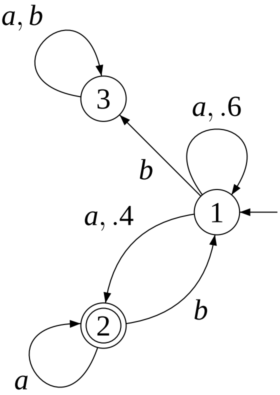
Deciding the value 1 problem for probabilistic leaktight automata

Nathanaël Fijalkow,
joint work with Hugo Gimbert and Youssouf Oualhadj

LIAFA, Université Paris 7, France,
University of Warsaw, Poland.

LICS, Dubrovnik, Croatia
June 26th 2012

Probabilistic automata (Rabin, 1963)



$$\mathbb{P}_{\mathcal{A}} : A^* \to [0, 1]$$

The value 1 problem

The value 1 problem is, given a probabilistic automaton $\mathcal{A}$:

“are there words accepted by $\mathcal{A}$ with arbitrarily high probability?”

The value 1 problem

The value 1 problem is, given a probabilistic automaton $\mathcal{A}$:

"are there words accepted by $\mathcal{A}$ with arbitrarily high probability?"

Define $\mathrm{val}(\mathcal{A}) = \sup_w \mathbb{P}_{\mathcal{A}}(w)$. An equivalent formulation of this problem is:

$$\text{“}\mathrm{val}(\mathcal{A}) \stackrel{?}{=} 1\text{”}.$$

The value 1 problem

The value 1 problem is, given a probabilistic automaton $\mathcal{A}$:

"are there words accepted by $\mathcal{A}$ with arbitrarily high probability?"

Define $\mathrm{val}(\mathcal{A}) = \sup_w \mathbb{P}_{\mathcal{A}}(w)$. An equivalent formulation of this problem is:

$$\text{“}\mathrm{val}(\mathcal{A}) \stackrel{?}{=} 1\text{”}.$$

Theorem (Gimbert, Oualhadj, 2010)

The value 1 problem is undecidable.

Our objective

Decide the value 1 problem for a *subclass* of probabilistic automata, by **algebraic** and **non-numerical** means.

Our objective

Decide the value 1 problem for a *subclass* of probabilistic automata, by **algebraic** and **non-numerical** means.

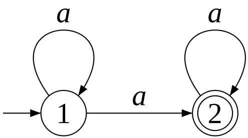
- **algebraic:** focus on the automaton structure,
- **non-numerical:** abstract away the values.

Our objective

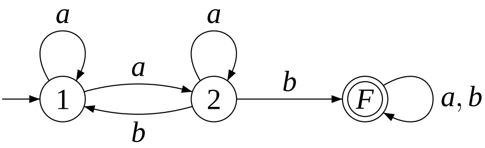
Decide the value 1 problem for a *subclass* of probabilistic automata, by **algebraic** and **non-numerical** means.

- **algebraic:** focus on the automaton structure,
- **non-numerical:** abstract away the values.

Hence we consider non-deterministic automata:



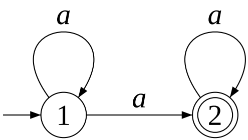
Weighted automata using algebra (Schützenberger)



$$\langle a \rangle = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \langle b \rangle = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

$I \cdot \langle abba \rangle \cdot F = 1$ if and only if $\mathbb{P}_{\mathcal{A}}(abba) > 0$

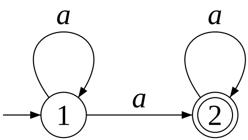
The stabilization operation $\sharp$



$$\langle a \rangle = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

In $\langle a \rangle$, the state 1 is transient and the state 2 is recurrent.

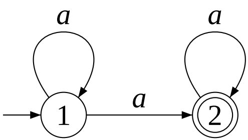
The stabilization operation $\sharp$



$$\langle a \rangle = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad \langle a^\sharp \rangle = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

In $\langle a \rangle$, the state 1 is transient and the state 2 is recurrent.

The stabilization operation $\sharp$



$$\langle a \rangle = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad \langle a^\sharp \rangle = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

In $\langle a \rangle$, the state 1 is transient and the state 2 is recurrent.

$$\text{“ } M^\sharp = \lim_n M^n \text{ ”}$$

A saturation algorithm

Compute a monoid inside the **finite** monoid $\mathcal{M}_{Q\times Q}(\{0,1\},+,\times)$.

- Compute $\langle a \rangle$ for $a \in A$
- Close under product and stabilization.

A saturation algorithm

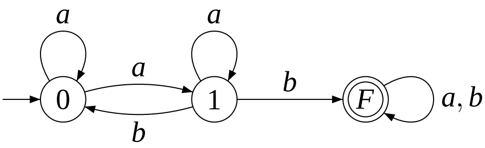
Compute a monoid inside the **finite** monoid $\mathcal{M}_{Q\times Q}(\{0,1\},+,\times)$.

- Compute $\langle a \rangle$ for $a \in A$
- Close under product and stabilization.
- If there exists a matrix M such that

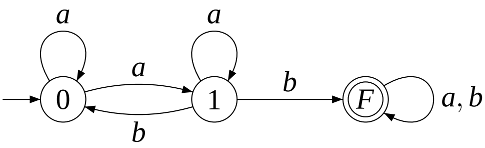
$$\forall t \in Q, \quad M(s_0,t) = 1 \Rightarrow t \in F$$

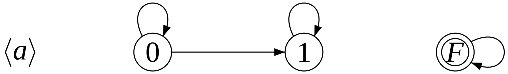
then “$\mathcal{A}$ has value 1”, otherwise “$\mathcal{A}$ does not have value 1”.

An example

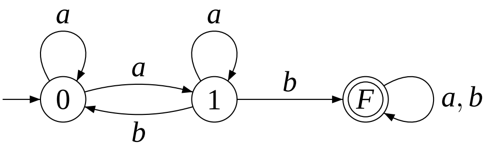


An example

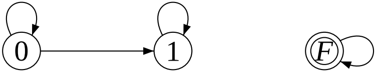




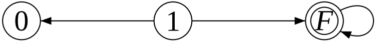
An example



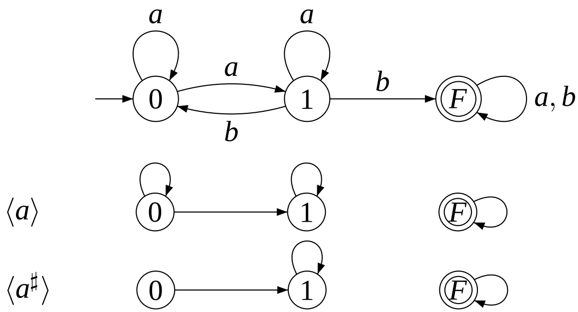
$\langle a \rangle$



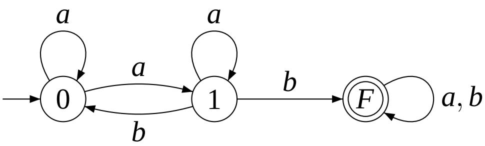
$\langle b \rangle$

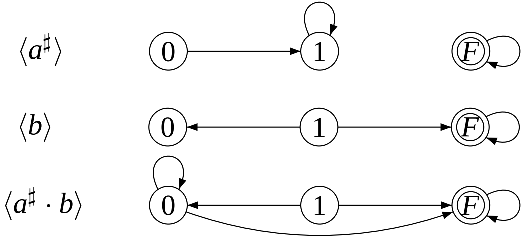


An example

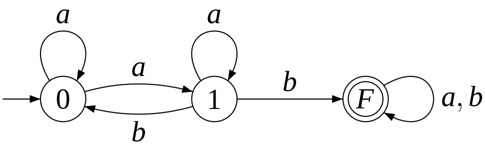


An example

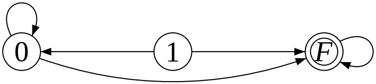

a
a
a
b
b
0
1
F
a, b


$\langle a^\sharp \rangle$
$\langle b \rangle$
$\langle a^\sharp \cdot b \rangle$
0
1
F

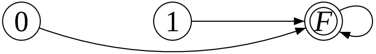
An example



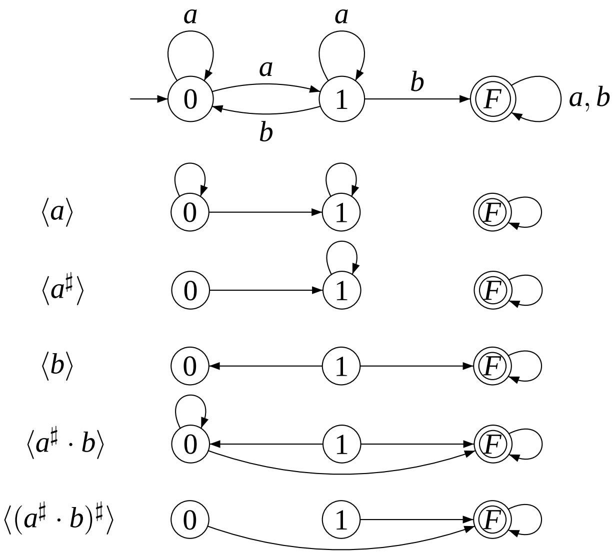
$\langle a^\sharp \cdot b \rangle$



$\langle (a^\sharp \cdot b)^\sharp \rangle$



An example



Correct, but not complete

Theorem (Correctness)

If the algorithm answers “$\mathcal{A}$ has value 1” then $\mathcal{A}$ has value 1.

Correct, but not complete

Theorem (Correctness)

If the algorithm answers "$\mathcal{A}$ has value 1" then $\mathcal{A}$ has value 1.

But the value 1 problem is undecidable, so the converse cannot hold!

Completeness in the absence of leaks

Definition

An automaton $\mathcal{A}$ is leaktight if it has no leak.

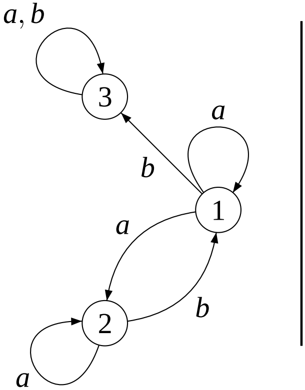
Theorem (Completeness)

If $\mathcal{A}$ is leaktight and has value 1,

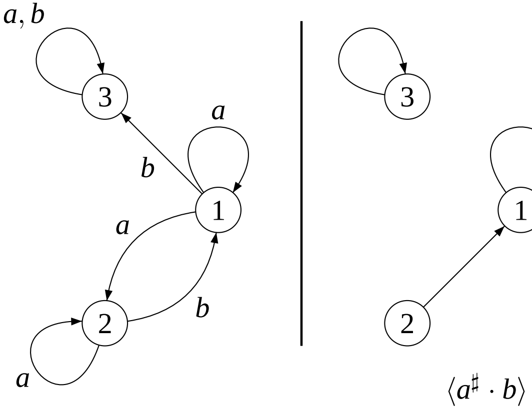
then the algorithm answers "$\mathcal{A}$ has value 1".

The proof relies on Simon's factorization forest theorem.

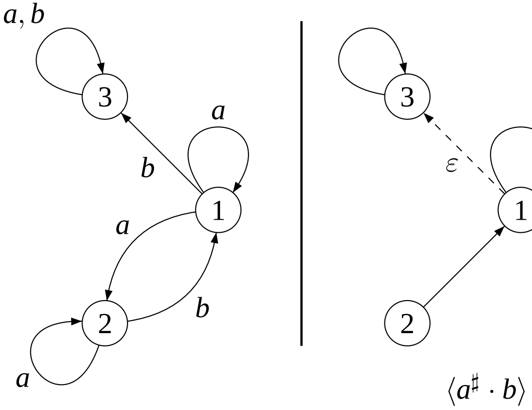
A leak



A leak



A leak



There is a leak from 1 to 3.

Conclusion and perspectives

- We defined a subclass of probabilistic automata which subsumes all subclasses of probabilistic automata whose value 1 problem is known to be decidable,

Conclusion and perspectives

- We defined a subclass of probabilistic automata which subsumes all subclasses of probabilistic automata whose value 1 problem is known to be decidable,

- We defined an algebraic algorithm for the value 1 problem and proved its completeness for the class of leaktight automata.

Conclusion and perspectives

- We defined a subclass of probabilistic automata which subsumes all subclasses of probabilistic automata whose value 1 problem is known to be decidable,

- We defined an algebraic algorithm for the value 1 problem and proved its completeness for the class of leaktight automata.

- What does this algorithm actually compute?

Conclusion and perspectives

- We defined a subclass of probabilistic automata which subsumes all subclasses of probabilistic automata whose value 1 problem is known to be decidable,
- We defined an algebraic algorithm for the value 1 problem and proved its completeness for the class of leaktight automata.
- What does this algorithm actually compute?
- Can we use similar algorithms for other semirings?