithmic pattern – resolving an automaton's nondeterminism before it is amplified by the product with the system – that unifies otherwise separate lines of work.

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

Keywords and phrases History-determinism, finite automata, probabilistic automata

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2026.3

Category Invited Talk

Funding This work has been supported in part by the EPSRC through the projects EP/X042596/1 and EP/X03688X/1, the CAS Project for Young Scientists in Basic Research (Grant No. YSBR-040), the National Natural Science Foundation of China (Grant No. 62102407), and the Beijing Natural Science Foundation (Grant No. IS26039).

Acknowledgements We would like to thank the anonymous reviewers for their constructive feedback, that helped improve the paper.

1 Introduction

Word automata provide a central bridge between logical specifications and algorithmic analysis. In verification [79], reactive synthesis [63], and probabilistic reasoning [4], a specification is often translated into an automaton, either to be analysed directly or to be composed with a system model. This automata-theoretic perspective is powerful because it separates the specification of desired behaviour from the algorithms used to analyse it. Traditionally, this

perspective has relied on deterministic automata: once the next input letter is fixed, the automaton state evolves uniquely, making products with games, Markov chains, and Markov decision processes conceptually straightforward.

The difficulty is that determinism is often expensive. For ω-regular specifications, determinisation can cause a large state blow-up, which is then inherited by the constructions used in model checking, synthesis, and planning. In contrast, nondeterministic automata can be substantially more succinct [54, 55, 56, 48]. However, unrestricted nondeterminism is usually not suitable for direct use in algorithmic products: choices may depend on future input, may interact badly with strategies, or may destroy quantitative correctness. This has motivated the study of automata models with restricted forms of nondeterminism that retain part of the succinctness of nondeterminism while imposing enough structure for algorithmic use.

This survey focuses on three notions of limited nondeterminism, each tailored to a different algorithmic setting. Unambiguous automata [75]allow nondeterministic branching but require every accepted word to have at most one accepting run. This property has proven to be useful in probabilistic model checking of Markov chains, where the uniqueness of runs enables quantitative analysis without full determinisation [26, 7, 5]. Good-for-games (GFG) automata [44] require nondeterminism to be resolved on the fly, using only the input prefix seen so far, making them suitable for games and synthesis, where inputs are chosen adversarially. Good-for-MDPs (GFM) automata [38] adapt this idea to probabilistic settings: their nondeterminism must be compatible with product constructions over Markov decision processes (MDPs), so that optimal satisfaction probabilities are preserved, a key requirement in MDP analysis in general and reinforcement learning in particular [37].

Our goal is to present these notions as a common landscape between deterministic and fully nondeterministic automata. They retain some of the succinctness of nondeterministic automata, while imposing enough structure to support model checking, synthesis, minimisation, and stochastic planning. From this perspective, limited nondeterminism is not just relaxed determinism, but a way to design automata that fit the algorithmic task at hand.

The survey is organised as follows. We first recall the common automata-theoretic and probabilistic background needed to compare the different models in Section 2. We then discuss unambiguous automata as a restriction based on the number of accepting runs, emphasising their succinctness and their role in Markov-chain model checking [54, 55, 56, 5] in Section 3. Next, we turn to GFG automata, where the central issue is the history-dependent resolution of nondeterministic choices, and review their expressiveness, decision procedures, minimisation results, and applications to synthesis [48, 2, 52, 70] in Section 4. We then consider GFM automata, which provide a semantic criterion for sound product-based MDP analysis and reinforcement learning by preserving optimal satisfaction probabilities under product constructions [38, 73, 74] in Section 5. The final part of the survey brings these threads together in Section 6. Recent work shows that limited nondeterminism can also be exploited inside broader verification and planning pipelines. In particular, GFG minimisation techniques [70] can be used as a preprocessing step to reduce unambiguous and GFM automata before product construction, leading to more compact constructions for probabilistic analysis and stochastic planning [57, 81]. This suggests a recurring algorithmic pattern: handle the automaton's nondeterminism before it is amplified by composition with the system.

In this way, the survey offers a unified perspective on these developments. The central question is not simply whether an automaton is deterministic, nondeterministic, unambiguous, GFG, or GFM, but what kind of nondeterminism can be tolerated by which algorithmic setting. This clarifies why different restrictions have emerged, how they relate to each another, and why they are useful in probabilistic model checking, synthesis, and reinforcement learning.

2 Preliminaries

We fix a finite *alphabet* Σ. A *word* is a finite or infinite sequence of letters of Σ; ε is the empty word, and Σ^*, Σ^ω are the sets of finite and infinite (ω-)words. We write $w[i]$ for the i-th letter of w and $u \cdot w$ (or uw) for concatenation.

Transition systems and automata. A *transition system* (TS) is a tuple $\mathcal{T} = (Q, I, \delta)$ with Q a finite set of states, $I \subseteq Q$ the initial states, and $\delta : Q \times \Sigma \to 2^Q$ a transition function, extended to sets and words in the standard way. $\mathcal{T}$ is *deterministic* if I is a singleton set and $|\delta(q, a)| \leq 1$ for all q, a, and *nondeterministic* otherwise. An *automaton* is a TS together with an acceptance criterion.

A *nondeterministic finite automaton* (NFA) is a pair $(\mathcal{T}, F)$ with $F \subseteq Q$ a set of *final* states. A *run* of an NFA on a finite word u is a sequence of states $q_0 q_1 \cdots q_n$ with $q_0 \in I$ and $q_{i+1} \in \delta(q_i, u[i])$; it is *accepting* if $q_n \in F$, and u is *accepted* if it has an accepting run.

For ω-words we use *transition-based* acceptance throughout, which is standard in recent work on GFGness and essential for the canonicity and minimisation results discussed later. Write $\Delta(\delta) = \{(q, a, q') \mid q' \in \delta(q, a)\}$ for the set of transitions. A *run* of a TS on $w \in \Sigma^\omega$ is an infinite sequence of transitions $\rho = (q_0, w[0], q_1)(q_1, w[1], q_2) \cdots$ with $q_0 \in I$, and $\inf(\rho)$ is the set of transitions occurring infinitely often in ρ. A *nondeterministic parity automaton* (NPA) is a pair $(\mathcal{T}, \mathsf{p})$ with $\mathsf{p} : \Delta(\delta) \to \{0, \ldots, k\}$ assigning a *priority* to each transition; ρ is *accepting* if $\max \mathsf{p}(\inf(\rho))$ is even. A *Büchi* automaton (NBA) is the case when $k = 1$, where ρ is accepting iff it takes a transition from the *accepting* set $\alpha = \mathsf{p}^{-1}(1)$ infinitely often; dually, a *co-Büchi* automaton (NCA) accepts iff it takes a transition from the *rejecting* set α only finitely often. An ω-word is *accepted* if it has an accepting run, and $\mathcal{L}(\mathcal{A})$ denotes the *language* of $\mathcal{A}$. NPAs – already NBAs – recognise exactly the ω-regular languages. A few cited results are specific to *state-based* acceptance, where priorities are placed on states rather than transitions; we flag these explicitly, as the distinction affects minimisation complexity (cf. Section 4).

Markov decision processes. A *Markov decision process* (MDP) is a tuple $\mathcal{M} = (S, \mathrm{Act}, \Sigma, \mathsf{P}, s_0, L)$ [4] with S a finite set of states, Act a set of actions, P a transition probability function, s_0 an initial state, and $L : S \times \mathrm{Act} \to \Sigma$ a labelling of state-action pairs; $\mathrm{Act}(s)$ denotes the actions available at s. $\mathcal{M}$ is a (labelled) *Markov chain* (MC) if $|\mathrm{Act}(s)| = 1$ for all s. A *path* alternates states and actions, $\xi = s_0 a_0 s_1 a_1 \cdots$ with $a_i \in \mathrm{Act}(s_i)$ and $\mathsf{P}(s_i, a_i, s_{i+1}) > 0$, and its *trace* is the induced label sequence $L(s_0, a_0)L(s_1, a_1) \cdots \in \Sigma^\omega$. We write FPaths for the set of finite paths.

A *strategy* is a function $\mu : \mathrm{FPaths} \to \mathsf{Distr}(\mathrm{Act})$ with $\mu(\xi) \in \mathsf{Distr}(\mathrm{Act}(\mathsf{lst}(\xi)))$, where $\mathsf{lst}(\xi)$ is the last state of ξ; it is *memoryless* if it depends only on $\mathsf{lst}(\xi)$ and *finite-memory* if realisable by a finite automaton reading the path. A strategy resolves the nondeterminism of $\mathcal{M}$ and induces a Markov chain $\mathcal{M}^\mu$ (finite-state when μ is finite-memory), which, together with an initial state and by the standard σ-algebra generated by cylinder-sets induces a probability measure $\mathbb{P}_{\mathcal{M}^\mu}$ on infinite paths; for ω-regular language $\mathcal{L}$ we write $\mathbb{P}_{\mathcal{M}^\mu}(\mathcal{L})$ for the measure of the paths whose trace lies in $\mathcal{L}$.

A *strongly connected component* (SCC) is a maximal set of states $C \subseteq S$ such that every pair of states in C can reach each other in the underlying graph. A sub-structure of $\mathcal{M}$ that is closed under its probabilistic transitions and strongly connected is an *end-component*; an inclusion-maximal one is a *maximal end-component* (MEC). Almost surely, a path of $\mathcal{M}$ eventually stays within some MEC, which makes MECs central to the analysis of ω-regular objectives.

3 Unambiguous Automata

We begin with the oldest and syntactically simplest restriction: limiting the number of accepting runs. This restriction is independent of how nondeterministic choices are made during a run, and is particularly useful in probabilistic model checking of Markov chains.

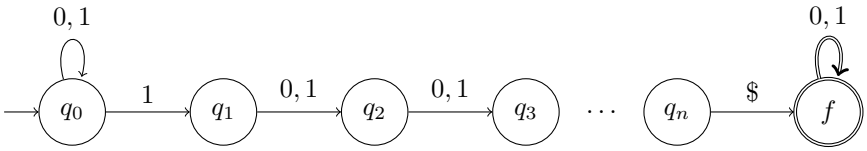

Figure 1 An NFA or NBA recognising the language of words that contain exactly one \$, and in which a 1 occurs exactly n positions before the occurrence of \$. Double circles denote final states in the NFAs, while double-lined transitions denote accepting transitions in the NBAs.

An automaton is unambiguous if it can make nondeterministic choices, but it is guaranteed that there is at most one accepting run for every input word. Unambiguous automata are as expressive as general nondeterministic automata, while being exponentially more succinct than deterministic automata [54, 55, 56].

▶ **Theorem 1** ([54, 55, 56]). *Unambiguous automata can be exponentially more succinct than deterministic automata. Nondeterministic automata can be exponentially more succinct than unambiguous ones.*

To illustrate this succinctness gap, fix $n \in \mathbb{N}$ and consider the language over the alphabet $\{0, 1, \$\}$ with words in which exactly one \$ occurs, and where a 1 occurs exactly n steps before this \$. This language is recognised by an unambiguous automaton with $n + 2$ states (see Figure 1). In contrast, any deterministic automaton recognising this language requires at least 2^n states, as it must encode the positions of the 1's among the last n input letters preceding the \$.

The exponential succinctness of unambiguous Büchi automata (UBAs) relative to deterministic automata also manifests itself in translations from linear temporal logic (LTL) to automata. In particular, the automata obtained from LTL formulas via the classical closure constructions [82, 79] are, in fact, unambiguous generalised Büchi automata (UGBAs), in which the acceptance condition consists of a finite collection of accepting sets, and an accepting run is required to visit every accepting set infinitely often. Moreover, these automata are separated, that is, different states recognise disjoint languages. (They can be degeneralised easily, but this costs the separation property.) While translating LTL formulas into deterministic ω-automata incurs a double-exponential blow-up in the worst case, translation into separating UGBAs requires only a single-exponential blow-up. This observation has been made in several works, see, e.g., [26, 60].

An important question is recognising unambiguous automata. The following theorem and its proof are presented in [24, Remark 1], with a more detailed treatment for tree automata given in [68, Lemma 5]. The proof is based on constructing a suitably modified product automaton, which captures pairs of distinct runs. The original automaton is unambiguous if and only if this product automaton has an empty language.

▶ **Theorem 2** ([24]). *Deciding whether a nondeterministic automaton is unambiguous is in* ***NL****.*

For unambiguous finite automata (UFAs), the inclusion problem – asking whether the language of one automaton is a subset of that of another – is decidable in polynomial time [77], whereas it is **PSPACE**-complete for general nondeterministic automata. Consequently, the equivalence problem for UFAs – whether two automata recognise the same language – and the universality problem – whether a given automaton accepts all finite words – are also decidable in polynomial time.

The minimisation problem for UFAs is **NP**-complete: membership in NP follows straightforwardly from polynomial-time equivalence checking, while **NP**-hardness is established in [46, Corollary 3.1].

For unambiguous ω-automata, the precise complexity of the inclusion, equivalence, and universality problems for UBAs remains a long-standing open problem.

3.1 Weaker and Stronger Notions

Degree of Ambiguity. Unambiguous automata can be generalised by relaxing the restriction to a single accepting run per word; the maximal cardinality of accepting runs is referred to as the *degree of ambiguity*. Formally, the degree of ambiguity of a word w in $\mathcal{A}$ is defined as the number of accepting runs of $\mathcal{A}$ on w. The degree of ambiguity of $\mathcal{A}$ is the maximum degree of ambiguity over all input words, if such a maximum exists, and is infinite otherwise. In the former case, $\mathcal{A}$ is called *finitely ambiguous*, while in the latter case it is called *infinitely ambiguous*. An unambiguous automaton has ambiguity degree of one. Infinitely ambiguous NFAs can be further classified as *polynomially ambiguous* or *exponentially ambiguous*, depending on the growth rate of the ambiguity as a function of the word-length. Weber and Seidl [80] investigated both finitely and infinitely ambiguous automata including *polynomially ambiguous* and *exponentially ambiguous* automata. They obtained polynomial-time algorithms for deciding the membership of each classes, which rely on structural characterisations of the respective classes. Language inclusion is in **P** for k-ambiguous NFAs where k is fixed [77]. If k is given as part of the input, the problem becomes **PSPACE**-complete. The lower bound follows from the **PSPACE**-completeness of the emptiness problem for the intersection of k DFAs: equivalently, universality of the union of k DFAs is **PSPACE**-complete, and connecting these DFAs through a fresh initial state yields a k-ambiguous NFA. A recent survey [40] provides a comprehensive overview of results on the degree of ambiguity and related measures of nondeterminism for NFAs.

For ω-automata, [59] initiated the study of finitely ambiguous NBAs. They investigate the possible degrees of ambiguity for NBAs and provide a classification of ambiguity degrees for Büchi automata, together with the complexity of the corresponding decision problems. Since a single infinite word may admit infinitely many accepting runs – even uncountably many – automata over ω-words exhibit additional forms of infinite ambiguity. In particular, one distinguishes between *finitely ambiguous*, *countably ambiguous*, and *uncountably ambiguous* automata. Similar to the finite word case, structural characterisations and polynomial algorithms for membership of the classes are provided in [59, 66]. These characterisations and algorithms can be adopted for other acceptance conditions, e.g., parity, Rabin, Muller, etc., [68, 67].

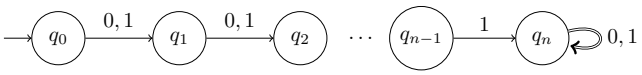

Figure 2 A DBA recognising the language L_n, the language of words whose n-th letter is one.

Separating Automata. One can further restrict unambiguity by requiring that the sets of accepting runs from any two states are disjoint, i.e., an ω-word can only have one final path in the automaton. Such automata are called *strongly* unambiguous or *separating*. Just like unambiguous automata, they are as expressive as general nondeterministic automata [18] and can be exponentially more succinct than deterministic automata, as illustrated by the example in Figure 1. In contrast, DBAs can be exponentially smaller than separating automata – unlike in the unambiguous case, where every DBA is already unambiguous. This is witnessed by the example in Figure 2 from [17, Remark 3]. Note that the same family of languages also witnesses that separating automata require more states than unambiguous automata, as DBAs are unambiguous. For a fixed $n \in \mathbb{N}$, let L_n be the set of words over $\{0,1\}$ whose n^{th} letter is 1. While L_n is recognised by a DBA with $n+1$ states, any separating Büchi automaton requires at least 2^{n-1} states. To see this, let $\mathcal{A}_n$ be a separating automaton for L_n with fewer than 2^{n-1} states. Then there exist distinct $w_1, w_2 \in \{0,1\}^{n-1}$ such that $w_1 1 0^\omega$ and $w_2 1 0^\omega$ are accepted from the same state q. Assume w.l.o.g. that $w_1 = w 1 w_1'$ and $w_2 = w 0 w_2'$. Let v be a word of length $n - |w| - 1$. Then $vw1w_1'10^\omega \in L_n$, whereas $vw0w_2'10^\omega \notin L_n$. Furthermore, any accepting run of the word $vw1w_1'10^\omega$, must be in q after reading v from the start state. Otherwise, if v goes to q' for an accepting run of $vw1w_1'10^\omega$, then $w1w_1'10^\omega$ is accepted from q', contradicting the separation property. But this yields an accepting run for $vw0w_2'10^\omega$ – a contradiction.

Unlike UBAs, the language inclusion, equivalence, and universality problems for separating Büchi automata are decidable in polynomial time [17]. This makes them, to the best of our knowledge, the only classical nondeterministic automaton model recognising all ω-regular languages for which language inclusion is known to be tractable.

The relevance of unambiguity becomes most apparent in quantitative analysis. Although a UBA may admit many runs on a given word, at most one of them can be accepting. This property is crucial in probabilistic model checking. For a general nondeterministic Büchi automaton, different accepting runs of the same word correspond to different accepting paths in the product with a Markov chain, making it difficult to relate acceptance probabilities to probabilities of paths in the product. In contrast, for a UBA, every accepted word contributes through a unique accepting run. Hence accepting runs correspond to disjoint probabilistic events, preventing probability mass from being counted multiple times and enabling direct quantitative analysis without determinisation.

3.2 Application to Probabilistic Model Checking

UBAs have found important applications in probabilistic model checking of finite-state Markov chains. A polynomial-time procedure for UBAs representing safety properties was proposed in [7], while [26] provides a polynomial-time algorithm for separating Büchi automata. Furthermore, [5] presents the first correct polynomial-time (indeed, **NC**, which is a subclass of **P** comprising the problems solvable in poly-logarithmic parallel time) algorithm for this problem. All these approaches are based on solving systems of linear equations.

Given a labelled Markov chain $\mathcal{M}$ and a UBA $\mathcal{U}$, the Markov chain model checking problem is to compute the probability that a trajectory generated by $\mathcal{M}$ is accepted by $\mathcal{U}$. Let $\mathcal{M}$ have state space S with initial state s_0, and $\mathcal{U}$ have state space Q with initial state q_0. Define $\vec{\chi} \in \mathbb{R}^{Q \times S}$ such that $\vec{\chi}(\langle q, s \rangle)$ is the probability that a trajectory of $\mathcal{M}$ starting in s is accepted in $\mathcal{U}$ starting from q. It suffices to compute $\vec{\chi}(\langle q_0, s_0 \rangle)$. We construct the product of $\mathcal{U}$ and $\mathcal{M}$, yielding a nonnegative matrix $B \in \mathbb{R}^{(Q \times S) \times (Q \times S)}$. The vector $\vec{\chi}$ satisfies the linear system $B\vec{\mathbf{x}} = \vec{\mathbf{x}}$, which does not uniquely determine the solution, and thus requires additional constraints. If $\mathcal{U}$ is separating, it suffices to add $\vec{\mathbf{x}}(\langle q, s \rangle) = 0$ for all pairs in rejecting bottom SCCs, and $\sum_{\langle q,s \rangle \in C} \vec{\mathbf{x}}(\langle q, s \rangle) = 1$ for each accepting bottom SCC C [17].

For general UBAs, the approach of [5] relies on linear-algebraic properties of the product matrix B. Intuitively, one seeks components of the product in which probability mass can persist indefinitely. These correspond to SCCs whose associated submatrices have spectral radius 1[1], meaning that probability can remain in the component forever rather than eventually leaking out. Such SCCs are called *recurrent*. These are classified as accepting or rejecting based on the automaton transitions they contain. For rejecting recurrent SCCs, it requires to impose $\vec{\mathbf{x}}(c) = 0$, while for each accepting recurrent SCC C, it requires $\vec{\mu}_C^{\intercal}\vec{\mathbf{x}}_C = 1$, where $\vec{\mu}_C$ is called a cut.

Overall, the system of linear equations is:

$$\begin{aligned} \vec{\mathbf{x}} &= B\vec{\mathbf{x}}, && \\ \vec{\mu}_C^{\intercal}\vec{\mathbf{x}}_C &= 1 && \text{for each accepting recurrent SCC } C, \\ \vec{\mathbf{x}}(c) &= 0 && \text{for all } c \text{ that cannot reach an accepting recurrent SCC.} \end{aligned} \tag{1}$$

Unambiguity restricts the number of accepting runs, but it does not explain how to choose transitions on the fly. For games, this is the central issue: nondeterministic choices must be resolved while the input is being produced. This leads to good-for-games automata.

4 Good-for-Games Automata

The notion of good-for-games (GFG) automata was introduced by Henzinger and Piterman [44], who observed that determinism is not necessary for solving games with ω-regular winning conditions. In the classical approach to reactive (Church) synthesis, one composes the game with a deterministic automaton for the specification to obtain a game with a standard acceptance condition. Henzinger and Piterman showed that some nondeterministic automata, which they called *good-for-games*, have enough compositional structure for this construction to work without determinisation.

Related ideas appeared independently in other settings. Kupferman, Safra, and Vardi studied word automata in tree-based settings, leading to the notion later called *good-for-trees* automata [50, 8]. Colcombet [23] introduced *history-determinism* for quantitative automata with cost functions, capturing nondeterministic automata whose choices can be resolved on the fly while preserving strong compositional and algorithmic properties. These notions were later unified: for ω-regular languages, good-for-games, good-for-trees, and history-deterministic automata coincide [12]. Since then, GFG automata have attracted significant attention because they form a natural intermediate model between deterministic and unrestricted nondeterministic automata.

Intuitively, a nondeterministic automaton is GFG if there exists a strategy to resolve its nondeterministic choices based only on the input prefix, without knowledge of the future. Formally, good-for-gameness (GFGness) of an automaton can be characterised by the outcome of the *letter game* introduced by [44] (also called the HD game in, for example, [52]). A play starts with a token in the initial state and proceeds in infinitely many rounds. In each round, Adam selects a letter from the input alphabet, and Eve responds by moving the token along a corresponding transition of the automaton. Adam constructs a word, while Eve constructs a run for this word. Eve wins the play if either the word is not in the language of the automaton, or the constructed run is accepting. In other words, there must exist a uniform strategy that resolves nondeterminism on the fly, constructing accepting runs for all words in the language without knowledge of future inputs.

[1] Spectral radius of a matrix is the maximum of the absolute values of its eigenvalues

The expressive power of GFG automata was first studied in the context of derivable tree languages [50, 61]. GFG automata are as expressive as deterministic ones. For finite words, this follows directly from the fact that GFG automata are determinisable by pruning (DBP). For ω-words, however, the situation is more subtle: GFG automata need not be DBP as first observed in [8]. Figure 3, taken from [49], shows a GFG Büchi automaton that is not DBP. It recognises words containing infinitely many occurrences of either $a0a0$ or $a1a1$. Its only nondeterministic choice is at q_0 on input a, and GFGness is witnessed by the strategy that moves to q_1 whenever q_0 is visited initially or reached from q_2 or q_5, and to q_2 otherwise. Intuitively, this strategy guesses that an occurrence of $a0$ will be followed by another $a0$. Since there are infinitely many occurrences of $a0a0$, this guess succeeds infinitely often, ensuring that the transition from q_1 to q_3 is taken infinitely often in the constructed run whenever $a0a0$ appears infinitely often in the word. Symmetrically, the same argument applies when $a1a1$ appears infinitely often.

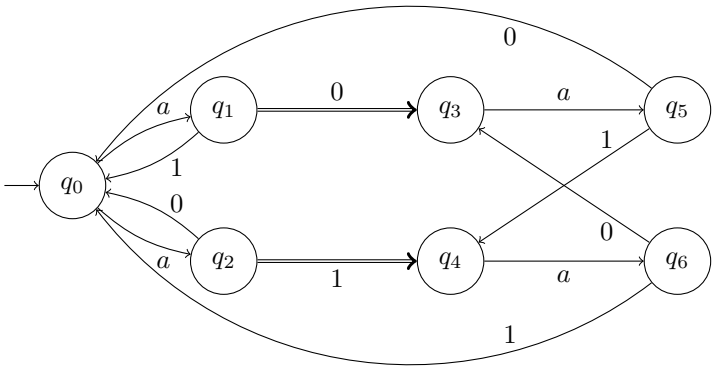

Figure 3 A GFG Büchi automaton that is not DBP from [49].

Later, Kuperberg and Skrzypczak [48] analysed the succinctness of GFG automata relative to deterministic automata. They showed that the cost of determinisation depends strongly on the acceptance condition: GFG Büchi automata can be determinised with only a quadratic blow-up, whereas GFG coBüchi automata may require an exponential blow-up [48]. In particular, this established that GFG Büchi automata can be quadratically more succinct than DBAs for the same language.

▶ **Theorem 3** ([48]). *GFG Büchi automata are up to quadratically more succinct than DBA.*

▶ **Theorem 4** ([48]). *GFG coBüchi automata can be exponentially more succinct than DCAs.*

Whether GFG Büchi automata can be strictly smaller than every language-equivalent DBA at all – the most basic succinctness question left open – was settled only recently [22].

▶ **Theorem 5** ([22]). *There is a GFG Büchi automaton that has strictly fewer states than every language-equivalent DBA.*

For GFG automata to be useful as a specification formalism, one must be able to recognise them and, ideally, reduce them. This leads to two central algorithmic questions: deciding GFGness and minimising GFG automata.

A straightforward approach to deciding GFGness for parity automata solves the letter game directly and incurs exponential time, as it requires determinisation [44]. In 2015, Kuperberg and Skrzypczak [48] gave a polynomial-time algorithm for coBüchi automata. Their approach first constructs a language-equivalent GFG automaton, provided Eve wins the *Joker game* (a necessary condition for GFGness), and then reduces the problem to checking whether the original automaton simulates the constructed one. This was followed by Bagnol and Kuperberg [2], who provided a polynomial-time algorithm for Büchi automata based on the *k-token game*: Adam builds a word letter by letter, Eve constructs a run on a designated token, and Adam simultaneously constructs runs on k additional tokens. Eve wins if whenever any of Adam's runs is accepting, so is her run. They showed that a Büchi automaton is GFG if and only if Eve wins the 2-token game, yielding a polynomial-time decision procedure. They also conjectured that this characterisation extends to all parity automata. Boker, Kuperberg, Lehtinen, and Skrzypczak [9] confirmed the conjecture for coBüchi automata, via a more involved and non-constructive argument. More recently, Lehtinen and Prakash [52] proved the conjecture for parity automata, giving a polynomial-time algorithm based on the 2-token game for a fixed number of priorities. Their proof is constructive, as they obtain a strategy for Eve in the letter game based on her strategy for the 2-token game.

▶ **Theorem 6** ([52]). *For every nondeterministic parity automaton $\mathcal{A}$, Eve wins the 2-token game on $\mathcal{A}$ if and only if $\mathcal{A}$ is GFG.*

Consequently, deciding GFGness is in **P** for parity automata with a bounded number of priorities. When the number of priorities is part of the input, the problem becomes **NP**-hard [64] and is in **PSPACE**; its exact complexity remains open.

The minimisation problem for GFG Büchi automata, whether there is a language equivalent GFG Büchi automaton with at most k states, is **NP**-complete [69, Theorem 14] (matching the complexity for state-based GFG Büchi automata [72]). The upper bound follows from the fact that checking whether a parity automaton is GFG and language equivalent to a given GFG parity automaton is in **NP** [72, Theorem 4]. Accordingly, one can nondeterministically guess a smaller Büchi automaton as a candidate solution and verify its correctness by checking GFGness and language equivalence.

▶ **Theorem 7** ([69]). *Deciding whether there is a language equivalent GFG NBA with at most k states is **NP**-complete.*

For GFG coBüchi automata, the picture is more promising. Abu Radi and Kupferman showed canonicity and gave a polynomial-time algorithm that minimises any GFG coBüchi automaton to a canonical minimal GFG automaton [70] (which is fundamentally different to the **NP**-completeness of minimising state-based GFG coBüchi automata [72]). This result has already proven useful in accelerating Markov chain model checking [57] and stochastic planning [81], which we will briefly discuss in Section 6.

On the foundational side, it has also enabled extensions of normal forms to generalised GFG coBüchi automata [21], full ω-regular languages, notably through chains-of-coBüchi automata (COCOA) [31] and rerailing automata [29].

▶ **Theorem 8** ([70]). *Every GFG NCA can be minimised into a canonical minimal GFG NCA in polynomial time.*

The above results apply to ω-automata only with transition-based acceptance. For state-based acceptance automata, the minimisation problem is **NP**-complete for GFG Büchi, coBüchi, and parity automata [72].

4.1 Related work and Generalisation

There is a rich body of work on GFG and history-deterministic automata. As noted above, the two notions coincide for ω-regular languages [12]. Beyond classical Büchi and coBüchi conditions, GFG automata have been studied for generalised Büchi and coBüchi [21], Rabin [20], Muller, parity, and Rabin conditions [27, 19], and alternating automata [12, 10]. In quantitative automata, however, GFG-ness and history determinism no longer coincide [13]; token-game techniques were later extended to decide history determinism in this setting [14]. The study has also expanded to richer models, including ω-pushdown automata [53], timed automata [16], one-counter nets [65], and Parikh automata [32].

GFG automata are generalised to explorable automata, where nondeterministic choices can be resolved by building finitely many simultaneous runs instead of just one [41]. It is k-explorable if the number of runs built simultaneously is k. Formally, one can extend the letter game to a k-explorability game by giving Eve k tokens, and Eve wins if either the word is not in the language of the automaton, or one of the k constructed runs is accepting. An automaton is called k-explorable if Eve wins the k-explorability game and is called explorable if it is k-explorable for some $k \in \mathbb{N}$. Obviously, a GFG automaton is 1-explorable. There are examples of automata that are k-explorable but not $(k-1)$-explorable for any k. Deciding whether an NFA or an NBA is explorable was shown to be decidable and **EXPTIME**-complete. They also studied the notion of ω-explorable for infinite word automata, that is, countably many runs are built simultaneously. It was shown all reachability automata are ω-explorable, but this is not the case for safety ones. It was also shown that deciding whether a safety automaton or a coBüchi automaton is ω-explorable is **EXPTIME**-complete. Observe that the finite/countable/uncountable hierarchy of explorability exhibits a similar structure to the finite/polynomial/exponential ambiguity hierarchy as noted in [41]. More recently, these notions have also been extended to pushdown automata [6]. Other recent works have generalised the resolver strategy to stochastic resolver, which gives a probabilistic automata and words are only needed to be accepted with a threshold probability [62, 45].

4.2 Applications of GFG Automata

The game-theoretic motivation of GFG automata was discussed above; here we focus on concrete applications. One synthesis application is given in [30], which extends *GR(1)* (Generalised Reactivity(1)) synthesis. Recall that a GR(1) specification consists of safety assumptions and guarantees together with finitely many fairness assumptions and guarantees, and admits symbolic synthesis algorithms based on nested fixpoint computations. The approach of [30] keeps the safety part still as a symbolic game graph while representing the liveness part by COCOA, rather than by the *GR(1)* (Generalised Reactivity(1)) fragment of LTL. This fragment admits efficient symbolic synthesis algorithms based on nested fixpoint computations. This yields a fixpoint formula that can be evaluated symbolically on the game graph, retaining much of the efficiency of GR(1) while supporting specifications beyond the GR(1) fragment.

GFG automata have also been connected to relaxed synthesis notions such as *good-enough synthesis* [1] and *best-value synthesis* [33]. Unlike classical Church synthesis, which requires the system to satisfy the specification against all environment behaviours, these relaxed notions ask for systems that succeed whenever a satisfying interaction with the environment is possible. Deciding good-enough synthesis for deterministic specifications is polynomially equivalent to deciding whether an automaton is GFG. Intuitively, the GFG resolver corresponds to an online strategy that incrementally constructs accepting behaviours during interaction with the environment. For further discussion, see [15].

Interestingly, for GFG automata, language inclusion coincides with fair simulation [43], reducing semantic model-checking questions to simpler game-theoretic simulation problems.

Klein et al. [47] shows that GFG automata can also be used in probabilistic model checking. Despite this theoretical potential, GFG automata have not yet been widely adopted in state-of-the-art verification tools for ω-regular properties, such as ePMC [34], PRISM [51], IscasMC [36], and Storm [42], mainly due to the lack of efficient and practical constructions.

5 Good-for-MDPs Automata

GFG automata resolve nondeterminism against all possible adversarial inputs. In probabilistic control, however, the relevant question is different: does the automaton preserve optimal satisfaction probabilities when combined with an MDP? This motivates the weaker semantic notion of good-for-MDPs (GFM) automata [38], which characterise nondeterministic automata suitable for the analysis of finite-state MDPs. Intuitively, an automaton is GFM if its nondeterminism can be resolved in a probabilistic setting without changing the optimal satisfaction probability.

Several well-studied automata classes, including standard classes of limit-deterministic automata (LDBAs) [78, 25, 35, 76, 37] and slim automata [38], fall within this framework. Limit determinism means that the automaton is deterministic on all states reachable after taking an accepting transition. Note, however, that LDBAs are *not* GFM in general, as illustrated by Figure 4. The automaton there is clearly limit-deterministic, but not GFM: once q_1 is chosen, one loses words with only finitely many 0's, and symmetrically, choosing q_2 loses words with only finitely many 1's. Thus the nondeterministic choice cannot be resolved on the fly in an MDP. Note, however, that adding a self-loop on q_0 labelled by both 0 and 1 makes the automaton GFM, see Figure 5, as the choice between q_1 and q_2 can then be deferred. The definition of GFM is semantic in nature and is based on the syntactic product construction of the automaton and an MDP, as formalised in [38].

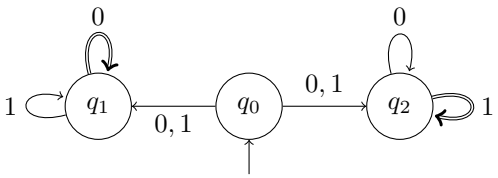

Figure 4 An LDBA that is not GFM.

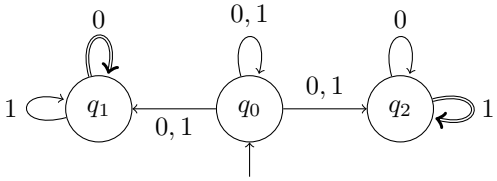

Figure 5 A GFM Büchi automaton recognising the universal language over $\{0, 1\}$.

Formally, let $\mathcal{M} = (S, \mathrm{Act}, \Sigma, \mathsf{P}, s_0, L)$ be an MDP and $\mathcal{A} = (Q, \{q_0\}, \delta, \alpha)$ be an NBA. We define the product MDP $\mathcal{M} \times \mathcal{A} = (S^\times, \mathrm{Act}^\times, \Sigma, \mathsf{P}^\times, \langle s_0, q_0\rangle, L^\times, \alpha^\times)$ augmented with the acceptance condition $\alpha^\times$ where

- $S^\times = S \times Q$ is the state space.
- $\mathrm{Act}^\times = \mathrm{Act} \times Q$ is the action set.
- $\mathsf{P}^\times : S^\times \times \mathrm{Act}^\times \times S^\times \to [0, 1]$ is the transition probability function such that $\mathsf{P}^\times(\langle s, q\rangle, \langle a, q'\rangle, \langle s', q'\rangle) = \mathsf{P}(s, a, s')$ if $\mathsf{P}(s, a, s') > 0$ and $q' \in \delta(q, L(s, a))$.
- $L^\times(\langle s, q\rangle, \langle a, q'\rangle) = L(s, a)$ for state $\langle s, q\rangle \in S \times Q$ and action $\langle a, q'\rangle \in \mathrm{Act}^\times$, and
- For Büchi, $\alpha^\times = \{(\langle s, q\rangle, \langle a, q'\rangle, \langle s', q'\rangle) \mid \mathsf{P}^\times(\langle s, q\rangle, \langle a, q'\rangle, \langle s', q'\rangle) > 0, (q, L(s, a), q') \in \alpha\}$.

The *semantic* satisfaction probability of $\mathcal{A}$ on $\mathcal{M}$ is given by

$$\mathsf{Psem}(\mathcal{M}, \mathcal{A}) = \sup_{\mu} \mathbb{P}_{\mathcal{M}^\mu}(\mathcal{A}),$$

which is the best probability, over all strategies of $\mathcal{M}$, that the trace generated by $\mathcal{M}$ lies in $\mathcal{A}$. The *syntactic* satisfaction probability is obtained instead from the product MDP: it is the best probability that a run of $\mathcal{M} \times \mathcal{A}$ is accepting. An MEC of $\mathcal{M} \times \mathcal{A}$ is *accepting* if its sub-MDP contains an accepting transition of $\alpha^\times$; once a run enters such an MEC, a suitable strategy almost surely satisfies the Büchi condition. Writing X for the union of the accepting MECs, the syntactic probability is therefore

$$\mathsf{Psyn}(\mathcal{M}, \mathcal{A}) = \sup_{\mu} \mathbb{P}_{(\mathcal{M}\times\mathcal{A})^\mu}(\mathsf{F}X).$$

Since $\mathsf{F}X$ is a reachability objective, this supremum is attained by a memoryless strategy of $\mathcal{M} \times \mathcal{A}$; computing $\mathsf{Psyn}(\mathcal{M}, \mathcal{A})$ thus reduces to a maximal-reachability computation on $\mathcal{M} \times \mathcal{A}$. An automaton $\mathcal{A}$ is *GFM* if $\mathsf{Psem}(\mathcal{M}, \mathcal{A}) = \mathsf{Psyn}(\mathcal{M}, \mathcal{A})$ for every finite-state MDP $\mathcal{M}$. Note that a memoryless strategy of $\mathcal{M} \times \mathcal{A}$ corresponds, in general, to a finite-memory strategy of $\mathcal{M}$ that uses the automaton component as memory; this is why Psem must range over arbitrary strategies of $\mathcal{M}$, even though Psyn is attained memorylessly on the product.

Every GFG automaton is also GFM, but the converse does not hold, as first observed in [38, Figure 3]. A simplified example is shown in Figure 5. In this example, the automaton is not GFG. Indeed, in the letter game, Eve must eventually commit to either q_1 or q_2 while reading the word online. Choosing q_1 loses words with infinitely many 1's and only finitely many 0's, whereas choosing q_2 loses words with infinitely many 0's and only finitely many 1's. Since the future behaviour of the word is not yet known, no online resolver can always make the correct choice.

The situation is different in the MDP setting. A GFM automaton does not require nondeterminism to be resolved uniformly for all possible futures. Instead, it only requires that the optimal satisfaction probability be preserved after taking the product with the MDP. In the automaton of Figure 5, the choice between q_1 and q_2 can therefore be postponed until sufficient information about the long-run behaviour of the MDP has been revealed. Intuitively, once the run enters an end-component that produces infinitely many 0's almost surely, the strategy moves to q_1; symmetrically, it moves to q_2 in an end-component producing infinitely many 1's almost surely. Thus, unlike GFG automata, GFM automata may exploit the probabilistic and asymptotic structure of the MDP when resolving nondeterminism. This makes GFM strictly more permissive than GFG while still preserving correctness for probabilistic verification and control.

In terms of expressiveness, GFM Büchi automata recognise the full class of ω-regular languages, matching the expressive power of general nondeterministic automata. GFM automata also enjoy strong succinctness advantages over deterministic and GFG automata, as witnessed by the automaton shown in Figure 6 from [74]. Let $n \in \mathbb{N}$. This GFM automaton, with $n+2$ states, recognises the language over $\{0, 1, \$\}$ of infinite words that contain infinitely many sequences of the form $1\,\{0,1\}^{n-1}\$$. Assume, towards a contradiction, that there exists a DBA $\mathcal{A}$ recognising the language with fewer than 2^{n-1} states. By a pigeonhole argument over words in $\{0,1\}^n$, at least three such words lead to the same state, among which two have runs from q_0 that either both visit an accepting state or both avoid accepting states altogether. From these two words, one can construct distinguishing suffixes and iterate the argument inductively, yielding two infinite words, one in the language and one outside it, whose runs in $\mathcal{A}$ repeatedly merge and either both satisfy the Büchi condition or both fail it

– a contradiction. Since GFG Büchi automata are only up to quadratically more succinct than DBA for the same language [48], it follows that GFM Büchi automata are exponentially more succinct than GFG Büchi automata.

▶ **Theorem 9** ([74]). *GFM Büchi automata are exponentially more succinct than DBAs. GFM Büchi automata are exponentially more succinct than GFG Büchi automata.*

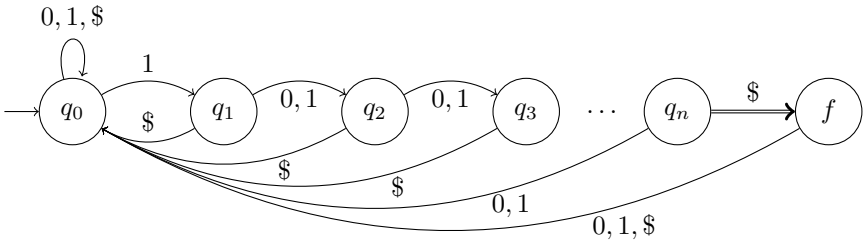

Figure 6 The GFM Büchi automaton recognising the language over $\{0, 1, \$\}$ of infinite words that contain infinitely many sequences of the form $1\,\{0,1\}^{n-1}\$$.

To check whether or not a given NBA is GFM is intractable, according to [73, Theorem 9, Corollary 18], and the exact complexity remains an open problem:

▶ **Theorem 10** ([73]). *Deciding whether an NBA is GFM is **PSPACE**-hard and in **EXPTIME**.*

The minimisation problem for GFM Büchi automata, whether there is a language equivalent GFM NBA with at most k states, is **PSPACE**-hard [73, Theorem 10], and lies in **EXPTIME** by enumerating all automata of size up to k and checking language equivalence (in **PSPACE**) together with GFMness (currently in **EXPTIME**). Its exact complexity remains open; in particular, a **PSPACE** procedure for checking GFMness would immediately yield a **PSPACE** upper bound for minimisation as well. The lower bound is known to be **PSPACE**-hard:

▶ **Theorem 11** ([73]). *Deciding whether there is a language equivalent GFM NBA with at most k states is **PSPACE**-hard. It is **PSPACE**-hard even for (fixed) $k = 2$.*

The GFM property has also been studied for alternating automata [39]. In particular, alternating GFM automata were shown to be exponentially more succinct than general NBAs.

5.1 Applications

Given an MDP $\mathcal{M}$ and an NBA $\mathcal{A}$, the *model checking* problem is to compute the maximal probability, over all strategies of $\mathcal{M}$, that a generated trajectory is accepted by $\mathcal{A}$. For GFM automata, this value can be computed directly from their semantic definition: it coincides with $\mathsf{Psem}(\mathcal{M}, \mathcal{A})$ and can be obtained by computing $\mathsf{Psyn}(\mathcal{M}, \mathcal{A})$. This is achieved by constructing the product MDP of $\mathcal{M}$ and $\mathcal{A}$ and computing the maximal probability of reaching an accepting MEC in the product.

In practice, unlike in the standard MDP model checking setting, the transition probabilities of the MDP modelling the environment are often not known a priori. *Reinforcement learning (RL)* provides an effective framework for such scenarios, allowing an agent to interact with

the environment and improve its behaviour based on feedback obtained during execution. Formally, the goal is to synthesise a strategy that maximises the probability that an MDP $\mathcal{M}$ with unknown transition probabilities satisfies a given ω-regular specification φ.

GFM automata have proved particularly useful in this context, including standard classes of LDBAs [78, 25, 35, 76, 37] and slim automata [38]. A natural approach is to first translate the objective φ into a suitable automaton, then construct the product of this automaton with the MDP modelling the environment. One then defines a reward function on the product MDP, which serves as feedback for the learning agent, and applies an off-the-shelf RL algorithm to learn an optimal strategy. Crucially, the automaton can be chosen to be GFM, ensuring that optimal control in the product MDP corresponds to an optimal strategy in the original MDP that maximises the probability of satisfying φ.

6 Combining the Models: Using GFG Minimisation for Probabilistic Model Checking

The previous sections treated unambiguous, GFG, and GFM automata as separate restrictions. Recent work shows that these notions can also be combined algorithmically: GFG minimisation can be used as a preprocessing step to reduce automata that are then used in probabilistic analysis.

6.1 Markov Chain Model Checking

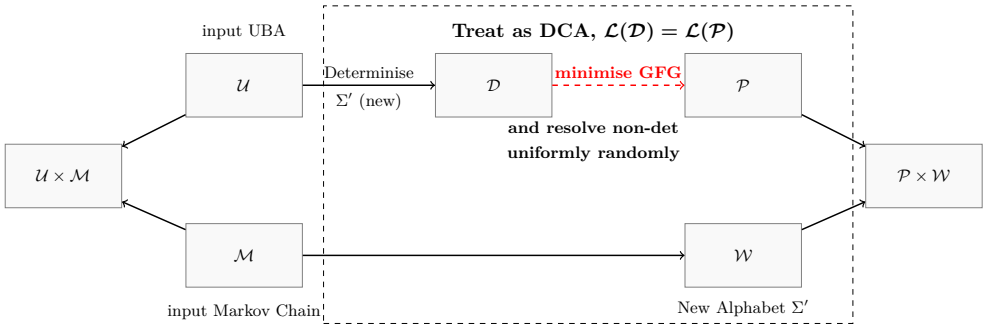

Figure 7 Overview of the pipeline constructions in [57].

As discussed in previous sections, unambiguous and GFG automata have been applied independently to Markov chain model checking for nearly a decade. These two restrictions of nondeterminism provide orthogonal generalisations of deterministic automata. However, combining them directly is of limited interest, since automata that are both GFG and unambiguous essentially collapse to deterministic automata [11].

To see the intuition behind this collapse, let $\mathcal{A}$ be an automaton that is both GFG and unambiguous. Consider a state with several a-successors. Since $\mathcal{A}$ is GFG, one of these successors must recognise a language containing the languages of all other a-successors. Indeed, suppose there are two successors p and q such that neither language contains the other, and moreover neither language is contained in the language of any other a-successor. Then there exists a word u accepted from p but not from any other a-successor, and symmetrically a word v accepted from q but not from any other a-successor. A resolver reading the initial letter a would therefore have to choose between p and q without knowledge of the future suffix: choosing p loses the word v, while choosing q loses the word u. Hence no GFG resolver exists, contradicting the assumption that $\mathcal{A}$ is GFG. Consequently, every set of a-successors

contains a successor whose language subsumes the languages of all others. Transitions to the remaining successors can therefore be pruned without changing the recognised language. Thus GFG automata may be assumed to be *semantically deterministic*, meaning that all remaining a-successors of a state recognise the same language. For further discussion of semantically deterministic automata and their relation to GFG automata, we refer to [71]. Since $\mathcal{A}$ is also unambiguous, there can be at most one such successor, as otherwise the same word would admit two distinct accepting runs. Hence every state has at most one successor for each input letter, and the automaton is deterministic up to semantic equivalence.

A recent work [57] reconciles these approaches via a non-trivial construction that integrates unambiguous automata with the polynomial-time minimisation of GFG coBüchi automata due to Abu Radi and Kupferman [70]. The key insight is to reinterpret unambiguous behaviour through a controlled introduction of probabilistic choices, enabling the use of GFG minimisation techniques while preserving quantitative semantics. More concretely, an input UBA is transformed into a probabilistic automaton [3] that is suitable for quantitative analysis over Markov chains. The automata has the additional property that every word is accepted with probability either 1 or 0. The resulting automata can be exponentially smaller than their deterministic counterparts [57]. A family of examples demonstrates significant state-space reductions, suggesting substantial speed-ups in model checking for LTL [28, 5], LDL, weak alternating automata [58], and UBAs in general, especially when the input Markov chains are large, as is typical in practical applications.

While the standard UBA-based approach [5] constructs the product of the input UBA and the Markov chain directly, the accelerated method proceeds via a polynomial sequence of transformations, illustrated in Figure 7:

1. The input UBA $\mathcal{U}$ is determinised into a DBA over an extended alphabet Σ', where each letter encodes successor information. This step encodes the nondeterministic choices explicitly at the level of the alphabet.
2. The DBA is then reinterpreted as a deterministic coBüchi automaton (DCA) $\mathcal{D}$ by dualising the acceptance condition. The polynomial-time GFG coBüchi minimisation procedure [70] is then applied to obtain a minimal GFG automaton while preserving language equivalence.
3. The resulting GFG automaton is transformed into a probabilistic automaton $\mathcal{P}$ by resolving nondeterministic choices via a uniform (or otherwise well-defined) probability distribution. This step is crucial: it preserves acceptance probabilities while enabling linear-algebraic analysis.

Before constructing the product, the input Markov chain $\mathcal{M}$ is lifted to a weighted structure $\mathcal{W}$ over the extended alphabet Σ', ensuring compatibility with the transformed automaton. The product $\mathcal{P} \times \mathcal{W}$ is then formed. As in the classical UBA-based approach, this induces a system of linear equations (cf. Equation (1)). By adding suitable constraints for strongly connected components with specific acceptance properties, one obtains a unique solution corresponding to the probability that a trajectory generated by $\mathcal{M}$ is accepted by $\mathcal{U}$. Compared to the direct product construction, the main advantage of this approach lies in separating nondeterminism resolution from the product construction, thereby enabling minimisation prior to composition and yielding substantial performance gains in practice.

6.2 MDP Model Checking

As discussed earlier, GFM automata can be used for MDP model checking by constructing the product with the input MDP and analysing the resulting system directly. Recall that the objective is to compute the maximal probability, over all strategies of a given MDP, that a generated trajectory is accepted by a GFM automaton.

Recent work [81] improves this approach by combining GFM automata with the polynomial-time minimisation of GFG coBüchi automata, in close analogy to the advances described above for Markov chains. The central idea is again to preprocess the automaton so that nondeterminism is handled in a structured and optimisable way before constructing the product. The input GFM automaton is transformed through a polynomial sequence of steps, similar to the one in Figure 7, yielding a probabilistic automaton over an extended alphabet Σ'. This construction parallels the transformation of unambiguous automata in the Markov chain setting. Simultaneously, the input MDP is adapted to operate over Σ' by refining its action space so that transitions encode the additional information required by the transformed automaton. This results in a modified MDP $\mathcal{M}'$ that is behaviourally equivalent with respect to the property under consideration. Intuitively, the nondeterminism of the input GFM automaton is transferred to the MDP by extending the action space of $\mathcal{M}$ to obtain $\mathcal{M}'$. The additional actions encode the choices that would otherwise be resolved nondeterministically by the automaton. Consequently, the transformed automaton can be viewed as deterministic over the extended alphabet Σ', since its nondeterministic choices are now resolved through the strategy of the MDP.

7 Conclusion

Overall, the survey shows that limited nondeterminism yields compact automata models that remain amenable to model checking, synthesis, and reinforcement learning.

A common lesson across these models is that nondeterminism need not be eliminated completely. Instead, it can be constrained according to the surrounding algorithmic context: unique runs for quantitative analysis over Markov chains, history-dependent resolvability for games, and probability preservation for MDPs. This perspective also explains why recent reduction pipelines first simplify the automaton before composing it with the system model.

Several problems remain open. Since their introduction, GFG automata have become increasingly well understood from a theoretical perspective, and their potential for solving games and synthesis is compelling. We believe that the next major challenge is to turn this potential into practical synthesis tools, as has been achieved for GFM and unambiguous automata. A key step in this direction is to translate full ω-regular specifications into GFG automata in a way that fulfils this promise.

Another open question is whether language inclusion for UBAs can be solved in polynomial time, as is the case for UFAs. Finally, the exact complexity of deciding whether a parity automaton is GFG, or whether a Büchi automaton is GFM, remains open.

References

1 Shaull Almagor and Orna Kupferman. Good-enough synthesis. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II*, Lecture Notes in Computer Science, pages 541–563. Springer, 2020. doi:10.1007/978-3-030-53291-8_28.

2 Marc Bagnol and Denis Kuperberg. Büchi good-for-games automata are efficiently recognizable. In Sumit Ganguly and Paritosh K. Pandya, editors, *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, Ahmedabad, India, December 11-13, 2018*, LIPIcs, pages 16:1–16:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.16.

3 Christel Baier, Marcus Größer, and Nathalie Bertrand. Probabilistic ω-automata. *J. ACM*, 59(1):1:1–1:52, 2012. doi:10.1145/2108242.2108243.

4 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

5 Christel Baier, Stefan Kiefer, Joachim Klein, David Müller, and James Worrell. Markov chains and unambiguous automata. *J. Comput. Syst. Sci.*, 136:113–134, 2023. doi:10.1016/J.JCSS.2023.03.005.

6 Ayaan Bedi and Karoliina Lehtinen. Explorability in pushdown automata. In C. Aiswarya, Ruta Mehta, and Subhajit Roy, editors, *45th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2025, BITS Pilani, K K Birla Goa Campus, India, December 17-19, 2025*, LIPIcs, pages 12:1–12:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.FSTTCS.2025.12.

7 Michael Benedikt, Rastislav Lenhardt, and James Worrell. Model checking markov chains against unambiguous buchi automata. *CoRR*, abs/1405.4560, 2014. arXiv:1405.4560.

8 Udi Boker, Denis Kuperberg, Orna Kupferman, and Michal Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, Lecture Notes in Computer Science, pages 89–100. Springer, 2013. doi:10.1007/978-3-642-39212-2_11.

9 Udi Boker, Denis Kuperberg, Karoliina Lehtinen, and Michal Skrzypczak. On succinctness and recognisability of alternating good-for-games automata. *CoRR*, abs/2002.07278, 2020. arXiv:2002.07278.

10 Udi Boker, Denis Kuperberg, Karoliina Lehtinen, and Michal Skrzypczak. On the succinctness of alternating parity good-for-games automata. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020, BITS Pilani, K K Birla Goa Campus, Goa, India (Virtual Conference), December 14-18, 2020*, LIPIcs, pages 41:1–41:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.FSTTCS.2020.41.

11 Udi Boker, Orna Kupferman, and Michal Skrzypczak. How deterministic are good-for-games automata? In Satya V. Lokam and R. Ramanujam, editors, *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India*, volume 93 of *LIPIcs*, pages 18:1–18:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.FSTTCS.2017.18.

12 Udi Boker and Karoliina Lehtinen. Good for games automata: From nondeterminism to alternation. In Wan J. Fokkink and Rob van Glabbeek, editors, *30th International Conference on Concurrency Theory, CONCUR 2019, Amsterdam, The Netherlands, August 27-30, 2019*, LIPIcs, pages 19:1–19:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CONCUR.2019.19.

13 Udi Boker and Karoliina Lehtinen. History determinism vs. good for gameness in quantitative automata. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, Virtual Conference, December 15-17, 2021*, LIPIcs, pages 38:1–38:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.38.

14 Udi Boker and Karoliina Lehtinen. Token games and history-deterministic quantitative-automata. *Log. Methods Comput. Sci.*, 19(4), 2023. doi:10.46298/LMCS-19(4:8)2023.

15 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.

16 Sougata Bose, Thomas A. Henzinger, Karoliina Lehtinen, Sven Schewe, and Patrick Totzke. History-deterministic timed automata. *Log. Methods Comput. Sci.*, 20(4), 2024. doi:10.46298/LMCS-20(4:1)2024.

17 Nicolas Bousquet and Christof Löding. Equivalence and inclusion problem for strongly unambiguous büchi automata. In Adrian-Horia Dediu, Henning Fernau, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications, 4th International Conference, LATA 2010, Trier, Germany, May 24-28, 2010. Proceedings*, Lecture Notes in Computer Science, pages 118–129. Springer, 2010. doi:10.1007/978-3-642-13089-2_10.

18 Olivier Carton and Max Michel. Unambiguous büchi automata. *Theor. Comput. Sci.*, 297(1-3):37–81, 2003. doi:10.1016/S0304-3975(02)00618-7.

19 Antonio Casares, Thomas Colcombet, Nathanaël Fijalkow, and Karoliina Lehtinen. From muller to parity and rabin automata: Optimal transformations preserving (history) determinism. *TheoretiCS*, 3, 2024. doi:10.46298/THEORETICS.24.12.

20 Antonio Casares, Thomas Colcombet, and Karoliina Lehtinen. On the size of good-for-games rabin automata and its link with the memory in muller games. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, Paris, France, July 4-8, 2022*, LIPIcs, pages 117:1–117:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.117.

21 Antonio Casares, Olivier Idir, Denis Kuperberg, Corto Mascle, and Aditya Prakash. On the minimisation of deterministic and history-deterministic generalised (co)büchi automata. In Jörg Endrullis and Sylvain Schmitz, editors, *33rd EACSL Annual Conference on Computer Science Logic, CSL 2025, Amsterdam, The Netherlands, February 10-14, 2025*, LIPIcs, pages 22:1–22:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.CSL.2025.22.

22 Antonio Casares, Aditya Prakash, and K. S. Thejaswini. History-deterministic büchi automata are succinct. *CoRR*, abs/2603.05380, 2026. doi:10.48550/arXiv.2603.05380.

23 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*, Lecture Notes in Computer Science, pages 139–150. Springer, 2009. doi:10.1007/978-3-642-02930-1_12.

24 Thomas Colcombet. Unambiguity in automata theory. In Jeffrey O. Shallit and Alexander Okhotin, editors, *Descriptional Complexity of Formal Systems - 17th International Workshop, DCFS 2015, Waterloo, ON, Canada, June 25-27, 2015. Proceedings*, Lecture Notes in Computer Science, pages 3–18. Springer, 2015. doi:10.1007/978-3-319-19225-3_1.

25 Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, July 1995. doi:10.1145/210332.210339.

26 Jean-Michel Couvreur, Nasser Saheb, and Grégoire Sutre. An optimal automata approach to LTL model checking of probabilistic systems. In Moshe Y. Vardi and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 10th International Conference, LPAR 2003, Almaty, Kazakhstan, September 22-26, 2003, Proceedings*, Lecture Notes in Computer Science, pages 361–375. Springer, 2003. doi:10.1007/978-3-540-39813-4_26.

27 Daniele Dell'Erba, Sven Schewe, Qiyi Tang, and Tansholpan Zhanabekova. Semantic flowers for good-for-games and deterministic automata. *Inf. Process. Lett.*, 185:106468, 2024. doi:10.1016/J.IPL.2023.106468.

28 Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu. Spot 2.0 - A framework for LTL and omega-automata manipulation. In Cyrille Artho, Axel Legay, and Doron Peled, editors, *Automated Technology for Verification and Analysis - 14th International Symposium, ATVA 2016, Chiba, Japan, October 17-20, 2016, Proceedings*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129, 2016. doi:10.1007/978-3-319-46520-3_8.

29 Rüdiger Ehlers. Rerailing automata. *CoRR*, abs/2503.08438, 2025. doi:10.48550/arXiv.2503.08438.

30 Rüdiger Ehlers and Ayrat Khalimov. Fully generalized reactivity(1) synthesis. In Bernd Finkbeiner and Laura Kovács, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 30th International Conference, TACAS 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part I*, Lecture Notes in Computer Science, pages 83–102. Springer, 2024. doi:10.1007/978-3-031-57246-3_6.

31 Rüdiger Ehlers and Ayrat Khalimov. A naturally-colored translation from LTL to parity and COCOA. *CoRR*, abs/2410.01021, 2024. doi:10.48550/arXiv.2410.01021.

32 Enzo Erlich, Mario Grobler, Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. History-deterministic parikh automata. *ACM Trans. Comput. Log.*, 26(3):18:1–18:36, 2025. doi:10.1145/3742431.

33 Emmanuel Filiot, Christof Löding, and Sarah Winter. Synthesis from weighted specifications with partial domains over finite words. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020, BITS Pilani, K K Birla Goa Campus, Goa, India (Virtual Conference), December 14-18, 2020*, LIPIcs, pages 46:1–46:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.FSTTCS.2020.46.

34 Chen Fu, Ernst Moritz Hahn, Yong Li, Sven Schewe, Meng Sun, Andrea Turrini, and Lijun Zhang. EPMC gets knowledge in multi-agent systems. In Bernd Finkbeiner and Thomas Wies, editors, *Verification, Model Checking, and Abstract Interpretation - 23rd International Conference, VMCAI 2022, Philadelphia, PA, USA, January 16-18, 2022, Proceedings*, volume 13182 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2022. doi:10.1007/978-3-030-94583-1_5.

35 E. M. Hahn, G. Li, S. Schewe, A. Turrini, and L. Zhang. Lazy probabilistic model checking without determinisation. In *Concurrency Theory*, pages 354–367, 2015.

36 Ernst Moritz Hahn, Yi Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. ISCASMC: A web-based probabilistic model checker. In Cliff B. Jones, Pekka Pihlajasaari, and Jun Sun, editors, *FM 2014: Formal Methods - 19th International Symposium, Singapore, May 12-16, 2014. Proceedings*, volume 8442 of *Lecture Notes in Computer Science*, pages 312–317. Springer, 2014. doi:10.1007/978-3-319-06410-9_22.

37 Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular objectives in model-free reinforcement learning. In Tomás Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part I*, volume 11427 of *Lecture Notes in Computer Science*, pages 395–412. Springer, 2019. doi:10.1007/978-3-030-17462-0_27.

38 Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Good-for-mdps automata for probabilistic analysis and reinforcement learning. In Armin Biere and David Parker, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part I*, Lecture Notes in Computer Science, pages 306–323. Springer, 2020. doi:10.1007/978-3-030-45190-5_17.

39 Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Alternating good-for-mdps automata. In Ahmed Bouajjani, Lukás Holík, and Zhilin Wu, editors, *Automated Technology for Verification and Analysis - 20th International Symposium, ATVA 2022, Virtual Event, October 25-28, 2022, Proceedings*, Lecture Notes in Computer Science, pages 303–319. Springer, 2022. doi:10.1007/978-3-031-19992-9_19.

40 Yo-Sub Han, Arto Salomaa, and Kai Salomaa. Ambiguity, nondeterminism and state complexity of finite automata. *Acta Cybern.*, 23(1):141–157, 2017. doi:10.14232/ACTACYB.23.1.2017.9.

41 Emile Hazard and Denis Kuperberg. Explorable automata. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual Conference on Computer Science Logic, CSL 2023, Warsaw, Poland, February 13-16, 2023*, LIPIcs, pages 24:1–24:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CSL.2023.24.

42 Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. The probabilistic model checker storm. *Int. J. Softw. Tools Technol. Transf.*, 24(4):589–610, 2022. doi:10.1007/S10009-021-00633-Z.

43 Thomas A. Henzinger, Orna Kupferman, and Sriram K. Rajamani. Fair simulation. *Inf. Comput.*, 173(1):64–81, 2002. doi:10.1006/INCO.2001.3085.

44 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, Lecture Notes in Computer Science, pages 395–410. Springer, 2006. doi:10.1007/11874683_26.

45 Thomas A. Henzinger, Aditya Prakash, and K. S. Thejaswini. Resolving nondeterminism with randomness. In Pawel Gawrychowski, Filip Mazowiecki, and Michal Skrzypczak, editors, *50th International Symposium on Mathematical Foundations of Computer Science, MFCS 2025, Warsaw, Poland, August 25-29, 2025*, LIPIcs, pages 57:1–57:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.MFCS.2025.57.

46 Tao Jiang and Bala Ravikumar. Minimal NFA problems are hard. *SIAM J. Comput.*, 22(6):1117–1141, 1993. doi:10.1137/0222067.

47 Joachim Klein, David Müller, Christel Baier, and Sascha Klüppelholz. Are good-for-games automata good for probabilistic model checking? In Adrian-Horia Dediu, Carlos Martín-Vide, José Luis Sierra-Rodríguez, and Bianca Truthe, editors, *Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings*, volume 8370 of *Lecture Notes in Computer Science*, pages 453–465. Springer, 2014. doi:10.1007/978-3-319-04921-2_37.

48 Denis Kuperberg and Michal Skrzypczak. On determinisation of good-for-games automata. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 2015. doi:10.1007/978-3-662-47666-6_24.

49 Orna Kupferman. Using the past for resolving the future. *Frontiers Comput. Sci.*, 4, 2022. doi:10.3389/FCOMP.2022.1114625.

50 Orna Kupferman, Shmuel Safra, and Moshe Y. Vardi. Relating word and tree automata. *Ann. Pure Appl. Log.*, 138(1-3):126–146, 2006. doi:10.1016/J.APAL.2005.06.009.

51 M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011. doi:10.1007/978-3-642-22110-1_47.

52 Karoliina Lehtinen and Aditya Prakash. The 2-token theorem: Recognising history-deterministic parity automata efficiently. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 1839–1850. ACM, 2025. doi:10.1145/3717823.3718310.

53 Karoliina Lehtinen and Martin Zimmermann. Good-for-games ω-pushdown automata. *Log. Methods Comput. Sci.*, 18(1), 2022. doi:10.46298/LMCS-18(1:3)2022.

54 Ernst L. Leiss. Succint representation of regular languages by boolean automata. *Theor. Comput. Sci.*, 13:323–330, 1981. doi:10.1016/S0304-3975(81)80005-9.

55 Hing Leung. Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. *SIAM J. Comput.*, 27(4):1073–1082, 1998. doi:10.1137/S0097539793252092.

56 Hing Leung. Descriptional complexity of nfa of different ambiguity. *Int. J. Found. Comput. Sci.*, 16(5):975–984, 2005. doi:10.1142/S0129054105003418.

57 Yong Li, Soumyajit Paul, Sven Schewe, and Qiyi Tang. Accelerating markov chain model checking: Good-for-games meets unambiguous automata. In Ruzica Piskac and Zvonimir Rakamaric, editors, *Computer Aided Verification - 37th International Conference, CAV 2025, Zagreb, Croatia, July 23-25, 2025, Proceedings, Part II*, Lecture Notes in Computer Science, pages 276–298. Springer, 2025. doi:10.1007/978-3-031-98679-6_13.

58 Yong Li, Sven Schewe, and Moshe Y. Vardi. Singly exponential translation of alternating weak Büchi automata to unambiguous Büchi automata. *Theor. Comput. Sci.*, 1006:114650, 2024. doi:10.1016/J.TCS.2024.114650.

59 Christof Löding and Anton Pirogov. On finitely ambiguous büchi automata. In Mizuho Hoshi and Shinnosuke Seki, editors, *Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings*, Lecture Notes in Computer Science, pages 503–515. Springer, 2018. doi:10.1007/978-3-319-98654-8_41.

60 Andreas Morgenstern. *Symbolic controller synthesis for LTL specifications*. PhD thesis, University of Kaiserslautern, 2010. URL: http://kluedo.ub.uni-kl.de/volltexte/2011/2572/index.html.

61 Damian Niwinski and Igor Walukiewicz. Relating hierarchies of word and tree automata. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *STACS 98, 15th Annual Symposium on Theoretical Aspects of Computer Science, Paris, France, February 25-27, 1998, Proceedings*, Lecture Notes in Computer Science, pages 320–331. Springer, 1998. doi:10.1007/BFB0028571.

62 Soumyajit Paul, David Purser, Sven Schewe, Qiyi Tang, Patrick Totzke, and Di-De Yen. Resolving nondeterminism by chance. In Patricia Bouyer and Jaco van de Pol, editors, *36th International Conference on Concurrency Theory, CONCUR 2025, Aarhus, Denmark, August 26-29, 2025*, LIPIcs, pages 32:1–32:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.CONCUR.2025.32.

63 Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 11-13, 1989*, pages 179–190. ACM Press, 1989. doi:10.1145/75277.75293.

64 Aditya Prakash. Checking history-determinism is np-hard for parity automata. In Naoki Kobayashi and James Worrell, editors, *Foundations of Software Science and Computation Structures - 27th International Conference, FoSSaCS 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part I*, Lecture Notes in Computer Science, pages 212–233. Springer, 2024. doi:10.1007/978-3-031-57228-9_11.

65 Aditya Prakash and K. S. Thejaswini. On history-deterministic one-counter nets. In Orna Kupferman and Pawel Sobocinski, editors, *Foundations of Software Science and Computation Structures - 26th International Conference, FoSSaCS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22-27, 2023, Proceedings*, Lecture Notes in Computer Science, pages 218–239. Springer, 2023. doi:10.1007/978-3-031-30829-1_11.

66 Alexander Rabinovich. Complementation of finitely ambiguous büchi automata. In Mizuho Hoshi and Shinnosuke Seki, editors, *Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings*, Lecture Notes in Computer Science, pages 541–552. Springer, 2018. doi:10.1007/978-3-319-98654-8_44.

67 Alexander Rabinovich and Doron Tiferet. Degrees of ambiguity for parity tree automata. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, Ljubljana, Slovenia (Virtual Conference), January 25-28, 2021*, LIPIcs, pages 36:1–36:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CSL.2021.36.

68 Alexander Rabinovich and Doron Tiferet. On degrees of ambiguity for büchi tree automata. *Inf. Comput.*, 281:104750, 2021. doi:10.1016/J.IC.2021.104750.

69 Bader Abu Radi and Rüdiger Ehlers. Characterizing the polynomial-time minimizable ω-automata. *CoRR*, abs/2504.20553, 2025. doi:10.48550/arXiv.2504.20553.

70 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.

71 Bader Abu Radi, Orna Kupferman, and Ofer Leshkowitz. A hierarchy of nondeterminism. *Log. Methods Comput. Sci.*, 21(4), 2025. doi:10.46298/LMCS-21(4:13)2025.

72 Sven Schewe. Minimising good-for-games automata is np-complete. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020, BITS Pilani, K K Birla Goa Campus, Goa, India (Virtual Conference), December 14-18, 2020*, LIPIcs, pages 56:1–56:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.FSTTCS.2020.56.

73 Sven Schewe, Qiyi Tang, and Tansholpan Zhanabekova. Deciding what is good-for-mdps. In Guillermo A. Pérez and Jean-François Raskin, editors, *34th International Conference on Concurrency Theory, CONCUR 2023, Antwerp, Belgium, September 18-23, 2023*, LIPIcs, pages 35:1–35:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CONCUR.2023.35.

74 Sven Schewe, Qiyi Tang, and Tansholpan Zhanabekova. On good-for-mdps automata, 2025. arXiv:2202.07629.

75 Erik Meineche Schmidt and Thomas G. Szymanski. Succinctness of descriptions of unambiguous context-free languages. *SIAM J. Comput.*, 6(3):547–553, 1977. doi:10.1137/0206039.

76 Salomon Sickert, Javier Esparza, Stefan Jaax, and Jan Kretínský. Limit-deterministic büchi automata for linear temporal logic. In Swarat Chaudhuri and Azadeh Farzan, editors, *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II*, Lecture Notes in Computer Science, pages 312–332. Springer, 2016. doi:10.1007/978-3-319-41540-6_17.

77 Richard Edwin Stearns and Harry B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM J. Comput.*, 14(3):598–611, 1985. doi:10.1137/0214044.

78 Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Foundations of Computer Science*, pages 327–338, 1985. doi:10.1109/SFCS.1985.12.

79 Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings of the Symposium on Logic in Computer Science (LICS '86), Cambridge, Massachusetts, USA, June 16-18, 1986*, pages 332–344. IEEE Computer Society, 1986.

80 Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theor. Comput. Sci.*, 88(2):325–349, 1991. doi:10.1016/0304-3975(91)90381-B.

81 Christoph Weinhuber, Giuseppe De Giacomo, Yong Li, Sven Schewe, and Qiyi Tang. Good-for-mdp state reduction for stochastic LTL planning. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor, editors, *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pages 36457–36465. AAAI Press, 2026. doi:10.1609/AAAI.V40I43.40967.

82 Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. Reasoning about infinite computation paths (extended abstract). In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*, pages 185–194. IEEE Computer Society, 1983. doi:10.1109/SFCS.1983.51.