Deciding the Value 1 Problem of Probabilistic Leaktight Automata

Nathanaël Fijalkow
LIAFA,
Université Denis Diderot-Paris 7, France
Email: nath@liafa.univ-paris-diderot.fr

Hugo Gimbert
CNRS, LaBRI
Université de Bordeaux, France
Email: hugo.gimbert@labri.fr

Youssouf Oualhadj
LaBRI
Université de Bordeaux, France
Email: youssouf.oualhadj@labri.fr

***Abstract*—The value 1 problem is a decision problem for probabilistic automata over finite words: given a probabilistic automaton, are there words accepted with probability arbitrarily close to 1?**

This problem was proved undecidable recently. We sharpen this result, showing that the undecidability holds even if the probabilistic automata have only one probabilistic transition.

Our main contribution is to introduce a new class of probabilistic automata, called *leaktight automata*, for which the value 1 problem is shown decidable (and PSPACE-complete). We construct an algorithm based on the computation of a monoid abstracting the behaviors of the automaton, and rely on algebraic techniques developed by Simon for the correctness proof. The class of leaktight automata is decidable in PSPACE, subsumes all subclasses of probabilistic automata whose value 1 problem is known to be decidable (in particular deterministic automata), and is closed under two natural composition operators.

***Index Terms*—Probabilistic automata, Value 1 problem, Algebraic Techniques in Automata Theory.**

I. Introduction

Probabilistic automata: Rabin invented a very simple yet powerful model of probabilistic machine called probabilistic automata, which, quoting Rabin, "are a generalization of finite deterministic automata" [20]. A probabilistic automaton has a finite set of states Q and reads input words over a finite alphabet A. The computation starts from the initial state i and consists in reading the input word sequentially; the state is updated according to transition probabilities determined by the current state and the input letter. The probability to accept a finite input word is the probability to terminate the computation in one of the final states $F \subseteq Q$.

From a language-theoretic perspective, several algorithmic properties of probabilistic automata are known: while language emptiness is undecidable [2], [14], [19], language equivalence is decidable [9], [21], [24] as well as other properties [8], [10].

Rather than formal language theory, our initial motivation for this work comes from control and game theory: we aim at solving algorithmic questions about partially observable Markov decision processes and stochastic games. For this reason, we consider probabilistic automata as machines controlled by a blind controller, who is in charge of choosing the sequence of input letters in order to maximize the acceptance probability. While in a fully observable Markov decision process the controller can observe the current state of the process to choose adequately the next input letter, a blind controller does not observe anything and its choice depends only on the number of letters already chosen. In other words, the strategy of a blind controller is an input word of the automaton.

The value of a probabilistic automaton: With this game-theoretic interpretation in mind, we define the *value* of a probabilistic automaton as the supremum of acceptance probabilities over all input words, and we would like to compute this value. Unfortunately, as a consequence of an undecidability result due to Paz [19], the value of an automaton is not computable in general. However, the following decision problem was conjectured by Bertoni to be decidable [2]:

Value 1 **problem:** *Given a probabilistic automaton, does the automaton have value* 1*? In other words, are there words accepted with probability arbitrarily close to* 1*?*

Actually, Bertoni formulated the value 1 problem in a different yet equivalent way: "Is the cut-point 1 isolated or not?". There is indeed a close relation between the value 1 problem and the notion of isolated cut-point introduced by Rabin in the very first paper about probabilistic automata. A real number $0 \leq \lambda \leq 1$ is an *isolated cut-point* if there exists a bound $\epsilon > 0$ such that the acceptance probability of any word is either greater than $\lambda + \epsilon$ or smaller than $\lambda - \epsilon$. A theorem of Rabin states that if the cut-point λ is isolated, then the language $L_\lambda = \{w \mid \mathbb{P}_\mathcal{A}(w) \geq \lambda\}$ is regular [20]. The value 1 problem can be reformulated in term of isolated cut-point: an automaton has value 1 if and only if 1 is not an isolated cut-point. Bertoni proved that for λ strictly between 0 and 1, the isolation of λ is undecidable in general, and left the special case $\lambda \in \{0, 1\}$ open.

Recently, the second and third authors of the present paper proved that the value 1 problem is undecidable as well [14]. However, probabilistic automata, and more generally partially observable Markov decision processes and stochastic games, are a widely used model of probabilistic machines considered in many fields like software verification [1], [5], image processing [11], computational biology [12] and speech processing [18]. As a consequence, it is crucial to understand which decision problems are algorithmically tractable for

This work was supported by French CNRS-PEPS Project "Stochastic Games and Verification" and by the ANR projet FREC, 2010 BLAN 0202 02 FREC.

probabilistic automata.

Our result: As a first step, we sharpen the undecidability result: we prove that the value 1 problem is undecidable even for probabilistic automata with only one probabilistic transition. This result motivated the introduction of a new class of probabilistic automata, called *leaktight automata*, for which the value 1 problem is decidable. This subclass subsumes all known subclasses of probabilistic automata sharing this decidability property and is closed under parallel composition and synchronized product. Our algorithm to decide the value 1 problem computes in polynomial space a finite monoid whose elements are directed graphs and checks whether it contains a certain type of elements that are value 1 witnesses.

Related works: The value 1 problem was proved decidable for a subclass of probabilistic automata called $\sharp$-acyclic automata [14]. Since the class of $\sharp$-acyclic automata is strictly contained in the class of leaktight automata, the result of the present paper extends the decidability result of [14]. Chadha et al. [3] recently introduced the class of hierarchical probabilistic automata, which is also strictly contained in the class of leaktight automata. As a consequence of our result, the value 1 problem is decidable for hierarchical probabilistic automata. Our proof techniques totally depart from the ones used in [3], [14]. Instead, we make use of algebraic techniques and in particular Simon's factorization forest theorem, which was successfully used to prove the decidability of the boundedness problem for distance automata [23].

A very related work on probabilistic automata will also appear [6]. The work of [6] focuses on probabilistic automata over infinite words and identifies a new class of probabilistic automata where the value 1 problem is decidable, called structurally simple automata. An interesting direction of future work would be to investigate the connection between leaktight and structurally simple automata.

Outline: We give the basic definitions in Section II. As a first step we present our algorithm to decide the value 1 problem of probabilistic leaktight automata in Section III, which is followed by the decidability of the leaktight property in Section IV. Next, in Section V, we present and prove the technical core of the paper, called the lower bound lemma. Finally, Section VI investigates properties and provides examples of leaktight automata. A full version with proofs is available in [13].

II. Definitions

A. Probabilistic automata

Let Q be a finite set of states. A probability distribution over Q is a row vector δ of size $|Q|$ whose coefficients are real numbers from the interval $[0,1]$ and such that $\sum_{q\in Q}\delta(q)=1$. A probabilistic transition matrix M is a square matrix in $[0,1]^{Q\times Q}$ such that every row of M is a probability distribution over Q.

Definition 1 (Probabilistic automata)**.** *A probabilistic automaton $\mathcal{A}$ is a tuple $(Q, A, (M_a)_{a\in A}, i, F)$, where Q is a finite set of states, A is the finite input alphabet, $(M_a)_{a\in A}$ are the probabilistic transition matrices, $i \in Q$ is the initial state and $F \subseteq Q$ 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. Given an input word $w \in A^*$, we denote by $w(s,t)$ the probability to go from state s to state t when reading the word w. Formally, if $w = a_1 a_2 \cdots a_n$ then $w(s,t) = (M_{a_1}\cdot M_{a_2}\cdots M_{a_n})(s,t)$. Note that $0 \le w(s,t) \le 1$, for all words w and states s and t. Furthermore, the definition of a probabilistic transition matrix implies that $\sum_{t\in Q} w(s,t) = 1$ for all states s.

Definition 2 (Value and acceptance probability)**.** *The* acceptance probability *of a word $w \in A^*$ by $\mathcal{A}$ is $\mathbb{P}_{\mathcal{A}}(w) = \sum_{f\in F} w(i,f)$. The* value *of $\mathcal{A}$, denoted $val(\mathcal{A})$, is the supremum of the acceptance probabilities over all possible input words:*

$$val(\mathcal{A}) = \sup_{w\in A^*} \mathbb{P}_{\mathcal{A}}(w) \ . \tag{1}$$

B. The value 1 problem for probabilistic automata

We are interested in the following decision problem:

Problem (Value 1 Problem)**.** *Given a probabilistic automaton $\mathcal{A}$, decide whether $val(\mathcal{A}) = 1$.*

Whereas the formulation of the value 1 problem only relies *qualitatively* on the asymptotic behaviour of probabilities (the probability to be in non-final states should be arbitrarily small), the answer to the value 1 problem depends *quantitatively* on the transition probabilities.

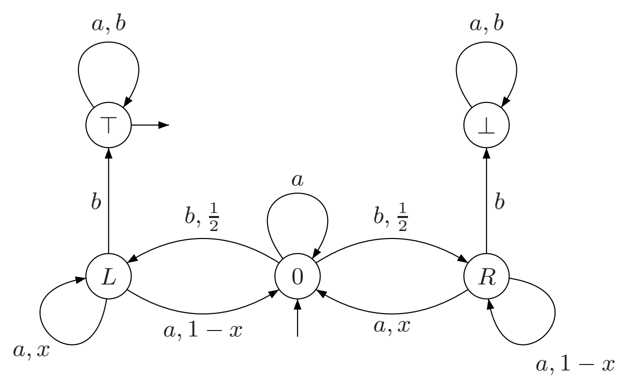

Fig. 1. This automaton has value 1 if and only if $x > \frac{1}{2}$.

For instance, the automaton depicted on Fig. 1 has value 1 if and only if $x > \frac{1}{2}$ and has value less or equal than half otherwise (see also [1], [14] for similar results). Note that in this example, the value is a discontinuous function of the transition probabilities. The input alphabet is $\{a,b\}$, the initial state is the central state 0 and the unique final state is $\top$. In order to maximize the probability to reach $\top$, playing two b's in a row is certainly not a good option because from state 0 this ensures to reach the non-accepting absorbing state $\bot$ with probability at least half. A smarter strategy consists in playing one b, then long sequences of a's followed by one letter b. If $x \le \frac{1}{2}$, there is still no hope to have a word accepted with probability strictly greater than half: starting from 0, and after a b and a sequence of a's, the probability to be in R is greater or equal than the probability to be in L, thus playing ba^nb from state 0 the probability to reach the sink $\bot$ is greater or equal than the probability to reach the final state $\top$. However, if $x > \frac{1}{2}$ then a simple calculation shows that the probability to accept $(ba^n)^{2^n}$ tends to 1 as n goes to infinity.

C. Undecidability in a very restricted case

As a first step we refine the undecidability result: we show that the value 1 problem is undecidable even when restricted to probabilistic automata having exactly one probabilistic transition. For such automata, there exists exactly one state s and one letter a such that $0 \leq M_a(s,t) < 1$ for all t, and the remaining transitions are deterministic: for all triple $(s', a', t) \in S \times A \times S$ such that $(s', a') \neq (s, a)$ then $M_{a'}(s', t) \in \{0, 1\}$.

Our proof goes by simulating a probabilistic automaton $\mathcal{A}$ with a probabilistic automaton $\mathcal{B}$ which has only one probabilistic transition, satisfying $\text{val}(\mathcal{A}) = 1$ if and only if $\text{val}(\mathcal{B}) = 1$.

As a first attempt, we define the automaton $\mathcal{B}$ with a larger alphabet: whenever $\mathcal{A}$ reads a letter a, then $\mathcal{B}$ reads a sequence of actions $\widehat{a}$ corresponding to a, allowing a state-by-state simulation of $\mathcal{A}$. The unique probabilistic transition of $\mathcal{B}$ is used to generate random bits for the simulation. However, the automaton $\mathcal{B}$ cannot check that the sequences of actions are well-formed and allow for a faithful simulation. Hence we modify the construction, such that to simulate the automaton $\mathcal{A}$ on the input word w, the automaton $\mathcal{B}$ now reads $(\widehat{w})^n$ for arbitrarily large n. Each time $\mathcal{B}$ reads a word $\widehat{w}$, it simulates $\mathcal{A}$ on w with a small yet positive probability and "delays" the rest of the simulation, also with positive probability. This delay process allows to run on parallel a deterministic automaton which checks that the sequences of actions are well-formed, ensuring a faithful simulation.

This undecidability result illustrates that even very restricted classes of probabilistic automata may have an undecidable value 1 problem. In Section III, we introduce a non-trivial yet decidable subclass of probabilistic automata, defined by the leaktight property.

D. Informal description of the leaktight property

One of the phenomena that makes tracking vanishing probabilities difficult are *leaks*. A leak occurs in an automaton when a sequence of words turns a set of states $C \subseteq Q$ into a recurrence class C on the long run but on the short run, some of the probability of the recurrence class is "leaking" outside the class.

Such leaks occur in the automaton of Fig. 1 with the input sequence $(a^n b)_{n \in \mathbb{N}}$. The set of states $\{L\}$ and $\{R\}$ are the two recurrence classes on the long run; however there is still a positive probability to reach $\top$ and $\bot$, which vanishes as n grows large. We identified two leaks, one from L to $\top$ and the other from R to $\bot$. As a consequence, the real asymptotic behaviour is complex and depends on the compared speeds of these leaks.

An automaton without leak is called a leaktight automaton. Our main result is to prove that the value 1 problem is decidable when restricted to the subclass of leaktight automata.

The definition of a leaktight automaton relies on two key notions, idempotent words and word-recurrent states.

A finite word u is *idempotent* if reading once or twice the word u does not change qualitatively the transition probabilities:

Definition 3 (Idempotent words)**.** *A finite word $u \in A^*$ is idempotent if for every states $s, t \in Q$,*

$$u(s,t) > 0 \iff (u \cdot u)(s,t) > 0 \ .$$

Idempotent words are everywhere: every word, if iterated a large number of times, becomes idempotent.

Lemma 1. *For every word $u \in A^*$, the word $u^{|Q|!}$ is idempotent.*

A finite word u naturally induces a finite homogeneous Markov chain on Q^*, which splits the set of states into two classes: recurrent states and transient states. Intuitively, a state is transient if there is some non-zero probability to leave it forever, and recurrent otherwise; equivalently from a recurrent state the probability to visit it again in the future is one.

Definition 4 (Recurrent states)**.** *Let $u \in A^*$ be a finite word. A state s is u-*recurrent *if it is recurrent in the finite Markov chain $\mathcal{M}_u$ induced by u, with states Q and transitions probabilities $(u(s,t))_{s,t \in Q}$.*

In the case of idempotent words, recurrence of a state can be easily characterized:

Lemma 2. *Let s be a state and u be an idempotent word. Then s is u-recurrent if and only if for every state t,*

$$u(s,t) > 0 \implies u(t,s) > 0 \ .$$

The formal definition of a leak is as follows:

Definition 5 (Leaks and leaktight automata)**.** *A leak from a state $r \in Q$ to a state $q \in Q$ is a sequence $(u_n)_{n \in \mathbb{N}}$ of idempotent words such that:*

1) *for every $s, t \in Q$, the sequence $(u_n(s,t))_{n \in \mathbb{N}}$ converges to some value $u(s,t)$. We denote by $\mathcal{M}_u$ the Markov chain with states Q and transition probabilities $(u(s,t))_{s,t \in Q}$,*
2) *r is recurrent in $\mathcal{M}_u$,*
3) *for all n in $\mathbb{N}$, $u_n(r,q) > 0$,*
4) *r is not reachable from q in $\mathcal{M}_u$.*

A probabilistic automaton is leaktight if it has no leak.

The automaton depicted in Fig. 1 is not leaktight when $0 < x < 1$ because the sequence $(u_n)_{n \in \mathbb{N}} = (a^n b a^n b)_{n \in \mathbb{N}}$ is a leak from L to $\top$, and from R to $\bot$. (Note that the word $a^n b$ is not idempotent, which is why we consider $a^n b a^n b$.) The limit Markov chain $\mathcal{M}_u$ sends state 0 to states L and R with probability half each, and all other states are absorbing (*i.e* loop with probability 1). In particular, state L is recurrent in $\mathcal{M}_u$, and for every n, $u_n(L, \top) > 0$ but there is no transition from $\top$ to L in $\mathcal{M}_u$.

Several examples of leaktight automata are given in Section VI.

III. The value 1 problem is decidable for leaktight automata

In this section we establish our main result:

Theorem 1. *The value* 1 *problem is decidable for leaktight automata.*

A. The Markov monoid algorithm

Our decision algorithm for the value 1 problem computes iteratively a set $\mathcal{G}$ of directed graphs called limit-words. Each limit-word is meant to represent the asymptotic effect of a sequence of input words, and some particular limit-words can witness that the automaton has value 1.

Algorithm 1 The Markov monoid algorithm.

Input: A probabilistic automaton $\mathcal{A}$.
Output: Decide whether $\mathcal{A}$ has value 1 or not.

```
G ← {a | a ∈ A} ∪ {1}.
repeat
    if there is u, v ∈ G such that u · v ∉ G then
        add u · v to G
    if there is u ∈ G idempotent such that u♯ ∉ G then
        add u♯ to G
until there is nothing to add
if there is a value 1 witness in G then
    return true
else
    return false
```

In the rest of the section, we explain the algorithm in details.

Definition 6 (Limit-word)**.** *A* limit-word *is a map* $\mathbf{u} : Q^2 \to \{0, 1\}$ *satisfying* $\forall s \in Q, \exists t \in Q, \mathbf{u}(s, t) = 1$.

As it will be clear from Definition 9 and 10, for a limit-word $\mathbf{u}$, we interpret $\mathbf{u}(s, t) = 1$ by a positive probability to reach t from s. The condition expresses that our automata are complete: whatever the input word, from any state s there exists some state t which is reached with positive probability. A limit-word $\mathbf{u}$ can be seen as a directed graph with no dead-end, whose vertices are the states of the automaton $\mathcal{A}$, where there is an edge from s to t if $\mathbf{u}(s, t) = 1$.

Initially, $\mathcal{G}$ only contains those limit-words $\mathbf{a}$ that are induced by input letters $a \in A$, where the limit-word $\mathbf{a}$ is defined by:

$$\forall s, t \in Q, (\mathbf{a}(s, t) = 1 \iff a(s, t) > 0) \ ,$$

plus the identity limit-word $\mathbf{1}$ defined by $(\mathbf{1}(s, t) = 1) \iff (s = t)$, which represents the constant sequence of the empty word.

The algorithm repeatedly adds new limit-words to $\mathcal{G}$. There are two ways for that: concatenating two limit-words in $\mathcal{G}$ or iterating an idempotent limit-word in $\mathcal{G}$.

Concatenation of two limit-words: The *concatenation* of two limit-words $\mathbf{u}$ and $\mathbf{v}$ is the limit-word $\mathbf{u} \cdot \mathbf{v}$ such that:

$$(\mathbf{u} \cdot \mathbf{v})(s, t) = 1 \iff \exists q \in Q, \mathbf{u}(s, q) = 1 \text{ and } \mathbf{v}(q, t) = 1 \ .$$

In other words, concatenation coincides with the multiplication of matrices with coefficients in the boolean semiring $(\{0, 1\}, \vee, \wedge)$. The concatenation of two limit-words intuitively corresponds to the concatenation of two sequences $(u_n)_{n \in \mathbb{N}}$ and $(v_n)_{n \in \mathbb{N}}$ of input words into the sequence $(u_n \cdot v_n)_{n \in \mathbb{N}}$. Note that the identity limit-word $\mathbf{1}$ is neutral for the concatenation.

Iteration of an idempotent limit-word: Intuitively, if a limit-word $\mathbf{u}$ represents a sequence $(u_n)_{n \in \mathbb{N}}$ then its iteration $\mathbf{u}^\sharp$ represents a sequence $\left(u_n^{f(n)}\right)_{n \in \mathbb{N}}$ for an increasing function $f : \mathbb{N} \to \mathbb{N}$.

The *iteration* $\mathbf{u}^\sharp$ of a limit-word $\mathbf{u}$ is only defined when $\mathbf{u}$ is idempotent *i.e* when $\mathbf{u} \cdot \mathbf{u} = \mathbf{u}$. It relies on the notion of $\mathbf{u}$-recurrent state.

Definition 7 ($\mathbf{u}$-recurrence)**.** *Let* $\mathbf{u}$ *be an idempotent limit-word. A state* s *is* $\mathbf{u}$-*recurrent if for every state* t,

$$\mathbf{u}(s, t) = 1 \implies \mathbf{u}(t, s) = 1 \ .$$

The *iterated limit-word* $\mathbf{u}^\sharp$ removes from $\mathbf{u}$ any edge that does not lead to a recurrent state:

$$\mathbf{u}^\sharp(s, t) = 1 \iff \mathbf{u}(s, t) = 1 \text{ and } t \text{ is } \mathbf{u}\text{-recurrent} \ .$$

This underlying idea is that iterating a great number of times the action $\mathbf{u}$, the transient states are left forever.

B. The Markov monoid and value 1 witnesses

The set $\mathcal{G}$ of limit-words computed by the Markov monoid algorithm is called the Markov monoid.

Definition 8 (Markov monoid)**.** *The Markov monoid is the smallest set of limit-words containing the set* $\{\mathbf{a} \mid a \in A\}$ *of limit-words induced by letters, the identity limit-word* $\mathbf{1}$*, and closed under concatenation and iteration.*

Two key properties, *consistency* and *completeness*, ensure that the limit-words of the Markov monoid reflect exactly every possible asymptotic effect of a sequence of input words.

Consistency ensures that every limit-word in $\mathcal{G}$ abstracts the asymptotic effect of an input sequence.

Definition 9 (Consistency)**.** *A set of limit-words* $\mathcal{G} \subseteq \{0, 1\}^{Q^2}$ *is* consistent *with a probabilistic automaton* $\mathcal{A}$ *if for each limit-word* $\mathbf{u} \in \mathcal{G}$*, there exists a sequence of input words* $(u_n)_{n \in \mathbb{N}}$ *such that for every states* $s, t \in Q$ *the sequence* $(u_n(s, t))_{n \in \mathbb{N}}$ *converges and:*

$$\mathbf{u}(s, t) = 1 \iff \lim_n u_n(s, t) > 0 \ . \qquad (2)$$

Conversely, completeness ensures that every input sequence reifies one of the limit-words.

Definition 10 (Completeness)**.** *A set of limit-words $\mathcal{G} \subseteq \{0,1\}^{Q^2}$ is* complete *for a probabilistic automaton $\mathcal{A}$ if for each sequence of input words $(u_n)_{n\in\mathbb{N}}$, there exists $\mathbf{u} \in \mathcal{G}$ such that for every states $s,t \in Q$:*

$$\limsup_n u_n(s,t) = 0 \implies \mathbf{u}(s,t) = 0 \ . \tag{3}$$

A limit-word may witness that the automaton has value 1.

Definition 11 (Value 1 witnesses)**.** *Let $\mathcal{A}$ be a probabilistic automaton. A* value 1 witness *is a limit-word $\mathbf{u}$ such that for every state $s \in Q$,*

$$\mathbf{u}(i,s) = 1 \implies s \in F \ . \tag{4}$$

Thanks to value 1 witnesses, the answer to the value 1 problem can be read in a consistent and complete set of limit-words:

Lemma 3 (A criterion for value 1)**.** *Let $\mathcal{A}$ be a probabilistic automaton and $\mathcal{G} \subseteq \{0,1\}^{Q^2}$ be a set of limit-words. Suppose that $\mathcal{G}$ is consistent with $\mathcal{A}$ and complete for $\mathcal{A}$. Then $\mathcal{A}$ has value 1 if and only if $\mathcal{G}$ contains a value 1 witness.*

Proof: Assume first that $\mathcal{A}$ has value 1. By definition, there exists a sequence $(u_n)_{n\in\mathbb{N}}$ of input words such that $\mathbb{P}_{\mathcal{A}}(u_n) = \sum_{f\in F} u_n(i,f) \xrightarrow[n]{} 1$. Since for all $n \in \mathbb{N}$, we have $\sum_{q\in Q} u_n(i,q) = 1$, then for all $s \notin F$, $u_n(i,s) \xrightarrow[n]{} 0$. Since $\mathcal{G}$ is complete, there exists a limit-word $\mathbf{u}$ such that (3) holds. Then $\mathbf{u}$ is a value 1 witness: for every $s \in Q$ such that $\mathbf{u}(i,s) = 1$, equation (3) implies $\limsup_n u_n(i,s) > 0$, hence $s \in F$.

Conversely, assume now that $\mathcal{G}$ contains a value 1 witness $\mathbf{u}$. Since $\mathcal{G}$ is consistent, there exists a sequence $(u_n)_{n\in\mathbb{N}}$ such that (2) holds. It follows from (2) and (4), that for all $s \notin F$, we have $u_n(i,s) \xrightarrow[n]{} 0$. Thus $\mathbb{P}_{\mathcal{A}}(u_n) = \sum_{f\in F} u_n(i,f) \xrightarrow[n]{} 1$ and $\mathcal{A}$ has value 1. ∎

The following theorem proves that the Markov monoid of a leaktight automaton is consistent and complete, thus according to Lemma 3 it can be used to decide the value 1 problem.

Theorem 2. *The Markov monoid associated with an automaton $\mathcal{A}$ is consistent. Moreover if $\mathcal{A}$ is leaktight then the Markov monoid is complete.*

The proof of the second part of this theorem relies on a subtle algebraic argument based on the existence of factorization forests of bounded height [22]. The same kind of argument was used by Simon to prove the decidability of the boundedness problem for distance automata [23].

The proof of completeness will follow from the lower bound lemma, which is the whole concern of Section V. For now we show that the Markov monoid is consistent.

Lemma 4 (Consistency)**.** *Let $\mathcal{G} \subseteq \{0,1\}^{Q^2}$ be a set of limit-words. Suppose that $\mathcal{G}$ is consistent. Then for every $\mathbf{u}, \mathbf{v} \in \mathcal{G}$ the set $\mathcal{G} \cup \{\mathbf{u}\cdot\mathbf{v}\}$ is consistent. If moreover $\mathbf{u}$ is idempotent then $\mathcal{G} \cup \{\mathbf{u}^\sharp\}$ is consistent as well.*

The proof uses the notion of reification.

Definition 12. *A sequence $(u_n)_{n\in\mathbb{N}}$ of input words reifies a limit-word $\mathbf{u}$ if for every states s,t the sequence $(u_n(s,t))_{n\in\mathbb{N}}$ converges and*

$$\mathbf{u}(s,t) = 1 \iff \lim_n u_n(s,t) > 0 \ . \tag{5}$$

In particular, a set of limit-words $\mathcal{G}$ is consistent for $\mathcal{A}$ if each limit-word in $\mathcal{G}$ is reified by some sequence of input words.

Proof: Let $\mathbf{u}, \mathbf{v} \in \mathcal{G}$. We build a sequence $(w_n)_{n\in\mathbb{N}}$ which reifies $\mathbf{u}\cdot\mathbf{v}$. By induction hypothesis on $\mathbf{u}$ and $\mathbf{v}$, there exists $(u_n)_n$ and $(v_n)_n$ which reify $\mathbf{u}$ and $\mathbf{v}$ respectively. Let $w_n = u_n \cdot v_n$. Then $(w_n)_{n\in\mathbb{N}}$ reifies $\mathbf{u}\cdot\mathbf{v}$, because

$$w_n(s,r) = \sum_{t\in Q} u_n(s,t)\cdot v_n(t,r)$$

and by definition of the concatenation of two limit-words.

Suppose now that $\mathbf{u}$ is idempotent, we build a sequence $(z_n)_{n\in\mathbb{N}}$ which reifies $\mathbf{u}^\sharp$. By induction hypothesis, there exists a sequence $(u_n)_{n\in\mathbb{N}}$ which reifies $\mathbf{u}$. For every states s,t we denote by $u(s,t)$ the value $\lim_n u_n(s,t)$. Since $\mathbf{u}$ is idempotent, the Markov chain $\mathcal{M}_u$ with state space Q and transition probabilities $(u(s,t))_{s,t\in Q}$ is 1-periodic thus aperiodic. According to standard results about finite Markov chains, the sequence of matrices $(u^k)_{k\in\mathbb{N}}$ has a limit $z \in [0,1]^{Q\times Q}$ such that transient states of $\mathcal{M}_u$ have no incoming edges in z. This implies:

$$\forall s,t \in Q,\ (z(s,t) > 0 \implies t \text{ is } z\text{-recurrent}) \ . \tag{6}$$

Since $(u_n)_{n\in\mathbb{N}}$ converges to u and by continuity of the matrix product, for every $k \in \mathbb{N}$ the sequence of matrices $(u_n^k)_{n\in\mathbb{N}}$ converges to u^k. It follows that there exists $\phi(k) \in \mathbb{N}$ such that $||u^k - u^k_{\phi(k)}||_\infty \leq \frac{1}{k}$. As a consequence the sequence of matrices $(z_n)_{n\in\mathbb{N}} = (u^n_{\phi(n)})_{n\in\mathbb{N}}$ converges to z.

Now we prove that $(z_n)_{n\in\mathbb{N}}$ reifies $\mathbf{u}^\sharp$ because,

$$\begin{aligned}
\mathbf{u}^\sharp(s,t) = 1 &\iff t \text{ is } \mathbf{u}\text{-recurrent and } \mathbf{u}(s,t) = 1 \\
&\iff t \text{ is } u\text{-recurrent and } u(s,t) > 0 \\
&\iff t \text{ is } z\text{-recurrent and } z(s,t) > 0 \\
&\iff z(s,t) > 0 \\
&\iff \lim_n z_n(s,t) > 0 \ ,
\end{aligned}$$

where the first equivalence is by definition of the iteration, the second holds because $(u_n)_{n\in\mathbb{N}}$ reifies $\mathbf{u}$, the third because the iterated Markov chain induced by $z = \lim_k u^k$ has the same recurrent states than the Markov chain $\mathcal{M}_u$, the fourth holds by (6), and the fifth by definition of z. ∎

C. Correctness of the Markov monoid algorithm

Proposition 1. *The Markov monoid algorithm solves the value 1 problem for leaktight automata.*

Proof: Termination of the Markov monoid algorithm is straightforward because each iteration adds a new element in $\mathcal{G}$ and there are at most $2^{|Q|^2}$ elements in $\mathcal{G}$.

The correctness is a corollary of Theorem 2: since the Markov monoid is consistent and complete then according to

Lemma 3, $\mathcal{A}$ has value 1 if and only if $\mathcal{G}$ contains a value 1 witness, if and only if the Markov monoid algorithm outputs "true". ■

In case the Markov monoid algorithm outputs "true", then for sure the input automaton has value 1. This positive result holds for every automaton, leaktight or not.

Proposition 2. *If the Markov monoid algorithm outputs "true", the input probabilistic automaton has value* 1.

Proof: According to Theorem 2, the Markov monoid is consistent. If it contains a value 1 witness, then according to the second part of the proof of Lemma 3, $\mathcal{A}$ has value 1. ■

In case the Markov monoid algorithm outputs "false" and the automaton is leaktight then the value of the automaton can be bounded from above:

Proposition 3. *Let $\mathcal{A}$ be a probabilistic automaton whose minimal non-zero transition probability is denoted $p_{\min}$. If the Markov monoid algorithm outputs "false" and if moreover $\mathcal{A}$ is leaktight, then val$(\mathcal{A}) \leq 1 - p_{\min}^{2^{3 \cdot J^2}}$, with $J = 2^{2|Q|^2}$.*

The proof of this proposition is a direct corollary of the lower bound lemma presented in Section V.

In case the Markov monoid algorithm outputs "false", one surely wishes to know whether the input automaton is leaktight or not. Fortunately, the leaktight property is decidable, as discussed in Section IV.

D. Complexity of the Markov monoid algorithm

Proposition 4. *The value* 1 *problem for leaktight automata is* PSPACE*-complete.*

The termination argument given above only implies an exponential-time algorithm. We improve this $\mathrm{EXPTIME}$ upper bound to PSPACE; for that we use the same arguments that Kirsten used to prove that limitedness of desert automata can be decided in PSPACE [16]. We avoid the explicit computation of the Markov monoid and look for value 1 witnesses in a non-deterministic way. The algorithm guesses non-deterministically the value 1 witness $\mathbf{u}$ and its decomposition by the product and iteration operations. The algorithm computes a $\sharp$-expression, *i.e* a finite tree with concatenation nodes of arbitrary degree on even levels and iteration nodes of degree one on odd levels and labelled consistently by limit-words. The depth of this tree is at most twice the $\sharp$-height (the number of nested applications of the iteration operation) plus one. The root of the $\sharp$-expression is labelled by $\mathbf{u}$ and the expression is computed non-deterministically from the root in a depth-first way.

For desert automata, the key observation made by Kirsten is that the $\sharp$-height is at most $|Q|$. The adaptation of Kirsten's proof to probabilistic automata is achieved by the two following lemmata:

Lemma 5. *Let $\mathbf{u}$ and $\mathbf{v}$ be two idempotent limit-words. Assume $\mathbf{u} \leq_{\mathcal{J}} \mathbf{v}$, then there are fewer non-trivial strongly connected component in $\mathbf{u}$ than in $\mathbf{v}$.*

Lemma 6. *Let $\mathbf{u}$ be an idempotent limit-word. The set of non-trivial strongly connected component of $\mathbf{u}$ is included in the set of non-trivial strongly connected component of $\mathbf{u}^\sharp$. Moreover if $\mathbf{u} \neq \mathbf{u}^\sharp$ this inclusion is strict.*

Since the number of non-trivial strongly connected component in a limit-word is bounded by $|Q|$, and if we require the iteration operation to be applied only to unstable idempotent, the $\sharp$-height of a $\sharp$-expression is bounded by $|Q|$ thus the depth of the expression is bounded by $2|Q| + 1$.

Consequently, the value 1 problem can be decided in PSPACE: to guess the value 1 witness, the non-deterministic algorithm needs to store at most $2|Q| + 1$ limit-words which can be done in space $\mathcal{O}(|Q|^2)$. Savitch's theorem implies that the deterministic complexity is PSPACE as well.

This PSPACE upper bound on the complexity is tight. The value 1 problem is known to be PSPACE-complete when restricted to $\sharp$-acyclic automata [14]. The same reduction to the PSPACE-complete problem of intersection of deterministic automata can be used to prove completeness of the value 1 problem for leaktight automata, relying on the facts that deterministic automata are leaktight (Proposition 5) and the class of leaktight automata is closed under parallel composition (Proposition 6). The completeness result is also a corollary of Proposition 5: since $\sharp$-acyclic automata are a subclass of leaktight automata, the decision problem is *a fortiori* complete for leaktight automata.

IV. Deciding whether an automaton is leaktight

At first sight, the decidability of the leaktight property is not obvious: to check the existence of a leak one would need to scan the uncountable set of all possible sequences of input words. Still:

Theorem 3. *The leaktight property is decidable in polynomial space.*

Algorithm 2 The leak-finder algorithm.

Input: A probabilistic automaton $\mathcal{A}$.
Output: Decide whether $\mathcal{A}$ is leaktight or not.

$\mathcal{G}_+ \leftarrow \{(\mathbf{a}, \mathbf{a}) \mid a \in A\} \cup \{(\mathbf{1}, \mathbf{1})\}$.
repeat
  if there is $(\mathbf{u}, \mathbf{u}_+), (\mathbf{v}, \mathbf{v}_+) \in \mathcal{G}_+$ such that $(\mathbf{u} \cdot \mathbf{u}, \mathbf{v}_+ \cdot \mathbf{v}_+) \notin \mathcal{G}_+$ **then**
    add $(\mathbf{u} \cdot \mathbf{v}, \mathbf{u}_+ \cdot \mathbf{v}_+)$ to $\mathcal{G}_+$
  if there is $(\mathbf{u}, \mathbf{u}_+) \in \mathcal{G}_+$ both idempotents such that $(\mathbf{u}^\sharp, \mathbf{u}_+) \notin \mathcal{G}_+$ **then**
    add $(\mathbf{u}^\sharp, \mathbf{u}_+)$ to $\mathcal{G}_+$
until there is nothing to add
if there is a leak witness in $\mathcal{G}_+$ **then**
  return false
else
  return true

The *leak-finder algorithm* deciding the leaktight property is very similar to the Markov monoid algorithm, except for two differences. First, the algorithm keeps track of those edges that are deleted by successive iteration operations. For that purpose, the algorithm stores together with each limit-word $\mathbf{u}$ another limit-word $\mathbf{u}_+$ to keep track of strictly positive transition probabilities. Second, the algorithm looks for *leak witnesses*.

Definition 13 (Extended limit-word)**.** *An extended limit-word is a pair of limit-words. The set of extended limit-words computed by the leak-finder algorithm is called the* extended Markov monoid.

The extended Markov monoid is indeed a monoid equipped with the component-wise concatenation operation:

$$(\mathbf{u}, \mathbf{u}_+) \cdot (\mathbf{v}, \mathbf{v}_+) = (\mathbf{u} \cdot \mathbf{v}, \mathbf{u}_+ \cdot \mathbf{v}_+) \ ,$$

It follows that an extended limit-word $(\mathbf{u}, \mathbf{u}_+)$ is idempotent if both $\mathbf{u}$ and $\mathbf{u}_+$ are idempotent.

Definition 14 (Leak witness)**.** *An extended limit-word $(\mathbf{u}, \mathbf{u}_+)$ is a* leak witness *if it is idempotent and there exists $r, q \in Q$ such that:*

1) *r is $\mathbf{u}$-recurrent,*
2) $\mathbf{u}_+(r, q) = 1$,
3) $\mathbf{u}(q, r) = 0$.

The correctness of the leak-finder algorithm is a consequence of:

Theorem 4. *An automaton $\mathcal{A}$ is leaktight if and only if its extended Markov monoid does not contain a leak witness.*

Although we chose to present Theorem 2 and Theorem 4 separately, their proofs are tightly linked.

As a consequence, the leaktight property is qualitative: it does not depend on the exact value of transition probabilities but only on their positivity.

V. The lower bound lemma

The lower bound lemma is the key to both our decidability result (via Proposition 3) and the characterization of leaktight automata (Theorem 4).

Lemma 7 (Lower bound lemma)**.** *Let $\mathcal{A}$ be a probabilistic automaton whose extended Markov monoid contains no leak witness. Let $p_{\min}$ the smallest non-zero transition probability of $\mathcal{A}$. Then for every word $u \in A^*$, there exists a pair $(\mathbf{u}, \mathbf{u}_+)$ in the extended Markov monoid such that, for all states $s, t \in Q$,*

$$\mathbf{u}_+(s, t) = 1 \iff u(s, t) > 0 \ , \tag{7}$$

$$\mathbf{u}(s, t) = 1 \implies u(s, t) \geq p_{\min}^{2^{3 \cdot J^2}} \ , \tag{8}$$

where $J = 2^{2|Q|^2}$.

To prove Lemma 7, we rely on the notion of Ramseyan factorization trees and decomposition trees introduced by Simon [22], [23].

Definition 15. *Let A be a finite alphabet, $(M, \cdot, 1)$ a monoid and $\phi : A^* \to M$ a morphism. A* Ramseyan factorization tree *of a word $u \in A^+$ for ϕ is a finite unranked ordered tree, whose nodes are labelled by pairs $(w, \phi(w))$ where w is a word in A^+ and such that:*

(i) *the root is labelled by $(u, \phi(u))$,*
(ii) *every internal node with two children labelled by $(u_1, \phi(u_1))$ and $(u_2, \phi(u_2))$ is labelled by $(u_1 \cdot u_2, \phi(u_1 \cdot u_2))$,*
(iii) *leaves are labelled by pairs $(a, \phi(a))$ with $a \in A$,*
(iv) *if an internal node t has three or more children $t_1, \ldots, t_n$ labelled by $(u_1, \phi(u_1)), \ldots, (u_n, \phi(u_n))$, then there exists $\mathbf{e} \in M$ such that $\mathbf{e}$ is idempotent and $\mathbf{e} = \phi(u_1) = \phi(u_2) = \ldots = \phi(u_n)$. In this case t is labelled by $(u_1 \cdots u_n, \mathbf{e})$.*

Internal nodes with one or two children are concatenation nodes, *the other internal nodes are* iteration nodes.

Not surprisingly, every word $u \in A^+$ can be factorized in a Ramseyan factorization tree, using only concatenation nodes: any *binary* tree whose leaves are labelled from left to right by the letters of u and whose internal nodes are labelled consistently is a Ramseyan factorization tree. Notice that if u has length n then such a tree has a height logarithmic in n, with the convention that the height of a leaf is 0. As a consequence, with this naïve factorization of u, the longer the word u, the deeper its factorization tree.

The following powerful result of Simon states that every word can be factorized with a Ramseyan factorization tree whose depth is bounded independently of the length of the word:

Theorem 5 ([4], [7], [22])**.** *Let $\mathcal{A}$ be a probabilistic automaton whose extended Markov monoid contains no leak witness. Every word $u \in A^+$ has a Ramseyan factorization tree of height at most $3 \cdot |M|$.*

In [23], Simon used the tropical semiring $(\mathbb{N} \cup \{\infty\}, \min, +)$ to prove the decidability of the boundedness problem for distance automata. Similarly to the Markov monoid, the tropical semiring is equipped with an iteration operation $\sharp$. Following the proof scheme of Simon, we introduce the notion of decomposition tree relatively to a monoid M equipped with an iteration operation $\sharp$.

Definition 16. *Let A be a finite alphabet, $(M, \cdot, 1)$ a monoid equipped with a function $\sharp$ that maps every idempotent $\mathbf{e} \in M$ to another idempotent element $\mathbf{e}^\sharp \in M$ and $\phi : A^* \to M$ a morphism. A* decomposition tree *of a word $u \in A^+$ is a finite unranked ordered tree, whose nodes have labels in (A^+, M) and such that:*

i) *the root is labelled by $(u, \mathbf{u})$, for some $\mathbf{u} \in M$,*
ii) *every internal node with two children labelled by $(u_1, \mathbf{u}_1)$ and $(u_2, \mathbf{u}_2)$ is labelled by $(u_1 \cdot u_2, \mathbf{u}_1 \cdot \mathbf{u}_2)$,*
iii) *every leaf is labelled by $(a, \mathbf{a})$ where a is a letter,*
iv) *for every internal node with three or more children, there exists $\mathbf{e} \in M$ such that $\mathbf{e}$ is idempotent and the node is labelled by $(u_1 \ldots u_n, \mathbf{e}^\sharp)$ and its children are labelled by $(u_1, \mathbf{e}), \ldots, (u_n, \mathbf{e})$.*

Internal nodes with one or two children are concatenation *nodes, the other internal nodes are* iteration *nodes.*

An iteration node labelled by $(u, \mathbf{e})$ *is* discontinuous *if* $\mathbf{e}^\sharp \neq \mathbf{e}$. *The* span *of a decomposition tree is the maximal length of a path that contains no discontinuous iteration node.*

Remark that decomposition and factorization trees are closely related:

Lemma 8. *A Ramseyan factorization tree is a decomposition tree if and only if it contains no discontinuous iteration nodes.*

Proof: The definitions 15 and 16 are similar except for condition iv). If there are no discontinuous nodes then $\mathbf{e} = \mathbf{e}^\sharp$ in iv) of Definition 16. ∎

The following theorem is adapted from [23, Lemma 10] and is a direct corollary of Theorem 5.

Theorem 6. *Let A be a finite alphabet, $(M, \cdot, 1)$ a monoid equipped with a function $\sharp$ that maps every idempotent $\mathbf{e} \in M$ to another idempotent element $\mathbf{e}^\sharp \in M$ and $\phi : A^* \to M$ a morphism. Every word $u \in A^+$ has a decomposition tree whose span is less than $3 \cdot |M|$.*

To obtain the lower bound lemma, we need to bound the depth of a decomposition tree; now that the span is bounded thanks to Theorem 6, we need to bound the number of discontinuous iteration nodes. Simon and Leung noticed that this number is actually bounded by the number of $\mathcal{J}$-classes in the monoid. The notion of $\mathcal{J}$-class of a monoid M is a classical notion in semigroup theory, derived from one of the four Green's relations called the $\mathcal{J}$-preorder: a $\mathcal{J}$-class is an equivalence class for this preorder (for details about Green's relations, see [15], [17]). The $\mathcal{J}$-preorder between elements of a monoid M is defined as follows:

$$\forall a, b \in M, a \leq_{\mathcal{J}} b \text{ if } a \in MbM \ ,$$

where MbM denotes the set $\{ubv \mid u, v \in M\}$.

The number of discontinuous nodes along a path in a decomposition tree can be bounded using the following result, adapted from [23, Lemma 3].

Lemma 9. *Let A be a finite alphabet, and M a monoid equipped with a function $\sharp$ that maps every idempotent $e \in M$ to another idempotent element $e^\sharp \in M$. Suppose moreover that for every idempotent $e \in M$,*

$$e^\sharp \cdot e = e^\sharp = e \cdot e^\sharp \ . \tag{9}$$

Then for every idempotent element $e \in M$, either $e^\sharp = e$ or $e^\sharp <_{\mathcal{J}} e$.

As a consequence, the number of discontinuous nodes along a path in a decomposition tree is at most J, where J is the number of $\mathcal{J}$-classes of the monoid.

Now we are ready to complete the proof of the lower bound lemma.

Proof of Lemma 7: Let M be the extended Markov monoid $\mathcal{G}_+$ associated with $\mathcal{A}$ and equipped with the concatenation operation:

$$(\mathbf{u}, \mathbf{u}_+) \cdot (\mathbf{v}, \mathbf{v}_+) = (\mathbf{u} \cdot \mathbf{v}, \mathbf{u}_+ \cdot \mathbf{v}_+) \ ,$$

and for idempotent pairs the iteration operation:

$$(\mathbf{u}, \mathbf{u}_+)^\sharp = (\mathbf{u}^\sharp, \mathbf{u}_+) \ .$$

Let $w \in A^+$. (The case of the empty word is easily settled, considering the extended limit-word $(\mathbf{1}, \mathbf{1})$.) We apply Theorem 6 to the word w, the extended Markov monoid $M = \mathcal{G}_+$ and the morphism $\phi : A \to M$ defined by $\phi(a) = (\mathbf{a}, \mathbf{a})$. According to Theorem 6, w has a decomposition tree T of span less than $3 \cdot |\mathcal{G}_+|$, whose root is labelled by $(w, (\mathbf{w}, \mathbf{w}_+))$ for some extended limit-word $(\mathbf{w}, \mathbf{w}_+) \in \mathcal{G}_+$.

According to the second part of Lemma 9, and since there are less $\mathcal{J}$-classes than there are elements in the monoid $\mathcal{G}_+$,

$$\text{the depth of } T \text{ is at most } 3 \cdot |\mathcal{G}_+|^2. \tag{10}$$

To complete the proof of Lemma 7, we prove that for every node t labelled $(u, (\mathbf{u}, \mathbf{u}_+))$ of depth h in the decomposition tree and for all states $s, t \in Q$,

$$\mathbf{u}_+(s, t) = 1 \iff u(s, t) > 0 \ , \tag{11}$$

$$\mathbf{u}(s, t) = 1 \implies u(s, t) \geq p_{\min}^{2^h} \ . \tag{12}$$

We prove (11) and (12) by induction on h.

If $h = 0$ then the node is a leaf, hence u is a letter a and $\mathbf{u} = \mathbf{u}_+ = \mathbf{a}$. Then (11) holds by definition of $\mathbf{a}$ and (12) holds by definition of $p_{\min}$.

For the induction, there are two cases.

First case: t is a concatenation node labelled by $(u, (\mathbf{u}, \mathbf{u}_+))$ with two sons labelled by $(u_1, (\mathbf{u}_1, \mathbf{u}_{+,1}))$ and $(u_2, (\mathbf{u}_2, \mathbf{u}_{+,2}))$.

We first prove that (11) holds. Let s, t such that $\mathbf{u}_+(s, t) = 1$. By definition of a decomposition tree, $\mathbf{u}_+ = \mathbf{u}_{+,1} \cdot \mathbf{u}_{+,2}$. Since $\mathbf{u}_+(s, t) = 1$ then by definition of the concatenation there exists $q \in Q$ such that $\mathbf{u}_{+,1}(s, q) = 1$ and $\mathbf{u}_{+,2}(q, t) = 1$. By induction hypothesis we have $u_1(s, q) \cdot u_2(q, t) > 0$; and since $u = u_1 \cdot u_2$ then $u(s, t) \geq u_1(s, q) \cdot u_2(q, t)$, which proves the direct implication of (11).

The converse implication is similar: if $u(s, t) > 0$ then by definition of matrix product, there exists $q \in Q$ such that $u_1(s, q) > 0$ and $u(q, t) > 0$, and we use the induction hypothesis to get $\mathbf{u}_+(s, t) = 1$. This concludes the proof of (11).

Now we prove that (12) holds. Let $s, t \in Q$ such that $\mathbf{u}(s, t) = 1$. By definition of a decomposition tree, $\mathbf{u} = \mathbf{u}_1 \cdot \mathbf{u}_2$. Since $\mathbf{u}(s, t) = 1$ then by definition of the product of two limit-words there exists $q \in Q$ such that $\mathbf{u}_1(s, q) = 1$ and $\mathbf{u}_2(q, t) = 1$. Then $u(s, t) \geq u_1(s, q) \cdot u_2(q, t) \geq p_{\min}^{2^h} \cdot p_{\min}^{2^h} = p_{\min}^{2^{h+1}}$ where the first inequality is by definition of the matrix product and the second inequality is by induction hypothesis. This completes the proof of (12).

Second case: t is an iteration node labelled by $(u, (\mathbf{u}^\sharp, \mathbf{u}_+))$ with k sons $t_1, \ldots, t_k$ labelled by $(u_1, (\mathbf{u}, \mathbf{u}_+)), \ldots, (u_k, (\mathbf{u}, \mathbf{u}_+))$. The proof that (11) holds is similar to the concatenation node case (and relies on the fact that $\mathbf{u}_+$ is idempotent). We focus on the proof of (12). Let $s, r \in Q$ such that $\mathbf{u}^\sharp(s, r) = 1$. By definition of a decomposition tree, $\mathbf{u} = \mathbf{u}_1 \cdots \mathbf{u}_k$. Since t is an iteration node, $k \geq 3$ thus:

$$u(s,r) \geq u_1(s,r) \cdot \sum_{q \in Q} \left((u_2 \cdots u_{k-1})(r,q) \cdot u_k(q,r) \right) \quad . \tag{13}$$

To establish (12) we prove that:

$$u_1(s,r) \geq p_{\min}^{2^h}, \tag{14}$$

$$\forall q \in Q, (u_2 \cdots u_{k-1})(r,q) > 0 \implies u_k(q,r) \geq p_{\min}^{2^h}. \tag{15}$$

First we prove (14). Since $\mathbf{u}^\sharp(s,r) = 1$ then by definition of the iteration operation, r is $\mathbf{u}$-recurrent and $\mathbf{u}(s,r) = 1$. By induction hypothesis applied to t_1, according to (12), it implies $u_1(s,r) \geq p_{\min}^{2^h}$ *i.e* (14).

Now we prove (15). For that we use the hypothesis that $(\mathbf{u}, \mathbf{u}_+)$ is not a leak witness. Let $q \in Q$ such that $(u_2 \cdots u_{k-1})(r,q) > 0$. Then by induction hypothesis applied to $t_2, \ldots, t_{k-1}$, according to (11), $\mathbf{u}_+^{k-2}(r,q) = 1$. Thus by idempotency of $\mathbf{u}_+$, $\mathbf{u}_+(r,q) = 1$. Since by hypothesis $\mathbf{u}^\sharp(s,r) = 1$ then r is $\mathbf{u}$-recurrent and since $(\mathbf{u}, \mathbf{u}_+)$ is not a leak witness then necessarily $\mathbf{u}(q,r) = 1$. Thus, by induction hypothesis and according to (12), $u_k(q,r) \geq p_{\min}^{2^h}$ *i.e* (15).

Now, putting (13), (14) and (15) altogether,

$$\begin{aligned} u(s,r) &\geq u_1(s,r) \cdot \sum_{q \in Q} (u_2 \cdots u_{k-1})(r,q) \cdot u_k(q,r) \\ &\geq p_{\min}^{2^h} \cdot \sum_{q \in Q} (u_2 \cdots u_{k-1})(r,q) \cdot p_{\min}^{2^h} \\ &\geq p_{\min}^{2^{h+1}}, \end{aligned}$$

where the second inequality holds because $\sum_{q \in Q} (u_2 \cdots u_{k-1})(r,q) = 1$ by basic property of transition matrices. This completes the proof of (12).

To conclude, according to (10) the depth of a decomposition tree can be bounded by $3 \cdot |\mathcal{G}_+|^2$, and since $\mathcal{G}_+$ has less than $J = 2^{2|Q|^2}$ elements the depth h is less than $3 \cdot J^2$. Then according to (11) and (12) this completes the proof of Lemma 7. ■

VI. A few leaktight automata

In this section, we present several properties and examples of leaktight automata.

A. Two basic examples

The automaton on Fig. 2 is leaktight. Its extended Markov monoid is depicted on the right-hand side (except for the neutral element $(\mathbf{1}, \mathbf{1})$). Each of the four directed graphs represents an extended limit-word $(\mathbf{u}, \mathbf{u}_+)$; the edges marked $+$ are the edges that are in $\mathbf{u}_+$ but not in $\mathbf{u}$.

The initial state of the automaton is state 0, and the unique final state is state 1. This automaton has value 1 and this can be checked using its Markov monoid: there is a single value 1 witness $\mathbf{a}^\sharp$, which correspond to two distinct extended limit-words labelled by $\mathbf{a}^\sharp$ and $\mathbf{b} \cdot \mathbf{a}^\sharp$ on Fig. 2.

The automaton on Fig 3 is leaktight. The initial state of the automaton is state 0, and the unique final state is state F.

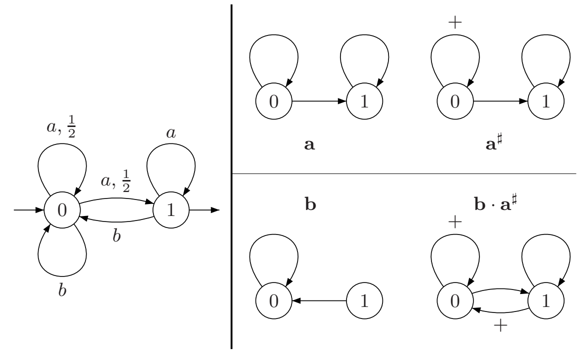

Fig. 2. A leaktight automaton and its extended Markov monoid.

The Markov monoid has too many elements to be represented here. This automaton does not have value 1.

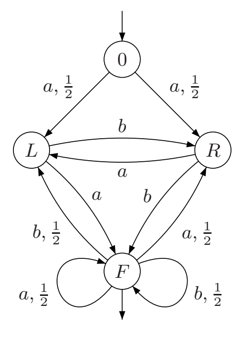

Fig. 3. A leaktight automaton which does not have value 1.

B. The class of leaktight automata is rich and stable

The class of leaktight automata contains all known classes of probabilistic automata with a decidable value 1 problem, in particular hierarchical automata defined in [3] and $\sharp$-acyclic automata defined in [14].

Proposition 5. *Deterministic automata, hierarchical probabilistic automata and $\sharp$-acyclic automata are leaktight and these inclusions are strict.*

Another interest of the class of leaktight automata is its stability under two natural composition operators: parallel composition and synchronized product. An automaton $\mathcal{A}||\mathcal{B}$ is the parallel composition of two automata $\mathcal{A}$ and $\mathcal{B}$ if its state space is the disjoint union of the state spaces of $\mathcal{A}$ and $\mathcal{B}$ plus a new initial state. For every input letter, the possible successors of the initial state are itself or one of the initial state of $\mathcal{A}$ and $\mathcal{B}$. An automaton $\mathcal{A} \times \mathcal{B}$ is the synchronized product of two automata $\mathcal{A}$ and $\mathcal{B}$ if its state space is the cartesian product of the state spaces of $\mathcal{A}$ and $\mathcal{B}$, with induced transition probabilities.

Proposition 6. *The leaktight property is stable by parallel composition and synchronized product.*

C. About $\sharp$-height

The $\sharp$-height of an automaton is the minimal number of nested applications of the iteration operator needed to obtain a value 1 witness, if there is one. As already mentioned, an adaptation of a result by Kirsten (Lemma 5.7 in [16]) shows that the $\sharp$-height of an automaton is at most $|Q|$. A natural question is whether this bound is tight. The answer is positive: the only value 1 witness of the automaton of Fig. 4 is $\mathbf{u} = (\cdots((\mathbf{a}_0^\sharp \mathbf{a}_1)^\sharp \mathbf{a}_2)^\sharp \mathbf{a}_3)^\sharp \cdots \mathbf{a}_{n-1})^\sharp$, whose $\sharp$-height is $n = |Q| - 2$.

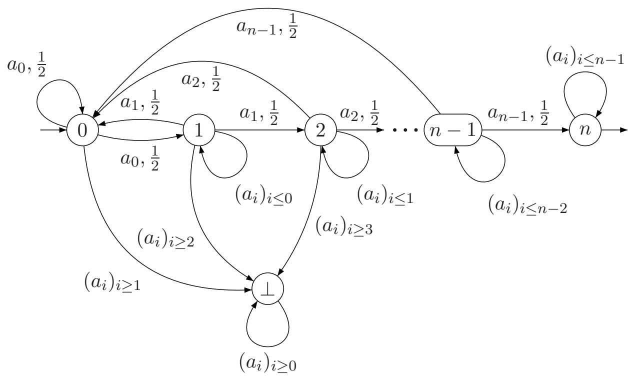

Fig. 4. A leaktight automaton with value 1 and $\sharp$-height n.

The above study of $\sharp$-height shows a crucial difference between leaktight automata and $\sharp$-acyclic automata, as it is easy to see that $\sharp$-acyclic automata have $\sharp$-height one.

Conclusion

We introduced a subclass of probabilistic automata, called leaktight automata, for which we proved that the value 1 problem is PSPACE-complete.

In the present paper we considered automata over finite words. Next step is the adaptation of our results to infinite words and probabilistic Büchi automata [1], [3], as well as partially observable Markov decision processes.

A natural question is "what does the Markov monoid say about a probabilistic automaton (leaktight or not)?". Since the Markov monoid is independent of the actual values of transition probabilities (only positivity matters), this suggests the two following questions. Given a probabilistic automaton whose transition probabilities are unspecified (only positivity is specified),

1) is it decidable whether the answer to the value 1 problem is the same for *any* choice of transition probabilities?
2) does the Markov monoid contain a value 1 witness if and only if the automaton has value 1 for *some* choice of transition probabilities?

The first question, suggested by a referee of the present paper, is open, while the answer to the second question seems to be negative.

Acknowledgment

We thank Thomas Colcombet for having pointed us to the work of Leung and Simon, as well as two referees for their careful reading and their constructive comments and help in improving this paper.

References

[1] Christel Baier, Nathalie Bertrand, and Marcus Größer. On decision problems for probabilistic Büchi automata. In *Foundations of Software Science and Computation Structures*, pages 287–301, 2008.

[2] 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.

[3] Rohit Chadha, A. Prasad Sistla, and Mahesh Viswanathan. Power of randomization in automata on infinite strings. In *International Conference on Concurrency Theory*, pages 229–243, 2009.

[4] Jérémie Chalopin and Hing Leung. On factorization forests of finite height. *Theoretical Computer Science*, 310(1-3):489–499, 2004.

[5] 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.

[6] Krishnendu Chatterjee and Mathieu Tracol. Decidable problems for probabilistic automata on infinite words. In *Logics In Computer Science*, 2012.

[7] Thomas Colcombet. Factorization forests for infinite words and applications to countable scattered linear orderings. *Theoretical Computer Science*, 411(4-5):751–764, 2010.

[8] 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.

[9] Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. L_p distance and equivalence of probabilistic automata. *International Journal of Foundations of Computer Science*, 18(4):761–779, 2007.

[10] Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. On the computation of the relative entropy of probabilistic automata. *International Journal of Foundations of Computer Science*, 19(1):219–242, 2008.

[11] Karel Culik and Jarkko Kari. *Digital images and formal languages*, pages 599–616. Springer-Verlag New York, Inc., 1997.

[12] 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.

[13] Nathanaël Fijalkow, Hugo Gimbert, and Youssouf Oualhadj. A class of probabilistic automata with a decidable value 1 problem. *CoRR*, abs/1104.3055, 2011.

[14] Hugo Gimbert and Youssouf Oualhadj. Probabilistic automata on finite words: Decidable and undecidable problems. In *International Colloquium on Automata, Languages and Programming*, pages 527–538, 2010.

[15] John M. Howie. *Fundamentals of semigroup theory*. Clarendon Press, Oxford, 1995.

[16] Daniel Kirsten. Distance desert automata and the star height problem. *Informatique Théorique et Applications*, 39(3):455–509, 2005.

[17] Gérard Lallement. *Semigroups and Combinatorial Applications*. Wiley, 1979.

[18] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23:269–311, June 1997.

[19] Azaria Paz. *Introduction to probabilistic automata*. Academic Press, 1971.

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

[21] Marcel-Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4, 1961.

[22] Imre Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990.

[23] Imre Simon. On semigroups of matrices over the tropical semiring. *Informatique Théorique et Applications*, 28(3-4):277–294, 1994.

[24] Wen-Guey Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM Journal on Computing*, 21(2):216–227, 1992.