The Value 1 Problem for Probabilistic Automata

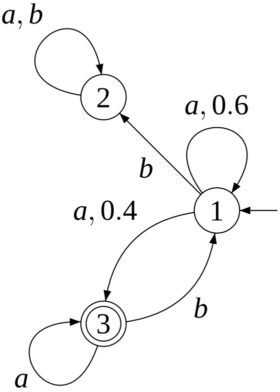
Bruxelles

Nathanaël Fijalkow

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

June 20th, 2014

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

The value 1 problem

This talk is about the value 1 problem:

INPUT: $\mathcal{A}$ a probabilistic automaton
OUTPUT: for all $\varepsilon > 0$, there exists $w \in A^*$, $\mathbb{P}_{\mathcal{A}}(w) \geq 1 - \varepsilon$.

In other words, define $\mathrm{val}(\mathcal{A}) = \sup_{w \in A^*} \mathbb{P}_{\mathcal{A}}(w)$, is $\mathrm{val}(\mathcal{A}) = 1$?

The value 1 problem

This talk is about the value 1 problem:

INPUT: $\mathcal{A}$ a probabilistic automaton
OUTPUT: for all $\varepsilon > 0$, there exists $w \in A^*$, $\mathbb{P}_{\mathcal{A}}(w) \geq 1 - \varepsilon$.

In other words, define $\text{val}(\mathcal{A}) = \sup_{w \in A^*} \mathbb{P}_{\mathcal{A}}(w)$, is $\text{val}(\mathcal{A}) = 1$?

It is undecidable (Gimbert and Oualhadj, 2010).

But *to what extent*?

Objective

Construct an algorithm to decide the value 1 problem, which is *often* correct.

Objective

Construct an algorithm to decide the value 1 problem,
which is *often* correct.

Quantify *how often*.

Objective

Construct an algorithm to decide the value 1 problem,
which is *often* correct.

Quantify *how often*.

Argue that you cannot do *more often* than that.

Outline

Outline

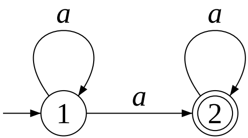
Outline

At first thought

Does the *undecidability* come from the *numerical* values?

Does the *undecidability* come from the *numerical* values?

Consider *numberless* probabilistic automata:



Two decision problems:

- for all Δ, $\mathrm{val}(\mathcal{A}[\Delta]) = 1$,
- there exists Δ, such that $\mathrm{val}(\mathcal{A}[\Delta]) = 1$.

No future (in this direction)

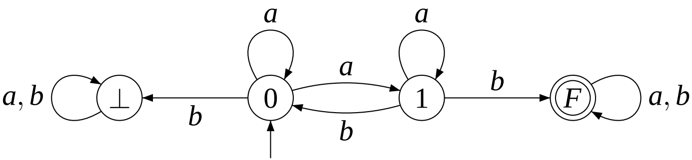
Theorem (F., Horn, Gimbert and Oualhadj)

There is no algorithm such that:
On input $\mathcal{A}$ (a non-deterministic automaton),

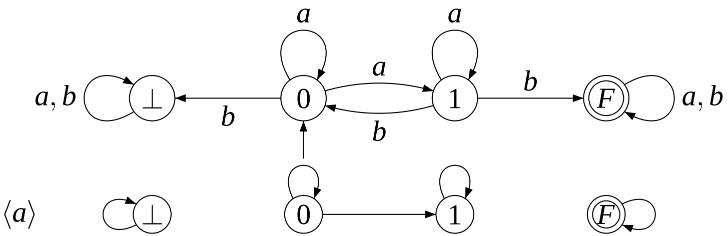
- *if for all* Δ, $val(\mathcal{A}[\Delta]) = 1$ *then "YES",*
- *if for all* Δ, $val(\mathcal{A}[\Delta]) < 1$ *then "NO",*
- *anything in the other cases.*

Outline

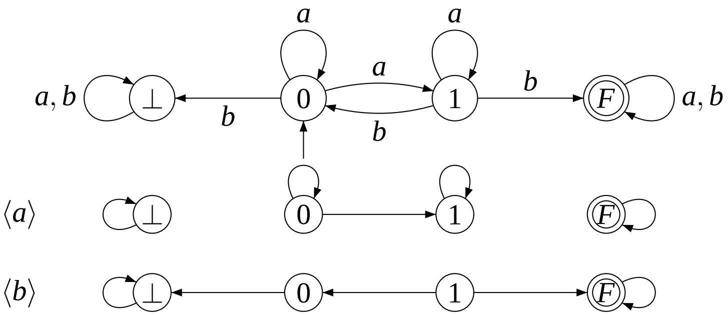
An example



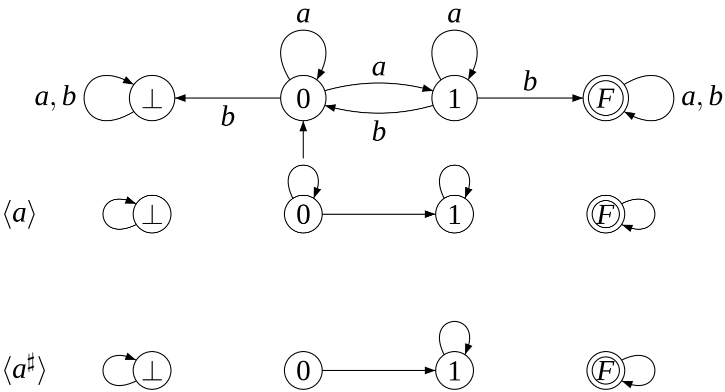
An example



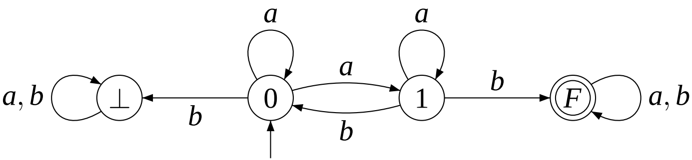
An example

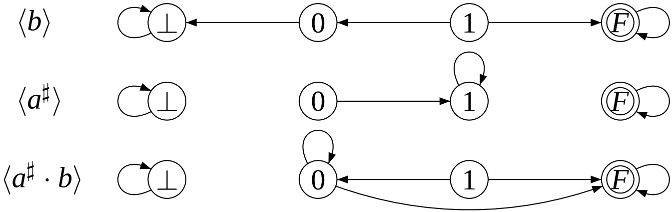


An example

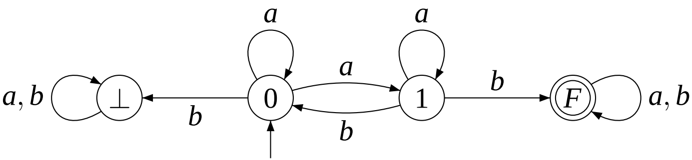


An example

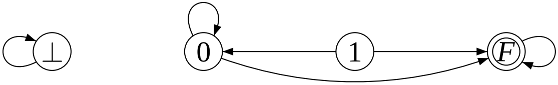

a
a
a
b
a, b
⊥
0
1
F
a, b
b
b


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

An example

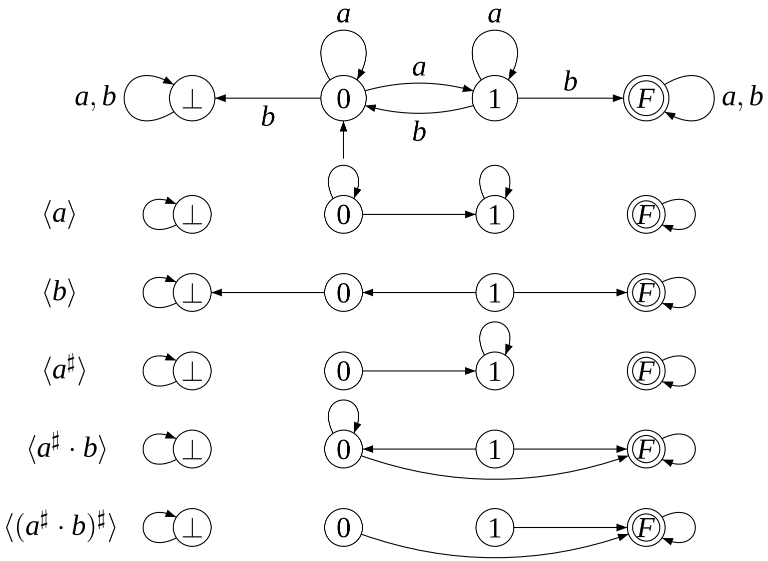


$\langle a^\sharp \cdot b \rangle$



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

An example



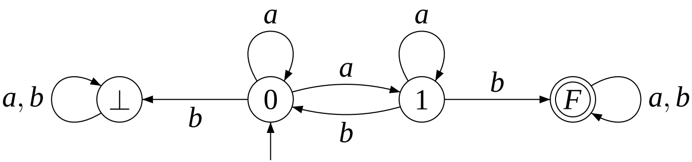
a
a
a
b
a, b
$\perp$
0
1
F
a, b
b
b
$\langle a \rangle$
$\langle b \rangle$
$\langle a^\sharp \rangle$
$\langle a^\sharp \cdot b \rangle$
$\langle (a^\sharp \cdot b)^\sharp \rangle$

Stabilization monoids (Colcombet)

This is an algebraic structure with two operations:

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

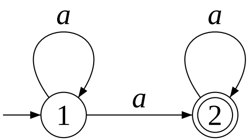
Boolean matrices representations



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

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

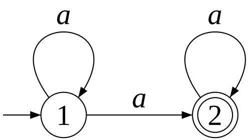
Defining stabilization



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

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

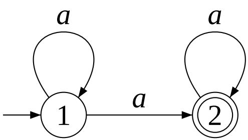
Defining stabilization



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

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 Markov Monoid Algorithm

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

- 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 Markov Monoid Algorithm

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

- 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

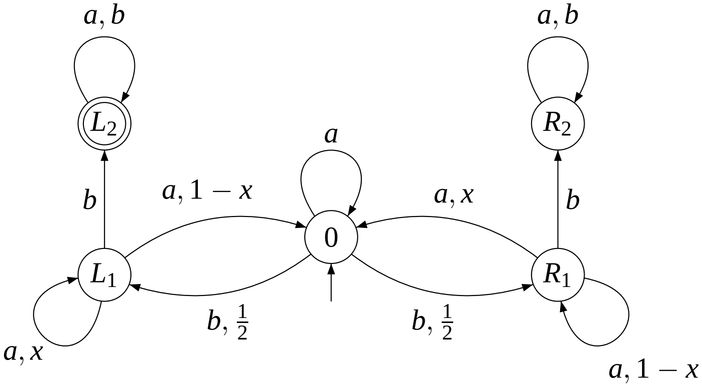
If there exists a matrix M such that

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

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

But the value 1 problem is undecidable, so…

No completeness

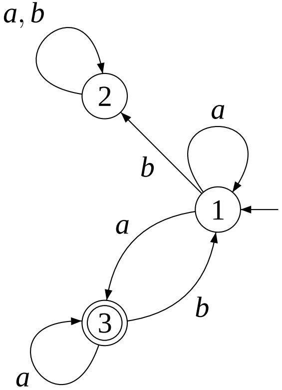


Left and right parts are symmetric, so for all M:

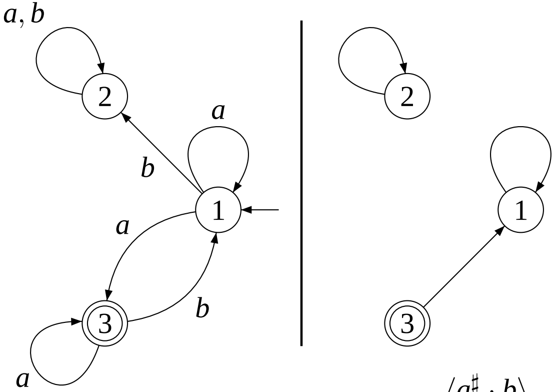
$$M(0, L_2) = 1 \iff M(0, R_2) = 1.$$

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

A leak

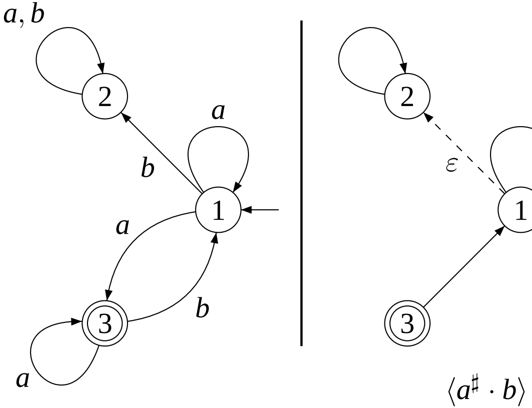


A leak



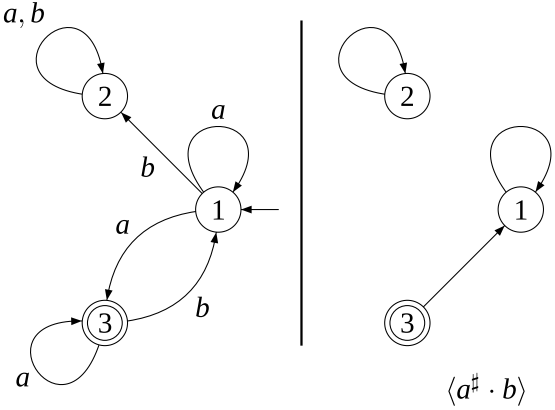
$\langle a^\sharp \cdot b \rangle$

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.

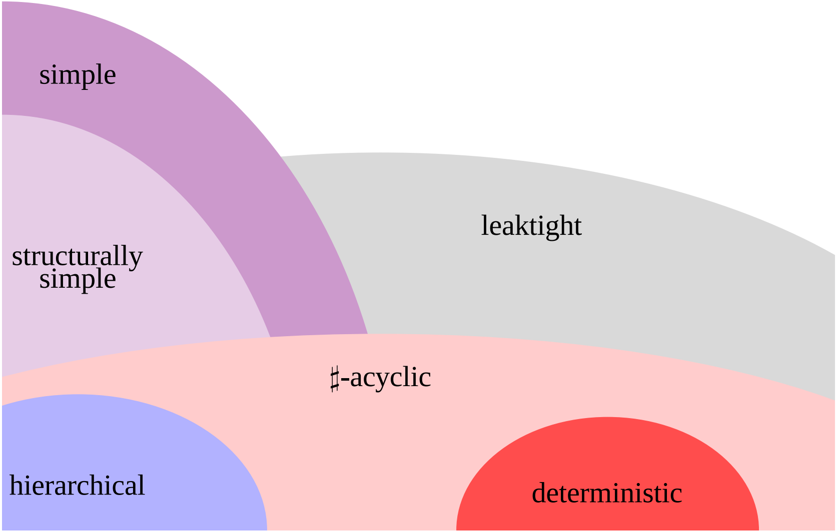
Leaktight automata

Theorem (F.,Gimbert and Oualhadj 2012)

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

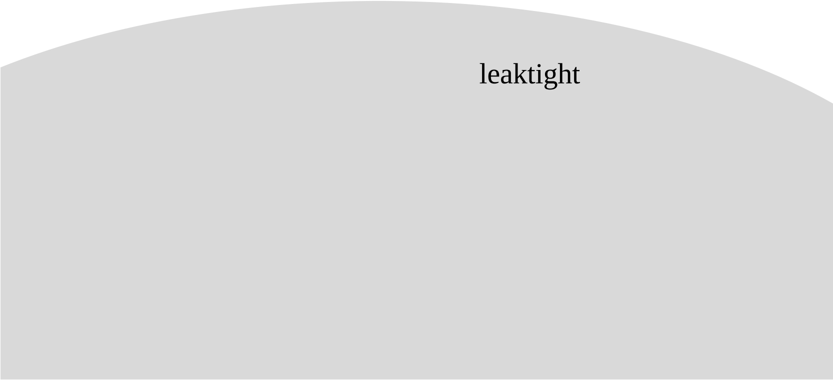
The proof relies on Simon’s factorization forest theorem.

Other decidable subclasses: in 2012



Other decidable subclasses: today

(F.,Gimbert,Kelmendi and Oualhadj 2013)



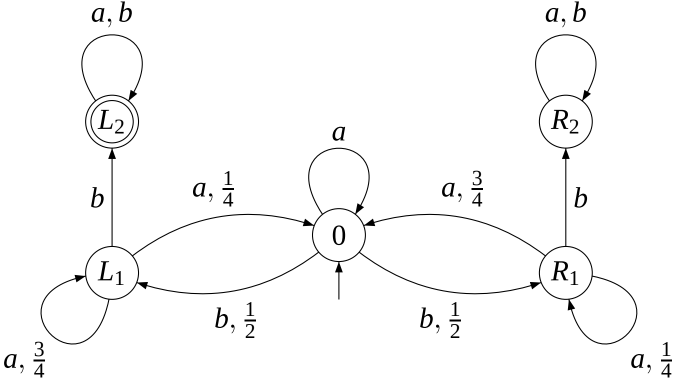
So far,
the Markov Monoid Algorithm is
the *most correct* algorithm known
to solve the value 1 problem.

So far,
the Markov Monoid Algorithm is
the *most correct* algorithm known
to solve the value 1 problem.

But for *how long*?

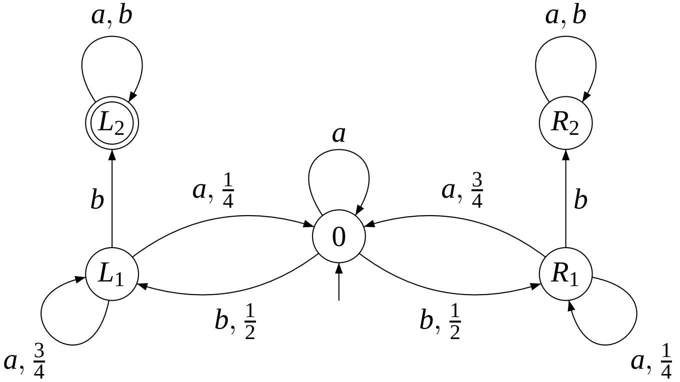
Outline

What it misses: different convergence speeds



$$\lim_n \mathbb{P}_{\mathcal{A}}\left(b(a^n b)^{f(n)}\right) = 1 \text{ if and only if } \lim_n f(n) \cdot \left(\frac{3}{4}\right)^n = \infty \ ,$$

What it misses: different convergence speeds



$$\lim_n \mathbb{P}_{\mathcal{A}} \left(b(a^n b)^{f(n)} \right) = 1 \text{ if and only if } \lim_n f(n) \cdot \left(\frac{3}{4}\right)^n = \infty \,,$$

so $f(n) = 2^n$ works but $f(n) = n$ does not.

A characterization

$\widetilde{A^*}$ is the space of prostochastic words.

$$A^* = \widetilde{A^*}[0] \subsetneq \widetilde{A^*}[1] \subsetneq \widetilde{A^*}[2] \subsetneq \cdots \subsetneq \widetilde{A^*} .$$

Lemma

The following are equivalent:

- *The value* 1 *problem over finite words,*
- *The emptiness problem over prostochastic words.*

A characterization

$\widetilde{A^*}$ is the space of prostochastic words.

$$A^* = \widetilde{A^*}[0] \subsetneq \widetilde{A^*}[1] \subsetneq \widetilde{A^*}[2] \subsetneq \cdots \subsetneq \widetilde{A^*} .$$

Lemma

The following are equivalent:

- *The value* 1 *problem over finite words,*
- *The emptiness problem over prostochastic words.*

Theorem

1. *The Markov Monoid Algorithm answers "YES" if and only if there exists* $x \in \widetilde{A^*}[1]$ *accepted by* $\mathcal{A}$*,*
2. *The following problem is undecidable: determine whether there exists* $x \in \widetilde{A^*}[2]$ *accepted by* $\mathcal{A}$*.*

Prostochastic words

Definition

$(u_n)_{n\in\mathbb{N}}$ converges if for every $\mathcal{A}$, the limit $\lim_n \mathbb{P}_{\mathcal{A}}(u_n)$ exists.

Prostochastic words

Definition

$(u_n)_{n\in\mathbb{N}}$ converges if for every $\mathcal{A}$, the limit $\lim_n \mathbb{P}_{\mathcal{A}}(u_n)$ exists.

Definition

Two (converging) sequences $(u_n)_{n\in\mathbb{N}}$ and $(v_n)_{n\in\mathbb{N}}$ are equivalent if for every $\mathcal{A}$,

$$\lim_n \mathbb{P}_{\mathcal{A}}(u_n) > 0 \iff \lim_n \mathbb{P}_{\mathcal{A}}(v_n) > 0 \ .$$

Prostochastic words

Definition

$(u_n)_{n\in\mathbb{N}}$ converges if for every $\mathcal{A}$, the limit $\lim_n \mathbb{P}_{\mathcal{A}}(u_n)$ exists.

Definition

Two (converging) sequences $(u_n)_{n\in\mathbb{N}}$ and $(v_n)_{n\in\mathbb{N}}$ are equivalent if for every $\mathcal{A}$,

$$\lim_n \mathbb{P}_{\mathcal{A}}(u_n) > 0 \iff \lim_n \mathbb{P}_{\mathcal{A}}(v_n) > 0 \; .$$

Definition

A prostochastic word is an equivalence class of converging sequences.

The ω operators

Definition

Let u be a converging sequence.
u^{ω_1} is the converging sequence $(u_n^{n!})_{n\in\mathbb{N}}$.

The ω operators

Definition

Let u be a converging sequence.
u^{ω_1} is the converging sequence $(u_n^{n!})_{n\in\mathbb{N}}$.

Definition

Let u be a converging sequence.
u^{ω_k} is the converging sequence $(u_n^{(n!)^k})_{n\in\mathbb{N}}$.

The ω operators

Definition

Let u be a converging sequence.
u^{ω_1} is the converging sequence $(u_n^{n!})_{n \in \mathbb{N}}$.

Definition

Let u be a converging sequence.
u^{ω_k} is the converging sequence $(u_n^{(n!)^k})_{n \in \mathbb{N}}$.

Example

The prostochastic words $(a^{\omega_1} b)^{\omega_1}$ and $(a^{\omega_1} b)^{\omega_2}$ are not equal.

An equivalent characterization

Theorem

The Markov Monoid Algorithm answers "YES" if and only if there exists a regular sequence $(u_n)_{n \in \mathbb{N}}$ *of finite words such that* $\lim_n \mathbb{P}_{\mathcal{A}}(u_n) = 1$.

The regular sequences are described by the following grammar:

$$u \quad = \quad a \quad | \quad u \cdot u \quad | \quad (u_n^n)_{n \in \mathbb{N}} \ .$$

Corollary

In *some* sense,

the Markov Monoid Algorithm is

the *most correct* algorithm
to solve the value 1 problem.

Outline

ACMÉ

The tool ACME (Automata with Counters, Monoids and Equivalence) has been written in OCaml by Nathanaël Fijalkow and Denis Kuperberg.

****

Statistics on the Markov Monoid Algorithm:

****

Automata that are leaktight and do not have value 1: 540
Automata that are leaktight and have value 1: 133
Automata that are not leaktight and may have value 1: 17
Automata that are not leaktight and have value 1: 310

The end.

Thank you for your attention!