Algorithms for Probabilistic Automata that Use Algebra

Séminaire 68NQRT, Rennes

Nathanaël Fijalkow

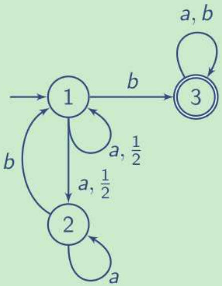
LIAFA, Université Denis Diderot - Paris 7, France
Institute of Informatics, Warsaw University, Poland
nath@liafa.univ-paris-diderot.fr

January 24th, 2013

Previously on 68NQRT: Youssouf Oualhadj

Probabilistic Automata

Example

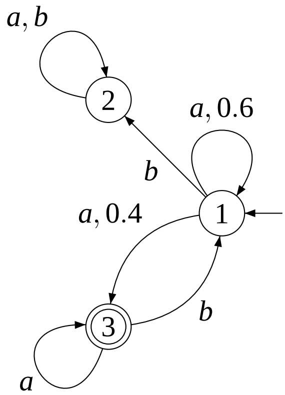


- 1 is the initial state.
- $\{3\}$ is the set of accepting states
- $\mathbb{P}_{\mathcal{A}}(aab) = \frac{1}{4}$ the acceptance probability.

Definition (Rabin 63)

A probabilistic automaton is a tuple $\mathcal{A} = (Q, A, (M_a)_{a \in A}, q_0, F)$.

Probabilistic automata (Rabin, 1963)



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

$\mathbb{P}_{\mathcal{A}}(w)$ is the probability that a run for w ends up in F

Decision problems

The equivalence problem:

INPUT: $\mathcal{A}, \mathcal{B}$ two probabilistic automata
OUTPUT: for all words $w \in A^*$, we have $\mathbb{P}_{\mathcal{A}}(w) = \mathbb{P}_{\mathcal{B}}(w)$.

The emptiness problem:

INPUT: $\mathcal{A}$ a probabilistic automaton and a threshold $\lambda \in \mathbb{Q}$
OUTPUT: there exists a word $w \in A^*$ such that $\mathbb{P}_{\mathcal{A}}(w) \geq \lambda$.

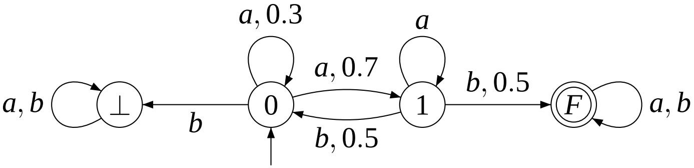
The isolation problem:

INPUT: $\mathcal{A}$ a probabilistic automaton and a threshold $\lambda \in \mathbb{Q}$
OUTPUT: there exists $\varepsilon > 0$ such that for all words $w \in A^*$, we have $\mathbb{P}_{\mathcal{A}}(w) \notin [\lambda - \varepsilon; \lambda + \varepsilon]$.

This talk

This talk is about algorithms based on *algebra*.

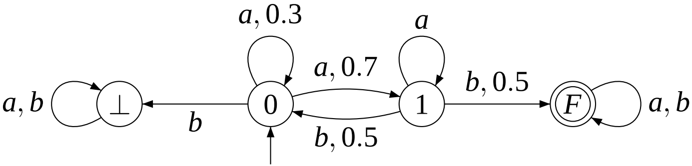
Weighted automata using algebra (Schützenberger)



$$\langle a \rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.3 & 0.7 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \langle b \rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$I = \begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix} \qquad F = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Weighted automata using algebra (Schützenberger)



$$\langle a \rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.3 & 0.7 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \langle b \rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbb{P}_{\mathcal{A}}(aaabaa) = I \cdot \langle aaabaa \rangle \cdot F = I \cdot \langle a \rangle \cdot \langle a \rangle \cdot \langle a \rangle \cdot \langle b \rangle \cdot \langle a \rangle \cdot \langle a \rangle \cdot F$$

Outline

Decidability

INPUT: $\mathcal{A}, \mathcal{B}$ two probabilistic automata
OUTPUT: for all words $w \in A^*$, we have $\mathbb{P}_{\mathcal{A}}(w) = \mathbb{P}_{\mathcal{B}}(w)$.

Theorem

The equivalence problem is decidable in polynomial time.

Decidability

INPUT: $\mathcal{A}, \mathcal{B}$ two probabilistic automata
OUTPUT: for all words $w \in A^*$, we have $\mathbb{P}_{\mathcal{A}}(w) = \mathbb{P}_{\mathcal{B}}(w)$.

Theorem

The equivalence problem is decidable in polynomial time.

- Schützenberger (1961) – minimization of weighted automata
- Tzeng (1992) – backward algorithm
- Doyen, Henzinger and Raskin (2008) – algebraic forward algorithm
- Kiefer, Murawski, Ouaknine, Wachter and Worrell (2010) – randomized NC

A simple algebraic algorithm

A probabilistic distribution is $\delta : Q \to [0,1]$ which sums up to 1.

Definition (Bisimilarity over distributions)

Two probabilistic distributions δ_1 and δ_2 are bisimilar if for all words $w \in A^*$, we have

$$\mathbb{P}^{\delta_1}_{\mathcal{A}}(w) = \mathbb{P}^{\delta_2}_{\mathcal{A}}(w)\ .$$

A simple algebraic algorithm

A probabilistic distribution is $\delta : Q \to [0,1]$ which sums up to 1.

Definition (Bisimilarity over distributions)

Two probabilistic distributions δ_1 and δ_2 are bisimilar if for all words $w \in A^*$, we have

$$\mathbb{P}_{\mathcal{A}}^{\delta_1}(w) = \mathbb{P}_{\mathcal{A}}^{\delta_2}(w) \ .$$

The equivalence problem reduces to the bisimilarity problem of two probabilistic distributions.

Checking bisimilarity

Two probabilistic distributions δ_1 and δ_2 are bisimilar if for all words $w \in A^*$, we have

$$\mathbb{P}_{\mathcal{A}}^{\delta_1}(w) = \mathbb{P}_{\mathcal{A}}^{\delta_2}(w) \ .$$

Equivalently:
$$\begin{cases} \mathbb{P}_{\mathcal{A}}^{\delta_1}(\varepsilon) = \mathbb{P}_{\mathcal{A}}^{\delta_2}(\varepsilon) \\ \mathbb{P}_{\mathcal{A}}^{\delta_1}(a) = \mathbb{P}_{\mathcal{A}}^{\delta_2}(a) \\ \mathbb{P}_{\mathcal{A}}^{\delta_1}(b) = \mathbb{P}_{\mathcal{A}}^{\delta_2}(b) \\ \mathbb{P}_{\mathcal{A}}^{\delta_1}(aa) = \mathbb{P}_{\mathcal{A}}^{\delta_2}(aa) \\ \ldots \end{cases}$$

Checking bisimilarity

Two probabilistic distributions δ_1 and δ_2 are bisimilar if for all words $w \in A^*$, we have

$$\mathbb{P}^{\delta_1}_{\mathcal{A}}(w) = \mathbb{P}^{\delta_2}_{\mathcal{A}}(w) \ .$$

Equivalently: $$\begin{cases} \mathbb{P}^{\delta_1}_{\mathcal{A}}(\varepsilon) = \mathbb{P}^{\delta_2}_{\mathcal{A}}(\varepsilon) \\ \mathbb{P}^{\delta_1}_{\mathcal{A}}(a) = \mathbb{P}^{\delta_2}_{\mathcal{A}}(a) \\ \mathbb{P}^{\delta_1}_{\mathcal{A}}(b) = \mathbb{P}^{\delta_2}_{\mathcal{A}}(b) \\ \mathbb{P}^{\delta_1}_{\mathcal{A}}(aa) = \mathbb{P}^{\delta_2}_{\mathcal{A}}(aa) \\ \dots \end{cases}$$

Let $N = |Q|$.

- each line is a linear equation involving N variables;
- there are at most N independent linear equations;
- no independent equations are added for words of length $\geq N$.

Linear algebra and dimensions

Denote X and Y the vectors for δ_1 and δ_2.

$$\begin{cases} {}^t(X - Y) \cdot F = 0 \\ {}^t(X - Y) \cdot \langle a \rangle \cdot F = 0 \\ {}^t(X - Y) \cdot \langle b \rangle \cdot F = 0 \\ {}^t(X - Y) \cdot \langle aa \rangle \cdot F = 0 \\ \dots \end{cases}$$

Linear algebra and dimensions

Denote X and Y the vectors for δ_1 and δ_2.

$$\begin{cases} {}^t(X-Y)\cdot F = 0 \\ {}^t(X-Y)\cdot \langle a\rangle \cdot F = 0 \\ {}^t(X-Y)\cdot \langle b\rangle \cdot F = 0 \\ {}^t(X-Y)\cdot \langle aa\rangle \cdot F = 0 \\ \ldots \end{cases}$$

Denote $W_k = \langle \{\langle w\rangle \cdot F \mid w \in A^{\leq k}\}\rangle$, generated vector space in $\mathbb{R}^N$.
The solutions live in $W_k^\perp$, orthogonal subspace of W_k.

Linear algebra and dimensions

Denote X and Y the vectors for δ_1 and δ_2.

$$\begin{cases} {}^t(X - Y) \cdot F = 0 \\ {}^t(X - Y) \cdot \langle a \rangle \cdot F = 0 \\ {}^t(X - Y) \cdot \langle b \rangle \cdot F = 0 \\ {}^t(X - Y) \cdot \langle aa \rangle \cdot F = 0 \\ \dots \end{cases}$$

Denote $W_k = \langle \{ \langle w \rangle \cdot F \mid w \in A^{\leq k} \} \rangle$, generated vector space in $\mathbb{R}^N$.
The solutions live in $W_k^{\perp}$, orthogonal subspace of W_k.

$$W_0 \subseteq W_1 \subseteq W_2 \subseteq \dots \ .$$

- For all k, if $W_k = W_{k+1}$, then $W_{k+1} = W_{k+2}$;
- Reasoning on dimensions shows that the sequence stabilizes before N steps.

Conclusion for the equivalence problem

With basic algebraic arguments we get a very simple algorithm checking the equivalence of two probabilistic automata, which runs in cubic time.

Bottom line: $(\mathbb{R}, +, \times)$ is a field, hence linear algebra comes into play!

Outline

Undecidability

INPUT: $\mathcal{A}$ a probabilistic automaton and a threshold $\lambda \in \mathbb{Q}$
OUTPUT: there exists $\varepsilon > 0$ such that for all words $w \in A^*$, we have $\mathbb{P}_{\mathcal{A}}(w) \notin [\lambda - \varepsilon; \lambda + \varepsilon]$.

Theorem (Bertoni, 1974)

The isolation problem is undecidable for $0 < \lambda < 1$.

Undecidability

INPUT: $\mathcal{A}$ a probabilistic automaton and a threshold $\lambda \in \mathbb{Q}$
OUTPUT: there exists $\varepsilon > 0$ such that for all words $w \in A^*$, we have $\mathbb{P}_{\mathcal{A}}(w) \notin [\lambda - \varepsilon; \lambda + \varepsilon]$.

Theorem (Bertoni, 1974)

The isolation problem is undecidable for $0 < \lambda < 1$.

What about $\lambda = 1$ and $\lambda = 0$?

A special case: the value 1 problem

For $\lambda = 1$ the isolation problem can be formulated as:

"are there words accepted with probability arbitrarily close to 1".

A special case: the value 1 problem

For $\lambda = 1$ the isolation problem can be formulated as:

"are there words accepted with probability arbitrarily close to 1".

Equivalently, define $\mathrm{val}(\mathcal{A}) = \sup_w \mathbb{P}_{\mathcal{A}}(w)$, then the problem is:

"$\mathrm{val}(\mathcal{A}) \stackrel{?}{=} 1$".

A special case: the value 1 problem

For $\lambda = 1$ the isolation problem can be formulated as:

"are there words accepted with probability arbitrarily close to 1".

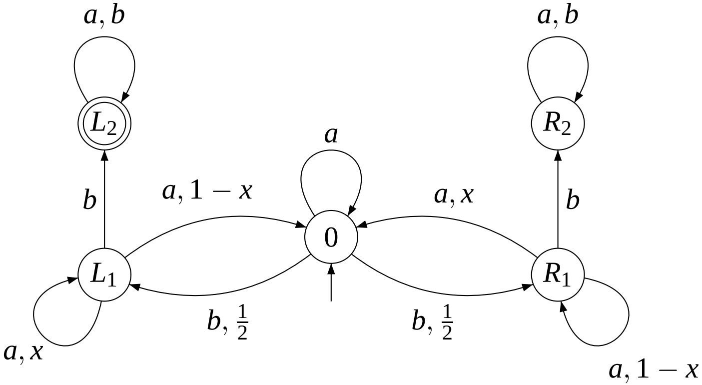
Equivalently, define $\mathrm{val}(\mathcal{A}) = \sup_w \mathbb{P}_{\mathcal{A}}(w)$, then the problem is:

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

Theorem (Gimbert and Oualhadj, 2010)

The value 1 *problem is undecidable.*

An intuition



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

A very restricted case

Theorem (F., Gimbert and Oualhadj, 2011)

The isolation problem is (still) undecidable if we randomise only on ***one*** *transition.*

Objective

Define a *partial* algorithm to decide the value 1 problem, by **algebraic** and **non-numerical** means.

Objective

Define a *partial* algorithm to decide the value 1 problem, by **algebraic** and **non-numerical** means.

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

Objective

Define a *partial* algorithm to decide the value 1 problem, by **algebraic** and **non-numerical** means.

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

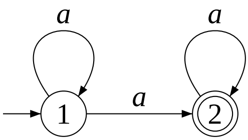
Hence we consider non-deterministic automata: we project $(\mathbb{R}, +, \cdot)$ into the boolean semiring $(\{0, 1\}, +, \cdot)$.

Objective

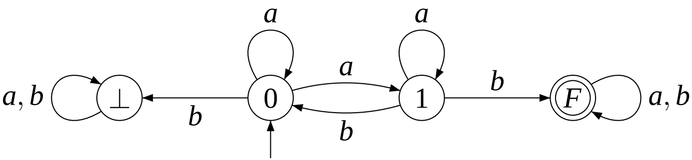
Define a *partial* algorithm to decide the value 1 problem, by **algebraic** and **non-numerical** means.

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

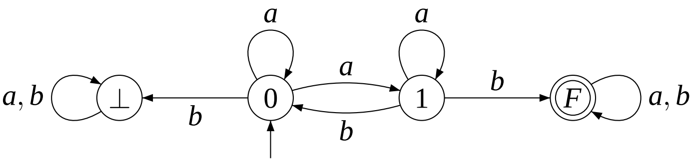
Hence we consider non-deterministic automata: we project $(\mathbb{R}, +, \cdot)$ into the boolean semiring $(\{0,1\}, +, \cdot)$.



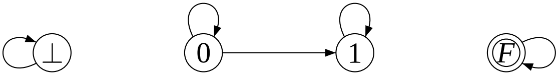
An example



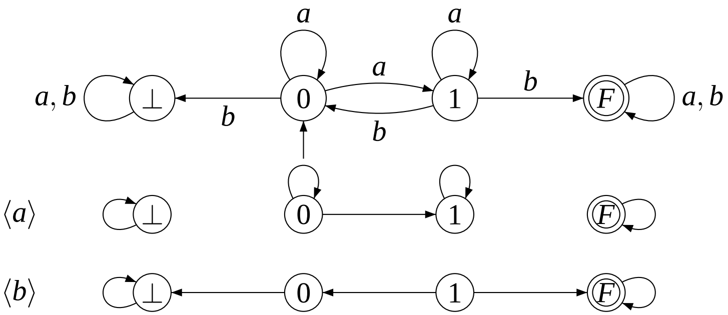
An example



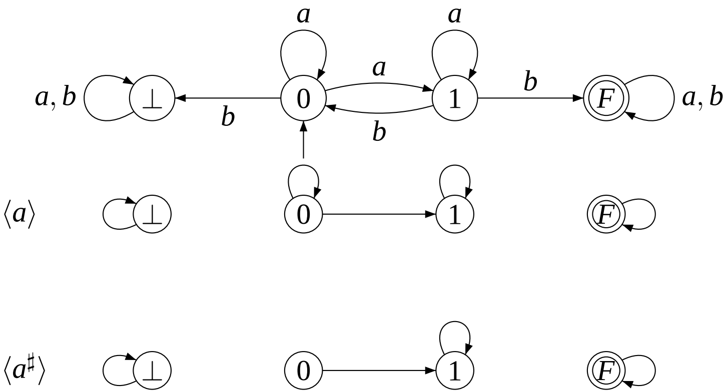
$\langle a \rangle$



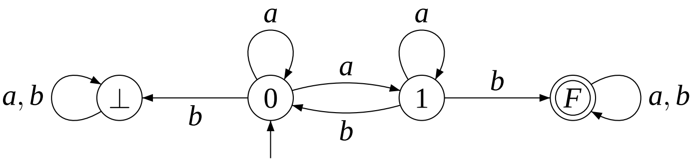
An example

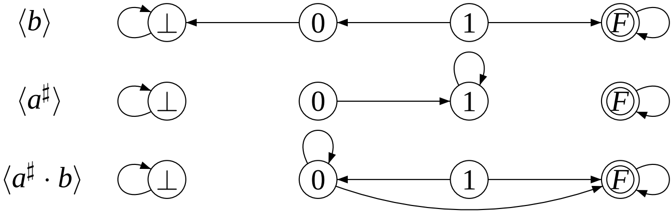


An example

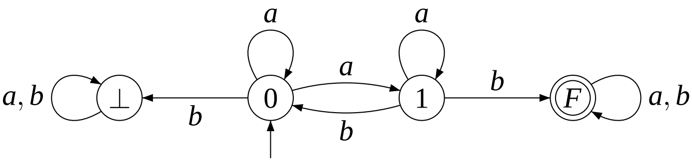


An example


a
a
a
b
a, b
⊥
b
0
1
F
a, b
b


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

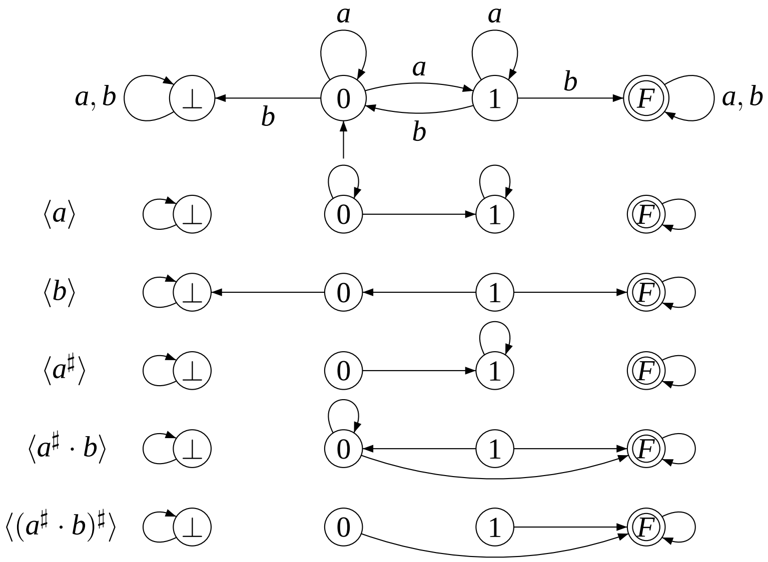
An example



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

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

An example

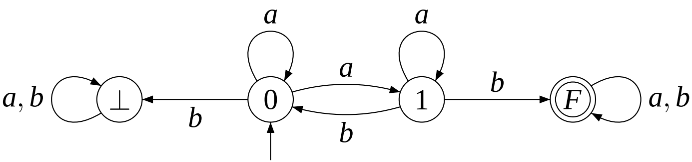


Stabilization monoids (Colcombet)

This is an algebraic structure with two operations:

- binary composition
- stabilization, denoted $\sharp$.

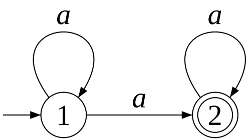
Boolean matrices representations



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

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

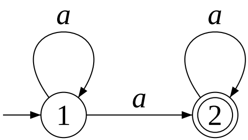
Defining stabilization



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

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

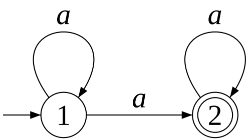
Defining stabilization



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

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

Defining stabilization



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

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

$$M^\sharp(s,t) = \begin{cases} 1 & \text{if } M(s,t) = 1 \text{ and } t \text{ recurrent in } M, \\ 0 & \text{otherwise.} \end{cases}$$

The algorithm

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

- Compute $\langle a\rangle$ for $a \in A$:

$$\langle a\rangle(s,t) = \begin{cases} 1 & \text{if } \mathbb{P}_{\mathcal{A}}(s \xrightarrow{a} t) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

- Close under product and stabilization.

The algorithm

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

- Compute $\langle a\rangle$ for $a \in A$:

$$\langle a\rangle(s,t) = \begin{cases} 1 & \text{if } \mathbb{P}_{\mathcal{A}}(s \xrightarrow{a} t) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

- 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”.

Correctness

Theorem

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.

Correctness

Theorem

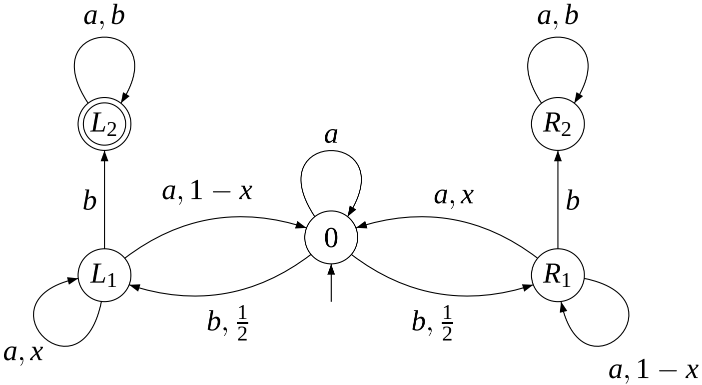
If there exists a matrix M such that

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

then $\mathcal{A}$ has value 1.

But the value 1 problem is undecidable, so…

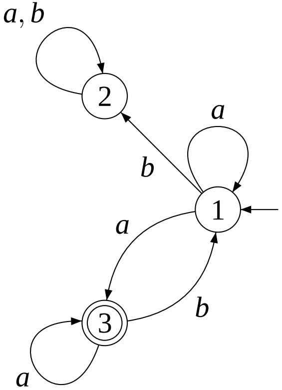
No completeness



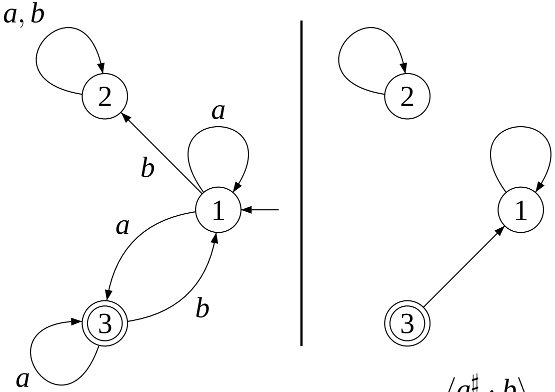
Left and right parts are symmetric, so for all M:

$$M(0, L_2) = 1 \iff M(0, R_2) = 1.$$

A leak

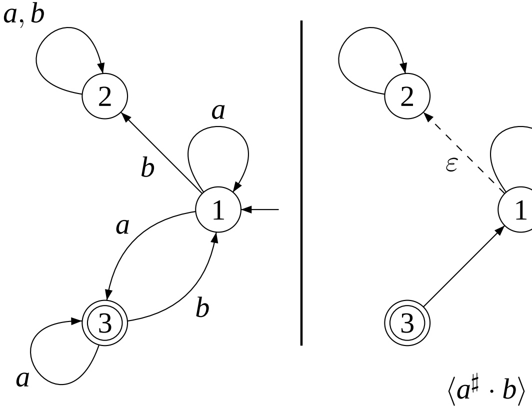


A leak



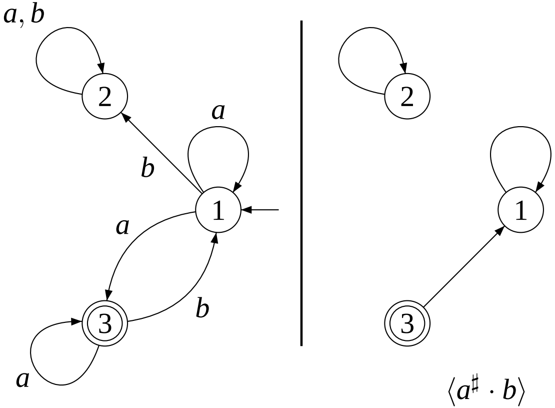
$\langle a^{\sharp} \cdot b \rangle$

A leak



There is a leak from 1 to 2.

A leak



There is a leak from 1 to 2.

Definition

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

Leaktight automata

Theorem (F.,Gimbert and Oualhadj, LICS 2012)

The algorithm is complete for leaktight automata.
Hence, the value 1 *problem is decidable for leaktight automata.*

The proof relies on Simon's factorization forest theorem.

Outline

Undecidability

INPUT: $\mathcal{A}$ a probabilistic automaton and a threshold $\lambda \in \mathbb{Q}$
OUTPUT: there exists a word $w \in A^*$ such that $\mathbb{P}_{\mathcal{A}}(w) \geq \lambda$.

Theorem (Paz, 1971)

The emptiness problem is undecidable for $0 < \lambda < 1$.

Approximations

INPUT: $\mathcal{A}$ a probabilistic automaton and a threshold $\lambda \in \mathbb{Q}$
OUTPUT:

- if there exists a word $w \in A^*$ such that $\mathbb{P}_{\mathcal{A}}(w) \geq \lambda$: YES,
- if for all words w we have $\mathbb{P}_{\mathcal{A}}(w) < \lambda$: NO.

Approximated:

INPUT: $\mathcal{A}$ a probabilistic automaton, two thresholds $\lambda, \varepsilon \in \mathbb{Q}$
OUTPUT:

- if there exists a word $w \in A^*$ such that $\mathbb{P}_{\mathcal{A}}(w) \geq \lambda$: YES,
- if for all words w we have $\mathbb{P}_{\mathcal{A}}(w) \leq \lambda - \varepsilon$: NO.

Inapproximability

INPUT: $\mathcal{A}$ a probabilistic automaton, two thresholds $\lambda, \varepsilon \in \mathbb{Q}$
OUTPUT:

- if there exists a word $w \in A^*$ such that $\mathbb{P}_{\mathcal{A}}(w) \geq \lambda$: YES,
- if for all words w we have $\mathbb{P}_{\mathcal{A}}(w) \leq \lambda - \varepsilon$: NO.

Inapproximability

INPUT: $\mathcal{A}$ a probabilistic automaton, two thresholds $\lambda, \varepsilon \in \mathbb{Q}$
OUTPUT:

- if there exists a word $w \in A^*$ such that $\mathbb{P}_{\mathcal{A}}(w) \geq \lambda$: YES,
- if for all words w we have $\mathbb{P}_{\mathcal{A}}(w) \leq \lambda - \varepsilon$: NO.

Theorem (Condon and Lipton, 1989)

There is no approximation algorithm for the emptiness problem.

Unary automata (*a.k.a* Markov chains)

Markov chains are the subcase where the alphabet has only one letter.

Theorem (Daviaud and F.)

There is an approximation algorithm for Markov chains.

Problem

Is the emptiness problem decidable for Markov chains?

Unary automata (*a.k.a* Markov chains)

Markov chains are the subcase where the alphabet has only one letter.

Theorem (Daviaud and F.)

There is an approximation algorithm for Markov chains.

Problem

Is the emptiness problem decidable for Markov chains?

Conclusions

A good understanding of algebra often allows to design simple algorithms for probabilistic automata.

The *equivalence problem* is decidable in polynomial time with a simple algebraic algorithm.

The *isolation problem* is undecidable, but there exists a partial algebraic algorithm.

The *emptiness problem* is undecidable and in general not even approximable.
Is it decidable for Markov chains?

The end.

Thanks for your attention!