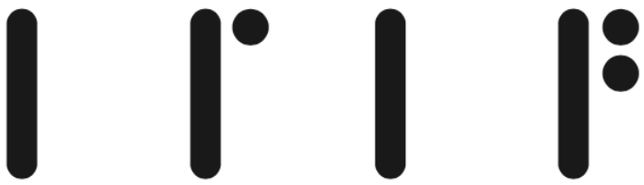
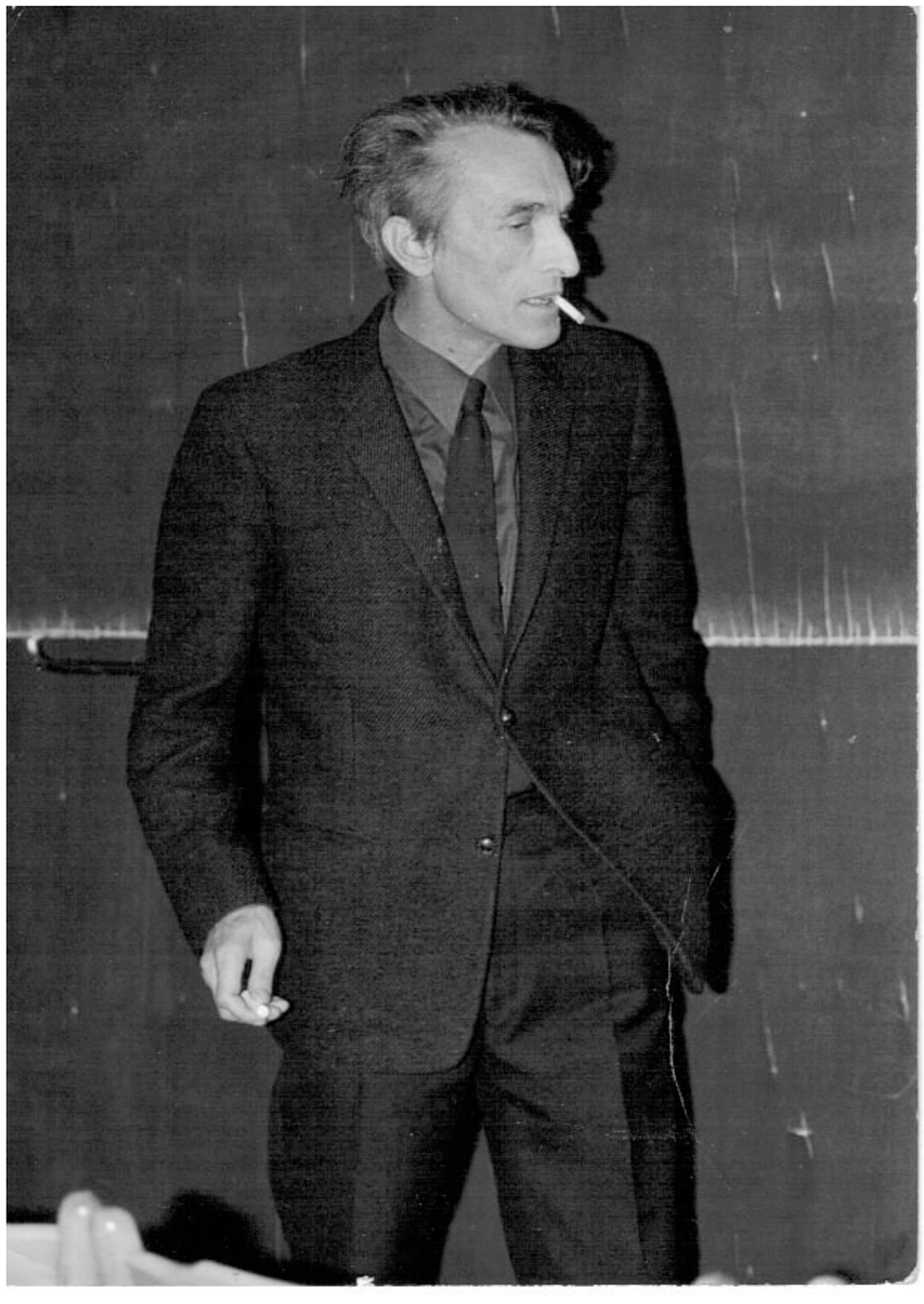
Regular cost functions

Conference dedicated to the scientific legacy
of Marcel-Paul Schützenberger

Thomas Colcombet

23 Mars 2016





The strength of rational / regular languages

The strength of rational / regular languages

Regular languages are effectively equivalently described by

- machines: non-deterministic, alternating automata (…)
- expressions: rational expressions, generalized versions, …
- algebra: recognizability by monoid, deterministic automata, …
- logically: definability in monadic second-order logic (MSO).

The strength of rational / regular languages

Regular languages are effectively equivalently described by

- machines: non-deterministic, alternating automata (…)
- expressions: rational expressions, generalized versions, …
- algebra: recognizability by monoid, deterministic automata, …
- logically: definability in monadic second-order logic (MSO).

Regular languages are closed under **union**, **intersection**, **complement**, **projection**, **concatenation**, etc…

The strength of rational / regular languages

Regular languages are effectively equivalently described by

- machines: non-deterministic, alternating automata (…)
- expressions: rational expressions, generalized versions, …
- algebra: recognizability by monoid, deterministic automata, …
- logically: definability in monadic second-order logic (MSO).

Regular languages are closed under **union**, **intersection**, **complement**, **projection**, **concatenation**, etc…

Regular languages have decidable **emptiness**, **inclusion**, **universality**, **equivalence**.

The strength of rational / regular languages

Regular languages are effectively equivalently described by

- machines: non-deterministic, alternating automata (…)
- expressions: rational expressions, generalized versions, …
- algebra: recognizability by monoid, deterministic automata, …
- logically: definability in monadic second-order logic (MSO).

Regular languages are closed under **union**, **intersection**, **complement**, **projection**, **concatenation**, etc…

Regular languages have decidable **emptiness**, **inclusion**, **universality**, **equivalence**.

Can we replicate these strong properties to a **more general setting** ?

The strength of rational / regular languages

Regular languages are effectively equivalently described by

- machines: non-deterministic, alternating automata (…)
- expressions: rational expressions, generalized versions, …
- algebra: recognizability by monoid, deterministic automata, …
- logically: definability in monadic second-order logic (MSO).

Regular languages are closed under **union**, **intersection**, **complement**, **projection**, **concatenation**, etc…

Regular languages have decidable **emptiness**, **inclusion**, **universality**, **equivalence**.

Can we replicate these strong properties to a **more general setting** ?

Models: Infinite words, countable words, trees, infinite trees, graphs….

The strength of rational / regular languages

Regular languages are effectively equivalently described by

- machines: non-deterministic, alternating automata (…)
- expressions: rational expressions, generalized versions, …
- algebra: recognizability by monoid, deterministic automata, …
- logically: definability in monadic second-order logic (MSO).

Regular languages are closed under **union**, **intersection**, **complement**, **projection**, **concatenation**, etc…

Regular languages have decidable **emptiness**, **inclusion**, **universality**, **equivalence**.

Can we replicate these strong properties to a **more general setting** ?

Models: Infinite words, countable words, trees, infinite trees, graphs….

Range: …

Quantitative acceptors

Quantitative acceptors

[Schützenberger 61] Automata need not be Boolean, these can be parameterized by a computation domain: a semiring.

Quantitative acceptors

[Schützenberger 61] Automata need not be Boolean, these can be parameterized by a computation domain: a semiring.

In a non-deterministic automaton, a word is accepted if

There exists a sequence of transitions such that	all transitions correspond, and the first state is initial and the last state final.

Quantitative acceptors

[Schützenberger 61] Automata need not be Boolean, these can be parameterized by a computation domain: a semiring.

In a non-deterministic automaton, a word is accepted if

There exists a sequence of transitions such that

all transitions correspond, and the first state is initial and the last state final.

$\oplus$

Quantitative acceptors

[Schützenberger 61] Automata need not be Boolean, these can be parameterized by a computation domain: a semiring.

In a non-deterministic automaton, a word is accepted if

There exists a sequence of transitions such that	all transitions correspond, and the first state is initial and the last state final.
$\oplus$	$\otimes$

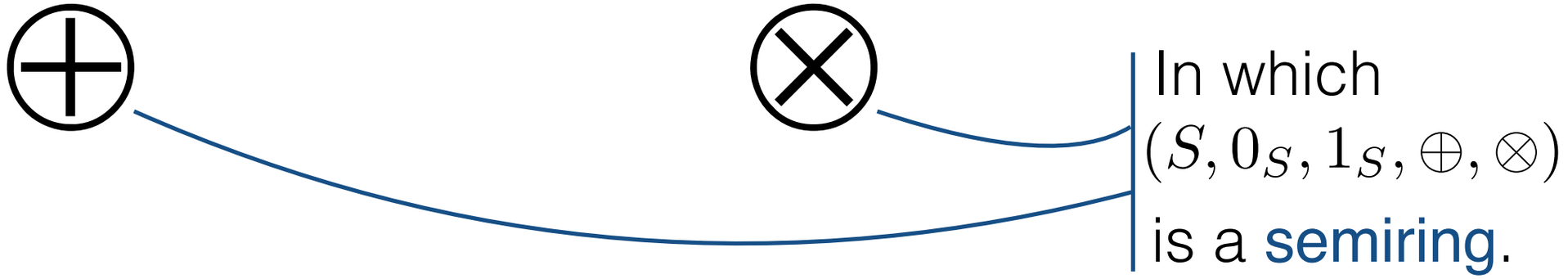
Quantitative acceptors

[Schützenberger 61] Automata need not be Boolean, these can be parameterized by a computation domain: a semiring.

In a non-deterministic automaton, a word is accepted if

There exists a sequence of transitions such that

all transitions correspond, and the first state is initial and the last state final.



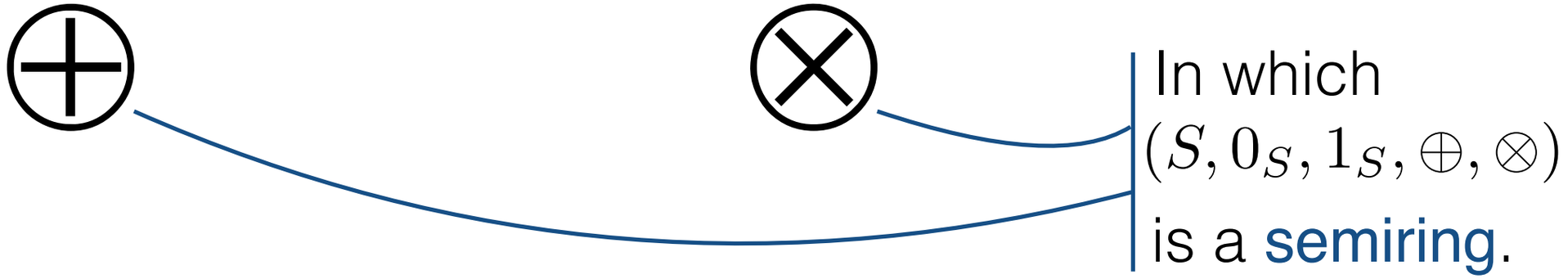
Quantitative acceptors

[Schützenberger 61] Automata need not be Boolean, these can be parameterized by a computation domain: a semiring.

In a non-deterministic automaton, a word is accepted if

There exists a sequence of transitions such that

all transitions correspond, and the first state is initial and the last state final.



If $(\{0,1\}, 0, 1, \vee, \wedge)$ one recovers non-deterministic automata.

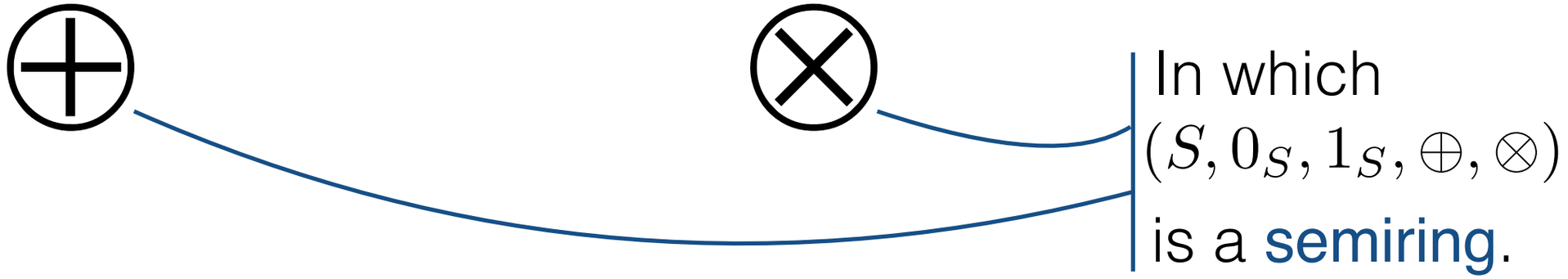
Quantitative acceptors

[Schützenberger 61] Automata need not be Boolean, these can be parameterized by a computation domain: a semiring.

In a non-deterministic automaton, a word is accepted if

There exists a sequence of transitions such that

all transitions correspond, and the first state is initial and the last state final.



If $(\{0,1\}, 0, 1, \vee, \wedge)$ one recovers non-deterministic automata.

Fields yield minimizable automata [Schützenberger61].

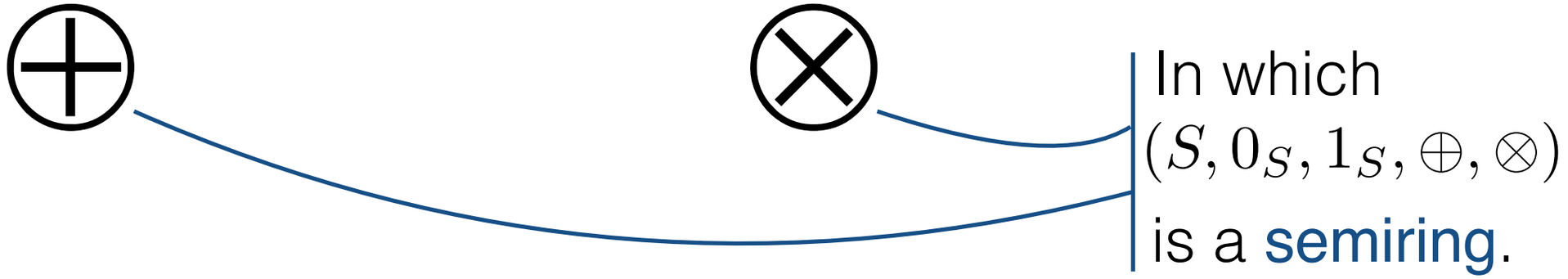
Quantitative acceptors

[Schützenberger 61] Automata need not be Boolean, these can be parameterized by a computation domain: a semiring.

In a non-deterministic automaton, a word is accepted if

There exists a sequence of transitions such that

all transitions correspond, and the first state is initial and the last state final.



If $(\{0,1\}, 0, 1, \vee, \wedge)$ one recovers non-deterministic automata.

Fields yield minimizable automata [Schützenberger61].

Probabilistic automata are a particular case.

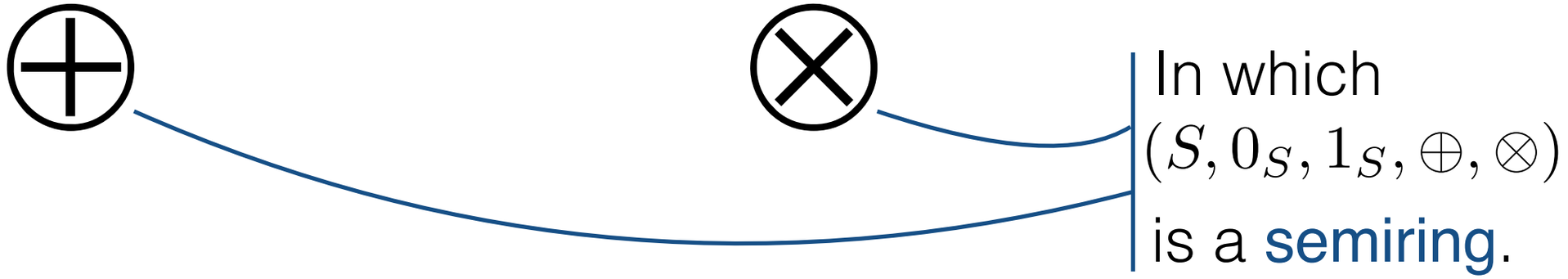
Quantitative acceptors

[Schützenberger 61] Automata need not be Boolean, these can be parameterized by a computation domain: a semiring.

In a non-deterministic automaton, a word is accepted if

There exists a sequence of transitions such that

all transitions correspond, and the first state is initial and the last state final.



If $(\{0,1\}, 0, 1, \vee, \wedge)$ one recovers non-deterministic automata.

Fields yield minimizable automata [Schützenberger61].

Probabilistic automata are a particular case.

Tropical automata: $(\mathbb{R} \cup \{-\infty\}, 0, -\infty, \max, +)$ $(\mathbb{R} \cup \{\infty\}, 0, \infty, \min, +)$

Max,+ automata Min,+ automata

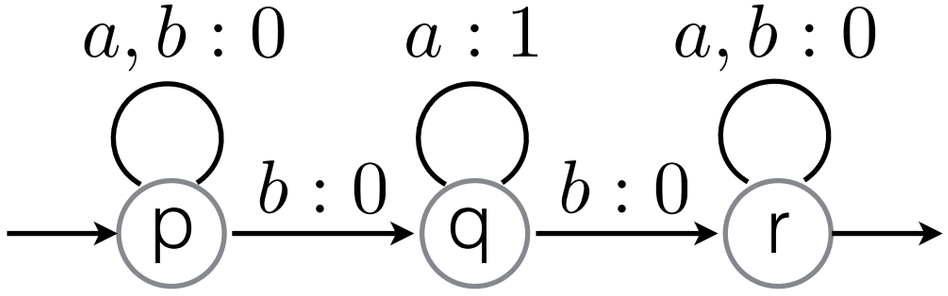
Distance automata

Distance automata

correspond to the semiring $(\mathbb{N} \cup \{\infty\}, 0, \infty, \min, +)$.

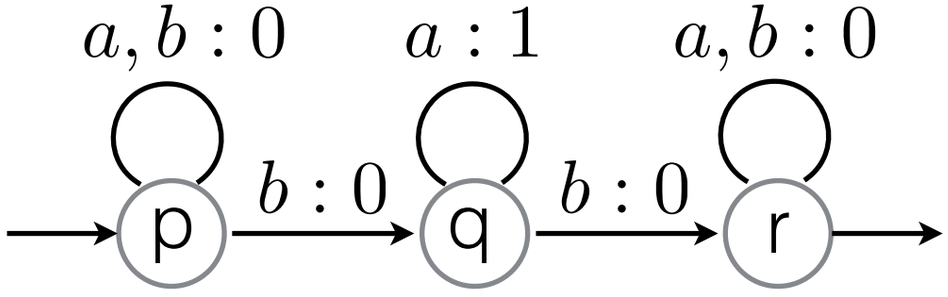
Distance automata

correspond to the semiring $(\mathbb{N} \cup \{\infty\}, 0, \infty, \min, +)$.



Distance automata

correspond to the semiring $(\mathbb{N} \cup \{\infty\}, 0, \infty, \min, +)$.



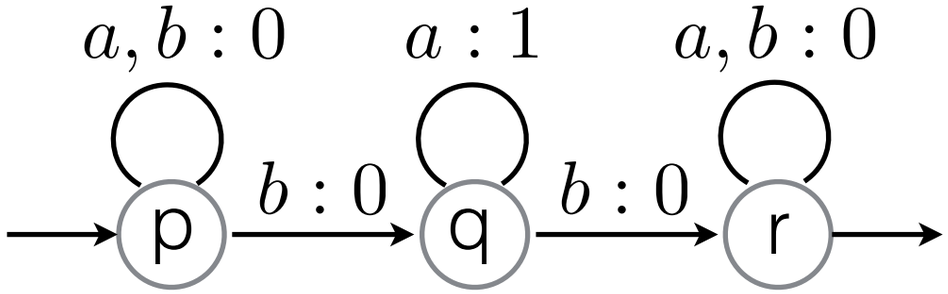
A distance automaton computes:

$$[\![\mathcal{A}]\!] : A^* \to \mathbb{N} \cup \{\infty\}$$

$u \mapsto$ the infimum over all accepting runs of the sum of weights.

Distance automata

correspond to the semiring $(\mathbb{N} \cup \{\infty\}, 0, \infty, \min, +)$.



minblock = The minimum length of a block of consecutive a-letters surrounded by b's.

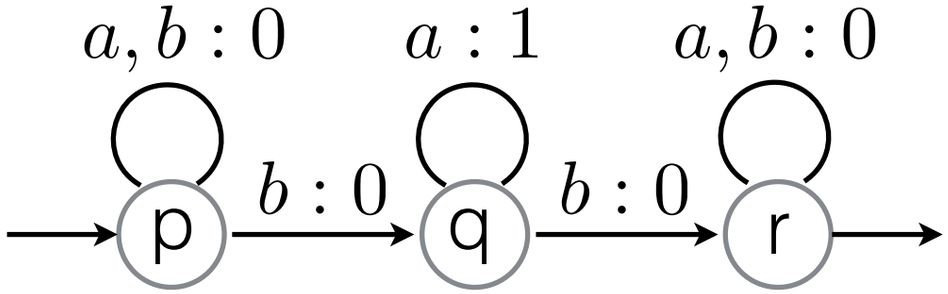
A distance automaton computes:

$$[\![\mathcal{A}]\!] : A^* \to \mathbb{N} \cup \{\infty\}$$

$u \mapsto$ the infimum over all accepting runs of the sum of weights.

Distance automata

correspond to the semiring $(\mathbb{N} \cup \{\infty\}, 0, \infty, \min, +)$.



minblock = The minimum length of a block of consecutive a-letters surrounded by b's.

A distance automaton computes:

$$[\![\mathcal{A}]\!] : A^* \to \mathbb{N} \cup \{\infty\}$$

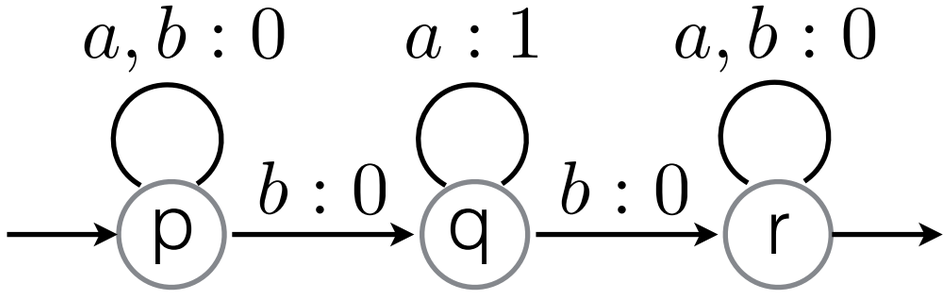
$u \mapsto$ the infimum over all accepting runs of the sum of weights.

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

(for solving the star-height problem)

Distance automata

correspond to the semiring $(\mathbb{N} \cup \{\infty\}, 0, \infty, \min, +)$.



minblock = The minimum length of a block of consecutive a-letters surrounded by b's.

A distance automaton computes:

$$[\![\mathcal{A}]\!] : A^* \to \mathbb{N} \cup \{\infty\}$$

$u \mapsto$ the infimum over all accepting runs of the sum of weights.

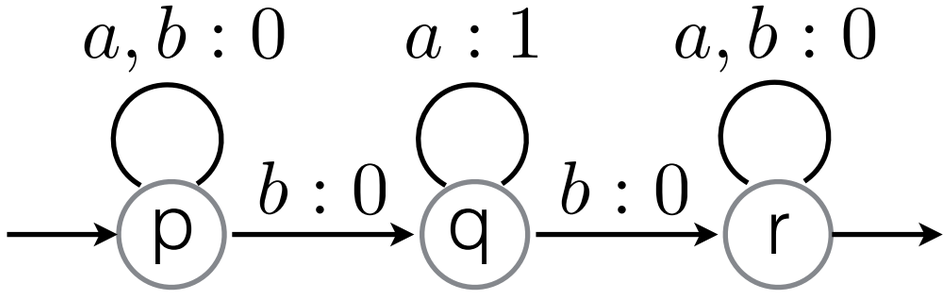
[Hashiguchi 81] The boundedness of distance automata is decidable.

(for solving the star-height problem)

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

Distance automata

correspond to the semiring $(\mathbb{N} \cup \{\infty\}, 0, \infty, \min, +)$.



minblock = The minimum length of a block of consecutive a-letters surrounded by b's.

A distance automaton computes:

$$[\![\mathcal{A}]\!] : A^* \to \mathbb{N} \cup \{\infty\}$$

$u \mapsto$ the infimum over all accepting runs of the sum of weights.

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

(for solving the star-height problem)

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

[Krob 94] The equivalence of distance automata is undecidable.

Cost functions

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

[Krob 94] The equivalence of distance automata is undecidable.

Cost functions

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

[Krob 94] The equivalence of distance automata is undecidable.

How can we hope a theory similar to regular languages ?

Cost functions

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

[Krob 94] The equivalence of distance automata is undecidable.

How can we hope a theory similar to regular languages ?

An answer: let us forget the exact values of functions, but keep sufficient information for boundedness.

Cost functions

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

[Krob 94] The equivalence of distance automata is undecidable.

How can we hope a theory similar to regular languages ?

An answer: let us forget the exact values of functions, but keep sufficient information for boundedness.

A cost function is an equivalence class for the relation $\approx$:

$f \preccurlyeq g$ (f is dominated by g) if

for all sets X (f|X is bounded implies g|X is bounded)

Boundedness equivalence

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A*
$g|_X$ is bounded implies $f|_X$ is bounded

f≈g if f≼g and g≼f

cost function = ≈-class

Boundedness equivalence

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|_X$ is bounded implies $f|_X$ is bounded

f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.

Boundedness equivalence

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|x$ is bounded implies $f|x$ is bounded

f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.

Boundedness equivalence

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|_X$ is bounded implies $f|_X$ is bounded

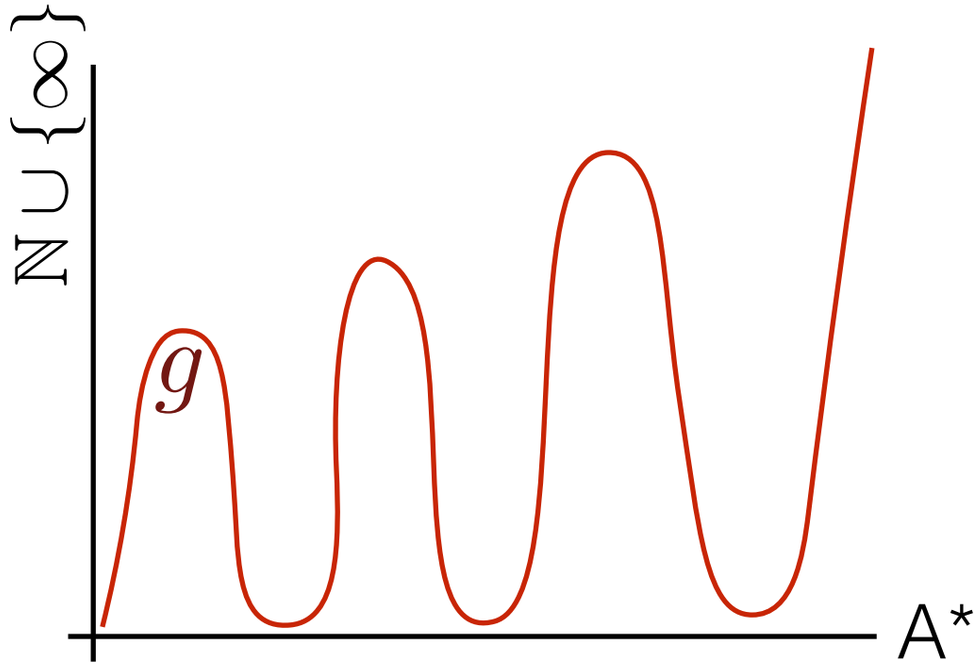
f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.



Boundedness equivalence

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|x$ is bounded implies $f|x$ is bounded

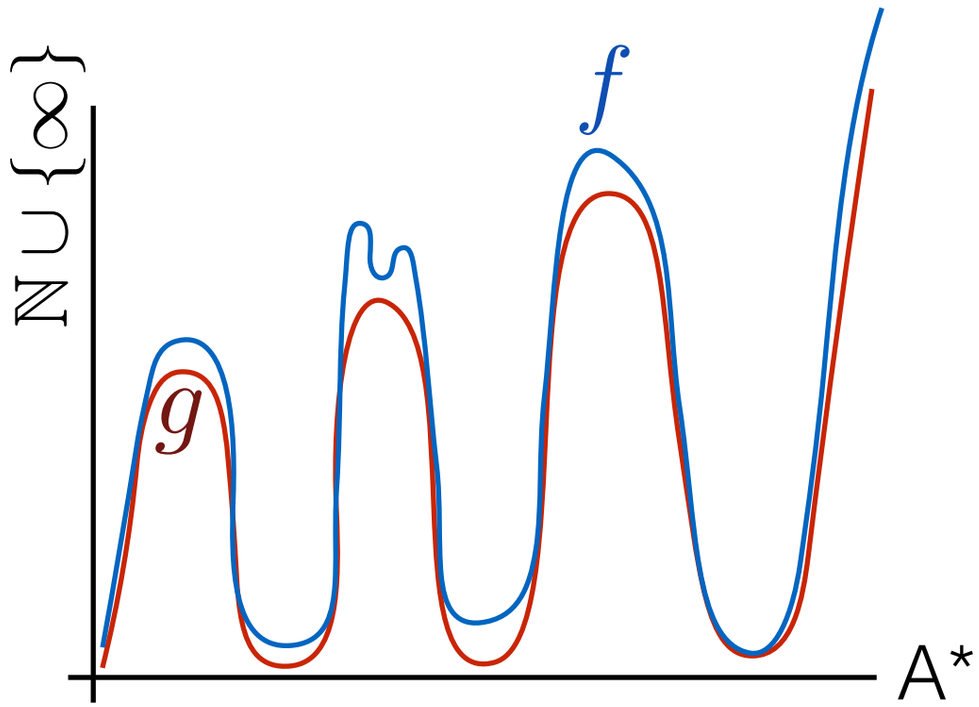
f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.



Boundedness equivalence

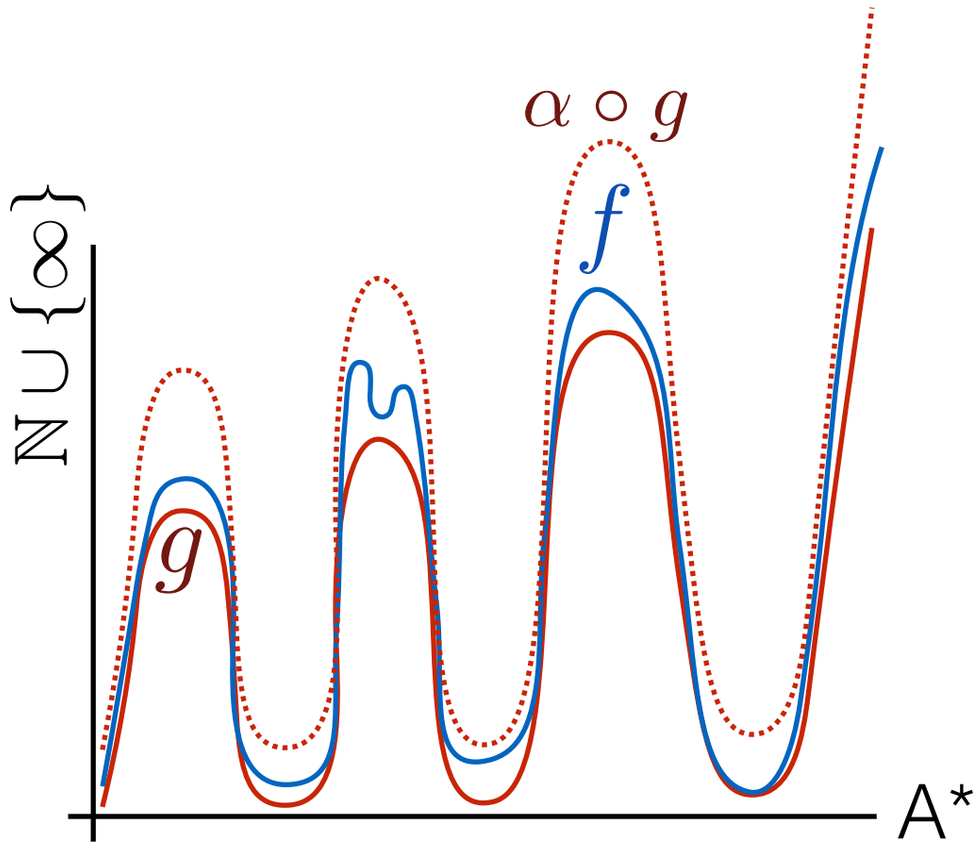
$f \preccurlyeq g$ (f **is dominated** by g) if for all sets X⊆A*
$g|x$ is bounded implies $f|x$ is bounded

f≈g if f≼g and g≼f
cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.



Boundedness equivalence

$f \preccurlyeq g$ (f **is dominated** by g) if for all sets X⊆A* $g|x$ is bounded implies $f|x$ is bounded

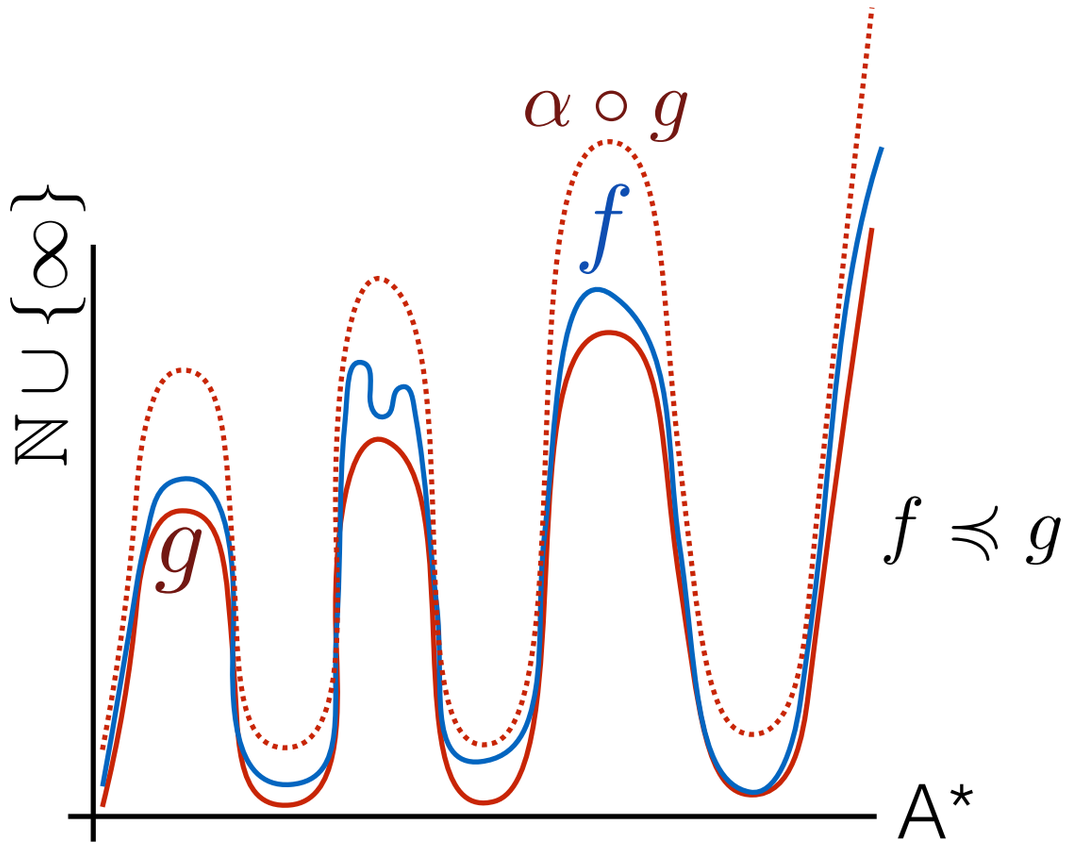
f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.



Boundedness equivalence

$f \preccurlyeq g$ (f **is dominated** by g) if for all sets X⊆A* $g|x$ is bounded implies $f|x$ is bounded

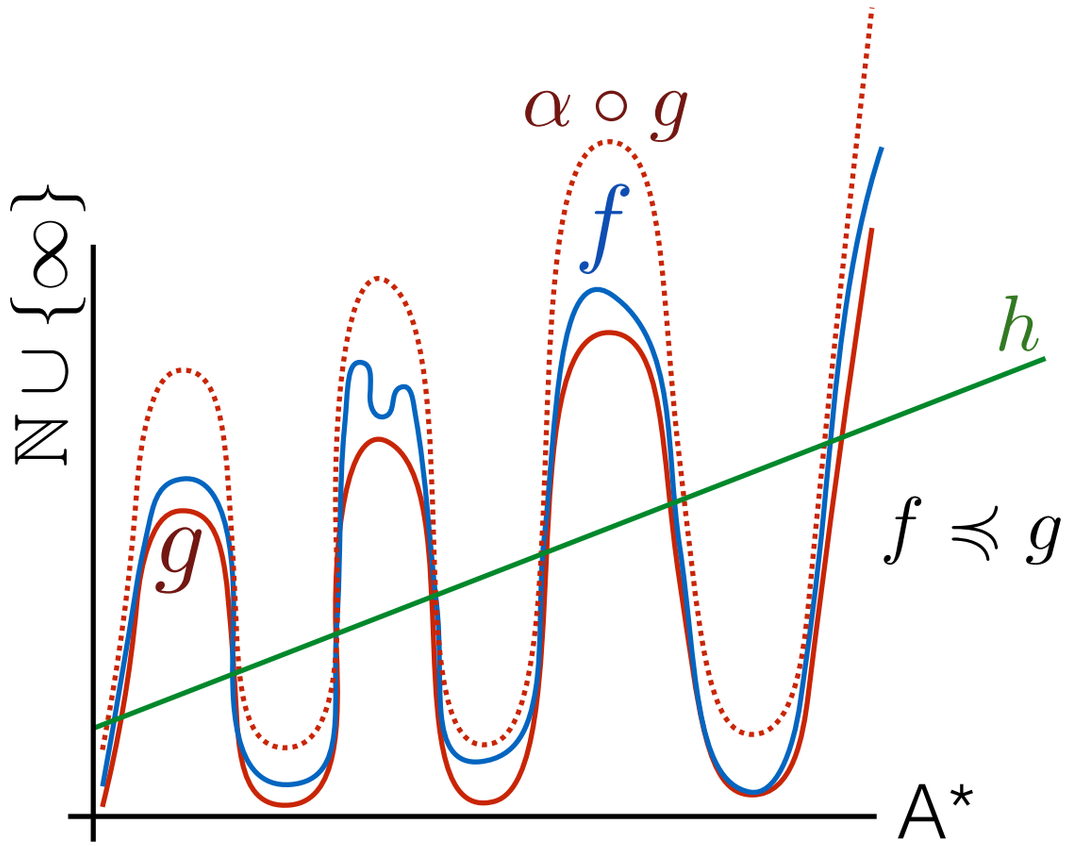
f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.



Boundedness equivalence

$f \preccurlyeq g$ (f **is dominated** by g) if for all sets X⊆A* $g|x$ is bounded implies $f|x$ is bounded

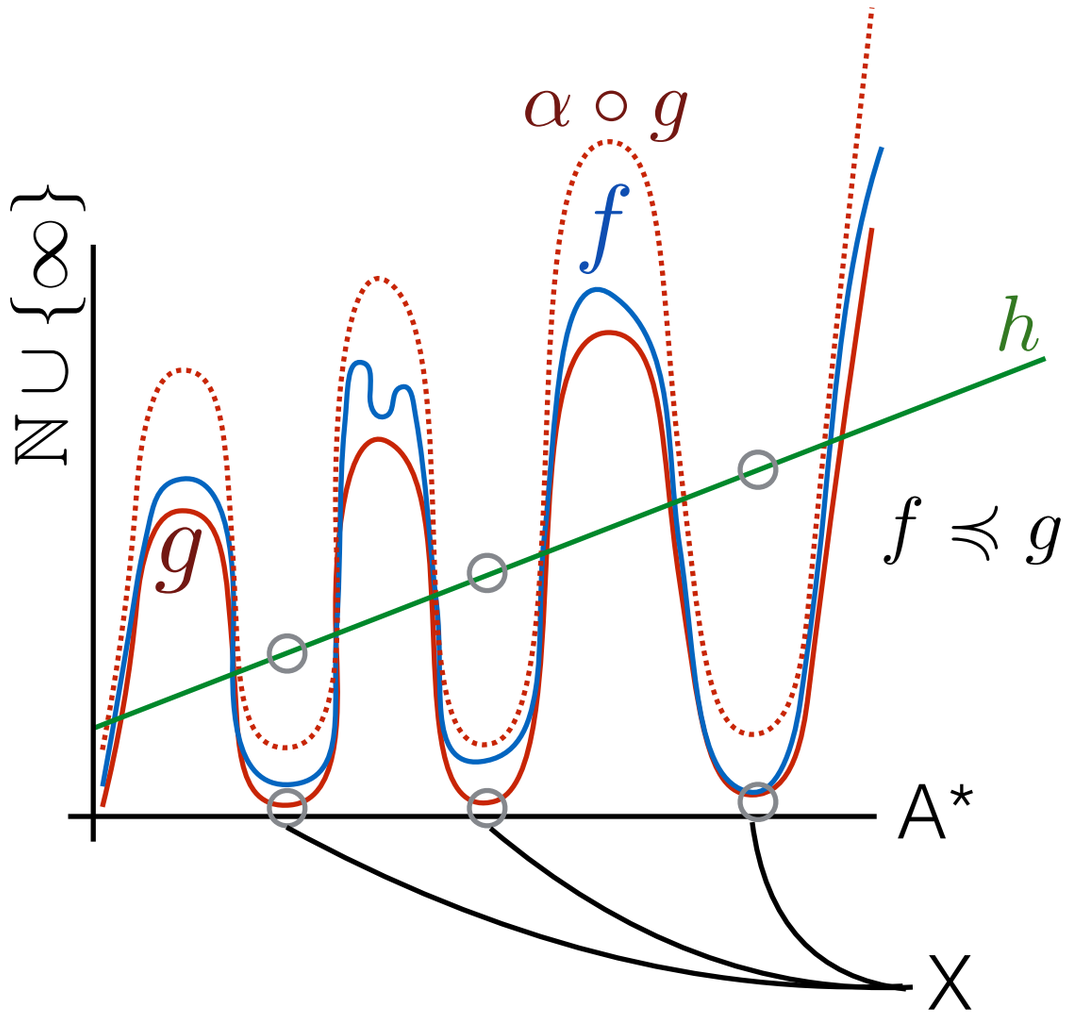
f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.



Boundedness equivalence

$f \preccurlyeq g$ (f **is dominated** by g) if for all sets X⊆A* $g|x$ is bounded implies $f|x$ is bounded

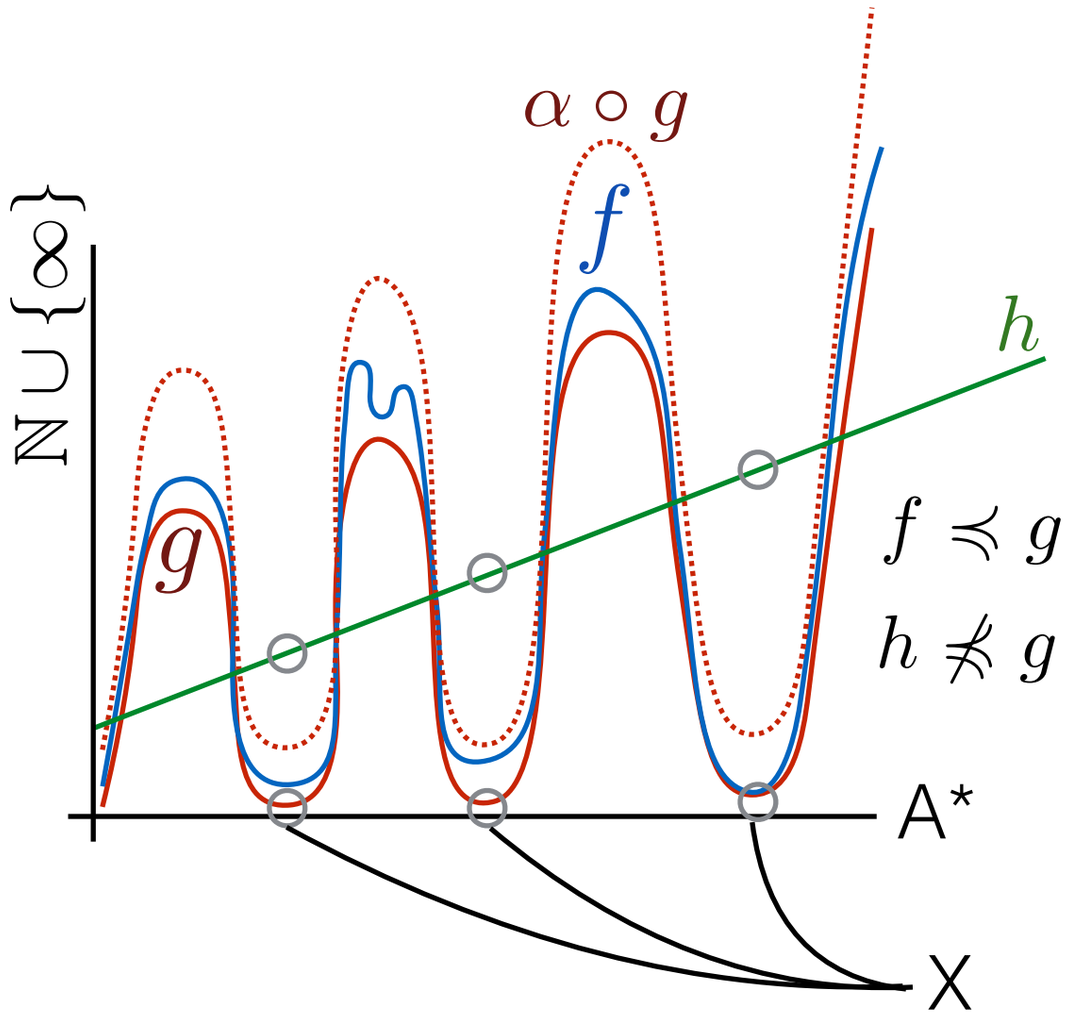
f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.



Boundedness equivalence

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|x$ is bounded implies $f|x$ is bounded

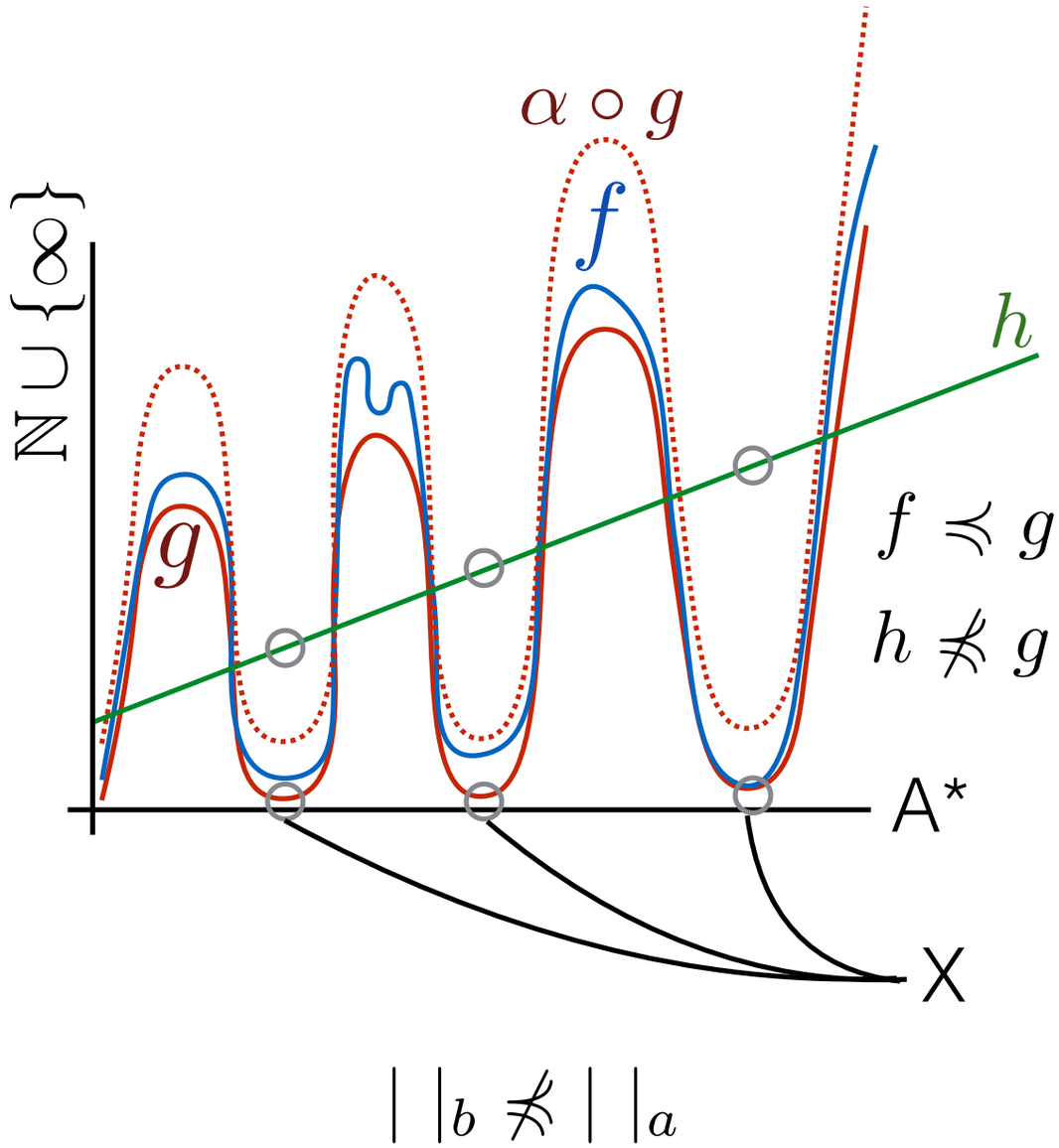
f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.



Boundedness equivalence

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|x$ is bounded implies $f|x$ is bounded

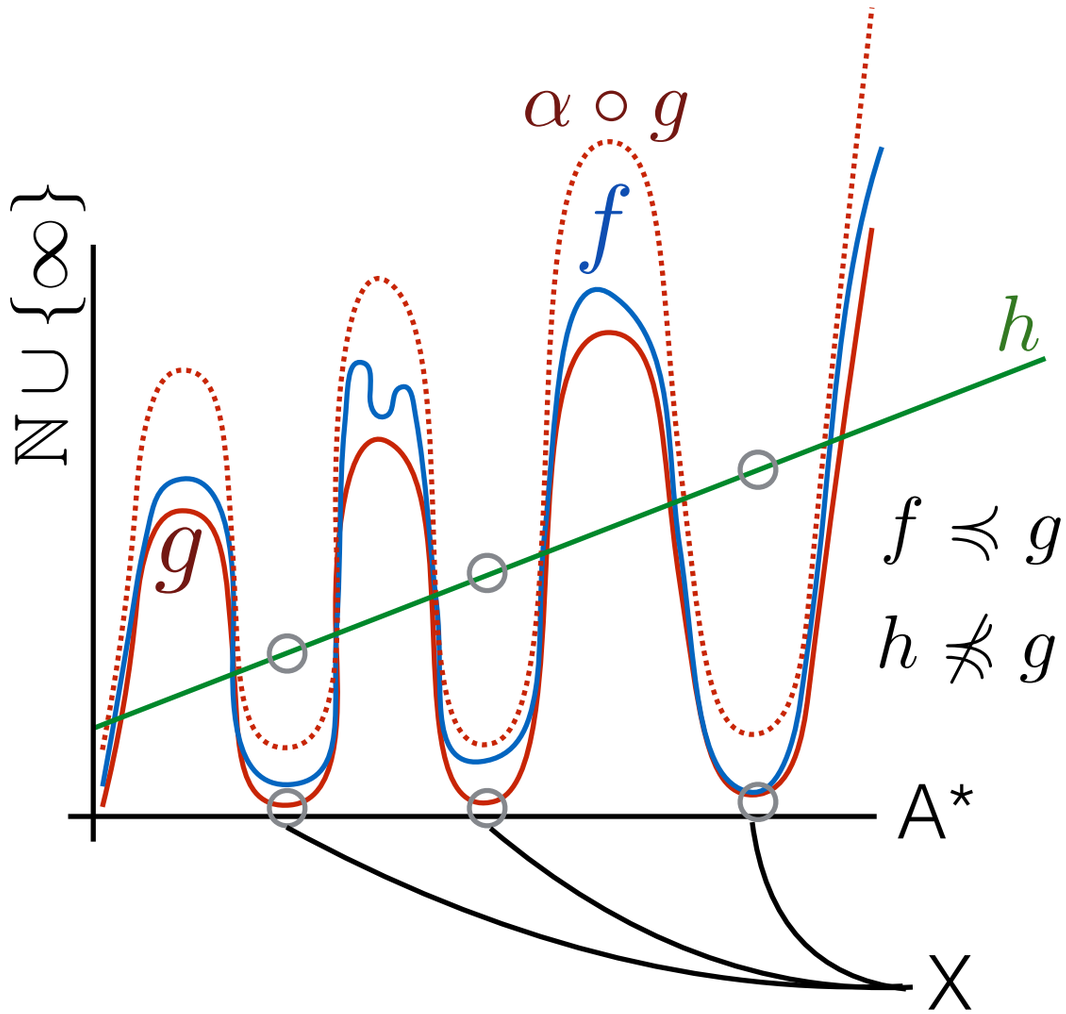
f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.



$||_b \not\preccurlyeq ||_a$

$||_{aa^+} \preccurlyeq ||_a$

Boundedness equivalence

$f \preccurlyeq g$ (f **is dominated** by g) if for all sets X⊆A* $g|x$ is bounded implies $f|x$ is bounded

f≈g if f≼g and g≼f

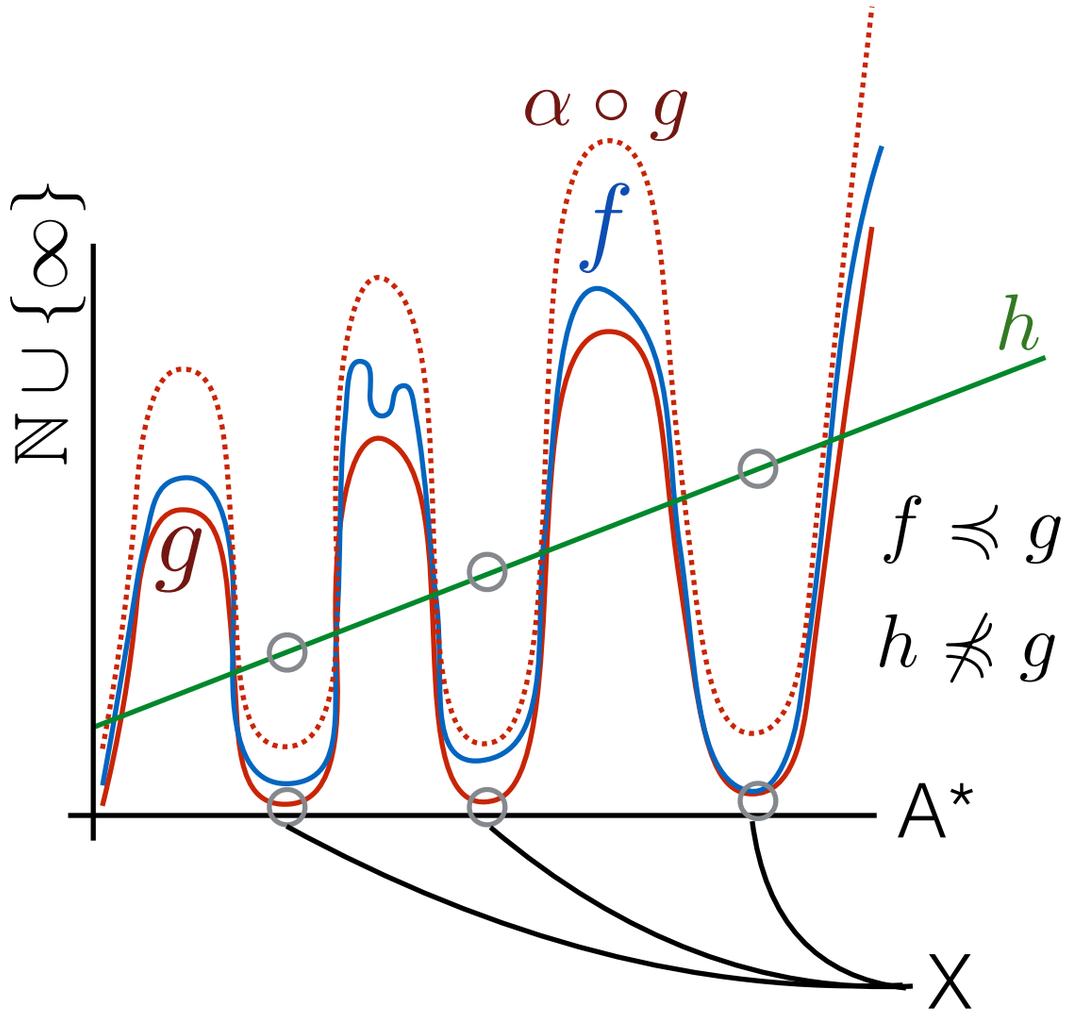
cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.

Let $F(f) = \{X \subseteq A^* : f|_X \text{is bounded}\}$



$|\ |_b \not\preccurlyeq |\ |_a$

$|\ |_{aa^+} \preccurlyeq |\ |_a$

Boundedness equivalence

$f \preccurlyeq g$ (f **is dominated** by g) if for all sets X⊆A* $g|_X$ is bounded implies $f|_X$ is bounded

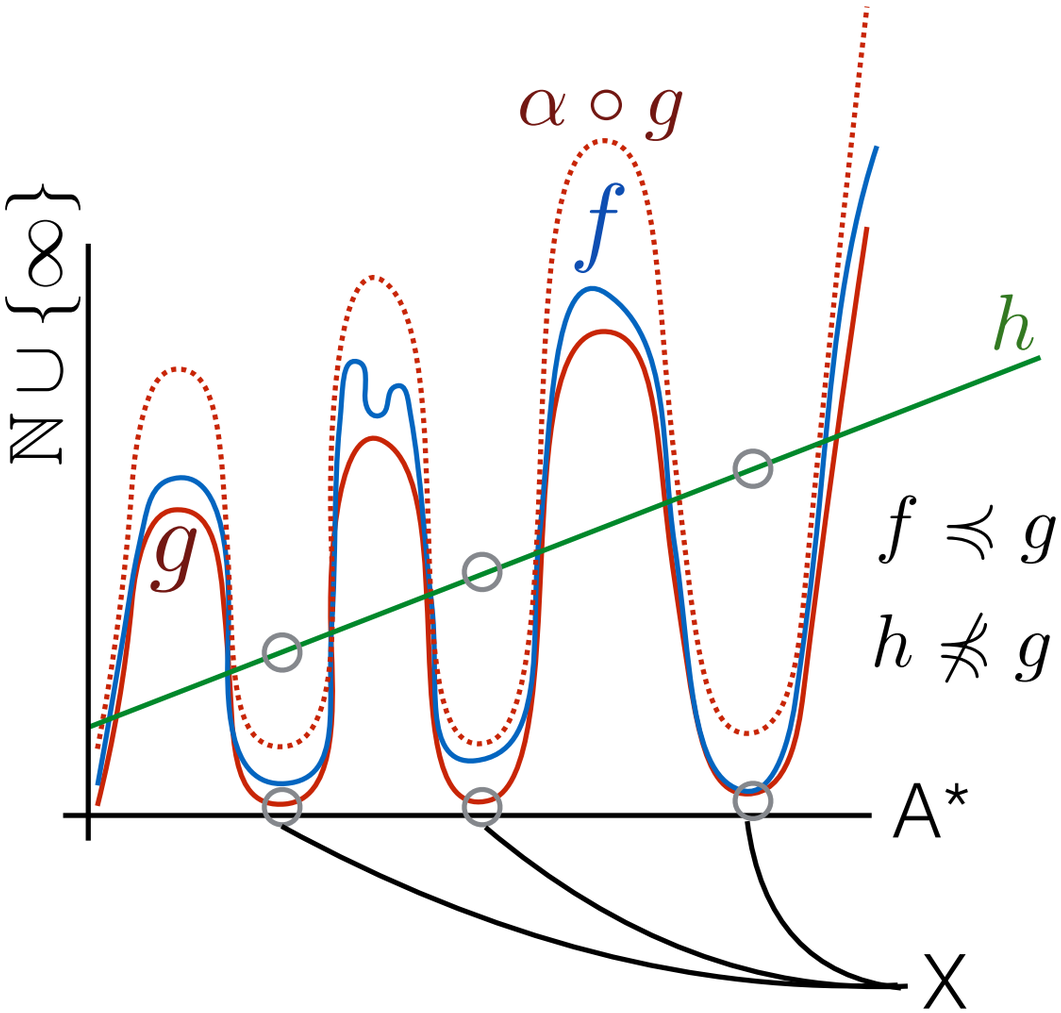
f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.



Let $F(f) = \{X \subseteq A^* : f|_X \text{is bounded}\}$

- $F(f)$ is a filter of countable basis.

$||_b \not\preccurlyeq ||_a$

$||_{aa^+} \preccurlyeq ||_a$

Boundedness equivalence

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|_X$ is bounded implies $f|_X$ is bounded

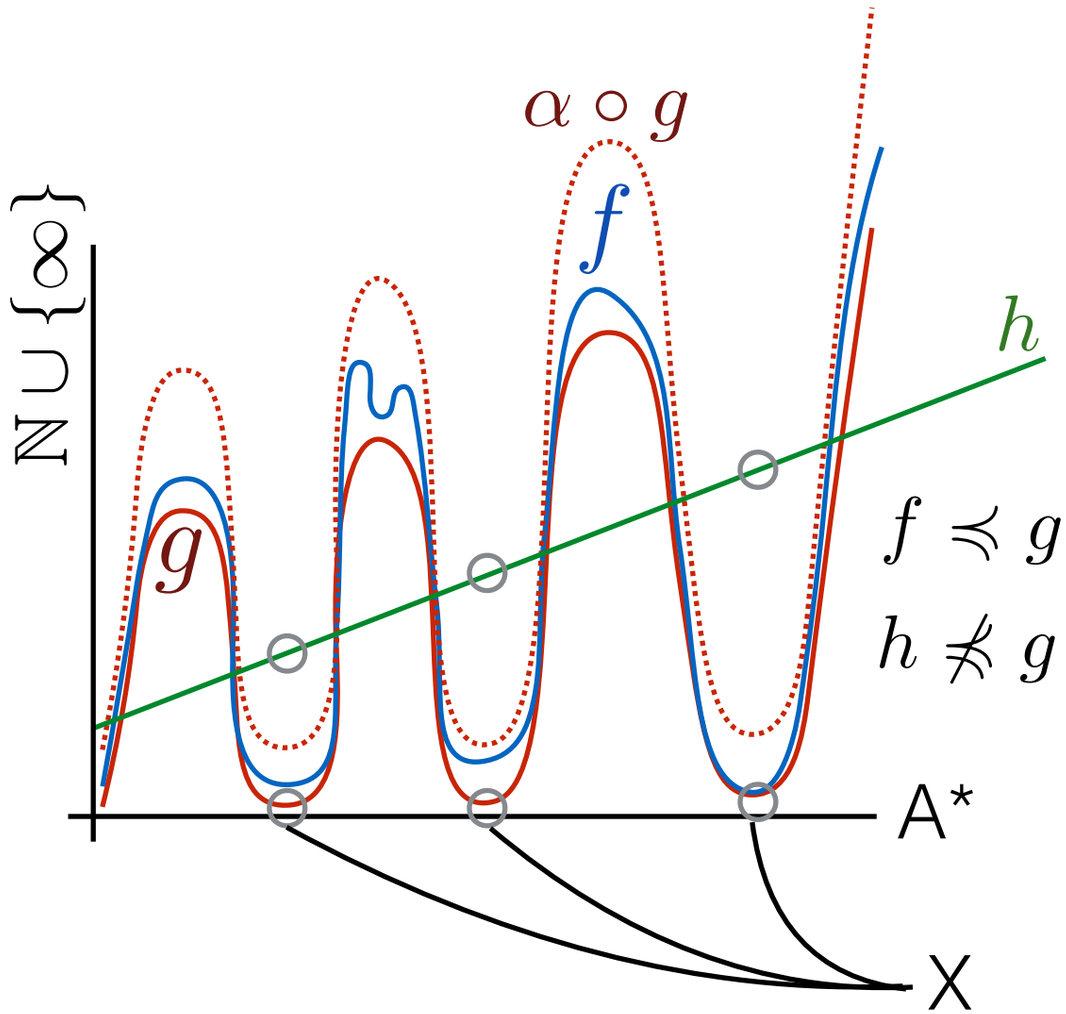
f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.



Let $F(f) = \{X \subseteq A^* : f|_X \text{is bounded}\}$

- $F(f)$ is a filter of countable basis.
- F is a bijection between cost functions and filters of countable basis.

$| \, |_b \not\preccurlyeq | \, |_a$

$| \, |_{aa^+} \preccurlyeq | \, |_a$

Boundedness equivalence

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|_X$ is bounded implies $f|_X$ is bounded

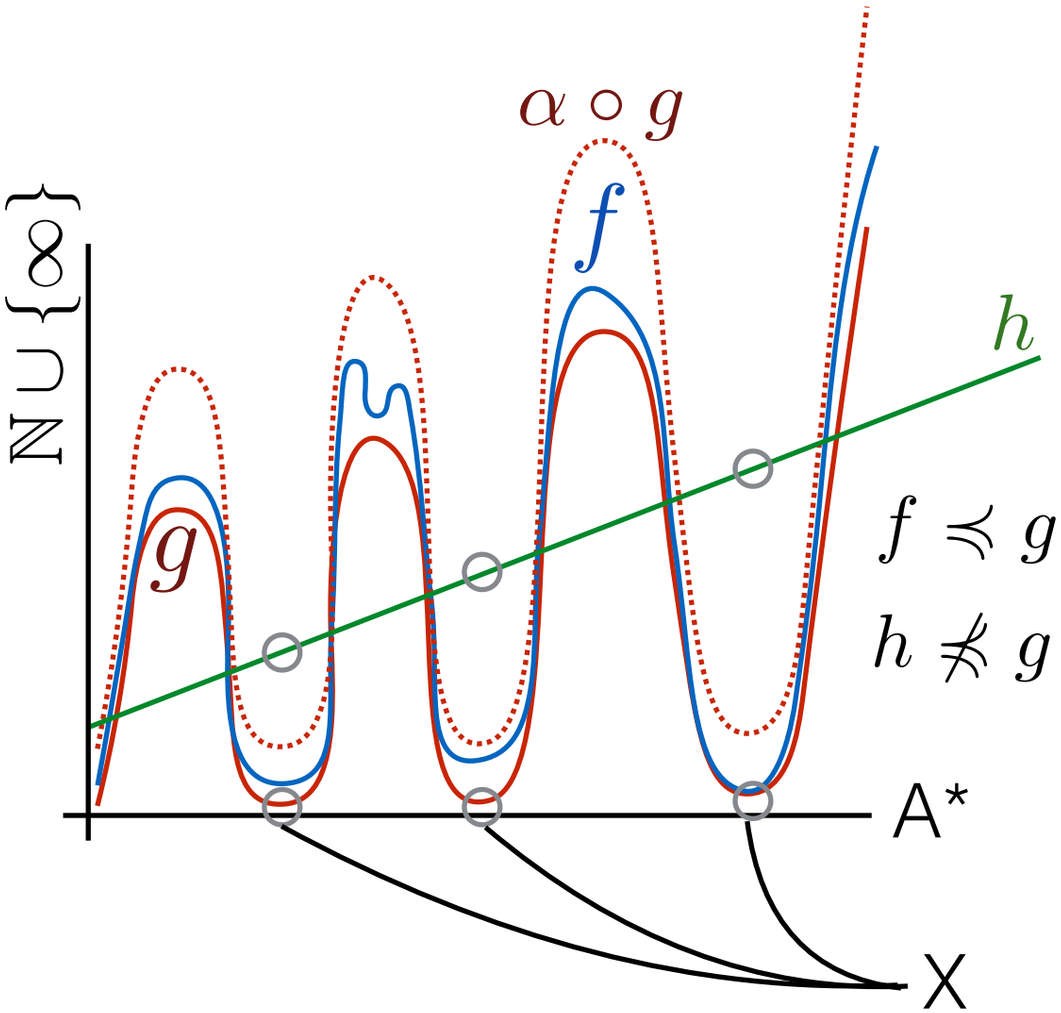
f≈g if f≼g and g≼f

cost function = ≈-class

Lemma: $f \preccurlyeq g$ if and only if

$$f \leqslant \alpha \circ g$$

for some non-decreasing map α from naturals to naturals extended with $\alpha(\infty)=\infty$.



Let $F(f) = \{X \subseteq A^* : f|_X \text{is bounded}\}$

- $F(f)$ is a filter of countable basis.
- F is a bijection between cost functions and filters of countable basis.

$| \, |_b \not\preccurlyeq | \, |_a$

$| \, |_{aa^+} \preccurlyeq | \, |_a$

Cost functions = filters of countable basis.

Regular cost functions over finite words

Cost functions are effectively equivalently described by

- machines: B-automata, S-automata
- expressions: B-rational/S-rational expressions,
- algebra: recognizability by stabilisation monoids,
- logically: definability in cost monadic second-order logic (cost MSO).

Regular cost functions over finite words

Cost functions are effectively equivalently described by

- machines: B-automata, S-automata
- expressions: B-rational/S-rational expressions,
- algebra: recognizability by stabilisation monoids,
- logically: definability in cost monadic second-order logic (cost MSO).

like distance automata with several counters and resets

Regular cost functions over finite words

Cost functions are effectively equivalently described by

- machines: B-automata, S-automata
- expressions: B-rational/S-rational expressions,
- algebra: recognizability by stabilisation monoids,
- logically: definability in cost monadic second-order logic (cost MSO).

like distance automata with several counters and resets

sup instead of inf

Regular cost functions over finite words

Cost functions are effectively equivalently described by

- machines: B-automata, S-automata
- expressions: B-rational/S-rational expressions,
- algebra: recognizability by stabilisation monoids,
- logically: definability in cost monadic second-order logic (cost MSO).

like distance automata with several counters and resets

sup instead of inf

new exponent $L^{\geqslant n}$

Regular cost functions over finite words

Cost functions are effectively equivalently described by

- machines: B-automata, S-automata
- expressions: B-rational/S-rational expressions,
- algebra: recognizability by stabilisation monoids,
- logically: definability in cost monadic second-order logic (cost MSO).

like distance automata with several counters and resets

sup instead of inf

new exponent $L^{\geqslant n}$

new exponent $L^{\leqslant n}$

Regular cost functions over finite words

Cost functions are effectively equivalently described by

- machines: B-automata, S-automata
- expressions: B-rational/S-rational expressions,
- algebra: recognizability by stabilisation monoids,
- logically: definability in cost monadic second-order logic (cost MSO).

like distance automata with several counters and resets

sup instead of inf

new exponent $L^{\geqslant n}$

new exponent $L^{\leqslant n}$

like ordered monoids with a stabilisation ‘#’ meaning ‘iterating a lot of times’.

Regular cost functions over finite words

Cost functions are effectively equivalently described by

- machines: B-automata, S-automata
- expressions: B-rational/S-rational expressions,
- algebra: recognizability by stabilisation monoids,
- logically: definability in cost monadic second-order logic (cost MSO).

like distance automata with several counters and resets

sup instead of inf

new exponent $L^{\geqslant n}$

new exponent $L^{\leqslant n}$

like ordered monoids with a stabilisation ‘#’ meaning ‘iterating a lot of times’.

new predicate $|X| \leqslant n$ that appears positively

Cost monadic second order logic

Cost monadic second order logic

Monadic second order logic (MSO) expresses properties using

- the predicates of the structure, and membership,
- Boolean connectives,
- existential and universal quantifiers over elements,
- existential and universal quantifiers over sets of elements.

Cost monadic second order logic

Monadic second order logic (MSO) expresses properties using

- the predicates of the structure, and membership,
- Boolean connectives,
- existential and universal quantifiers over elements,
- existential and universal quantifiers over sets of elements.

Example: over directed graphs,

Cost monadic second order logic

Monadic second order logic (MSO) expresses properties using

- the predicates of the structure, and membership,
- Boolean connectives,
- existential and universal quantifiers over elements,
- existential and universal quantifiers over sets of elements.

Example: over directed graphs,

$\texttt{Reach}(x, y)$ = all sets that contain x and are closed under the edge relation also contain y

Cost monadic second order logic

Monadic second order logic (MSO) expresses properties using

- the predicates of the structure, and membership,
- Boolean connectives,
- existential and universal quantifiers over elements,
- existential and universal quantifiers over sets of elements.

Example: over directed graphs,

$\mathtt{Reach}(x, y)$ = all sets that contain x and are closed under the edge relation also contain y

$$\forall Z \quad ((x \in Z) \wedge (\forall z \forall z' (z \in Z \wedge \mathtt{Edge}(z, z')) \to z' \in Z)) \quad \to y \in Z$$

Cost monadic second order logic

Monadic second order logic (MSO) expresses properties using

- the predicates of the structure, and membership,
- Boolean connectives,
- existential and universal quantifiers over elements,
- existential and universal quantifiers over sets of elements.

Example: over directed graphs,

$\mathtt{Reach}(x, y)$ = all sets that contain x and are closed under the edge relation also contain y

$$\forall Z \quad ((x \in Z) \wedge (\forall z \forall z' (z \in Z \wedge \mathtt{Edge}(z, z')) \to z' \in Z)) \quad \to y \in Z$$

Cost MSO = MSO + $|X| \leqslant n$ appearing positively.

Cost monadic second order logic

Monadic second order logic (MSO) expresses properties using

- the predicates of the structure, and membership,
- Boolean connectives,
- existential and universal quantifiers over elements,
- existential and universal quantifiers over sets of elements.

Example: over directed graphs,

$\mathtt{Reach}(x, y)$ = all sets that contain x and are closed under the edge relation also contain y

$$\forall Z \quad ((x \in Z) \wedge (\forall z \forall z' (z \in Z \wedge \mathtt{Edge}(z, z')) \to z' \in Z)) \quad \to y \in Z$$

Cost MSO = MSO + $|X| \leqslant n$ appearing positively.

Unique new variable ranging over naturals

Cost monadic second order logic

Monadic second order logic (MSO) expresses properties using

- the predicates of the structure, and membership,
- Boolean connectives,
- existential and universal quantifiers over elements,
- existential and universal quantifiers over sets of elements.

Example: over directed graphs,

$\mathtt{Reach}(x, y)$ = all sets that contain x and are closed under the edge relation also contain y

$$\forall Z \quad ((x \in Z) \wedge (\forall z \forall z' (z \in Z \wedge \mathtt{Edge}(z, z')) \to z' \in Z)) \quad \to y \in Z$$

Cost MSO = MSO + $|X| \leqslant n$ appearing positively.

Unique new variable ranging over naturals

A formula computes: $[\![\varphi]\!](\mathfrak{A}) = \inf\{n \mid \mathfrak{A} \models \varphi(n)\}$

Cost monadic second order logic

Monadic second order logic (MSO) expresses properties using

- the predicates of the structure, and membership,
- Boolean connectives,
- existential and universal quantifiers over elements,
- existential and universal quantifiers over sets of elements.

Example: over directed graphs,

$\texttt{Reach}(x, y)$ = all sets that contain x and are closed under the edge relation also contain y

$$\forall Z \quad ((x \in Z) \wedge (\forall z \forall z' (z \in Z \wedge \texttt{Edge}(z, z')) \to z' \in Z)) \quad \to y \in Z$$

Cost MSO = MSO + $|X| \leqslant n$ appearing positively.

Unique new variable ranging over naturals

A formula computes: $[\![\varphi]\!](\mathfrak{A}) = \inf\{n \mid \mathfrak{A} \models \varphi(n)\}$

Example: over directed graphs,

$\texttt{Diameter}() = \forall x \forall y \exists Z \, \texttt{Reach}(x, y, Z) \wedge |Z| \leqslant n$

Cost monadic second order logic

Monadic second order logic (MSO) expresses properties using

- the predicates of the structure, and membership,
- Boolean connectives,
- existential and universal quantifiers over elements,
- existential and universal quantifiers over sets of elements.

Example: over directed graphs,

$\mathtt{Reach}(x, y)$ = all sets that contain x and are closed under the edge relation also contain y

$$\forall Z \quad ((x \in Z) \wedge (\forall z \forall z' (z \in Z \wedge \mathtt{Edge}(z, z')) \to z' \in Z)) \quad \to y \in Z$$

Cost MSO = MSO + $|X| \leqslant n$ appearing positively.

Unique new variable ranging over naturals

A formula computes: $[\![\varphi]\!](\mathfrak{A}) = \inf\{n \mid \mathfrak{A} \models \varphi(n)\}$

Example: over directed graphs,

$\mathtt{Diameter}() = \forall x \forall y \exists Z\, \mathtt{Reach}(x, y, Z) \wedge |Z| \leqslant n$

Over words, $|\ |_a$, $\mathrm{maxblock}$, $\mathrm{minblock}$, distance automata...

Regular cost functions over finite words

Cost functions are effectively equivalently described by

- machines: B-automata, S-automata
- expressions: B-rational/S-rational expressions,
- algebra: recognizability by stabilisation monoids,
- logically: definability in cost monadic second-order logic (cost MSO).

like distance automata with several counters and resets

sup instead of inf

new exponent $L^{\geqslant n}$

new exponent $L^{\leqslant n}$

like ordered monoids with a stabilisation ‘#’ meaning ‘iterating a lot of times’.

new predicate $|X| \leqslant n$ that appears positively

Regular cost functions over finite words

Cost functions are effectively equivalently described by

- machines: B-automata, S-automata
- expressions: B-rational/S-rational expressions,
- algebra: recognizability by stabilisation monoids,
- logically: definability in cost monadic second-order logic (cost MSO).

Regular cost functions are closed under **min**, **max**, **inf-projection**, **sup-projection**, etc…

like distance automata with several counters and resets

sup instead of inf

new exponent $L^{\geqslant n}$

new exponent $L^{\leqslant n}$

like ordered monoids with a stabilisation '#' meaning 'iterating a lot of times'.

new predicate $|X| \leqslant n$ that appears positively

Regular cost functions over finite words

Cost functions are effectively equivalently described by

- machines: B-automata, S-automata
- expressions: B-rational/S-rational expressions,
- algebra: recognizability by stabilisation monoids,
- logically: definability in cost monadic second-order logic (cost MSO).

Regular cost functions are closed under **min**, **max**, **inf-projection**, **sup-projection**, etc…

Boundedness, **divergence**, **domination**, **equivalence** are decidable.

like distance automata with several counters and resets

sup instead of inf

new exponent $L^{\geqslant n}$

new exponent $L^{\leqslant n}$

like ordered monoids with a stabilisation ‘#’ meaning ‘iterating a lot of times’.

new predicate $|X| \leqslant n$ that appears positively

Regular cost functions over finite words

Cost functions are effectively equivalently described by

- machines: B-automata, S-automata
- expressions: B-rational/S-rational expressions,
- algebra: recognizability by stabilisation monoids,
- logically: definability in cost monadic second-order logic (cost MSO).

like distance automata with several counters and resets

sup instead of inf

new exponent $L^{\geqslant n}$

new exponent $L^{\leqslant n}$

like ordered monoids with a stabilisation ‘#’ meaning ‘iterating a lot of times’.

new predicate $|X| \leqslant n$ that appears positively

Regular cost functions are closed under **min**, **max**, **inf-projection**, **sup-projection**, etc…

Boundedness, **divergence**, **domination**, **equivalence** are decidable.

These results (in essence) extend regular languages.

Languages as cost functions

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|_X$ is bounded implies $f|_X$ is bounded

f≈g if f≼g and g≼f

cost function = ≈-class

Languages as cost functions

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|_X$ is bounded implies $f|_X$ is bounded

f≈g if f≼g and g≼f

cost function = ≈-class

Given a language L, its characteristic map is:

$$\chi_L : A^* \to \mathbb{N} \cup \{\infty\}$$

$$u \mapsto \begin{cases} 0 & \text{if } u \in L \\ \infty & \text{otherwise} \end{cases}$$

Languages as cost functions

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|_X$ is bounded implies $f|_X$ is bounded

f≈g if f≼g and g≼f

cost function = ≈-class

Given a language L, its characteristic map is:

$$\chi_L : A^* \to \mathbb{N} \cup \{\infty\}$$

$$u \mapsto \begin{cases} 0 & \text{if } u \in L \\ \infty & \text{otherwise} \end{cases}$$

Remark: $K \subseteq L$ iff $\chi_L \preccurlyeq \chi_K$.

Languages as cost functions

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|_X$ is bounded implies $f|_X$ is bounded

f≈g if f≼g and g≼f

cost function = ≈-class

Given a language L, its characteristic map is:

$$\chi_L : A^* \to \mathbb{N} \cup \{\infty\}$$

$$u \mapsto \begin{cases} 0 & \text{if } u \in L \\ \infty & \text{otherwise} \end{cases}$$

Remark: $K \subseteq L$ iff $\chi_L \preccurlyeq \chi_K$.

Consider an MSO formula defining L, then

Languages as cost functions

$f \preccurlyeq g$ (f is dominated by g) if for all sets X⊆A* $g|_X$ is bounded implies $f|_X$ is bounded

f≈g if f≼g and g≼f

cost function = ≈-class

Given a language L, its characteristic map is:

$$\chi_L : A^* \to \mathbb{N} \cup \{\infty\}$$

$$u \mapsto \begin{cases} 0 & \text{if } u \in L \\ \infty & \text{otherwise} \end{cases}$$

Remark: $K \subseteq L$ iff $\chi_L \preccurlyeq \chi_K$.

Consider an MSO formula defining L, then

$$[\![\varphi]\!](u) = \inf\{n \mid u \models \varphi(n)\}$$

Languages as cost functions

$f \preccurlyeq g$ (f **is dominated** by g) if for all sets X⊆A*
$g|_X$ is bounded implies $f|_X$ is bounded

f≈g if f≼g and g≼f
cost function = ≈-class

Given a language L, its **characteristic map** is:

$$\chi_L : A^* \to \mathbb{N} \cup \{\infty\}$$

$$u \mapsto \begin{cases} 0 & \text{if } u \in L \\ \infty & \text{otherwise} \end{cases}$$

Remark: $K \subseteq L$ iff $\chi_L \preccurlyeq \chi_K$.

Consider an MSO formula defining L, then

$$[\![\varphi]\!](u) = \inf\{n \mid u \models \varphi(n)\} = \begin{cases} 0 & \text{if } u \models \varphi \\ \infty & \text{otherwise} \end{cases}$$

Languages as cost functions

$f \preccurlyeq g$ (f **is dominated** by g) if for all sets X⊆A* $g|_X$ is bounded implies $f|_X$ is bounded

f≈g if f≼g and g≼f

cost function = ≈-class

Given a language L, its **characteristic map** is:

$$\chi_L : A^* \to \mathbb{N} \cup \{\infty\}$$

$$u \mapsto \begin{cases} 0 & \text{if } u \in L \\ \infty & \text{otherwise} \end{cases}$$

Remark: $K \subseteq L$ iff $\chi_L \preccurlyeq \chi_K$.

Consider an MSO formula defining L, then

$$[\![\varphi]\!](u) = \inf\{n \mid u \models \varphi(n)\} = \begin{cases} 0 & \text{if } u \models \varphi \\ \infty & \text{otherwise} \end{cases} = \chi_L$$

Languages as cost functions

$f \preccurlyeq g$ (f **is dominated** by g) if for all sets X⊆A* $g|_X$ is bounded implies $f|_X$ is bounded

f≈g if f≼g and g≼f

cost function = ≈-class

Given a language L, its **characteristic map** is:

$$\chi_L : A^* \to \mathbb{N} \cup \{\infty\}$$

$$u \mapsto \begin{cases} 0 & \text{if } u \in L \\ \infty & \text{otherwise} \end{cases}$$

Remark: $K \subseteq L$ iff $\chi_L \preccurlyeq \chi_K$.

Consider an MSO formula defining L, then

$$[\![\varphi]\!](u) = \inf\{n \mid u \models \varphi(n)\} = \begin{cases} 0 & \text{if } u \models \varphi \\ \infty & \text{otherwise} \end{cases} = \chi_L$$

Similar relation with all classical objects exist.

Some known results about regular cost functions

Some known results about regular cost functions

Boundedness/domination of cost MSO is decidable

Some known results about regular cost functions

Boundedness/domination of cost MSO is decidable

- over finite words [C.09,13]

Some known results about regular cost functions

Boundedness/domination of cost MSO is decidable

- over finite words [C.09,13]
- over infinite words (of countable length ? !)
  [C.09], [vanden Boom&Kuperberg12]

Some known results about regular cost functions

Boundedness/domination of cost MSO is decidable

- over finite words [C.09,13]
- over infinite words (of countable length ? !)
  [C.09], [vanden Boom&Kuperberg12]
- over finite trees [C.&Löding10]

Some known results about regular cost functions

Boundedness/domination of cost MSO is decidable

- over finite words [C.09,13]
- over infinite words (of countable length ? !)
  [C.09], [vanden Boom&Kuperberg12]
- over finite trees [C.&Löding10]
- over graphs of tree-width at most k

Some known results about regular cost functions

Boundedness/domination of cost MSO is decidable

- over finite words [C.09,13]
- over infinite words (of countable length ? !)
  [C.09], [vanden Boom&Kuperberg12]
- over finite trees [C.&Löding10]
- over graphs of tree-width at most k
- partially over infinite trees (tamed trees, weak costMSO, …)
  [vanden Boom11], [vanden Boom&Kuperberg12] …

Some known results about regular cost functions

Boundedness/domination of cost MSO is decidable

- over finite words [C.09,13]
- over infinite words (of countable length ? !)
  [C.09], [vanden Boom&Kuperberg12]
- over finite trees [C.&Löding10]
- over graphs of tree-width at most k
- partially over infinite trees (tamed trees, weak costMSO, …)
  [vanden Boom11], [vanden Boom&Kuperberg12] …

Regular cost functions = Toolbox of results for deciding boundedness questions.

Some application of regular cost functions

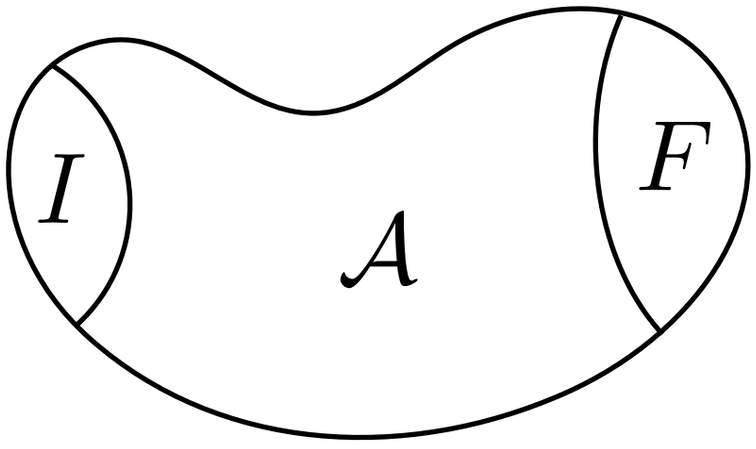
Finite power property

Finite power property

Finite power property: $L^* = L^{\leqslant n}$ for some n ?

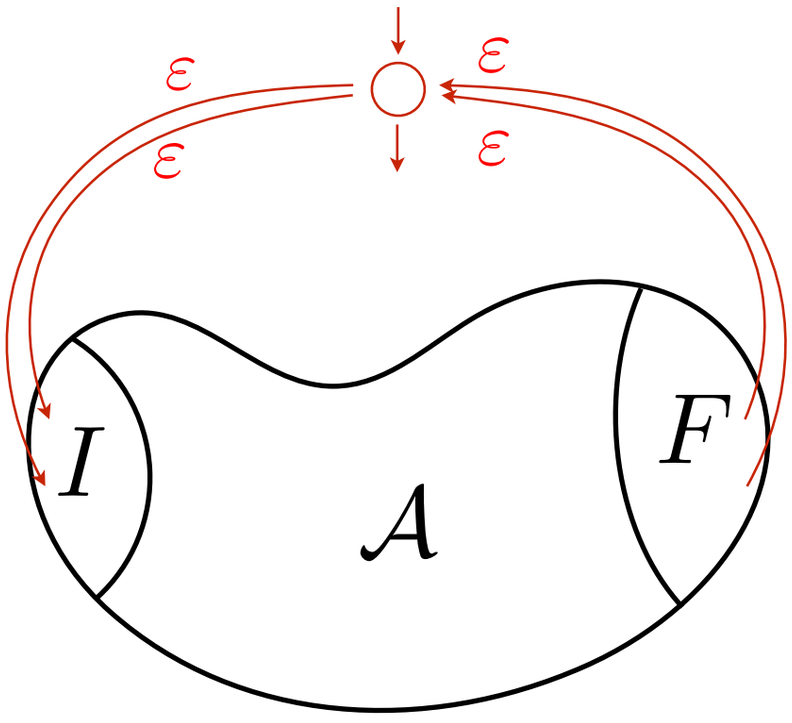
Finite power property

Finite power property: $L^* = L^{\leqslant n}$ for some n ?



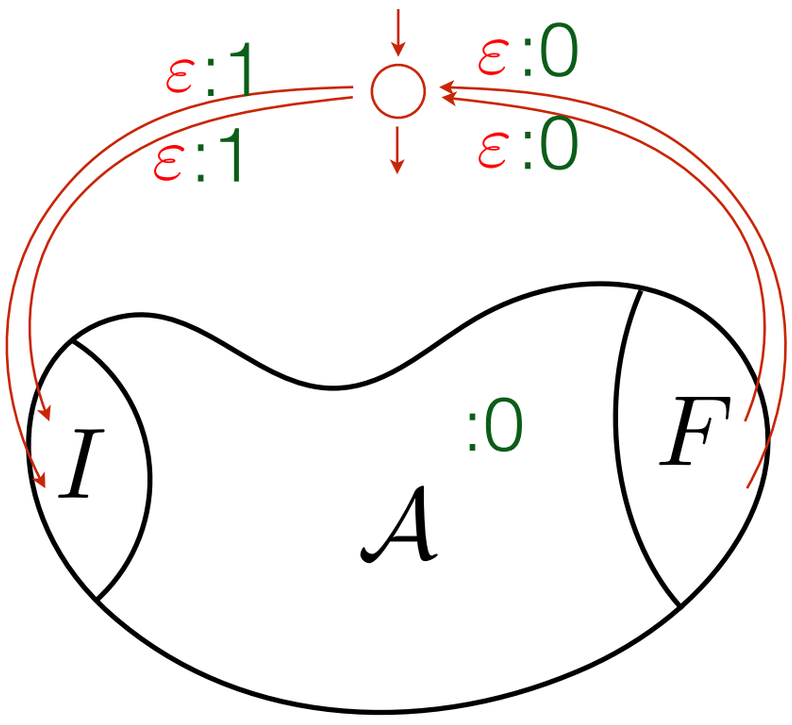
Finite power property

Finite power property: $L^* = L^{\leqslant n}$ for some n ?



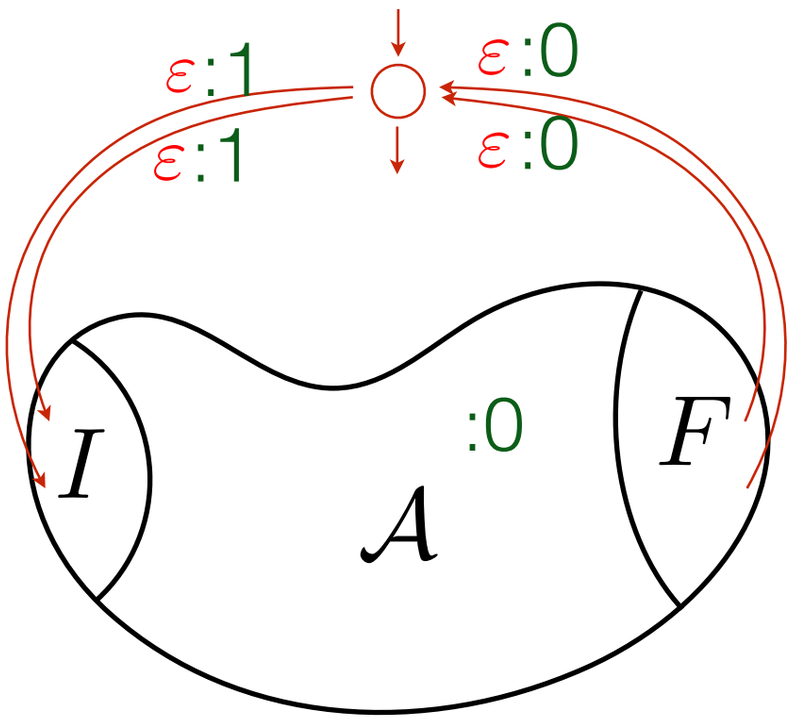
Finite power property

Finite power property: $L^* = L^{\leqslant n}$ for some n ?



Finite power property

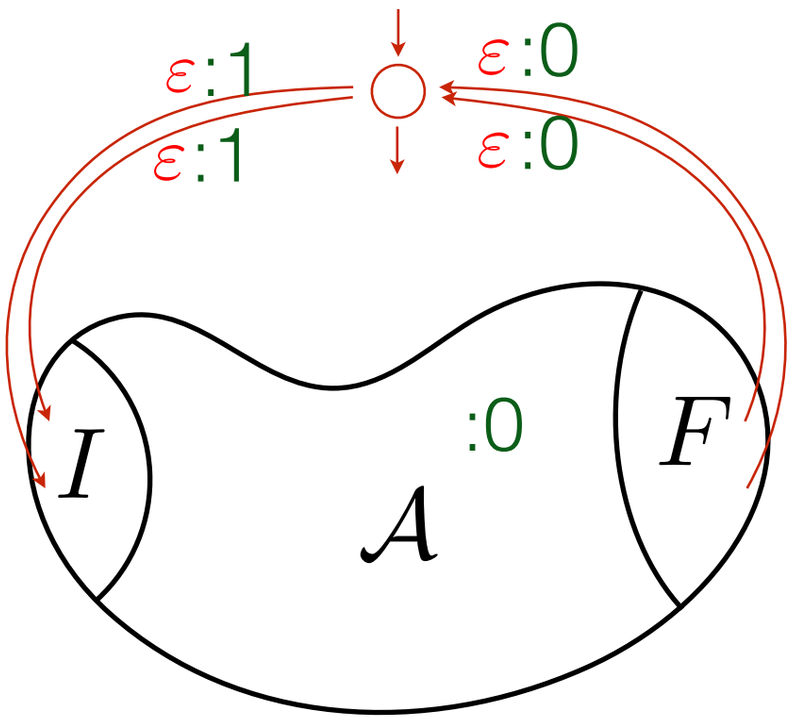
Finite power property: $L^* = L^{\leqslant n}$ for some n ?



Accepts u with weight $\leqslant n$
if and only if
$u \in L^{\leqslant n}$

Finite power property

Finite power property: $L^* = L^{\leqslant n}$ for some n ?

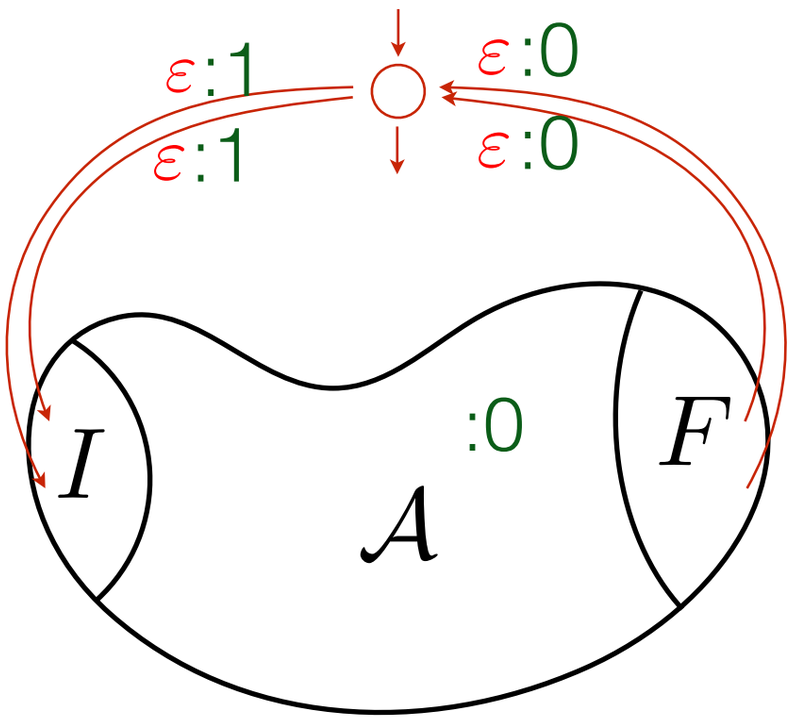


Accepts u with weight $\leqslant n$
if and only if
$u \in L^{\leqslant n}$

Hence $[\![\mathcal{A}]\!](u) = \inf\{n \mid u \in L^{\leqslant n}\}$

Finite power property

Finite power property: $L^* = L^{\leqslant n}$ for some n ?



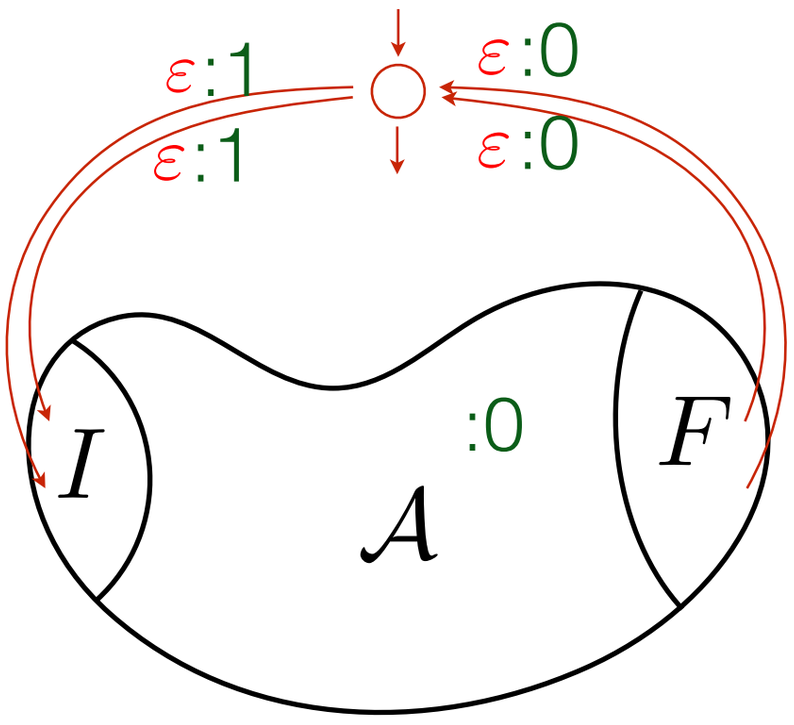
Accepts u with weight $\leqslant n$
if and only if
$u \in L^{\leqslant n}$

Hence $[\![\mathcal{A}]\!](u) = \inf\{n \mid u \in L^{\leqslant n}\}$

Lemma: $[\![\mathcal{A}]\!]$ is bounded over L^* if and only if $L^* = L^{\leqslant n}$.

Finite power property

Finite power property: $L^* = L^{\leqslant n}$ for some n ?



Accepts u with weight $\leqslant n$
if and only if
$u \in L^{\leqslant n}$

Hence $[\![\mathcal{A}]\!](u) = \inf\{n \mid u \in L^{\leqslant n}\}$

Lemma: $[\![\mathcal{A}]\!]$ is bounded over L^* if and only if $L^* = L^{\leqslant n}$.

[Simon78] The finite power property is decidable.

Bounding fixpoints

Bounding fixpoints

Boundedness: Let $\varphi(x, Y)$ be a monadic second-order formula positive in Y. It has a fixpoint which is the least set Y s.t. $\{x \mid \mathfrak{A} \models \varphi(x, Y)\} \subseteq Y$.

Bounding fixpoints

Boundedness: Let $\varphi(x, Y)$ be a monadic second-order formula positive in Y. It has a fixpoint which is the least set Y s.t. $\{x \mid \mathfrak{A} \models \varphi(x, Y)\} \subseteq Y$.

It is the least fixpoint of $F_{\mathfrak{A}} : Z \mapsto \{x \mid \mathfrak{A} \models \varphi(x, Z)\}$.

Bounding fixpoints

Boundedness: Let $\varphi(x, Y)$ be a monadic second-order formula positive in Y. It has a fixpoint which is the least set Y s.t. $\{x \mid \mathfrak{A} \models \varphi(x, Y)\} \subseteq Y$.

It is the least fixpoint of $F_{\mathfrak{A}} : Z \mapsto \{x \mid \mathfrak{A} \models \varphi(x, Z)\}$.

Problem:
Does there exist $n \in \mathbb{N}$ such that $F^n_{\mathfrak{A}}(\emptyset) = \mathit{fixpoint}(F_{\mathfrak{A}})$ for all $\mathfrak{A}$?

Bounding fixpoints

Boundedness: Let $\varphi(x, Y)$ be a monadic second-order formula positive in Y. It has a fixpoint which is the least set Y s.t. $\{x \mid \mathfrak{A} \models \varphi(x, Y)\} \subseteq Y$.

It is the least fixpoint of $F_{\mathfrak{A}} : Z \mapsto \{x \mid \mathfrak{A} \models \varphi(x, Z)\}$.

Problem:
Does there exist $n \in \mathbb{N}$ such that $F^n_{\mathfrak{A}}(\emptyset) = \mathit{fixpoint}(F_{\mathfrak{A}})$ for all $\mathfrak{A}$?

The map $(t, x) \mapsto \inf\{n \mid x \in F^n_t(\emptyset)\}$ for x a node of an infinite tree t is computed by an alternating two-way distance parity automaton running over t starting from x.

Bounding fixpoints

Boundedness: Let $\varphi(x, Y)$ be a monadic second-order formula positive in Y. It has a fixpoint which is the least set Y s.t. $\{x \mid \mathfrak{A} \models \varphi(x, Y)\} \subseteq Y$.

It is the least fixpoint of $F_{\mathfrak{A}} : Z \mapsto \{x \mid \mathfrak{A} \models \varphi(x, Z)\}$.

Problem:
Does there exist $n \in \mathbb{N}$ such that $F^n_{\mathfrak{A}}(\emptyset) = \mathit{fixpoint}(F_{\mathfrak{A}})$ for all $\mathfrak{A}$?

The map $(t, x) \mapsto \inf\{n \mid x \in F^n_t(\emptyset)\}$ for x a node of an infinite tree t is computed by an alternating two-way distance parity automaton running over t starting from x.

Boundedness of such automata is decidable.

Bounding fixpoints

Boundedness: Let $\varphi(x, Y)$ be a monadic second-order formula positive in Y. It has a fixpoint which is the least set Y s.t. $\{x \mid \mathfrak{A} \models \varphi(x, Y)\} \subseteq Y$.

It is the least fixpoint of $F_{\mathfrak{A}} : Z \mapsto \{x \mid \mathfrak{A} \models \varphi(x, Z)\}$.

Problem:
Does there exist $n \in \mathbb{N}$ such that $F^n_{\mathfrak{A}}(\emptyset) = \mathit{fixpoint}(F_{\mathfrak{A}})$ for all $\mathfrak{A}$?

The map $(t, x) \mapsto \inf\{n \mid x \in F^n_t(\emptyset)\}$ for x a node of an infinite tree t is computed by an alternating two-way distance parity automaton running over t starting from x.

Boundedness of such automata is decidable.

[Blumensath&Otto&Weyer12] The boundedness of monadic second-order logic fixpoints is decidable over infinite trees.

Membership to (non-deterministic) classes

Membership to (non-deterministic) classes

A very inspiring family of questions over regular languages is the ability to decide the membership of regular languages to subclasses.

Membership to (non-deterministic) classes

A very inspiring family of questions over regular languages is the ability to decide the membership of regular languages to subclasses.

[Schützenberger65] A regular language is definable by a star-free expressions if and only if its syntactic monoid is aperiodic.
This is decidable.

Membership to (non-deterministic) classes

A very inspiring family of questions over regular languages is the ability to decide the membership of regular languages to subclasses.

[Schützenberger65] A regular language is definable by a star-free expressions if and only if its syntactic monoid is aperiodic.
This is decidable.

Many other classes known, over finite words,

Σ_1 FO_2 Σ_2 Σ_3 Σ_4 $B(\Sigma_1)$ $B(\Sigma_2)$ etc…

The star-height problem

The star-height problem

Def: A regular language is of star height k if it can be described by a rational expression using at most k nesting of Kleene stars, but not less.

The star-height problem

Def: A regular language is of star height k if it can be described by a rational expression using at most k nesting of Kleene stars, but not less.

[Eggan 63] The star-height forms a strict hierarchy over regular languages.

The star-height problem

Def: A regular language is of star height k if it can be described by a rational expression using at most k nesting of Kleene stars, but not less.

[Eggan 63] The star-height forms a strict hierarchy over regular languages.(*)

(*) This is the restricted star height, in which intersection and negations are not allowed. Nobody knows if the hierarchy is strict in the unrestricted case.

The star-height problem

Def: A regular language is of star height k if it can be described by a rational expression using at most k nesting of Kleene stars, but not less.

[Eggan 63] The star-height forms a strict hierarchy over regular languages.(*)

But, is the star-height of a language computable ?

(*) This is the restricted star height, in which intersection and negations are not allowed. Nobody knows if the hierarchy is strict in the unrestricted case.

The star-height problem

Def: A regular language is of star height k if it can be described by a rational expression using at most k nesting of Kleene stars, but not less.

[Eggan 63] The star-height forms a strict hierarchy over regular languages.(*)

But, is the star-height of a language computable ?

[Hashiguchi 81–88] Yes!

(*) This is the restricted star height, in which intersection and negations are not allowed. Nobody knows if the hierarchy is strict in the unrestricted case.

The star-height problem

Def: A regular language is of star height k if it can be described by a rational expression using at most k nesting of Kleene stars, but not less.

[Eggan 63] The star-height forms a strict hierarchy over regular languages.(*)

But, is the star-height of a language computable ?

[Hashiguchi 81–88] Yes!

Decidability of the boundedness of distance automata

(*) This is the restricted star height, in which intersection and negations are not allowed. Nobody knows if the hierarchy is strict in the unrestricted case.

The star-height problem

Def: A regular language is of star height k if it can be described by a rational expression using at most k nesting of Kleene stars, but not less.

[Eggan 63] The star-height forms a strict hierarchy over regular languages.(*)

But, is the star-height of a language computable ?

[Hashiguchi 81–88] Yes! (but very complicated)

Decidability of the boundedness of distance automata

(*) This is the restricted star height, in which intersection and negations are not allowed. Nobody knows if the hierarchy is strict in the unrestricted case.

The star-height problem

Def: A regular language is of star height k if it can be described by a rational expression using at most k nesting of Kleene stars, but not less.

[Eggan 63] The star-height forms a strict hierarchy over regular languages.(*)

But, is the star-height of a language computable ?

[Hashiguchi 81–88] Yes! (but very complicated)

[Kirsten 05] Yes, with a self contained and simpler proof.

Decidability of the boundedness of distance automata

(*) This is the restricted star height, in which intersection and negations are not allowed. Nobody knows if the hierarchy is strict in the unrestricted case.

The star-height problem

Def: A regular language is of star height k if it can be described by a rational expression using at most k nesting of Kleene stars, but not less.

[Eggan 63] The star-height forms a strict hierarchy over regular languages.(*)

But, is the star-height of a language computable ?

[Hashiguchi 81–88] Yes! (but very complicated)

[Kirsten 05] Yes, with a self contained and simpler proof.

Decidability of the boundedness of distance automata

boundedness of hB-automata

(*) This is the restricted star height, in which intersection and negations are not allowed. Nobody knows if the hierarchy is strict in the unrestricted case.

The star-height problem

Def: A regular language is of star height k if it can be described by a rational expression using at most k nesting of Kleene stars, but not less.

[Eggan 63] The star-height forms a strict hierarchy over regular languages.(*)

But, is the star-height of a language computable ?

[Hashiguchi 81–88] Yes! (but very complicated)

[Kirsten 05] Yes, with a self contained and simpler proof.

[C.&Löding 08] The star-height is decidable over trees.

Decidability of the boundedness of distance automata

boundedness of hB-automata

(*) This is the restricted star height, in which intersection and negations are not allowed. Nobody knows if the hierarchy is strict in the unrestricted case.

Star-height and boundedness

Star-height and boundedness

Fix a regular language L, and an integer k. Define:

$$f_{L,k} : A^* \to \mathbb{N} \cup \{\infty\}$$
$$u \mapsto \inf\{\texttt{size}(E) \mid \text{there exists an expression } E, \text{sh}(E) \leqslant k, u \in L_E \subseteq L\}$$

Star-height and boundedness

Fix a regular language L, and an integer k. Define:

$$f_{L,k} : A^* \to \mathbb{N} \cup \{\infty\}$$
$$u \mapsto \inf\{\texttt{size}(E) \mid \text{there exists an expression } E, \operatorname{sh}(E) \leqslant k, u \in L_E \subseteq L\}$$

Lemma: $f_{L,k}$ is bounded over L if and only if L has star-height at most k.

Star-height and boundedness

Fix a regular language L, and an integer k. Define:

$$f_{L,k} : A^* \to \mathbb{N} \cup \{\infty\}$$
$$u \mapsto \inf\{\texttt{size}(E) \mid \text{there exists an expression } E, \mathrm{sh}(E) \leqslant k, u \in L_E \subseteq L\}$$

Lemma: $f_{L,k}$ is bounded over L if and only if L has star-height at most k.

Proof:

Assume L is of star-height k.

Let E be the corresponding expression.

It is a witness that $f_{L,k}$ is bounded over L (by the size of E).

Star-height and boundedness

Fix a regular language L, and an integer k. Define:

$$f_{L,k} : A^* \to \mathbb{N} \cup \{\infty\}$$
$$u \mapsto \inf\{\texttt{size}(E) \mid \text{there exists an expression } E, \text{sh}(E) \leqslant k, u \in L_E \subseteq L\}$$

Lemma: $f_{L,k}$ is bounded over L if and only if L has star-height at most k.

Proof:

Assume L is of star-height k.

Let E be the corresponding expression.

It is a witness that $f_{L,k}$ is bounded over L (by the size of E).

Assume $f_{L,k}$ bounded over L by n. Define:

$$E_{L,n} = \sum_{\substack{\texttt{size}(E) \leqslant n \\ \texttt{sh}(E) \leqslant k \\ L_E \subseteq L}} E$$

Star-height and boundedness

Fix a regular language L, and an integer k. Define:

$$f_{L,k} : A^* \to \mathbb{N} \cup \{\infty\}$$
$$u \mapsto \inf\{\texttt{size}(E) \mid \text{there exists an expression } E, \text{sh}(E) \leqslant k, u \in L_E \subseteq L\}$$

Lemma: $f_{L,k}$ is bounded over L if and only if L has star-height at most k.

Proof:

Assume L is of star-height k.

Let E be the corresponding expression.

It is a witness that $f_{L,k}$ is bounded over L (by the size of E).

Assume $f_{L,k}$ bounded over L by n. Define:

$$E_{L,n} = \sum_{\substack{\texttt{size}(E) \leqslant n \\ \texttt{sh}(E) \leqslant k \\ L_E \subseteq L}} E$$

We have $L \subseteq \mathcal{L}(L_{E,n})$.

Star-height and boundedness

Star-height and boundedness

Fix a regular language L, and an integer k. Define:

$$f_{L,k} : A^* \to \mathbb{N} \cup \{\infty\}$$
$$u \mapsto \inf\{\texttt{size}(E) \mid \text{there exists an expression } E, \text{sh}(E) \leqslant k, u \in L_E \subseteq L\}$$

Star-height and boundedness

Fix a regular language L, and an integer k. Define:

$$f_{L,k} : A^* \to \mathbb{N} \cup \{\infty\}$$
$$u \mapsto \inf\{\texttt{size}(E) \mid \text{there exists an expression } E, \operatorname{sh}(E) \leqslant k, u \in L_E \subseteq L\}$$

Lemma: $f_{L,k}$ is bounded over L if and only if L has star-height at most k.

Star-height and boundedness

Fix a regular language L, and an integer k. Define:

$$f_{L,k} : A^* \to \mathbb{N} \cup \{\infty\}$$
$$u \mapsto \inf\{\texttt{size}(E) \mid \text{there exists an expression } E, \text{sh}(E) \leqslant k, u \in L_E \subseteq L\}$$

Lemma: $f_{L,k}$ is bounded over L if and only if L has star-height at most k.

[Kirsten05] There exists a hB-automaton computing $f_{L,k}$.

Star-height and boundedness

Fix a regular language L, and an integer k. Define:

$$f_{L,k} : A^* \to \mathbb{N} \cup \{\infty\}$$
$$u \mapsto \inf\{\texttt{size}(E) \mid \text{there exists an expression } E, \text{sh}(E) \leqslant k, u \in L_E \subseteq L\}$$

Lemma: $f_{L,k}$ is bounded over L if and only if L has star-height at most k.

[Kirsten05] There exists a hB-automaton computing $f_{L,k}$.

[Kirsten05] The star-height problem is decidable.

Star-height and boundedness

Fix a regular language L, and an integer k. Define:

$$f_{L,k} : A^* \to \mathbb{N} \cup \{\infty\}$$
$$u \mapsto \inf\{\texttt{size}(E) \mid \text{there exists an expression } E, \mathrm{sh}(E) \leqslant k, u \in L_E \subseteq L\}$$

Lemma: $f_{L,k}$ is bounded over L if and only if L has star-height at most k.

[Kirsten05] There exists a hB-automaton computing $f_{L,k}$.

[Kirsten05] The star-height problem is decidable.

This approach is generic. It works for trees, and for the Mostowski index of languages of infinite trees (open).

Star-height and boundedness

Fix a regular language L, and an integer k. Define:

$$f_{L,k} : A^* \to \mathbb{N} \cup \{\infty\}$$
$$u \mapsto \inf\{\texttt{size}(E) \mid \text{there exists an expression } E, \text{sh}(E) \leqslant k, u \in L_E \subseteq L\}$$

Lemma: $f_{L,k}$ is bounded over L if and only if L has star-height at most k.

[Kirsten05] There exists a hB-automaton computing $f_{L,k}$.

[Kirsten05] The star-height problem is decidable.

This approach is generic. It works for trees, and for the Mostowski index of languages of infinite trees (open).

It works also for separation.

Church synthesis with bounded error

Church synthesis with bounded error

[Church problem] Two players (controller and environment) alternatively choose a letter, producing in the end an infinite word. Controller wins if he can guarantee this word to be in a given ω-regular language.

- decide the winner,
- construct a finite automaton implementing the strategy.

Church synthesis with bounded error

[Church problem] Two players (controller and environment) alternatively choose a letter, producing in the end an infinite word. Controller wins if he can guarantee this word to be in a given ω-regular language.

- decide the winner,
- construct a finite automaton implementing the strategy.

[Büchi&Landweber69] This is doable.

Church synthesis with bounded error

[Church problem] Two players (controller and environment) alternatively choose a letter, producing in the end an infinite word. Controller wins if he can guarantee this word to be in a given ω-regular language.

- decide the winner,
- construct a finite automaton implementing the strategy.

[Büchi&Landweber69] This is doable.

[Rabinowich&Velner11] is it possible to decide if there exists n such that controller can guarantee being in L up to a change of n of his letters.

Church synthesis with bounded error

[Church problem] Two players (controller and environment) alternatively choose a letter, producing in the end an infinite word. Controller wins if he can guarantee this word to be in a given ω-regular language.

- decide the winner,
- construct a finite automaton implementing the strategy.

[Büchi&Landweber69] This is doable.

[Rabinowich&Velner11] is it possible to decide if there exists n such that controller can guarantee being in L up to a change of n of his letters.

[Regular cost functions] immediately yes.

Regular cost functions as languages

Motivation

Motivation

Working with regular cost functions usually involves the terms large and small used (with care) as Boolean predicates:

'f ≼ g if, whenever f(u) is large, g(u) is large'.

Motivation

Working with regular cost functions usually involves the terms large and small used (with care) as Boolean predicates:

'f ≼ g if, whenever f(u) is large, g(u) is large'.

Example: Given a tree, then

max(**maxdegree**,**height**) ≈ size

Motivation

Working with regular cost functions usually involves the terms large and small used (with care) as Boolean predicates:

'f ≼ g if, whenever f(u) is large, g(u) is large'.

Example: Given a tree, then

max(**maxdegree**,**height**) ≈ size

Proof:

If **maxdegree** is large, then there is a large number of nodes.
If **height** is large, then there is a large number of nodes.

Motivation

Working with regular cost functions usually involves the terms large and small used (with care) as Boolean predicates:

'f ≼ g if, whenever f(u) is large, g(u) is large'.

Example: Given a tree, then

max(**maxdegree**,**height**) ≈ size

Proof:

If **maxdegree** is large, then there is a large number of nodes.
If **height** is large, then there is a large number of nodes.

Conversely, assume both **maxdegree** and **height** would be small, then there would be a small number of nodes.

Motivation

Working with regular cost functions usually involves the terms large and small used (with care) as Boolean predicates:

'$f \preccurlyeq g$ if, whenever f(u) is large, g(u) is large'.

Example: Given a tree, then

max(**maxdegree**,**height**) $\approx$ size

Proof:
If **maxdegree** is large, then there is a large number of nodes.
If **height** is large, then there is a large number of nodes.

Conversely, assume both **maxdegree** and **height** would be small, then there would be a small number of nodes.

[Toruńczyk 11] One can identify regular cost functions with a language of profinite words, in a natural way.

Non-standard analysis (internal set theory)

Non-standard analysis (internal set theory)

[Robinson 66] …

Non-standard analysis (internal set theory)

[Robinson 66] …

[Nelson 77] Internal set theory (**IST**) a conservative extension of ZFC.

Non-standard analysis (internal set theory)

[Robinson 66] …

[Nelson 77] Internal set theory (**IST**) a conservative extension of ZFC.

This is an extension of set theory (ZFC) in which one can prove exactly the same statements of ZFC.

Non-standard analysis (internal set theory)

[Robinson 66] ...

[Nelson 77] Internal set theory (**IST**) a conservative extension of ZFC.

This is an extension of set theory (ZFC) in which one can prove exactly the same statements of ZFC.

All the language of ZFC is available.

Non-standard analysis (internal set theory)

[Robinson 66] ...

[Nelson 77] Internal set theory (**IST**) a conservative extension of ZFC.

This is an extension of set theory (ZFC) in which one can prove exactly the same statements of ZFC.

All the language of ZFC is available.

There is a new unary symbol $St(x)$ to be read 'x is standard'.

Non-standard analysis (internal set theory)

[Robinson 66] ...

[Nelson 77] Internal set theory (**IST**) a conservative extension of ZFC.

This is an extension of set theory (ZFC) in which one can prove exactly the same statements of ZFC.

All the language of ZFC is available.

There is a new unary symbol $St(x)$ to be read 'x is standard'.

All the proof arguments usable in ZFC are available.

Non-standard analysis (internal set theory)

[Robinson 66] ...

[Nelson 77] Internal set theory (**IST**) a conservative extension of ZFC.

This is an extension of set theory (ZFC) in which one can prove exactly the same statements of ZFC.

All the language of ZFC is available.

There is a new unary symbol $St(x)$ to be read 'x is standard'.

All the proof arguments usable in ZFC are available.

Three new axioms (schema) are available:

Non-standard analysis (internal set theory)

[Robinson 66] ...

[Nelson 77] Internal set theory (**IST**) a conservative extension of ZFC.

This is an extension of set theory (ZFC) in which one can prove exactly the same statements of ZFC.

All the language of ZFC is available.

There is a new unary symbol $St(x)$ to be read 'x is standard'.

All the proof arguments usable in ZFC are available.

Three new axioms (schema) are available:

- transfer: $\forall x \varphi$ iff $\forall^{St} x \varphi$

  ($\exists x \varphi$ iff $\exists^{St} x \varphi$)

φ has to be an internal formula,
i.e., no symbol $St(x)$.

Non-standard analysis (internal set theory)

[Robinson 66] ...

[Nelson 77] Internal set theory (**IST**) a conservative extension of ZFC.

This is an extension of set theory (ZFC) in which one can prove exactly the same statements of ZFC.

All the language of ZFC is available.

There is a new unary symbol $St(x)$ to be read 'x is standard'.

All the proof arguments usable in ZFC are available.

Three new axioms (schema) are available:

- **transfer**: $\forall x\varphi$ iff $\forall^{St} x\varphi$

  ($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has to be an internal formula, i.e., no symbol $St(x)$.

- **standardization**: given a standard set X and a formula φ, there exists a standard set Y such that for all standard elements x ,

$$x \in X \text{ and } x \models \varphi \text{ iff } x \in Y$$

Non-standard analysis (internal set theory)

[Robinson 66] ...

[Nelson 77] Internal set theory (**IST**) a conservative extension of ZFC.

This is an extension of set theory (ZFC) in which one can prove exactly the same statements of ZFC.

All the language of ZFC is available.

There is a new unary symbol $St(x)$ to be read 'x is standard'.

All the proof arguments usable in ZFC are available.

Three new axioms (schema) are available:

- transfer: $\forall x\varphi$ iff $\forall^{St} x\varphi$

  ($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has to be an internal formula, i.e., no symbol $St(x)$.

- standardization: given a standard set X and a formula φ, there exists a standard set Y such that for all standard elements x ,

$$x \in X \text{ and } x \models \varphi \text{ iff } x \in Y$$

- idealization: (simplified) there exists a non-standard integer.

More on IST

More on IST

- transfer: $\forall x\varphi$ iff $\forall^{St} x\varphi$

($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has be an internal formula,

i.e., no symbol $St(x)$.

More on IST

- transfer: $\forall x\varphi$ iff $\forall^{St} x\varphi$

($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has be an internal formula,

i.e., no symbol $St(x)$.

All objects definable in standard mathematics are standard.

More on IST

- transfer: $\forall x\varphi$ iff $\forall^{St} x\varphi$

($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has be an internal formula,
i.e., no symbol $St(x)$.

All objects definable in standard mathematics are standard.

This is the case for $0, 1, \pi, \mathbb{N}, \mathbb{R}, A^*$ ...

More on IST

- **transfer**: $\forall x\varphi$ iff $\forall^{St} x\varphi$

  ($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has be an internal formula,
i.e., no symbol $St(x)$.

All objects definable in standard mathematics are standard.

This is the case for $0, 1, \pi, \mathbb{N}, \mathbb{R}, A^*$ ...

Things definable by internal formulae with standard parameters are standard.

→ Successor, predecessor, square of standard integers are integers.

More on IST

- transfer: $\forall x\varphi$ iff $\forall^{St} x\varphi$

($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has be an internal formula,
i.e., no symbol $St(x)$.

All objects definable in standard mathematics are standard.

This is the case for $0, 1, \pi, \mathbb{N}, \mathbb{R}, A^* \ldots$

Things definable by internal formulae with standard parameters are standard.

→ Successor, predecessor, square of standard integers are integers.

Lemma: If m<n for n standard, then m is standard.

More on IST

- transfer: $\forall x\varphi$ iff $\forall^{St} x\varphi$

  ($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has be an internal formula,
i.e., no symbol $St(x)$.

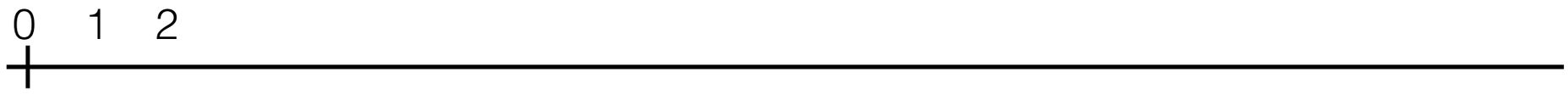
All objects definable in standard mathematics are standard.

This is the case for $0, 1, \pi, \mathbb{N}, \mathbb{R}, A^* \ldots$

Things definable by internal formulae with standard parameters are standard.

→ Successor, predecessor, square of standard integers are integers.

Lemma: If m<n for n standard, then m is standard.



More on IST

- transfer: $\forall x\varphi$ iff $\forall^{St} x\varphi$

  ($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has be an internal formula,
i.e., no symbol $St(x)$.

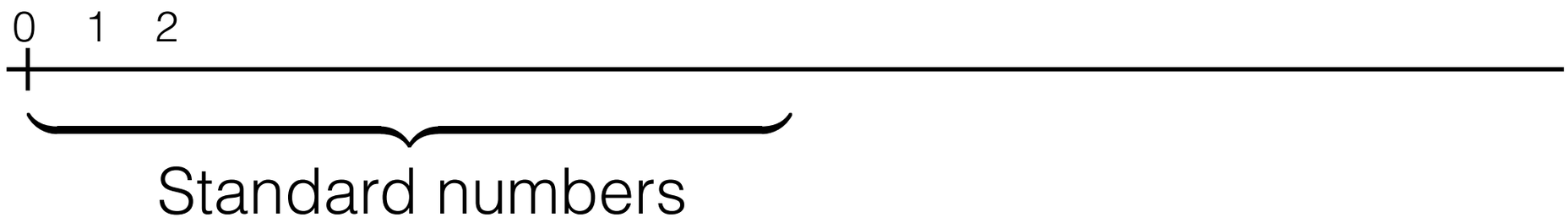
All objects definable in standard mathematics are standard.

This is the case for $0, 1, \pi, \mathbb{N}, \mathbb{R}, A^* \ldots$

Things definable by internal formulae with standard parameters are standard.

→ Successor, predecessor, square of standard integers are integers.

Lemma: If m<n for n standard, then m is standard.



More on IST

- transfer: $\forall x\varphi$ iff $\forall^{St} x\varphi$

  ($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has be an internal formula,
i.e., no symbol $St(x)$.

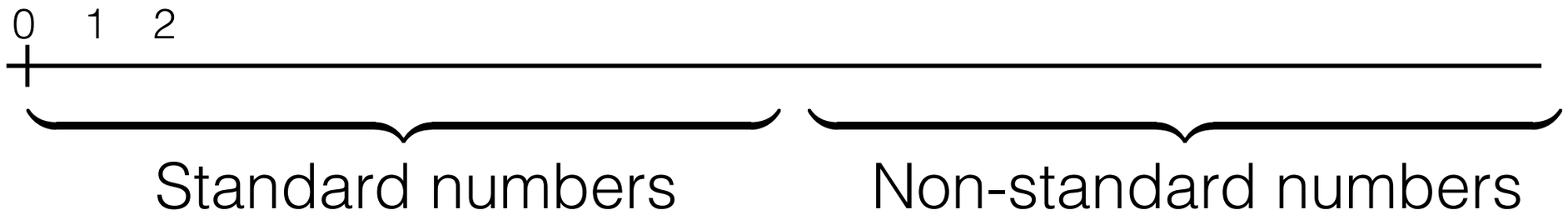
All objects definable in standard mathematics are standard.

This is the case for $0, 1, \pi, \mathbb{N}, \mathbb{R}, A^* \ldots$

Things definable by internal formulae with standard parameters are standard.

→ Successor, predecessor, square of standard integers are integers.

Lemma: If m<n for n standard, then m is standard.



More on IST

- transfer: $\forall x\varphi$ iff $\forall^{St} x\varphi$

  ($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has be an internal formula,
i.e., no symbol $St(x)$.

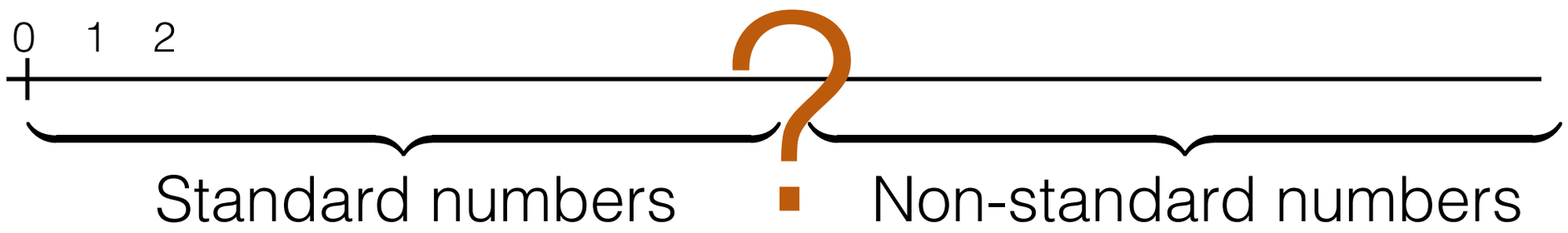
All objects definable in standard mathematics are standard.

This is the case for $0, 1, \pi, \mathbb{N}, \mathbb{R}, A^* \ldots$

Things definable by internal formulae with standard parameters are standard.

→ Successor, predecessor, square of standard integers are integers.

Lemma: If m<n for n standard, then m is standard.



More on IST

- transfer: $\forall x\varphi$ iff $\forall^{St} x\varphi$

  ($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has be an internal formula,
i.e., no symbol $St(x)$.

All objects definable in standard mathematics are standard.

This is the case for $0, 1, \pi, \mathbb{N}, \mathbb{R}, A^* \ldots$

Things definable by internal formulae with standard parameters are standard.

→ Successor, predecessor, square of standard integers are integers.

Lemma: If m<n for n standard, then m is standard.

The set of standard integers does not exist!

More on IST

- **transfer**: $\forall x\varphi$ iff $\forall^{St} x\varphi$

  ($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has be an internal formula,
i.e., no symbol $St(x)$.

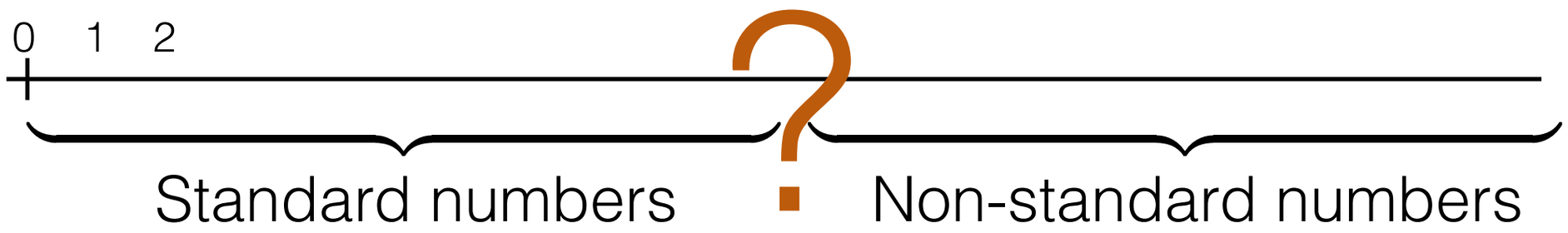
All objects definable in standard mathematics are standard.

This is the case for $0, 1, \pi, \mathbb{N}, \mathbb{R}, A^* \ldots$

Things definable by internal formulae with standard parameters are standard.

→ Successor, predecessor, square of standard integers are integers.

Lemma: If m<n for n standard, then m is standard.



The set of standard integers does not exist!

The comprehension schema

$\{x \in A \mid \varphi\}$

More on IST

- **transfer**: $\forall x\varphi$ iff $\forall^{St} x\varphi$

  ($\exists x\varphi$ iff $\exists^{St} x\varphi$)

φ has be an internal formula,
i.e., no symbol $St(x)$.

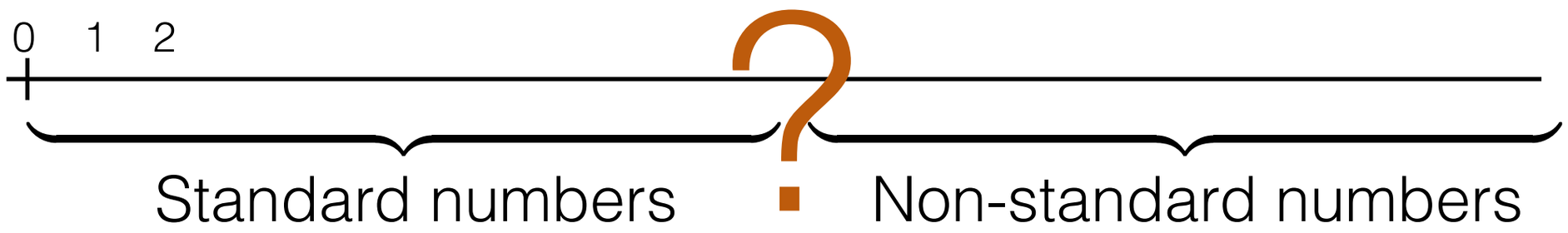
All objects definable in standard mathematics are standard.

This is the case for $0, 1, \pi, \mathbb{N}, \mathbb{R}, A^* \ldots$

Things definable by internal formulae with standard parameters are standard.

→ Successor, predecessor, square of standard integers are integers.

Lemma: If m<n for n standard, then m is standard.



The set of standard integers does not exist!

The comprehension schema

$$\{x \in A \mid \varphi\}$$

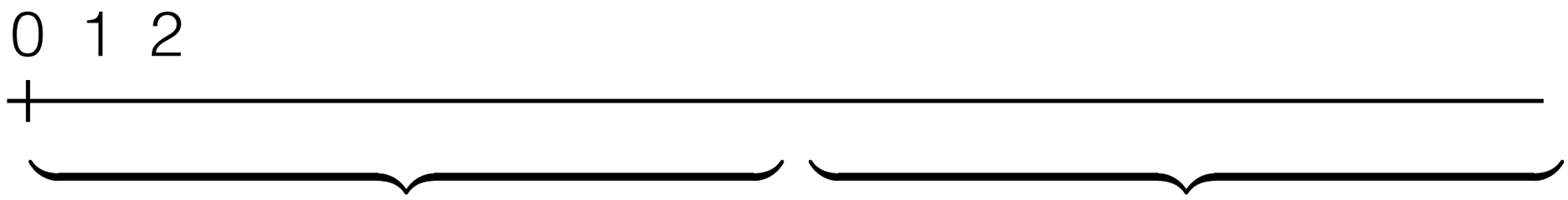
is only valid for internal formulae!!!!!

Regular cost functions in IST

Regular cost functions in IST

Regular cost functions in IST

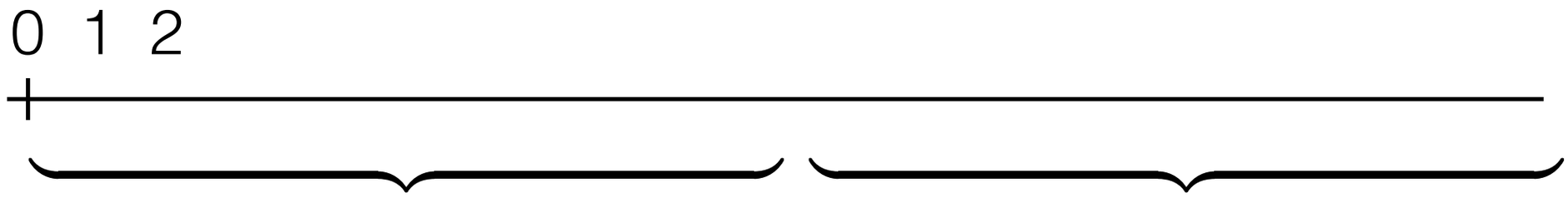
Regular cost functions in IST



Given a standard function $f : A^* \to \mathbb{N} \cup \{\infty\}$, define:

$$L_f = \{u \in A^* \mid \underbrace{\texttt{St}(f(u)) \wedge f(u) \neq \infty}_{f(u) \text{ is small}}\}^{external}$$

Regular cost functions in IST

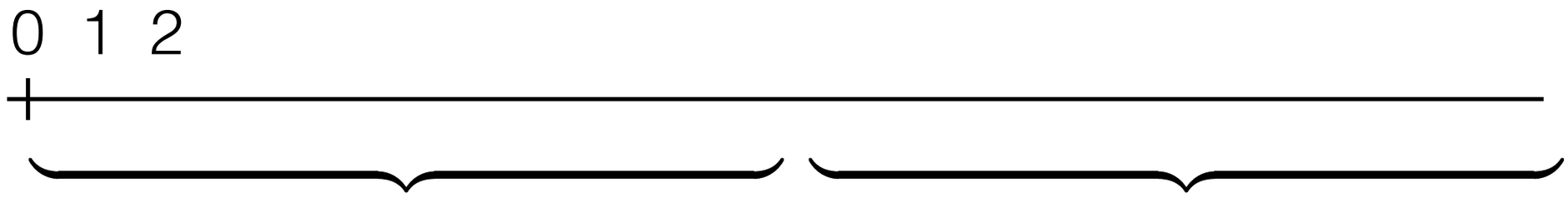


Given a standard function $f : A^* \to \mathbb{N} \cup \{\infty\}$, define:

$$L_f = \{u \in A^* \mid \underbrace{\mathtt{St}(f(u)) \wedge f(u) \neq \infty}_{f(u) \text{ is small}}\}^{external}$$

Then $f \preccurlyeq g$ if and only if $L_g \subseteq L_f$.

Regular cost functions in IST



Given a standard function $f : A^* \to \mathbb{N} \cup \{\infty\}$, define:

$$L_f = \{u \in A^* \mid \underbrace{\mathtt{St}(f(u)) \wedge f(u) \neq \infty}_{f(u) \text{ is small}}\}^{external}$$

Then $f \preccurlyeq g$ if and only if $L_g \subseteq L_f$.

Under this view, the theory of regular cost functions can be recast into a theory of regular external languages.

Regular external languages

Regular external languages

B-rational expression

rational expression + $L^{\mathtt{St}}$

$(a^{\mathtt{St}}b)^*a^{\mathtt{St}}$

Regular external languages

B-rational expression

rational expression + $L^{\mathtt{St}}$

$(a^{\mathtt{St}}b)^*a^{\mathtt{St}}$

S-rational expression

rational expression + $L^{\neg\mathtt{St}}$

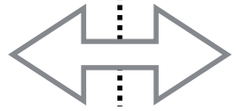
$(a^*b)^*a^{\neg\mathtt{St}}(ba^*)^*$

Regular external languages

B-rational expression

rational expression + $L^{\mathtt{St}}$

$(a^{\mathtt{St}}b)^*a^{\mathtt{St}}$



complement

S-rational expression

rational expression + $L^{\neg\mathtt{St}}$

$(a^*b)^*a^{\neg\mathtt{St}}(ba^*)^*$

Regular external languages

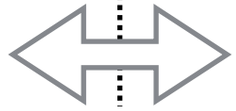
B-rational expression

rational expression + $L^{\mathtt{St}}$

$(a^{\mathtt{St}}b)^*a^{\mathtt{St}}$

B-automata

ND automata + counters
with standardness constraints



complement

S-rational expression

rational expression + $L^{\neg\mathtt{St}}$

$(a^*b)^*a^{\neg\mathtt{St}}(ba^*)^*$

Regular external languages

B-rational expression

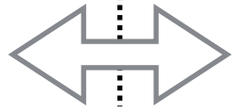
rational expression + $L^{\mathtt{St}}$

$(a^{\mathtt{St}}b)^*a^{\mathtt{St}}$

S-rational expression

rational expression + $L^{\neg\mathtt{St}}$

$(a^*b)^*a^{\neg\mathtt{St}}(ba^*)^*$



complement

B-automata

ND automata + counters
with standardness constraints

S-automata

ND automata + counters
with non-standardness constraints

Regular external languages

B-rational expression

rational expression + $L^{\mathtt{St}}$

$(a^{\mathtt{St}}b)^*a^{\mathtt{St}}$

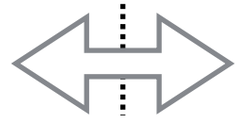
B-automata

ND automata + counters
with standardness constraints

cost MSO

MSO + $\mathtt{St}(|X|)$ (positively)

‘there is a maximal set of consecutive a’s of standard size’



complement

S-rational expression

rational expression + $L^{\neg\mathtt{St}}$

$(a^*b)^*a^{\neg\mathtt{St}}(ba^*)^*$

S-automata

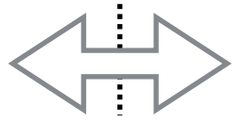
ND automata + counters
with non-standardness constraints

Regular external languages

B-rational expression
rational expression + $L^{\mathtt{St}}$
$(a^{\mathtt{St}}b)^*a^{\mathtt{St}}$

B-automata
ND automata + counters
with standardness constraints

cost MSO
MSO + $\mathtt{St}(|X|)$ (positively)
‘there is a maximal set of consecutive a’s of standard size’



complement

S-rational expression
rational expression + $L^{\neg\mathtt{St}}$
$(a^*b)^*a^{\neg\mathtt{St}}(ba^*)^*$

S-automata
ND automata + counters
with non-standardness constraints

cost MSO’
MSO + $\neg\mathtt{St}(|X|)$ (positively)
‘all maximal sets of consecutive a’s have non-standard size’

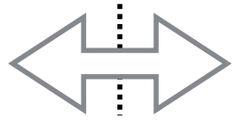
Regular external languages

B-rational expression
rational expression + $L^{\mathtt{St}}$
$(a^{\mathtt{St}}b)^*a^{\mathtt{St}}$

B-automata
ND automata + counters
with standardness constraints

cost MSO
MSO + $\mathtt{St}(|X|)$ (positively)
‘there is a maximal set of consecutive a’s of standard size’

stabilisation monoids
ordered monoid + stabilisation
+ upward closed accepting set



complement

S-rational expression
rational expression + $L^{\neg\mathtt{St}}$
$(a^*b)^*a^{\neg\mathtt{St}}(ba^*)^*$

S-automata
ND automata + counters
with non-standardness constraints

cost MSO’
MSO + $\neg\mathtt{St}(|X|)$ (positively)
‘all maximal sets of consecutive a’s have non-standard size’

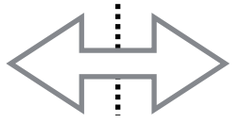
Regular external languages

B-rational expression
rational expression + $L^{\mathtt{St}}$
$(a^{\mathtt{St}}b)^*a^{\mathtt{St}}$

B-automata
ND automata + counters
with standardness constraints

cost MSO
MSO + $\mathtt{St}(|X|)$ (positively)
‘there is a maximal set of consecutive a’s of standard size’

stabilisation monoids
ordered monoid + stabilisation
+ upward closed accepting set



complement

S-rational expression
rational expression + $L^{\neg\mathtt{St}}$
$(a^*b)^*a^{\neg\mathtt{St}}(ba^*)^*$

S-automata
ND automata + counters
with non-standardness constraints

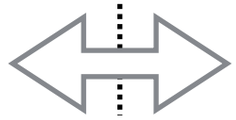
cost MSO’
MSO + $\neg\mathtt{St}(|X|)$ (positively)
‘all maximal sets of consecutive a’s have non-standard size’

stabilisation monoids
ordered monoid + stabilisation
+ downward closed accepting set

Regular external languages

B-rational expression
rational expression + $L^{\mathtt{St}}$
$(a^{\mathtt{St}}b)^*a^{\mathtt{St}}$

S-rational expression
rational expression + $L^{\neg\mathtt{St}}$
$(a^*b)^*a^{\neg\mathtt{St}}(ba^*)^*$



complement

B-automata
ND automata + counters
with standardness constraints

S-automata
ND automata + counters
with non-standardness constraints

cost MSO
MSO + $\mathtt{St}(|X|)$ (positively)
‘there is a maximal set of consecutive a’s of standard size’

cost MSO’
MSO + $\neg\mathtt{St}(|X|)$ (positively)
‘all maximal sets of consecutive a’s have non-standard size’

stabilisation monoids
ordered monoid + stabilisation
+ upward closed accepting set

stabilisation monoids
ordered monoid + stabilisation
+ downward closed accepting set

Emptiness of Boolean combinations is decidable.

Conclusion

Regular cost functions are functions computed by variants of tropical automata modulo a ‘boundedness equivalence relation’.

The most important results that hold for regular languages are still valid.

It can be used to decide several problems involving the existence of bounds, be it explicitly or implicitly.

Thanks to non-standard analysis, the results can be seen as a language theory over less usual words, in the same spirit as infinite words/words of countable length.