When is Containment Decidable for Probabilistic Automata?

Laure Daviaud  
University of Warwick, Coventry, UK  
L.Daviaud@warwick.ac.uk

Marcin Jurdziński  
University of Warwick, Coventry, UK  
Marcin.Jurdzinski@warwick.ac.uk

Ranko Lazić  
University of Warwick, Coventry, UK  
R.S.Lazic@warwick.ac.uk

Filip Mazowiecki  
Université de Bordeaux, Bordeaux, France  
filip.mazowiecki@u-bordeaux.fr

Guillermo A. Pérez  
Université libre de Bruxelles, Brussels, Belgium  
gperezme@ulb.ac.be  
https://orcid.org/0000-0002-1200-4952

James Worrell  
University of Oxford, Oxford, UK  
James.Worrell@cs.ox.ac.uk  
https://orcid.org/0000-0001-8151-2443

Abstract

The containment problem for quantitative automata is the natural quantitative generalisation of the classical language inclusion problem for Boolean automata. We study it for probabilistic automata, where it is known to be undecidable in general. We restrict our study to the class of probabilistic automata with bounded ambiguity. There, we show decidability (subject to Schanuel’s conjecture) when one of the automata is assumed to be unambiguous while the other one is allowed to be finitely ambiguous. Furthermore, we show that this is close to the most general decidable fragment of this problem by proving that it is already undecidable if one of the automata is allowed to be linearly ambiguous.

2012 ACM Subject Classification  Theory of computation → Quantitative automata, Theory of computation → Probabilistic computation

Keywords and phrases  Probabilistic automata, Containment, Emptiness, Ambiguity

Funding  L. Daviaud, M. Jurdziński, and R. Lazić have been supported by the EPSRC grant EP/P020992/1.

---

1 R. Lazić has been supported by a Leverhulme Trust Research Fellowship RF-2017-579.
2 F. Mazowiecki has been supported by the French National Research Agency (ANR) in the frame of the “Investments for the future” Programme IdEx Bordeaux (ANR-10-IDEX-03-02).
3 G. A. Pérez has been supported by an F.R.S.-FNRS Aspirant fellowship and an FWA postdoc fellowship.
4 J. Worrell has been supported by the EPSRC Fellowship EP/N008197/1.
Acknowledgements We thank Shaull Almagor for some helpful remarks.

1 Introduction

Probabilistic automata (PA) are a quantitative extension of classical Boolean automata that were first introduced by Rabin [19]. Non-deterministic choices are replaced by probabilities: each transition carries a rational number which gives its probability to be chosen amongst all the other transitions going out of the same state and labelled by the same letter. Then, instead of simply accepting or rejecting a word, such an automaton measures the probability of it being accepted.

PA can be seen as (blind) partially observable Markov decision processes [15]. The latter have numerous applications in the field of artificial intelligence [21, 10]. Further applications for PA include, amongst others, verification of probabilistic systems [22, 13, 5], reasoning about inexact hardware [17], quantum complexity theory [24], uncertainty in runtime modelling [9], as well as text and speech processing [16]. PA are very expressive, as witnessed by the mentioned applications, most natural verification-related decision problems for them are consequently undecidable. However, equivalence and minimisation do admit efficient algorithms [12].

Due to the aforementioned negative results, many sub-classes of probabilistic automata have been studied. These include hierarchical [7] and leaktight [2] automata; and more recently, bounded-ambiguity automata [8] (see [6] for a survey).

In this paper, we continue the study of the class of PA with bounded ambiguity. We focus on the containment problem: given two automata $A$ and $B$, determine whether for all words $w$, the probability of it being accepted by $A$ is at most the probability of it being accepted by $B$.

The problem is known to be undecidable even for the subclass of automata with polynomial ambiguity, more specifically, already for automata with quadratic ambiguity [8].

Contributions. In this paper, we refine the undecidability result by extending it to the class of linearly ambiguous automata.

\begin{itemize}
\item \textbf{Theorem 1.} The containment problem is undecidable for the class of linearly ambiguous probabilistic automata.
\end{itemize}

The proof we provide gives in fact two stronger results. Firstly, the containment problem for linearly ambiguous PA is already undecidable if one of the two input automata is unambiguous. Secondly, and perhaps more importantly, the better-known emptiness problem (given a probabilistic automaton, does there exist a word accepted with probability at least $1/2$?) is also undecidable for the class of linearly ambiguous PA. This strictly refines the previous best known result [8].

This negative result motivates us to turn our attention to the class of finitely ambiguous PA. For this class, we prove that the containment problem is decidable, provided that one of the two input automata is unambiguous (and conditional on Schanuel’s conjecture).

\begin{itemize}
\item \textbf{Theorem 2.} If Schanuel’s conjecture holds then the containment problem is decidable for the class of finitely ambiguous probabilistic automata, provided that at least one of the input automata is unambiguous.
\end{itemize}

The intermediate problem, i.e., when both input PA are finitely ambiguous, remains open.
Organisation of the paper. In Section 2, we give the formal definition of probabilistic automata, the notion of ambiguity, and the problems under consideration. We also recall classical results that will be useful in the paper. In Section 3, we explain how to translate the containment problem into a problem about the existence of integral exponents for certain exponential inequalities. Using this formalism, we prove that the containment problem for $\mathcal{A}$ and $\mathcal{B}$, as stated above, is decidable if $\mathcal{A}$ is finitely ambiguous and $\mathcal{B}$ is unambiguous. In Section 4, we tackle the more challenging direction and prove that the containment problem is also decidable if $\mathcal{A}$ is unambiguous and $\mathcal{B}$ is finitely ambiguous. Finally, in Section 5, we prove that the containment problem is undecidable provided that one of the automata is linearly ambiguous.

2 Preliminaries

In this section, we define probabilistic automata and recall some classical results.

Notation. We use boldface lower-case letters, e.g., $a, b, \ldots$, to denote vectors and uppercase letters, e.g., $M, N, \ldots$, for matrices. For a vector $a$, we write $a_i$ for its $i$-th component, and $a^\top$ for its transpose.

2.1 Probabilistic automata and ambiguity

For a finite set $S$, we say that a function $f : S \to \mathbb{Q}_{\geq 0}$ is a distribution over $S$ if $\sum_{s \in S} f(s) \leq 1$. We write $\mathcal{D}(S)$ for the set of all distributions over $S$. We also say that a vector $d = (d_1, d_2, \ldots, d_n) \in \mathbb{Q}_{\geq 0}^n$ of non-negative rationals is a distribution if $\sum_{i=1}^n d_i \leq 1$.

A probabilistic automaton (PA) $\mathcal{A}$ is a tuple $(\Sigma, Q, \delta, \iota, F)$, where:

- $\Sigma$ is the finite alphabet,
- $Q$ is the finite set of states,
- $\delta : Q \times \Sigma \to \mathcal{D}(Q)$ is the (probabilistic) transition function,
- $\iota \in \mathcal{D}(Q)$ is the initial distribution, and
- $F \subseteq Q$ is the set of final states.

We write $\delta(q, a, p)$ instead of $\delta(q, a)(p)$ for the probability of moving from $q$ to $p$ reading $a$.

Consider the word $w = a_1 \ldots a_n \in \Sigma^*$. A run $\rho$ of $\mathcal{A}$ over $w = a_1 \ldots a_n$ is a sequence of transitions $(q_0, a_1, q_1), (q_1, a_2, q_2), \ldots, (q_{n-1}, a_n, q_n)$ where $\delta(q_{i-1}, a_i, q_i) > 0$ for all $1 \leq i \leq n$. It is an accepting run if $\iota(q_0) > 0$ and $q_n \in F$. The probability of the run $\rho$ is $\Pr_\mathcal{A}(\rho) \triangleq \iota(q_0) \cdot \prod_{i=1}^n \delta(q_{i-1}, a_i, q_i)$.

The automaton $\mathcal{A}$ realizes a function $[\mathcal{A}]$ mapping words over the alphabet $\Sigma$ to values in $[0, 1]$. Formally, for all $w \in \Sigma^*$, we set: $[\mathcal{A}](w) \triangleq \sum_{\rho \in \text{Acc}_\mathcal{A}(w)} \Pr_\mathcal{A}(\rho)$ where $\text{Acc}_\mathcal{A}(w)$ is the set of all accepting runs of $\mathcal{A}$ over $w$.

Ambiguity. The notion of ambiguity depends only on the structure of the underlying automaton (i.e., whether a probability is null or not, but not on its actual value). An automaton $\mathcal{A}$ is said to be unambiguous (resp. $k$-ambiguous) if for all words $w$, there is at most one accepting run (resp. $k$ accepting runs) over $w$ in $\mathcal{A}$. If an automaton is $k$-ambiguous for some $k$, then it is said to be finitely ambiguous. If there exists a polynomial $P$, such that for every word $w$, the number of accepting runs of $\mathcal{A}$ on $w$ is bounded by $P(|w|)$ (where $|w|$ is the length of $w$), then $\mathcal{A}$ is said to be polynomially ambiguous, and linearly ambiguous whenever the degree of $P$ is at most 1.
Containment for Probabilistic Automata

It is well-known that if an automaton is not finitely ambiguous then it is at least linearly ambiguous (see, for example, the criterion in [23, Section 3]). The same paper shows that if an automaton is finitely ambiguous then it is $k$-ambiguous for $k$ bounded exponentially in the number of states of that automaton.

We give two examples of PA and discuss their ambiguity in Figure 1. As usual, they are depicted as graphs. The initial distribution is denoted by ingoing arrows associated with their probability (when there is no such arrow, the initial probability is 0) and the final states are denoted by outgoing arrows.

2.2 Decision problems

In this work, we are interested in comparing the functions computed by PA. We write $[A] \leq [B]$ if “$A$ is contained in $B$”, that is if $[A](w) \leq [B](w)$ for all $w \in \Sigma^*$; and we write $[A] < \frac{1}{2}$ if $[A](w) < \frac{1}{2}$ for all $w \in \Sigma^*$. We are interested in the following decision problems for PA.

- **Containment problem:** Given probabilistic automata $A$ and $B$, does $[A] \leq [B]$ hold?
- **Emptiness problem:** Given a probabilistic automaton $A$, does $[A] < \frac{1}{2}$ hold?

We will argue that the containment and emptiness problems are both undecidable when considered for the class of linearly ambiguous automata (Section 5). The emptiness problem is known to be decidable for the class of finitely ambiguous automata [8]. We tackle here the more difficult containment problem (Sections 3 and 4).

2.3 Classical results

**Weighted-sum automaton.** For PA $A_1, A_2, \ldots, A_n$ over the same alphabet, and for a discrete distribution $d = (d_1, d_2, \ldots, d_n)$, the **weighted sum** (of $A_1, A_2, \ldots, A_n$ with weights $d$) is defined to be the disjoint union of the $n$ automata with the initial distribution $\iota(q) \overset{\text{def}}{=} d_i \cdot \iota_i(q)$ if $q$ is a state of $A_i$, where $\iota_i$ is the initial distribution of $A_i$. Note that if $B$ is the weighted sum of $A_1, A_2, \ldots, A_n$ with weights $d$ then it is also a probabilistic automaton and $[B] = \sum_{i=1}^n d_i \cdot [A_i]$.

**Complement automaton.** For a PA $A$, we define its **complement automaton** $\overline{A}$ in the following way. First, define the PA $A'$ by modifying $A$ as follows:

- add a new sink state $q_{\perp}$;
- obtain the transition function $\delta'$ from $\delta$ by adding transitions:
  - $\delta'(q_{\perp}, a, q_{\perp}) = 1$ for all $a \in \Sigma$,
  - $\delta'(q, a, q_{\perp}) = 1 - \sum_{r \in Q} \delta(q, a, r)$ for all $(q, a) \in Q \times \Sigma$;
- obtain the initial distribution $\iota'$ from $\iota$ by adding $\iota'(q_{\perp}) = 1 - \sum_{q \in Q} \iota(q)$.
Observe that $\llbracket A' \rrbracket = \llbracket A \rrbracket$, that $\sum_{q \in Q} \delta'(q, a, r) = 1$ for all $(q, a) \in Q \times \Sigma$, and that $\sum_{q \in Q} \ell'(q) = 1$. We obtain $\overline{A}$ from $A'$ by swapping its final and non-final states. As expected, it is the case that $\llbracket \overline{A} \rrbracket = 1 - \llbracket A \rrbracket$.

\textbf{Remark (Preserving ambiguity).} The ambiguity of a weighted-sum automaton is the sum of the ambiguities of the individual automata, and the ambiguity of a complement automaton may be larger than the ambiguity of the original one (see Figure 1).

### 3 Decidability of the case finitely ambiguous vs. unambiguous

Our aim is to decide whether $\llbracket A \rrbracket \leq \llbracket B \rrbracket$. We first give a translation of the problem into a problem about the existence of integral exponents for certain exponential inequalities.

\textbf{Notation.} In the rest of the paper, we write $\exp(x)$ to denote the exponential function $x \mapsto e^x$, and $\log(y)$ for the natural logarithm function $y \mapsto \log_e(y)$. For a real number $x$ and a positive real number $y$, we write $y^x$ for $\exp(x \log(y))$.

#### 3.1 Translating the containment problem into exponential inequalities

We are going to translate the negation of the containment problem: Given two finitely ambiguous PA $A$ and $B$, does there exist a word $w$, such that $\llbracket A \rrbracket(w) > \llbracket B \rrbracket(w)$? Consider two positive integers $k$ and $n$, and vectors $p \in \mathbb{Q}_+^k$ and $q_1, \ldots, q_k \in \mathbb{Q}_+^n$. We denote by $S(p, q_1, \ldots, q_k) : \mathbb{N}^n \to \mathbb{R}$ the function associating a vector $x \in \mathbb{N}^n$ to $\sum_{i=1}^k p_i q_{i,1}^\ell q_{i,2}^\ell \cdots q_{i,n}^\ell$, where $q_{i,j}$ is the $j$-th component of vector $q_i$.

\textbf{Proposition 3.} Given a $k$-ambiguous automaton $A$ and an $\ell$-ambiguous automaton $B$, one can compute a positive integer $n$ and a finite set $\Delta$ of tuples $(p, q_1, \ldots, q_k, r, s_1, \ldots, s_{\ell'})$ of vectors $p \in \mathbb{Q}_+^k, r \in \mathbb{Q}_+^{\ell'}$ for some $k' \leq k$ and $\ell' \leq \ell$; and $q_i \in \mathbb{Q}_+^n, s_j \in \mathbb{Q}_+^n$, for all $i$ and $j$; such that the following two conditions are equivalent:

- there exists $w \in \Sigma^*$ such that $\llbracket A \rrbracket(w) > \llbracket B \rrbracket(w)$,
- there exist $(p, q_1, \ldots, q_k, r, s_1, \ldots, s_{\ell'}) \in \Delta$ and $x \in \mathbb{N}^n$ such that $S(p, q_1, \ldots, q_k)(x) > S(r, s_1, \ldots, s_{\ell'})(x)$.

It thus follows that to prove Theorem 2 it suffices to show decidability of the second item of Proposition 3 for a given element of $\Delta$ in the cases where either $k$ or $\ell$ are equal to 1. The proof of Proposition 3 is the most technical part and it is postponed to Section 3.3.

\textbf{Example 4.} Consider the following instance of the problem, where $k = n = 2$, $\ell = 1$, and $p$ is a fixed rational number $0 \leq p \leq 1$: Do there exist $x, y \in \mathbb{N}$ such that $p \cdot \left(\frac{1}{4}\right)^x \cdot \left(\frac{1}{3}\right)^y + (1 - p) \cdot \left(\frac{1}{2}\right)^x \cdot \left(\frac{1}{2}\right)^y < \left(\frac{1}{4}\right)^x \cdot \left(\frac{1}{3}\right)^y$. This can be rewritten as

$$p \cdot \left(\frac{1}{2}\right)^x \cdot 3^y + (1 - p) \cdot 2^x \cdot \left(\frac{1}{3}\right)^y < 1$$

or equivalently, using the exponential function, as follows

$$\exp(\log(p) - x \log(2) + y \log(3)) + \exp(\log(1 - p) + x \log(2) - y \log(3)) < 1.$$

Consider the set $V = \{(x, y) \in \mathbb{R}^2 \mid e^x + e^y < 1\}$ and denote by $b$ the point $(\log(p), \log(1 - p))$. Let $u = (\log(2), \log(2))$ and $v = (\log(3), -\log(3))$ be two vectors. See Figure 2 for a
We write that the first state of a run is the first state of the first transition, and the last state of a run is the last state of the last transition. A run is simple if every state appearing in the sequence of transitions composing the run, appears at most twice. A cycle is a run in which the first and the last states coincide. A simple cycle is a cycle which is a simple run.

For a state \( q \), we say that a cycle is a \( q \)-cycle if the first (and hence also the last) state is \( q \). For a run \( \rho = \rho’ \cdot \rho'' \), such that the last state of \( \rho' \) (and hence also the first state of \( \rho'' \)) is \( q \), and for a \( q \)-cycle \( \omega \), the result of injecting \( \omega \) (after \( \rho' \)) into \( \rho \) is the run \( \rho' \cdot \omega \cdot \rho'' \).
For a run \( \rho \), we write \( Q(\rho) \) for the set of states that occur in it. For a set of states \( P \), we write \( \text{Periods}(P) \) for the set of simple cycles in which only states in \( P \) occur. A **simple cycle decomposition** is a pair \((\gamma, \sigma)\), where \( \gamma \) is a run of length less than \( |Q(\gamma)|^2 \) and \( \sigma : \text{Periods}(Q(\gamma)) \to \mathbb{N} \). We say that a simple cycle decomposition \((\gamma, \sigma)\) is a simple cycle decomposition of a run \( \rho \) if the run \( \rho \) can be obtained from \( \gamma \) by injecting \( \sigma(\omega) \) cycles \( \omega \), for every simple cycle \( \omega \in \text{Periods}(Q(\gamma)) \), in some order.

**Proposition 6.** Every run has a simple cycle decomposition.

**Proof.** The above result is classic, cf. e.g. [20, proof of Lemma 4.5]. It follows by repeatedly removing from a run \( \rho \), as long as its length is at least \( |Q(\rho)|^2 \), some simple cycle whose removal does not decrease the set of states \( Q(\rho) \). To see that such a simple cycle must exist, observe that if the length of \( \rho \) is at least \( |Q(\rho)|^2 \), then it contains \( |Q(\rho)| \) non-overlapping simple cycles \( \omega_1, \ldots, \omega_{|Q(\rho)|} \); and if \( W_i \) is the set of all states that occur strictly inside \( \omega_i \) but nowhere else in \( \rho \), then the sets \( W_1, \ldots, W_{|Q(\rho)|} \) are mutually disjoint and their union has size less than \( |Q(\rho)| \), so some \( W_i \) must be empty.

If \((\gamma, \sigma)\) is a simple cycle decomposition, then we refer to \( \gamma \) as its **spine** and to \( \sigma \) as its **simple cycle count**. Observe that the number of distinct spines is finite; more specifically, it is at most exponential in the size of the automaton, as is the set of simple cycles \( \text{Periods}(Q(\gamma)) \) for every spine \( \gamma \).

We say that a simple cycle decomposition \((\gamma, \sigma)\) is **accepting** if the run \( \gamma \) is. By Proposition 6, every accepting run has an accepting simple cycle decomposition. Moreover, for every accepting spine \( \gamma \), and for every function \( \sigma : \text{Periods}(Q(\gamma)) \to \mathbb{N} \), there is at least one accepting run \( \rho \), such that \((\gamma, \sigma)\) is its simple cycle decomposition.

**Proposition 7.** There is an algorithm that given a finitely ambiguous probabilistic automaton \( \mathcal{A} \) and a nonnegative integer \( i \), outputs a finite automaton that accepts the language of words on which \( \mathcal{A} \) has exactly \( i \) accepting runs.

**Proof.** We can assume that \( \mathcal{A} \) is trimmed (i.e., all states are reachable from some initial state and can reach some final state). It is known that the number of all active runs in a trimmed finite ambiguous automaton is bounded exponentially in the number of states in \( \mathcal{A} \) [23]. We can therefore add an extra component to \( \mathcal{A} \) that, using the powerset construction, keeps track of all active runs in the automaton. Using this component, the automaton \( \mathcal{A} \) can extract the number of all accepting runs.

We are ready to prove Proposition 3.

**Proof of Proposition 3.** First, use Proposition 7 to compute finite automata \( \mathcal{A}_{k'} \), \( 0 \leq k' \leq k \), and \( \mathcal{B}_{\ell'} \), \( 0 \leq \ell' \leq \ell \), that accept the languages of words on which \( \mathcal{A} \) has exactly \( k' \) accepting runs and \( \mathcal{B} \) has exactly \( \ell' \) accepting runs, respectively.

For all \( k' \), \( 0 \leq k' \leq k \), and for all \( \ell' \), \( 0 \leq \ell' \leq \ell \), we perform the following. Consider the synchronized product of \( \mathcal{A}_{k'}, \mathcal{B}_{\ell'} \), \( k' \) copies of \( \mathcal{A} \), and \( \ell' \) copies of \( \mathcal{B} \). Moreover, equip the synchronized product with another component, a finite automaton that maintains (in its state space) the partition of the \( k' \) components corresponding to copies of \( \mathcal{A} \), and of the partition of the \( \ell' \) components corresponding to copies of \( \mathcal{B} \), that reflects which of the \( k' \) runs of \( \mathcal{A} \), and which of the \( \ell' \) runs of \( \mathcal{B} \), respectively, have been identical so far. Consider as final the states of this additional component in which all sets in both partitions are singletons. The purpose of the last component is to be able to only consider runs of the synchronized product in which the \( k' \) components corresponding to copies of \( \mathcal{A} \), and the \( \ell' \) components
corresponding to copies of $\mathcal{B}$, have all distinct runs. Similarly, the purpose of the copies of $A_{k'}$ and $B_{k'}$ is to be able to only consider runs of the synchronized product which record all the $k'$ distinct accepting runs of $A$ and all of the $\ell'$ distinct accepting runs of $B$, respectively, on the underlying words. Let $C_{k',\ell'}$ be the resulting finite automaton with $k' + \ell' + 3$ components.

The following proposition follows by the construction of automaton $C_{k',\ell'}$.

**Proposition 8.** There are exactly $k'$ distinct runs of $A$ on $w$ and exactly $\ell'$ runs of $B$ on $w$, if and only if there is an accepting run of $C_{k',\ell'}$ on $w$.

Consider the set of spines of $C_{k',\ell'}$ in which all $k' + \ell' + 3$ components of the last state are accepting states; let $m$ be the size of this set of accepting spines. For every such accepting spine $\gamma$, we define an instance of vectors $p^{\gamma}, q_{1}^{\gamma}, \ldots, q_{m}^{\gamma}, r^{\gamma}, s_{1}^{\gamma}, \ldots, s_{m}^{\gamma}$. If we set $n = |\text{Periods}(Q(\gamma))|$ and (arbitrarily) enumerate all simple cycles in $\text{Periods}(Q(\gamma))$ from 1 to $n$, then

- $p^{\gamma}$ has $k'$ components: for every $i$, such that $1 \leq i \leq k'$, we set $p_{i}^{\gamma}$ to be the product of the probabilities of the transitions in the $i$-th copy of $A$ in spine $\gamma$;
- $q_{i}^{\gamma}$ has $n$ components: for every $1 \leq j \leq n$, we set the $j$-th component of $q_{i}^{\gamma}$ to be the product of the probabilities of the transitions in $i$-th copy of $A$ in the $j$-th cycle in the set $\text{Periods}(Q(\gamma))$;
- $r^{\gamma}$ has $\ell'$ components: for every $i$, such that $1 \leq i \leq \ell'$, we set $r_{i}^{\gamma}$ to be the product of the probabilities of the transitions in the $i$-th copy of $B$ in spine $\gamma$;
- $s_{i}^{\gamma}$ has $n$ components: for $1 \leq j \leq n$, we set the $j$-th component of $s_{i}^{\gamma}$ to be the product of the probabilities of the transitions in $i$-th copy of $B$, in the $j$-th cycle in the set $\text{Periods}(Q(\gamma))$.

In the special case when $k' = 0$ or $\ell' = 0$ we put 0 everywhere (which can be understood as a 0-dimensional vector).

For an arithmetic expression $E$ over $n$ variables $x$ indexed by elements of a set $I$, and for a function $\sigma : I \rightarrow \mathbb{N}$, we write $E[\sigma/x]$ for the numerical value of the expression $E$ in which every occurrence of variable $x_i$ was replaced by $\sigma(i)$, for every $i \in I$. The following proposition follows again by the construction of automaton $C_{k',\ell'}$, taking into account the following observations:

- the probability of a run of a probabilistic automaton can be determined from its simple cycle decomposition $(\gamma, \sigma)$, by taking the product of the following:
  - the product of the probabilities of the transitions in spine $\gamma$,
  - for every simple cycle $\omega \in \text{Periods}(Q(\gamma))$, the $\sigma(\omega)$-th power of the product of the probabilities of the transitions in $\omega$;
- in an accepting run of $C_{k',\ell'}$ on a word $w \in \Sigma^*$, the $k'$ components that correspond to $k'$ copies of $A$ all follow a distinct run of $A$ on $w$, and hence by Proposition 8, $[A](w)$ is the sum of the probabilities of the $k'$ distinct runs followed by the $k'$ copies of $A$;
- in an accepting run of $C_{k',\ell'}$ on a word $w \in \Sigma^*$, the $\ell'$ components that correspond to $\ell'$ copies of $B$ all follow a distinct run of $B$ on $w$, and hence by Proposition 8, $[B](w)$ is the sum of the probabilities of the $\ell'$ distinct runs followed by the $\ell'$ copies of $B$.

**Proposition 9.** If there is an accepting run $\rho$ of $C_{k',\ell'}$ on word $w \in \Sigma^*$, then for every simple cycle decomposition $(\gamma, \sigma)$ of $\rho$, we have $[A](w) = S(p^\gamma, q_{1}^{\gamma}, \ldots, q_{m}^{\gamma})[\sigma/x]$ and $[B](w) = S(r^{\gamma}, s_{1}^{\gamma}, \ldots, s_{m}^{\gamma})[\sigma/x]$. 




Let $\Delta$ be the set of all tuples $(p^γ, q^γ_1, \ldots, q^γ_{k'}, r^γ, s^γ_1, \ldots, s^γ_{l'})$ given by all accepting spines, in particular $|\Delta|$ is finite. We now argue that the two conditions in the statement of Proposition 3 are indeed equivalent. Let $w \in \Sigma^*$ be a word such that $[A](w) > [B](w)$. Let the numbers of distinct accepting runs of $A$ and $B$, respectively, on $w$ be $k'$ and $l'$, respectively. Then, by Proposition 3 there is an accepting run $\rho$ of $C_{k',l'}$ on $w$. Let $(\gamma, \sigma)$ be a simple cycle decomposition of $\rho$; note that since $\gamma$ is an accepting spine, we have $(p^γ, q^γ_1, \ldots, q^γ_{k'}, r^γ, s^γ_1, \ldots, s^γ_{l'}) \in \Delta$. It then follows by Proposition 3 that $\sigma$ is a non-negative integer solution of $S(p^γ, q^γ_1, \ldots, q^γ_{k'})[\sigma/x] > S(r^γ, s^γ_1, \ldots, s^γ_{l'})[\sigma/x]$.

Conversely, suppose that there is a non-negative integer solution $\sigma$ of the inequality

$$S(p^γ, q^γ_1, \ldots, q^γ_{k'})[\sigma/x] > S(r^γ, s^γ_1, \ldots, s^γ_{l'})[\sigma/x],$$

for some quadruple

$$(p^γ, q^γ_1, \ldots, q^γ_{k'}, r^γ, s^γ_1, \ldots, s^γ_{l'}) \in \Delta.$$ 

This quadruple is in $\Delta$ because $\gamma$ is an accepting spine of the automaton $C_{k',l'}$ for some $k'$ and $l'$, such that $0 \leq k' \leq k$ and $0 \leq l' \leq l$. Let $\rho$ be an accepting run of $C_{k',l'}$ that is obtained by injecting into $\gamma$, in some order, $\sigma(\omega)$ copies of the simple cycle $\omega$, for all $\omega \in \text{Periods}(Q(\gamma))$; let $w \in \Sigma^*$ be the word underlying the run $\rho$. By Proposition 3 it follows that $[A](w) > [B](w)$.

4 Decidability of the case unambiguous vs. finitely ambiguous

In this section we will show the more challenging part of Theorem 10 i.e., that the containment problem is decidable for $A$ unambiguous and $B$ finitely ambiguous. Our proof is conditional on the first-order theory of the reals with the exponential function being decidable. In [14], the authors show that this is the case if a conjecture due to Schanuel and regarding transcendental number theory is true.

|$\blacktriangleright$ Theorem 10. Determining whether $[A] \leq [B]$ is decidable when $A$ is unambiguous and $B$ is finitely ambiguous, assuming Schanuel’s conjecture is true.

4.1 Integer programming problem with exponentiation

Given two positive integers $n$ and $\ell$, we define $F_{n,\ell}$ to be the set of all the functions $f : \mathbb{R}^n \to \mathbb{R}$ such that there exist $r \in \mathbb{Q}_0^{\ell,n}$ and $s_1, \ldots, s_\ell \in \mathbb{Q}_0^n$ such that $f(x) = \sum_{i=1}^{\ell} r_i s_{i,1} \cdots s_{i,n}$. Observe that this is just a lifting of the $S(\cdot)$ function, defined in the previous section, to real-valued parameters. Consider the following integer programming problem with exponentiation.

|$\blacktriangleright$ Problem 11 (IP+EXP).

$\begin{itemize}
\item \textbf{Input:} Three positive integers $n$, $\ell$ and $m$, a function $f \in F_{n,\ell}$, a matrix $M \in \mathbb{Z}^{m \times n}$, and a vector $c \in \mathbb{Z}^m$.
\item \textbf{Question:} Does there exist $x \in \mathbb{Z}^n$ such that $f(x) < 1$ and $Mx < c$?
\end{itemize}$

In the sequel, we will show that the above problem is decidable.

|$\blacktriangleright$ Theorem 12. The IP+EXP problem is decidable, assuming Schanuel’s conjecture is true.

Theorem 10 is a direct corollary of Theorem 12.
Proof of Theorem 10. Proposition 3 shows that, in order to prove Theorem 10, it is sufficient to decide, given an integer $n$ and positive rational numbers $p, r_i, q_j, s_i, j$ for $i \in \{1, \ldots, \ell\}$, $j \in \{1, \ldots, n\}$, whether there exist $x_1, \ldots, x_n \in \mathbb{N}$ such that $pq^1_1 \cdots q^\ell_1 > \sum_{i=1}^{\ell} r_i s^1_i \cdots s^{x_n}_i$ or equivalently, whether there exist $x_1, \ldots, x_n \in \mathbb{N}$ such that:

$$\sum_{i=1}^{\ell} r_i p^{-1}(s_i q^{-1}_1 x_1 \cdots (s_i n q^{-1}_n x_n) < 1. \tag{2}$$

Define $f : \mathbb{R}^n \to \mathbb{R}$ such that $f(x) = \sum_{i=1}^{\ell} r_i p^{-1}(s_i q^{-1}_1 x_1 \cdots (s_i n q^{-1}_n x_n$. Then, inequality (2) becomes $f(x) < 1$. We can now apply Theorem 12 with $m$ set to be $n$; $M$, to be $-Id$, where $Id$ is the identity matrix; and $c$ to be the null vector. $\blacktriangleleft$

Since the IP+EXP problem is semi-decidable (indeed, we can enumerate the vectors $x$ in $\mathbb{Z}^n$ to find one satisfying the conditions), it will suffice to give a semi-decision procedure to determine whether the inequalities $f(x) < 1 \land Mx < c$ have no integer solution. We give now such a procedure.

4.2 Semi-decision procedure for the complement of IP+EXP

Consider as input for the IP+EXP problem three positive integers $n, \ell, m$, a function $f \in F_{n, \ell}$, a matrix $M \in \mathbb{Z}^{m \times n}$, and a vector $c \in \mathbb{Z}^m$. Denote by $X$ the set of real solutions of the problem, i.e., the set of vectors $X = \{x \in \mathbb{R}^n \mid f(x) < 1 \land Mx < c\}$.

Proc($n, \ell, m, f, M, c$):

1. Search for a non-zero vector $d \in \mathbb{Z}^n$ and $a, b \in \mathbb{Z}$ such that $\{d^\top x \mid x \in X\} \subseteq [a, b]$. Set $i = a$.
2. If $i > b$, then stop and return YES. Otherwise, let $Y_i$ be the set of vectors $x \in \mathbb{Z}^n$ satisfying $d_1 x_1 + \cdots + d_n x_n = i$. If $Y_i$ is empty, then increment $i$ and start again from step 2. Otherwise:
   a. Compute $N \in \mathbb{Z}^{n \times (n-1)}$ and $h \in \mathbb{Z}^n$ such that $Y_i = \{Ny + h \mid y \in \mathbb{Z}^{n-1}\}$.
   b. If $n - 1 = 0$ and $f(h) < 1 \land Mh < c$ then return NO, otherwise increment $i$ and start again from step 2.
   c. If $n - 1 > 0$ then recursively call Proc($n - 1, \ell, m, f', M', c'$), where $f' \in F_{n-1, \ell}$ is defined as $f'(y) = f(Ny + h); M' \in \mathbb{Z}^{m \times (n-1)}$, as $M' = MN$; and $c' \in \mathbb{Z}^m$, as $c - Mh$. If the procedure stops and returns YES then increment $i$ and start again from step 2. If the procedure stops and returns NO then return NO.

$\blacktriangleleft$ Lemma 13. The above semi-decision procedure stops and outputs YES if and only if there is no integer valuation of $x$ that satisfies the constraints, i.e. $X \cap \mathbb{Z}^n$ is empty.

We prove this lemma in Section 4.3. Before, let us shortly comment on both steps of the procedure.

Step 1 of the procedure

First, notice that the only step which might not terminate in a call to our procedure is step 1. Indeed, once $d, a$, and $b$ are fixed, there are only finitely many integers $i \in [a, b]$ that have to be considered in step 2.

Moreover, for each integer vector $d \in \mathbb{Z}^n$ and $a, b \in \mathbb{Z}$, the inclusion $\{d^\top x \mid x \in X\} \subseteq [a, b]$ that needs to be checked in step 1 can be formulated as a decision problem in the first-order
logic over the structure \((\mathbb{R}, +, \times, \exp)\). Since this structure has a decidable first-order theory subject to Schanuel’s conjecture [14], the inclusion can be decided for each fixed \(d, a,\) and \(b\).

**Step 2 of the procedure**

For fixed \(d, a,\) and \(b,\) one can compute in a standard way the set of all integer solutions \(Y_i\) (see, e.g., [3]), as we now explain. By performing elementary column operations, find a \(n \times n\) unimodular (i.e. with determinant equal to 1 or \(-1\)) integer matrix \(U\) such that

\[
(d_1, \ldots, d_n)U = (g, 0, \ldots, 0),
\]

where \(g = \gcd(d_1, \ldots, d_n)\). Recall that \(Y_i\) is the set of integer solutions of \(d^\top x = i\). We apply the change of variables \(Uy = x\) to it, where \(y = (y_1, \ldots, y_n)\), to obtain \(d^\top Uy = i\). Since \(d^\top Uy = (g, 0, \ldots, 0)y = gy_1\), the transformed equation is \(gy_1 = i\) and the matrix \(U\) gives a one-to-one correspondence between integer solutions \(y\) of the transformed equation and solutions \(x \in Y_i\). Now the transformed equation has a solution if and only if \(g\) divides \(i\), in which case \(y_1 = i/g\). Furthermore, in this case a general solution from \(Y_i\) can be written in the form \(x = Ny' + h\) for \(N\) a \(n \times (n - 1)\) integer matrix and \(h \in \mathbb{Z}^n\) (both derived from \(U\)) and \(y' = (y_2, \ldots, y_n)\).

### 4.3 Proof of Lemma [13]

The proof of Lemma [13] relies on the two following lemmas. The first one is the most technical contribution of the paper and is proved in Section 4.4. It ensures termination of step 1 in the procedure when there is no integer solution.

- **Lemma 14.** If the set \(X\) contains no integer point then there must exist a non-zero integer vector \(d \in \mathbb{Z}^n\) and \(a, b \in \mathbb{Z}\) such that \(\{d^\top x \mid x \in X\} \subseteq [a, b]\).

This second lemma guarantees that the recursive calls in step 2 guarantee the correct output.

- **Lemma 15.** Given a non-zero vector \(d \in \mathbb{Z}^n\) and an integer \(i,\) there exists \(x \in \mathbb{Z}^n\) such that \(f(x) < 1 \wedge Mx < c \wedge d^\top x = i\) if and only if there exists \(y \in \mathbb{Z}^{n-1}\) such that \(f'(y) < 1 \wedge M'y < c'\) where \(f', M',\) and \(c'\) are as defined in the procedure.

**Proof.** We want to prove that given a non-zero vector \(d \in \mathbb{Z}^n\) and an integer \(i,\) there exists \(x \in \mathbb{Z}^n\) such that \(f(x) < 1 \wedge Mx < c \wedge d^\top x = i\) if and only if there exists \(y \in \mathbb{Z}^{n-1}\) such that \(f'(y) < 1 \wedge M'y < c'\) where \(f', M',\) and \(c'\) are as defined in the procedure. Recall that \(Y_i\) is the set of vectors \(x\) such that \(d^\top x = i\) and that \(Y_i = \{Ny + h \mid y \in \mathbb{Z}^{n-1}\}\) for some \(N \in \mathbb{Z}^{n \times (n-1)}\) and \(h \in \mathbb{Z}^n\).

Let \(x \in \mathbb{Z}^n\) such that \(f(x) < 1 \wedge Mx < c \wedge d^\top x = i\).

Then \(x \in Y_i\) and thus there is \(y \in \mathbb{Z}^{n-1}\) such that \(x = Ny + h\). We have: \(f'(y) = f(Ny + h) = f(x) < 1\) and \(M'y = MNy = M(x - h) = Mx - Mh < c - Mh = c'\).

Conversely, consider \(y \in \mathbb{Z}^{n-1} \cap Y_i\) such that \(f'(y) < 1 \wedge M'y < c'\). Let \(x = Ny + h\). Then \(x \in Y_i\) and thus \(d^\top x = i\). Moreover, \(f(x) = f(Ny + h) = f'(y) < 1\) and \(Mx = M(Ny + h) = MNy + Mh = M'y + Mh < c' + Mh = c\).

We prove Lemma [13]
First direction: when the procedure returns YES. Suppose first that the semi-decision procedure stops and outputs YES. Then there exist a non-zero vector \( d \in \mathbb{Z}^n \) and \( a, b \in \mathbb{Z} \) such that \( \{ d^\top x \mid x \in X \} \subseteq [a, b] \) as in step 1, and for all integers \( i \in [a, b] \), one of the following situations occurs:

1. \( Y_i \) is empty,
2. \( n - 1 = 0 \), \( Y_i = \{ h \} \) as defined in step 2.a but \( h \) is not an integer solution of the problem,
3. \( n - 1 > 0 \) and the recursive call stops and outputs YES.

By definition of \( d \), in order to prove that there is no integer solution of the problem, we need to show that in all those cases, and for all \( i \in [a, b] \), \( Y_i \cap X = \emptyset \). It is clear for items 1 and 2 and we use Lemma 15 and an induction for item 3.

Second direction: when \( X \cap \mathbb{Z}^n = \emptyset \). If there is no integer solution then by Lemma 14 there must exist a non-zero vector \( d \in \mathbb{Z}^n \) and \( a, b \in \mathbb{Z} \) such that \( \{ d^\top x \mid x \in X \} \subseteq [a, b] \) as in step 1. Moreover, for any of those choices, if for an integer \( i \in [a, b] \), the set \( Y_i \) of vectors \( x \in \mathbb{Z}^n \) satisfying \( d_1x_1 + \cdots + d_nx_n = i \) is non-empty, then,

1. if \( n = 1 \), then \( h \) as defined in step 2.a is not a solution of the problem (by hypothesis) and thus the procedure stops and returns YES,
2. if \( n > 1 \), we use Lemma 15 and, by induction, the recursive call must return YES.

4.4 Proof of Lemma 14

Fix three positive integers \( n, \ell, m \), a function \( f \in F_{n, \ell} \), a matrix \( M \in \mathbb{Z}^{m \times n} \), and a vector \( c \in \mathbb{Z}^n \). Recall that we denote by \( X \) the set of vectors

\[
X = \{ x \in \mathbb{R}^n \mid f(x) < 1 \land Mx < c \}.
\]

We want to prove that if the set \( X \) contains no integer point then there must exist a non-zero integer vector \( d \in \mathbb{Z}^n \) and \( a, b \in \mathbb{Z} \) such that \( \{ d^\top x \mid x \in X \} \subseteq [a, b] \).

We will use the following corollary of Kronecker’s theorem on simultaneous Diophantine approximation. It generalises the fact that any line in the plane with irrational slope passes arbitrarily close to integer points in the plane.

\[\blacktriangleright\text{Proposition 16. (Corollary 2.8).}\] Let \( u, u_1, \ldots, u_s \) be vectors in \( \mathbb{R}^n \). Suppose that for all \( d \in \mathbb{Z}^n \) we have \( d^\top u = 0 \) whenever \( d^\top u_1 = \cdots = d^\top u_s = 0 \). Then for all \( \varepsilon > 0 \) there exist real numbers \( \lambda_1, \ldots, \lambda_s \geq 0 \) and a vector \( v \in \mathbb{Z}^n \) such that \( \| u + \sum_{i=1}^s \lambda_i u_i - v\|_{\infty} \leq \varepsilon \).

By definition, there exist vectors \( r \in \mathbb{Q}_{\geq 0}^\ell \) and \( s_1, \ldots, s_{\ell} \in \mathbb{Q}_{\geq 0}^{n_0} \) such that \( f(x) = \sum_{i=1}^\ell r_i s_{i,1} \cdots s_{i,n_0} \). Let \( a \in \mathbb{R}^\ell \) and \( b_1 \in \mathbb{R}^n \) be defined by \( a_i = \log(r_i) \) and \( b_i = (\log(s_{i,1}), \ldots, \log(s_{i,n_0})) \). We can then rewrite \( f(x) \) as follows

\[
f(x) = \exp(b_1^\top x + a_1) + \cdots + \exp(b_\ell^\top x + a_\ell).
\]

Let us now consider the cone

\[
C = \left\{ x \in \mathbb{R}^n \mid b_1^\top x \leq 0 \land \cdots \land b_\ell^\top x \leq 0 \land Mx \leq 0 \right\}.
\]

It is easy to see that \( X \cap C \subseteq X \).

\[\blacktriangleright\text{Lemma 17.}\] Suppose that \( X \) is non-empty and that no non-zero integer vector in \( \mathbb{Z}^n \) is orthogonal to \( C \). Then \( X \cap \mathbb{Z}^n \) is non-empty.
Two-counter machines (or Minsky machines) bounded.

The reduction resembles the one used to prove undecidability of the comparison problem for deterministic finite-state machine with two counters that can be incremented, decremented, or tested for 0. Formally, it is given by a tuple $(Q, T_1^+, T_2^+, T_2^-, q_{init}, q_{halt})$ where:

- $Q$ is a finite set of states.
- $T_1^+$ (resp. $T_2^+$) is a subset of $Q^2$. If $(p, q) \in T_1^+$ (resp. $T_2^+$) then there is a transition from the state $p$ to the state $q$ which increments the first counter (resp. second counter).

Proof. Let $u \in X$. Since $X$ is open, there exists $\varepsilon > 0$ such that the open ball $B_\varepsilon(u)$ is contained in $X$. We therefore have that $B_\varepsilon(u) + C \subseteq X$.

We will apply Proposition 16 to show that $B_\varepsilon(u) + C$ contains an integer point and hence that $X$ contains an integer point. To this end, let vectors $u_1, \ldots, u_s \in C$ be such that $\text{span}\{u_1, \ldots, u_s\} = \text{span}(C)$. Then no non-zero vector in $\mathbb{Z}^n$ is orthogonal to $u_1, \ldots, u_s$. By Proposition 16 there exist real numbers $\lambda_1, \ldots, \lambda_s \geq 0$ and an integer vector $v \in \mathbb{Z}^n$ such that $\|u + \sum_{i=1}^s \lambda_i u_i - v\|_\infty \leq \varepsilon$. Thus, $v \in B_\varepsilon(u) + C \subseteq X$.

The contrapositive of the above result states that if $X$ contains no integer point, then there must exist an integer vector that is orthogonal to $C$. For the desired result, it remains for us to prove the boundedness claim.

Lemma 18. Suppose that $d \in \mathbb{Z}^n$ is orthogonal to the cone $C$. Then $\{d^\top u \mid u \in X\}$ is bounded.

Proof. Define the “enveloping polygon” of $X$ to be

$$\hat{X} = \left\{ x \in \mathbb{R}^n \mid b_1^\top x + a_1 \leq 0 \wedge \cdots \wedge b_\ell^\top x + a_\ell \leq 0 \wedge Mx \leq c \right\}.$$ 

Clearly it holds that $X \subseteq \hat{X}$. Moreover, by the Minkowski-Weyl decomposition theorem we can write $\hat{X}$ as a sum $\hat{X} = B + C$ for $B$ a bounded polygon and $C$ the cone defined in (3). Since $d$ is orthogonal to $C$ by assumption, it follows that $\{d^\top u \mid u \in \hat{X}\} = \{d^\top u \mid u \in B\}$ is bounded and hence $\{d^\top u \mid u \in X\}$ is bounded. The result immediately follows.

We can now complete the proof of Lemma 14.

Proof of Lemma 14. By Lemma 17 there exists a non-zero integer vector $d \in \mathbb{Z}^n$ such that $d$ is orthogonal to the cone $C$ defined in (3). Then by Lemma 18 we obtain that $\{d^\top u \mid u \in X\}$ is contained in a bounded interval.

5 Undecidability for linearly ambiguous automata

In this section we prove Theorem 1. That is, we argue that the containment problem is undecidable for the class of linearly ambiguous PA. We will prove a more general result in Proposition 20 that the boundedness problem is undecidable. Theorem 1 is an immediate corollary of Proposition 20 since it is trivial to construct a PA that outputs probability $\frac{1}{2}$ for all words.

The proof is done by a reduction from the halting problem for two-counter machines. The reduction resembles the one used to prove undecidability of the comparison problem for another quantitative extension of Boolean automata: max-plus automata [3, 11].

5.1 Two-counter machines

Two-counter machines (or Minsky machines) can be defined in several ways, all equivalent in terms of expressiveness. We use here the following description: A two-counter machine is a deterministic finite-state machine with two counters that can be incremented, decremented, or tested for 0. Formally, it is given by a tuple $(Q, T_1^+, T_2^+, T_2^-, q_{init}, q_{halt})$ where:

- $Q$ is a finite set of states.
- $T_1^+$ (resp. $T_2^+$) is a subset of $Q^2$. If $(p, q) \in T_1^+$ (resp. $T_2^+$) then there is a transition from the state $p$ to the state $q$ which increments the first counter (resp. second counter).
Containment for Probabilistic Automata

- $T_1^-$ (resp. $T_2^-$) is a subset of $Q^2$. If $(p, q, r) \in T_1^-$ (resp. $T_2^-$) then there is a transition from the state $p$ which goes to the state $q$ if the current value of the first (resp. second) counter is 0 (it does not change the counters), and which goes to the state $r$ otherwise and decrements the first (resp. second) counter.

- $q_{\text{init}} \in Q$ is the initial state and $q_{\text{halt}} \in Q$ is the final state such that there is no outgoing transition from $q_{\text{halt}}$ (for all transitions $(q, p) \in T_1^+ \cup T_2^+$ or $(q, p, r) \in T_1^- \cup T_2^-$, $q \neq q_{\text{halt}}$).

We also assume that $q_{\text{init}} \neq q_{\text{halt}}$.

Moreover the machine is deterministic: for every state there is at most one action that can be performed, i.e. for all $q \in Q$ there is at most one transition of the form $(q, p)$ or $(q, p, r)$ in $T_1^+ \cup T_2^+ \cup T_1^- \cup T_2^-$ and $T_1^- \cap T_2^+ = \emptyset$ and $T_1^+ \cap T_2^- = \emptyset$.

The semantics of a two-counter machine are given by means of the valuations of the counters that are pairs of non-negative integers. An execution with counters initialised to $(n_0^1, n_0^2)$ is a sequence of compatible transitions and valuations denoted by

$$(n_0^1, n_0^2) \xrightarrow{t_1} (n_1^1, n_1^2) \xrightarrow{t_2} (n_2^1, n_2^2) \xrightarrow{\ldots} t_k \xrightarrow{(n_k^1, n_k^2)}$$

such that:

- for all $i \in \{1, \ldots, k\}$, if $t_i \in T_1^+$ (resp. $T_2^+$), then $n_1^i = n_1^{i-1} + 1$ and $n_2^i = n_2^{i-1}$ (resp. $n_1^i = n_1^{i-1}$ and $n_2^i = n_2^{i-1} + 1$);
- for all $i \in \{1, \ldots, k\}$, if $t_i \in T_1^-$ (resp. $T_2^-$), then $n_1^i = n_1^{i-1} = 0$ or $n_1^i = n_1^{i-1} - 1$ and $n_2^i = n_2^{i-1}$ (resp. $n_2^i = n_2^{i-1} = 0$ or $n_2^i = n_2^{i-1} - 1$ and $n_1^i = n_1^{i-1}$);
- for all $i \in \{1, \ldots, k-1\}$, if $t_i = (p_i, q_i) \in T_1^+ \cup T_2^+$ then $t_{i+1} \in \{q_i\} \times (Q \cup Q^2)$;
- for all $i \in \{1, \ldots, k-1\}$, if $t_i = (p_i, q_i, r_i) \in T_1^- \cup T_2^-$ and $n_{i-1} = 0$ then $t_{i+1} \in \{q_i\} \times (Q \cup Q^2)$, otherwise if $n_{i-1} \neq 0$ then $t_{i+1} \in \{r_i\} \times (Q \cup Q^2)$.

We say that the machine halts if there is a (unique) execution with counters initialised to $(0, 0)$ starting in $q_{\text{init}}$ reaching the state $q_{\text{halt}}$.

- **Proposition 19** ([15]). The halting problem for two-counter machines is undecidable.

### 5.2 Reduction from the halting problem for two-counter machines

- **Proposition 20.** Given a two counter machine, one can construct a linearly ambiguous probabilistic automaton $A$ such that the machine halts if and only if there exists a word $w$ such that $|A|(w) \geq \frac{1}{2}$ (resp. $>, \leq, <$).

We follow these steps:

1. We construct two linearly ambiguous PA $A$ and $B$ such that the machine halts if and only if there is a word $w$ such that $|A|(w) \leq |B|(w)$.
2. From $A$ and $B$, we construct $A'$ and $B'$, also linearly ambiguous, such that the machine halts if and only if there is a word $w$ such that $|A'(w)| < |B'(w)|$.
3. We show that the functions $1 - |A|$, $1 - |B|$, $1 - |A'|$, $1 - |B'|$ are also computed by linearly ambiguous PA.

We show that Proposition [20] follows from steps 1, 2, and 3. We have that for all words $w$

$$|A|(w) \leq |B|(w)$$

$$\iff \frac{1}{2} |A|(w) + \frac{1}{4} (1 - |B|(w)) \leq \frac{1}{2}$$

$$\iff \frac{1}{2} |B|(w) + \frac{1}{4} (1 - |A|(w)) \geq \frac{1}{2}$$
We then define to prove undecidability of the two other variants with strict inequalities.

Let \( T = T_1^+ \cup T_2^+ \cup T_1^- \cup T_2^- \) and \( \Sigma = \{a, b\} \cup T \). The idea is to encode the executions of the two-counter machine into words over the alphabet \( \Sigma \). A block \( a^m \) (resp. \( b^m \)) encodes the fact that the value of the first (resp. second) counter is \( m \). For example, given \( t \in T_1^+ \) and \( t' \in T_2^- \), a word \( a^n b^m t a^{n+1} b^m t' a^{n+1} b^m \), encodes an execution starting with value \( n \) in the first counter and \( m \) in the second counter. Transition \( t \) then increases the value of the first counter to \( n + 1 \) without changing the value of the second one. The configuration is thus encoded by the infix \( a^{n+1} b^m \). Next, transition \( t' \) is taken, and either \( m' = m = 0 \) or \( m' = m - 1 \). Moreover, if \( t = (p, q) \) and \( t' = (r, s, u) \) then \( q = r \) (i.e., the states between transitions have to match).

The PA \( A \) and \( B \) are constructed in such a way that for all words \( w \) it holds that

\[
\begin{cases}
[A](w) = [B](w) & \text{if } w \text{ represents a valid halting execution of the machine} \\
[A](w) > [B](w) & \text{otherwise}.
\end{cases}
\]

The automata \( A \) and \( B \) are constructed as a weighted sum of seven PA, each of them checking some criteria that a word \( w \) should fulfill in order to represent a valid halting execution.

**Automaton \( A_0 \).** Conditions such as asking that the encoded execution start in \( q_{\text{init}} \), end in \( q_{\text{halt}} \), represent a valid path in the machine (with respect to the states), and that the encoding be of the good shape, i.e. contain alternating blocks of \( a \)'s, blocks of \( b \)'s and letters from \( T \), are all regular conditions that can thus be checked by a (deterministic Boolean) automaton. The exhaustive list of such conditions and their explanations are given below.

We then define \( A_0 \) to be a deterministic PA such that \( [A_0](w) = 0 \) if and only if \( w \) satisfies all these regular conditions and \( [A_0](w) = 1 \) otherwise.

We give here a precise description of the conditions checked by \( A_0 \).

1. The word \( w \) belongs to \( T_{\text{init}}((a^* b^*) T)^* \) where \( T_{\text{init}} \) is the subset of \( T \) of the transitions started in \( q_{\text{init}} \).
2. The word \( w \) represents an execution ending in \( q_{\text{halt}} \), i.e. it either ends
   - with a letter \((q, q_{\text{halt}})\),
   - with a word of the form \( t a^n(q, q_{\text{halt}}, r) \) where \( t \in T \), \((q, q_{\text{halt}}, r) \in T_2^− \) and \( n \) is a non-negative integer (resp. \( t b^n(q, q_{\text{halt}}, r) \) where \( t \in T \), \((q, q_{\text{halt}}, r) \in T_1^− \) and \( n \) a non-negative integer),
   - or with a word of the form \( t a^n b^m(q, r, q_{\text{halt}}) \) where \( t \in T \), \((q, r, q_{\text{halt}}) \in T_2^− \) and \( m \) a positive integer (resp. \( t a^n b^m(q, r, q_{\text{halt}}) \) where \( t \in T \), \((q, r, q_{\text{halt}}) \in T_1^− \) and \( n \) a positive integer).
3. The transitions are state-compatible, i.e. if \( w \) contains a factor \((p, q) a^n b^m t \) with \( t \in T \) then \( t \) starts in \( q \) and if \( w \) contains a factor \( t' a^n b^m(p, q, r) a^n b^m t \) with \( t', t' \in T \) and \((p, q, r) \in T_1^− \) (resp. \( T_2^− \)) then \( t \) starts in \( q \) if \( n = 0 \) and in \( r \) if \( n > 0 \) (resp. \( t \) starts in \( q \) if \( m = 0 \) and in \( r \) if \( m > 0 \)).
4. We also check that if in the execution represented by the word, at some point the value in the first (resp. second) counter is 0 and a transition from $T_1^- \ (\text{resp. } T_2^-)$ is taken then the value in the counter is still 0 after the transition. In terms of words, this means that if $t b^m t' a^n b^m t''$ is a factor of the word with $t, t' \in T$ and $t' \in T_1^-$ then $n = 0$ (and similarly for the second counter).

The automaton $A_0$ will make sure that $[A](w) = 0$ only if $w$ is proper, i.e. of the good shape as given above. We are now left to check that the counters are properly incremented and decremented.

Automata $A_1$ and $B_1$. The automata $A_1$ and $B_1$ check that a proper word encodes an execution where the first counter is always correctly incremented after reading transitions from $T_1^+$. Consider the automaton $C(x,y,z)$ in Figure 3. It is parameterised by three probability variables $x,y,z > 0$. The parameter $x$ is the probability used by the initial distribution, and parameters $y$ and $z$ are used by some transitions.

![Figure 3](image)

Figure 3 Gadget automaton $C(x,y,z)$ used to check if the first counter is incremented properly.

We only take into consideration proper words as given by the automaton $A_0$. Notice that the only non-deterministic transitions in $C(x,y,z)$ are the ones going out from the the leftmost state upon reading letters from $T$. It follows that $C(x,y,z)$ is linearly ambiguous. In fact, for every position in $w$ labelled by an element $t$ from $T_1^+$ there is a unique accepting run that first reaches a final state upon reading $t$. By construction, we have

$$[C(x,y,z)](w) = \sum_{t_i \in T_1^+} x \left(\frac{1}{2}\right)^{i-1} y^{n_i+1} z^{n_{i+1}}. \quad (4)$$

Let $x = \frac{1}{2}$, $y = 1$ and $z = \frac{1}{2}$. We define $B_1$ as $C(x,x,x)$ and $A_1$ as a weighted sum of $C(x,y,z)$ and $C(x,z,y)$ with weights $(\frac{1}{2}, \frac{1}{2})$. Since $C(\cdot,\cdot,\cdot)$ is linearly ambiguous the obtained automata are also linearly ambiguous. We prove that $[A_1](w) = [B_1](w)$ only if $n_i + 1 = n_{i+1}$ for all $i$ such that $t_i \in T_1^+$ and $[A_1](w) > [B_1](w)$ otherwise.

By (4) it suffices to show that for every $i$ it holds that

$$\left(\frac{1}{2}\right)^{n_i+1} \leq \frac{1}{2} \left(\frac{1}{4}\right)^{n_i} + \left(\frac{1}{4}\right)^{n_{i+1}}$$

and that the equality holds only if $n_i + 1 = n_{i+1}$. Let $p = \frac{1}{2}^{n_i+1}$ and $q = \frac{1}{2}^{n_{i+1}}$ then this reduces to

$$pq \leq \frac{1}{2} (p^2 + q^2).$$

This is true for every $p, q$ and moreover the equality holds if and only if $p = q$, which is equivalent to $n_i + 1 = n_{i+1}$. We conclude with the following remark that will be useful for Step 2.
Remark. If $\llbracket A_1 \rrbracket(w) > \llbracket B_1 \rrbracket(w)$ then $\llbracket A_1 \rrbracket(w) \geq \llbracket B_1 \rrbracket(w) + \left(\frac{1}{2}\right)^{2(|w| + 1)}$. To obtain this bound notice that the probabilities and initial distribution of the automata $A_1$ and $B_1$ are 0, 1, $\frac{1}{2}$ and $\frac{1}{4}$. Hence for every word $w$ the probability assigned to it is either 0 or a multiple of $\left(\frac{1}{2}\right)^{|w| + 1}$. It follows that if $A_1$ and $B_1$ assign different probabilities to $w$ then the difference is at least $\left(\frac{1}{2}\right)^{|w| + 1}$, which proves the remark.

Automata $A_2, B_2, \ldots$ Similarly, we construct automata $A_2, B_2, A_3, B_3, \ldots A_6, B_6$ to check the other criteria:

- the value of the second counter is correctly incremented when taking a transition from $T_2^+$,
- the value of the first (resp. second) counter remains the same when using a transition from $T_2^+ \cup T_2^-$ (resp. $T_1^+ \cup T_1^-$),
- the first (resp. second) counter is correctly decremented when using a transition from $T_1^-$ (resp. $T_2^-$) and the current value is not 0.

For example we define $A_5$ and $B_5$ to check if the decrements of the first counter are correct using the gadget automaton $D(x, y, z)$ in Figure 4. Let $x = \frac{1}{2}, y = 1$ and $z = \frac{1}{4}$. We define $B_5$ as $C(x, x, x)$ and $A_5$ as a weighted sum of $D(x, y, z)$ and $D(x, z, y)$ with weights $(\frac{1}{2}, \frac{1}{2})$.

![Figure 4](image)

Gadget automaton $D(x, y, z)$ used to check if the first counter is decremented properly.

For all these automata $A_1, B_1, \ldots A_6, B_6$, we have that for every proper word $w$, $\llbracket A_i \rrbracket(w) = \llbracket B_i \rrbracket(w)$ if the corresponding increments or decrements are properly performed, and $\llbracket A_i \rrbracket(w) > \llbracket B_i \rrbracket(w)$ otherwise. We remark again that, in that case, for all $i$, we have $\llbracket A_i \rrbracket(w) \geq \llbracket B_i \rrbracket(w) + \left(\frac{1}{2}\right)^{2(|w| + 1)}$. Note also that all the automata constructed above are linearly ambiguous (the only non-deterministic choices are in the first states when reading $T$).

Let us define $A$ (resp. $B$) as the weighted sum of the above automata computing the function $\frac{7}{13}[A_0] + \frac{1}{13}[A_1] + \cdots + \frac{1}{13}[A_6]$ (resp. $\frac{1}{13}[B_1] + \cdots + \frac{1}{13}[B_6]$). The PA $A$ and $B$ are linearly ambiguous.

Fact 21. For every word $w$ that does not represent the halting execution exactly one of the two cases below applies:

- either $w$ is not proper, $\llbracket A_0 \rrbracket(w) = 1$, and $\llbracket A \rrbracket(w) > \frac{7}{13} > \frac{6}{13} \geq \llbracket B \rrbracket(w)$;
- or $w$ encodes an execution where the counters are not properly valued in at least one place, and in that case, there exists an $i$ such that $A_i(w) > B_i(w)$ (and $\geq$ for the other $i$). We thus obtain that $\llbracket A \rrbracket(w) > \llbracket B \rrbracket(w)$.

Remark that in the case when $\llbracket A \rrbracket(w) > \llbracket B \rrbracket(w)$ we have $\llbracket A \rrbracket(w) \geq \llbracket B \rrbracket(w) + \frac{1}{13}\left(\frac{1}{2}\right)^{2(|w| + 1)}$.

Fact 22. If the two-counter machine halts then for the word representing the halting execution we have that $\llbracket A \rrbracket(w) = \llbracket B \rrbracket(w)$. 
Step 2

As noticed previously, by construction of $A$ and $B$, we have that:

- there exists a word $w$ such that $[A](w) > [B](w)$ if and only if
- there exists a word $w$ such that $[A](w) \geq [B](w) + \frac{1}{11} \left( \frac{1}{2} \right)^{2(|w|+1)}$ if and only if
- there exists a word $w$ such that

$$\frac{1}{2}[A](w) \geq \frac{1}{2}[B](w) + \frac{1}{26} \left( \frac{1}{2} \right)^{2(|w|+1)}.$$  \hspace{1cm} (5)

Both functions in the last inequality are computed by linearly ambiguous PA. To obtain the weighted sum of the right hand side we need to construct an automaton that outputs, for every word $w$, the probability $\frac{1}{26} \left( \frac{1}{2} \right)^{2(|w|+1)}$. This is easy to obtain by slightly modifying the example in Figure 1. Notice that this is an unambiguous automaton and its complement is linearly ambiguous.

For the rest of this section, we denote by $A'$ and $B'$ the PA for the right and left sides of the inequality (5), respectively.

Step 3

We will now argue that $1 - [A]$, $1 - [B]$, $1 - [A']$, $1 - [B']$ are also computed by linearly ambiguous PA. In Figure 5, we show the complement automaton of the gadget $C(x, y, z)$ from Figure 3. We show that the so-obtained automaton is linearly ambiguous. In the end we will conclude the argument using the property that all components $A_i$ and $B_i$ for $i > 0$ are weighted sums of such gadgets.

![Figure 5](image)

**Figure 5** Complement automaton of $C(x, y, z)$ after trimming.

First, notice that we can trim the automaton to have only three states $p$, $q$ and $\bot$ because the remaining states are not accepting and it is not possible to reach an accepting state from them. Recall that the ambiguity of an automaton relies only on its underlying structure (Section 2.1). Hence we can focus on the Boolean automaton (i.e. without probabilities) from Figure 6 and analyze its ambiguity. In the trimmed automaton there are only two places with nondeterministic choices: when reading a letter in $T$ from $p$ the automaton can either remain in $p$ or move to $q$; when reading $a$ from $q$ the automaton can either remain in $q$ or move to $\bot$.

Notice that all three states are both initial and accepting. Let us decompose the set of accepting runs of the automaton on a word $w$ depending on where the run starts and where the run ends, which is 9 cases in total. We focus only on the case for run starting in $p$ and
ending in \( \perp \); the remaining cases obviously provide at most a linear number of runs. The automaton can move from \( p \) to \( q \) only when reading an element of \( T \). Fix a word \( w \) and consider positions \( i \) and \( i' \) such that \( t_i \in T \) and \( i' \) is maximal such that positions between \( i \) and \( i' \) have labels from \( \Sigma \setminus T \) (i.e. position \( i' \) is labelled with a letter from \( T \) or is equal to \(|w| + 1 \)). We show that there are at most \( f(i) \equiv i' - i + 1 \) accepting runs starting from \( p \) and ending in \( q \) after reading \( t_i \). This is because in state \( q \), when reading an element from \( T \), the automaton has to move to \( \perp \). Hence the number of all accepting runs from \( p \) to \( \perp \) is bounded by the sum of all \( f(i) \) through all positions \( 1 \leq i \leq |w| \) such that \( t_i \in T \). We conclude that the automaton is linearly ambiguous.

To conclude our argument about the ambiguity of all constructed automata, recall that automata \( A, B, A', \) and \( B' \) are constructed as weighted sums of PA obtained from \( A_i \) and \( B_j \), which are weighted sums of gadgets like \( C(x, y, z) \). For example the function \( 1 - \lfloor A \rfloor \) is

\[
1 - \lfloor A \rfloor = 1 - \left( \frac{7}{13} \lfloor A_0 \rfloor + \frac{1}{13} \lfloor A_1 \rfloor + \cdots + \frac{1}{13} \lfloor A_6 \rfloor \right) = \frac{7}{13} \left( 1 - \lfloor A_0 \rfloor \right) + \frac{1}{13} \left( 1 - \lfloor A_1 \rfloor \right) + \cdots + \frac{1}{13} \left( 1 - \lfloor A_6 \rfloor \right)
\]

It is thus sufficient to complement each member of the sum. Showing that all these complements are still linearly ambiguous follows the ideas given above for the automaton in Figure 5.

\section{Conclusion}

In this work we have shown that the containment problem for PA is decidable if one of the automata is finitely ambiguous and the other one is unambiguous. Interestingly, for one of the two cases, our proposed algorithm uses a satisfiability oracle for a theory whose decidability is equivalent to a weak form of Schanuel’s conjecture. We have complemented our decidability results with a proof of undecidability for the case when the given automata are linearly ambiguous.

Decidability of the containment problem when both automata are allowed to be finitely ambiguous remains open. One way to tackle it is to study generalizations of the IP+EXP problem introduced in Section 4. This problem asks whether there exists \( x \in \mathbb{N}^n \) such that \( f(x) < 1 \) and \( Mx < c \) for a given function \( f \) defined using exponentiations, a given matrix \( M \), and vector \( c \). A natural way to extend the latter would be to ask that \( f(x) < g(x) \), where \( g \) is obtained in a similar way as \( f \). The main obstacle, when trying to generalize our decidability proof for that problem, is that we lack a replacement for the cone \( C \) needed in order to obtain a result similar to Lemma 18 using the Minkowski-Weyl decomposition.
References

1. Shaull Almagor, Udi Boker, and Orna Kupferman. What’s decidable about weighted automata? In Tevfik Bultan and Pao-Ann Hsiung, editors, Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011. Proceedings, volume 6996 of Lecture Notes in Computer Science, pages 482–491. Springer, 2011. URL: https://doi.org/10.1007/978-3-642-24372-1_37

2. Rohit Chadha, A. Prasad Sistla, Mahesh Viswanathan, and Yue Ben. Decidable and expressive classes of probabilistic automata. In Andrew M. Pitts, editor, Foundations of Software Science and Computation Structures - 18th International Conference, FoSSaCS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings, volume 9034 of Lecture Notes in Computer Science, pages 200–214. Springer, 2015. URL: https://doi.org/10.1007/978-3-662-46678-0_13

3. Henri Cohen. A course in computational algebraic number theory, volume 138 of Graduate texts in mathematics. Springer, 1993. URL: http://www.worldcat.org/oclc/27810276

4. Thomas Colcombet. On distance automata and regular cost function. Presented at the Dagstuhl seminar “Advances and Applications of Automata on Words and Trees”, 2010.

5. Lu Feng, Tingting Han, Marta Z. Kwiatkowska, and David Parker. Learning-based compositional verification for synchronous probabilistic systems. In Tevfik Bultan and Pao-Ann Hsiung, editors, Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011, volume 6996 of Lecture Notes in Computer Science, pages 511–521. Springer, 2011. URL: https://doi.org/10.1007/978-3-642-24372-1_40

6. Nathanaël Fijalkow. Undecidability results for probabilistic automata. SIGLOG News, 4(4):10–17, 2017. URL: http://doi.acm.org/10.1145/3157831.3157833

7. Nathanaël Fijalkow, Hugo Gimbert, Edon Kelmendi, and Youssouf Oualhadj. Deciding the value 1 problem for probabilistic leaktight automata. Logical Methods in Computer Science, 11(2), 2015. URL: https://doi.org/10.2168/LMCS-11(2:12)2015

8. Nathanaël Fijalkow, Cristian Riveros, and James Worrell. Probabilistic automata of bounded ambiguity. In Roland Meyer and Uwe Nestmann, editors, 28th International Conference on Concurrency Theory, CONCUR 2017, September 5-8, 2017, Berlin, Germany, volume 85 of LIPIcs, pages 19:1–19:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2017.19

9. Holger Giese, Nelly Bencomo, Liliana Pasquale, Andres J. Ramirez, Paola Inverardi, Sebastian Witzoldt, and Siobhán Clarke. Living with uncertainty in the age of runtime models. In Nelly Bencomo, Robert B. France, Betty H. C. Cheng, and Uwe Assmann, editors, Models@run.time - Foundations, Applications, and Roadmaps [Dagstuhl Seminar 14181, November 27 - December 2, 2011], volume 8378 of Lecture Notes in Computer Science, pages 47–100. Springer, 2014. URL: https://doi.org/10.1007/978-3-319-08915-7_3

10. Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. Journal of Artificial Intelligence Research, 4:237–285, 1996. URL: https://doi.org/10.1613/jair.301

11. Leonid Khachiyan and Lorant Porkolab. Computing integral points in convex semi-algebraic sets. In 38th Annual Symposium on Foundations of Computer Science, FOCS ’97, Miami.
Stefan Kiefer, Andrzej S. Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. On the complexity of equivalence and minimisation for q-weighted automata. *Logical Methods in Computer Science*, 9(1), 2013. URL: [https://doi.org/10.2168/LMCS-9(1:8)2013](https://doi.org/10.2168/LMCS-9(1:8)2013)

Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Assume-guarantee verification for probabilistic systems. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings, volume 6015 of *Lecture Notes in Computer Science*, pages 23–37. Springer, 2010. URL: [https://doi.org/10.1007/978-3-642-12002-2_3](https://doi.org/10.1007/978-3-642-12002-2_3)

Angus Macintyre and Alex J. Wilkie. On the decidability of the real exponential field. In Piergiorgio Odifreddi, editor, *Kreiseliana. About and Around Georg Kreisel*, pages 441–467. AK Peters, 1996.

Marvin Lee Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002. URL: [https://doi.org/10.1016/csla.2001.0184](https://doi.org/10.1016/csla.2001.0184)

Krishna V. Palem and Lingamneni Avinash. Ten years of building broken chips: The physics and engineering of inexact computing. *ACM Transactions on Embedded Computing Systems*, 12(2s):87:1–87:23, 2013. URL: [http://doi.acm.org/10.1145/2465787.2465789](http://doi.acm.org/10.1145/2465787.2465789)

Martin L. Puterman. *Markov Decision Processes*. Wiley-Interscience, 2005.

Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963. URL: [https://doi.org/10.1016/S0019-9958(63)90290-0](https://doi.org/10.1016/S0019-9958(63)90290-0)

Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6:223–231, 1978. URL: [https://doi.org/10.1016/0304-3975(78)90036-1](https://doi.org/10.1016/0304-3975(78)90036-1)

Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach* (3. internat. ed.). Pearson Education, 2010. URL: [http://vig.pearsoned.com/store/product/1,1207,store-12521_isbn-0136042597,00.html](http://vig.pearsoned.com/store/product/1,1207,store-12521_isbn-0136042597,00.html)

Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In 26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985, pages 327–338. IEEE Computer Society, 1985. URL: [https://doi.org/10.1109/SFCS.1985.12](https://doi.org/10.1109/SFCS.1985.12)

Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theoretical Computer Science*, 88(2):325–349, 1991. URL: [https://doi.org/10.1016/0304-3975(91)90381-B](https://doi.org/10.1016/0304-3975(91)90381-B)

Abuzer Yakaryılmaz and A. C. Cem Say. Unbounded-error quantum computation with small space bounds. *Information and Computation*, 209(6):873–892, 2011. URL: [https://doi.org/10.1016/j.ic.2011.01.008](https://doi.org/10.1016/j.ic.2011.01.008)