

# Two Recursively Inseparable Problems for Probabilistic Automata<sup>\*</sup>

Nathanaël Fijalkow<sup>1,2</sup>, Hugo Gimbert<sup>3</sup>, Florian Horn<sup>1</sup>, and Youssouf Oualhadj<sup>4</sup>

LIAFA, Université Paris 7, France,  
University of Warsaw, Poland,  
LaBRI, Université de Bordeaux, France,  
Université de Mons, Belgium.

**Abstract.** This paper introduces and investigates decision problems for *numberless* probabilistic automata, *i.e.* probabilistic automata where the *support* of each probabilistic transitions is specified, but the exact values of the probabilities are *not*. A numberless probabilistic automaton can be *instantiated* into a probabilistic automaton by specifying the exact values of the non-zero probabilistic transitions.

We show that the two following properties of numberless probabilistic automata are recursively inseparable:

- all instances of the numberless automaton have value 1,
- no instance of the numberless automaton has value 1.

## 1 Introduction

In 1963 Rabin [12] introduced the notion of probabilistic automata, which are finite automata able to randomise over transitions. A probabilistic automaton has a finite set of control states  $Q$ , and processes finite words; each transition consists in updating the control state according to a given probabilistic distribution determined by the current state and the input letter. This powerful model has been widely studied and has applications in many fields like software verification [3], image processing [5], computational biology [6] and speech processing [10].

Several algorithmic properties of probabilistic automata have been considered in the literature, sometimes leading to efficient algorithms. For instance, *functional equivalence* is decidable in polynomial time [13,14], and even faster with randomised algorithms, which led to applications in software verification [9].

However, many natural decision problems are undecidable, and part of the literature on probabilistic automata is about *intractability results*. For example the *emptiness*, the *threshold isolation* and the *value 1* problems are undecidable [11,2,8].

A striking result due to Condon and Lipton [4] states that, for every  $\epsilon > 0$ , the two following problems are recursively inseparable: given a probabilistic automaton  $\mathcal{A}$ ,

---

<sup>\*</sup> The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 259454 (GALE) and from the French Agence Nationale de la Recherche projects EQINOCs (ANR-11-BS02-004) and STOCH-MC (ANR-13-BS02-0011-01).

- does  $\mathcal{A}$  accept some word with probability greater than  $1 - \varepsilon$ ?
- does  $\mathcal{A}$  accept every word with probability less than  $\varepsilon$ ?

In the present paper we focus on *numberless* probabilistic automata, *i.e.* probabilistic automata whose non-zero probabilistic transitions are specified but the exact values of the probabilities are not. A numberless probabilistic automaton can be *instantiated* into a probabilistic automaton by specifying the exact values of the non-zero probabilistic transitions (see Section 2 for formal definitions).

The notion of numberless probabilistic automaton is motivated by the following example. Assume we are given a digital chip modelled as a finite state machine controlled by external inputs. The internal transition structure of the chip is known but the transitions themselves are not observable. We want to compute an initialisation input sequence that puts the chip in a particular initial state. In case some of the chip components have failure probabilities, this can be reformulated as a value 1 problem for the underlying probabilistic automaton: is there an input sequence whose acceptance probability is arbitrarily close to 1? Assume now that the failure probabilities are not fixed *a priori* but we can tune the quality of our components and choose the failure probabilities, for instance by investing in better components. Then we are dealing with a numberless probabilistic automaton and we would like to determine whether it can be instantiated into a probabilistic automaton with value 1, in other words we want to solve an *existential value 1 problem* for the numberless probabilistic automaton. If the failure probabilities are unknown then we are facing a second kind of problem called the *universal value 1 problem*: determine whether all instances of the automaton have value 1. We also consider variants where the freedom to choose transition probabilities is restricted to some intervals, that we call *noisy value 1 problems*.

One may think that relaxing the constraints on the exact transition probabilities makes things algorithmically much easier. However this is not the case, and we prove that the existential and universal value 1 problems are recursively inseparable: given a numberless probabilistic automaton  $\mathcal{C}$ ,

- do all instances of  $\mathcal{C}$  have value 1?
- does no instance of  $\mathcal{C}$  have value 1?

This result is actually a corollary of a generic construction which constitutes the technical core of the paper and has the following properties. For every simple probabilistic automaton  $\mathcal{A}$ , we construct a numberless probabilistic automaton  $\mathcal{C}$  such that the three following properties are equivalent:

- (i)  $\mathcal{A}$  has value 1,
- (ii) one of the instances of  $\mathcal{C}$  has value 1,
- (iii) all instances of  $\mathcal{C}$  have value 1.

The definitions are given in Section 2. The main technical result appears in Section 3, where we give the generic construction whose properties are described

above. In Section 4, we discuss the implications of our results, first for the noisy value 1 problems, and second for probabilistic Büchi automata [1].

## 2 Definitions

Let  $A$  be a finite alphabet. A (finite) word  $u$  is a (possibly empty) sequence of letters  $u = a_0 a_1 \cdots a_{n-1}$ ; the set of finite words is denoted by  $A^*$ .

A probability distribution over a finite set  $Q$  is a function  $\delta : Q \rightarrow \mathbb{Q}_{\geq 0}$  such that  $\sum_{q \in Q} \delta(q) = 1$ ; we denote by  $\frac{1}{3} \cdot q + \frac{2}{3} \cdot q'$  the distribution that picks  $q$  with probability  $\frac{1}{3}$  and  $q'$  with probability  $\frac{2}{3}$ , and by  $q$  the trivial distribution picking  $q$  with probability 1. The support of a distribution  $\delta$  is the set of states picked with positive probability, *i.e.*,  $\text{supp}(\delta) = \{q \in Q \mid \delta(q) > 0\}$ . Finally, the set of probability distributions over  $Q$  is  $\mathcal{D}(Q)$ .

**Definition 1 (Probabilistic automaton).** A probabilistic automaton (PA) is a tuple  $\mathcal{A} = (Q, A, q_0, \Delta, F)$ , where  $Q$  is a finite set of states,  $A$  is the finite input alphabet,  $q_0 \in Q$  is the initial state,  $\Delta : Q \times A \rightarrow \mathcal{D}(Q)$  is the probabilistic transition function, and  $F \subseteq Q$  is the set of accepting states.

For convenience, we also use  $\mathbb{P}_{\mathcal{A}}(s \xrightarrow{u} t)$  to denote the probability of going from the state  $s$  to the state  $t$  reading  $u$ ,  $\mathbb{P}_{\mathcal{A}}(s \xrightarrow{u} S)$  to denote the probability of going from the state  $s$  to a state in  $S$  reading  $u$ , and  $\mathbb{P}_{\mathcal{A}}(u)$ , the *acceptance probability* of a word  $u \in A^*$ , to denote  $\mathbb{P}_{\mathcal{A}}(q_0 \xrightarrow{u} F)$ .

We often consider the case of *simple probabilistic automata*, where the transition probabilities can only be 0,  $\frac{1}{2}$ , or 1.

**Definition 2 (Value).** The value of a PA  $\mathcal{A}$ , denoted  $\text{val}(\mathcal{A})$ , is the supremum acceptance probability over all input words

$$\text{val}(\mathcal{A}) = \sup_{u \in A^*} \mathbb{P}_{\mathcal{A}}(u) .$$

The value 1 problem asks, given a (simple) PA  $\mathcal{A}$  as input, whether  $\text{val}(\mathcal{A}) = 1$ .

**Theorem 1 ([8]).** The value 1 problem is undecidable for simple PA.

**Definition 3 (Numberless probabilistic automaton).** A numberless probabilistic automaton (NPA) is a tuple  $\mathcal{A} = (Q, A, q_0, T, F)$ , where  $Q$  is a finite set of states,  $A$  is the finite input alphabet,  $q_0 \in Q$  is the initial state,  $T \subseteq Q \times A \times Q$  is the numberless transition function, and  $F \subseteq Q$  is the set of accepting states.

The numberless transition function  $T$  is an abstraction of probabilistic transition functions. We say that  $\Delta$  is consistent with  $T$  if for all letters  $a$  and states  $s$  and  $t$ ,  $\Delta(s, a, t) > 0$  if, and only if  $(s, a, t) \in T$ .

A numberless probabilistic automaton is an equivalence class of probabilistic automata, which share the same set of states, input alphabet, initial and accepting states, and whose transition functions have the same support.

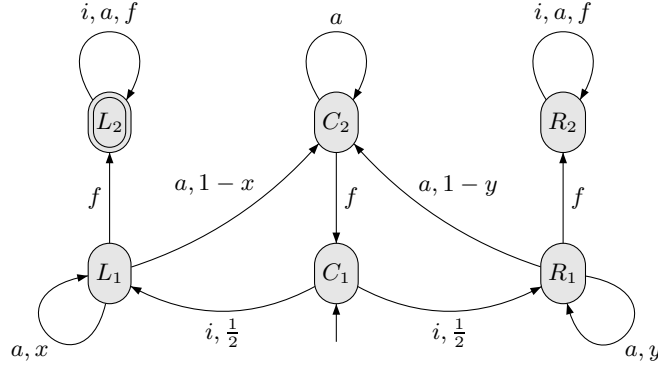
A NPA  $\mathcal{A} = (Q, A, q_0, T, F)$  together with a probabilistic transition function  $\Delta$  consistent with  $T$  defines a PA  $\mathcal{A}[\Delta] = (Q, A, q_0, \Delta, F)$ . Conversely, a PA  $\mathcal{A} = (Q, A, q_0, \Delta, F)$  induces an underlying NPA  $[\mathcal{A}] = (Q, A, q_0, T, F)$ , where  $T \subseteq Q \times A \times Q$  is defined by  $(q, a, p) \in T$  if  $\Delta(q, a)(p) > 0$ .

We consider two decision problems for NPA:

- **The existential value 1 problem:** given a NPA  $\mathcal{A}$ , determine whether there exists  $\Delta$  such that  $\text{val}(\mathcal{A}[\Delta]) = 1$ .
- **The universal value 1 problem:** given a NPA  $\mathcal{A}$ , determine whether for all  $\Delta$ , we have  $\text{val}(\mathcal{A}[\Delta]) = 1$ .

**Proposition 1.** *There exists a NPA such that:*

- *there exists  $\Delta$  such that  $\text{val}(\mathcal{A}[\Delta]) = 1$ ,*
- *there exists  $\Delta'$  such that  $\text{val}(\mathcal{A}[\Delta']) < 1$ .*



**Fig. 1.** This NPA has value 1 if and only if  $x > y$

In this automaton, adapted from [8,7], the shortest word that can be accepted is  $i \cdot f$ , as  $i$  goes from  $C_1$  to  $L_1$ , and  $f$  goes from  $L_1$  to  $L_2$ . However, there are as much chances to go to  $R_2$ , so the value of  $i \cdot f$  is  $1/2$ .

If  $x$  is strictly less than  $y$ , one can tip the scales to the left by adding  $a$ 's between the  $i$  and  $f$ : each time, the run will have more chances to stay left than to stay right. After reading  $i \cdot a^n \cdot f$ , the probability of reaching  $L_2$  is equal to  $x^n$ , while the probability of reaching  $R_2$  is only  $y^n$ . There is also a very high chance that the run went back to  $C_1$ , but from there we can simply repeat our word an arbitrary number of times.

Let  $x, y$ , and  $\varepsilon$  be three real numbers such that  $0 \leq y < x \leq 1$ , and  $0 < \varepsilon \leq 1$ . There is an integer  $n$  such that  $x^n/(x^n+y^n)$  is greater than  $1 - \varepsilon/2$ , and an integer  $m$  such that  $(1 - x^n - y^n)^m$  is less than  $\varepsilon/2$ . The word  $(i \cdot a^n \cdot f)^m$  is accepted with probability greater than  $1 - \varepsilon$ .

On the other hand, if  $x \leq y$ , there is no word with value higher than  $1/2$ .

### 3 Recursive inseparability for numberless value 1 problems

In this section, we prove the following theorem:

**Theorem 2.** *The two following problems for numberless probabilistic automata are recursively inseparable:*

- *all instances have value 1,*
- *no instance has value 1.*

Recall that two decision problems  $A$  and  $B$  are recursively inseparable if their languages  $L_A$  and  $L_B$  of accepted inputs are disjoint and there exists no recursive language  $L$  such that  $L_A \subseteq L$  and  $L \cap L_B = \emptyset$ .

Note that it implies that both  $A$  and  $B$  are undecidable.

Equivalently, this means there exists no terminating algorithm which has the following behaviour on input  $x$ :

- if  $x \in L_A$ , then the algorithm answers “YES”
- if  $x \in L_B$ , then the algorithm answers “NO”.

On an input that belongs neither to  $L_A$  nor to  $L_B$ , the algorithm’s answer can be either “YES” or “NO”.

#### 3.1 Overall construction

**Lemma 1.** *There exists an effective construction which takes as input a simple PA  $\mathcal{A}$  and constructs a NPA  $\mathcal{C}$  such that*

$$\text{val}(\mathcal{A}) = 1 \iff \forall \Delta, \text{val}(\mathcal{C}[\Delta]) = 1 \iff \exists \Delta, \text{val}(\mathcal{C}[\Delta]) = 1 .$$

We first explain how Lemma 1 implies Theorem 2. Assume towards contradiction that the problems “all instances have value 1” and “no instance has value 1” are recursively separable. Then there exists an algorithm  $A$  taking a NPA as input and such that: if all instances have value 1, then it answers “YES”, and if no instance has value 1, then it answers “NO”. We show using Lemma 1 that this would imply that the value 1 problem is decidable for simple PA, contradicting Theorem 1. Indeed, let  $\mathcal{A}$  be a simple PA, applying the construction yields a NPA  $\mathcal{C}$  such that

$$\text{val}(\mathcal{A}) = 1 \iff \forall \Delta, \text{val}(\mathcal{C}[\Delta]) = 1 \iff \exists \Delta, \text{val}(\mathcal{C}[\Delta]) = 1 .$$

In particular, either all instances of  $\mathcal{C}$  have value 1, or no instance of  $\mathcal{C}$  has value 1. Hence, if it answers “YES” then  $\text{val}(\mathcal{A}) = 1$  and if it answers “NO” then  $\text{val}(\mathcal{A}) < 1$ , allowing to decide whether  $\mathcal{A}$  has value 1 or not. This concludes the proof of Theorem 2 assuming Lemma 1.

The construction follows two steps.

The first step is to build from  $\mathcal{A}$  a family of PA's  $\mathcal{B}_\lambda$  whose transitions are all of the form  $q \xrightarrow{a} (\lambda \cdot r, (1 - \lambda) \cdot s)$  in such a way that  $\mathcal{B}_\lambda$  has the same value as  $\mathcal{A}$  for any value of  $\lambda$ . Note that, while all the  $\mathcal{B}_\lambda$ 's belong to the same NPA, they are *not*, in general, a NPA: for example, if  $\mathcal{A}$  were the simple version of the automaton of Figure 1, the  $\mathcal{B}_\lambda$ 's would be the instances where  $x = y = \lambda$ , while the underlying NPA would also include the cases where  $x \neq y$ .

The second step is to build from the  $\mathcal{B}_\lambda$ 's a NPA  $\mathcal{C}$  such that, for each probabilistic transition function  $\Delta$ , there is a  $\lambda$  such that  $\mathcal{C}[\Delta]$  has value 1 if, and only if  $\mathcal{B}_\lambda$  has value 1.

It follows that:

$$\begin{aligned} \exists \Delta, \text{val}(\mathcal{C}[\Delta]) = 1 &\implies \exists \lambda, \text{val}(\mathcal{B}_\lambda) = 1 \\ &\implies \text{val}(\mathcal{A}) = 1 \\ &\implies \forall \lambda, \text{val}(\mathcal{B}_\lambda) = 1 \\ &\implies \forall \Delta, \text{val}(\mathcal{C}[\Delta]) = 1 . \end{aligned}$$

### 3.2 The fair coin construction

Let  $\mathcal{A} = (Q, A, q_0, \Delta, F)$  be a simple PA over the alphabet  $A$ . We construct a family of PAs  $(\mathcal{B}_\lambda)_{\lambda \in ]0,1[}$  over the alphabet  $B = A \cup \{\#\}$ , whose transitions have probabilities 0,  $\lambda$ ,  $1 - \lambda$  or 1, as follows.

The automaton  $\mathcal{B}_\lambda$  is a copy of  $\mathcal{A}$  where each transition of  $\mathcal{A}$  is replaced by the gadget illustrated in Figure 2 (for simplicity, we assume that all the transitions of  $\mathcal{A}$  are probabilistic). The initial and final states are the same in  $\mathcal{A}$  and  $\mathcal{B}_\lambda$ .

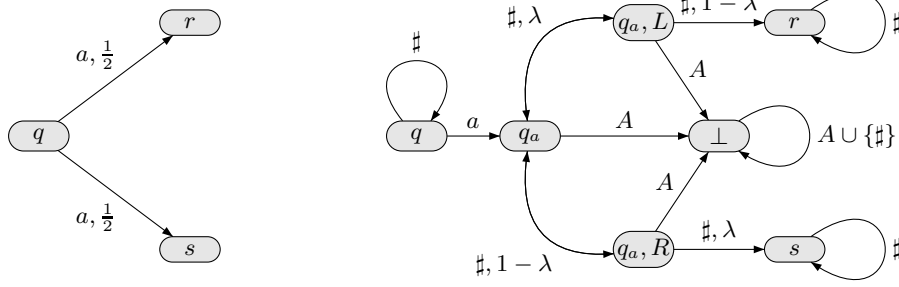
The left hand side shows part of automaton  $\mathcal{A}$ : a probabilistic transition from  $q$  reading  $a$ , leading to  $r$  or  $s$  each with probability half. The right hand side shows how this behaviour is simulated by  $\mathcal{B}_\lambda$ : the letter  $a$  leads to an intermediate state  $q_a$ , from which we can read a new letter  $\#$ . Each time a pair of  $\#$ 's is read, the automaton  $\mathcal{B}_\lambda$  goes to  $r$  with probability  $\lambda \cdot (1 - \lambda)$ , goes to  $s$  with probability  $(1 - \lambda) \cdot \lambda$ , and stays in  $q_a$  with probability  $\lambda^2 + (1 - \lambda)^2$ . Reading a letter other than  $\#$  while the automata is still in one of the new states leads to the new sink state  $\perp$ , which is not accepting. Thus, the probability of going to  $r$  is equal to the probability of going to  $s$ , and we can make the probability of a simulation failure as low as we want by increasing the number of  $\#$ 's between two successive "real" letters.

Let  $u$  be a word of  $A^*$ . We denote by  $[u]^k$  the word of  $B^*$  where each letter  $a \in A$  of  $u$  is replaced by  $a \cdot \#^{2k}$ . Conversely, if  $w$  is a word of  $B^*$ , we denote by  $\tilde{w}$  the word obtained from  $w$  by removing all occurrences of the letter  $\#$ .

Intuitively, a run of  $\mathcal{A}$  on the word  $u$  is simulated by a run of  $\mathcal{B}_\lambda$  on the word  $[u]^k$ . Whenever there is a transition in  $\mathcal{A}$ ,  $\mathcal{B}_\lambda$  makes  $k$  attempts to simulate it through the gadget of Figure 2, and each attempt succeeds with probability  $(1 - 2\lambda \cdot (1 - \lambda))$ , so each transition fails with probability:

$$A_{\lambda,k} = 1 - (1 - 2\lambda \cdot (1 - \lambda))^k .$$

**Proposition 2.** *The probabilistic automaton  $\mathcal{B}_\lambda$  satisfies:*



**Fig. 2.** The fair coin gadget.

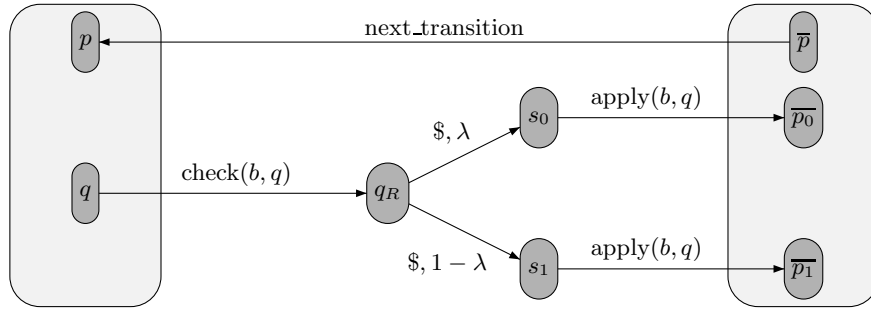
1. For all  $q, r \in Q$ ,  $a \in A$ , and  $k \in \mathbb{N}$ ,  $\mathbb{P}_{\mathcal{B}_\lambda}(q \xrightarrow{[a]^k} r) = A_{\lambda,k} \cdot \mathbb{P}_{\mathcal{A}}(q \xrightarrow{a} r)$ ,
2. For all  $q, r \in Q$ ,  $u \in A^*$ , and  $k \in \mathbb{N}$ ,  $\mathbb{P}_{\mathcal{B}_\lambda}(q \xrightarrow{[u]^k} r) = A_{\lambda,k}^{[u]} \cdot \mathbb{P}_{\mathcal{A}}(q \xrightarrow{u} r)$ .
3. For all  $q, r \in Q$ ,  $w \in (A \cup \{\#\})^*$ , and  $k \in \mathbb{N}$ ,  $\mathbb{P}_{\mathcal{B}_\lambda}(q \xrightarrow{w} r) \leq \mathbb{P}_{\mathcal{A}}(q \xrightarrow{\bar{w}} r)$ .

It follows from Proposition 2 that for any  $\lambda$ , the value of  $\mathcal{B}_\lambda$  is equal to the value of  $\mathcal{A}$ .

### 3.3 The simulation construction

All the  $\mathcal{B}_\lambda$ 's induce the same NPA, that we denote by  $\mathcal{B}$ . The problem is that there are many other instances of  $\mathcal{B}$ , whose values may be higher than the value of  $\mathcal{A}$  (recall the example of Figure 1, where the  $\mathcal{B}_\lambda$  have value  $1/2$ , while there are instances of  $\mathcal{B}$  with value 1). In this subsection, we construct a NPA  $\mathcal{C}$  (over an extended alphabet  $C$ ) whose instances simulate all the  $\mathcal{B}_\lambda$ 's, but only them.

The idea is that the new NPA should only have *one* probabilistic transition. An instance of this transition translates to a value for  $\lambda$ . Figure 3 describes a first attempt at this (notice that our convention is that a non-drawn transition means a loop, rather than a transition to a sink state).



**Fig. 3.** Naive fusion of the probabilistic transitions.

In this automaton, that we call  $\mathcal{B}'$ , there are two copies of the set of states, and a single shared probabilistic transition  $q_R \xrightarrow{\$} (\lambda \cdot s_0, (1-\lambda) \cdot s_1)$  between them. In order to make all the probabilistic transitions of  $\mathcal{B}$  happen in this center area, we use new letters to detect where the runs come from before the probabilistic transition, and where it should go afterwards.

For each pair of a letter  $b$  in  $B$  and a state  $q$  in  $Q$ , we introduce two new letters  $\text{check}(b, q)$  and  $\text{apply}(b, q)$ . The letter  $\text{check}(b, q)$  loops over each state except the left copy of  $q$ , which goes to  $q_R$ . The letter  $\text{apply}(b, q)$  loops over each state except  $s_0$ , from where it goes to the  $\lambda$ -valued successor of  $(q, b)$  in  $\mathcal{B}_\lambda$ , and  $s_1$ , from where it goes to the  $(1-\lambda)$ -valued successor of  $(q, b)$  in  $\mathcal{B}_\lambda$ . The new letter  $\text{next\_transition}$  sends the run back to the left part once each possible state has been tested.

Thus, if we define the morphism  $\hat{\cdot}$  by its action on letters:

$$\hat{b} = \text{check}(b, q_0) \cdot \$ \cdot \text{apply}(b, q_0) \cdots \text{check}(b, q_{n-1}) \cdot \$ \cdot \text{apply}(b, q_{n-1}) \cdot \text{next\_transition},$$

where the  $q_i$ 's are the states of  $\mathcal{B}_\lambda$ , we get for any word  $u$  on the alphabet  $B$ ,  $\mathbb{P}_{\mathcal{B}_\lambda}(u) = \mathbb{P}_{\mathcal{B}'}(\hat{u})$ .

The problem with this automaton is that one can “cheat”, either by not testing an unwelcome state, or by changing state and letter between a check and the subsequent apply. In order to avoid this kind of behaviour, we change the automaton so that it is necessary to win an arbitrarily large number of successive times in order to approach the value 1, and we can test whether the word is fair after the first successful attempt. A side effect is that the simulation only works for the value 1: for other values, it might be better to cheat. The resulting automaton is described in Figure 4.

The structure of the automaton of Figure 3 is still there, but it has been augmented with an extra layer of scrutiny: each time we use the probabilistic transition, there is now a positive probability  $(1-\theta)$  to go to a new wait state. There is also a new letter  $\text{next\_word}$  which has the following effect:

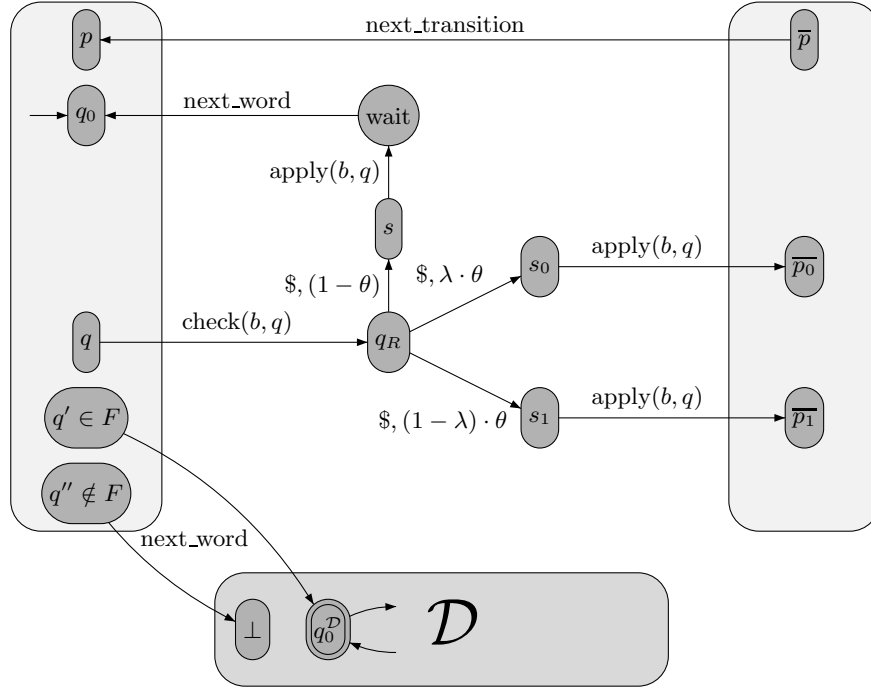
- if the run is in an accepting state, it goes to the initial state of the fairness checker  $\mathcal{D}$ ;
- if the run is in a non-accepting state, it goes to the non-accepting sink state  $\perp$  of  $\mathcal{D}$ ;
- if the run is in the wait state, it goes back to the initial state.

The fairness checker  $\mathcal{D}$  is a deterministic automaton which accepts the language  $\{\hat{u} \cdot \text{next\_word} \mid u \in B^*\}^*$ . Its final state is the only final state in all of  $\mathcal{C}$ .

Intuitively, a run can still cheat before the *first*  $\text{next\_word}$  letter, but the benefits of doing so are limited: the probability that  $\mathcal{C}[\lambda, \theta]$  accepts a word at that point is at most  $\theta$  (except if the empty word is accepting, but that case is trivial). After that point, cheating is risky: if the run already reached  $\mathcal{D}$ , a false move will send it to the sink.

A more formal proof follows. A simple inspection of the construction of  $\mathcal{C}$  yields Proposition 3:





**Fig. 4.** The Numberless Probabilistic Automaton  $\mathcal{C}$ .

**Proposition 3.** Let  $u$  be a word of  $B^*$  of length  $k$ . We have:

$$\begin{aligned} \mathbb{P}_{\mathcal{C}[\lambda, \theta]}(\widehat{u} \cdot \text{next\_word}) &= \theta^k \cdot \mathbb{P}_{\mathcal{B}_\lambda}(u) \\ \mathbb{P}_{\mathcal{C}[\lambda, \theta]}((\widehat{u} \cdot \text{next\_word})^\ell) &= (1 - (1 - \theta^k)^\ell) \cdot \mathbb{P}_{\mathcal{B}_\lambda}(u) . \end{aligned}$$

It follows from Proposition 3 that the value of  $\mathcal{C}[\lambda, \theta]$  is at least the value of  $\mathcal{B}_\lambda$ .

**Proposition 4.** Let  $u$  be a word of  $(C \setminus \{\text{next\_word}\})^+$ . We have:

$$\mathbb{P}_{\mathcal{C}[\lambda, \theta]}(u) \leq \theta .$$

Proposition 5 formalises the fact that there is no point in cheating after the first `next_word` letter:

**Proposition 5.** Let  $u_1, \dots, u_k$  be  $k$  words of  $(C \setminus \{\text{next\_word}\})^*$  and  $w$  be the word  $u_1 \cdot \text{next\_word} \cdots u_k \cdot \text{next\_word}$ . Then, for any  $1 \leq i \leq k$ , if  $u_i \notin \widehat{B}^*$ , we have:

$$\mathbb{P}_{\mathcal{C}[\lambda, \theta]}(w) \leq \mathbb{P}_{\mathcal{C}[\lambda, \theta]}(u_i \cdot \text{next\_word} \cdots u_k \cdot \text{next\_word}) .$$

*Proof.* After reading  $u_1 \cdots u_{i-1} \cdot \text{next\_word}$ , a run must be in one of the following three states:  $q_0$ ,  $q_0^{\mathcal{D}}$ , and  $\perp$ . As  $u_i \notin \widehat{B}^*$ , reading it from  $q_0^{\mathcal{D}}$  will lead to  $\perp$ . Thus,

$$\mathbb{P}_{\mathcal{C}[\lambda, \theta]}(w) = \mathbb{P}_{\mathcal{C}[\lambda, \theta]}(q_0 \xrightarrow{u_1 \cdots u_{i-1} \cdot \text{next\_word}} q_0) \cdot \mathbb{P}_{\mathcal{C}[\lambda, \theta]}(u_i \cdots u_k \cdot \text{next\_word}) ,$$

and Proposition 5 follows.

Finally, Proposition 6 shows that  $\mathcal{C}[\lambda, \theta]$  cannot have value 1 if  $\mathcal{B}_\lambda$  does not.

**Proposition 6.** *For all words  $w \in C^*$  such that  $\mathbb{P}_{\mathcal{C}[\lambda, \theta]}(w) > \theta$ , there exists a word  $v \in B^*$  such that*

$$\mathbb{P}_{\mathcal{B}_\lambda}(u) \geq \frac{\mathbb{P}_{\mathcal{C}[\lambda, \theta]}(w) - \theta}{1 - \theta} .$$

*Proof.* Let us write  $w = u_1 \cdot \text{next\_word} \cdots u_k \cdot \text{next\_word}$  with  $u_1, \dots, u_k \in (C \setminus \{\text{next\_word}\})^*$ . By Proposition 4,  $k > 1$ , and by Proposition 5 we can assume that  $u_2, \dots, u_k$  belong to  $\widehat{B}^*$ . Let  $v_2, \dots, v_k$  be the words of  $B^*$  such that  $u_i = \widehat{v}_i$ . The  $\mathcal{C}[\lambda, \theta]$ -value of  $w$  can be seen as a weighted average of 1 (the initial cheat, with a weight of  $\theta$ ) and the  $\mathcal{B}_\lambda$  values of the  $v_i$ 's (the weight of  $\mathbb{P}_{\mathcal{B}_\lambda}(v_i)$  is the probability that the run enters  $\mathcal{D}$  while reading  $u_i$ ). It follows that at least one of the  $v_i$ 's has a  $\mathcal{B}_\lambda$ -value greater than the  $\mathcal{C}[\lambda, \theta]$ -value of  $w$ .

Thus, for each  $\lambda$  and  $\theta$ , the value of  $\mathcal{C}[\lambda, \theta]$  is 1 if and only if the value of  $\mathcal{B}_\lambda$  is 1. As all the  $\mathcal{B}_\lambda$ 's have the same value, which is equal to the value of  $\mathcal{A}$ , we get:

$$\begin{aligned} \exists \Delta, \text{val}(\mathcal{C}[\Delta]) = 1 &\implies \exists \lambda, \text{val}(\mathcal{B}_\lambda) = 1 \\ &\implies \text{val}(\mathcal{A}) = 1 \\ &\implies \forall \lambda, \text{val}(\mathcal{B}_\lambda) = 1 \\ &\implies \forall \Delta, \text{val}(\mathcal{C}[\Delta]) = 1 . \end{aligned}$$

Theorem 2 follows.

## 4 Consequences

In this section, we show several consequences of the recursive inseparability results from Theorem 2 and of the construction from Lemma 1. The first is a series of undecidability results for variants of the value 1 problem. The second is about probabilistic Büchi automata with probable semantics, as introduced in [1].

### 4.1 The noisy value 1 problems

Observe that Theorem 2 implies the following corollary:

**Corollary 1.** *Both the universal and the existential value 1 problems are undecidable.*

We can go further. Note that the universal and existential value 1 problems quantify over all possible probabilistic transition functions. Here we define two more realistic problems for NPA, where the quantification is restricted to probabilistic transition functions that are  $\varepsilon$ -close to a given probabilistic transition function:

- **The noisy existential value 1 problem:** given a NPA  $\mathcal{A}$ , a probabilistic transition function  $\Delta$  and  $\varepsilon > 0$ , determine whether there exists  $\Delta'$  such that  $|\Delta' - \Delta| \leq \varepsilon$  and  $\text{val}(\mathcal{A}[\Delta']) = 1$ .

- **The noisy universal value 1 problem:** given a NPA  $\mathcal{A}$ , a probabilistic transition function  $\Delta$  and  $\varepsilon > 0$ , determine whether for all  $\Delta'$  such that  $|\Delta' - \Delta| \leq \varepsilon$ , we have  $\text{val}(\mathcal{A}[\Delta']) = 1$ .

It follows from Lemma 1 that both problems are undecidable:

**Corollary 2.** *Both the noisy universal and the noisy existential value 1 problems are undecidable.*

Indeed, we argue that the construction from Lemma 1 implies a reduction from either of these problems to the value 1 problem for simple PA, hence the undecidability. Let  $\mathcal{A}$  be a simple PA, the construction yields a NPA  $\mathcal{C}$  such that:

$$\text{val}(\mathcal{A}) = 1 \iff \forall \Delta, \text{val}(\mathcal{C}[\Delta]) = 1 \iff \exists \Delta, \text{val}(\mathcal{C}[\Delta]) = 1 .$$

It follows that for any probabilistic transition function  $\Delta$  and any  $\varepsilon > 0$ , we have:

$$\begin{aligned} \text{val}(\mathcal{A}) = 1 &\iff (\forall \Delta', |\Delta' - \Delta| \leq \varepsilon \implies \text{val}(\mathcal{C}[\Delta']) = 1) \\ &\iff (\exists \Delta', |\Delta' - \Delta| \leq \varepsilon \wedge \text{val}(\mathcal{C}[\Delta']) = 1) . \end{aligned}$$

This completes the reduction.

## 4.2 Probabilistic Büchi automata with probable semantics

We consider PA over infinite words, as introduced in [1]. A probabilistic Büchi automaton (PBA)  $\mathcal{A}$  can be equipped with the so-called probable semantics, defining the language (over infinite words):

$$L^{>0}(\mathcal{A}) = \{w \in A^\omega \mid \mathbb{P}_{\mathcal{A}}(w) > 0\} .$$

It was observed in [1] that the value 1 problem for PA (over finite words) easily reduces to the emptiness problem for PBA with probable semantics (over infinite words).

Informally, from a PA  $\mathcal{A}$ , construct a PBA  $\mathcal{A}'$  by adding a transition from every final state to the initial state labelled with a new letter  $\sharp$ . (From a non-final state, the letter  $\sharp$  leads to a rejecting sink.) As explained in [1], this simple construction ensures that  $\mathcal{A}$  has value 1 if and only if  $\mathcal{A}'$  is non-empty, equipped with the probable semantics.

This simple reduction, together with Theorem 2, implies the following corollary:

**Corollary 3.** *The two following problems for numberless PBA with probable semantics are recursively inseparable:*

- *all instances have a non-empty language,*
- *no instance has a non-empty language.*

## Acknowledgments

We would like to thank the referees for their helpful comments.

## References

1. Christel Baier, Nathalie Bertrand, and Marcus Größer. Probabilistic  $\omega$ -automata. *Journal of the ACM*, 59(1):1, 2012.
2. 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.
3. Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Algorithms for omega-regular games of incomplete information. *Logical Methods in Computer Science*, 3(3), 2007.
4. Anne Condon and Richard J. Lipton. On the complexity of space bounded interactive proofs (extended abstract). In *Foundations of Computer Science*, pages 462–467, 1989.
5. Karel Culik and Jarkko Kari. *Digital images and formal languages*, pages 599–616. Springer-Verlag New York, Inc., 1997.
6. Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, July 1999.
7. Nathanaël Fijalkow, Hugo Gimbert, and Youssef Oualhadj. Deciding the value 1 problem for probabilistic leaktight automata. In *Logics in Computer Science*, pages 295–304, 2012.
8. Hugo Gimbert and Youssef Oualhadj. Probabilistic automata on finite words: Decidable and undecidable problems. In *International Colloquium on Automata, Languages and Programming*, pages 527–538, 2010.
9. Stefan Kiefer, Andrzej S. Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. Language equivalence for probabilistic automata. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *CAV*, volume 6806 of *Lecture Notes in Computer Science*, pages 526–540. Springer, 2011.
10. Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23:269–311, June 1997.
11. Azaria Paz. *Introduction to probabilistic automata*. Academic Press, 1971.
12. Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.
13. Marcel-Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4, 1961.
14. Wen-Guey Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM Journal on Computing*, 21(2):216–227, 1992.