Pushing undecidability of the isolation problem for probabilistic automata

Nathanaël Fijalkow, Hugo Gimbert, Youssouf Oualhadj

Pushing undecidability of the isolation problem for probabilistic automata

Nathanaël Fijalkow[1], Hugo Gimbert[2] and Youssouf Oualhadj[3]

[1] LIAFA, CNRS & Université Denis Diderot - Paris 7, France
nath@liafa.jussieu.fr
[2] LaBRI, CNRS, France
hugo.gimbert@labri.fr
[3] LaBRI, Université Bordeaux 1, France
youssouf.oualhadj@labri.fr

Abstract. This short note aims at proving that the isolation problem is undecidable for probabilistic automata with only one probabilistic transition. This problem is known to be undecidable for general probabilistic automata, without restriction on the number of probabilistic transitions. In this note, we develop a simulation technique that allows to simulate any probabilistic automaton with one having only one probabilistic transition.

1 Introduction

Probabilistic automata. Rabin introduced probabilistic automata over finite words as a natural and simple computation model [Rab63]. A probabilistic automaton can be thought as a non-deterministic automaton, where non-deterministic transitions are chosen according to a fixed probabilistic distribution. Probabilistic automata drew attention and have been extensively studied (see [Buk80] for a survey).
The isolation problem. However, on the algorithmic side, most of the results are undecidability results. The isolation problem asks, given some probability $0 \leq \lambda \leq 1$, whether there exists words accepted with probability arbitrarily close to λ. Bertoni showed that this problem is undecidable [Ber74,BMT77].
Contribution. In this note, we prove that the isolation problem is undecidable, even for probabilistic automata having only one probabilistic transition. To do this, we develop a simulation technique that allows to simulate any probabilistic automaton with one having only one probabilistic transition.
Outline. Section 2 is devoted to definitions. In section 3, we develop a simulation technique, which allows to simulate any probabilistic automaton with one having only one probabilistic transition. Using this technique we show that the isolation problem is undecidable for this very restricted class of automata.

2 Definitions

Given a finite set of states Q, a probability distribution (distribution for short) over Q is a row vector δ of size $|Q|$ with rational entries in $[0, 1]$ such that

$\sum_{q\in Q}\delta(q)=1$. We denote by δ_q the distribution such that $\delta_q(q')=1$ if $q'=q$ and 0 otherwise. A probabilistic transition matrix M is a square matrix of size $|Q|\times|Q|$, such that for a state s, $M_a(s,\_)$ is a distribution over Q.

Definition 1 (Probabilistic automaton). *A probabilistic automaton is a tuple $\mathcal{A}=(Q,A,(M_a)_{a\in A},q_0,F)$, where Q is a finite set of states, A is the finite input alphabet, $(M_a)_{a\in A}$ are the probabilistic transition matrices, q_0 is the initial state and F is the set of accepting states.*

For each letter $a\in A$, $M_a(s,t)$ is the probability to go from state s to state t when reading letter a. A probabilistic transition is a couple (s,a) such that $M_a(s,t)\notin\{0,1\}$ for some t.

A probabilistic automaton is said *simple* if for all a, for all states s and t, we have $M_a(s,t)\in\{0,\frac{1}{2},1\}$.

Given an initial distribution δ and an input word w, we define $\delta\cdot w$ by induction on w: we have $\delta\cdot\varepsilon=\delta$, then for a letter a in A, we have $\delta\cdot a=M_a\cdot\delta$ and if $w=v\cdot a$, then $\delta\cdot(v\cdot a)=(\delta\cdot v)\cdot a$.

We denote by $\mathbb{P}_{\mathcal{A}}(s\xrightarrow{w}T)$ the probability to reach the set T from state s when reading the word w, that is $\sum_{t\in T}(\delta_s\cdot w)(t)$.

Definition 2 (Value and acceptance probability). *The* acceptance probability *of a word $w\in A^*$ by $\mathcal{A}$ is $\mathbb{P}_{\mathcal{A}}(w)=\mathbb{P}_{\mathcal{A}}(q_0\xrightarrow{w}F)$. The* value *of $\mathcal{A}$, denoted val($\mathcal{A}$), is the supremum acceptance probability: val($\mathcal{A}$) $=\sup_{w\in A^*}\mathbb{P}_{\mathcal{A}}(w)$.*

3 Simulation with one probabilistic transition

We first show how to simulate a probabilistic automaton with one having only one probabilistic transition, *up to a regular language*:

Proposition 1. *For any simple probabilistic automata $\mathcal{A}=(Q,A,(M_a)_{a\in A},q_0,F)$, there exists a simple probabilistic automaton $\mathcal{B}$ over a new alphabet B, with one probabilistic transition, and a morphism $\widehat{\_}\colon A^*\mapsto B^*$ such that:*

$$\forall w\in A^*,\mathbb{P}_{\mathcal{A}}(w)=\mathbb{P}_{\mathcal{B}}(\widehat{w}).$$

The morphism $\widehat{\_}$ will not be onto, so this simulation works up to the regular language $\{\widehat{w}\mid w\in A^*\}$. We shall see that the automaton $\mathcal{B}$ will not be able to check that a word read belongs to this language, which makes this restriction unavoidable in this construction.

We first give the intuitions behind the construction. Intuitively, while reading the word w, the probabilistic automaton $\mathcal{A}$ "throw parallel threads". A computation of $\mathcal{A}$ over w can be viewed as a tree, where probabilistic transitions correspond to branching nodes.

On the figure, reading a from q_0 or b from q_1 leads deterministically to the next state. Reading b from q_2 leads at random to r or to s, hence the corresponding node is branching. Our interpretation is that two parallel threads are thrown. Let us make two observations:

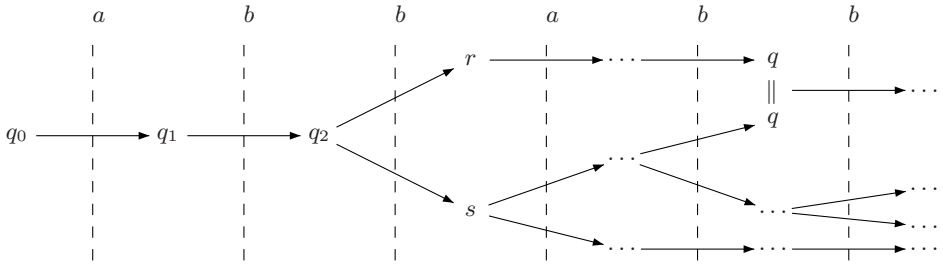


Fig. 1. An example of a computation

- threads are not synchronised: reading the fourth letter (an a), the first thread leads deterministically to the next state, while the second thread randomizes;
- threads are merged so there are at most $n = |Q|$ parallel threads: whenever two threads synchronize to the same state q, they are merged. This happens in the figure after reading the fifth letter (b).

The automaton $\mathcal{B}$ we construct will simulate the n threads from the beginning, and take care of the merging process each step.

Proof. We denote by q_i the states of $\mathcal{A}$, *i.e* $Q = \{q_0, \dots, q_{n-1}\}$. The alphabet B is made of two new letters '$*$' and 'merge' plus, for each letter $a \in A$ and state $q \in Q$, two new letters $\text{check}(a, q)$ and $\text{apply}(a, q)$, so that:

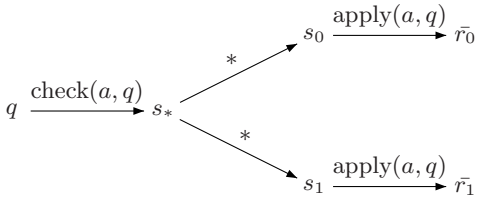
$$B = \{*, \text{merge}\} \cup \bigcup_{a \in A, q \in Q} \{\text{check}(a, q), \text{apply}(a, q)\}$$

We now define the automaton $\mathcal{B}$. We duplicate each state $q \in Q$, and denote the fresh copy by $\bar{q}$. Intuitively, $\bar{q}$ is a temporary state that will be merged at the next merging process. States in $\mathcal{B}$ are either a state from Q or its copy, or one of the three fresh states s_*, s_0 and s_1.

The initial state remains q_0 as well as the set of final states remains F.

The transitions of $\mathcal{B}$ are as follows:

- for every letter $a \in A$ and state $q \in Q$, the new letter $\text{check}(a, q)$ from state q leads deterministically to state s_* *i.e* $M_{\text{check}(a,q)}(q) = s_*$,
- the new letter $*$ from state s_* leads with probability half to s_0 and half to s_1, *i.e* $M_{s_*}(*) = \frac{1}{2}s_0 + \frac{1}{2}s_1$ (this is the only probabilistic transition of $\mathcal{B}$);
- the new letter $\text{apply}(a, q)$ from states s_0 and s_1 applies the transition function from q reading a: if the transition $M_a(q)$ is deterministic, *i.e* $M_a(q, r) = 1$ for some state r then $M_{\text{apply}(a,q)}(s_0) = \bar{r}$ and $M_{\text{apply}(a,q)}(s_1) = \bar{r}$, else the transition $M_a(q)$ is probabilistic *i.e* $M_a(q) = \frac{1}{2}r + \frac{1}{2}r'$ for some states r, r', then $M_{\text{apply}(a,q)}(s_0) = \bar{r}$ and $M_{\text{apply}(a,q)}(s_1) = \bar{r'}$;
- the new letter merge activates the merging process: it consists in replacing $\bar{q}$ by q for all $q \in Q$.



Whenever a couple (letter, state) does not fall in the previous cases, it has no effect. The gadget simulating a transition is illustrated in the figure.

Now we define the morphism $\widehat{\ }: A^* \mapsto B^*$ by its action on letters:

$$\widehat{a} = \text{check}(a, q_0) \cdot * \cdot \text{apply}(a, q_0) \ldots \text{check}(a, q_{n-1}) \cdot * \cdot \text{apply}(a, q_{n-1}) \cdot \text{merge}.$$

The computation of $\mathcal{A}$ while reading w in A^* is simulated by $\mathcal{B}$ on $\widehat{w}$, *i.e* we have:

$$\mathbb{P}_{\mathcal{A}}(w) = \mathbb{P}_{\mathcal{B}}(\widehat{w})$$

This completes the proof. □

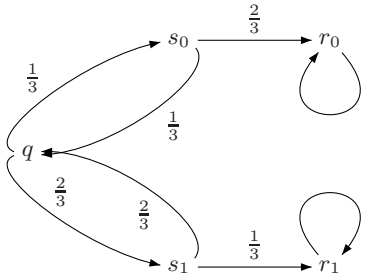
Let us remark that $\mathcal{B}$ is indeed unable to check that a letter check(a, q) is actually followed by the corresponding apply(a, q): inbetween, it will go through s_* and "forget" the state it was in.

We now improve the above construction: we get rid of the regular external condition. To this end, we will use probabilistic automata whose transitions have probabilities 0, $\frac{1}{3}$, $\frac{2}{3}$ or 1. This is no restriction, as stated in the following lemma:

Lemma 1. *For any simple probabilistic automata $\mathcal{A} = (Q, A, (M_a)_{a\in A}, q_0, F)$, there exists a probabilistic automaton $\mathcal{B}$ whose transitions have probabilities* 0, $\frac{1}{3}$, $\frac{2}{3}$ *or* 1*, such that for all w in A^*, we have:*

$$\mathit{val}(\mathcal{A}) = \mathit{val}(\mathcal{B}).$$

Proof. We provide a construction to pick with probability half, using transitions with probability 0, $\frac{1}{3}$, $\frac{2}{3}$ and 1. The construction is illustrated in the figure.



In this gadget, the only letter read is a fresh new letter ♯. The idea is the following: to pick with probability half r_0 or r_1, we sequentially pick with probability a third or two thirds. Whenever the two picks are different, if the first was

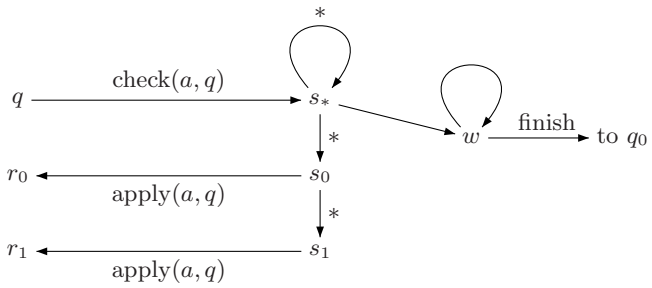
a third, then choose r_0, else choose r_1. This happens with probability half each. We easily see that $\mathbb{P}_{\mathcal{A}}(a_0 \cdot a_1 \cdot \ldots a_{k-1}) = \sup_p \mathbb{P}_{\mathcal{B}}(a_0 \cdot \sharp^p \cdot a_1 \cdot \sharp^p \ldots a_{k-1} \cdot \sharp^p)$. □

Proposition 2. *For any simple probabilistic automata $\mathcal{A} = (Q, A, (M_a)_{a\in A}, q_0, F)$, there exists a simple probabilistic automaton $\mathcal{B}$ over a new alphabet B, with one probabilistic transition, such that:*

$$val(\mathcal{A}) \geq \lambda \Leftrightarrow val(\mathcal{B}) \geq \lambda.$$

Thanks to the lemma, we assume that in $\mathcal{A}$, transitions have probabilities 0, $\frac{1}{3}$, $\frac{2}{3}$ or 1.

We first deal with the case where $\lambda = 1$. The new gadget used to simulate a transition is illustrated in the figure.



The automaton $\mathcal{B}$ reads words of the form $u_1 \cdot \text{finish} \cdot u_2 \cdot \text{finish} \ldots$, where 'finish' is a fresh new letter. The idea is to "skip", or "delay" part of the computation of $\mathcal{A}$: each time the automaton $\mathcal{B}$ reads a word u_i, it will be skipped with some probability.

Simulating a transition works as follows: whenever in state s_*, reading two times the letter '$*$' leads with probability half to s_1, quarter to s_0 and quarter to s. As before, from s_0 and s_1, we proceed with the simulation. However, in the last case, we "wait" for the next letter 'finish' that will restart from q_0. Thus each time a transition is simulated, the word being read is skipped with probability $\frac{1}{4}$.

Delaying part of the computation allows to multiply the number of threads. We will use the accepted threads to check the extra regular condition we had before. To this end, as soon as a simulated thread is accepted in $\mathcal{B}$, it will go through an automaton (denoted $\mathcal{C}$ in the construction) that checks the extra regular condition.

Proof. We keep the same notations. The alphabet B is made of three new letters: '$*$', 'merge' and 'finish' plus, for each letter $a \in A$ and state $q \in Q$, two new letters check(a, q) and apply(a, q), so that:

$$B = \{*, \text{merge}, \text{finish}\} \cup \bigcup_{a\in A, q\in Q} \{\text{check}(a, q), \text{apply}(a, q)\}$$

We first define a syntactic automaton $\mathcal{C}$. We define a morphism $\widehat{\_}: A^* \mapsto B^*$ by its action on letters:

$$\widehat{a} = \text{check}(a, q_0) \cdot * \cdot * \cdot \text{apply}(a, q_0) \ldots \text{check}(a, q_{n-1}) \cdot * \cdot * \cdot \text{apply}(a, q_{n-1}) \cdot \text{merge}.$$

Consider the regular language $L = \{\widehat{w} \cdot \text{finish} \mid w \in A^*\}^*$, and $\mathcal{C} = (Q_\mathcal{C}, \delta_\mathcal{C}, s_\mathcal{C}, F_\mathcal{C})$ an automaton recognizing it.

We now define the automaton $\mathcal{B}$. We duplicate each state $q \in Q$, and denote the fresh copy by $\bar{q}$. States in $\mathcal{B}$ are either a state from Q or its copy, a state from $Q_\mathcal{C}$ or one of the four fresh states s_*, s_0, s_1 and wait.

The initial state remains q_0, and the set of final states is $F_\mathcal{C}$.

The transitions of $\mathcal{B}$ are as follows:

- for every letter $a \in A$ and state $q \in Q$, the new letter $\text{check}(a, q)$ from state q leads deterministically to state s_* *i.e* $M_{\text{check}(a,q)}(q) = s_*$,
- the new letter $*$ from state s_* leads with probability half to s_* and half to s_0, *i.e* $M_{s_*}(*) = \frac{1}{2}s_* + \frac{1}{2}s_0$ (this is the only probabilistic transition of $\mathcal{B}$);
- any other letter from state s_* leads deterministically to w, *i.e* $M_{s_*}(\_) = \text{wait}$;
- the new letter $*$ from state s_0 leads deterministically to s_1, *i.e* $M_{s_0}(*) = s_1$;
- the new letter $\text{apply}(a, q)$ from states s_0 and s_1 applies the transition function from q reading a: if the transition $M_a(q)$ is deterministic, *i.e* $M_a(q, r) = 1$ for some state r then $M_{\text{apply}(a,q)}(s_0) = \bar{r}$ and $M_{\text{apply}(a,q)}(s_1) = \bar{r}$, else the transition $M_a(q)$ is probabilistic *i.e* $M_a(q) = \frac{1}{2}r + \frac{1}{2}r'$ for some states r, r', then $M_{\text{apply}(a,q)}(s_0) = \bar{r}$ and $M_{\text{apply}(a,q)}(s_1) = \bar{r'}$;
- the new letter merge activates the merging process: it consists in replacing $\bar{q}$ by q for all $q \in Q$;
- the new letter finish from state wait leads deterministically to q_0;
- the new letter finish from state q in F leads deterministically to $s_\mathcal{C}$;
- the new letter finish from any other state is not defined (there is a deterministic transition to a bottom non-accepting state).

Transitions in $\mathcal{C}$ are not modified. Whenever a couple (letter, state) does not fall in the previous cases, it has no effect.

We now show that this construction is correct.

We first prove that for all $w \in A^*$, there exists a sequence of words $(w_p)_{p \geq 1}$ such that $\mathbb{P}_\mathcal{A}(w) = \sup_p \mathbb{P}_\mathcal{B}(w_p)$.

We have, for δ a distribution over Q:

$$\delta_\mathcal{B}(\delta, \hat{a}) = \frac{3}{4}\delta_\mathcal{A}(\delta, a) + \frac{1}{4}\text{wait}.$$

It follows:

$$\delta_\mathcal{B}(\delta, \hat{w}) = \left(\frac{3}{4}\right)^k \delta_\mathcal{A}(\delta, w) + 1 - \left(\frac{3}{4}\right)^k \text{wait},$$

where $k = |w|$. Hence:

$$\delta_\mathcal{B}(q_0, \hat{w} \cdot \text{finish}) = \left(\frac{3}{4}\right)^k \mathbb{P}_\mathcal{A}(w) + 1 - \left(\frac{3}{4}\right)^k q_0.$$

The computation of $\mathcal{A}$ while reading w is simulated by $\mathcal{B}$ on $\widehat{w} \cdot \text{finish}$. This implies that $\sup_p \mathbb{P}_{\mathcal{B}}((\widehat{w} \cdot \text{finish})^p) = \mathbb{P}_{\mathcal{A}}(w)$, hence if $\text{val}(\mathcal{A}) = 1$, then $\text{val}(\mathcal{B}) = 1$.

Conversely, we prove that if $\text{val}(\mathcal{B}) = 1$, then $\text{val}(\mathcal{A}) = 1$. Let w a word read by $\mathcal{B}$ accepted with probability close to 1, we slice it as follows: $w = u_1 \cdot \text{finish} \cdot \ldots \cdot u_k \cdot \text{finish}$, such that u_i does not contain the letter finish. The key observation is that if $k = 1$, the word w is accepted with probability at most $\frac{3}{4}$. Hence we consider only the case $k > 1$. We assume without loss of generality that $\mathbb{P}_{\mathcal{A}}(u_1) > 0$ (otherwise we delete $u_1 \cdot \text{finish}$ and proceed). In this case, a thread has been thrown while reading u_1 that reached $s_{\mathcal{C}}$, so the syntactic process started: it follows that u_i for $i > 1$ are in the image of $\widehat{\_}$. This implies that the simulation is sound: from w we can recover a word in A^* accepted with probability arbitrarily close to 1 by $\mathcal{A}$.

The case where λ is any positive rational is handled similarly. We only need to ensure that the previous key observation still holds: a word of the form $u \cdot \text{finish}$ where u does not contain finish cannot be accepted with probability more than $\frac{3\lambda}{4}$. This is made possible by slightly modifying the simulation gadget, adding new intermediate states.

This completes the proof. □

We conclude:

Theorem 1. *The isolation problem is undecidable for simple automata with one probabilistic transition.*

References

Ber74. Alberto Bertoni. The solution of problems relative to probabilistic automata in the frame of the formal languages theory. In *GI Jahrestagung*, pages 107–112, 1974.

BMT77. Alberto Bertoni, Giancarlo Mauri, and Mauro Torelli. Some recursive unsolvable problems relating to isolated cutpoints in probabilistic automata. In *International Colloquium on Automata, Languages and Programming*, pages 87–94, 1977.

Buk80. R. G. Bukharaev. Probabilistic automata. *Journal of Mathematical Sciences*, 13(3):359–386, 1980.

Rab63. M. O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.