Resolving Nondeterminism by Chance

Soumyajit Paul
University of Liverpool, UK

David Purser
University of Liverpool, UK

Sven Schewe
University of Liverpool, UK

Qiyi Tang
University of Liverpool, UK

Patrick Totzke
University of Liverpool, UK

Di-De Yen
University of Liverpool, UK

Abstract

History-deterministic automata are those in which nondeterministic choices can be correctly resolved stepwise: there is a strategy to select a continuation of a run given the next input letter so that if the overall input word admits some accepting run, then the constructed run is also accepting.

Motivated by checking qualitative properties in probabilistic verification, we consider the setting where the resolver strategy can randomise and only needs to succeed with lower-bounded probability. We study the expressiveness of such stochastically-resolvable automata as well as consider the decision questions of whether a given automaton has this property. In particular, we show that it is undecidable to check if a given NFA is λ-stochastically resolvable. This problem is decidable for finitely-ambiguous automata. We also present complexity upper and lower bounds for several well-studied classes of automata for which this problem remains decidable.

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.2025.32

Related Version *Full Version*: https://arxiv.org/abs/2504.10234

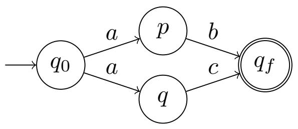
Funding This work has been supported by the European Union's Horizon Europe (HORIZON) programme under the Marie Skłodowska-Curie grant agreement No 101208673 and the EPSRC through the projects EP/X042596/1 and EP/X03688X/1.

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

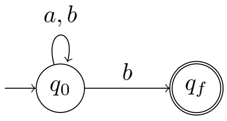
1 Introduction

Many successful verification techniques rely on automata representations for specifications that capture the languages of acceptable system traces. These automata models are typically nondeterministic: any given trace may give rise to multiple runs of the automaton, and is accepted if at least one of these runs is successful. This enables succinct representations but is also a major source of complexity due to costly intermediate determinisation steps.

It is therefore natural to put extra constraints on the extent to which an automaton allows nondeterministic choice to avoid determinisation. One way to do this is to bound the level of ambiguity: an automaton is k-ambiguous if on every accepted word it has at most k



(a) NFA $\mathcal{A}$ is 1/2-resolvable.



(b) NFA $\mathcal{B}$ is not positively resolvable.

Figure 1 Two unambiguous NFA (all missing transitions implicitly go to a non-accepting sink). The one on the left is λ-resolvable for all $\lambda \leq 1/2$. The one on the right is not positively resolvable, because no matter the choice of transition probability, the probability of b^n tends to zero as n grows.

many distinct accepting runs (see e.g. [42, 35, 16]). Unambiguous automata have proven to be useful for model-checking of Markov chain models [7, 33]. An orthogonal restriction on nondeterminism, which has been extensively studied in recent years (cf. [24, 14, 27, 3, 8, 1, 41, 9, 10, 32, 33]), is to demand that choices can be resolved "on-the-fly": An automaton is *history-deterministic* (HD) if there is a strategy to select a continuation of a run given a next letter, so that if the overall word admits some accepting run then the constructed run is also accepting. This condition is strict in the sense that the resolver strategy must guarantee to produce an accepting run if one exists. In some scenarios less strict guarantees may suffice, for example when model-checking Markov Decision Processes [22, 21, 4, 6]. This motivates the study of automata that can be resolved in a weaker sense, namely, where the resolver strategy can randomise and is only required to succeed with a lower-bounded confidence.

A *stochastic resolver* for some nondeterministic automaton $\mathcal{A}$ is a function that, for any finite run and input letter, gives a distribution over the possible transitions. This produces a probabilistic automaton $\mathcal{P}$ that assigns a probability of acceptance to every input word and, together with a threshold $\lambda > 0$, defines the language $\mathcal{L}(\mathcal{P}_{\geq\lambda})$ consisting of all words whose probability of acceptance is at least λ (see, e.g., [37]). Unless stated otherwise, we will consider *memoryless* stochastic resolvers, which base their decisions solely on the last state of the given run and the input letter. Fixing a memoryless resolver turns $\mathcal{A}$ into a probabilistic automaton $\mathcal{P}$ over the same set of states and where transitions that appear positively in $\mathcal{P}$ are also contained in $\mathcal{A}$. Consequently then, for every threshold $0 < \lambda$, the language $\mathcal{L}(\mathcal{P}_{\geq\lambda})$ is included in that of $\mathcal{A}$. We now ask which automata admit positional resolvers that also guarantee the inclusion in the other direction.

An automaton $\mathcal{A}$ called λ-*memoryless stochastically resolvable* (or simply λ-resolvable) if there exists a memoryless stochastic resolver with $\mathcal{L}(\mathcal{P}_{\geq\lambda}) = \mathcal{L}(\mathcal{A})$. An automaton is *positively memoryless stochastically resolvable* (or simply positively resolvable) if it is λ-resolvable for some $\lambda > 0$. By varying the threshold λ, stochastic resolvability defines a spectrum where 0-resolvable corresponds to unrestricted nondeterminism and, on finite words, 1-resolvable coincides with history-determinism. See Figure 1 for distinguishing examples.

Related Work. The notion of history-determinism was introduced independently, with slightly different definitions, by Henzinger and Piterman [24] for solving games without determinisation, by Colcombet [14] for cost-functions, and by Kupferman, Safra, and Vardi [28] for recognising derived tree languages of word automata. For ω-regular automata, these variations all coincide [8]. For coBüchi-recognisable languages, history-deterministic automata can be exponentially more succinct than any equivalent deterministic ones [27]. Checking whether a given automaton is HD is decidable in polynomial time for Büchi and coBüchi automata [3, 27] and more generally also for parity automata of any fixed parity index [31]. History-determinism has been studied for richer automata models, such as pushdown automata [32, 20] timed automata [12, 11, 23] and quantitative automata [9, 10].

Recently, and independently from us, Henzinger, Prakash and Thejaswini [25] introduced classes of *stochastically resolvable* automata. The main difference of their work compared to ours is that they study the qualitative setting, where resolvers are required to succeed *almost-surely*, with probability one. In the terminology introduced above these are 1-resolvable automata. They study such automata on infinite words and compare their expressiveness, relative succinctness, against history-deterministic and semantically deterministic automata. Over finite words, these notions coincide: an NFA is 1-stochastically resolvable iff it is 1-memoryless stochastically resolvable iff it is semantically deterministic iff it is history-deterministic. They also consider the complexity of determining whether an automaton is 1-resolvable and establish that this is in polynomial time for safety automata, and PSPACE-complete for reachability and weak automata, and remains open for more general classes.

For a fixed probabilistic automaton, deciding whether a given λ is a lower bound corresponds to the undecidable emptiness problem [37], while asking if such a λ exists relates to the undecidable zero-isolation problem [19]. These problems become more tractable under restrictions such as bounded ambiguity [15, 16, 17, 18]. In the unary case, checking if λ is a lower bound is equivalent to the long-standing open positivity problem for linear recurrence sequences [36] and is also hard for Markov chains [45]. In contrast, our setting allows the probabilistic automaton to vary, asking whether some choice of probabilities satisfies the desired properties.

The unary case can be seen as a synthesis problem on parametric Markov chains. Consider the distribution transformer perspective [2, 5], where a Markov chain induces a sequence of distributions over states. Then the λ-resolvability problem asks, for universal NFA, whether there exists a parameter assignment such that the sequence of distributions is "globally" in the semi-algebraic set $S = \{x \in [0,1]^Q \mid \sum_{q\in Q} x_q = 1 \text{ and } \sum_{q\in F} x_q \geq \lambda\}$ (distributions with probability mass at least λ in accepting states). When the NFA is non-universal the "globally" condition must be adjusted to the eventually periodic pattern of the NFA. Existing work characterises, up to a set of measure zero, parameters that satisfy prefix-independent properties or "eventually" properties such as Finally Globally in S [5]. For general parametric questions, there is always the special case where the parameters are redundant, which encodes the aforementioned positivity problem [45]. This special case does not necessarily arise in our setting, as the problem has *all* transitions parameterised independently (only the structure is fixed), which leaves hope for λ-resolvability.

Our Contributions. We focus on automata over finite words and consider the decision problems whether a given automaton is resolvable. We distinguish the two variants of this problem, asking whether the automaton is positively resolvable, or whether it is λ-resolvable for a given value of λ. Our results are as follows, see also Table 1 for a summary.

- We introduce the quantitative notion of λ-memoryless stochastic resolvability, and show λ-resolvability induces a strict hierarchy of automata with varying parameter $\lambda \in (0,1)$.
- We show that checking λ-resolvability is undecidable already for NFA, on finite words, and therefore also for automata on infinite words regardless of the accepting condition.
- We complement this by showing that both positive resolvability and λ-resolvability remain decidable for finitely-ambiguous automata.
- We present complexity upper and lower bounds for several well-studied variations of finitely-ambiguous automata (summarised in Table 1). In particular, checking positive resolvability is PSPACE-complete for finitely-ambiguous automata, NL-complete for unambiguous (1-ambiguous) automata, and in the polynomial hierarchy for (unrestricted) unary automata. Checking λ-resolvability is PSPACE-hard even for k-ambiguous automata (and coNP-hard over a unary alphabet).

Table 1 Deciding positive resolvability (PR) and λ-resolvability (λR) of NFAs where $\lambda \in (0, 1)$.

		unambiguous	**finitely-ambiguous**	**general**
PR	unary	NL (Thm. 8)	coNP-hard (Thm. 11) Σ_2^P (Thm. 4)	
	non-unary	NL-complete (Thms. 9 and 8)	PSPACE-complete (Thms. 12 and 19)	open
λR	unary	PTIME (Thm. 10)	coNP-hard (Thm. 11) decidable (Thm. 20)	open
	non-unary	NL-hard (Thm. 9) PTIME (Thm. 10)	PSPACE-hard (Thm. 12) decidable (Thm. 20)	undecidable (Thm. 2)

2 Preliminaries

In this paper, we use $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Z}_+$, $\mathbb{Q}$, $\mathbb{Q}_+$, and $\mathbb{R}$ to denote the sets of non-negative integers, integers, positive integers, rational numbers, positive rational numbers, and real numbers, respectively. For $S \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ and two numbers $i, j \in S$ with $i < j$, the notation $[i, j]_S$ (resp., $(i, j)_S$) denotes the set $\{d \in S \mid i \leq d \leq j\}$ (resp., $\{d \in S \mid i < d < j\}$). The half-open intervals $(i, j]_S$ and $[i, j)_S$ are defined analogously. We also write $[j]_S$ to denote $[1, j]_S$. The subscript S is omitted if it is clear from the context.

Nondeterministic Finite Automata. A *nondeterministic finite automaton* (NFA) $\mathcal{A}$ consists of a finite alphabet Σ, a finite set of states Q, an initial state q_0, a set of accepting states $F \subseteq Q$, and a set of transitions $\Delta \subseteq Q \times \Sigma \times Q$. We also use $p \xrightarrow{\sigma} q$ to signify that $(p, \sigma, q) \in \Delta$. $\mathcal{A}$ is *unary* if $|\Sigma| = 1$; a *deterministic finite automaton* (DFA) if, for every $(p, \sigma) \in Q \times \Sigma$, there exists at most one state $q \in Q$ such that $p \xrightarrow{\sigma} q$; and *complete* if, for every $(p, \sigma) \in Q \times \Sigma$, there exists some $q \in Q$ such that $p \xrightarrow{\sigma} q$.

Given a word $w = \sigma_1 \ldots \sigma_n \in \Sigma^*$, a *run* π of $\mathcal{A}$ on w is a sequence of transitions $\tau_1 \ldots \tau_n$, where each transition $\tau_i = (p_i, \sigma_i, q_i) \in \Delta$ for $i \in [1, n]$, and $q_i = p_{i+1}$ for $i \in [1, n-1]$. The run π is *accepting* if $p_1 = q_0$ and $q_n \in F$. A word is *accepted* by $\mathcal{A}$ if there exists an accepting run on it. We denote by $\mathcal{L}(\mathcal{A})$ the set of words accepted by $\mathcal{A}$. Additionally, we use $\mathrm{ACC}_{\mathcal{A}}(w)$ to denote the set of all accepting runs of $\mathcal{A}$ on w. We omit the subscript $\mathcal{A}$ when it is clear from the context. An NFA $\mathcal{A}$ is k*-ambiguous* if, for every word $w \in \Sigma^*$, it holds that $|\mathrm{ACC}(w)| \leq k$. It is unambiguous when $k = 1$. Moreover, $\mathcal{A}$ is *finitely-ambiguous* if it is k-ambiguous for some $k \in \mathbb{Z}_+$. We use UFA and FNFA to refer to unambiguous and finitely-ambiguous NFA, respectively.

For $q \in Q$, let $\mathcal{A}_q$ denote the automaton $\mathcal{A}$ with q as its initial state, and for $S \subseteq \Delta$, let $\mathcal{A}_S$ denote the automaton obtained from $\mathcal{A}$ by restricting its transitions to S. For $\tau \in S$, we often say τ is nondeterministic in S to mean τ is nondeterministic in $\mathcal{A}_S$. Δ induces a transition function $\delta_{\mathcal{A}} : 2^Q \times \Sigma^* \mapsto 2^Q$, where $\delta_{\mathcal{A}}(Q', w)$ is the set of all states where runs on word w can end up in when starting from any state $q \in Q'$. Formally, for all $Q' \subseteq Q$, $\delta_{\mathcal{A}}(Q', \epsilon) = Q'$, and for all $w \in \Sigma^*$ and $\sigma \in \Sigma$, $\delta_{\mathcal{A}}(Q', w \cdot \sigma) = \{q \mid (p, \sigma, q) \in \Delta \text{ and } p \in \delta_{\mathcal{A}}(Q', w)\}$. When $Q' = \{p\}$, we often abuse notation and write $\delta_{\mathcal{A}}(p, w)$ instead of $\delta_{\mathcal{A}}(\{p\}, w)$, and we omit the subscript $\mathcal{A}$ when it is clear from the context. An NFA is *trim* if every state $q \in Q$ is reachable from the initial state, and can reach an accepting state, i.e., $q \in \delta(q_0, w)$ and $\delta(q, w') \cap F \neq \emptyset$ for some words w, w'.

Probabilistic Finite Automata. A *probabilistic finite automaton* (PFA) $\mathcal{P}$ is an extension of nondeterministic finite automaton that assigns an acceptance probability to each word. Specifically, $\mathcal{P}$ is an NFA augmented with an assignment $\Theta : \Delta \to [0, 1]_{\mathbb{R}}$, where for every $(p, \sigma) \in Q \times \Sigma$, $\sum_{\tau=(p,\sigma,q)\in\Delta} \Theta(\tau) = 1$. Accordingly, $\mathcal{P}$ is a 6-tuple of the form $(\Sigma, Q, q_0, \Delta, F, \Theta)$. Given a transition $\tau = (p, \sigma, q)$, the value $\Theta(\tau)$ represents the probability of transitioning from state p to state q upon reading the symbol σ. With a slight abuse of language, we also call (p, σ, q, d) a transition of $\mathcal{P}$, where $d = \Theta(p, \sigma, q)$. We say that $\mathcal{P}$ is *based on* the NFA $\mathcal{A} = (\Sigma, Q, q_0, \Delta, F)$ and call this its *underlying* NFA.[1]

We use $\mathcal{P}(w)$ to denote the probability that $\mathcal{P}$ accepts word w where:

$$\mathcal{P}(w) := \sum_{\pi=\tau_1\ldots\tau_n \in \mathrm{ACC}(w)} \prod_{i=1}^{n} \Theta(\tau_i).$$

For every threshold $\lambda \in [0,1]_{\mathbb{R}}$, let $\mathcal{L}(\mathcal{P}_{\geq\lambda})$ denote the set of all words accepted with probability at least λ, that is: $\mathcal{L}(\mathcal{P}_{\geq\lambda}) := \{w \in \Sigma^* \mid \mathcal{P}(w) \geq \lambda\}$.

We also use $\mathcal{P}_{\geq\lambda}$ to refer to the PFA $\mathcal{P}$ with a given threshold λ. Similarly, we define $\mathcal{P}_{\leq\lambda}$, $\mathcal{P}_{>\lambda}$, and $\mathcal{P}_{<\lambda}$. A PFA $\mathcal{P}$ is called *simple* if, for every transition (p, a, q, d) of $\mathcal{P}$, the probability d belongs to $\{0, \frac{1}{2}, 1\}$. Since transitions with $d = 0$ do not affect the language accepted by a PFA, we assume $d \neq 0$ for all transitions in the remainder of this paper.

Stochastic Resolvers. Given an NFA $\mathcal{A}$, a (memoryless) *stochastic resolver* $\mathcal{R}$ for $\mathcal{A}$ resolves nondeterminism during the run on a word on-the-fly, randomly, and positionally. Formally, $\mathcal{R} : \Delta \mapsto [0,1]$, such that $\sum_{\tau=(p,\sigma,q)\in\Delta} \mathcal{R}(\tau) = 1$ holds for every pair $(p, \sigma) \in Q \times \Sigma$. A resolver $\mathcal{R}$ for $\mathcal{A}$ turns it into a PFA $\mathcal{P}^{\mathcal{A}}_{\mathcal{R}}$. We call $\{\tau \in \Delta \mid \mathcal{R}(\tau) > 0\}$ the *support* of $\mathcal{R}$.

Given a threshold $\lambda \in [0,1]$, we say that $\mathcal{A}$ is *λ-memoryless stochastically resolvable* (or simply *λ-resolvable*) if there exists a resolver $\mathcal{R}$ for $\mathcal{A}$ such that $\mathcal{L}(\mathcal{P}'_{\geq\lambda}) = \mathcal{L}(\mathcal{A})$, where $\mathcal{P}' = \mathcal{P}^{\mathcal{A}}_{\mathcal{R}}$. An automaton $\mathcal{A}$ is *positively memoryless stochastically resolvable* (or simply *positively resolvable*) if it is λ-resolvable for some $\lambda \in (0,1]$.

3 λ-Resolvability

In this section we demonstrate the importance of the threshold λ: we first show that the choice of this threshold defines a spectrum of automata classes in terms of their resolvability. We then turn to the main negative result, that checking λ-resolvability is undecidable.

▶ **Theorem 1.** *For every $\lambda \in (0,1)_{\mathbb{Q}}$ there exists a unary NFA $\mathcal{A}_\lambda$ such that $\mathcal{A}_\lambda$ is λ-resolvable but not $(\lambda+\epsilon)$-resolvable for any $\epsilon > 0$.*

Proof. Let $\lambda = \frac{m}{n}$ where $m, n \in \mathbb{Z}_+$ and $m < n$. The NFA $\mathcal{A}_\lambda$ over unary alphabet $\{a\}$ is as follows. In the initial state q_0 upon reading the letter a, $\mathcal{A}_\lambda$ nondeterministically goes to one of n branches, each with $\binom{n}{m}$ states. Consider an arbitrary ordering of all m sized subsets of $[n]$, $S_1, \ldots, S_{\binom{n}{m}}$. For any branch $j \in [n]$ and $i \in [\binom{n}{m}]$, the i-th state on the j-th branch - $q^j_{S_i}$, is accepting iff $j \in S_i$. Refer to Figure 2 for an example with $\lambda = \frac{2}{3}$. The automaton $\mathcal{A}_\lambda$ is m-ambiguous, where each word in $\{a^i | i \in [\binom{n}{m}]\}$ has exactly m accepting runs. Moreover, $\mathcal{A}_\lambda$ is $\frac{m}{n}$-resolvable, since the uniform resolver that picks any transition at q_0 with probability $\frac{1}{n}$, accepts every word with probability exactly $\frac{m}{n}$. For any other resolver $\mathcal{R}$, consider the m branches $i_1, \ldots, i_m$, with the lowest assigned probabilities. Let ℓ be the unique index such that $S_\ell = \{i_1, \ldots, i_m\}$. Then for the word a^ℓ, we have $\mathcal{P}^{\mathcal{A}_\lambda}_{\mathcal{R}}(a^\ell) < \frac{m}{n}$. Hence, $\mathcal{A}_\lambda$ is not $(\lambda+\epsilon)$-resolvable for any $\epsilon > 0$. ◀

[1] The condition $\sum_{\tau=(p,\sigma,q)\in\Delta} \Theta(\tau) = 1$ may not hold for some pairs (p, σ) if the underlying NFA is not complete. Any NFA can be made complete by introducing a non-accepting sink state. Thus, throughout this paper, we assume that the underlying NFA of every PFA is implicitly made complete.

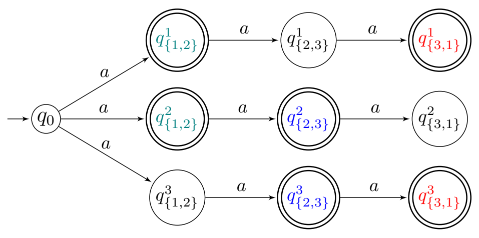


Figure 2 The automaton $\mathcal{A}_\lambda$ for $\lambda = \frac{2}{3}$, in the proof of Theorem 1.

$$\begin{tikzcd}
\underset{\text{over }\Sigma}{\mathcal{P}} \arrow[r, shift left, "\text{based on}"] & \underset{\text{over }\Sigma}{\mathcal{A}} \arrow[l, shift left, "\text{underlying}"] \\
\underset{\text{over }\Sigma'}{\mathcal{P}'} \arrow[r, "\text{unique simple PFA}", "\text{based on}"'] & \underset{\text{over }\Sigma'}{\mathcal{A}'} \arrow[u, "\text{extends}"']
\end{tikzcd}$$

Figure 3 The construction for Theorem 2.

▶ **Theorem 2.** *The λ-resolvability problem is undecidable for NFA.*

Our approach relies on a reduction from the undecidable universality problem for simple[2] PFAs [37, 19] to the λ-resolvability problem for an NFA. Consider a simple PFA $\mathcal{P}$ over the alphabet Σ, whose underlying NFA is $\mathcal{A}$. For a simple PFA, the structure of its underlying NFA uniquely determines the probabilities on its transitions: for each state q and letter σ, there are either one or two outgoing transitions from q labelled with σ; if there is one transition it must have probability 1 and if there are two, each must have probability $\frac{1}{2}$.

Intuitively, we want the resolver (if it exists) for the underlying automaton $\mathcal{A}$ of $\mathcal{P}$ to always induce a simple PFA. So, $\mathcal{A}$ is λ-resolvable if and only if $\mathcal{P}_{\geq\lambda}$ and $\mathcal{A}$ recognise the same language. In principle, a resolver for an NFA does not necessarily enforce that two outgoing transitions both have probability $\frac{1}{2}$ and may use any arbitrary split. To enforce an equal split, we construct an NFA $\mathcal{A}'$ over an extended alphabet Σ', such that $\mathcal{A}'$ is a *super-automaton* of $\mathcal{A}$. By this, we mean that the alphabet, state set, set of accepting states, and transition set of $\mathcal{A}$ are all subsets of those of $\mathcal{A}'$, and we refer to $\mathcal{A}$ as a sub-automaton of $\mathcal{A}'$. The constructed NFA $\mathcal{A}'$ is designed so that it admits only one possible $\frac{1}{4}$-resolver, and that the resolver matches the underlying probabilities of the simple PFA $\mathcal{P}$ on the part of the automaton of $\mathcal{A}'$ corresponding to $\mathcal{A}$. These relations are described in Figure 3.

Our construction has the property that $\mathcal{A}'$ is $\frac{1}{4}$-resolvable if and only if $\mathcal{P}_{\geq\frac{1}{2}}$ is universal. Essentially, the only possible resolver induces a probabilistic automaton $\mathcal{P}'$ on $\mathcal{A}'$ such that $\mathcal{P}'(w' \cdot w) = \frac{1}{2} \cdot \mathcal{P}(w)$ for all words $w \in \mathcal{L}(A)$, where w' is a fixed word over $\Sigma' \setminus \Sigma$. This ensures $\mathcal{P}(w) \geq \frac{1}{2} \iff \mathcal{P}'(w' \cdot w) \geq \frac{1}{4}$ for $w \in \mathcal{L}(A)$. The construction itself introduces new words, however these words are $\frac{1}{4}$-resolvable by construction.

Proof Sketch of Theorem 2. Without loss of generality, we fix $\Sigma = \{a, b\}$. $\mathcal{P}_{\geq\frac{1}{2}}$ is not universal if $\mathcal{A}$ is not, and since universality for NFA is decidable, we assume that $\mathcal{L}(\mathcal{A}) = \Sigma^*$. We construct a super-automaton $\mathcal{A}'$ of $\mathcal{A}$ such that it satisfies the following three properties:

C.1 For every state p of $\mathcal{A}'$ and every $\sigma \in \Sigma'$, $\mathcal{A}'$ has at most two distinct transitions of the form (p, σ, q) for some state q.

C.2 If $\mathcal{A}'$ is $\frac{1}{4}$-resolvable, then its $\frac{1}{4}$-resolvable solution must be a simple PFA.

C.3 The language $\mathcal{L}(\mathcal{A}')$ is the disjoint union of two languages, $\mathcal{L}_{\text{ext}}$ (essentially an extension of $\mathcal{L}(\mathcal{A})$) and $\mathcal{L}_{\text{ind}}$ (essentially corresponding to the new part of $\mathcal{A}'$), satisfying:
  - **(a)** For every $w \in \mathcal{L}_{\text{ind}}$, $\mathcal{P}'(w) \geq \frac{1}{4}$.
  - **(b)** For every $w'' \in \mathcal{L}_{\text{ext}}$, where $\mathcal{L}_{\text{ext}} = S \cdot \Sigma^*$ for some singleton set $S = \{w'\}$ over $\Sigma' \setminus \Sigma$ and $w'' = w' \cdot w$ with $w \in \Sigma^*$, we have $\mathcal{P}'(w'') = \frac{1}{2} \cdot \mathcal{P}(w)$.

[2] Note that [19] proves the (strict) emptiness problem for simple PFA is undecidable. Since a PFA is universal if and only if its complement is empty, the universality problem is also undecidable.

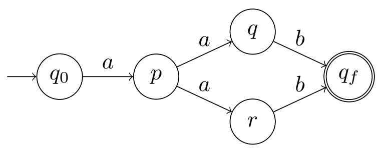


Figure 4 NFA $\mathcal{A}$.

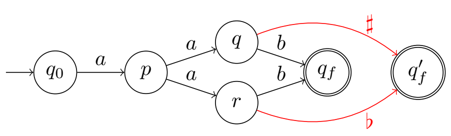


Figure 5 NFA $\mathcal{B}$.

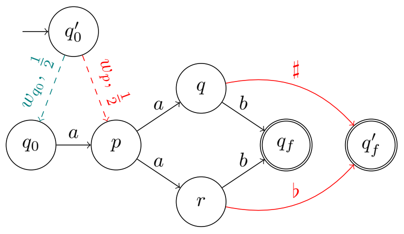


Figure 6 Partial structure of NFA $\mathcal{A}'$.

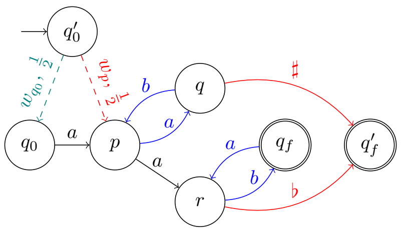


Figure 7 Additional words.

Properties C.1 and C.2 ensure that the simple PFA $\mathcal{P}'$ based on $\mathcal{A}'$ is the unique $\frac{1}{4}$-resolvable solution to $\mathcal{A}'$ if it is $\frac{1}{4}$-resolvable. Notice that regardless of whether $\mathcal{A}'$ has a $\frac{1}{4}$-resolvable solution, $\mathcal{P}'$ is based on $\mathcal{A}'$, and for each word $w \in \mathcal{L}(\mathcal{A}')$, $\mathcal{P}'$ returns its acceptance probability, which may be less than $\frac{1}{4}$. Property C.3 states that $\mathcal{P}'$ is a $\frac{1}{4}$-resolvable solution to $\mathcal{A}'$ if and only if $\mathcal{P}_{\geq \frac{1}{2}}$ is universal. Accordingly, we have:

▶ **Lemma 3.** *Let $\mathcal{A}$ be the underlying NFA of a given simple PFA $\mathcal{P}$, and let $\mathcal{A}'$ be the super-automaton of $\mathcal{A}$ satisfying properties C.1–C.3. Then $\mathcal{P}_{\geq \frac{1}{2}}$ is universal iff $\mathcal{A}'$ is $\frac{1}{4}$-resolvable.*

By Lemma 3, we can prove the undecidability of λ-resolvability by constructing, for any simple PFA $\mathcal{P}$ with underlying NFA $\mathcal{A}$, an NFA $\mathcal{A}'$ satisfying C.1–C.3. In the remainder of this section, we show how to do this and give intuitions for why these properties hold for the constructed automata. Property C.1 will hold trivially by construction, so we will focus on C.2 and C.3.

The automaton $\mathcal{A}'$ will consist of two sub-automata $\mathcal{A}_{\text{ind}}$ and $\mathcal{A}_{\text{ext}}$. The first one, $\mathcal{A}_{\text{ind}}$ recognises the language $\mathcal{L}_{\text{ind}}$, which does not depend on the structure of the original NFA $\mathcal{A}$, but rather on the numbers of $\mathcal{A}$'s states and transitions. In contrast, $\mathcal{A}_{\text{ext}}$ is an extension (super-automaton) of $\mathcal{A}$ corresponding to the language $\mathcal{L}_{\text{ext}}$.

Consider the NFA $\mathcal{A}$ given in Figure 4, where $\mathcal{L}(A) = \{aab\}$. This NFA is $\frac{1}{2}$-resolvable. Moreover, every PFA based on $\mathcal{A}$ is a $\frac{1}{2}$-resolvable solution, which is neither unique nor necessarily simple. However, if we add two transitions to $\mathcal{A}$ and derive the NFA $\mathcal{B}$ as shown in Figure 5, where for every word $w \in \{aa\sharp, aa\flat\} \subsetneq \mathcal{L}(B)$, the accepting run on w is unique, then the $\frac{1}{2}$-resolvable solution PFA to $\mathcal{B}$ becomes both unique and simple. Note that we not only introduce new transitions but also add a newly accepting state q'_f to ensure that no additional words over $\{a, b, \sharp, \flat\}$ are introduced in a more general case. This example illustrates how property C.2 can be enforced.

More specifically, referring to Figure 6, if $\mathcal{A}$ has nondeterministic transitions to states q and r from p on reading a, we introduce two new transitions, $(q, \sharp, q'_f)$ and $(r, \flat, q'_f)$, in $\mathcal{A}'$, where $\sharp$ and $\flat$ are newly introduced symbols corresponding to the original transitions (p, a, q) and (p, a, r), respectively. If we can ensure that there exists a unique word w_p corresponding to state p, such that there is a run from the initial state q'_0 of $\mathcal{A}'$ to p with probability exactly $\frac{1}{2}$, and that the accepting runs on the words $w_p \cdot a \cdot \sharp$ and $w_p \cdot a \cdot \flat$ are unique, then we can guarantee that if the PFA $\mathcal{P}'$ is a $\frac{1}{4}$-resolvable solution to $\mathcal{A}'$, then the transition probabilities for both (p, a, q) and (p, a, r) in $\mathcal{P}'$ must be $\frac{1}{2}$. The details of the run on the word w_p are shown in Figure 8. For words of the form $\$_0\$_i\$_j$ where $i = 1, 2$ and $j = 3, \ldots, 6$, the NFA in this figure has a unique accepting run. Analogous to the NFA $\mathcal{B}$, the symbols $\$_0, \ldots, \$_6$, p, and q_0 are newly introduced and are not in Σ. Every PFA based on this automaton is a $\frac{1}{4}$-resolvable solution if the transitions labelled with $\$_0$, $\$_1$, and $\$_2$ have probability exactly $\frac{1}{2}$. For each state p, we define $w_p = \$_0 p$, treating p as a symbol. Consequently, the run of $\mathcal{P}'$ from q_0' to p on w_p has probability exactly $\frac{1}{2}$. We apply this technique to every nondeterministic transition in $\mathcal{A}$. Note that the automaton shown in Figure 8 is a part of $\mathcal{A}'$, is disjoint from $\mathcal{A}$, and connects to $\mathcal{A}$ through transitions of the form (y, p, p) for $p \in Q$. Hence, if $\mathcal{A}'$ has a $\frac{1}{4}$-resolvable solution $\mathcal{P}'$, then for all transitions inherited from $\mathcal{A}$, their probability values in $\mathcal{P}'$ must be $\frac{1}{2}$. Accordingly, this construction ensures that property C.2 holds.

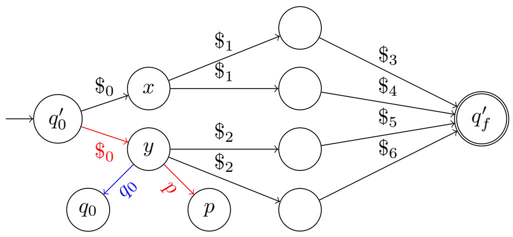


Figure 8 Details of the runs on words $w_p = \$_0 p$ and $w_{q_0} = \$_0 q_0$.

Apart from the states of $\mathcal{A}$ that have outgoing nondeterministic transitions, we also introduce a transition (y, q_0, q_0) to $\mathcal{A}'$, regardless of whether q_0 itself has nondeterministic transitions. See Figure 6 and Figure 8. Consequently, the probability of the run of $\mathcal{P}'$ from q_0' to q_0 on the word $w_{q_0} = \$_0 q_0$ is exactly $\frac{1}{2}$. Moreover, for every word $w \in \Sigma^*$, we have $w_{q_0} \cdot w \in \mathcal{L}(\mathcal{A}')$ if and only if $w \in \mathcal{L}(A)$. Hence, property C.3b follows.

It is important to explain why we consider the $\frac{1}{4}$-resolvable problem of $\mathcal{A}'$ instead of the $\frac{1}{2}$-resolvable one, as in the previous example with the NFA $\mathcal{B}$. Refer to Figure 7 for a modified example. Our intention is to force the $\frac{1}{2}$ probability split using words of the form $w_p \cdot w \cdot \sigma'$, where $|w| = 1$ and $\sigma' \in \{\sharp, \flat\}$, however the construction introduces other words that need to be managed. For example, the words $w_p \cdot a \cdot (ab)^i \cdot \sharp$ and $w_p \cdot a \cdot (ab)^i \cdot \flat$ belong to $\mathcal{L}(\mathcal{A}')$ for all $i \in \mathbb{Z}_+$. However, for the simple PFA $\mathcal{P}'$ based on $\mathcal{A}'$, we have $\mathcal{P}'(w_p \cdot a \cdot (ab)^i \cdot \sharp) = \frac{1}{2^{i+1}} < \frac{1}{2}$ and $\mathcal{P}'(w_p \cdot a \cdot (ab)^i \cdot \flat) \geq \frac{1}{2}$ for all $i \in \mathbb{Z}_+$. Even if we set the probability of the run of $\mathcal{P}'$ from the state q_0' to p on w_p to be 1, we still have $\mathcal{P}'(w_p \cdot a \cdot (ab)^i \cdot \sharp) \to 0$ as $i \to \infty$. That is, when we introduce the transitions $(q, \sharp, q_f')$ and $(r, \flat, q_f')$, we may also introduce additional words into $\mathcal{L}(\mathcal{A}')$. However, based on our tentative construction of $\mathcal{A}'$, it is possible that $\mathcal{L}(\mathcal{P}'_{\geq \frac{1}{2}}) \subsetneq \mathcal{L}(\mathcal{A}')$, regardless of whether $\mathcal{P}$ is universal or not. The words in $\mathcal{L}(\mathcal{A}') \setminus \mathcal{L}(\mathcal{P}'_{\geq \frac{1}{2}})$ are of the form $\$_0 \cdot p \cdot w \cdot \sigma'$, where $p \in Q$, $w \in \Sigma^*$ with $|w| \neq 1$, and σ' is a symbol introduced for a nondeterministic transition, such as $\sharp$ or $\flat$, etc.

To address this issue, we ensure the probability is at least $\frac{1}{4}$ probability when $|w| \neq 1$, we construct a DFA $\mathcal{A}_{\text{extra}}$ recognising the language $\{\$_0\} \cdot Q \cdot (\Sigma^* \setminus \Sigma) \cdot \{\sharp, \flat, \ldots\}$ and add it to $\mathcal{A}'$. The automaton $\mathcal{A}_{\text{extra}}$ includes a unique transition from the initial state q_0' to a state x upon reading symbol $\$_0$. This transition $(q_0', \$_0, x)$ is the same as the one shown in Figure 8, with the rest of $\mathcal{A}_{\text{extra}}$ continuing from x. For every simple PFA based on the automaton given in Figure 8, the probability value of the transition $(q_0', \$_0, x)$ is $\frac{1}{2}$. Thus, for every additional word $w \notin \mathcal{L}_{\text{ext}}$, we have $\mathcal{P}'(w) \geq \frac{1}{4}$, ensuring that property C.3a holds. Hence, the constructed $\mathcal{A}'$ satisfies properties C.1–C.3.

From Lemma 3, it follows that the λ-resolvability problem for NFA is undecidable. ◀

4 Deciding Resolvability for Unary Finite Automata

▶ **Theorem 4.** *Positive resolvability is in* $\mathsf{NP}^{\mathsf{coNP}}$ *for unary NFA.*

For a unary NFA a memoryless stochastic resolver induces a unary probabilistic automaton, or equivalently a Markov chain. This is given by an initial distribution as a row vector $I \in [0,1]^d$, a stochastic matrix $P \in [0,1]^{d\times d}$ and a final distribution as a column vector $F \in [0,1]^d$ such that $\mathcal{P}(a^n) = IP^nF$. We make use of the following lemma, which summarises standard results in Markov chain theory to our needs (see e.g. [30, Sec 1.2-1.5] or [34, Sec 1.8]). The lemma shows that there are a sequence of limit distributions that are visited periodically. Essentially, we get, for each word length n, a set of reachable states, and except for a possible initial sequence, these sets of states repeat periodically. Once in the periodical part, whenever there is a reachable accepting state, there must be some reachable accepting state in a bottom strongly connected component (SCC) to ensure the probability is bounded away from zero. The key point of the following lemma is to translate this into the support of the limit distributions for these states to be non-zero / zero for reachable states in- / outside of bottom SCCs, so that we only need to consider the support graph of the Markov chain, not the probabilities themselves. This allows us to guess the support of the resolver in NP and verify in coNP that any resolver with this support would suffice.

▶ **Lemma 5.** *Let P be a d-dimensional Markov chain with initial distribution I. There exists T, bounded by an exponential in d, and limit distributions $\pi^{(0)}, \dots, \pi^{(T-1)} \in [0,1]^d$, such that $\lim_{n\to\infty} IP^k(P^T)^n = \pi^{(k)}$. Furthermore, $\pi^{(k)}(q)$ is non-zero if and only if*

1. *q is in a bottom SCC, and*
2. *q is reachable in P from some state in I by some path of length $k + T\ell$ for some $\ell \le d$.*

We now show how to decide whether there is a stochastic memoryless resolver.

Proof of Theorem 4. Let $\mathcal{A} = (\{a\}, Q, q_0, \Delta, F)$ be a unary NFA over the alphabet $\{a\}$, with $|Q| = d$, initial state q_0 encoded in a row vector $I \in \{0,1\}^d$ and final states encoded in a column vector $F \in \{0,1\}^d$. We ask if there is a $P \in [0,1]^{d\times d}$ with support at most Δ and λ such that $IP^nF \ge \lambda$ whenever $a^n \in \mathcal{L}(\mathcal{A})$.

If P exists, it has some support $S \subseteq \Delta$ and we guess this support of P non-deterministically. The choice of support induces a new NFA $\mathcal{A}_S \subseteq \mathcal{A}$, in which every edge will be attributed a non-zero probability and $\mathcal{L}(\mathcal{A}_S) = \{a^n \mid IP^nF > 0\}$. We verify, in coNP [43], that $\mathcal{L}(\mathcal{A}_S) = \mathcal{L}(\mathcal{A})$. If not, this is not a good support.

Henceforth, we assume $\mathcal{L}(\mathcal{A}_S) = \mathcal{L}(\mathcal{A})$ and consider the condition to check whether the support S is good. For any P with support S we have $IP^nF > 0$ for $a^n \in \mathcal{L}(\mathcal{A}_S)$. It remains to decide whether this probability can be bounded away from zero, that is for some $\lambda > 0$, $IP^nF > \lambda$ whenever $a^n \in \mathcal{L}(\mathcal{A}_S)$. The condition we describe is independent of the exact choice of P and is based only on the structure of $\mathcal{A}_S$. Using T from Lemma 5 we decompose $\mathcal{L}(\mathcal{A}_S)$ into T phases and define for all $k \in \{0, \dots, T-1\}$ the set $\mathcal{L}_k = \{a^{k+Tn} \mid n \in \mathbb{N}\} \cap \mathcal{L}(\mathcal{A})$. Each $\mathcal{L}_k$ is either finite, in which case it turns out there is nothing to do, or eventually periodic with period T, in which case we use Lemma 5's characterisation of non-zero limit:

$\mathcal{L}_k$ is finite. There is nothing to decide: any P with support S induces a lower bound $\min\{IP^k(P^T)^nF \mid n \in \mathbb{N}, a^{k+Tn} \in \mathcal{L}_k\} > 0$ on the weight of the words in $\mathcal{L}_k$, as the minimum over a finite set of non-zero values.

$\mathcal{L}_k$ is eventually periodic. We check there exists a path from q_0 to some final state $q_f \in F$ of length $k + T\ell$ for some $\ell \le d$, and that q_f is in some bottom SCC in $\mathcal{A}_S$. Then for any P with support S, Lemma 5 entails a limit distribution $\pi^{(k)} \in [0,1]^d$ such that $\lim_{n\to\infty} IP^k(P^T)^n = \pi^{(k)}$, in which $\pi^{(k)}(q_f)$ is non-zero. Thus $\lim_{n\to\infty} IP^k(P^T)^nF =$

$\pi^{(k)}F = \epsilon_k > 0$. In particular, there exists n_k such that, for all $n' > n_k$, we have $IP^k(P^T)^{n'}F > \epsilon_k/2$. Hence, there is a lower bound on the weights of words in $\mathcal{L}_k$: $\min\{IP^k(P^T)^nF \mid n : a^{k+Tn} \in \mathcal{L}_k, n \leq n_k\} \cup \{\epsilon_k/2\} > 0$, which is a minimum over a finite set of non-zero values.

If there is no such path, then for any P with support S, the limit distribution is zero in all final states: the probability tends to zero no matter the exact choice of probabilities.

For a given k, it can be decided in polynomial time whether $\mathcal{L}_k$ is finite and whether there is a path of the given type. On a boolean matrix A_S representing the reachability graph of $\mathcal{A}_S$, the vector $I(A_S)^k$ and boolean matrix $(A_S)^T$ can be computed by iterated squaring in which the reachability questions can be checked.

Thus to summarise, our procedure guesses a support $S \subseteq \Delta$ and the period T in NP, and verifies in coNP both that $\mathcal{L}(\mathcal{A}_S) = \mathcal{L}(\mathcal{A})$ and guess the $k < T$ to verify the required reachability queries for k. ◀

In Theorem 11 we show that the problem is coNP-hard already in the finite-ambiguous case, we present this result alongside the non-unary case in Section 6 due to commonalities.

5 Unambiguous Finite Automata

Given an NFA $\mathcal{A}$, a support S is *bad* if there is no resolver $\mathcal{R}$ over S that positively resolves $\mathcal{A}$, that is, $\mathcal{P}^{\mathcal{A}}_{\mathcal{R}}$ is not a positively resolvable solution to $\mathcal{A}$ for each $\mathcal{R}$ over S. We start this section with a property related to bad supports for finitely-ambiguous automata, which we use to derive the decidability and complexity bounds for the resolvability problems.

▶ **Lemma 6.** *Let S be a support for an FNFA $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_S)$. For a word w with k accepting runs, $\pi_1, \ldots, \pi_k$, let $b(w)$ be the minimum number of nondeterministic transitions present in any of the π_i. Then S is a bad support if and only if there is an infinite sequence $\{w_i\}_i$ of words such that $\{b(w_i)\}_i$ is a strictly increasing sequence.*

For a UFA– where each word has exactly one accepting run – removing a transition without changing the language implies that the transition either originates from an unreachable state or leads to a state with empty language. Such transitions can thus be safely removed or assigned probability zero. Assuming the UFA is trim, its support is unique. It is not positively resolvable if and only if this support is bad.

It is easy to see that a UFA is not positively resolvable if it admits an accepting run of the form $q_0 \xrightarrow{x} q \xrightarrow{y} q \xrightarrow{z} f$, where q_0 is the initial state, f is an accepting state, and there is a nondeterministic transition taken while reading the infix y. By pumping y, we obtain a sequence of accepted words of the form xy^iz, each with a unique accepting run, but with decreasing probabilities assigned by any stochastic resolver. This follows from the fact that the loop induced by y contains a nondeterministic transition, which causes the probability mass to diminish. The UFA in Figure 1b illustrates this phenomenon: it is unambiguous but not positively resolvable. This is witnessed by the word bb, where the first b corresponds to the nondeterministic, pumpable part of the word.

In the following, we show that the existence of such a witness word is both a necessary and sufficient condition for a UFA not to be positively resolvable.

▶ **Lemma 7.** *A UFA $\mathcal{A}$ is not positively resolvable if and only if there exists a word $w = xyz \in \mathcal{L}(\mathcal{A})$ such that the accepting run on w satisfies the following conditions:*

1. *After reading the prefix x, the run reaches a state $q \in \delta_{\mathcal{A}}(q_0, x)$, and after subsequently reading y, it returns to the same state, i.e., $q \in \delta_{\mathcal{A}}(q, y)$. Moreover, reading z from q leads to an accepting state: $\delta_{\mathcal{A}}(q, z) \cap F \neq \emptyset$.*
2. *The words y and z begin with the same letter a, it follows that the transition taken when reading y from q is nondeterministic.*

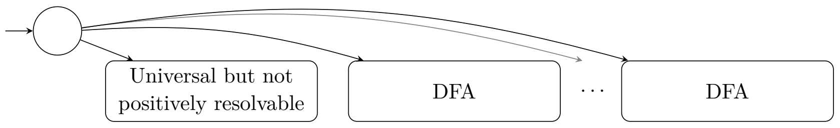


Figure 9 Conceptual idea of hardness in both unary and non-unary cases (exact details differ).

By guessing this witnessing word of Lemma 7 letter by letter, we have an NL procedure for deciding whether a UFA is positively resolvable or not.

▶ **Theorem 8.** *The positive resolvability problem for UFA is in NL.*

We can show a matching lower bound in the unambiguous case by a straightforward reduction from the emptiness problem for UFA, which is known to be NL-complete.

▶ **Theorem 9.** *Given a UFA, it is NL-hard to decide if it is positively resolvable, and NL-hard to decide if it is λ-resolvable for any fixed $\lambda > 0$.*

For a positively resolvable UFA $\mathcal{A}$, Lemma 7 essentially states that no accepting run may contain nondeterministic transitions in its looping part. This is equivalent to requiring that all transitions within SCCs of $\mathcal{A}$ are deterministic. Exploiting this structural property, we can compute the largest possible value of λ such that $\mathcal{A}$ is λ-resolvable. To do so, we collapse all SCCs into single nodes, yielding a rooted directed acyclic graph (DAG), and reduce the problem to counting the maximum number of disjoint paths in this DAG, which can be computed in polynomial time.

▶ **Theorem 10.** *It is polynomial time to decide whether a UFA is λ-resolvable.*

6 Finitely-Ambiguous Finite Automata

In this section we establish decidability and hardness for both positive resolvability and λ-resolvability for finitely-ambiguous NFA.

6.1 Hardness of Resolvability

We show that when an NFA is finitely-ambiguous, deciding the existence of a resolver, and the existence of a λ-resolver are both coNP-hard in the unary case and PSPACE-hard otherwise. While the unary and non-unary cases differ slightly, both are based on deciding whether the union of k given DFAs are universal.

The ideas of both proofs are the same, we will extend the given automaton with a new component that ensures it is universal, but the new component itself will not be positively resolvable (see Figure 9). Therefore, if the existing k-DFAs are universal then the resolver that uniformly resolves between these k-DFAs (assigning zero weight to the new component) is sufficient. However, if the k-DFAs are not themselves universal, then the new component must be used to recognise some words, but this component is not positively resolvable, and so neither is the whole automaton. The difference between the proofs will be how we ensure this works even if only a single short word is not in the union of the k-DFAs.

In the unary case we reduce from the problem of whether a unary NFA is universal which is coNP-hard [43], but similarly to [13, Theorem 7.20], we can assume that the shape is the union of simple cycles. By bringing into Chrobak normal form in polynomial time the automaton has an initial stem and a nondeterministic transition to k cycles at the end of the stem. But since universality requires that all words are accepting, each state of the stem must be accepting, which can be pre-checked in polynomial time. Thus the hard case remains to determine whether the union of the remaining k-cycle DFAs are universal.

▶ **Theorem 11.** *Given a unary k-ambiguous NFA it is* coNP*-hard to decide if it is positively resolvable, and* coNP*-hard to decide if it is $1/(k-1)$-resolvable for $1 < k \in \mathbb{Z}$.*

In the non-unary case we show PSPACE-hardness via the problem asking whether the union language of k-DFAs is universal, which is PSPACE-complete. This problem can be seen as the complement of the (well-known) problem of intersection emptiness for DFAs, that is, whether the intersection of k (not fixed) DFAs is empty, which is known to be PSPACE-complete [26]. The unary case exploits the cyclic structure to ensure that if some word is not in the union, then there are indeed infinitely many missing words. There is no such natural analogue in the non-unary case requiring a modified construction.

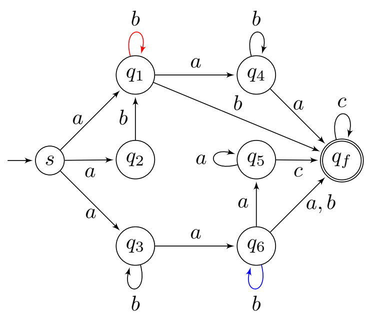
▶ **Theorem 12.** *Given a k-ambiguous NFA, it is* PSPACE*-hard to decide if it is positively resolvable, and* PSPACE*-hard to decide if it is $1/(k-2)$-resolvable for $2 < k \in \mathbb{Z}$.*

6.2 Deciding Positive Resolvability for Finitely-Ambiguous NFA

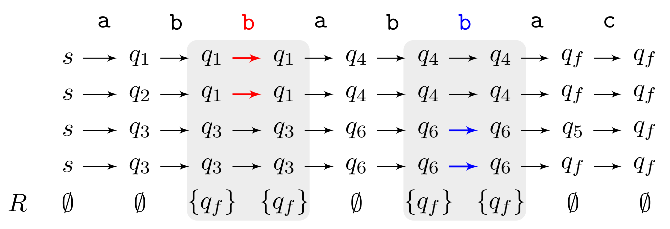
In Section 5, we established positive resolvability for unambiguous automata by identifying a witnessing "bad" word. In this section, we generalise the notion of a "bad" word to finitely-ambiguous NFA. For any UFA $\mathcal{A}$, there exists a unique support S such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_S)$ assuming $\mathcal{A}$ is trim. However, when $\mathcal{A}$ is no longer unambiguous, there may be multiple such supports S satisfying $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_S)$. If $\mathcal{A}$ is not positively resolvable, then all of its supports are *bad*, in the sense that no stochastic resolver over any of these supports can positively resolve $\mathcal{A}$. Our algorithm for deciding positive resolvability works by checking if supports of $\mathcal{A}$ are bad. We characterise a bad support S, using a witness *bad* word that satisfies several conditions. This is a generalisation of the witnessing word in the UFA case. Intuitively, a bad word for a support is a word, which contain subwords, that can be pumped arbitrarily to produce new words with acceptance probability arbitrarily close to 0. These "pumpable" subwords must preserve some reachability behaviour as well as contain probabilistic choices with probability in $(0, 1)$.

▶ **Definition 13** (Bad word). *Let $\mathcal{A}$ be an FNFA and $S \subseteq \Delta$ be a support. Then a word $w \in \Sigma^*$ with M accepting runs is a bad word for S if for some $\ell \leq M$, there exists a decomposition $w = x_0 y_1 x_1 \dots y_\ell x_\ell$ with words $x_0, \dots, x_\ell \in \Sigma^*$, $y_1, \dots, y_\ell \in \Sigma^+$, sets of states $R_1, \dots, R_\ell$, $Q_1, \dots, Q_\ell \subseteq Q$, and states $q_1, \dots, q_\ell$ with $q_i \in Q_i$ for each $i \in [\ell]$ such that:*

1. *For each $i \in [\ell]$, all accepting runs of w in $\mathcal{A}_S$ from q_0 end in Q_i after reading $x_0 y_1 \dots x_{i-1}$, i.e., $q \in \delta_{\mathcal{A}_S}(q_0, x_0 y_1 \dots x_{i-1})$ and $\delta_{\mathcal{A}_S}(q, y_i \dots x_\ell) \cap F \neq \emptyset \iff q \in Q_i$.*
2. *An accepting run of w in $\mathcal{A}_S$ must be in some q_i for some $i \in [\ell]$ after reading $x_0 y_1 x_1 \dots y_i$.*
3. *For each $i \in [\ell]$ and each $q \in Q_i$, every accepting run of w that reaches q after reading the prefix $x_0 y_1 \dots x_{i-1}$ also returns to q after subsequently reading y_i. Moreover, when $q = q_i$, this run of y_i from q_i to q_i contains at least one transition from S which is nondeterministic in $\mathcal{A}_S$.*
4. *For each $i \in [\ell]$, R_i contains all states that are reached from q_0 after reading $x_0 y_1 \dots x_{i-1}$, but has no continuation to an accepting run in $\mathcal{A}_S$ for the remaining suffix $y_i x_i \dots x_\ell$, i,e., $q \in \delta_{\mathcal{A}_S}(q_0, x_0 y_1 \dots x_{i-1})$ and $\delta_{\mathcal{A}_S}(q, y_i \dots x_\ell) \cap F = \emptyset \iff q \in R_i$. Moreover, R_i is also exactly the set of states that are reached from q_0 after reading $x_0 y_1 \dots x_{i-1} y_i$, with no continuation to an accepting run in $\mathcal{A}_S$ for the remaining suffix $x_i \dots x_\ell$, i.e., $q \in \delta_{\mathcal{A}_S}(q_0, x_0 y_1 \dots x_{i-1} y_i)$ and $\delta_{\mathcal{A}_S}(q, x_i \dots x_\ell) \cap F = \emptyset \iff q \in R_i$.*



(a) Non-positively resolvable FNFA.



a b b a b b a c
$s \to q_1 \to q_1 \to q_1 \to q_4 \to q_4 \to q_4 \to q_f \to q_f$
$s \to q_2 \to q_1 \to q_1 \to q_4 \to q_4 \to q_4 \to q_f \to q_f$
$s \to q_3 \to q_3 \to q_3 \to q_6 \to q_6 \to q_6 \to q_5 \to q_f$
$s \to q_3 \to q_3 \to q_3 \to q_6 \to q_6 \to q_6 \to q_f \to q_f$
R $\emptyset$ $\emptyset$ $\{q_f\}$ $\{q_f\}$ $\emptyset$ $\{q_f\}$ $\{q_f\}$ $\emptyset$ $\emptyset$

(b) Four accepting runs on w.

Figure 10 A bad word $w = x_0y_1x_1y_2x_2 =$ abbabbac, where $x_0 =$ ab, $y_1 =$ b, $x_1 =$ ab, $y_2 =$ b, and $x_2 =$ ac. The set R includes all reachable states from which no accepting run exists for the remaining suffix.

▶ **Example 14.** The NFA $\mathcal{A}$ in Figure 10a is 4-ambiguous. Consider the full support S containing all transitions of $\mathcal{A}$. The word $w =$ abbabbac serves as a bad word witness for S, with decomposition $w = x_0y_1x_1y_2x_2$ where $\ell = 2$, $x_0 =$ ab, $y_1 =$ b, $x_1 =$ ab, $y_2 =$ b, and $x_2 =$ ac. There are four distinct accepting runs on w; see Figure 10b for an illustration. Based on Definition 13, for w, we have $Q_1 = \{q_1, q_3\}$, $Q_2 = \{q_4, q_6\}$ and $R_1 = R_2 = \{q_f\}$. Among the four accepting runs, two take nondeterministic transitions from S when reading y_1, and the other two do so when reading y_2. Runs from Q_i to Q_i are highlighted and nondeterministic transitions are coloured in the figure. This shows that w is a bad word for S: the entire sequence of words $\{x_0y_1^ix_1y_2^ix_2\}_{i\geq 1}$ admits four accepting runs each, and the probability assigned to each such word diminishes as i increases under any stochastic resolver over S. In fact, this automaton is not positively resolvable. Although there exists another support which still preserves the language of $\mathcal{A}$ – the one that excludes the transitions going through q_2, we can show that this support is also bad, witnessed by the same bad word. With this support, we no longer have the second accepting run in Figure 10b. ⌟

Detecting bad words involves finding a decomposition as well as suitable Q_i's, q_i's and R_i's. In order to explicitly track these objects for some word, we consider an automata $\Gamma_{\mathcal{A}}$, called the *run automaton* of $\mathcal{A}$. Intuitively, a run of a word on $\Gamma_{\mathcal{A}}$, records all the states reached in accepting runs of w as well as the reachable states from which there are no continuation to accepting runs.

Run automata. Let $\mathcal{A}$ be k-ambiguous. Then the run automaton $\Gamma_{\mathcal{A}}$ has state space $Q^k \times 2^Q$, and transitions $((q_1, \ldots, q_k), R) \xrightarrow{a} ((q'_1, \ldots, q'_k), R')$ iff $q_i \xrightarrow{a} q'_i$ in $\mathcal{A}$ and $\forall q \in R, \delta(q, a) \subseteq R'$. Also, for every state $(\mathbf{q}, R)$ in $\Gamma_{\mathcal{A}}$, the set of states in the tuple $\mathbf{q}$ and the set R are disjoint. Every word $w = a_1 \ldots a_t \in \mathcal{L}(\mathcal{A})$ with k accepting runs has a run on $\Gamma_{\mathcal{A}}$ from initial state $((q_0)^k, \emptyset)$ to $(\mathbf{q}, Q')$ where $\mathbf{q} \in F^k$ and $Q' \cap F = \emptyset$ such that 1) in the ith step of the run, it reaches $((q_1^i, \ldots, q_k^i), R_i)$ if the jth accepting run of w is in q_j^i after reading $a_1 \ldots a_i$ and 2) $q \in R_i$ iff there is no accepting run of $a_{i+1} \ldots a_t$ from the reachable state q. We call such a run, a *nice run* of w in $\Gamma_{\mathcal{A}}$. For a word in $\mathcal{L}(\mathcal{A})$ with strictly fewer than k accepting runs will have such a run on $\Gamma_{\mathcal{A}}$, where for some $i_1 \neq i_2$, $q_j^{i_1} = q_j^{i_2}$ for all j, i.e. there will be at least two copies of same run. Every word w has a unique nice run on $\Gamma_{\mathcal{A}}$ up to duplication and shuffling of accepting runs. A bad word for S is essentially a word in $\mathcal{L}(A_S)$ which has a nice run on $\Gamma_{\mathcal{A}_S}$ such that in this nice run, the component of the run on the subword y_i is a self-loop in $\Gamma_{\mathcal{A}_S}$ with several accepting runs containing a nondeterministic transition from S. In Example 14 these self loops happen at states $((q_1, q_1, q_3, q_3), \{q_f\})$ and $((q_4, q_4, q_6, q_6), \{q_f\})$ in the nice run of the bad word in the run automata.

The following lemma ensures that for the loop words y_i in Definition 13, there is always a unique loop from any state $q \in Q_i$ back to itself. Assume there are two loops from q to q over the word y. Let x be a word that there is a run from q_0 to q and z a word that there is run from q to a final state. Then, the sequence of words $w_i = xy^iz$ admits an unbounded number of accepting runs, contradicting that the automaton is finitely-ambiguous.

▶ **Lemma 15.** *In an FNFA, for any accepting run containing a loop from q to q over the word y, it is the unique loop from q to q over y.*

A nice run for a bad word contains a cycle corresponding to each subword y_i. The following lemma ensures that we can construct new words by pumping these subwords while preserving the number of accepting runs.

▶ **Lemma 16.** *Let $w = xyz$ be a word in $\mathcal{L}(\mathcal{A}_S)$ with M accepting runs, whose nice run in $\Gamma_{\mathcal{A}_S}$ has a cycle on the subword y. Then for each $j \in \mathbb{N}$, xy^jz also has M accepting runs.*

▶ **Corollary 17.** *Let $w = x_0y_1x_1 \ldots x_\ell$ be a bad word for support S with M accepting runs. Then for each $(j_1, \ldots, j_\ell) \in \mathbb{N}^\ell$, $x_0y_1^{j_1}x_1y_2^{j_2} \ldots y_\ell^{j_\ell}x_\ell$ also has M accepting runs.*

For FNFA, a support S is bad when either $\mathcal{L}(\mathcal{A}_S) \subsetneq \mathcal{L}(\mathcal{A})$ or there is a bad word to witness that the support is bad. The following is a key lemma of this section, demonstrating that the existence of such a word is necessary and sufficient for a support to be bad.

▶ **Lemma 18** (Bad support-bad word). *Let $\mathcal{A}$ be an FNFA, and let $S \subseteq \Delta$ be a support. Then, S is a bad support if and only if $\mathcal{L}(\mathcal{A}_S) \subsetneq \mathcal{L}(\mathcal{A})$ or there exists a bad word for S.*

Proof Sketch. One direction is straightforward: If there exists a bad word as defined in Definition 13, we construct a sequence of words by simultaneously pumping all the loop words. By Corollary 17, each word in this sequence has the same number of accepting runs. However, the number of nondeterministic transitions in the accepting runs of each word increases. By Lemma 6, this implies that the support is bad.

Towards the other direction, suppose we are given a bad support. Then, by Lemma 6, there exists a sequence of words whose accepting runs contain an increasing number of nondeterministic transitions. We select a word w from this sequence such that every accepting run of w contains more nondeterministic transitions than the number of states in the run automaton. This ensures that, in any nice run over w in the run automaton, each accepting run must traverse a self-loop that includes a nondeterministic transition. Formally, for the ith accepting run, the word w can be decomposed as $x_iy_iz_i$, where a nice run reaches the state $(\mathbf{q}_i, R_i)$ after reading x_i, and returns to $(\mathbf{q}_i, R_i)$ after reading the loop word y_i. Moreover, the subrun over y_i contains at least one nondeterministic transition. Since the loop segments y_i may overlap across different accepting runs, we construct a new word w' by pumping these loop segments to ensure they are disjoint. The resulting word w' satisfies all the conditions of Definition 13, and is therefore a bad word. ◀

To decide the positive resolvability of an FNFA $\mathcal{A}$, we consider all possible supports S such that $\mathcal{L}(\mathcal{A}_S) = \mathcal{L}(\mathcal{A})$, and check whether each support is bad by analysing the run automaton $\Gamma_{\mathcal{A}_S}$. According to Lemma 18, if a support is bad, there must exist a bad word for it, and there must also be a corresponding nice run over this word in $\Gamma_{\mathcal{A}_S}$. This nice run encodes the structure of all accepting runs of $\mathcal{A}_S$ on the bad word. In particular, for each accepting run π of $\mathcal{A}_S$, the nice run in $\Gamma_{\mathcal{A}_S}$ includes a loop that contains at least one nondeterministic transition from S in its component corresponding to π. To show that a support is bad, it suffices to search for such nice runs in $\Gamma_{\mathcal{A}_S}$, which gives us a decidable procedure. A naive algorithm to check for a bad word would be to nondeterministically guess a nice run of a bad word in $\Gamma_{\mathcal{A}_S}$ from $((q_0)^k, \emptyset)$ to $(\mathbf{q}, Q')$ with $\mathbf{q} \in F^k$ and $Q' \cap F = \emptyset$. This involves guessing a letter at each step, keeping track of states reached in each run, i.e. at jth step computing the states $(q_1^j, \ldots, q_k^j)$ and guessing R_j for state $((q_1^j, \ldots, q_k^j), R_j)$ in $\Gamma_{\mathcal{A}_S}$. For the bad word decomposition, we would also need to guess the states Q_i and the state $q_i \in Q_i$ for simulating the cycle in $\Gamma_{\mathcal{A}_S}$ with nondeterministic transition in some run. We also track runs that have already seen nondeterministic transition in some cycle on $\mathcal{A}_S$. Finally, we have a bad word, if all runs have seen nondeterministic transitions in some cycle and the sets R_j's are consistent with the final guessed word.

However, given an FNFA, since its degree of ambiguity k is known to be bounded by $5^{\frac{|Q|}{2}}|Q|^{|Q|}$ [46], the size of the run automaton $\Gamma_{\mathcal{A}_S}$ can be $|Q|^k 2^{|Q|}$ in the worst case, which is doubly exponential in the size of $\mathcal{A}$. Hence the described procedure is not in PSPACE. By slightly modifying this procedure, instead of storing the state tuple $(q_1^j, \ldots, q_k^j)$ reached at every step, we store the set of states $A_j = \{q_1^j, \ldots, q_k^j\}$ along with keeping track of those states G_j in A_j, to which all accepting runs are yet to see nondeterministic transitions in some cycle in $\Gamma_{\mathcal{A}_S}$. Additionally, for the cycle part, when we guess Q_i, since from Lemma 15 we know that every state $q \in Q_i$ has unique run to itself for the subword y_i, there are at most n runs to track in this part. So we arbitrarily order the runs and track the progression of these runs in the cycle. Note that, in this part any transition doesn't change the size of the sets A_i and hence are bijections on A_i. We guess these bijections until the compositions of all the bijections is the identity map, i.e. each run has returned to the point of entering the cycle. At the end of the cycle we remove all states from G_i, whose cycles have encountered nondeterministic transitions from S. Since this procedure stores information only about sets A_j, G_j, R_j of size at most n, as well as bijections on $[n]$, the state space can be described using $O(n^{2n+4})$, giving us a PSPACE procedure for detecting bad words.

▶ **Theorem 19.** *It can be decided in* PSPACE *if an FNFA is positively resolvable. Moreover, the shortest bad word is of length at most* $O(n^{2n+4})$.

Proof. We provide a PSPACE algorithm for the complement problem, i.e. checking if the given FNFA is not positively resolvable. The first step is to go through all possible supports $S \subseteq \Delta$ that are language equivalent to $\mathcal{A}$. This can be checked in PSPACE[43] for general NFA.

Given a support S with $\mathcal{L}(\mathcal{A}_S) = \mathcal{L}(\mathcal{A})$, we check whether it is bad by nondeterministically guessing a bad word w. Specifically, we guess a word of the form $w = x_0 y_1 x_1 \ldots y_\ell x_\ell$, satisfying the conditions of Definition 13, letter by letter. During this process, we also guess the start and end points of each y_i, and simultaneously track an abstraction of the accepting runs, as well as of all non-accepting runs.

While reading a part x_i, which we call *branching*, or y_i, which we call *looping*, the abstraction intuitively preserves the following for a bad word: (1) which states are reachable on the overall word read so far, but only appear on rejecting runs in a set R, (2) which states are part of an accepting run of the overall word in a set A, and (3) which states are part of the accepting runs but have not yet been shown to be "bad", in a set G; formally, these are the states reached by an accepting run whose certificate state is q_j (2nd item in Definition 13) with $j > i$. The formal rules for these sets are: $R \cap A = \emptyset$ and $G \subseteq A$.

The update rules from sets R, A, G to R', A', G' when guessing a letter a are that all a-successors of a state in R must be in R', all a-successors of a state in A must be in $A' \uplus R'$ and at least one of them must be in A', and all states in G' must be a-successors of states in G.

When entering a looping part y_i, we remember the set R and denote as R_0. On reading y_i the abstraction additionally tracks (4) the set A of states that occurs on some accepting run at the beginning of y_i and remember as A_0(A_0 is the Q_i from Lemma 18), (5) which state $q' \in A_0$ is on the path for the runs with index i (that start and end with q_i in terms of Lemma 18), (6) whether this path through y_i has seen a non-deterministic transition using a boolean flag, and (7) a bijection $f\colon A_0 \to A$.

The update rules when guessing a letter a in the looping part additionally require that, for all states $q \in A$ resp. $q \in G$, exactly one a-successor is in A' resp. G'; q' is left unchanged, and the boolean flag is set to true if the a-transition from $f(q')$ is non-deterministic; otherwise it is left unchanged. The bijection f is updated to f' such that, if $f\colon q \mapsto p$, then $f'\colon q \mapsto p'$ such that $p' \in A'$ and p' is an a-successor of p.

Moving from the branching mode to looping mode can be done through an ε-transition, setting A_0 to A, f to the identity and R_0 to R, guessing $q_i \in G$ and setting the boolean flag to false. Moving back to branching mode happens through an ε-transition that can be taken when the boolean flag is set to true, f is the identity, A is A_0 and R is R_0. q' is removed from G in this ε-transition.

Finally, we accept the run when we reach a state with $R \cap F = \emptyset$ and $G = \emptyset \neq A \subseteq F$ in the branching mode.

The algorithm essentially reduces to a reachability problem on a graph with an exponential number of nodes. However, rather than explicitly constructing this graph, we operate on abstractions involving only a constant number of sets, variables, and functions, thereby requiring only polynomial space. The algorithm is then in NPSPACE, hence, in PSPACE, since NPSPACE=PSPACE[39].

To show the correctness of the algorithm, we prove that a bad word exists if and only if we can construct an accepting run in the graph. For a bad word $w = x_0y_1x_1 \dots y_\ell x_\ell$ for some ℓ with M accepting runs, we observe that the M accepting runs do not have a natural order among them, and that a repetition of each y_i may diminish the probability mass of several accepting runs. We give each accepting run an index i that refers to the first y_i that diminishes it. In particular, the accepting runs with index i are in q_i' after reading $x_0y_1 \dots y_i$. We simply take the terms from Definition 13 and construct an accepting run in the graph.

We can likewise construct a bad word from an accepting run on the graph as follows: starting with x_0, the words between two ϵ-transitions define words $x_0, y_1, x_1, y_2, \dots$; for the y_i we also store their respective A_0 and q' as Q_i' and q_i', respectively. It is now easy to see that this satisfies all local conditions of Definition 13, so that all we need to show is that all accepting runs are in q_i' after reading $x_0y_1 \dots y_i$ for some i. But this is guaranteed by $G = \emptyset$.

We get as a side result, the shortest bad word cannot be longer than the size of the graph described, and thus, $O\big(n^{2n+4}\big)$. This is a rough bound: each state can appear in both the domain and range of the bijection, leading to at most $2n$ possibilities. There are further 2 possibilities: if a state is in the range of the bijection, then it belongs to A, and consequently, it can also be in G; if it is not in the range of the bijection, then it does not belong to A and can either be in R (since R and A are disjoint) or not. In addition, it can be the state q' that we need to track in step 5. It may be in the set R_0 or not. ◀

6.3 Deciding λ-Resolvability for Finitely-Ambiguous NFA

Here we will provide an algorithm to decide λ-resolvability for FNFA. We do this by building a finite system of inequalities and check if it is satisfiable. Decidability follows from the decidability of first-order theory of reals [44]. Towards this, we consider *primitive words*: a word w is *primitive* for S if a nice run of w in $\Gamma_{\mathcal{A}_S}$ has no cycles.

▶ **Theorem 20.** *It is decidable whether an FNFA is λ-resolvable.*

Proof Sketch. The key idea is that for each primitive word w and corresponding nice run π in the run automaton, we can obtain a number z_w, which is a lower bound on the probability of any word whose nice run is the same as π without the self loops. Moreover, we can find a sequence of such words whose limit probability is z_w. Hence, the automaton is λ-resolvable if and only if there is a support that satisfies $z_w \geq \lambda$ for every primitive word w.

To encode this idea in a formula in the theory of the reals, we have a variable x_τ^S for each $\tau \in S$, to express the probabilities assigned by some stochastic resolver over a given support S. Let Θ_S denote the set of constraints ensuring that the variables $\{x_\tau^S\}_{\tau\in S}$ satisfy the probabilistic constraints and take values strictly within $(0, 1]$.

The set P_S of all primitive words associated with the support S can be obtained from the run automaton $\Gamma_\mathcal{A}$, along with the corresponding symbolic probability z_w. We define the formula for the support S as $\varphi_S := \bigwedge_{w\in P_S} \varphi_w^S$, where $\varphi_w^S := z_w \geq \lambda$ for each $w \in P_S$. This gives our final system of inequalities $\varphi := \exists S \subseteq \Delta \; \exists \; \{x_\tau^S\}_{\tau\in S}(\Theta_S \wedge \varphi_S)$. ◀

We conclude by noting that our analysis does not extend to NFA that are not finitely-ambiguous, as Lemma 6 fails for bad supports in this setting.

7 Discussion

We have introduced the notion of λ-resolvability as a new measure for the degree of non-determinism. We showed that checking this property is undecidable for NFA, and provided further decidability and complexity results for important special cases, in particular finitely-ambiguous automata.

This opens new fields of study, especially whether there are more meaningful classes of NFA where positive or λ-resolvability are decidable. More immediate future work includes closing the remaining complexity gaps and discovering whether or not checking positive resolvability for general NFA is decidable.

Other challenges that arise are how these results extend to more expressive languages, such as (visible) counter or pushdown languages, and how they extend to ω-regular languages. For the latter challenge, several of our results transfer elegantly. The undecidability of λ-resolvability holds even for weak ω-automata. For finitely-ambiguous case, we can just adjust our definition of bad words to *bad prefixes*: where the accepting and rejecting runs in a bad word end in final and non-final states for bad words, for bad prefixes there needs to be some infinite word that is accepted from all states the prefixes of accepting runs end in but from none of the states the prefixes of non-accepting runs end in. Checking if such a word exists can be done in PSPACE for nondeterministic parity automata (NPA). It follows that the positive resolvability problem for finitely-ambiguous NPA is PSPACE-complete. The λ-resolvability problem is also decidable for finitely-ambiguous NPA. We give details in Appendix A.

References

1 Bader Abu Radi and Orna Kupferman. Minimizing gfg transition-based automata. In *International Colloquium on Automata, Languages and Programming (ICALP)*, 2019. doi:10.4230/LIPIcs.ICALP.2019.100.

2 Rajab Aghamov, Christel Baier, Toghrul Karimov, Joris Nieuwveld, Joël Ouaknine, Jakob Piribauer, and Mihir Vahanwala. Model checking markov chains as distribution transformers. In *Principles of Verification: Cycling the Probabilistic Landscape - Essays Dedicated to Joost-Pieter Katoen on the Occasion of His 60th Birthday, Part II*, volume 15261 of *Lecture Notes in Computer Science*, pages 293–313. Springer, 2024. doi:10.1007/978-3-031-75775-4_13.

3 Marc Bagnol and Denis Kuperberg. Büchi Good-for-Games Automata Are Efficiently Recognizable. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 16:1–16:14, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.16.

4 Christel Baier, Luca de Alfaro, Vojtěch Forejt, and Marta Kwiatkowska. *Model Checking Probabilistic Systems*, pages 963–999. Springer International Publishing, 2018. doi:10.1007/978-3-319-10575-8_28.

5 Christel Baier, Florian Funke, Simon Jantsch, Toghrul Karimov, Engel Lefaucheux, Joël Ouaknine, David Purser, Markus A. Whiteland, and James Worrell. Parameter synthesis for parametric probabilistic dynamical systems and prefix-independent specifications. In *33rd International Conference on Concurrency Theory, CONCUR 2022, September 12-16, 2022, Warsaw, Poland*, volume 243 of *LIPIcs*, pages 10:1–10:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.CONCUR.2022.10.

6 Christel Baier, Holger Hermanns, and Joost-Pieter Katoen. *The 10,000 Facets of MDP Model Checking*, pages 420–451. Springer-Verlag, 2022. doi:10.1007/978-3-319-91908-9_21.

7 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.

8 Udi Boker and Karoliina Lehtinen. Good for games automata: From nondeterminism to alternation. In *International Conference on Concurrency Theory (CONCUR)*, volume 140, pages 19:1–19:16, 2019. doi:10.4230/LIPIcs.CONCUR.2019.19.

9 Udi Boker and Karoliina Lehtinen. History Determinism vs. Good for Gameness in Quantitative Automata. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 38:1–38:20, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.38.

10 Udi Boker and Karoliina Lehtinen. Token games and history-deterministic quantitative automata. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 120–139. Springer International Publishing, 2022. doi:10.1007/978-3-030-99253-8_7.

11 Sougata Bose, Thomas A. Henzinger, Karoliina Lehtinen, Sven Schewe, and Patrick Totzke. History-deterministic timed automata are not determinizable. In *International Workshop on Reachability Problems (RP)*, 2022. doi:10.1007/978-3-031-19135-0_5.

12 Sougata Bose, Thomas A. Henzinger, Karoliina Lehtinen, Sven Schewe, and Patrick Totzke. History-deterministic timed automata. *Log Meth Comput Sci*, Volume 20, Issue 4, October 2024. doi:10.46298/lmcs-20(4:1)2024.

13 Dmitry Chistikov, Stefan Kiefer, Andrzej S. Murawski, and David Purser. The Big-O Problem. *Log Meth Comput Sci*, 18(1), 2022. doi:10.46298/LMCS-18(1:40)2022.

14 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 139–150, 2009. doi:10.1007/978-3-642-02930-1_12.

15 Wojciech Czerwinski, Engel Lefaucheux, Filip Mazowiecki, David Purser, and Markus A. Whiteland. The boundedness and zero isolation problems for weighted automata over nonnegative rationals. In *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 15:1–15:13. ACM, 2022. doi:10.1145/3531130.3533336.

16 Laure Daviaud, Marcin Jurdziński, Ranko Lazić, Filip Mazowiecki, Guillermo A. Pérez, and James Worrell. When are emptiness and containment decidable for probabilistic automata? *J. Comput. Syst. Sci.*, 119:78–96, 2021. doi:10.1016/j.jcss.2021.01.006.

17 Nathanaël Fijalkow, Hugo Gimbert, Edon Kelmendi, and Youssouf Oualhadj. Deciding the value 1 problem for probabilistic leaktight automata. *Log. Methods Comput. Sci.*, 11(2), 2015. doi:10.2168/LMCS-11(2:12)2015.

18 Nathanaël Fijalkow, Cristian Riveros, and James Worrell. Probabilistic automata of bounded ambiguity. *Information and Computation*, 282:104648, 2022. Special issue on 9th International Workshop Weighted Automata: Theory and Applications (WATA 2018). doi:10.1016/j.ic.2020.104648.

19 Hugo Gimbert and Youssouf Oualhadj. Probabilistic automata on finite words: Decidable and undecidable problems. In *Automata, Languages and Programming*, pages 527–538, 2010. doi:10.1007/978-3-642-14162-1_44.

20 Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. A Bit of Nondeterminism Makes Pushdown Automata Expressive and Succinct. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 53:1–53:20, 2021. doi:10.4230/LIPIcs.MFCS.2021.53.

21 Ernst Moritz Hahn, Guangyuan Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. Lazy probabilistic model checking without determinisation. In *International Conference on Concurrency Theory (CONCUR)*, pages 354–367, 2015. doi:10.4230/LIPIcs.CONCUR.2015.354.

22 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 *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 12078, pages 306–323. Springer, 2020. doi:10.1007/978-3-030-45190-5_17.

23 Thomas A. Henzinger, Karoliina Lehtinen, and Patrick Totzke. History-deterministic timed automata. In *International Conference on Concurrency Theory (CONCUR)*, 2022. doi:10.4230/LIPIcs.CONCUR.2022.14.

24 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *Computer Science Logic (CSL)*, pages 395–410. Springer Berlin Heidelberg, 2006. doi:10.1007/11874683_26.

25 Thomas A. Henzinger, Aditya Prakash, and K. S. Thejaswini. Resolving nondeterminism with randomness, 2025.

26 Dexter Kozen. Lower bounds for natural proof systems. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 254–266, 1977. doi:10.1109/SFCS.1977.16.

27 Denis Kuperberg and Michał Skrzypczak. On determinisation of good-for-games automata. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 299–310, 2015. doi:10.1007/978-3-662-47666-6_24.

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

29 Orna Kupferman and Moshe Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Log.*, 2(3):408–429, 2001. doi:10.1145/377978.377993.

30 Gregory F. Lawler. *Introduction to Stochastic Processes.* Chapman and Hall/CRC, 2nd edition, 2010.

31 Karoliina Lehtinen and Aditya Prakash. The 2-token theorem: Recognising history-deterministic parity automata efficiently, 2025.

32 Karoliina Lehtinen and Martin Zimmermann. Good-for-games ω-pushdown automata. *Log Meth Comput Sci*, 18, 2022. doi:10.1145/3373718.3394737.

33 Yong Li, Soumyajit Paul, Sven Schewe, and Qiyi Tang. Accelerating markov chain model checking: Good-for-games meets unambiguous automata. In *Computer Aided Verification (CAV)*, 2025.

34 J.R. Norris. *Markov Chains.* Cambridge University Press, 1998.

35 Alexander Okhotin. Unambiguous finite automata over a unary alphabet. *Information and Computation*, 212:15–36, 2012. doi:10.1016/j.ic.2012.01.003.

36 Joël Ouaknine and James Worrell. Positivity problems for low-order linear recurrence sequences. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 366–379. SIAM, 2014. doi:10.1137/1.9781611973402.27.

37 Azaria Paz. *Introduction to Probabilistic Automata.* Academic Press, 2014.

38 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, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*, volume 183 of *LIPIcs*, pages 36:1–36:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.CSL.2021.36.

39 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.

40 Sven Schewe. Büchi complementation made tight. In Susanne Albers and Jean-Yves Marion, editors, *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, volume 3 of *LIPIcs*, pages 661–672. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany, 2009. doi:10.4230/LIPICS.STACS.2009.1854.

41 Sven Schewe. Minimising good-for-games automata is NP-complete. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2020. doi:10.4230/LIPIcs.FSTTCS.2020.56.

42 Erik Meineche Schmidt. Succinctness of description of context-free, regular and unambiguous languages. *Cornell University*, 1978. doi:10.5555/908560.

43 L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time (preliminary report). In *Symposium on Theory of Computing (STOC)*, pages 1–9. Association for Computing Machinery, 1973.

44 Alfred Tarski and J. C. C. McKinsey. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, dgo - digital original, 1 edition, 1951. doi:10.2307/jj.8501420.

45 Mihir Vahanwala. Skolem and positivity completeness of ergodic markov chains. *Inf. Process. Lett.*, 186:106481, 2024. doi:10.1016/J.IPL.2024.106481.

46 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.

A Application to Automata over Infinite Words

A *nondeterministic parity automaton* (NPA) $\mathcal{A}$ is a finite alphabet Σ, a finite set of states Q, an initial state q_0, a set of transitions $\Delta \subseteq Q \times \Sigma \times Q$, a priority function $\rho : Q \mapsto [d]$ for some $d \in \mathbb{N}$. An NPA accepts a set of infinite words in Σ^ω. A run π of $\mathcal{A}$ on an infinite word $w = \sigma_1\sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of transitions $\tau_1\tau_2\ldots$, where each $\tau_i = (p_i, \sigma_i, q_i)$ for $i \in \mathbb{Z}_+$. A run π naturally generates an infinite trace $\rho(\pi)$ over $[d]$, where the i-th element in this sequence is $\rho(p_i)$. Let $\mathit{inf}_\rho(\pi)$ be the numbers that occur infinitely often in $\rho(\pi)$. The run π is accepting if $p_1 = q_0$ and the maximum element in $\mathit{inf}_\rho(\pi)$ is even. A Büchi automaton is an NPA where ρ takes values in $\{1, 2\}$. A Büchi automaton can be described by providing the set of states with priority 2 as the accepting set of states. A Büchi automata is weak iff every SCC contains either only accepting states or only rejecting states. Let $\mathrm{ACC}_{\mathcal{A}}(w)$ denote the set of all accepting runs of $\mathcal{A}$ on w. The notions of k-ambiguous and finitely-ambiguous naturally extends to NPA based on the cardinality of $\mathrm{ACC}_{\mathcal{A}}(w)$.

A word $w \in \Sigma^\omega$ induces a probability measure $\mathit{Prob}^{\mathcal{P}}_w$ over the measurable subsets of Δ^ω, defined via the σ-algebra generated by basic cylinder sets in the standard way. The acceptance probability of w in $\mathcal{P}$ is then given by:

$$\mathcal{P}(w) = \mathit{Prob}^{\mathcal{P}}_w(\mathrm{ACC}(w)).$$

Definitions of positive resolvability and λ-resolvability extends naturally to NPA.

A.1 Undecidability of λ-Resolvability

We have shown that the λ-resolvability problem for automata over finite words (i.e., NFA) is undecidable. In this subsection, we extend this result to automata over infinite words.

Given an NFA $\mathcal{A} = (\Sigma, Q, q_0, F, \Delta)$, define the Büchi automaton $\mathcal{A}_\omega = (\Sigma', Q', q_0, F', \Delta')$ as follows:

- $\Sigma' = \Sigma \cup \{\sharp\}$, where $\sharp \notin \Sigma$.
- $Q' = Q \cup \{q'_f\}$, where $q'_f \notin Q$.
- $F' = \{q'_f\}$.
- $\Delta' = \Delta \cup \{(p, \sharp, q'_f) \mid p \in F \cup F'\}$.

By construction, $\mathcal{A}_\omega$ has a unique accepting state q'_f, which is also a sink state, making $\mathcal{A}_\omega$ a weak automaton. Suppose $\mathcal{P}$ is a PFA based on $\mathcal{A}$. We construct the corresponding probabilistic automaton $\mathcal{P}_\omega$ based on $\mathcal{A}_\omega$ by adding transitions of the form $(p, \sharp, q'_f, 1)$ to $\mathcal{P}_\omega$ for all $p \in F \cup F'$. This construction ensures that $\mathcal{P}(w) = \mathcal{P}_\omega(w \cdot \sharp^\omega)$ for every word $w \in \Sigma^*$. Consequently, for every $\lambda \in (0,1)$, $\mathcal{A}$ is λ-resolvable if and only if $\mathcal{A}_\omega$ is λ-resolvable. Together with the undecidability of the λ-resolvability problem for NFA (Theorem 2), we obtain the following:

▶ **Theorem 21.** *The λ-resolvability problem for weak nondeterministic Büchi automata is undecidable.*

A.2 Decidability for Finitely-ambiguous NPA

Our algorithm for FNFA also extends to finitely-ambiguous NPA using similar analysis. Since the number of accepting runs of every word is bounded by some k, Lemma 6 holds for finitely-ambiguous NPA as well.

▶ **Lemma 22.** *Let S be a support for a finitely-ambiguous NPA $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_S)$. Then S is a bad support if and only if for every $b \in \mathbb{N}$, there is at least one word $w \in \mathcal{L}(\mathcal{A}_S)$ such that w has at least b nondeterministic transitions from S in each of its accepting runs.*

Hence it is enough to track the behaviour of the NPA up to a finite prefix. Towards this goal, the notion of bad words can be adapted to NPA using *bad prefixes*. Let $\mathcal{A}^q$ denote the ω-automaton $\mathcal{A}$ with q as its initial state.

▶ **Definition 23** (Bad prefix). *Let $\mathcal{A}$ be a finitely-ambiguous NPA and $S \subseteq \Delta$ be a support. Then a word $u \in \Sigma^*$ is a bad prefix for S if for some ℓ, there exists a decomposition $u = x_0y_1x_1 \dots y_\ell$ with words $x_0, \dots, x_{\ell-1} \in \Sigma^*$, $y_1, \dots, y_\ell \in \Sigma^+$, sets of states $Q_1, \dots, Q_\ell, R_1, \dots, R_\ell \subseteq Q$, states $q_1, \dots, q_\ell$ with $q_i \in Q_i$ for each $i \in [\ell]$ and an ω-word $v \in \Sigma^\omega$ such that $uv \in \mathcal{L}(\mathcal{A}_S)$ and*

1. *For each $i \in [\ell]$, all accepting runs of uv from q_0 end in Q_i after reading $x_0y_1 \dots x_{i-1}$, i.e., $q \in \delta_{\mathcal{A}_S}(q_0, x_0y_1 \dots x_{i-1})$ and $y_i \dots y_\ell v \in \mathcal{L}(\mathcal{A}_S^q) \iff q \in Q_i$.*
2. *An accepting run of uv must be in some q_i for some $i \in [\ell]$ after reading $x_0y_1x_1 \dots y_i$.*
3. *For each $i \in [\ell]$ and each $q \in Q_i$, every accepting run of uv that reaches q after reading the prefix $x_0y_1 \dots x_{i-1}$ also returns to q after subsequently reading y_i. Moreover, when $q = q_i$, this run of y_i from q_i to q_i contains at least one transition from S which is nondeterministic in $\mathcal{A}_S$.*
4. *For each $i \in [\ell]$, R_i contains all states that are reached from q_0 after reading $x_0y_1 \dots x_{i-1}$, but has no continuation to an accepting run for the remaining suffix $y_ix_i \dots y_\ell v$, i,e., $q \in \delta_{\mathcal{A}_S}(q_0, x_0y_1 \dots x_{i-1})$ and $y_i \dots y_\ell v \notin \mathcal{L}(\mathcal{A}_S^q) \iff q \in R_i$. Moreover, R_i is also exactly*

the set of states that are reached from q_0 after reading $x_0y_1 \dots x_{i-1}y_i$, with no continuation to an accepting run for the remaining suffix $x_i \dots y_\ell v$, i.e., $q \in \delta_{\mathcal{A}_S}(q_0, x_0y_1 \dots x_{i-1}y_i)$ and $x_i \dots y_\ell v \notin \mathcal{L}(\mathcal{A}_S^q) \iff q \in R_i$.

▶ **Lemma 24** (Bad support-bad prefix)**.** *Let $\mathcal{A}$ be a finitely-ambiguous NPA and S be a support of $\mathcal{A}$. Then S is a bad support for $\mathcal{A}$ iff $\mathcal{L}(\mathcal{A}_S) \subsetneq \mathcal{L}(A)$ or there exists a bad prefix for S.*

Hence, our PSPACE algorithm for deciding the positive resolvability of FNFA can be extended to finitely-ambiguous NPA by adding an *accepting mode.* This mode verifies whether there exists an ω-word v in the set $\bigcap_{q \in A} \mathcal{L}(\mathcal{A}_S^q) \setminus \bigcup_{q \in R} \mathcal{L}(\mathcal{A}_S^q)$. In the branching mode, a transition to the accepting mode is always possible when $G = \emptyset$. Once the system enters the accepting mode, it remains there.

To check whether or not $\bigcap_{q \in A} \mathcal{L}(\mathcal{A}_S^q) \setminus \bigcup_{q \in R} \mathcal{L}(\mathcal{A}_S^q)$ is empty, we check if $\bigcap_{q \in A} \mathcal{L}(\mathcal{A}_S^q) \subseteq \bigcup_{q \in R} \mathcal{L}(\mathcal{A}_S^q)$ holds. To do this, we can simply translate the individual $\mathcal{A}_S^q$ for all $q \in A$ to NBAs $\mathcal{N}_q$ and construct an NBA $\mathcal{N}$ recognising $\bigcup_{q \in R} \mathcal{L}(\mathcal{A}_S^q)$; the blow-up for this is small.

We can now check whether or not $\bigcap_{q \in A} \mathcal{L}(\mathcal{N}_q) \subseteq \mathcal{L}(\mathcal{N})$ holds.

For this, we complement $\mathcal{N}$ using [40] to the complement NBA $\mathcal{C}$, but note that this complement – including the states and transitions – can be represented symbolically by $\mathcal{N}$ itself – it only uses subsets of the states in $\mathcal{N}$ (the reachable states and a subset thereof used for a breakpoint construction) and level rankings, which are functions from the reachable states of $\mathcal{N}$ to natural numbers up to $2n$, where n is the number of states of $\mathcal{N}$ [29, 40].

We can now check whether or not $\bigcap_{q \in A} \mathcal{L}(\mathcal{N}_q) \cap \mathcal{L}(\mathcal{C}) = \emptyset$ holds.

To do this, we can search in the exponentially larger, but polynomially represented, product space of $\bigotimes_{q \in A} \mathcal{N}_q \times \mathcal{C}$ (the product automaton of $\mathcal{N}_q$ for all $q \in A$ and $\mathcal{C}$) to check if we can reach a product state, which can loop to itself while seeing an accepting state of each of the automata. This can be checked in NL in the explicit size of the product automaton and thus in PSPACE in its representation.

▶ **Theorem 25.** *The positive resolvability problem for finitely-ambiguous NPA is* PSPACE*-complete.*

Let the degree of ambiguity of a finitely-ambiguous NPA $\mathcal{A}$ be k, which can be decided [38, Proposition 21]. Note that in [38], the notion of finite ambiguity differs from ours: an automaton is called finitely ambiguous if $|\,\mathrm{ACC}(w)|$ is finite for every word w in the language. What we refer to as finitely-ambiguous is instead termed *boundedly ambiguous* in their terminology.

Let S be a support of $\mathcal{A}$, we can construct a run automaton for $\mathcal{A}_S$, similar to the construction for an NFA on a support. As in the NFA case, by analysing the run automaton, we can build a finite system of inequalities and decide the λ-resolvability problem for finitely-ambiguous NPA.

▶ **Theorem 26.** *The λ-resolvability problem for finitely-ambiguous NPA is decidable.*