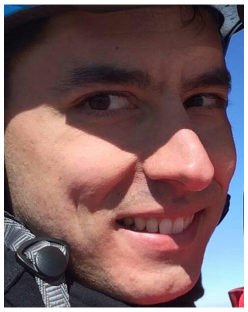
Undecidability Results for Probabilistic Automata

Nathanaël Fijalkow,
Alan Turing Institute of Data Science and University of Warwick, UK



The model of probabilistic automata was introduced by Rabin in 1963. Ever since, undecidability results were obtained for this model, showing that although simple, it is very expressive. This paper provides streamlined constructions implying the most important negative results, including the celebrated inapproximability result of Condon and Lipton.

1. INTRODUCTION AND DEFINITIONS

By way of introducing the model of *probabilistic automata* defined by Rabin [Rabin 1963], we highlight its characteristic features. As a starting point, we model processes:

— with *finitely many states*, each of them representing a configuration of the process,
— evolving at *discrete time steps*, meaning that one transition is fired at each time unit leading from a state to another,
— with *probabilistic behaviour*, *i.e.* the choice of transition follows a fixed probabilistic distribution.

We let Q denote the finite set of states. A (probabilistic) *distribution* over Q is a function $\delta : Q \to [0,1]$ such that $\sum_{q \in Q} \delta(q) = 1$. The set of distributions over Q is denoted $\mathcal{D}(Q)$.

Reactive and generative processes. An important distinction to be made for probabilistic processes is *reactive* versus *generative* processes [van Glabbeek et al. 1995], which arises when the transitions are labelled by symbols from a finite set Σ. In the reactive model [Pnueli 1985], the symbols from Σ are *inputs* to which the process reacts; Milner describes reactive models using the mechanistic metaphor of pushing buttons [Milner 1980]. In the generative model, the symbols from Σ are *outputs* which are observed.

We are in this paper interested in *reactive* probabilistic automata, sometimes called Rabin probabilistic automata [Rabin 1963]. In this setting the transition function is defined by:

$$\Delta : Q \times \Sigma \to \mathcal{D}(Q),$$

which reads: from a state q and an input letter a in Σ, the probability to reach p is $\Delta(q,a)(p)$.

For the sake of comparison, we mention *generative* probabilistic automata, commonly called Segala automata [Segala and Lynch 1995] (note that Segala automata usually combine probabilistic and non-deterministic behaviour). In this setting the transition function is defined by:

$$\Delta : Q \to \mathcal{D}(Q \times \Sigma),$$

which reads: from a state q, the probability to reach p and output the letter a in Σ is $\Delta(q)(p,a)$. Although syntactically close, reactive and generative automata behave in a very different way, and the problems studied here for reactive probabilistic automata do not make much sense for Segala automata.

For the remainder of this paper by probabilistic automata we mean reactive probabilistic automata.

Definitions. A *probabilistic automaton* is a tuple $\mathcal{A} = (Q, q_{in}, \Delta, F)$, where q_{in} is the initial state, $\Delta : Q \times \Sigma \to \mathcal{D}(Q)$ is the transition function, and F is the set of accepting states.

Given a word $w = a_1 \cdots a_n$, a run ρ over w is a sequence of states $q_0, q_1, \ldots, q_n$. The probability of such a run is

$$\mathcal{A}(\rho) = \prod_{\ell \in \{1,\ldots,n\}} \Delta(q_{\ell-1}, a_\ell)(q_\ell).$$

We let $\mathrm{Run}_{\mathcal{A}}(p \xrightarrow{w} q)$ denote the set of runs ρ over w starting in p and finishing in q with $\mathcal{A}(\rho) > 0$. The number $\mathcal{A}(p \xrightarrow{w} q)$ is the probability to go from p to q reading w, defined as the sum of the probabilities of its runs:

$$\mathcal{A}(p \xrightarrow{w} q) = \sum_{\rho \in \mathrm{Run}_{\mathcal{A}}(p \xrightarrow{w} q)} \mathcal{A}(\rho).$$

A run ρ is *accepting* if it starts in q_{in} and finishes in an accepting state, *i.e.* a state in F. We let $\mathrm{Run}_{\mathcal{A}}(w)$ denote the set of accepting runs over w. The *probability* of w over $\mathcal{A}$ is defined as the sum of the probabilities of its accepting runs:

$$\mathcal{A}(w) = \sum_{\rho \in \mathrm{Run}_{\mathcal{A}}(w)} \mathcal{A}(\rho).$$

Algorithmic analysis of probabilistic automata. A large part of the literature on probabilistic automata is about constructing algorithms for determining the properties of the function

$$\mathcal{A} : \Sigma^* \to [0, 1].$$

for a probabilistic automaton $\mathcal{A}$ given as input. The properties of interest depend on which of the two main views one adopts on probabilistic automata: either as an automaton model, or as a subclass of partially observable Markov decision processes (POMDP).

Probabilistic automata in automata theory. Probabilistic automata are naturally rooted in automata theory. For instance, weighted automata over the semiring $(\mathbb{R}, +, \times)$ are essentially probabilistic automata. Interestingly, Schützenberger developed the framework of weighted automata in 1961 [Schützenberger 1961], before Rabin introduced the subcase of probabilistic automata.

With a formal language theory approach, Rabin [Rabin 1963] defines the *threshold language* induced by a probabilistic automaton $\mathcal{A}$ and a threshold c as:

$$L^{\geq c}(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A}(w) \geq c\}.$$

The *emptiness problem* asks, given a probabilistic automaton $\mathcal{A}$ and a threshold c, whether the language $L^{\geq c}(\mathcal{A})$ is non-empty, *i.e.* whether there exists a word w such that $\mathcal{A}(w) \geq c$. The dual problem is the *universality problem*: given a probabilistic automaton $\mathcal{A}$ and a threshold c, is it true that for all words w we have $\mathcal{A}(w) \geq c$?

Probabilistic automata as partially observable Markov decision processes. Probabilistic automata form the subclass of blind POMDP where the controller has no observation at all on the evolution of the system. Indeed in this setting a strategy reduces to a single word. This game-theoretic interpretation yields the notion of the *value* of a probabilistic automaton $\mathcal{A}$:

$$\mathrm{val}(\mathcal{A}) = \sup_{w \in \Sigma^*} \mathcal{A}(w).$$

The natural questions here are to compute, approximate, or compare the value to a given threshold.

One undecidability result. Many negative results are known: all the problems mentioned above are undecidable, as well as many variants. The aim of this paper is to give one result implying the most important undecidability results, and a simple self-contained proof for it.

More specifically, the following theorem subsumes the undecidability of the problems mentioned below: the emptiness problem (1), the universality problem (2), the isolation problem (3), the Condon-Lipton approximation problem (4), and the value 1 problem (5).

THEOREM 1.1. *There exists no algorithm such that: given a probabilistic automaton $\mathcal{A}$,*

— *if val$(\mathcal{A}) = 1$, then the algorithm outputs "Yes",*
— *if val$(\mathcal{A}) \leq \frac{1}{2}$, then the algorithm outputs "No".*

The value $\frac{1}{2}$ will appear in the constructions, but of course it can easily be replaced by any other value between 0 and 1. It is important to remark that the algorithm is not completely specified: if $\mathrm{val}(\mathcal{A}) \in \left(\frac{1}{2}, 1\right)$, then the algorithm can do anything, including not terminating.

Corollaries. We discuss the most important undecidability results from the literature, which are all implied by Theorem 1.1 as special cases. The first two questions to be considered were the emptiness and universality problems described above, which were shown undecidable by Paz [Paz 1971]:

$$\exists w \in \Sigma^*,\ \mathcal{A}(w) \geq c, \tag{1}$$

$$\forall w \in \Sigma^*,\ \mathcal{A}(w) \geq c. \tag{2}$$

Almost equivalently, one may ask whether $\mathrm{val}(\mathcal{A}) \geq c$ or $\mathrm{val}(\mathcal{A}) \leq c$, which are undecidable as well.

One may think that the reason for undecidability is that words may get arbitrarily close to the threshold c. In this direction, Rabin proved that if the threshold c is isolated, *i.e.* if there exists $\varepsilon > 0$ such that for all words w, we have $|\mathcal{A}(w) - c| \geq \varepsilon$, then $L^{\geq c}(\mathcal{A})$ is regular [Rabin 1963]. Determining whether a fixed threshold c is isolated was proved undecidable by Bertoni [Bertoni et al. 1977]:

$$\exists \varepsilon > 0, \forall w \in \Sigma^*,\ |\mathcal{A}(w) - c| \geq \varepsilon. \tag{3}$$

The seminal result of Condon and Lipton [Condon and Lipton 1989] pushes this even further, by showing that one cannot even approximate the value. The statement is the following: given $0 < \alpha < \beta < 1$, there exists no algorithm such that given a probabilistic automaton $\mathcal{A}$,

$$\begin{cases} \text{if } \mathrm{val}(\mathcal{A}) \geq \beta, & \text{then the algorithm outputs "Yes"}, \\ \text{if } \mathrm{val}(\mathcal{A}) \leq \alpha, & \text{then the algorithm outputs "No"}. \end{cases} \tag{4}$$

The equivalent original formulation was as a promise problem: we assume that the input satisfies either $\mathrm{val}(\mathcal{A}) \geq \beta$ or $\mathrm{val}(\mathcal{A}) \leq \alpha$, and we want to determine which one it is.

Gimbert and Oualhadj [Gimbert and Oualhadj 2010] showed that one does not recover decidability by replacing the threshold c by 1, *i.e.* asking a qualitative question rather than a quantitative one. More specifically, the value 1 problem is undecidable: given a probabilistic automaton $\mathcal{A}$, determine whether $\text{val}(\mathcal{A}) = 1$, or equivalently:

$$\forall \varepsilon > 0, \exists w \in \Sigma^*,\ \mathcal{A}(w) \geq 1 - \varepsilon. \tag{5}$$

Positive Results

Not all is dark and gloomy for probabilistic automata. Indeed, an undecidability result is merely an invitation to refine the model and to find decidable subclasses.

Several positive results were obtained for the problems discussed above. For instance, algorithms were constructed for the emptiness problem and value approximation for hierarchical automata [Chadha et al. 2011; 2013; Chadha et al. 2015] and automata of bounded ambiguity [Fijalkow et al. 2017], or for the value 1 problem for leaktight automata [Gimbert and Oualhadj 2010; Chatterjee and Tracol 2012; Fijalkow et al. 2012; Fijalkow et al. 2015; Fijalkow 2016; 2017]. A remarkable example is the equivalence problem, which asks whether two probabilistic automata define the same function. It was proved to be decidable in polynomial time by Schützenberger [Schützenberger 1961], and later by Tzeng [Tzeng 1992]. Recently, this problem has been further analysed, leading to very efficient randomised algorithms with applications to software verification [Kiefer et al. 2011; 2013].

We study in this paper finite probabilistic automata over finite words. Extensions to infinite words have been studied by Baier, Bertrand, and Grösser [Baier et al. 2012], see also [Chadha et al. 2011], and then to infinite trees [Carayol et al. 2014]. Extensions to infinite probabilistic automata have also been considered, for instance pushdown versions [Brázdil et al. 2013; Brázdil et al. 2014] and timed versions [Bertrand et al. 2014].

2. EXAMPLES AND CONSTRUCTIONS

The binary value automaton

The very first automaton given as an example in the paper by Rabin [Rabin 1963] introducing probabilistic automata recognises the binary function, *i.e.* it computes the value of a rational number given in binary with least significant digit on the left:

$$\mathbf{bin}^R(a_1 \cdots a_n) = \sum_{i=1}^{n} \frac{a_i}{2^{n-i+1}}.$$

We present on the left-hand side of Figure 1 a simpler automaton computing the same function but *reversing* the input:

$$\mathbf{bin}(a_1 \cdots a_n) = \sum_{i=1}^{n} \frac{a_i}{2^i}.$$

The expanding automaton

The automaton presented on the right-hand side of Figure 1 was used in the proof of the undecidability of the value 1 problem [Gimbert and Oualhadj 2010]. The alphabet is $\Sigma = \{\texttt{check}, \texttt{sim}\}$, the initial state is q_0 and the unique accepting state is $\top$. The choice of names for the two letters will make more sense when using this automaton in the proof of Theorem 1.1.

In this informal explanation, we are computing the value of $\mathcal{A}$, so we are looking for words maximising the probability to be accepted by $\mathcal{A}$. After reading one `check`,

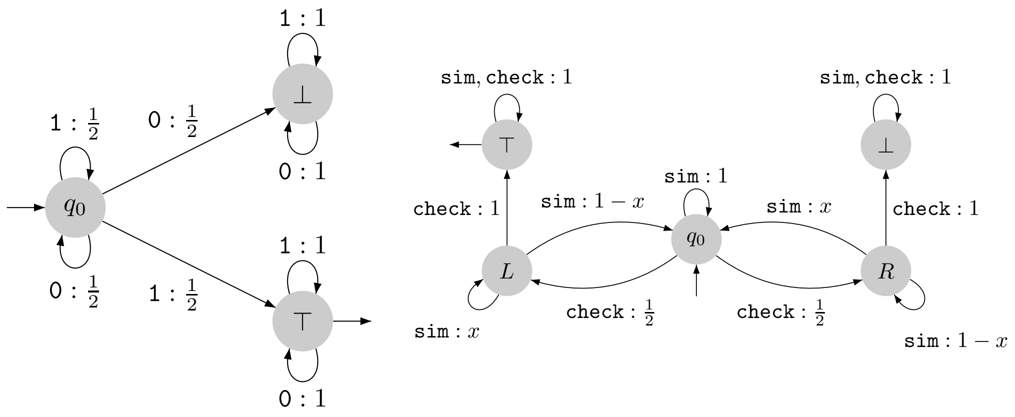


Fig. 1. The probabilistic automaton on the left-hand side computes the function **bin**. The probabilistic automaton on the right-hand side has value $\frac{1}{2}$ if $x \leq \frac{1}{2}$, and has value 1 if $x > \frac{1}{2}$.

the distribution is uniform over L, R. To reach $\top$, one needs to read a `check` from the state L, but on the right-hand side this leads to the non-accepting absorbing state $\perp$. In order to maximise the probability to reach $\top$, one tries to "tip the scales" to the left, which can only be achieved by reading `sim` a certain number of times. If $x \leq \frac{1}{2}$, there is no hope to achieve this: reading a letter `sim` gives more chance to stay in R than in L thus all words are accepted with probability at most $\frac{1}{2}$, and $\text{val}(\mathcal{A}) = \frac{1}{2}$. However, if $x > \frac{1}{2}$ then we show that $\mathcal{A}$ has value 1.

We have:

$$\mathcal{A}(q_0 \xrightarrow{\texttt{check}\ \texttt{sim}^n} L) = \frac{1}{2} \cdot x^n \qquad \text{and} \qquad \mathcal{A}(q_0 \xrightarrow{\texttt{check}\ \texttt{sim}^n} R) = \frac{1}{2} \cdot (1-x)^n.$$

We fix an integer N and analyse the action of reading $(\texttt{check}\ \texttt{sim}^n)^N$: there are N "rounds", each of them corresponding to reading $\texttt{check}\ \texttt{sim}^n$ from q_0. In a round, there are three outcomes: winning (that is, remaining in L) with probability $p_n = \frac{1}{2} \cdot x^n$, losing (that is, remaining in R) with probability $q_n = \frac{1}{2} \cdot (1-x)^n$, or going to the next round (that is, reaching q_0) with probability $1 - (p_n + q_n)$. If a round is won or lost, then the next `check` leads to an accepting or rejecting sink; otherwise it goes on to the next round, for N rounds. Hence:

$$\begin{aligned}
\mathcal{A}((\texttt{check}\ \texttt{sim}^n)^N) &= \textstyle\sum_{k=1}^{N-1} (1-(p_n+q_n))^{k-1} \cdot p_n \\
&= p_n \cdot \tfrac{1-(1-(p_n+q_n))^{N-1}}{1-(1-(p_n+q_n))} \\
&= \tfrac{1}{1+\frac{q_n}{p_n}} \cdot \left(1 - (1-(p_n+q_n))^{N-1}\right)
\end{aligned}$$

We now set $N = 2^n$ and assume $x > \frac{1}{2}$. A simple calculation shows that the sequence $((1-(p_n+q_n))^{2^n-1})_{n \in \mathbb{N}}$ converges to 0 as n goes to infinity. Furthermore, $\frac{1-x}{x} < 1$, so $\frac{q_n}{p_n} = \left(\frac{1-x}{x}\right)^n$ converges to 0 as n goes to infinity. It follows that the acceptance probability converges to 1 as n goes to infinity:

$$\lim_n \mathcal{A}((\texttt{check}\ \texttt{sim}^n)^{2^n}) = 1.$$

Simple constructions

We conclude this section by discussing three simple constructions for probabilistic automata.

Complementation. Given a probabilistic automaton $\mathcal{A}$, we can construct a probabilistic automaton computing the function $1 - \mathcal{A}$. This is achieved by switching accepting and non-accepting states.

Convex combinations. Given two probabilistic automata $\mathcal{A}$ and $\mathcal{B}$, we can construct a probabilistic automaton computing the function $\frac{1}{2}\mathcal{A}+\frac{1}{2}\mathcal{B}$. Indeed, we consider the union of the two automata, and add a new state which is initial and leads with probability $\frac{1}{2}$ to each automata. This construction generalises to arbitrary convex combinations.

Products. Given two probabilistic automata $\mathcal{A}$ and $\mathcal{B}$, we can construct a probabilistic automaton computing the function $\mathcal{A} \cdot \mathcal{B}$. To this end we consider the synchronised product of the two automata, with a pair of states being accepting if both states are accepting.

3. PROOFS

The proof of Theorem 1.1 is obtained by a reduction from the emptiness problem. For the sake of completeness, we first briefly recall the proof of undecidability for the emptiness problem.

Undecidability of the emptiness problem

Gimbert and Oualhadj [Gimbert and Oualhadj 2010] gave a simple exposition of the undecidability proof of Bertoni [Bertoni 1974] for the emptiness problem. The construction is based on the automaton computing the function $\mathbf{bin}$ given in Section 2.

— First, show that the equality problem is undecidable: given a probabilistic automaton $\mathcal{A}$, does there exist a word w such that $\mathcal{A}(w) = \frac{1}{2}$? The proof is by reduction from Post's Correspondence Problem (PCP), which can be defined as follows: given a pair of monoid homomorphisms $\varphi_1, \varphi_2 : \Sigma^* \to \{0,1\}^*$, does there exist a non-empty word w such that $\varphi_1(w) = \varphi_2(w)$? Using the automaton above computing $\mathbf{bin}$, complementation and convex combinations, we construct a probabilistic automaton $\mathcal{A}$ such that $\mathcal{A}(w) = \frac{1}{2}\mathbf{bin}(\varphi_1(w)) + \frac{1}{2}(1 - \mathbf{bin}(\varphi_2(w)))$. Since the function $\mathbf{bin}$ is (essentially[1]) injective, $\mathcal{A}(w) = \frac{1}{2}$ is equivalent to $\varphi_1(w) = \varphi_2(w)$, proving the correctness of the reduction.

— Second, show that the emptiness problem is undecidable by reduction to the equality problem above. Given a probabilistic automaton $\mathcal{A}$, one can construct a probabilistic automaton $\mathcal{A}'$ such that $\mathcal{A}'(w) = \mathcal{A}(w) \cdot (1 - \mathcal{A}(w))$. This is achieved by constructing a complement and a product as described above. Since for x in $[0, 1]$, the following equivalence holds: $x = \frac{1}{2}$ if, and only if, $x \cdot (1 - x) \geq \frac{1}{4}$, the first undecidability result implies the undecidability of the emptiness problem.

Note that this proves that the emptiness problem is undecidable with non-strict inequalities: $\mathcal{A}(w) \geq \frac{1}{4}$. (It is easy to replace the constant $\frac{1}{4}$ by any constant in $(0, 1)$.) However, since the equality problem is undecidable, this implies that the emptiness problem with strict inequalities is also undecidable.

Proof of Theorem 1.1

We now prove Theorem 1.1 by constructing a reduction from the emptiness problem. The construction is essentially the same as for the undecidability of the value 1 prob-

[1]One needs to ensure that the last letter is a 1, which is achieved using a small technical twist.

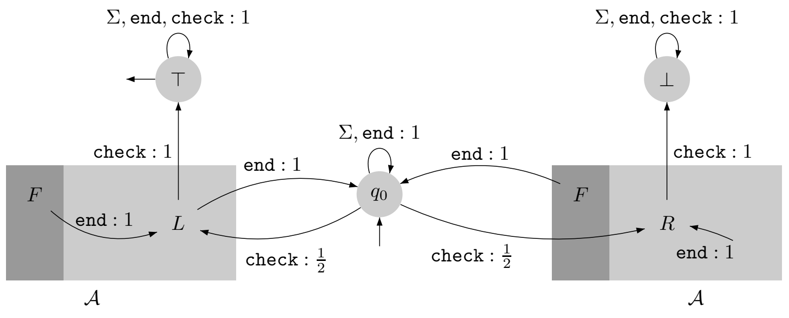


Fig. 2. The automaton $\mathcal{B}$ constructed in the reduction.

lem by Gimbert and Oualhadj [Gimbert and Oualhadj 2010], the main improvements are in the correctness proof. Let $\mathcal{A}$ be a probabilistic automaton on the alphabet Σ, we construct a probabilistic automaton $\mathcal{B}$ on the alphabet $\Sigma \cup \{\texttt{check}, \texttt{end}\}$ such that:

$$\text{if there exists a word } w \text{ such that } \mathcal{A}(w) > \tfrac{1}{2}, \text{ then val}(\mathcal{B}) = 1,$$
$$\text{otherwise for all words } w \text{ we have } \mathcal{A}(w) \leq \tfrac{1}{2}, \text{ and then val}(\mathcal{B}) \leq \tfrac{1}{2}.$$

The reduction is illustrated in Figure 2. We start from the expanding automaton described in Section 1, and substitute the transitions for the letter sim by a simulation of $\mathcal{A}$ on a word w. In the expanding automaton, reading sim from L has two outcomes: staying in L with probability x, and going to q_0 with probability $1-x$. The probabilistic automaton $\mathcal{B}$ mimics this behaviour: reading w end from L has two outcomes, staying in L with probability $\mathcal{A}(w)$, and going to q_0 with probability $1 - \mathcal{A}(w)$.

Hence the analysis of the expanding automaton $\mathcal{A}$ can be repeated mutatis mutandis for $\mathcal{B}$. Assume that there exists a word w such that $\mathcal{A}(w) > \frac{1}{2}$, then $\lim_n \mathcal{B}((\texttt{check} \cdot (w \cdot \texttt{end})^n)^{2^n}) = 1$, so val$(\mathcal{B}) = 1$. Conversely, assume that for all words w, we have $\mathcal{A}(w) \leq \frac{1}{2}$, then every finite word is accepted by $\mathcal{B}$ with probability at most $\frac{1}{2}$. The words accepted with non-zero probability are concatenations of words of the form

$$w = \texttt{check} \cdot w_1 \cdot \texttt{end} \cdot w_2 \cdot \texttt{end} \cdot \cdots \; w_n \cdot \texttt{end},$$

with $w_i \in \Sigma^*$. Since $\mathcal{A}(w_i) \leq \frac{1}{2}$ for every i, it follows that in $\mathcal{B}$, after reading w, the probability to be in L is smaller than or equal to the probability to be in R. It follows that the value of $\mathcal{B}$ is at most $\frac{1}{2}$.

We now conclude the proof of Theorem 1.1. Assume towards contradiction that there exists an algorithm A as stated in Theorem 1.1, we show how to use the reduction above to construct an algorithm for the emptiness problem. Let $\mathcal{A}$ be a probabilistic automaton given as input, we construct $\mathcal{B}$ following the reduction above, and run the algorithm A on $\mathcal{B}$. By construction, the probabilistic automaton $\mathcal{B}$ either has value 1 or value at most $\frac{1}{2}$, so the algorithm A terminates on input $\mathcal{B}$. Moreover, $\mathcal{B}$ has value 1 if, and only if, $\mathcal{A}$ is non-empty, so the answer given by algorithm A solves the emptiness problem for $\mathcal{A}$, contradicting the undecidability of the emptiness problem.

REFERENCES

Christel Baier, Nathalie Bertrand, and Marcus Größer. 2012. Probabilistic ω-automata. *J. ACM* 59, 1 (2012).

Alberto Bertoni. 1974. The Solution of Problems Relative to Probabilistic Automata in the Frame of the Formal Languages Theory. In *GI Jahrestagung*.

Alberto Bertoni, Giancarlo Mauri, and Mauro Torelli. 1977. Some Recursive Unsolvable Problems Relating to Isolated Cutpoints in Probabilistic Automata. In *ICALP*.

Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, Quentin Menet, Christel Baier, Marcus Größer, and Marcin Jurdziński. 2014. Stochastic Timed Automata. *Logical Methods in Computer Science* 10, 4 (2014).

Tomás Brázdil, Václav Brozek, Vojtech Forejt, and Antonín Kucera. 2014. Branching-time model-checking of probabilistic pushdown automata. *Journal of Computer Science Systems* 80, 1 (2014), 139–156.

Tomás Brázdil, Javier Esparza, Stefan Kiefer, and Antonín Kucera. 2013. Analyzing probabilistic pushdown automata. *Formal Methods in System Design* 43, 2 (2013), 124–163.

Arnaud Carayol, Axel Haddad, and Olivier Serre. 2014. Randomization in Automata on Infinite Trees. *ACM Transactions on Computational Logics* 15, 3 (2014), 24:1–24:33.

Rohit Chadha, A. Prasad Sistla, and Mahesh Viswanathan. 2011. Power of Randomization in Automata on Infinite Strings. *Logical Methods in Computer Science* 7, 3 (2011).

Rohit Chadha, A. Prasad Sistla, and Mahesh Viswanathan. 2013. Probabilistic Automata with Isolated Cut-Points. In *MFCS*.

Rohit Chadha, A. Prasad Sistla, Mahesh Viswanathan, and Yue Ben. 2015. Decidable and Expressive Classes of Probabilistic Automata. In *FoSSaCS*.

Krishnendu Chatterjee and Mathieu Tracol. 2012. Decidable Problems for Probabilistic Automata on Infinite Words. In *LICS*.

Anne Condon and Richard J. Lipton. 1989. On the Complexity of Space Bounded Interactive Proofs (Extended Abstract). In *FOCS*.

Nathanaël Fijalkow. 2016. Characterisation of an Algebraic Algorithm for Probabilistic Automata. In *STACS*.

Nathanaël Fijalkow. 2017. Profinite techniques for probabilistic automata and the Markov Monoid algorithm. *Theoretical Computer Science* 680 (2017), 1–14.

Nathanaël Fijalkow, Hugo Gimbert, Edon Kelmendi, and Youssouf Oualhadj. 2015. Deciding the value 1 Problem for Probabilistic Leaktight Automata. *Logical Methods in Computer Science* 11, 1 (2015).

Nathanaël Fijalkow, Hugo Gimbert, and Youssouf Oualhadj. 2012. Deciding the Value 1 Problem for Probabilistic Leaktight Automata. In *LICS*.

Nathanaël Fijalkow, Cristian Riveros, and James Worrell. 2017. Probabilistic Automata of Bounded Ambiguity. In *CONCUR*.

Hugo Gimbert and Youssouf Oualhadj. 2010. Probabilistic Automata on Finite Words: Decidable and Undecidable Problems. In *ICALP*.

Stefan Kiefer, Andrzej S. Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. 2011. Language Equivalence for Probabilistic Automata. In *CAV*.

Stefan Kiefer, Andrzej S. Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. 2013. On the Complexity of Equivalence and Minimisation for Q-weighted Automata. *Logical Methods in Computer Science* 9, 1 (2013).

Robin Milner. 1980. *A Calculus of Communicating Systems*. Lecture Notes in Computer Science, Vol. 92. Springer.

Azaria Paz. 1971. *Introduction to probabilistic automata*. Academic Press.

Amir Pnueli. 1985. Linear and Branching Structures in the Semantics and Logics of Reactive Systems. In *ICALP*.

Michael O. Rabin. 1963. Probabilistic automata. *Information and Control* 6(3) (1963), 230–245.

Marcel-Paul Schützenberger. 1961. On the definition of a family of automata. *Information and Control* 4 (1961).

Roberto Segala and Nancy A. Lynch. 1995. Probabilistic Simulations for Probabilistic Processes. *Nordic Journal of Computing* 2, 2 (1995), 250–273.

Wen-Guey Tzeng. 1992. A Polynomial-Time Algorithm for the Equivalence of Probabilistic Automata. *SIAM J. Comput.* 21(2) (1992), 216–227.

Rob J. van Glabbeek, Scott A. Smolka, and Bernhard Steffen. 1995. Reactive, Generative and Stratified Models of Probabilistic Processes. *Information and Computation* 121, 1 (1995), 59–80.