# Automata and program analysis

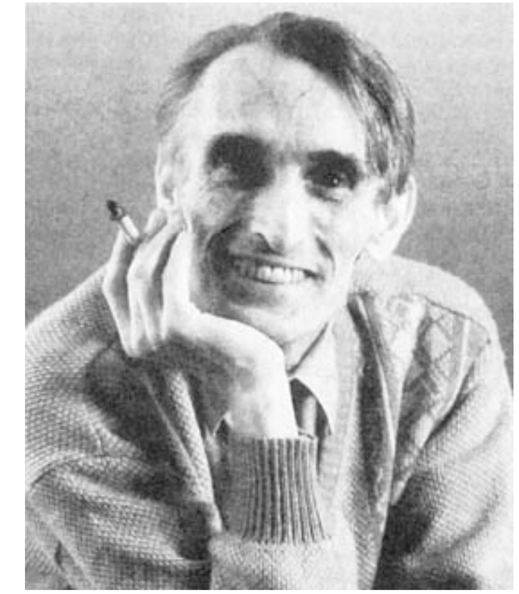
Thomas Colcombet
FCT
Bordeaux 13 September 2017

based on joint work with Laure Daviaud et Florian Zuleger

# Weighted automata
# and
# tropical automata

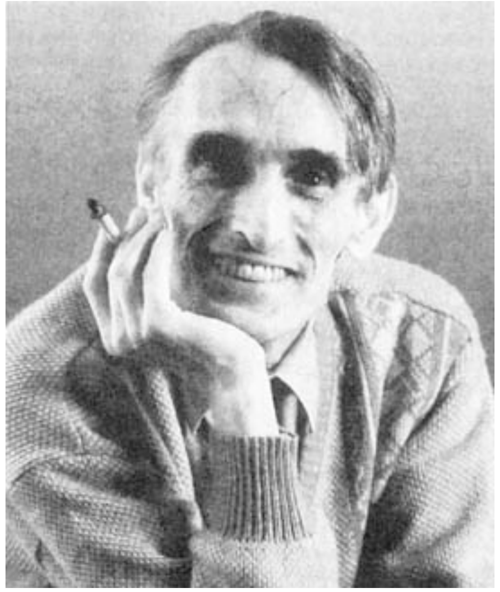
# Weighted automata

[Schützenberger 61]



# Weighted automata

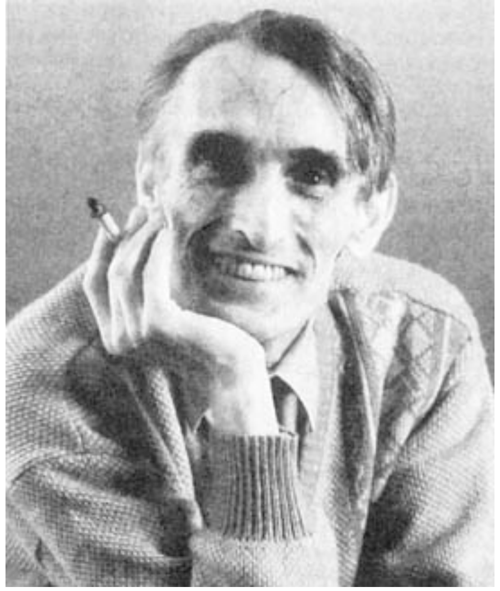
[Schützenberger 61]



Consider a non-deterministic automaton (A,Q,I,F,Δ).

# Weighted automata

[Schützenberger 61]

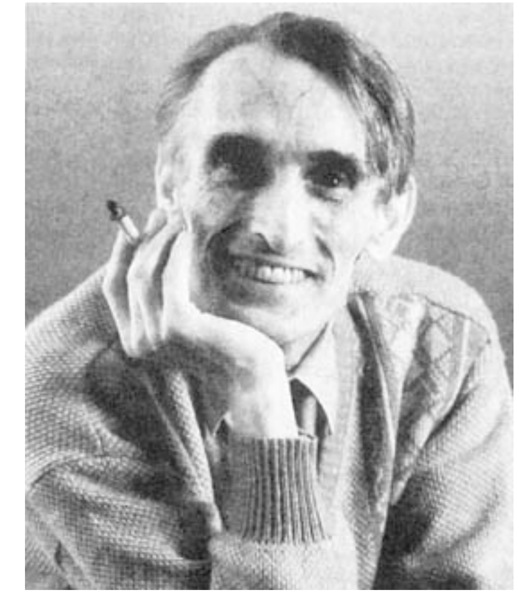


Consider a non-deterministic automaton (A,Q,I,F,Δ).

It computes a language $L: A^* \to \{0,1\}$

# Weighted automata

[Schützenberger 61]



Consider a non-deterministic automaton (A,Q,I,F,Δ).

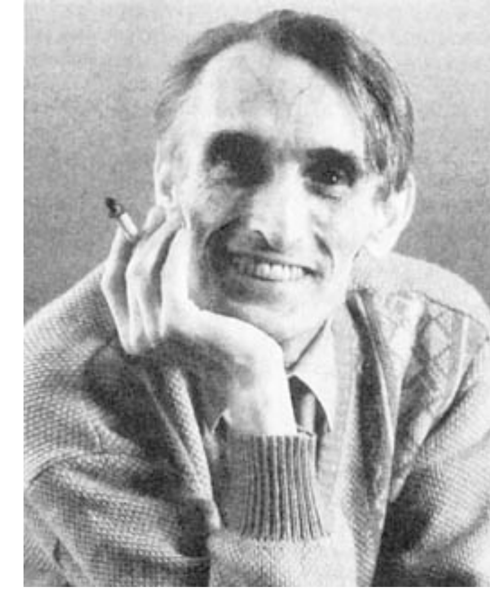
It computes a language L: A* → {0,1}

accepted

not accepted

# Weighted automata

[Schützenberger 61]



Consider a non-deterministic automaton $(A,Q,I,F,\Delta)$.

It computes a language $L: A^* \to \{0,1\}$

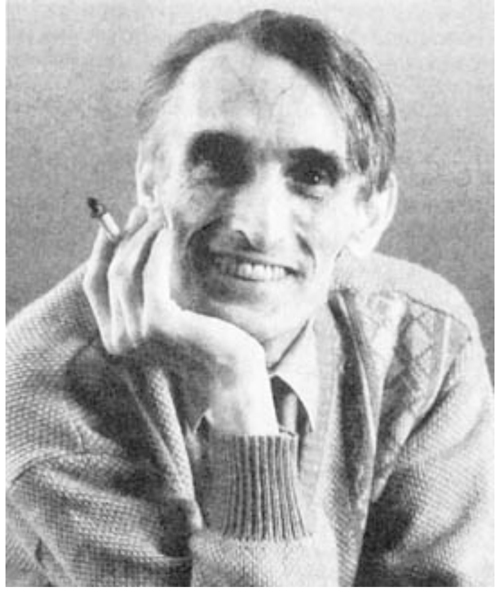
accepted

not accepted

$Q$ states, initial $I: Q \to \{0,1\}$, final $F: Q \to \{0,1\}$, weights $\Delta: Q\times A\times Q \to \{0,1\}$

# Weighted automata

[Schützenberger 61]



Consider a non-deterministic automaton (A,Q,I,F,Δ).

It computes a language $L: A^* \to \{0,1\}$

accepted

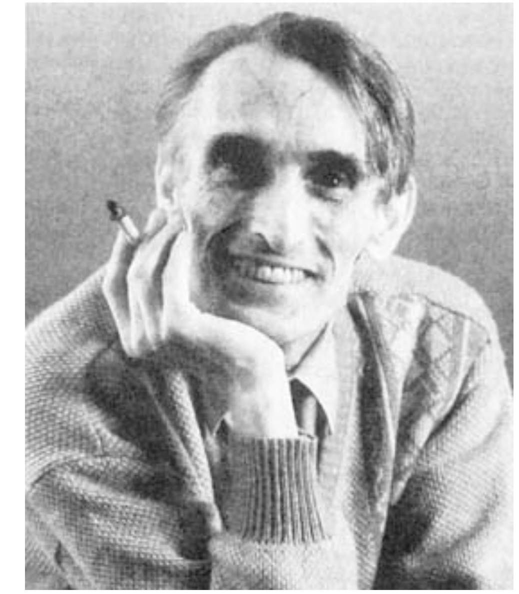
not accepted

Q states, initial I: $Q \to \{0,1\}$, final F: $Q \to \{0,1\}$, weights Δ: $Q \times A \times Q \to \{0,1\}$

Definition: $u = a_1, a_2, \ldots, a_n \in L$ iff there exists an accepting run over it.

# Weighted automata

[Schützenberger 61]



Consider a non-deterministic automaton (A,Q,I,F,Δ).

It computes a language $L: A^* \to \{0,1\}$

accepted

not accepted

Q states, initial I: $Q \to \{0,1\}$, final F: $Q \to \{0,1\}$, weights Δ: $Q \times A \times Q \to \{0,1\}$

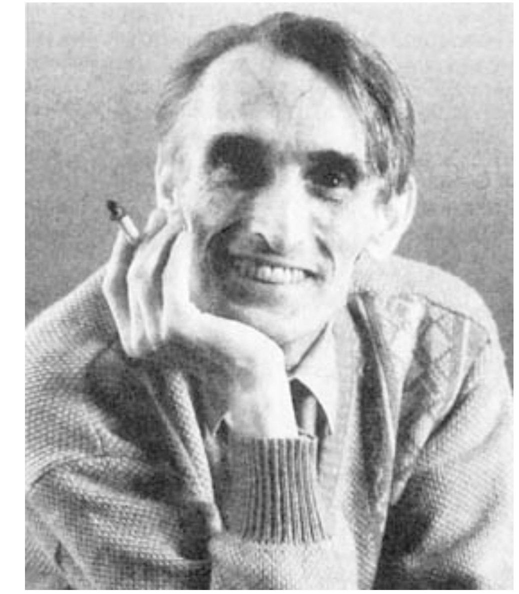
Definition: $u = a_1,a_2,\ldots,a_n \in L$ iff there exists an accepting run over it.

Logically, there exist $p_0,p_1,\ldots,p_n$ such that

$$I(p_0) \wedge \Delta(p_0,a_1,p_1) \wedge \Delta(p_1,a_2,p_2) \wedge \ldots \wedge \Delta(p_{n-1},a_n,p_n) \wedge F(p_n)$$

# Weighted automata

[Schützenberger 61]



Consider a non-deterministic automaton $(A,Q,I,F,\Delta)$.

It computes a language $L: A^* \to \{0,1\}$

accepted

not accepted

$Q$ states, initial $I: Q \to \{0,1\}$, final $F: Q \to \{0,1\}$, weights $\Delta: Q\times A\times Q \to \{0,1\}$

Definition: $u = a_1,a_2,\ldots,a_n \in L$ iff there exists an accepting run over it.

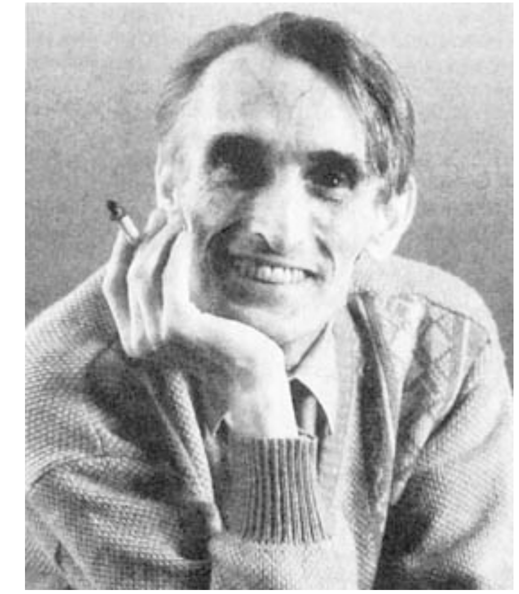
Logically, there exist $p_0,p_1,\ldots,p_n$ such that

$$I(p_0) \wedge \Delta(p_0,a_1,p_1) \wedge \Delta(p_1,a_2,p_2) \wedge \ldots \wedge \Delta(p_{n-1},a_n,p_n) \wedge F(p_n)$$

[Schützenberger 61] disjunction and conjunction can be replaced by the operation over an arbitrary semiring $(S,\oplus,\otimes,0,1)$.

# Weighted automata

[Schützenberger 61]



Consider a non-deterministic automaton (A,Q,I,F,Δ).

It computes a language $L: A^* \to \{0,1\}$

accepted

not accepted

Q states, initial I: $Q \to \{0,1\}$, final F: $Q \to \{0,1\}$, weights Δ: $Q \times A \times Q \to \{0,1\}$

Definition: $u = a_1,a_2,\ldots,a_n \in L$ iff there exists an accepting run over it.

Logically, there exist $p_0,p_1,\ldots,p_n$ such that

$$I(p_0) \wedge \Delta(p_0,a_1,p_1) \wedge \Delta(p_1,a_2,p_2) \wedge \ldots \wedge \Delta(p_{n-1},a_n,p_n) \wedge F(p_n)$$

[Schützenberger 61] disjunction and conjunction can be replaced by the operation over an arbitrary semiring $(S,\oplus,\otimes,0,1)$.

An automaton (A,Q,I,F,Δ) with I: $Q\to S$, F: $Q\to S$, and Δ: $Q\times A\times Q$, computes a map $L: A^* \to S$ defined as

$$L(a_1a_2\ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

# Example of weighted automata

# Example of weighted automata

A semiring $(S,\oplus,\otimes,0,1)$ is such that:

- $(R, \oplus)$ is a commutative monoid with identity element 0:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element 1:
  $(a\otimes b)\cdot c = a\otimes(b\otimes c)$ ; $1\otimes a = a\otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a\otimes(b \oplus c) = (a\otimes b) \oplus (a\otimes c)$ ; $(a \oplus b)\otimes c = (a\otimes c) \oplus (b\otimes c)$
- Multiplication by 0 annihilates S:
  $0\otimes a = a\otimes 0 = 0$

# Example of weighted automata

multiplication
addition

A semiring $(S,\oplus,\otimes,0,1)$ is such that:

- $(R, \oplus)$ is a commutative monoid with identity element 0:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element 1:
  $(a\otimes b)\cdot c = a\otimes(b\otimes c)$ ; $1\otimes a = a\otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a\otimes(b \oplus c) = (a\otimes b) \oplus (a\otimes c)$ ; $(a \oplus b)\otimes c = (a\otimes c) \oplus (b\otimes c)$
- Multiplication by 0 annihilates S:
  $0\otimes a = a\otimes 0 = 0$

# Example of weighted automata

multiplication
addition

A semiring $(S,\oplus,\otimes,0,1)$ is such that:

- $(R, \oplus)$ is a commutative monoid with identity element $0$:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element $1$:
  $(a \otimes b) \cdot c = a \otimes (b \otimes c)$ ; $1 \otimes a = a \otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ ; $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$
- Multiplication by 0 annihilates S:
  $0 \otimes a = a \otimes 0 = 0$

Gives rise to product of S valued matrices that form a monoid.

# Example of weighted automata

multiplication
addition

A semiring $(S,\oplus,\otimes,0,1)$ is such that:

- $(R, \oplus)$ is a commutative monoid with identity element 0:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element 1:
  $(a\otimes b)\cdot c = a\otimes(b\otimes c)$ ; $1\otimes a = a\otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a\otimes(b \oplus c) = (a\otimes b) \oplus (a\otimes c)$ ; $(a \oplus b)\otimes c = (a\otimes c) \oplus (b\otimes c)$
- Multiplication by 0 annihilates S:
  $0\otimes a = a\otimes 0 = 0$

Gives rise to product of S valued matrices that form a monoid.

$$L(a_1a_2\dots a_n) = \bigoplus_{p_0,\dots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

# Example of weighted automata

A semiring $(S,\oplus,\otimes,0,1)$ is such that:

multiplication
addition

- $(R, \oplus)$ is a commutative monoid with identity element 0:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element 1:
  $(a\otimes b)\cdot c = a\otimes(b\otimes c)$ ; $1\otimes a = a\otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a\otimes(b \oplus c) = (a\otimes b) \oplus (a\otimes c)$ ; $(a \oplus b)\otimes c = (a\otimes c) \oplus (b\otimes c)$
- Multiplication by 0 annihilates S:
  $0\otimes a = a\otimes 0 = 0$

Gives rise to product of S valued matrices that form a monoid.

$$L(a_1 a_2 \ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

Boolean semiring: $(\{0,1\},\vee,\wedge,0,1)$

# Example of weighted automata

multiplication
addition

A semiring $(S,\oplus,\otimes,0,1)$ is such that:

- $(R, \oplus)$ is a commutative monoid with identity element 0:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element 1:
  $(a\otimes b)\cdot c = a\otimes(b\otimes c)$ ; $1\otimes a = a\otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a\otimes(b \oplus c) = (a\otimes b) \oplus (a\otimes c)$ ; $(a \oplus b)\otimes c = (a\otimes c) \oplus (b\otimes c)$
- Multiplication by 0 annihilates S:
  $0\otimes a = a\otimes 0 = 0$

Gives rise to product of S valued matrices that form a monoid.

$$L(a_1a_2\ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

Boolean semiring: $(\{0,1\},\vee,\wedge,0,1)$

Non-deterministic automata

# Example of weighted automata

multiplication
addition

A semiring $(S,\oplus,\otimes,0,1)$ is such that:

- $(R, \oplus)$ is a commutative monoid with identity element 0:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element 1:
  $(a\otimes b)\cdot c = a\otimes(b\otimes c)$ ; $1\otimes a = a\otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a\otimes(b \oplus c) = (a\otimes b) \oplus (a\otimes c)$ ; $(a \oplus b)\otimes c = (a\otimes c) \oplus (b\otimes c)$
- Multiplication by 0 annihilates S:
  $0\otimes a = a\otimes 0 = 0$

Gives rise to product of S valued matrices that form a monoid.

$$L(a_1a_2\ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

Boolean semiring: $(\{0,1\},\vee,\wedge,0,1)$ Non-deterministic automata

Reals/Integers/Rationals/Natural numbers: $(\mathbf{R},+,\times,0,1)$

# Example of weighted automata

multiplication
addition

A semiring $(S,\oplus,\otimes,0,1)$ is such that:

- $(R, \oplus)$ is a commutative monoid with identity element 0:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element 1:
  $(a\otimes b)\cdot c = a\otimes(b\otimes c)$ ; $1\otimes a = a\otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a\otimes(b \oplus c) = (a\otimes b) \oplus (a\otimes c)$ ; $(a \oplus b)\otimes c = (a\otimes c) \oplus (b\otimes c)$
- Multiplication by 0 annihilates S:
  $0\otimes a = a\otimes 0 = 0$

Gives rise to product of S valued matrices that form a monoid.

$$L(a_1 a_2 \ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

Boolean semiring: $(\{0,1\},\vee,\wedge,0,1)$ — Non-deterministic automata

Reals/Integers/Rationals/Natural numbers: $(\mathbf{R},+,\times,0,1)$ — Computes the number of runs of the NDA

# Example of weighted automata

multiplication
addition

A semiring $(S,\oplus,\otimes,0,1)$ is such that:

- $(R, \oplus)$ is a commutative monoid with identity element 0:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element 1:
  $(a\otimes b)\cdot c = a\otimes(b\otimes c)$ ; $1\otimes a = a\otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a\otimes(b \oplus c) = (a\otimes b) \oplus (a\otimes c)$ ; $(a \oplus b)\otimes c = (a\otimes c) \oplus (b\otimes c)$
- Multiplication by 0 annihilates S:
  $0\otimes a = a\otimes 0 = 0$

Gives rise to product of S valued matrices that form a monoid.

$$L(a_1a_2\ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

Boolean semiring: $(\{0,1\},\vee,\wedge,0,1)$ Non-deterministic automata

Reals/Integers/Rationals/Natural numbers: $(\mathbf{R},+,\times,0,1)$ Computes the number of runs of the NDA

« Rat semiring »: $(\mathrm{Rat}(A), \cup , \cdot , \varnothing, \{\varepsilon\} )$

# Example of weighted automata

A semiring $(S,\oplus,\otimes,0,1)$ is such that:

multiplication

addition

- $(R, \oplus)$ is a commutative monoid with identity element 0:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element 1:
  $(a\otimes b)\cdot c = a\otimes(b\otimes c)$ ; $1\otimes a = a\otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a\otimes(b \oplus c) = (a\otimes b) \oplus (a\otimes c)$ ; $(a \oplus b)\otimes c = (a\otimes c) \oplus (b\otimes c)$
- Multiplication by 0 annihilates S:
  $0\otimes a = a\otimes 0 = 0$

Gives rise to product of S valued matrices that form a monoid.

$$L(a_1a_2\ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

Boolean semiring: $(\{0,1\},\vee,\wedge,0,1)$ — Non-deterministic automata

Reals/Integers/Rationals/Natural numbers: $(\mathbf{R},+,\times,0,1)$ — Computes the number of runs of the NDA

« Rat semiring »: $(\mathrm{Rat}(A), \cup , \cdot , \varnothing, \{\varepsilon\} )$ — Rational transducers

# Example of weighted automata

A semiring $(S,\oplus,\otimes,0,1)$ is such that:

multiplication
addition

- $(R, \oplus)$ is a commutative monoid with identity element 0:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element 1:
  $(a \otimes b) \cdot c = a \otimes (b \otimes c)$ ; $1 \otimes a = a \otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ ; $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$
- Multiplication by 0 annihilates S:
  $0 \otimes a = a \otimes 0 = 0$

Gives rise to product of S valued matrices that form a monoid.

$$L(a_1 a_2 \ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

Boolean semiring: $(\{0,1\},\vee,\wedge,0,1)$ — Non-deterministic automata

Reals/Integers/Rationals/Natural numbers: $(\mathbf{R},+,\times,0,1)$ — Computes the number of runs of the NDA

« Rat semiring »: $(\mathrm{Rat}(A), \cup , \cdot , \varnothing, \{\varepsilon\})$ — Rational transducers

Tropical semiring: $(\mathbf{R}\cup\{-\infty\},\max,+,-\infty,0)$
$(\mathbf{R}\cup\{+\infty\},\min,+,+\infty,0)$, $(\mathbf{N}\cup\{-\infty\},\max,+,-\infty,0)$, $(\mathbf{N}\cup\{+\infty\},\min,+,+\infty,0)$

# Example of weighted automata

A semiring $(S,\oplus,\otimes,0,1)$ is such that: multiplication addition

- $(R, \oplus)$ is a commutative monoid with identity element 0:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element 1:
  $(a\otimes b)\cdot c = a\otimes(b\otimes c)$ ; $1\otimes a = a\otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a\otimes(b \oplus c) = (a\otimes b) \oplus (a\otimes c)$ ; $(a \oplus b)\otimes c = (a\otimes c) \oplus (b\otimes c)$
- Multiplication by 0 annihilates S:
  $0\otimes a = a\otimes 0 = 0$

Gives rise to product of S valued matrices that form a monoid.

$$L(a_1a_2\ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

Boolean semiring: $(\{0,1\},\vee,\wedge,0,1)$ — Non-deterministic automata

Reals/Integers/Rationals/Natural numbers: $(\mathbf{R},+,\times,0,1)$ — Computes the number of runs of the NDA

« Rat semiring »: $(\mathrm{Rat}(A), \cup , \cdot , \varnothing, \{\varepsilon\})$ — Rational transducers

Tropical semiring: $(\mathbf{R}\cup\{-\infty\},\max,+,-\infty,0)$ — Tropical automata

$(\mathbf{R}\cup\{+\infty\},\min,+,+\infty,0)$, $(\mathbf{N}\cup\{-\infty\},\max,+,-\infty,0)$, $(\mathbf{N}\cup\{+\infty\},\min,+,+\infty,0)$

# Example of weighted automata

multiplication
addition

A semiring $(S,\oplus,\otimes,0,1)$ is such that:

- $(R, \oplus)$ is a commutative monoid with identity element $0$:
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ ; $0 \oplus a = a \oplus 0 = a$ ; $a \oplus b = b \oplus a$
- $(R, \otimes)$ is a monoid with identity element $1$:
  $(a\otimes b)\cdot c = a\otimes(b\otimes c)$ ; $1\otimes a = a\otimes 1 = a$
- Multiplication left and right distributes over addition:
  $a\otimes(b \oplus c) = (a\otimes b) \oplus (a\otimes c)$ ; $(a \oplus b)\otimes c = (a\otimes c) \oplus (b\otimes c)$
- Multiplication by $0$ annihilates $S$:
  $0\otimes a = a\otimes 0 = 0$

Gives rise to product of S valued matrices that form a monoid.

$$L(a_1 a_2 \ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

Boolean semiring: $(\{0,1\},\vee,\wedge,0,1)$ — Non-deterministic automata

Reals/Integers/Rationals/Natural numbers: $(\mathbf{R},+,\times,0,1)$ — Computes the number of runs of the NDA

« Rat semiring »: $(\mathrm{Rat}(A), \cup, \cdot, \varnothing, \{\varepsilon\})$ — Rational transducers

Tropical semiring: $(\mathbf{R}\cup\{-\infty\},\max,+,-\infty,0)$ — Tropical automata

$(\mathbf{R}\cup\{+\infty\},\min,+,+\infty,0)$, $(\mathbf{N}\cup\{-\infty\},\max,+,-\infty,0)$, $(\mathbf{N}\cup\{+\infty\},\min,+,+\infty,0)$

# Tropical automata

# Tropical automata

$$L(a_1 a_2 \ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

# Tropical automata

$$L(a_1 a_2 \ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

$(\mathbf{N} \cup \{-\infty\}, \max, +, -\infty, 0)$

$L(u) \geq n$ if and only if ($\exists$ run $\rho$ over $u$) $\text{weight}(\rho) \geq n$

# Tropical automata

$$L(a_1 a_2 \ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

$(\mathbf{N} \cup \{-\infty\}, \max, +, -\infty, 0)$

$L(u) \geq n$ if and only if ($\exists$ run $\rho$ over u) weight($\rho$)$\geq n$

$(\mathbf{N} \cup \{\infty\}, \min, +, \infty, 0)$

$L(u) \geq n$ if and only if ($\forall$ run $\rho$ over u) weight($\rho$)$\geq n$

# Tropical automata

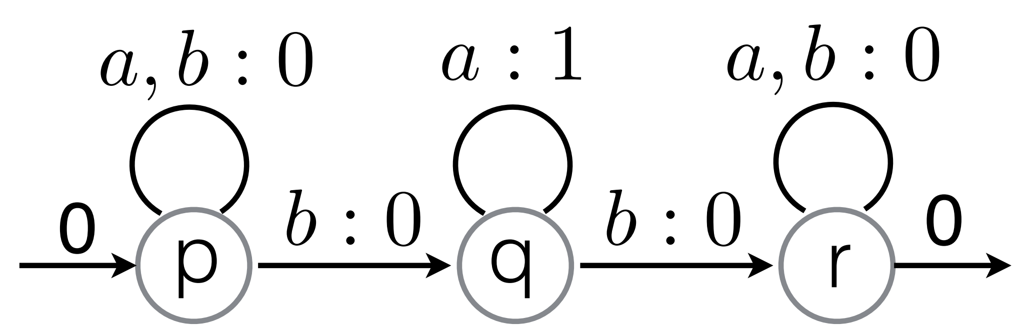
$$L(a_1 a_2 \ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

$(\mathbf{N} \cup \{-\infty\}, \max, +, -\infty, 0)$

$L(u) \geq n$ if and only if ($\exists$ run $\rho$ over $u$) $\mathrm{weight}(\rho) \geq n$

$(\mathbf{N} \cup \{\infty\}, \min, +, \infty, 0)$

$L(u) \geq n$ if and only if ($\forall$ run $\rho$ over $u$) $\mathrm{weight}(\rho) \geq n$



# Tropical automata

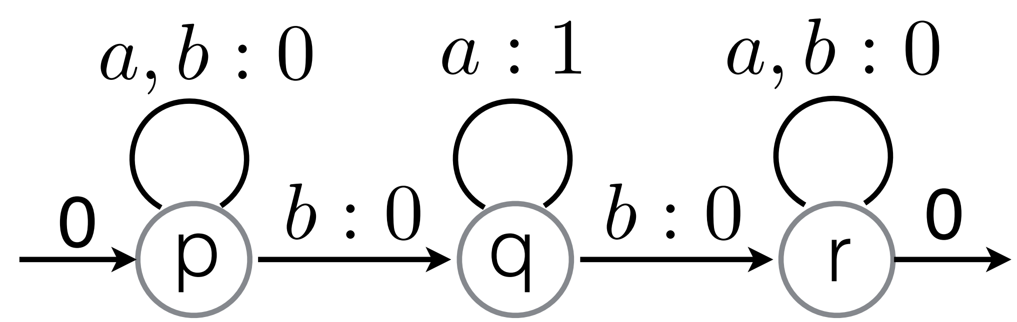
$$L(a_1a_2\ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

$(\mathbf{N}\cup\{-\infty\},\max,+,-\infty,0)$
$L(u)\geq n$ if and only if ($\exists$ run $\rho$ over u) weight($\rho$)$\geq n$

$(\mathbf{N}\cup\{\infty\},\min,+,\infty,0)$
$L(u)\geq n$ if and only if ($\forall$ run $\rho$ over u) weight($\rho$)$\geq n$

by convention zero-transitions ($-\infty$/$+\infty$)
are not displayed

(neutral for $\otimes$ and absorbing for $\otimes$)



# Tropical automata

$$L(a_1 a_2 \ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

$(\mathbf{N} \cup \{-\infty\}, \max, +, -\infty, 0)$

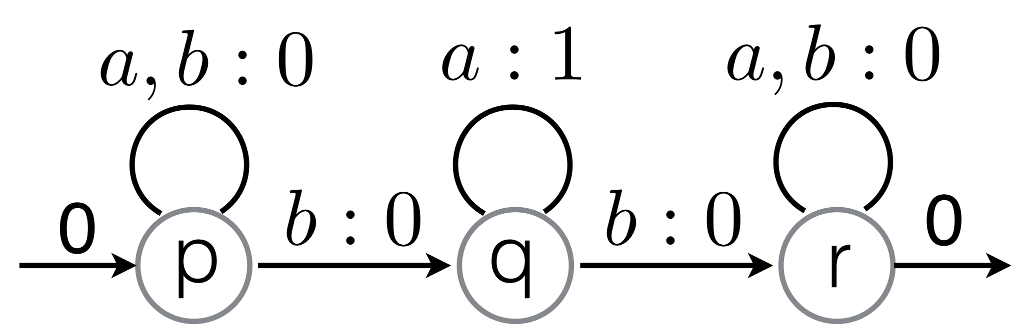
$L(u) \geq n$ if and only if ($\exists$ run $\rho$ over $u$) $\mathrm{weight}(\rho) \geq n$

$(\mathbf{N} \cup \{\infty\}, \min, +, \infty, 0)$

$L(u) \geq n$ if and only if ($\forall$ run $\rho$ over $u$) $\mathrm{weight}(\rho) \geq n$

by convention zero-transitions ($-\infty$/$+\infty$) are not displayed

(neutral for $\otimes$ and absorbing for $\otimes$)




The max-plus automaton computes:

# Tropical automata

$$L(a_1a_2\ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

$(\mathbf{N}\cup\{-\infty\},\max,+,-\infty,0)$

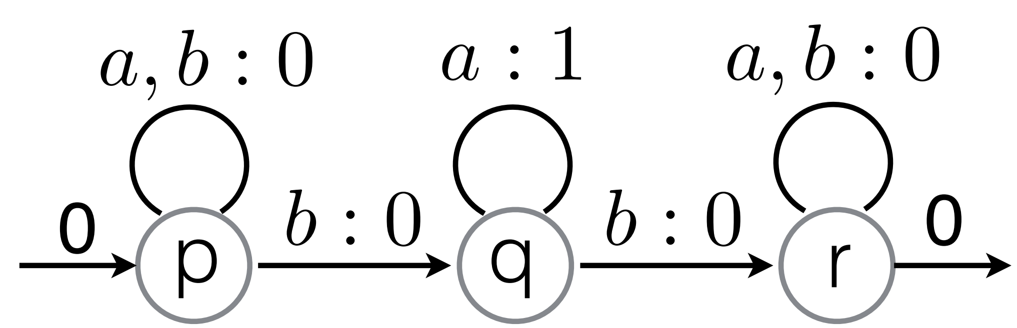
$L(u)\geq n$ if and only if $(\exists$ run $\rho$ over $u)$ weight$(\rho)\geq n$

$(\mathbf{N}\cup\{\infty\},\min,+,\infty,0)$

$L(u)\geq n$ if and only if $(\forall$ run $\rho$ over $u)$ weight$(\rho)\geq n$

by convention zero-transitions ($-\infty/+\infty$) are not displayed

(neutral for $\otimes$ and absorbing for $\otimes$)




The max-plus automaton computes:

$L_A: \quad A^* \rightarrow \mathbf{N}\cup\{-\infty\}$

$\qquad u \longmapsto$

# Tropical automata

$$L(a_1a_2\ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \Big( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \Big) \otimes F(q_n)$$

($\mathbf{N}\cup\{-\infty\}$,max,+,$-\infty$,0)

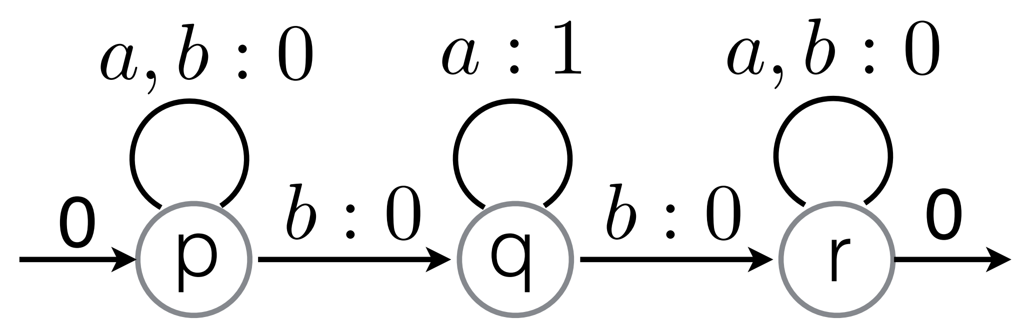
$L(u)\geq n$ if and only if ($\exists$ run $\rho$ over u) weight($\rho$)$\geq n$

($\mathbf{N}\cup\{\infty\}$,min,+,$\infty$,0)

$L(u)\geq n$ if and only if ($\forall$ run $\rho$ over u) weight($\rho$)$\geq n$

by convention zero-transitions ($-\infty$/$+\infty$)
are not displayed

(neutral for $\otimes$ and absorbing for $\otimes$)




The max-plus automaton computes:

$L_A$: $A^* \rightarrow \mathbf{N}\cup\{-\infty\}$

u $\longmapsto$ the size of the longest block of consecutive a's surrounded by 2 b's

# Tropical automata

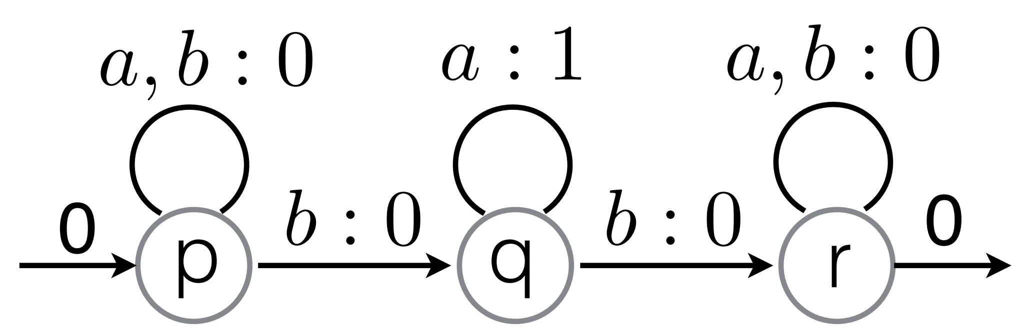
$$L(a_1 a_2 \ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

($\mathbf{N} \cup \{-\infty\}$, max, +, $-\infty$, 0)
L(u)≥n  if and only if  (∃ run ρ over u) weight(ρ)≥n

($\mathbf{N} \cup \{\infty\}$, min, +, $\infty$, 0)
L(u)≥n  if and only if  (∀ run ρ over u) weight(ρ)≥n

by convention zero-transitions ($-\infty$/$+\infty$) are not displayed

(neutral for ⊗ and absorbing for ⊗)




[Krob 94] The equality of max-plus definable functions is undecidable.

The max-plus automaton computes:

$L_A$: $A^* \to \mathbf{N} \cup \{-\infty\}$

u ⟼ the size of the longest block of consecutive a's surrounded by 2 b's

# Tropical automata

$$L(a_1 a_2 \ldots a_n) = \bigoplus_{p_0,\ldots,p_n} I(q_0) \otimes \left( \bigotimes_{i=1}^{n} \Delta(q_{i-1},a_i,q_i) \right) \otimes F(q_n)$$

($\mathbf{N}\cup\{-\infty\}$,max,+,$-\infty$,0)

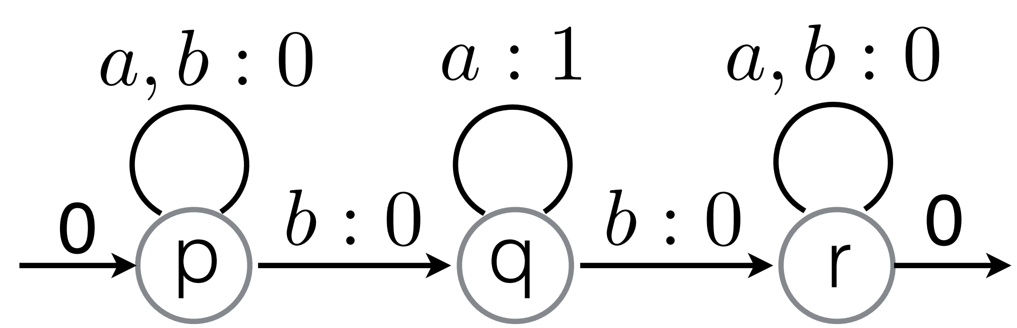
$L(u)\geq n$ if and only if ($\exists$ run $\rho$ over u) weight($\rho$)$\geq n$

($\mathbf{N}\cup\{\infty\}$,min,+,$\infty$,0)

$L(u)\geq n$ if and only if ($\forall$ run $\rho$ over u) weight($\rho$)$\geq n$

by convention zero-transitions ($-\infty$/$+\infty$) are not displayed

(neutral for $\otimes$ and absorbing for $\otimes$)




[Krob 94] The equality of max-plus definable functions is undecidable.

The max-plus automaton computes:

$L_A$: $A^* \rightarrow \mathbf{N}\cup\{-\infty\}$

u $\longmapsto$ the size of the longest block of consecutive a's surrounded by 2 b's

[Hashiguchi 81] The boundedness of distance automata is decidable.

[Leung88] [Simon78,94] [Kirsten05] [C. & Bojanczyk 06] [C. 09] [Bojanczyk15]

# Alternation of quantifiers

# Alternation of quantifiers

Emptiness of NDA ?

(∃ word w) (∃ run ρ over w) ρ is accepting

# Alternation of quantifiers

Emptiness of NDA ?

NL-c

(∃ word w) (∃ run ρ over w) ρ is accepting

# Alternation of quantifiers

Emptiness of NDA ?

NL-c

(∃ word w) (∃ run ρ over w) ρ is accepting

Universality of NDA ?

(∀ word w) (∃ run ρ over w) ρ is accepting

# Alternation of quantifiers

Emptiness of NDA ? NL-c

(∃ word w) (∃ run ρ over w) ρ is accepting

Universality of NDA ? PSPACE-c (powerset)

(∀ word w) (∃ run ρ over w) ρ is accepting

# Alternation of quantifiers

Emptiness of NDA ? NL-c

(∃ word w) (∃ run ρ over w) ρ is accepting

Universality of NDA ? PSPACE-c (powerset)

(∀ word w) (∃ run ρ over w) ρ is accepting

Is a ($\mathbf{Z} \cup \{\infty\}$,max,+) automaton ≤ 0 ?

(∀ word w) (∀ run ρ over w) weight(ρ)≤0

# Alternation of quantifiers

Emptiness of NDA ? NL-c

(∃ word w) (∃ run ρ over w) ρ is accepting

Universality of NDA ? PSPACE-c (powerset)

(∀ word w) (∃ run ρ over w) ρ is accepting

Is a ($\mathbf{Z} \cup \{\infty\}$,max,+) automaton ≤ 0 ? NL-c

(∀ word w) (∀ run ρ over w) weight(ρ)≤0

# Alternation of quantifiers

Emptiness of NDA ? NL-c

(∃ word w) (∃ run ρ over w) ρ is accepting

Universality of NDA ? PSPACE-c (powerset)

(∀ word w) (∃ run ρ over w) ρ is accepting

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton ≤ 0 ? NL-c

(∀ word w) (∀ run ρ over w) weight(ρ)≤0

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton ≥ 0 ?

(∀ word w) (∃ run ρ over w) weight(ρ)≥0

# Alternation of quantifiers

Emptiness of NDA ? — NL-c

(∃ word w) (∃ run ρ over w) ρ is accepting

Universality of NDA ? — PSPACE-c (powerset)

(∀ word w) (∃ run ρ over w) ρ is accepting

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton ≤ 0 ? — NL-c

(∀ word w) (∀ run ρ over w) weight(ρ)≤0

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton ≥ 0 ? — undecidable [Krob92, other form]

(∀ word w) (∃ run ρ over w) weight(ρ)≥0

# Alternation of quantifiers

Emptiness of NDA ? NL-c

(∃ word w) (∃ run ρ over w) ρ is accepting

Universality of NDA ? PSPACE-c (powerset)

(∀ word w) (∃ run ρ over w) ρ is accepting

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton ≤ 0 ? NL-c

(∀ word w) (∀ run ρ over w) weight(ρ)≤0

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton ≥ 0 ? undecidable [Krob92, other form]

(∀ word w) (∃ run ρ over w) weight(ρ)≥0

Is a ($\mathbf{N}\cup\{-\infty\}$,max,+) automaton bounded?

(∃ n∈**N**) (∀ word w) (∀ run ρ over w) weight(ρ)≤n

# Alternation of quantifiers

Emptiness of NDA ? NL-c

($\exists$ word w) ($\exists$ run ρ over w) ρ is accepting

Universality of NDA ? PSPACE-c (powerset)

($\forall$ word w) ($\exists$ run ρ over w) ρ is accepting

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton $\leq 0$ ? NL-c

($\forall$ word w) ($\forall$ run ρ over w) weight(ρ)≤0

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton $\geq 0$ ? undecidable [Krob92, other form]

($\forall$ word w) ($\exists$ run ρ over w) weight(ρ)≥0

Is a ($\mathbf{N}\cup\{-\infty\}$,max,+) automaton bounded? NL-c

($\exists$ n$\in\mathbf{N}$) ($\forall$ word w) ($\forall$ run ρ over w) weight(ρ)≤n

# Alternation of quantifiers

Emptiness of NDA ? NL-c

(∃ word w) (∃ run ρ over w) ρ is accepting

Universality of NDA ? PSPACE-c (powerset)

(∀ word w) (∃ run ρ over w) ρ is accepting

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton $\leq 0$ ? NL-c

(∀ word w) (∀ run ρ over w) weight(ρ)≤0

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton $\geq 0$ ? undecidable [Krob92, other form]

(∀ word w) (∃ run ρ over w) weight(ρ)≥0

Is a ($\mathbf{N}\cup\{-\infty\}$,max,+) automaton bounded? NL-c

(∃ $n\in\mathbf{N}$) (∀ word w) (∀ run ρ over w) weight(ρ)≤n

Is a ($\mathbf{N}\cup\{\infty\}$,min,+) automaton bounded?

(∃ $n\in\mathbf{N}$) (∀ word w) (∃ run ρ over w) weight(ρ)≤n

# Alternation of quantifiers

Emptiness of NDA ? — NL-c

(∃ word w) (∃ run ρ over w) ρ is accepting

Universality of NDA ? — PSPACE-c (powerset)

(∀ word w) (∃ run ρ over w) ρ is accepting

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton ≤ 0 ? — NL-c

(∀ word w) (∀ run ρ over w) weight(ρ)≤0

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton ≥ 0 ? — undecidable [Krob92, other form]

(∀ word w) (∃ run ρ over w) weight(ρ)≥0

Is a ($\mathbf{N}\cup\{-\infty\}$,max,+) automaton bounded? — NL-c

(∃ n∈**N**) (∀ word w) (∀ run ρ over w) weight(ρ)≤n

Is a ($\mathbf{N}\cup\{\infty\}$,min,+) automaton bounded? — PSPACE-c [Hashiguchi81,Leung84]

(∃ n∈**N**) (∀ word w) (∃ run ρ over w) weight(ρ)≤n

# Alternation of quantifiers

Emptiness of NDA ? — NL-c

(∃ word w) (∃ run ρ over w) ρ is accepting

Universality of NDA ? — PSPACE-c (powerset)

(∀ word w) (∃ run ρ over w) ρ is accepting

Is a (**Z**∪{∞},max,+) automaton ≤ 0 ? — NL-c

(∀ word w) (∀ run ρ over w) weight(ρ)≤0

Is a (**Z**∪{∞},max,+) automaton ≥ 0 ? — undecidable [Krob92, other form]

(∀ word w) (∃ run ρ over w) weight(ρ)≥0

Is a (**N**∪{-∞},max,+) automaton bounded? — NL-c

(∃ n∈**N**) (∀ word w) (∀ run ρ over w) weight(ρ)≤n

Is a (**N**∪{∞},min,+) automaton bounded? — PSPACE-c [Hashiguchi81,Leung84]

(∃ n∈**N**) (∀ word w) (∃ run ρ over w) weight(ρ)≤n

Given a (**N**∪{∞},max,+) automaton, find the least θ∈[0,1] such that

(∃ a) (∀ s∈**N**) (∃ word w, |w|≥s) (∀ run ρ over w) weight(ρ) ≤ $as^{\theta}$

# Alternation of quantifiers

Emptiness of NDA ? NL-c

(∃ word w) (∃ run ρ over w) ρ is accepting

Universality of NDA ? PSPACE-c (powerset)

(∀ word w) (∃ run ρ over w) ρ is accepting

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton ≤ 0 ? NL-c

(∀ word w) (∀ run ρ over w) weight(ρ)≤0

Is a ($\mathbf{Z}\cup\{\infty\}$,max,+) automaton ≥ 0 ? undecidable [Krob92, other form]

(∀ word w) (∃ run ρ over w) weight(ρ)≥0

Is a ($\mathbf{N}\cup\{-\infty\}$,max,+) automaton bounded? NL-c

(∃ n∈**N**) (∀ word w) (∀ run ρ over w) weight(ρ)≤n

Is a ($\mathbf{N}\cup\{\infty\}$,min,+) automaton bounded? PSPACE-c

(∃ n∈**N**) (∀ word w) (∃ run ρ over w) weight(ρ)≤n [Hashiguchi81,Leung84]

Given a ($\mathbf{N}\cup\{\infty\}$,max,+) automaton, find the least θ∈[0,1] such that

(∃ a) (∀ s∈**N**) (∃ word w, |w|≥s) (∀ run ρ over w) weight(ρ) ≤ $as^\theta$

[C., Daviaud, Zuleger 14] This θ exists and is rational.
Furthermore, it can be constructed in EXPSPACE, likely to be PSPACE-complete.

# More on asymptotic analysis

# More on asymptotic analysis

Given a ($\mathbf{N}\cup\{\infty\}$,max,+) automaton, find the least $\theta\in[0,1]$ such that

$(\exists\ a)\ (\forall\ s\in\mathbf{N})\ (\exists \text{ word } w, |w|\geq s)\ (\forall \text{ run } \rho \text{ over } w)\ \ \text{weight}(\rho) \leq a s^{\theta}$

# More on asymptotic analysis

Given a $(\mathbf{N} \cup \{\infty\}, \max, +)$ automaton, find the least $\theta \in [0,1]$ such that

$(\exists\ a)\ (\forall\ s \in \mathbf{N})\ (\exists \text{ word } w, |w| \geq s)\ (\forall \text{ run } \rho \text{ over } w)\ \ \text{weight}(\rho) \leq a s^{\theta}$

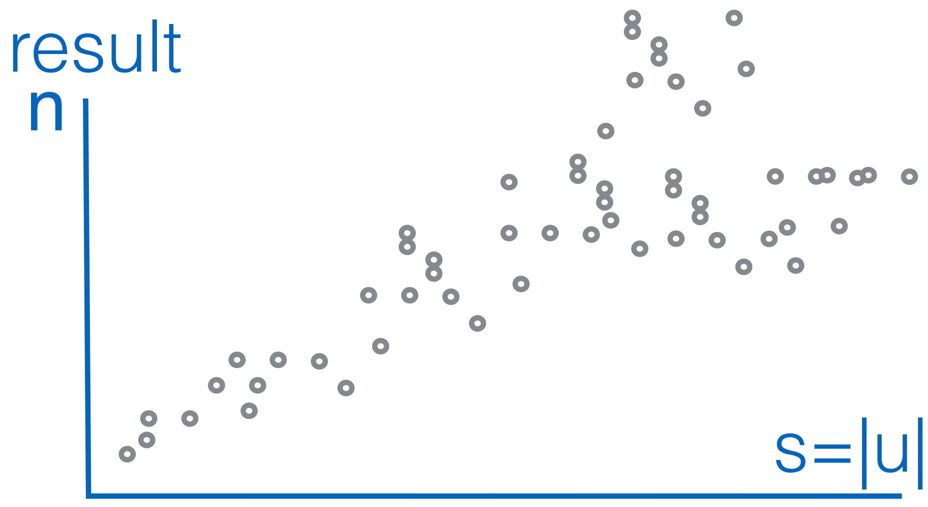
[C., Daviaud, Zuleger 14] This $\theta$ exists and is rational.
Furthermore, it can be constructed in EXPSPACE, likely to be PSPACE-complete.

# More on asymptotic analysis

Given a $(\mathbf{N} \cup \{\infty\}, \max, +)$ automaton, find the least $\theta \in [0,1]$ such that

$(\exists\ a)\ (\forall\ s \in \mathbf{N})\ (\exists \text{ word } w, |w| \geq s)\ (\forall \text{ run } \rho \text{ over } w)\ \ \text{weight}(\rho) \leq a s^{\theta}$

[C., Daviaud, Zuleger 14] This $\theta$ exists and is rational.
Furthermore, it can be constructed in EXPSPACE, likely to be PSPACE-complete.

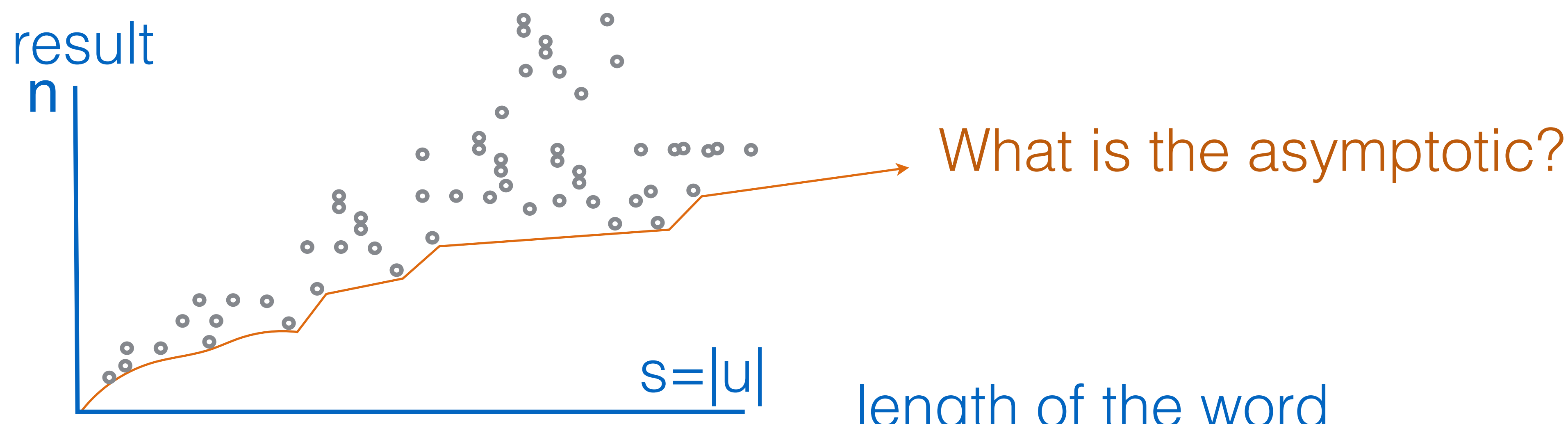



length of the word

# More on asymptotic analysis

Given a **(N**∪{∞},max,+) automaton, find the least $\theta \in [0,1]$ such that
$(\exists\, a)\ (\forall\, s \in \mathbf{N})\ (\exists \text{ word } w, |w| \geq s)\ (\forall \text{ run } \rho \text{ over } w)\ \ \mathrm{weight}(\rho) \leq a s^{\theta}$

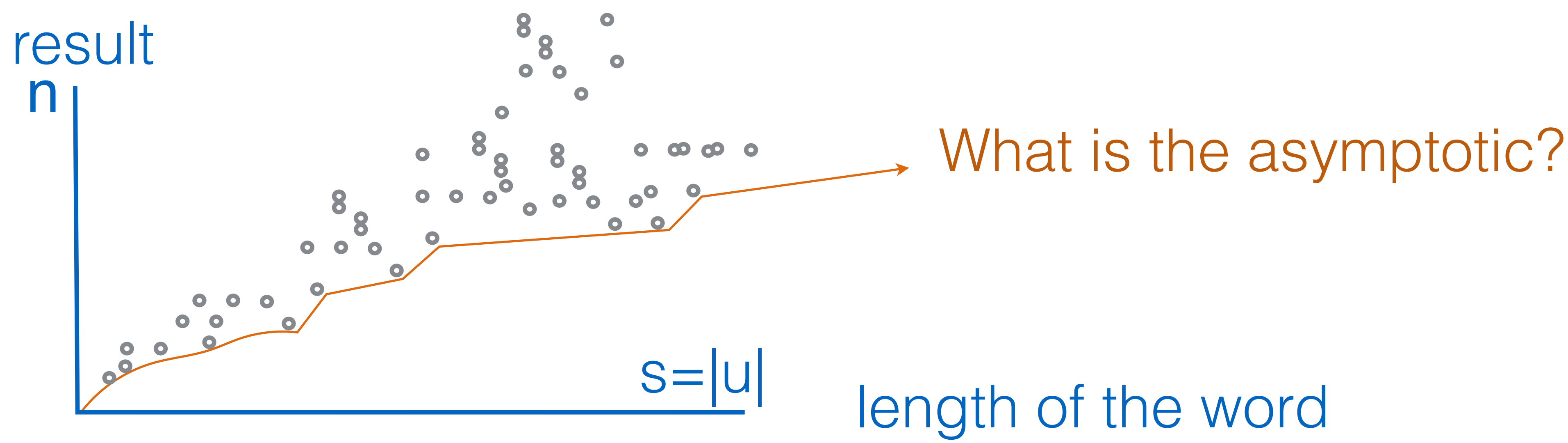
[C., Daviaud, Zuleger 14] This θ exists and is rational.
Furthermore, it can be constructed in EXPSPACE, likely to be PSPACE-complete.



# More on asymptotic analysis

Given a ($\mathbf{N}\cup\{\infty\}$,max,+) automaton, find the least $\theta\in[0,1]$ such that
($\exists$ a) ($\forall$ s$\in\mathbf{N}$) ($\exists$ word w, |w|$\geq$s) ($\forall$ run $\rho$ over w) weight($\rho$) $\leq$ a$s^\theta$

[C., Daviaud, Zuleger 14] This $\theta$ exists and is rational.
Furthermore, it can be constructed in EXPSPACE, likely to be PSPACE-complete.



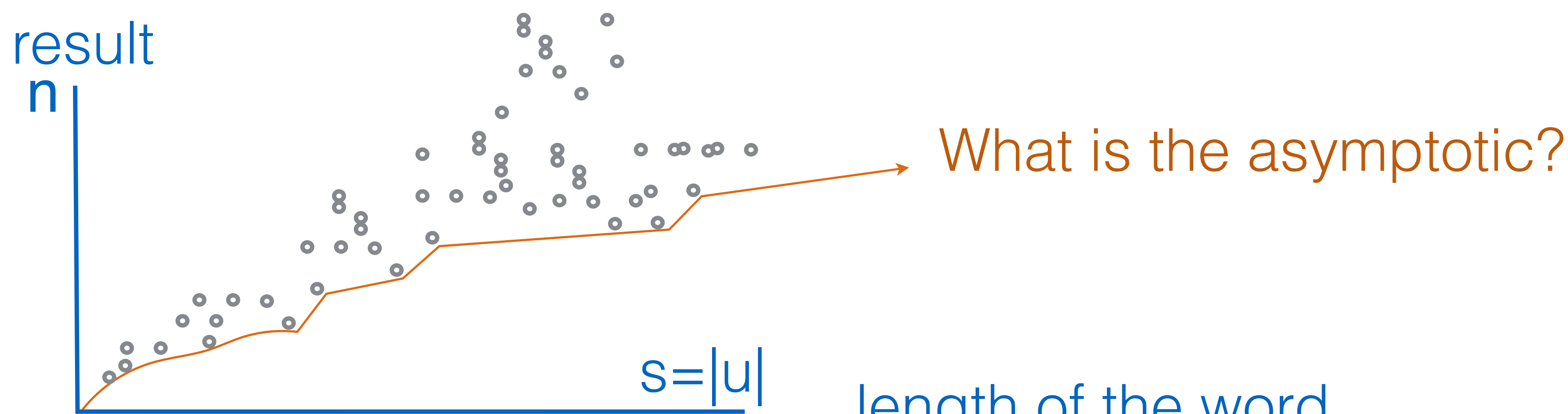

Compute: $$\liminf_{u\in A^*} \frac{\log f(u)}{\log |u|} = \theta$$

find the least value of a word
of length at least s

# More on asymptotic analysis

Given a $(\mathbf{N}\cup\{\infty\},\max,+)$ automaton, find the least $\theta\in[0,1]$ such that
$(\exists\, a)\ (\forall\, s\in\mathbf{N})\ (\exists \text{ word } w, |w|\geq s)\ (\forall \text{ run } \rho \text{ over } w)\ \ \text{weight}(\rho)\leq a s^{\theta}$

[C., Daviaud, Zuleger 14] This $\theta$ exists and is rational.
Furthermore, it can be constructed in EXPSPACE, likely to be PSPACE-complete.




length of the word

Compute: $$\liminf_{u\in A^*}\frac{\log f(u)}{\log|u|}=\theta \quad\Longleftrightarrow\quad \limsup_{u\in A^*}\frac{\log|u|}{\log f(|u|)}=\frac{1}{\theta}$$

find the least value of a word
of length at least s

find the longest size of a word
of value at most n

# Ingredients of the proof

# Ingredients of the proof

Ingredient 1.

Given a set of words W, collect an information I(W) sufficient for understanding its behavior in any context.

# Ingredients of the proof

Ingredient 1.

Given a set of words W, collect an information I(W) sufficient for understanding its behavior in any context.

e.g. for universality $I(W) = \{P \subseteq Q : P = \mathrm{Reach}(I,u) \text{ for some } u \in W\}$

# Ingredients of the proof

Ingredient 1.

Given a set of words W, collect an information $I(W)$ sufficient for understanding its behavior in any context.

e.g. for universality $I(W) = \{P \subseteq Q : P = \mathrm{Reach}(I,u) \text{ for some } u \in W\}$

In our case,

$I(W) = \{ f{:}Q \times Q \to \mathbf{N}$ : there is a run that displays this behavior $\} \subseteq P(N^{Q \times Q})$

# Ingredients of the proof

Ingredient 1.

Given a set of words W, collect an information I(W) sufficient for understanding its behavior in any context.

e.g. for universality $I(W) = \{P \subseteq Q : P = \mathrm{Reach}(I,u) \text{ for some } u \in W\}$

In our case,

$I(W) = \{ f{:}Q \times Q \to \mathbf{N} : \text{there is a run that displays this behavior} \} \subseteq P(N^{Q\times Q})$

Ingredient 2.

Give a notion of approximation for such sets: Hausdorff-like keeping asymptotes.

# Ingredients of the proof

Ingredient 1.

Given a set of words W, collect an information $I(W)$ sufficient for understanding its behavior in any context.

e.g. for universality $I(W) = \{P \subseteq Q : P = \mathrm{Reach}(I,u) \text{ for some } u \in W\}$

In our case,
$I(W) = \{\ f{:}Q\times Q \to \mathbf{N} : \text{there is a run that displays this behavior}\ \} \subseteq P(N^{Q\times Q})$

Ingredient 2.

Give a notion of approximation for such sets: Hausdorff-like keeping asymptotes.

Ingredient 3.

Define presentable sets families of such sets of maps that are nicely behaved (that can be algorithmically handled). In our case unions of convex polytopes in $R^{Q\times Q}$ representing simultaneous asymptotic behaviors.

# Ingredients of the proof

Ingredient 1.

Given a set of words W, collect an information I(W) sufficient for understanding its behavior in any context.

e.g. for universality $I(W) = \{P \subseteq Q : P = \mathrm{Reach}(I,u) \text{ for some } u \in W\}$

In our case,

$I(W) = \{ f : Q \times Q \to \mathbf{N} : \text{there is a run that displays this behavior} \} \subseteq P(N^{Q \times Q})$

Ingredient 2.

Give a notion of approximation for such sets: Hausdorff-like keeping asymptotes.

Ingredient 3.

Define presentable sets families of such sets of maps that are nicely behaved (that can be algorithmically handled). In our case unions of convex polytopes in $R^{Q \times Q}$ representing simultaneous asymptotic behaviors.

Step 4.

Compute a presentable equivalent (up to approximation) of $I(A^*)$

# Ingredients of the proof

**Ingredient 1.**

Given a set of words W, collect an information I(W) sufficient for understanding its behavior in any context.

e.g. for universality $I(W) = \{P \subseteq Q : P = \mathrm{Reach}(I,u) \text{ for some } u \in W\}$

In our case,

$I(W) = \{ f : Q \times Q \to \mathbf{N} : \text{there is a run that displays this behavior} \} \subseteq P(N^{Q \times Q})$

**Ingredient 2.**

Give a notion of approximation for such sets: Hausdorff-like keeping asymptotes.

**Ingredient 3.**

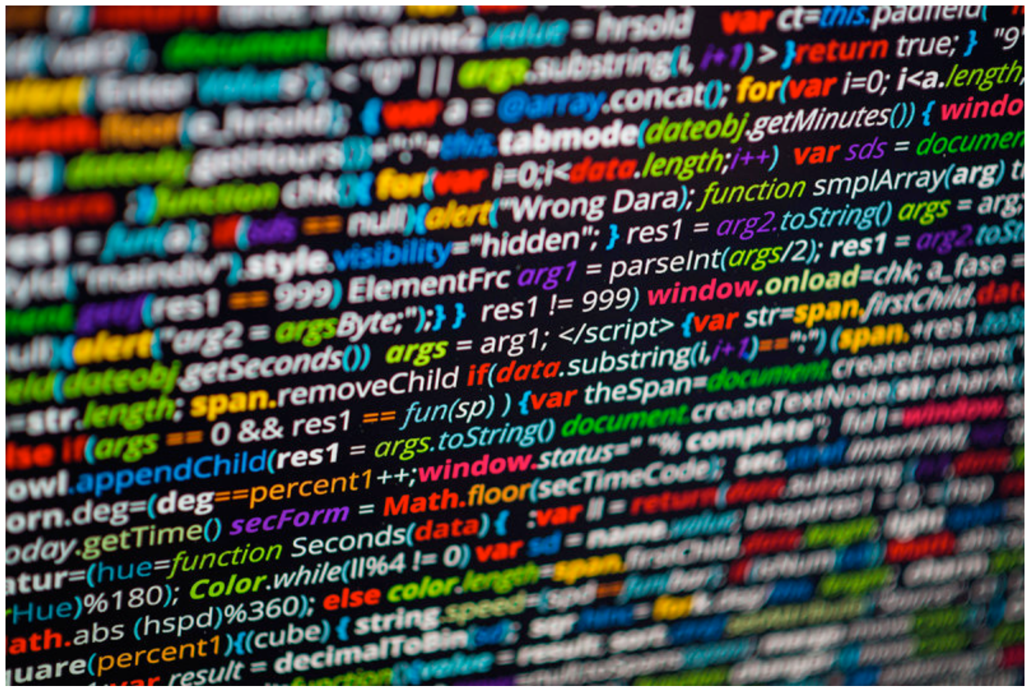
Define presentable sets families of such sets of maps that are nicely behaved (that can be algorithmically handled). In our case unions of convex polytopes in $R^{Q \times Q}$ representing simultaneous asymptotic behaviors.

**Step 4.**

Compute a presentable equivalent (up to approximation) of $I(A^*)$
This is done by induction of the factorisation forest height [Simon].

# Program analysis
# and
# the size-change abstraction

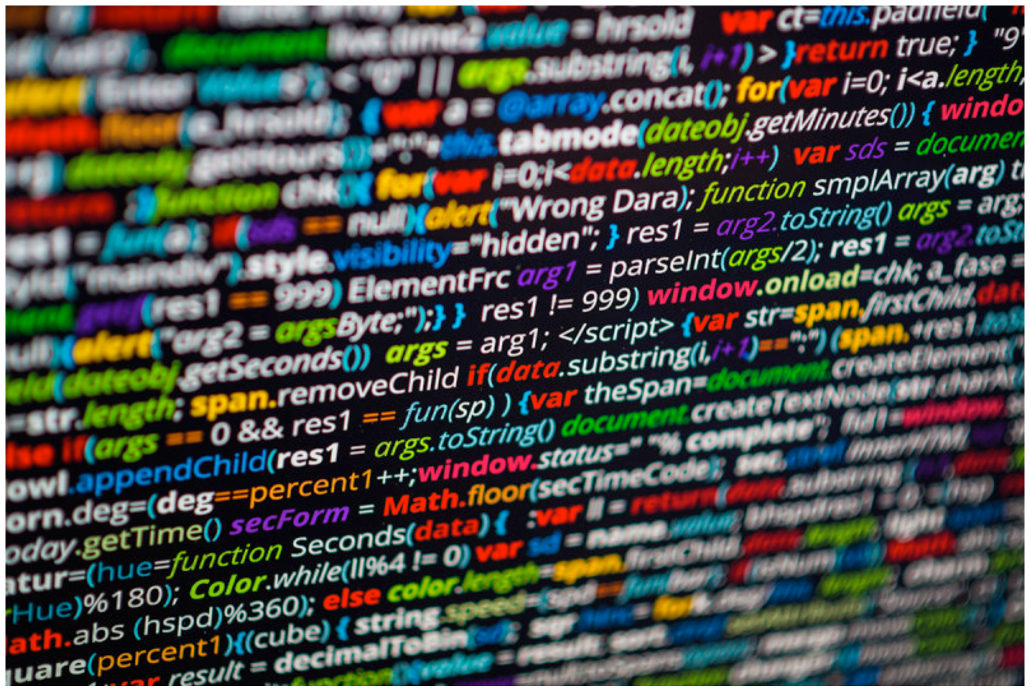
# Program analysis



Given an input program/piece of program:

- Does it perform a zero division?
- Does it access a non-allocated memory area?
- Is there a dynamic type problem?
- Does it comply to the specification?
- Is there a memory leakage?
- Does it terminate?
- What is its running time?

# Program analysis

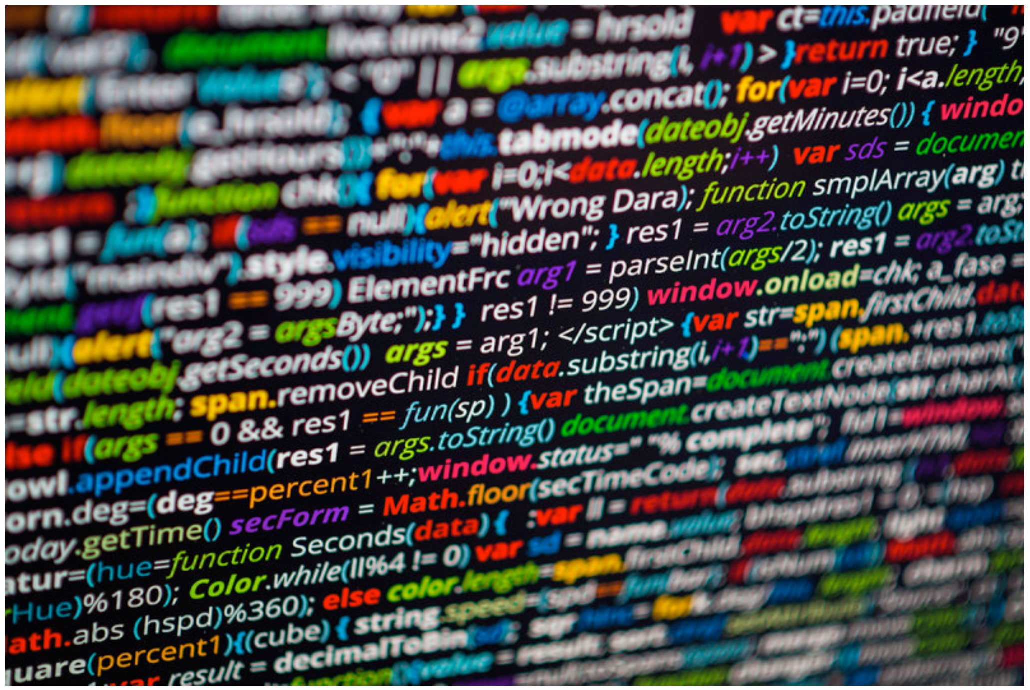


Given an input program/piece of program:

- Does it perform a zero division?
- Does it access a non-allocated memory area?
- Is there a dynamic type problem?
- Does it comply to the specification?
- Is there a memory leakage?
- Does it terminate?
- What is its running time?

[Rice-like] Essentially, all these questions are undecidable.

# Program analysis

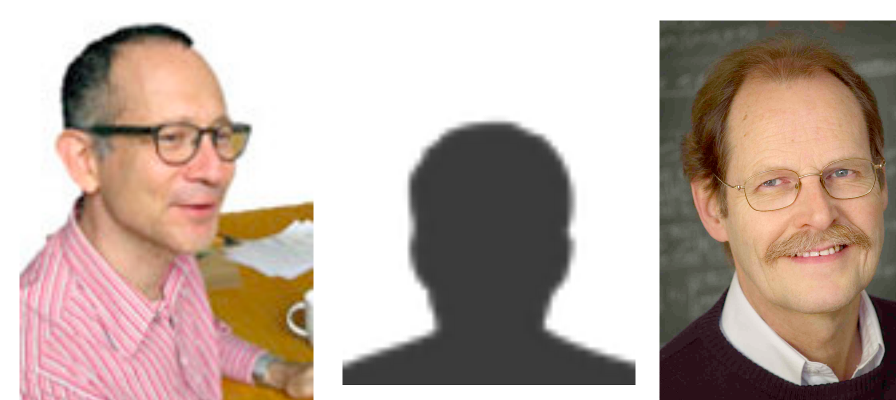



Given an input program/piece of program:

- Does it perform a zero division?
- Does it access a non-allocated memory area?
- Is there a dynamic type problem?
- Does it comply to the specification?
- Is there a memory leakage?
- Does it terminate?
- What is its running time?

[Rice-like] Essentially, all these questions are undecidable.

**Solution here**: in this talk, we use the size-change abstract model ([Ben-Amram, Chin Soon Lee, Neil D. Jones 01]).



# Example

```
void main() {
 uint x,y;
 x = read_input();
 y = read_input();
 while (x > 0) {
   if (y > 0)
     { y--; }
   else
     { y = read_input();
       x--;  }
  }
}
```

# Example

```
void main() {
 uint x,y;
 x = read_input();
 y = read_input();
 while (x > 0) {
   if (y > 0)
     { y--; }
   else
     { y = read_input();
      x--;  }
  }
}
```

these variables remain non-negative.

# Example

```
void main() {
  uint x,y;
  x = read_input();
  y = read_input();
  while (x > 0) {
    if (y > 0)
      { y--; }
    else
      { y = read_input();
        x--;  }
  }
}
```

these variables remain non-negative.

are initialized with an uncontrolled value

# Example

```
void main() {
  uint x,y;
  x = read_input();
  y = read_input();
  while (x > 0) {
    if (y > 0)
      { y--; }
    else
      { y = read_input();
        x--;  }
  }
}
```

these variables remain non-negative.

are initialized with an uncontrolled value

either y decreases

# Example

```
void main() {
 uint x,y;
 x = read_input();
 y = read_input();
 while (x > 0) {
   if (y > 0)
      { y--; }
   else
      { y = read_input();
        x--;  }
  }
}
```

these variables remain non-negative.

are initialized with an uncontrolled value

either y decreases

or x decreases,
and y gets an uncontrolled value

# Example

```
void main() {
 uint x,y;
 x = read_input();
 y = read_input();
 while (x > 0) {
   if (y > 0)
     { y--; }
   else
     { y = read_input();
       x--;  }
  }
}
```

these variables remain non-negative.

are initialized with an uncontrolled value

either y decreases

or x decreases,
and y gets an uncontrolled value

Remark: This program terminates.

# Example

```
void main() {
  uint x,y;
  x = read_input();
  y = read_input();
  while (x > 0) {
    if (y > 0)
      { y--; }
    else
      { y = read_input();
       x--;  }
  }
}
```

these variables remain non-negative.

are initialized with an uncontrolled value

either y decreases

or x decreases,
and y gets an uncontrolled value

Remark: This program terminates.

Question: what method can automatically establish it ?

# Principle of abstraction

# Principle of abstraction

Principle: replace the program by an abstraction:

- Information that is lost is replaced by non-determinism.
  This includes:
  + The dynamic information resulting from the interactions with the environment.
  + All the tests and computations that cannot be abstracted in the restricted model of the abstraction.
- The resulting abstraction can be analyzed: it can be decided whether the resulting abstraction stops an all its executions.
- If the abstraction stops on all its executions, then the original programs stops an all its executions.

# Principle of abstraction

Principle: replace the program by an abstraction:

- Information that is lost is replaced by non-determinism.
  This includes:
  + The dynamic information resulting from the interactions with the environment.
  + All the tests and computations that cannot be abstracted in the restricted model of the abstraction.
- The resulting abstraction can be analyzed: it can be decided whether the resulting abstraction stops an all its executions.
- If the abstraction stops on all its executions, then the original programs stops an all its executions.

Remark: Of course, this is a compromise between the efficiency of the decision problem, and the loss of information during the abstraction.

# Principle of abstraction

Principle: replace the program by an abstraction:

- Information that is lost is replaced by non-determinism.
  This includes:
  + The dynamic information resulting from the interactions with the environment.
  + All the tests and computations that cannot be abstracted in the restricted model of the abstraction.
- The resulting abstraction can be analyzed: it can be decided whether the resulting abstraction stops an all its executions.
- If the abstraction stops on all its executions, then the original programs stops an all its executions.

Remark: Of course, this is a compromise between the efficiency of the decision problem, and the loss of information during the abstraction.

⇒ In this talk, we use the model of size-change abstraction.

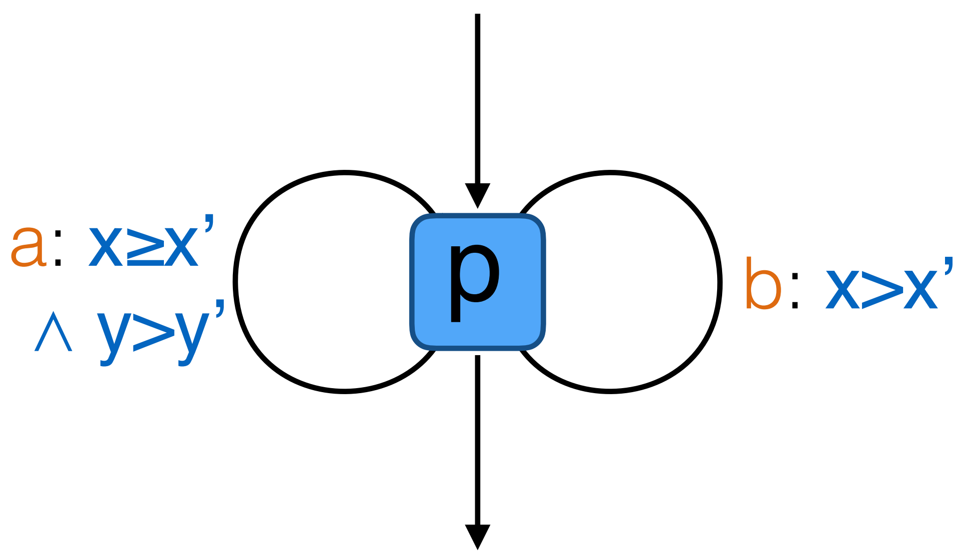
# Size-change abstraction

# Size-change abstraction

[Ben-Amram et al. 01] A size-change abstraction (SCA):
- this is a non-determininistic finite state machine
- that uses a finite set variables ($x$,$y$,$z$…) ranging over non-negative integers
- during each transition, a guards relate the variables before and after:
  $x \geq y'$ meaning « val of x before the transition ≥ val of y after the transition »
  $x > y'$ meaning « val of x before the transition > val of y after the transition »

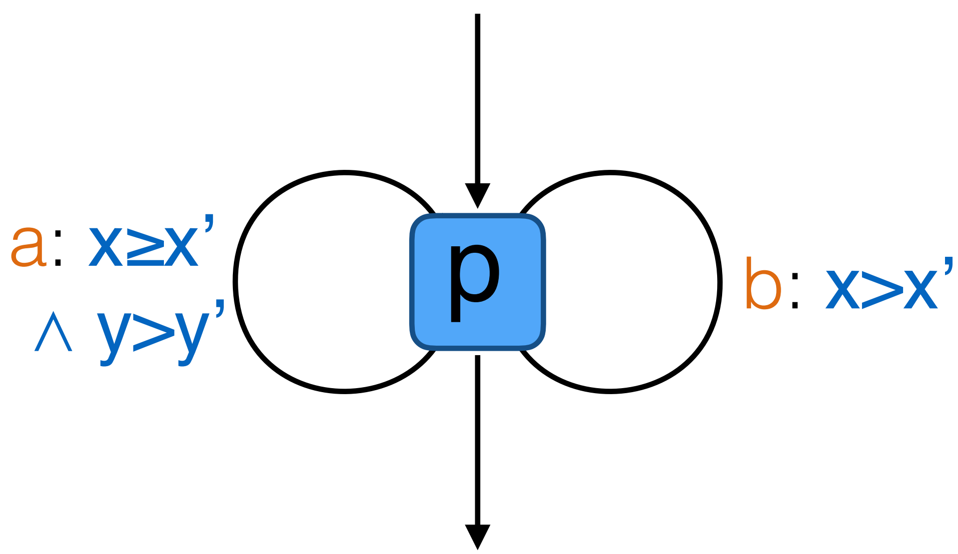
# Size-change abstraction

[Ben-Amram et al. 01] A size-change abstraction (SCA):
- this is a non-determininistic finite state machine
- that uses a finite set variables (x,y,z…) ranging over non-negative integers
- during each transition, a guards relate the variables before and after:
  - $x \geq y'$ meaning « val of x before the transition ≥ val of y after the transition »
  - $x > y'$ meaning « val of x before the transition > val of y after the transition »



# Size-change abstraction

[Ben-Amram et al. 01] A size-change abstraction (SCA):
- this is a non-determininistic finite state machine
- that uses a finite set variables (x,y,z…) ranging over non-negative integers
- during each transition, a guards relate the variables before and after:
  $x \geq y'$ meaning « val of x before the transition ≥ val of y after the transition »
  $x > y'$ meaning « val of x before the transition > val of y after the transition »




A configuration is a state together with a non-negative integer value for each of the variables.

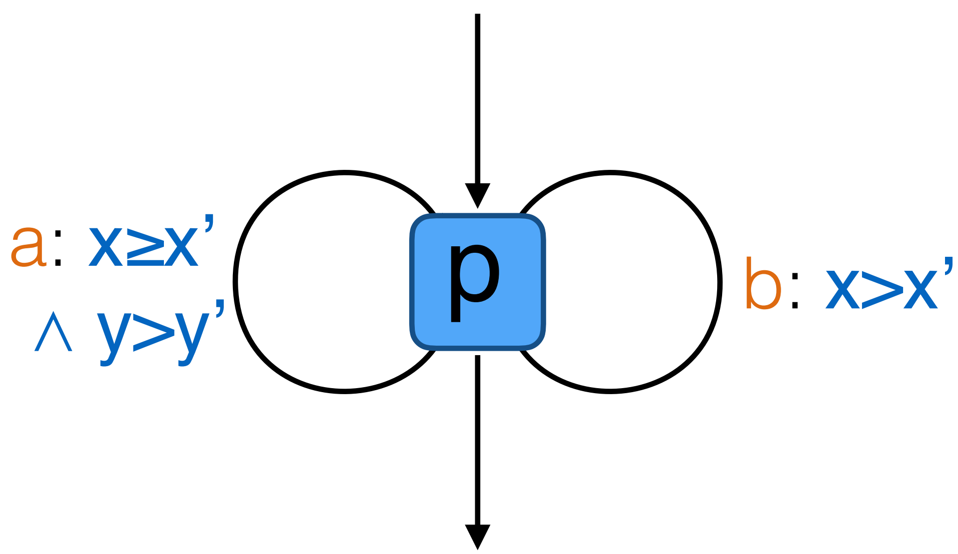
# Size-change abstraction

[Ben-Amram et al. 01] A size-change abstraction (SCA):

- this is a non-determininistic finite state machine
- that uses a finite set variables (x,y,z…) ranging over non-negative integers
- during each transition, a guards relate the variables before and after:

  x ≥ y' meaning « val of x before the transition ≥ val of y after the transition »

  x > y' meaning « val of x before the transition > val of y after the transition »




A configuration is a state together with a <u>non-negative integer</u> value for each of the variables.

A run of the SCA is a sequence of configurations that starts in an initial configuration, ends in a final one, and each consecutive configurations satisfy the guard of some possible transition.

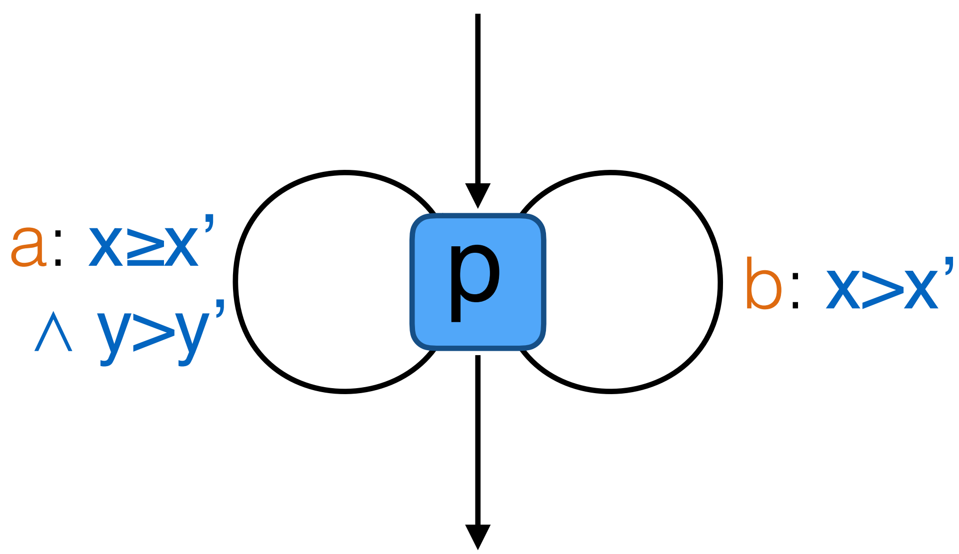
# Size-change abstraction

[Ben-Amram et al. 01] A size-change abstraction (SCA):

- this is a non-determininistic finite state machine
- that uses a finite set variables ($x$,$y$,$z$…) ranging over non-negative integers
- during each transition, a guards relate the variables before and after:

  $x \geq y'$ meaning « val of $x$ before the transition $\geq$ val of $y$ after the transition »

  $x > y'$ meaning « val of $x$ before the transition $>$ val of $y$ after the transition »




A configuration is a state together with a <u>non-negative integer</u> value for each of the variables.

A run of the SCA is a sequence of configurations that starts in an initial configuration, ends in a final one, and each consecutive configurations satisfy the guard of some possible transition.

(p,2,2) (p,2,1) (p,2,0) (p,1,2) (p,1,1) (p,0,2) (p,0,1) (p,0,0)

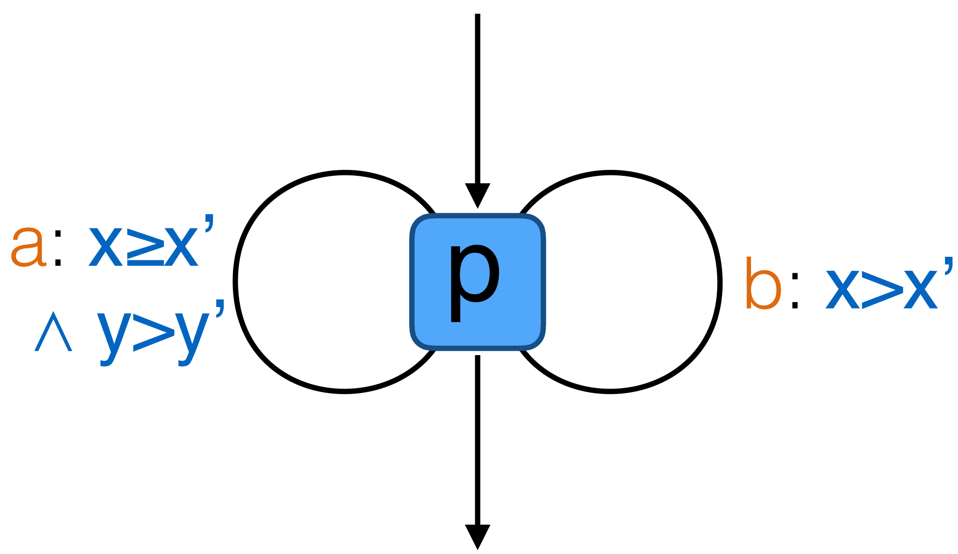
# Size-change abstraction

[Ben-Amram et al. 01] A size-change abstraction (SCA):

- this is a non-determininistic finite state machine
- that uses a finite set variables (x,y,z…) ranging over non-negative integers
- during each transition, a guards relate the variables before and after:

  x ≥ y’ meaning « val of x before the transition ≥ val of y after the transition »

  x > y’ meaning « val of x before the transition > val of y after the transition »




A configuration is a state together with a non-negative integer value for each of the variables.

A run of the SCA is a sequence of configurations that starts in an initial configuration, ends in a final one, and each consecutive configurations satisfy the guard of some possible transition.

(p,2,2) $\xrightarrow{a}$ (p,2,1) $\xrightarrow{a}$ (p,2,0) $\xrightarrow{b}$ (p,1,2) $\xrightarrow{a}$ (p,1,1) $\xrightarrow{b}$ (p,0,2) $\xrightarrow{a}$ (p,0,1) $\xrightarrow{a}$ (p,0,0)

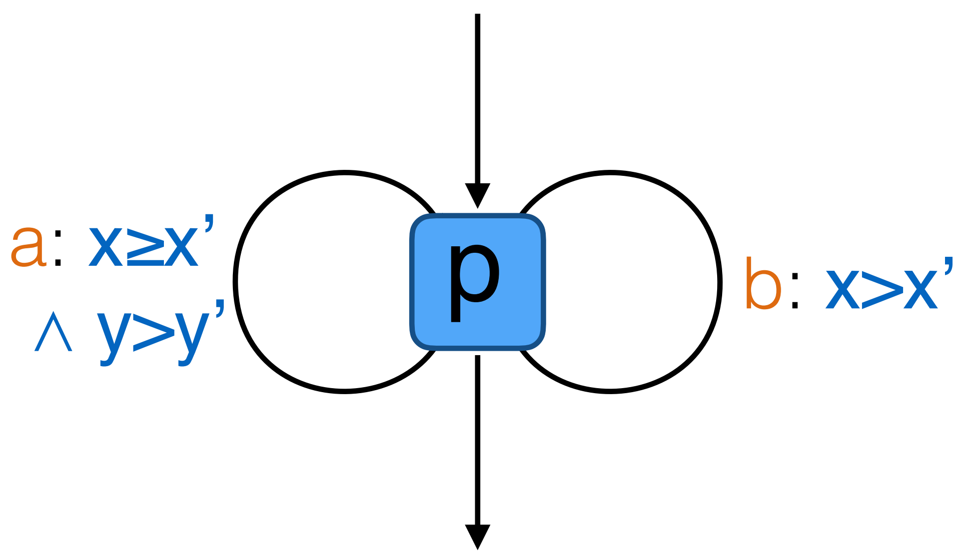
# Size-change abstraction

[Ben-Amram et al. 01] A size-change abstraction (SCA):

- this is a non-determinininistic finite state machine
- that uses a finite set variables (x,y,z…) ranging over non-negative integers
- during each transition, a guards relate the variables before and after:

  $x \geq y'$ meaning « val of x before the transition ≥ val of y after the transition »

  $x > y'$ meaning « val of x before the transition > val of y after the transition »




A configuration is a state together with a <u>non-negative integer</u> value for each of the variables.

A run of the SCA is a sequence of configurations that starts in an initial configuration, ends in a final one, and each consecutive configurations satisfy the guard of some possible transition.

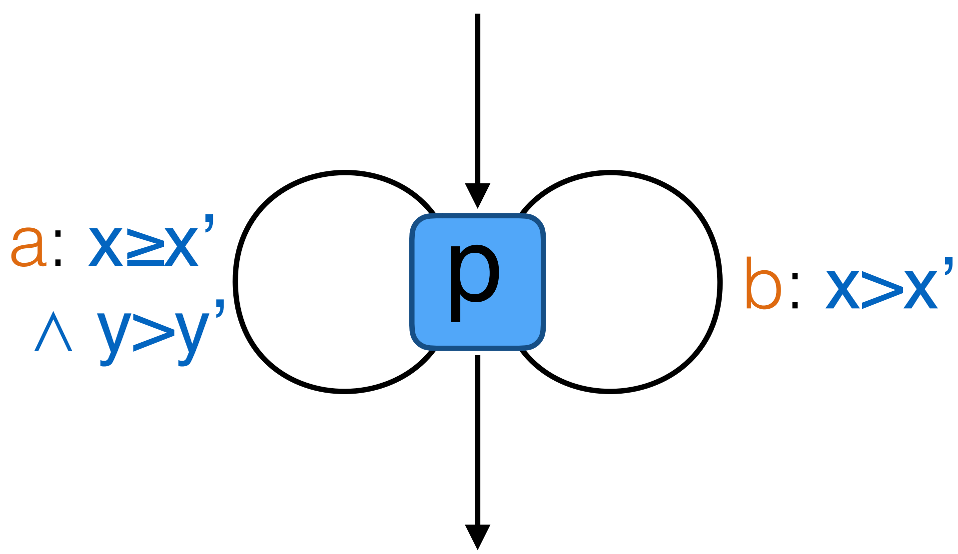
$$(p,2,2) \xrightarrow{a} (p,2,1) \xrightarrow{a} (p,2,0) \xrightarrow{b} (p,1,2) \xrightarrow{a} (p,1,1) \xrightarrow{b} (p,0,2) \xrightarrow{a} (p,0,1) \xrightarrow{a} (p,0,0)$$

A size-change abstraction terminates if it has no infinite run.

# Size-change abstraction

[Ben-Amram et al. 01] A size-change abstraction (SCA):

- this is a non-determinininistic finite state machine
- that uses a finite set variables (x,y,z…) ranging over non-negative integers
- during each transition, a guards relate the variables before and after:
  $x \geq y'$ meaning « val of x before the transition ≥ val of y after the transition »
  $x > y'$ meaning « val of x before the transition > val of y after the transition »




A configuration is a state together with a non-negative integer value for each of the variables.

A run of the SCA is a sequence of configurations that starts in an initial configuration, ends in a final one, and each consecutive configurations satisfy the guard of some possible transition.

$(p,2,2) \xrightarrow{a} (p,2,1) \xrightarrow{a} (p,2,0) \xrightarrow{b} (p,1,2) \xrightarrow{a} (p,1,1) \xrightarrow{b} (p,0,2) \xrightarrow{a} (p,0,1) \xrightarrow{a} (p,0,0)$

A size-change abstraction terminates if it has no infinite run.

[Ben-Aram et al. 01] Termination of size-change abstraction is PSPACE.

# Abstracting

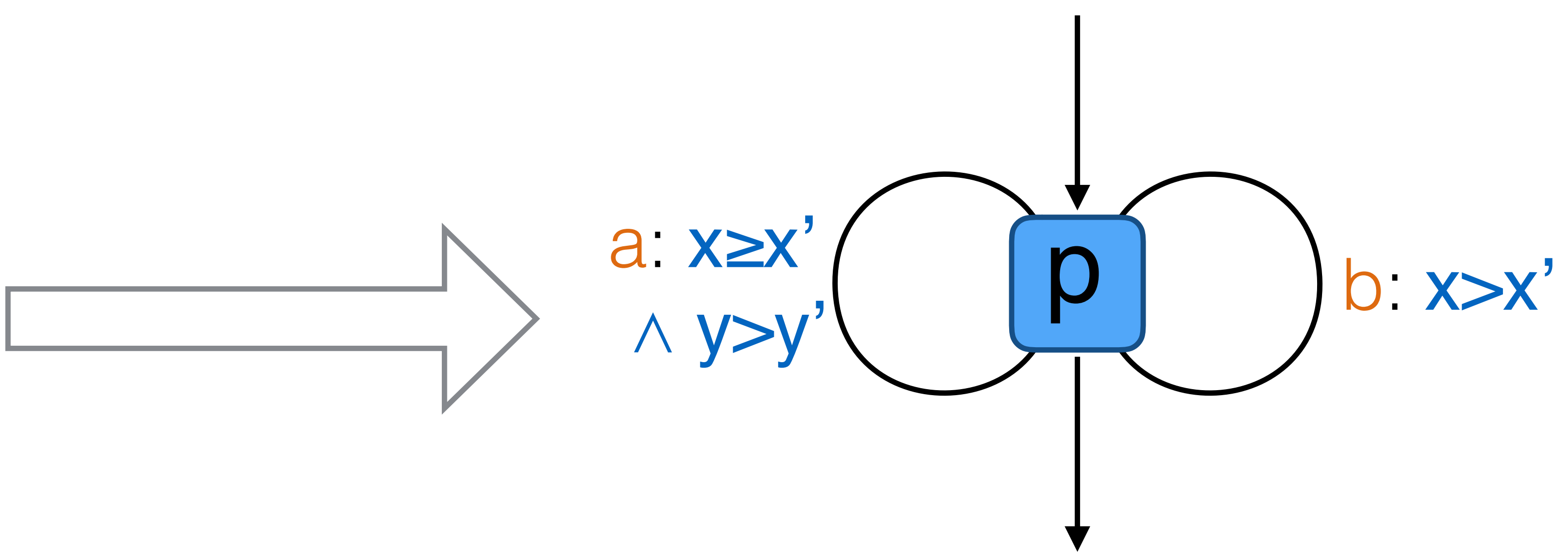
- fix quantities to keep track of, here x,y (can be other quantities)
- construct the control flow graph of the code
- use as guard the best ones you can infer

```
void main() {
 uint x,y;
 x = read_input();
 y = read_input();
 while (x > 0) {
   if (y > 0)
     { y--; }
   else
     { y = read_input();
      x--;  }
  }
}
```

# Abstracting

- fix quantities to keep track of, here x,y (can be other quantities)
- construct the control flow graph of the code
- use as guard the best ones you can infer

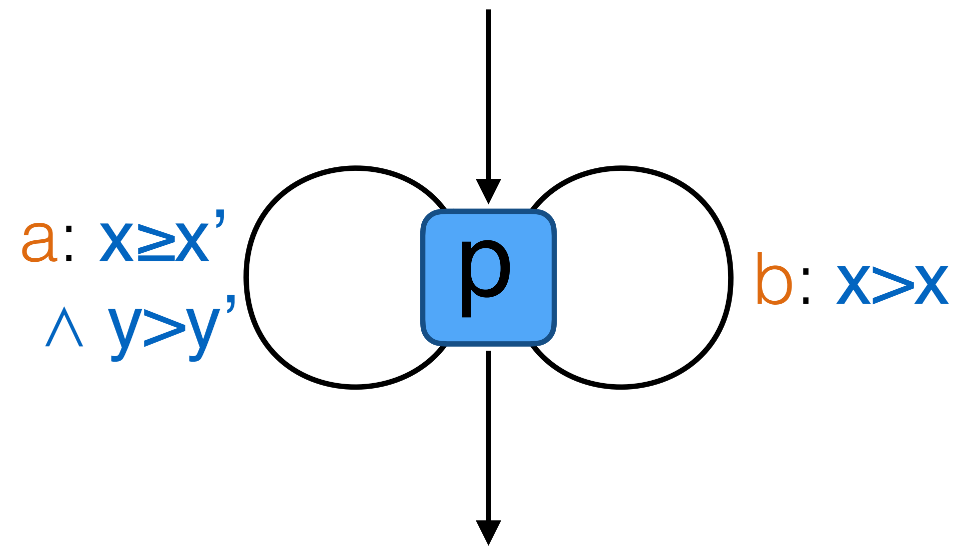
```
void main() {
 uint x,y;
 x = read_input();
 y = read_input();
 while (x > 0) {
   if (y > 0)
     { y--; }
   else
     { y = read_input();
      x--;  }
  }
}
```



# Abstracting

- fix quantities to keep track of, here x,y (can be other quantities)
- construct the control flow graph of the code
- use as guard the best ones you can infer

```
void main() {
 uint x,y;
 x = read_input();
 y = read_input();
 while (x > 0) {
   if (y > 0)
     { y--; }
   else
     { y = read_input();
      x--;  }
  }
}
```




Remark: every run of the original program induces a run of the SCA of game size. Hence if the SCA terminates, the original program also does (on all its executions).

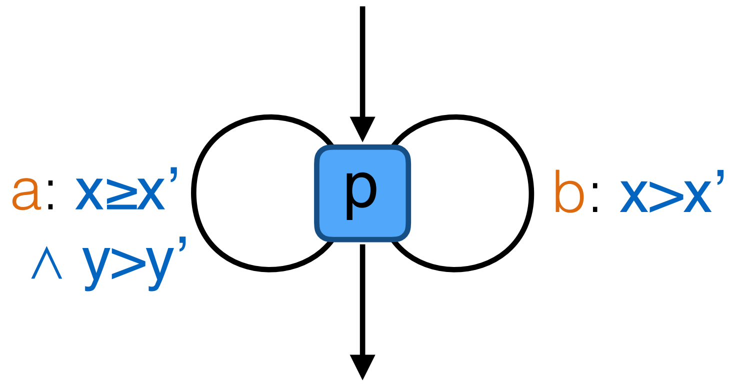
# Deciding the termination of size-change abstraction

# Deciding the termination of size-change abstraction

[Ben-Amram et al. 01]: The termination of SCA is decidable.

# Deciding the termination of size-change abstraction

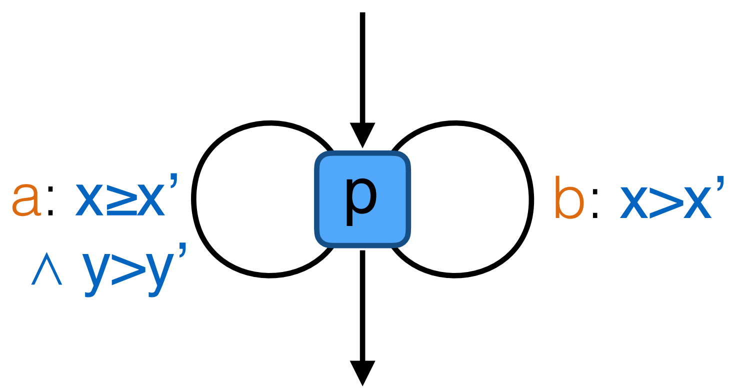
[Ben-Amram et al. 01]: The termination of SCA is decidable.



# Deciding the termination of size-change abstraction

[Ben-Amram et al. 01]: The termination of SCA is decidable.

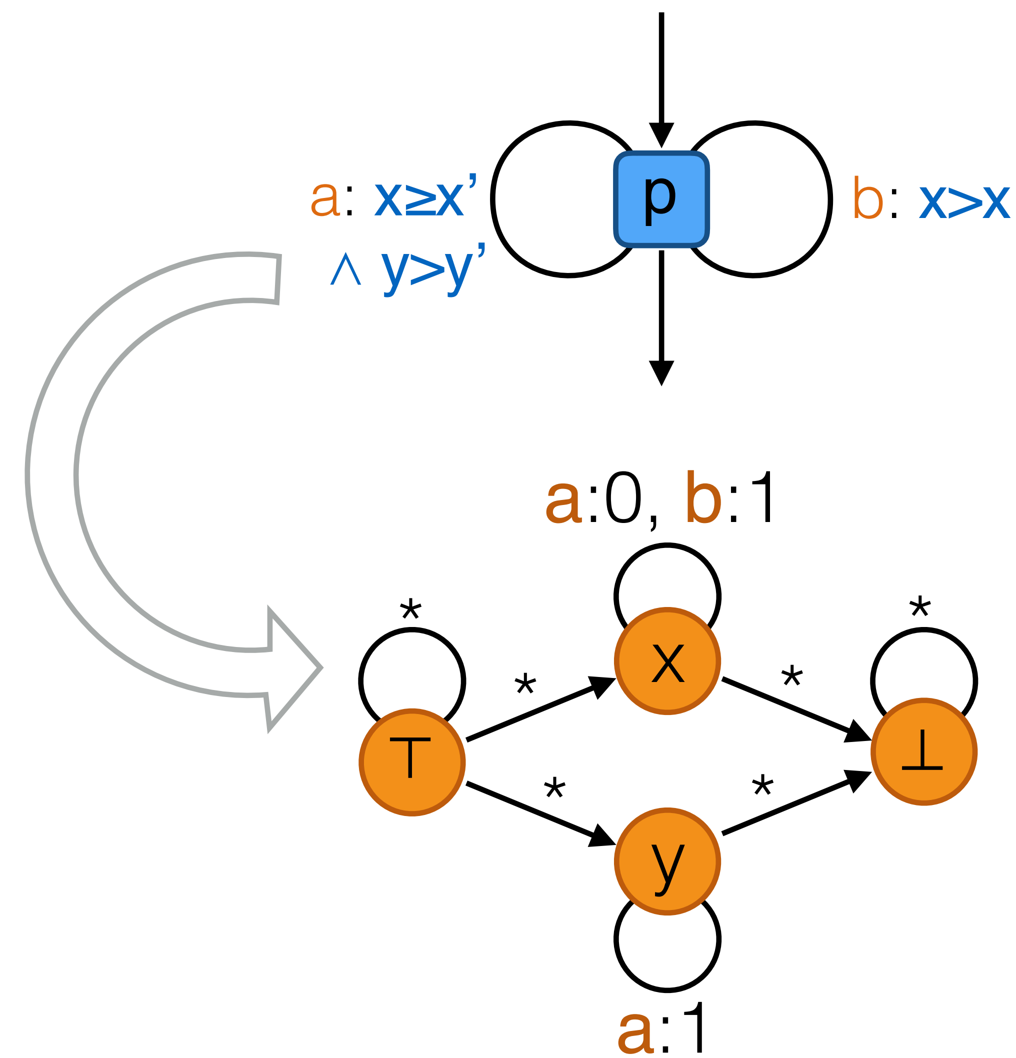
Proof: We construct a Büchi automaton Aut as follows.



# Deciding the termination of size-change abstraction

[Ben-Amram et al. 01]: The termination of SCA is decidable.

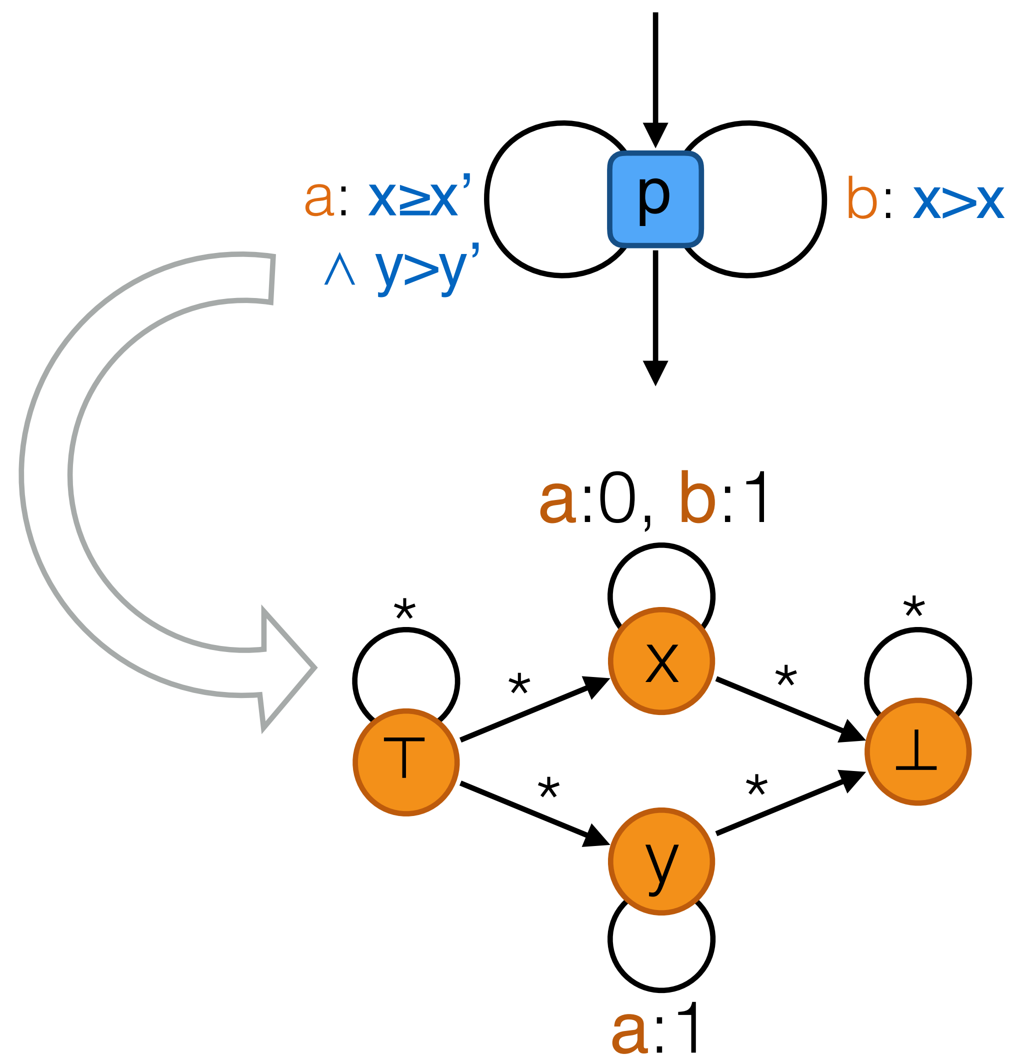
Proof: We construct a Büchi automaton Aut as follows.



# Deciding the termination of size-change abstraction

[Ben-Amram et al. 01]: The termination of SCA is decidable.

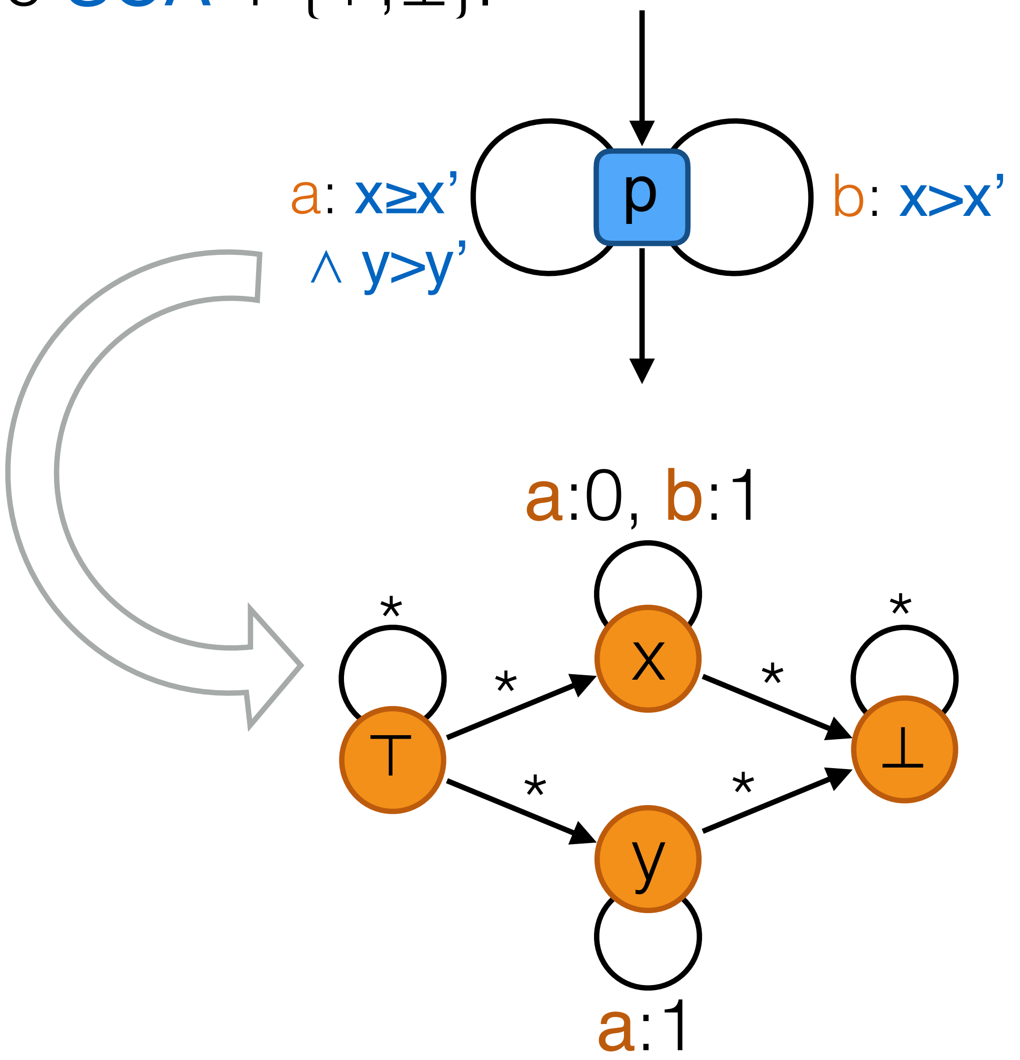
Proof: We construct a Büchi automaton Aut as follows.
Take as alphabet the transitions of the SCA.



# Deciding the termination of size-change abstraction

[Ben-Amram et al. 01]: The termination of SCA is decidable.

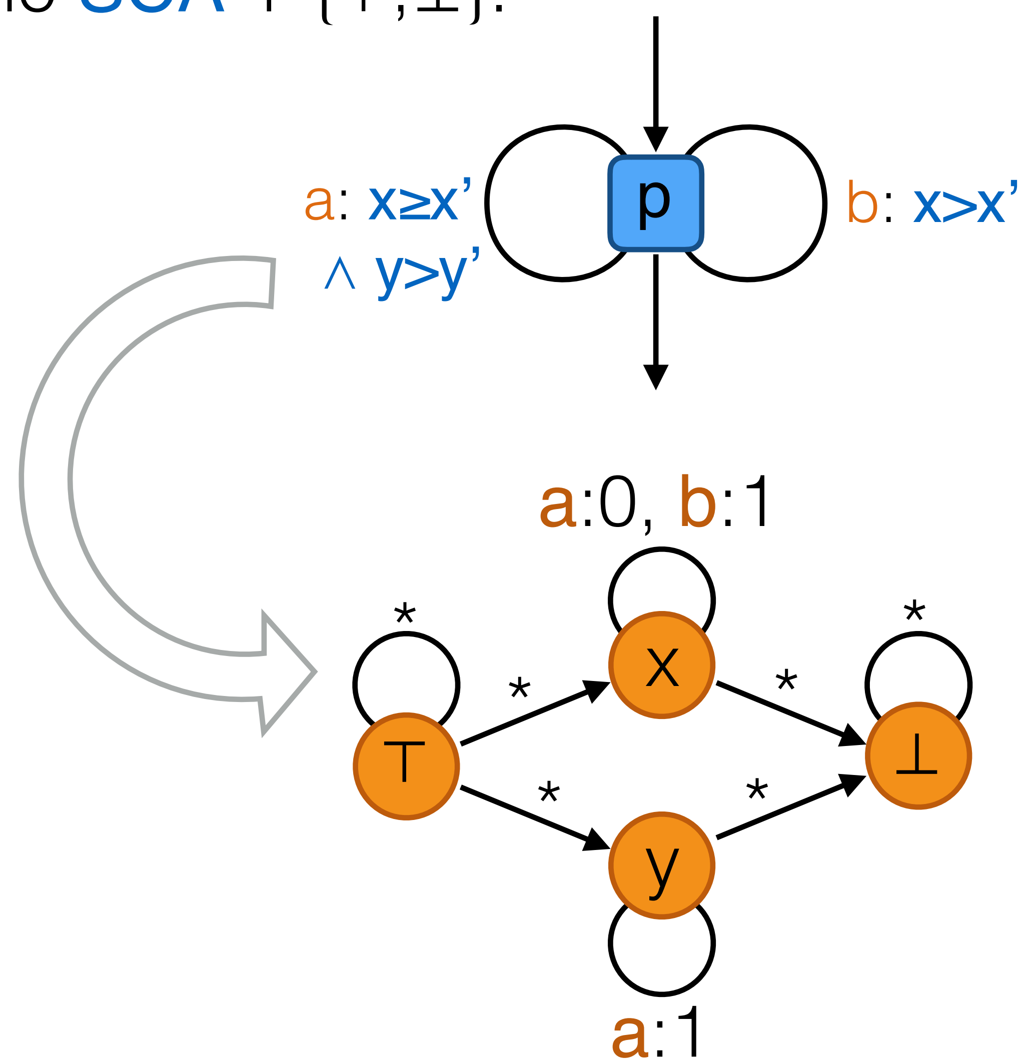
Proof: We construct a Büchi automaton Aut as follows.
Take as alphabet the transitions of the SCA.
Take as states of the automaton, the variables of the SCA + {⊤,⊥}.



# Deciding the termination of size-change abstraction

[Ben-Amram et al. 01]: The termination of SCA is decidable.

Proof: We construct a Büchi automaton Aut as follows.
Take as alphabet the transitions of the SCA.
Take as states of the automaton, the variables of the SCA + {⊤,⊥}.
All states of the automaton are initial.

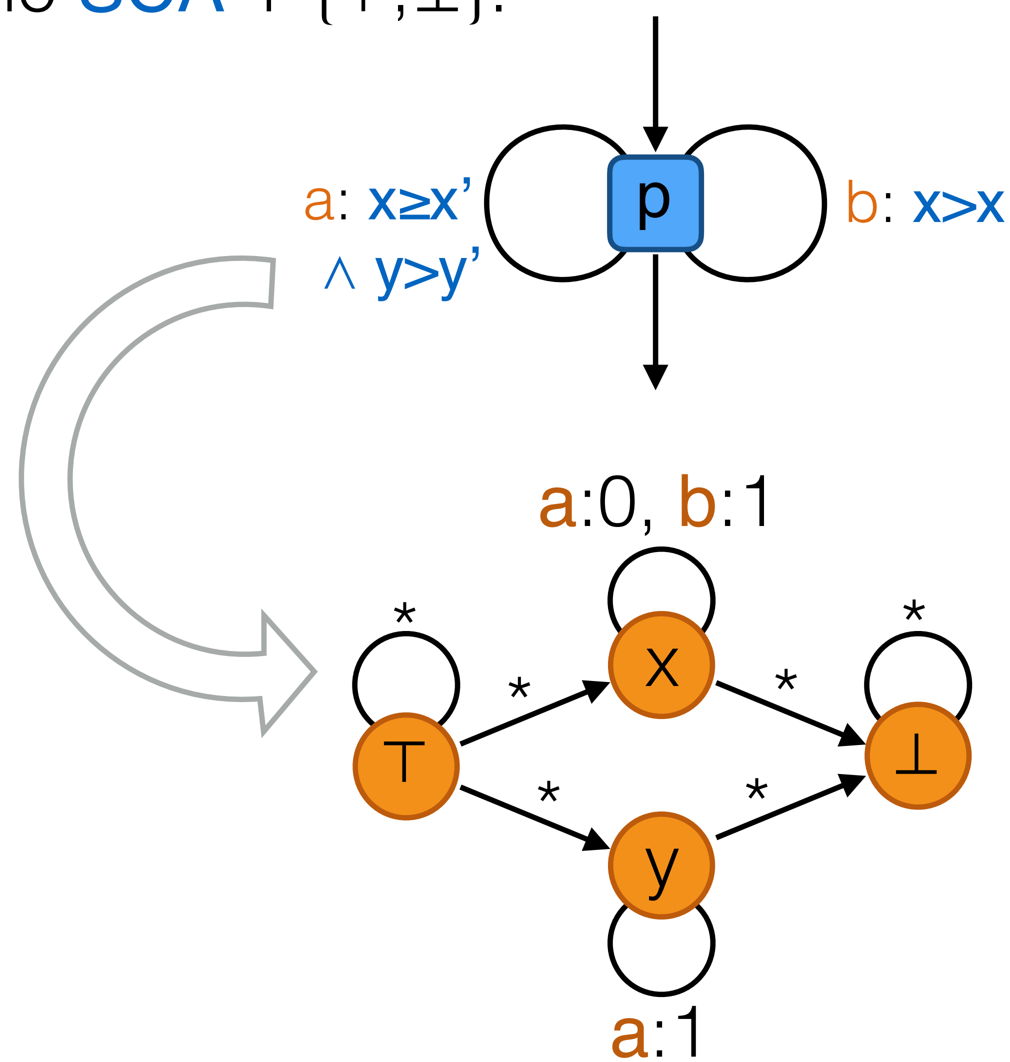


# Deciding the termination of size-change abstraction

[Ben-Amram et al. 01]: The termination of SCA is decidable.

Proof: We construct a Büchi automaton Aut as follows.
Take as alphabet the transitions of the SCA.
Take as states of the automaton, the variables of the SCA + $\{\top,\bot\}$.
All states of the automaton are initial.

$$\Delta(x,a,y) = \begin{cases} 0 & \text{if there is a guard } x \geq y' \text{ in } a \\ 1 & \text{if there is a guard } x > y' \text{ in } a \\ -\infty & \text{otherwise (no guard)} \end{cases}$$



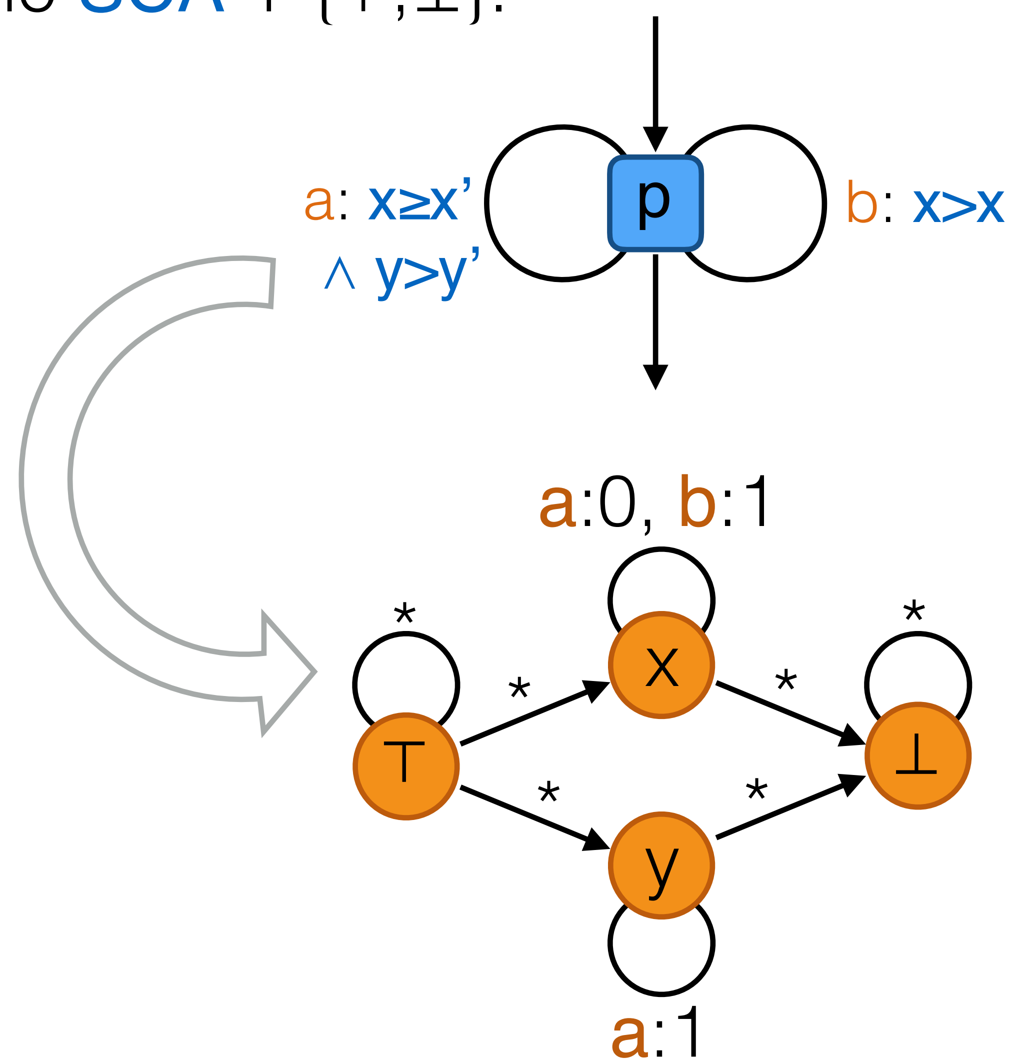
# Deciding the termination of size-change abstraction

[Ben-Amram et al. 01]: The termination of SCA is decidable.

Proof: We construct a Büchi automaton Aut as follows.
Take as alphabet the transitions of the SCA.
Take as states of the automaton, the variables of the SCA + $\{\top,\bot\}$.
All states of the automaton are initial.

$$\Delta(x,a,y) = \begin{cases} 0 & \text{if there is a guard } x \geq y' \text{ in } a \\ 1 & \text{if there is a guard } x > y' \text{ in } a \\ -\infty & \text{otherwise (no guard)} \end{cases}$$

$(\Delta(\bot,?,?)=0,\ \Delta(?,?,\top)=0)$



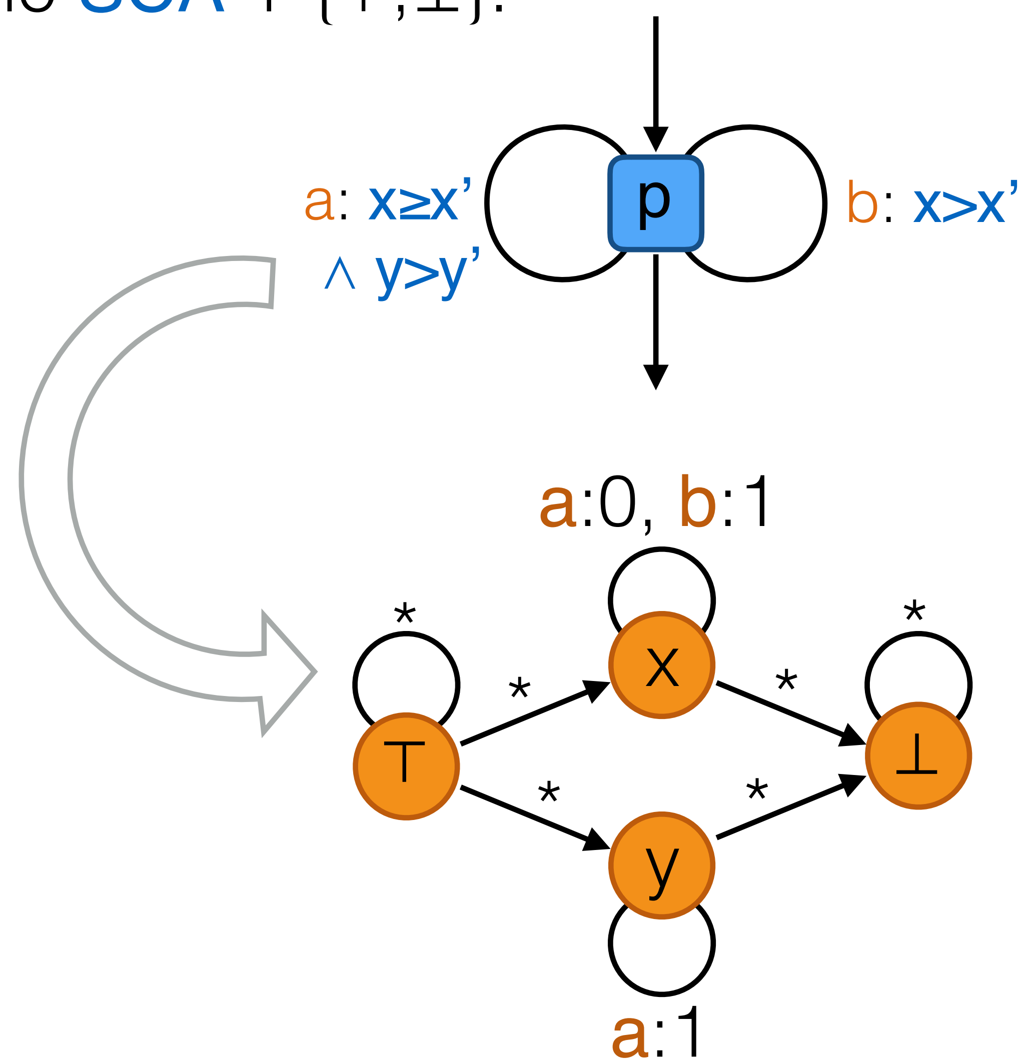
# Deciding the termination of size-change abstraction

[Ben-Amram et al. 01]: The termination of SCA is decidable.

Proof: We construct a Büchi automaton Aut as follows.
Take as alphabet the transitions of the SCA.
Take as states of the automaton, the variables of the SCA + $\{\top,\bot\}$.
All states of the automaton are initial.

$$\Delta(x,a,y) = \begin{cases} 0 & \text{if there is a guard } x \geq y' \text{ in } a \\ 1 & \text{if there is a guard } x > y' \text{ in } a \\ -\infty & \text{otherwise (no guard)} \end{cases}$$

$(\Delta(\bot,?,?)=0,\ \Delta(?,?,\top)=0)$




Claim: $\exists$ run $\rho$ of SCA

⇕

$\exists$ input word u for Aut of same length such that

1) it is a value-free valid run (regular)
2) there is no run of Aut with infinitely many 1's (Büchi condition)

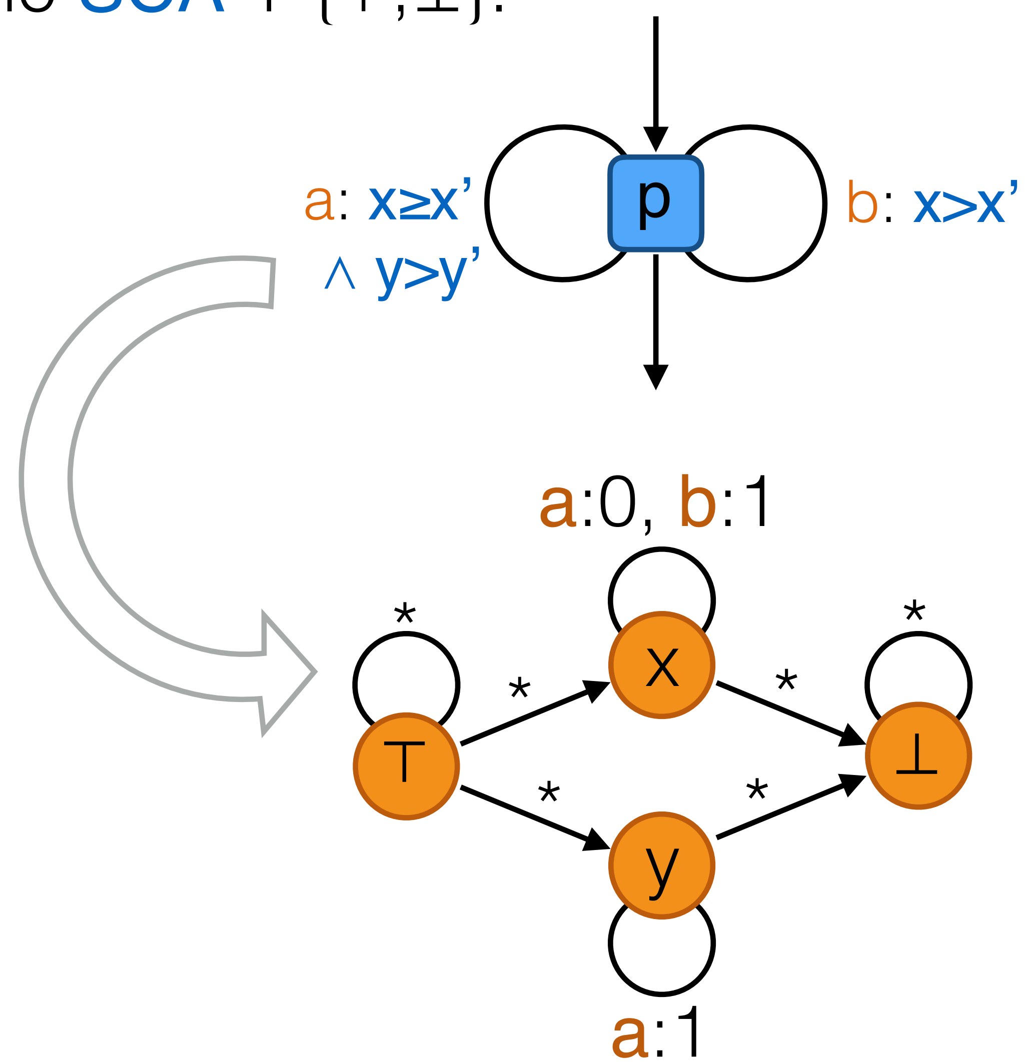
# Deciding the termination of size-change abstraction

[Ben-Amram et al. 01]: The termination of SCA is decidable.

Proof: We construct a Büchi automaton Aut as follows.
Take as alphabet the transitions of the SCA.
Take as states of the automaton, the variables of the SCA + {⊤,⊥}.
All states of the automaton are initial.

$$\Delta(x,a,y) = \begin{cases} 0 & \text{if there is a guard } x \geq y' \text{ in } a \\ 1 & \text{if there is a guard } x > y' \text{ in } a \\ -\infty & \text{otherwise (no guard)} \end{cases}$$

$(\Delta(\perp,?,?)=0,\ \Delta(?,?,\top)=0)$




Claim: ∃ run ρ of SCA

⇕

∃ input word u for Aut of same length such that

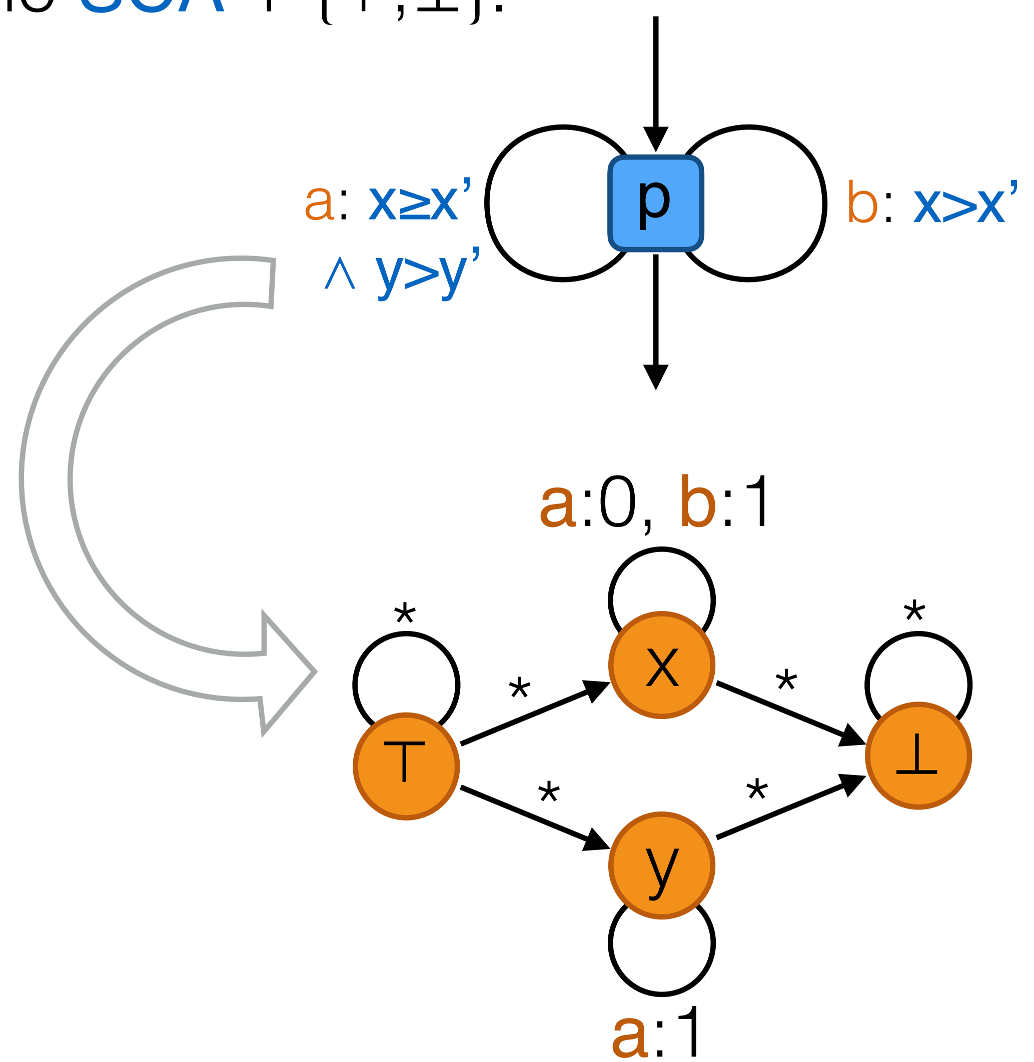
1) it is a value-free valid run (regular)
2) there is no run of Aut with infinitely many 1's (Büchi condition)

⇒ Runs/Aut=∅ ?

# Deciding the termination of size-change abstraction

[Ben-Amram et al. 01]: The termination of SCA is decidable.

Proof: We construct a Büchi automaton Aut as follows.
Take as alphabet the transitions of the SCA.
Take as states of the automaton, the variables of the SCA + {⊤,⊥}.
All states of the automaton are initial.




$$\Delta(x,a,y) = \begin{cases} 0 & \text{if there is a guard } x \geq y' \text{ in } a \\ 1 & \text{if there is a guard } x > y' \text{ in } a \\ -\infty & \text{otherwise (no guard)} \end{cases}$$

$(\Delta(\bot,?,?)=0,\ \Delta(?,?,\top)=0)$

Claim: ∃ run ρ of SCA

⇕

∃ input word u for Aut of same length such that

1) it is a value-free valid run (regular)
2) there is no run of Aut with infinitely many 1's (Büchi condition)

⇒ Runs/Aut=∅ ?

⇒ PSPACE

# Overall picture

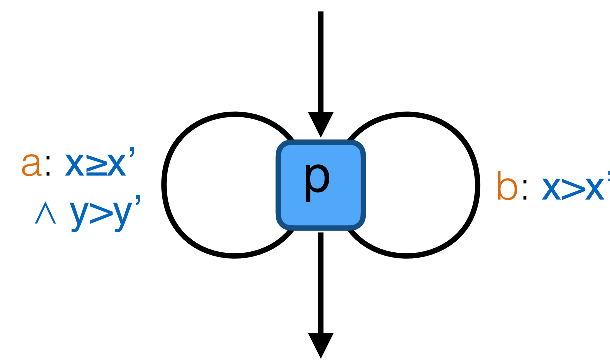
```
void main() {
  uint x,y;
  x = read_input();
  y = read_input();
  while (x > 0) {
    if (y > 0)
      { y--; }
    else
      { y = read_input();
        x--; }
  }
}
```

Some code

?

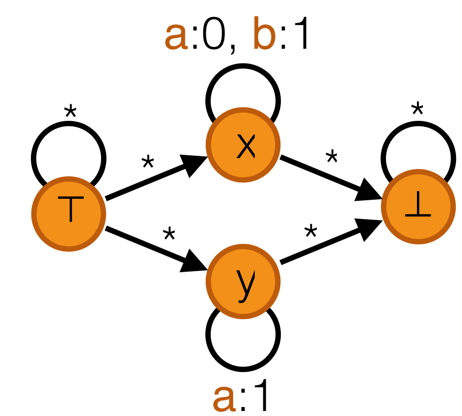
Does it terminate?

reflects termination




size-change abstraction

equivalent for termination




Büchi automaton

Decide an inclusion problem for Büchi automata

# Finer program analysis

# Termiation

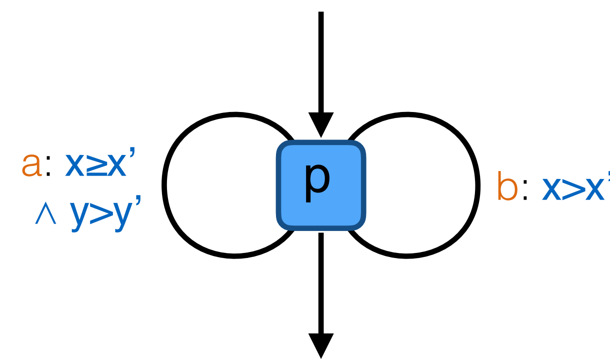
```
void main() {
  uint x,y;
  x = read_input();
  y = read_input();
  while (x > 0) {
    if (y > 0)
      { y--; }
    else
      { y = read_input();
       x--;  }
  }
}
```

Some code

?

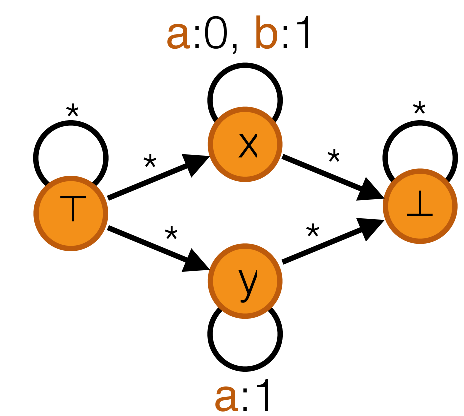
does it terminate?

reflects termination




size-change abstraction

equivalent for termination




Büchi automaton

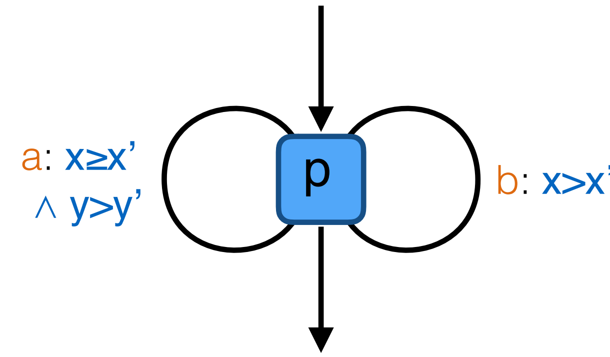
Decide an inclusion problem for Büchi automata

# Asymptotic complexity

```
void main(uint n) {
  uint x,y;
  x = read_input(n);
  y = read_input(n);
  while (x > 0) {
    if (y > 0)
      { y--; }
    else
      { y = read_input(n);
        x--;  }
  }
}
```

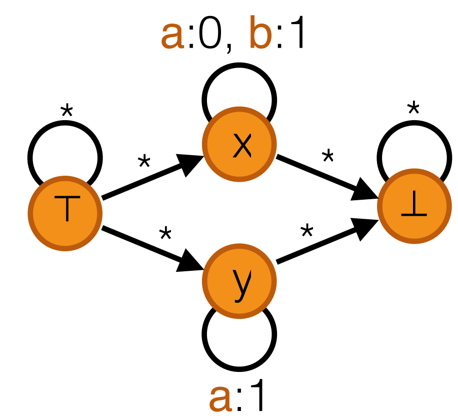
Some code

reflects complexity




size-change abstraction

equivalent for complexity




**N**-max-plus automaton

?

What is its complexity?
(as a function of a parameter **n**)

More precisely, find α such that the program stops in $\Theta(n^{\alpha})$.

Compute the asymptotic worst-case behavior

# Abstracting

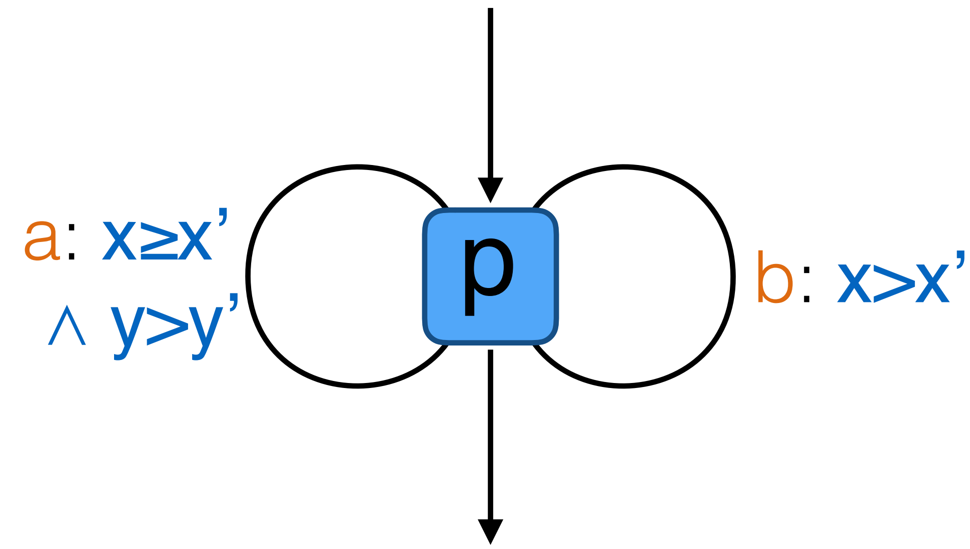
- fix quantities to keep track of, here x,y (can be other quantities)
- construct the control flow graph of the code
- use as guard the best ones you can infer

```
void main(uint n) {
 uint x,y;
 x = read_input(n);
 y = read_input(n);
 while (x > 0) {
   if (y > 0)
     { y--; }
   else
     { y = read_input(n);
      x--;  }
  }
}
```

# Abstracting

- fix quantities to keep track of, here x,y (can be other quantities)
- construct the control flow graph of the code
- use as guard the best ones you can infer

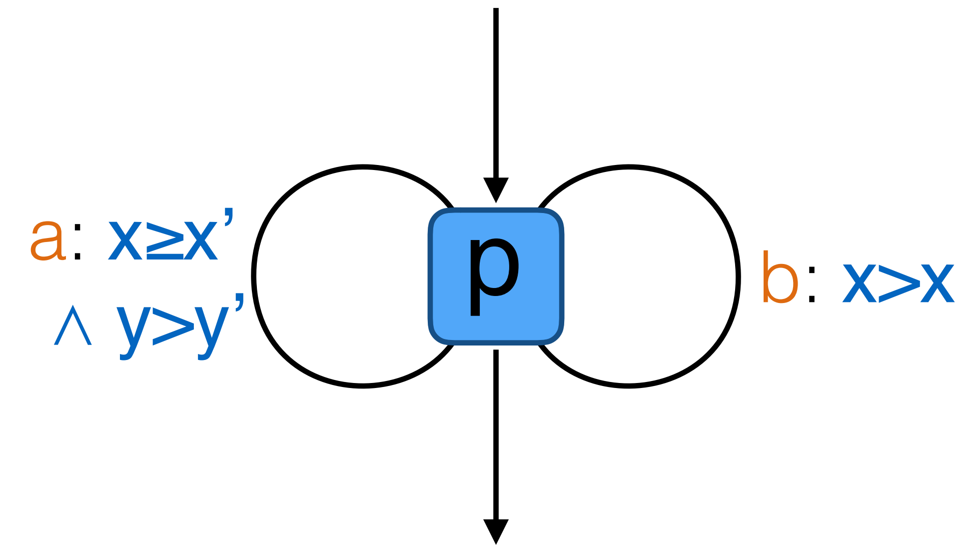
```
void main(uint n) {
 uint x,y;
 x = read_input(n);
 y = read_input(n);
 while (x > 0) {
   if (y > 0)
     { y--; }
   else
     { y = read_input(n);
      x--;  }
  }
}
```



# Abstracting

- fix quantities to keep track of, here x,y (can be other quantities)
- construct the control flow graph of the code
- use as guard the best ones you can infer

```
void main(uint n) {
 uint x,y;
 x = read_input(n);
 y = read_input(n);
 while (x > 0) {
   if (y > 0)
     { y--; }
   else
     { y = read_input(n);
      x--;  }
  }
}
```

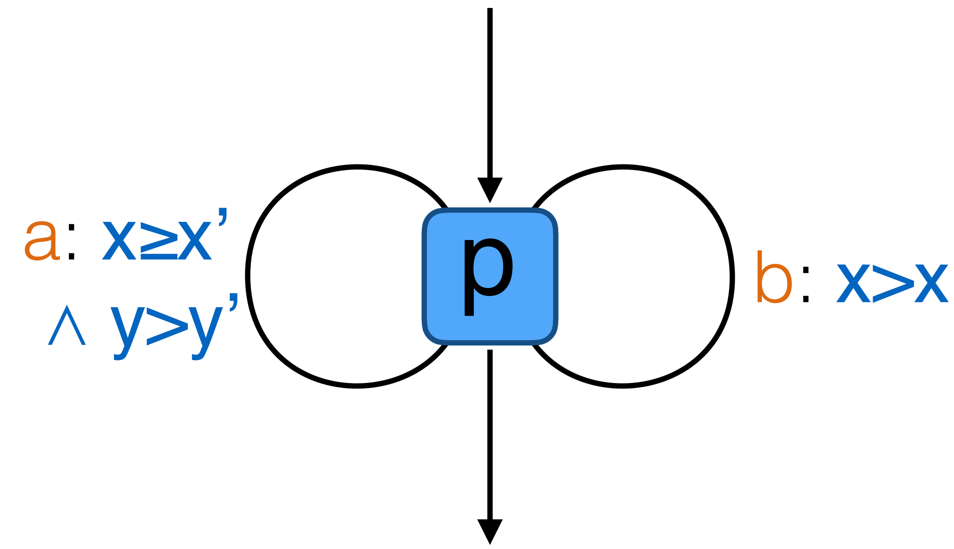



An n-run of the SCA is a run in which all the variables take their values in [1,n]

# Abstracting

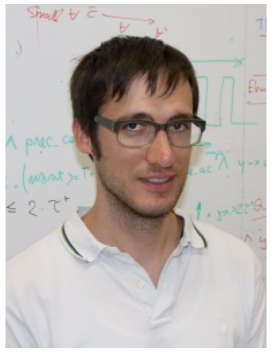
- fix quantities to keep track of, here x,y (can be other quantities)
- construct the control flow graph of the code
- use as guard the best ones you can infer

```
void main(uint n) {
 uint x,y;
 x = read_input(n);
 y = read_input(n);
 while (x > 0) {
   if (y > 0)
     { y--; }
   else
     { y = read_input(n);
      x--;  }
  }
}
```



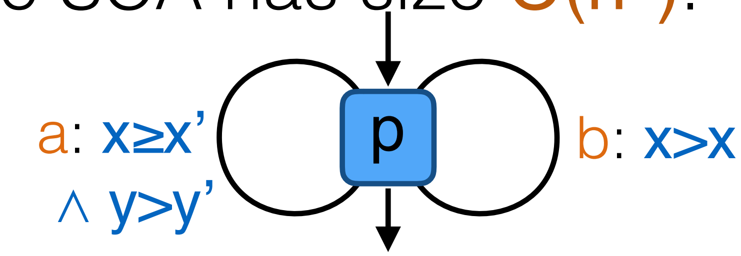

An n-run of the SCA is a run in which all the variables take their values in [1,n]

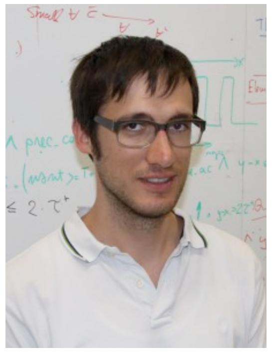
**Remark**: every run of the original program for a given n induces an n-run of the SCA of same length. Hence if the SCA terminates in time **t** for a given n, the original program also does (on all its executions).

# Complexity analysis

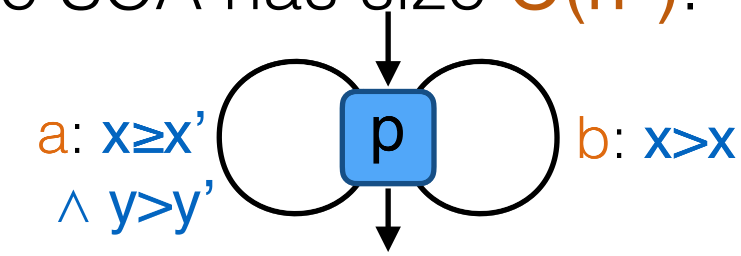
[C., Daviaud, Zuleger 14] If the SCA terminates, there exists a computable rational α such that the worst-case length of an **n-run** of the SCA has size $\Theta(n^\alpha)$.



# Complexity analysis

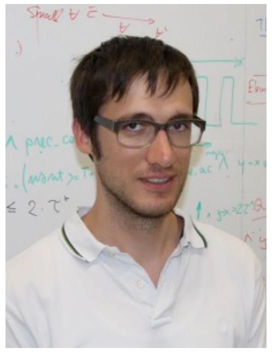
[C., Daviaud, Zuleger 14] If the SCA terminates, there exists a computable rational $\alpha$ such that the worst-case length of an **n-run** of the SCA has size $\Theta(n^{\alpha})$.




**Proof:** We construct a Büchi automaton Aut as follows:
Take as alphabet the transitions of the SCA.
Take as states of the automaton, the variables of the SCA + $\{\top,\bot\}$.
All states of the automaton are initial and final.

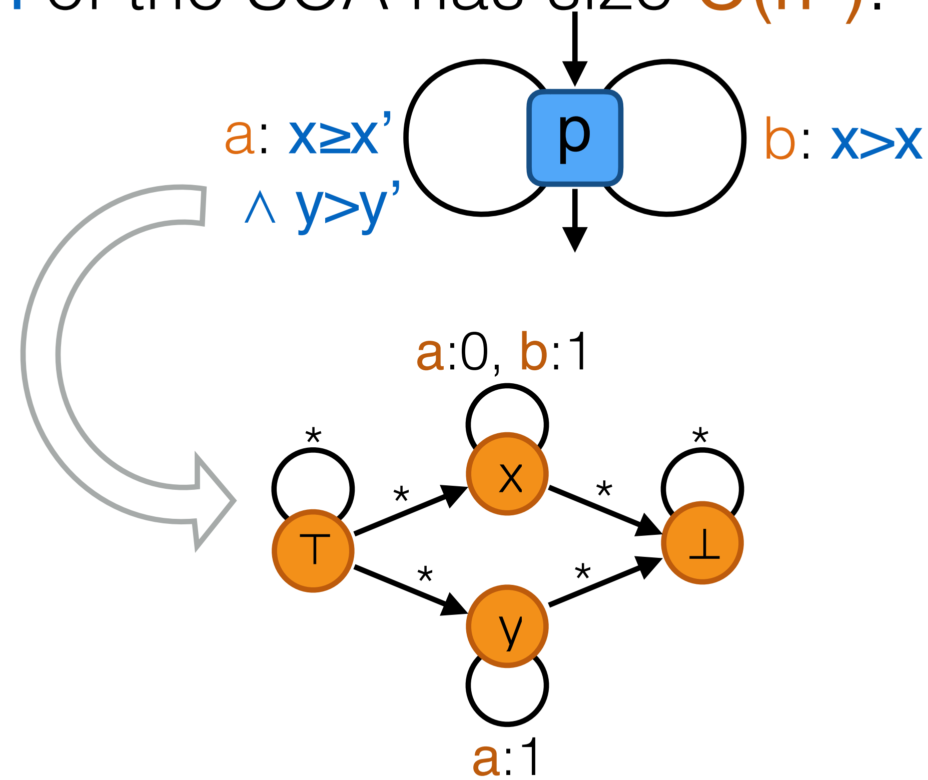
$$\Delta(x,a,y) = \begin{cases} 0 & \text{if there is a guard } x \geq y' \text{ in } a \\ 1 & \text{if there is a guard } x > y' \text{ in } a \\ -\infty & \text{otherwise (no guard)} \end{cases}$$

($\Delta(\bot,?,?)=0$, $\Delta(?,?,\top)=0$)



# Complexity analysis

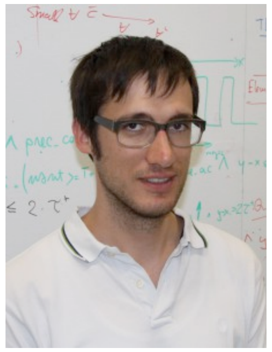
[C., Daviaud, Zuleger 14] If the SCA terminates, there exists a computable rational $\alpha$ such that the worst-case length of an **n-run** of the SCA has size $\Theta(n^{\alpha})$.




**Proof:** We construct a Büchi automaton Aut as follows:
Take as alphabet the transitions of the SCA.
Take as states of the automaton, the variables of the SCA + $\{\top,\bot\}$.
All states of the automaton are initial and final.

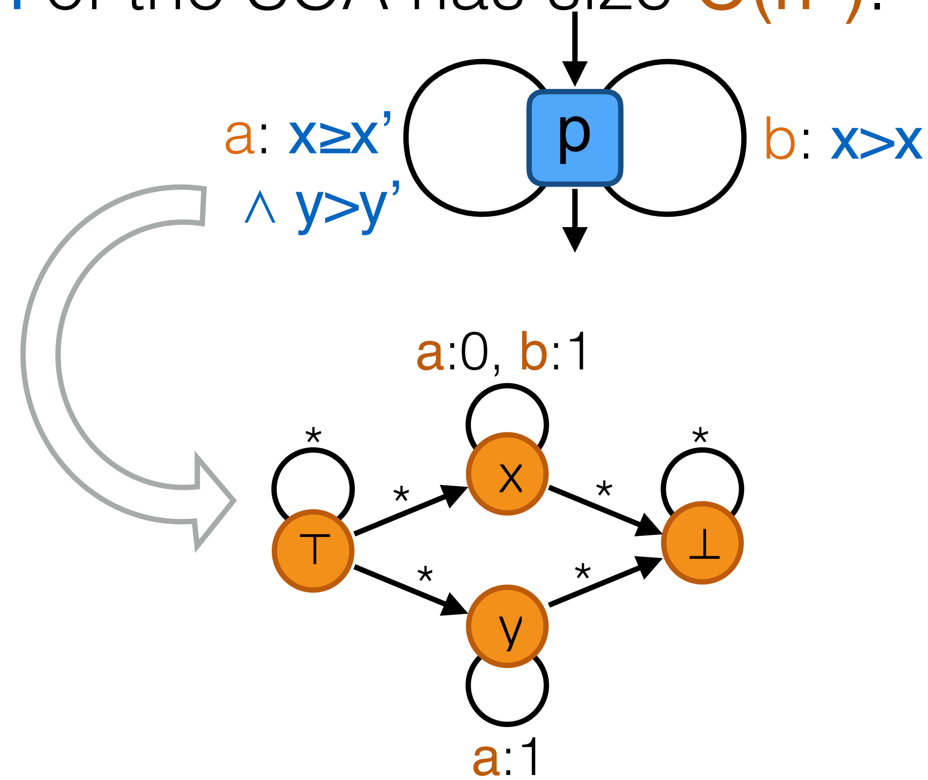
$$\Delta(x,a,y) = \begin{cases} 0 & \text{if there is a guard } x \geq y' \text{ in } a \\ 1 & \text{if there is a guard } x > y' \text{ in } a \\ -\infty & \text{otherwise (no guard)} \end{cases}$$

$(\Delta(\bot,?,?)=0,\ \Delta(?,?,\top)=0)$



# Complexity analysis

[C., Daviaud, Zuleger 14] If the SCA terminates, there exists a computable rational α such that the worst-case length of an **n-run** of the SCA has size $\Theta(n^{\alpha})$.



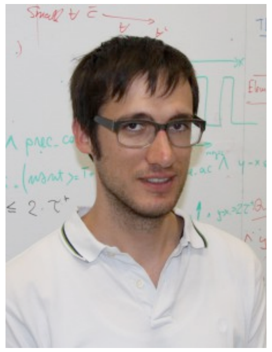

**Proof:** We construct a Büchi automaton Aut as follows:
Take as alphabet the transitions of the SCA.
Take as states of the automaton, the variables of the SCA + {⊤,⊥}.
All states of the automaton are initial and final.

$$\Delta(x,a,y) = \begin{cases} 0 & \text{if there is a guard } x \geq y' \text{ in a} \\ 1 & \text{if there is a guard } x > y' \text{ in a} \\ -\infty & \text{otherwise (no guard)} \end{cases}$$

($\Delta(\perp,?,?)=0$, $\Delta(?,?,\top)=0$)

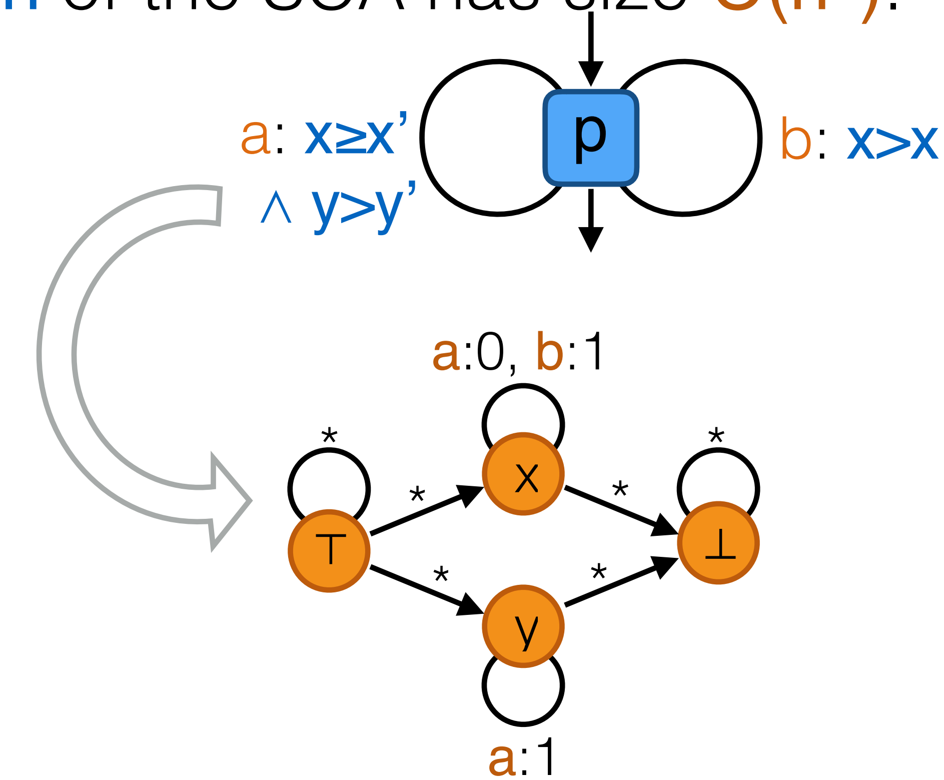
**Claim:** (∃ n-run of SCA of size s)

if and only if ( ∃ input word u of size s such that
1) it is a value-free valid run (regular)
2) there is no run of Aut with weight >n. )

# Complexity analysis

[C., Daviaud, Zuleger 14] If the SCA terminates, there exists a computable rational $\alpha$ such that the worst-case length of an n-run of the SCA has size $\Theta(n^\alpha)$.




**Proof:** We construct a Büchi automaton Aut as follows:

Take as alphabet the transitions of the SCA.

Take as states of the automaton, the variables of the SCA + $\{\top,\bot\}$.

All states of the automaton are initial and final.

$$\Delta(x,a,y) = \begin{cases} 0 & \text{if there is a guard } x \geq y' \text{ in a} \\ 1 & \text{if there is a guard } x > y' \text{ in a} \\ -\infty & \text{otherwise (no guard)} \end{cases}$$

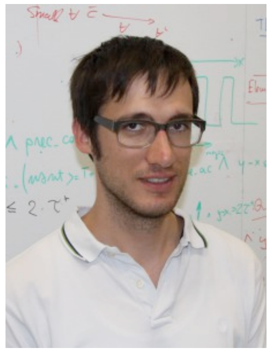
($\Delta(\bot,?,?)=0$, $\Delta(?,?,\top)=0$)

**Claim:** ($\exists$ n-run of SCA of size s)

if and only if ( $\exists$ input word u of size s such that

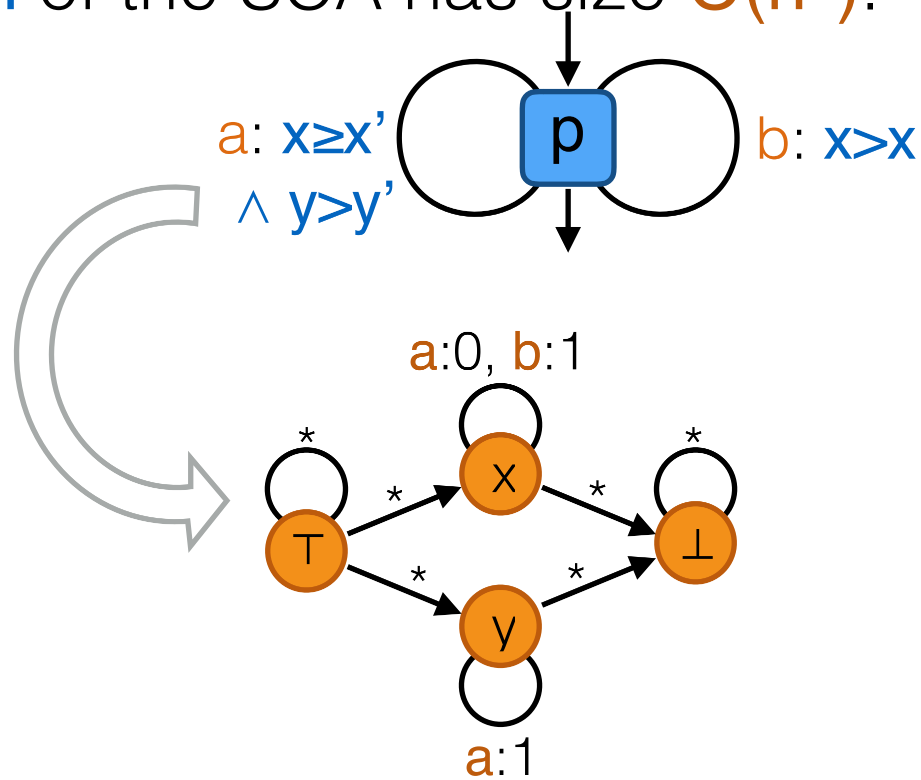
~~1) it is a value-free valid run (regular)~~

2) there is no run of Aut with weight >n. )



# Complexity analysis

[C., Daviaud, Zuleger 14] If the SCA terminates, there exists a computable rational $\alpha$ such that the worst-case length of an n-run of the SCA has size $\Theta(n^{\alpha})$.




**Proof:** We construct a Büchi automaton Aut as follows:

Take as alphabet the transitions of the SCA.
Take as states of the automaton, the variables of the SCA + {⊤,⊥}.
All states of the automaton are initial and final.

$$\Delta(x,a,y) = \begin{cases} 0 & \text{if there is a guard } x \geq y' \text{ in } a \\ 1 & \text{if there is a guard } x > y' \text{ in } a \\ -\infty & \text{otherwise (no guard)} \end{cases}$$

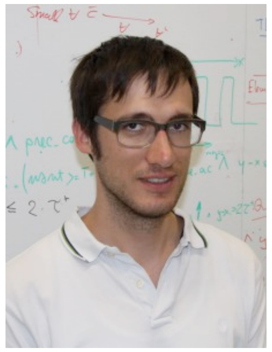
($\Delta(\perp,?,?)=0$, $\Delta(?,?,\top)=0$)

**Claim:** (∃ n-run of SCA of size s)

if and only if ( ∃ input word u of size s such that
~~1) it is a value-free valid run (regular)~~
2) there is no run of Aut with weight >n. )

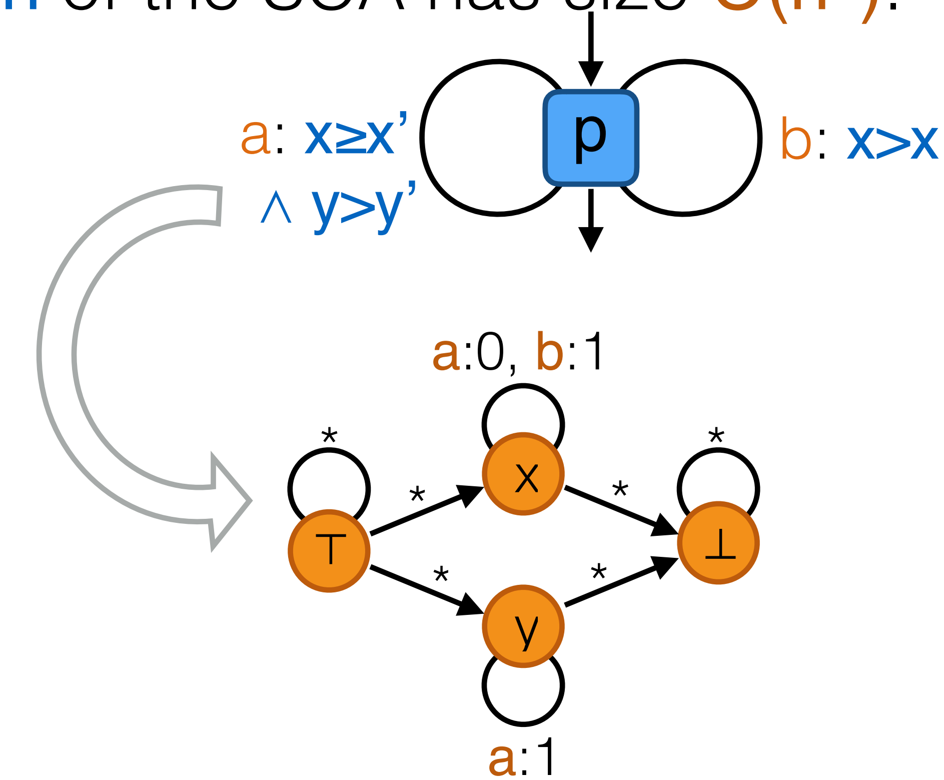
One needs to find the asymptotic exponent of the size of the longest word that is has only run of value at most n:

$$\limsup_{u \in A^*} \frac{\log |u|}{\log \mathsf{Aut}(|u|)} = \alpha$$

# Complexity analysis

[C., Daviaud, Zuleger 14] If the SCA terminates, there exists a computable rational α such that the worst-case length of an n-run of the SCA has size $\Theta(n^\alpha)$.




**Proof:** We construct a Büchi automaton Aut as follows:
Take as alphabet the transitions of the SCA.
Take as states of the automaton, the variables of the SCA + {⊤,⊥}.
All states of the automaton are initial and final.

$$\Delta(x,a,y) = \begin{cases} 0 & \text{if there is a guard } x \geq y' \text{ in } a \\ 1 & \text{if there is a guard } x > y' \text{ in } a \\ -\infty & \text{otherwise (no guard)} \end{cases}$$

$(\Delta(\bot,?,?)=0,\ \Delta(?,?,\top)=0)$

**Claim:** (∃ n-run of SCA of size s)

if and only if ( ∃ input word u of size s such that
~~1) it is a value-free valid run (regular)~~
2) there is no run of Aut with weight >n. )

One needs to find the asymptotic exponent of the size of the longest word that is has only run of value at most n:

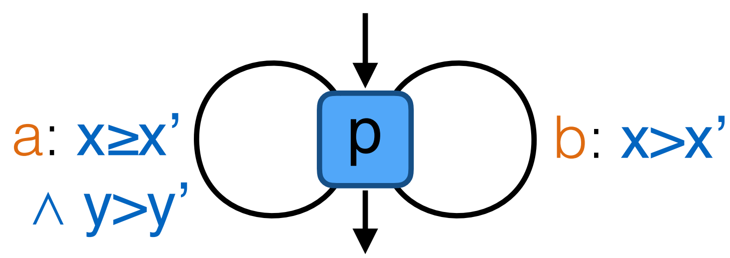
$$\limsup_{u\in A^*} \frac{\log |u|}{\log \mathbf{Aut}(|u|)} = \alpha$$

⇒ Decidable.

# An unexpected phenomenon

# An unexpected phenomenon

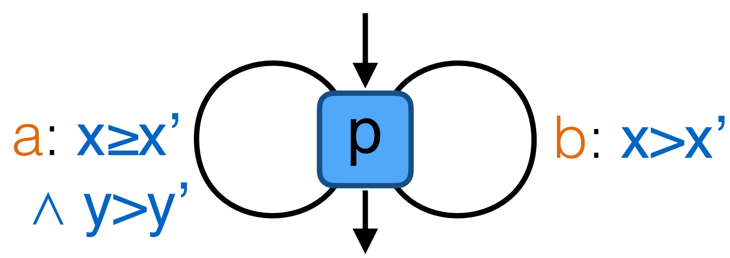
For instance,




has worst-case complexity $n^2$.

# An unexpected phenomenon

For instance,

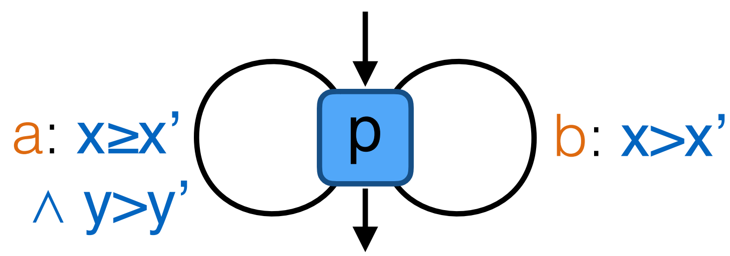



has worst-case complexity $n^2$.

It was conjectured that the asymptotic worst-case could only have integer exponent.

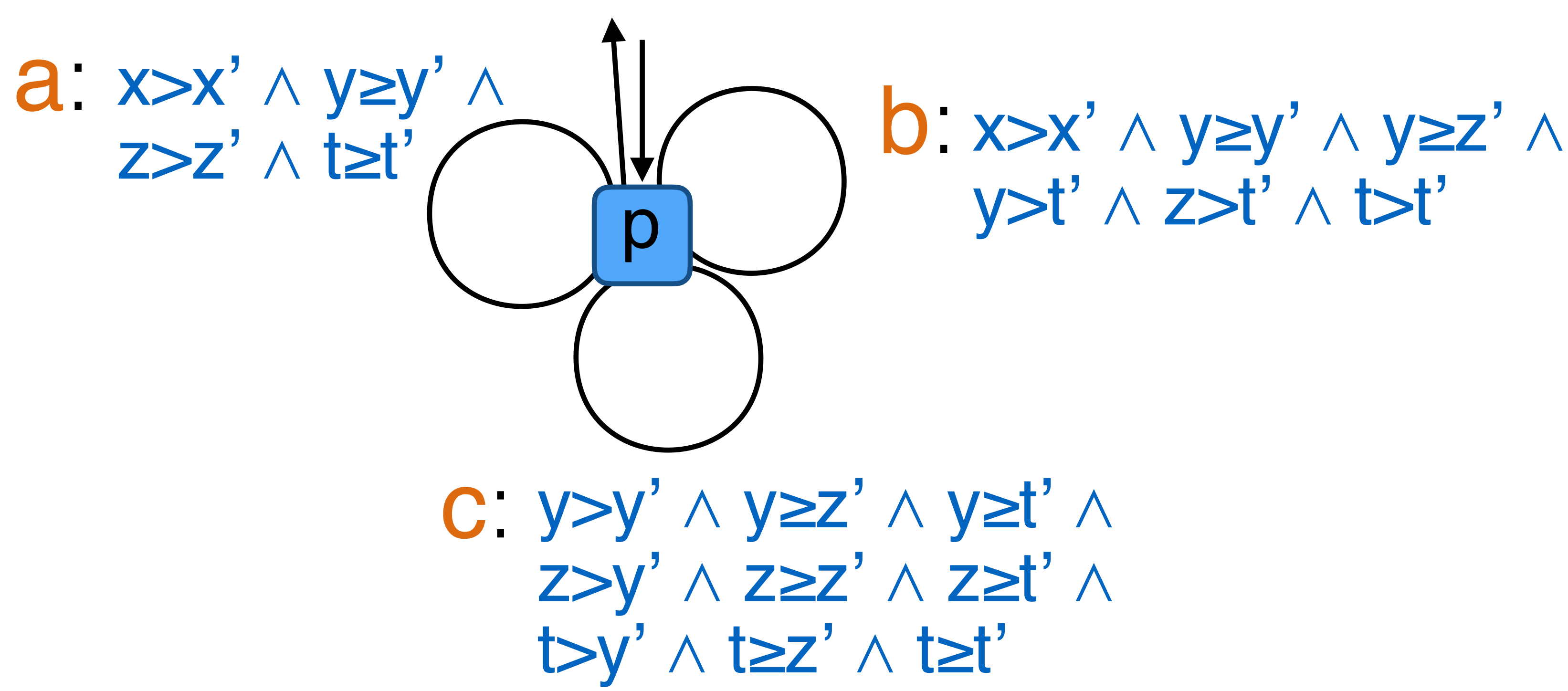
# An unexpected phenomenon

For instance,



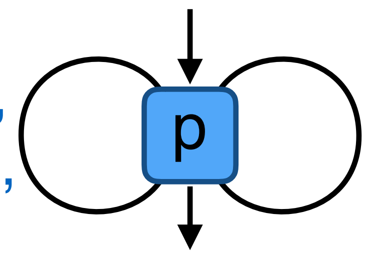

has worst-case complexity $n^2$.

It was conjectured that the asymptotic worst-case could only have integer exponent.



# An unexpected phenomenon

For instance, a: x≥x' ∧ y>y'

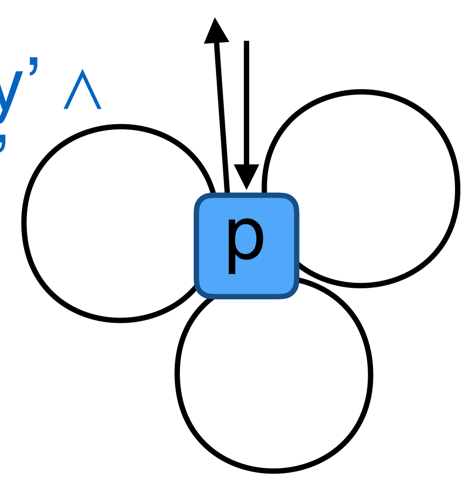


b: x>x' has worst-case complexity $n^2$.

It was conjectured that the asymptotic worst-case could only have integer exponent.

However:
The longest n-run of the following SCA has asymptotical length $\Theta(n^{3/2})$.

a: x>x' ∧ y≥y' ∧ z>z' ∧ t≥t'



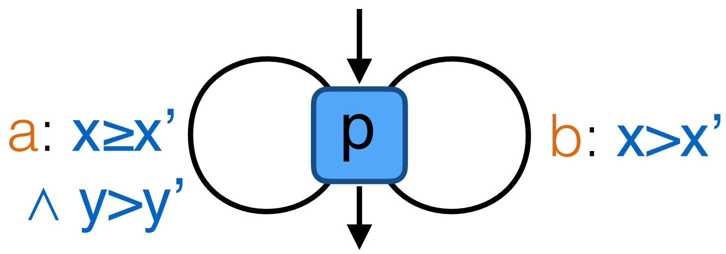
b: x>x' ∧ y≥y' ∧ y≥z' ∧ y>t' ∧ z>t' ∧ t>t'

c: y>y' ∧ y≥z' ∧ y≥t' ∧ z>y' ∧ z≥z' ∧ z≥t' ∧ t>y' ∧ t≥z' ∧ t≥t'

# Summary

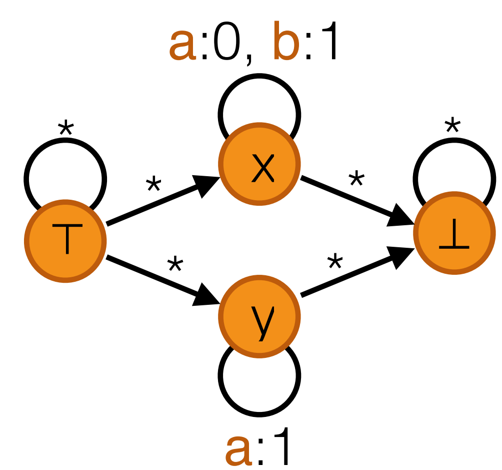
```
void main(uint n) {
  uint x,y;
  x = read_input(n);
  y = read_input(n);
  while (x > 0) {
    if (y > 0)
      { y--; }
    else
      { y = read_input(n);
        x--; }
  }
}
```

The size-change abstraction is good model for proving the termination of some forms of programs. This offers a natural reduction to question of automata theory.




We have shown that this technique can be greatly refined for computing asymptotic worst-case complexity of some programs.

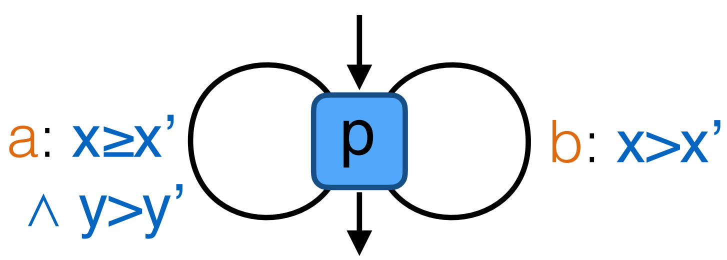
This relies on advanced results on the asymptotic analysis of tropical automata.



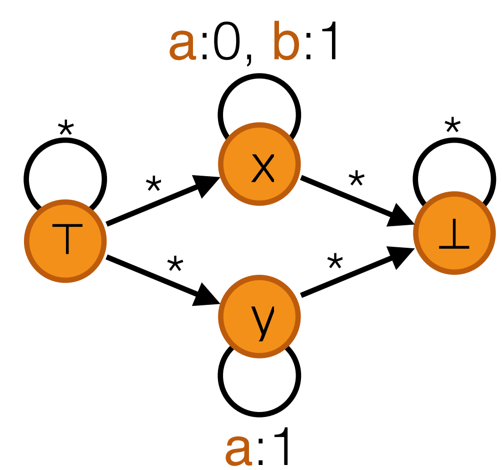
# Summary

```
void main(uint n) {
  uint x,y;
  x = read_input(n);
  y = read_input(n);
  while (x > 0) {
    if (y > 0)
      { y--; }
    else
      { y = read_input(n);
        x--; }
  }
}
```

The size-change abstraction is good model for proving the termination of some forms of programs. This offers a natural reduction to question of automata theory.




We have shown that this technique can be greatly refined for computing asymptotic worst-case complexity of some programs.




This relies on advanced results on the asymptotic analysis of tropical automata.

# Some open questions

What is the exact complexity?
How to construct ranking functions?
Is there a more general model of automata and results?

Thanks !