Deciding complexity of languages via games

Michał Skrzypczak

University of Warsaw

Part 1

Regular languages of things

Things

Things labelled by an alphabet A

Things labelled by an alphabet A

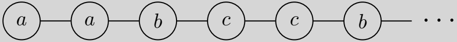
Finite / infinite **words**:

Things labelled by an alphabet A

Finite / infinite **words**: (seen as strings, terms, and logical structures)

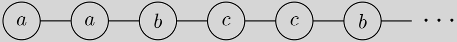
Things labelled by an alphabet A

Finite / infinite **words**: (seen as strings, terms, and logical structures)



Things labelled by an alphabet A

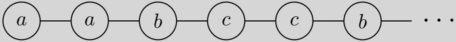
Finite / infinite **words**: (seen as strings, terms, and logical structures)



Signature: $s(x)$, $\leqslant$, $a(x)$ for $a \in A$

Things labelled by an alphabet A

Finite / infinite **words**: (seen as strings, terms, and logical structures)

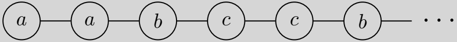


Signature: $s(x)$, $\leqslant$, $a(x)$ for $a \in A$

Finite / infinite **trees**:

Things labelled by an alphabet A

Finite / infinite **words**: (seen as strings, terms, and logical structures)

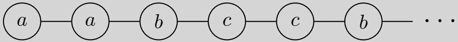


Signature: $s(x)$, $\leqslant$, $a(x)$ for $a \in A$

Finite / infinite **trees**: (seen as XML, terms, and logical structures)

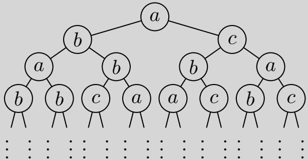
Things labelled by an alphabet A

Finite / infinite **words**: (seen as strings, terms, and logical structures)



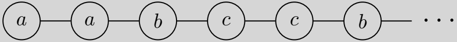
Signature: $s(x)$, $\leqslant$, $a(x)$ for $a \in A$

Finite / infinite **trees**: (seen as XML, terms, and logical structures)



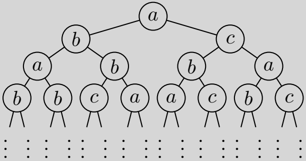
Things labelled by an alphabet A

Finite / infinite **words**: (seen as strings, terms, and logical structures)



Signature: $s(x)$, $\leqslant$, $a(x)$ for $a \in A$

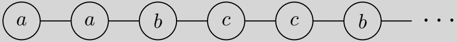
Finite / infinite **trees**: (seen as XML, terms, and logical structures)



Signature: $s_{\mathtt{L}}(x)$, $s_{\mathtt{R}}(x)$, $\preceq, \leqslant_{\mathrm{lex}}$, $a(x)$ for $a \in A$

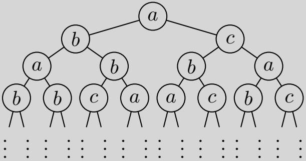
Things labelled by an alphabet A

Finite / infinite **words**: (seen as strings, terms, and logical structures)



Signature: $s(x)$, $\leqslant$, $a(x)$ for $a \in A$

Finite / infinite **trees**: (seen as XML, terms, and logical structures)

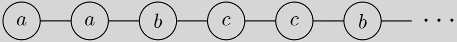


Signature: $s_{\mathtt{L}}(x)$, $s_{\mathtt{R}}(x)$, $\preceq, \leqslant_{\mathrm{lex}}$, $a(x)$ for $a \in A$

Monadic Second-order Logic:

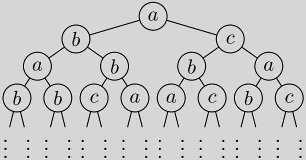
Things labelled by an alphabet A

Finite / infinite **words**: (seen as strings, terms, and logical structures)



Signature: $s(x)$, $\leqslant$, $a(x)$ for $a \in A$

Finite / infinite **trees**: (seen as XML, terms, and logical structures)

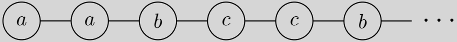


Signature: $s_{\mathtt{L}}(x)$, $s_{\mathtt{R}}(x)$, $\preceq, \leqslant_{\mathrm{lex}}$, $a(x)$ for $a \in A$

Monadic Second-order Logic: $\exists_x \quad \varphi \vee \psi \quad \neg\psi$

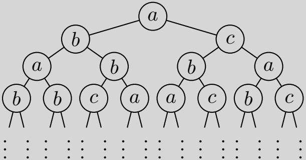
Things labelled by an alphabet A

Finite / infinite **words**: (seen as strings, terms, and logical structures)



Signature: $s(x)$, $\leqslant$, $a(x)$ for $a \in A$

Finite / infinite **trees**: (seen as XML, terms, and logical structures)



Signature: $s_{\mathtt{L}}(x)$, $s_{\mathtt{R}}(x)$, $\preceq, \leqslant_{\text{lex}}$, $a(x)$ for $a \in A$

Monadic Second-order Logic: $\exists_x \quad \varphi \vee \psi \quad \neg\psi \quad \exists_X \quad x \in X$

Regular languages of things

Regular languages of things

Fin. words

Regular languages of things

	logic
Fin. words	MSO

Regular languages of things

logic

Fin. words MSO

$$\mathrm{L}(\varphi) = \{w \mid w \models \varphi\}$$

Regular languages of things

	logic
Fin. words	MSO

Regular languages of things

	logic	**automata**
Fin. words	MSO	DFA

Regular languages of things

	logic	automata
Fin. words	MSO	DFA

$$\mathrm{L}(\mathcal{A}) = \{w \mid \mathcal{A} \text{ accepts } w\}$$

Regular languages of things

	logic	automata
Fin. words	MSO	DFA

Regular languages of things

	logic	automata	algebra
Fin. words	MSO	DFA	monoids

Regular languages of things

	logic	automata	algebra
Fin. words	MSO	DFA	monoids
			$h\colon A^* \to M$, $F \subseteq M$

Regular languages of things

	logic	automata	algebra
Fin. words	MSO	DFA	monoids

$$h\colon A^* \to M,\ F \subseteq M$$
$$\mathrm{L}(h) = h^{-1}(F)$$

Regular languages of things

	logic	automata	algebra
Fin. words	MSO	DFA	monoids

Regular languages of things

	logic	automata	algebra	expressions
Fin. words	MSO	DFA	monoids	regexp

Regular languages of things

	logic	automata	algebra	expressions
Fin. words	MSO	DFA	monoids	regexp
				$\mathrm{L}(e) \cong e$

Regular languages of things

	logic	automata	algebra	expressions
Fin. words	MSO	DFA	monoids	regexp

Regular languages of things

	logic	automata	algebra	expressions	⋯
Fin. words	MSO	DFA	monoids	regexp	

Regular languages of things

	logic	automata	algebra	expressions	⋯
Fin. words	MSO	DFA	monoids	regexp	
Inf. words					

Regular languages of things

	logic	automata	algebra	expressions	...
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO				

Regular languages of things

	logic	automata	algebra	expressions	···
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity			

Regular languages of things

	logic	automata	algebra	expressions	⋯
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.		

Regular languages of things

	logic	automata	algebra	expressions	⋯
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	

Regular languages of things

	logic	**automata**	**algebra**	**expressions**	···
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees					

Regular languages of things

	logic	automata	algebra	expressions	···
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO				

Regular languages of things

	logic	automata	algebra	expressions	···
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up			

Regular languages of things

	logic	automata	algebra	expressions	⋯
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.		

Regular languages of things

	logic	automata	algebra	expressions	···
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	

Regular languages of things

	logic	automata	algebra	expressions	⋯
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Inf. trees					

Regular languages of things

	logic	automata	algebra	expressions	···
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Inf. trees	MSO				

Regular languages of things

	logic	automata	algebra	expressions	···
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Inf. trees	MSO	nondet. parity			

Regular languages of things

	logic	automata	algebra	expressions	···
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Inf. trees	MSO	nondet. parity	ω-clones		

Regular languages of things

	logic	automata	algebra	expressions	···
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Inf. trees	MSO	nondet. parity	ω-clones	—	

Regular languages of things

	logic	automata	algebra	expressions	···
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Inf. trees	MSO	nondet. parity	ω-clones	—	

Regular languages of things

	logic	automata	algebra	expressions	...
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Inf. trees	MSO	nondet. parity	ω-clones	—	

$\rightsquigarrow$ MSO is universal

Regular languages of things

	logic	automata	algebra	expressions	...
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Inf. trees	MSO	nondet. parity	ω-clones	—	

⇝ MSO is universal

⇝ complicated structures require complicated devices

Regular languages of things

	logic	automata	algebra	expressions	...
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Inf. trees	MSO	nondet. parity	ω-clones	—	

$\rightsquigarrow$ MSO is universal

$\rightsquigarrow$ complicated structures require complicated devices

$\rightsquigarrow$ infinite trees inherently require non-determinism

Working with languages

Working with languages

Juggling representations:

Working with languages

Juggling representations:

Input: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for a given language L

Working with languages

Juggling representations:

Input: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / $\ldots$ for a given language L

Output: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / $\ldots$ for L

Working with languages

Juggling representations:

Input: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for a given language L

Output: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for L

Effective operations:

Working with languages

Juggling representations:

Input: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for a given language L

Output: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for L

Effective operations:

Input: L and M

Working with languages

Juggling representations:

Input: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for a given language L

Output: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for L

Effective operations:

Input: L and M

Output: $L \cup M$, $L \cap M$, $L \setminus M$, L^{c} (also $h(L)$, ...)

Working with languages

Juggling representations:

Input: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for a given language L

Output: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for L

Effective operations:

Input: L and M

Output: $L \cup M$, $L \cap M$, $L \setminus M$, L^{c} (also $h(L)$, ...)

Deciding properties:

Working with languages

Juggling representations:

Input: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for a given language L

Output: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for L

Effective operations:

Input: L and M

Output: $L \cup M$, $L \cap M$, $L \setminus M$, L^{c} (also $h(L)$, ...)

Deciding properties:

Input: L

Working with languages

Juggling representations:

Input: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for a given language L

Output: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for L

Effective operations:

Input: L and M

Output: $L \cup M$, $L \cap M$, $L \setminus M$, L^{c} (also $h(L)$, ...)

Deciding properties:

Input: L

Output: Is L: non-empty, infinite, uncountable, measure 1, ...

Working with languages

Juggling representations:

Input: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for a given language L

Output: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for L

Effective operations:

Input: L and M

Output: $L \cup M$, $L \cap M$, $L \backslash M$, L^{c} (also $h(L)$, ...)

Deciding properties:

Input: L

Output: Is L: non-empty, infinite, uncountable, measure 1, ...

Effective characterisations:

Working with languages

Juggling representations:

Input: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for a given language L

Output: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for L

Effective operations:

Input: L and M

Output: $L \cup M$, $L \cap M$, $L \setminus M$, L^{c} (also $h(L)$, ...)

Deciding properties:

Input: L

Output: Is L: non-empty, infinite, uncountable, measure 1, ...

Effective characterisations:

Input: L

Working with languages

Juggling representations:

Input: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for a given language L

Output: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for L

Effective operations:

Input: L and M

Output: $L \cup M$, $L \cap M$, $L \setminus M$, L^{c} (also $h(L)$, ...)

Deciding properties:

Input: L

Output: Is L: non-empty, infinite, uncountable, measure 1, ...

Effective characterisations:

Input: L

Output: Is L definable in a simple way (e.g. in $\textsc{fo}$)

Working with languages

Juggling representations:

Input: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for a given language L

Output: $\mathrm{L}(\varphi)$ / $\mathrm{L}(\mathcal{A})$ / $\mathrm{L}(h)$ / ... for L

Effective operations:

Input: L and M

Output: $L \cup M$, $L \cap M$, $L \setminus M$, L^{c} (also $h(L)$, ...)

Deciding properties:

Input: L

Output: Is L: non-empty, infinite, uncountable, measure 1, ...

Effective characterisations:

Input: L

Output: Is L definable in a simple way (e.g. in FO)

Part 2

Effective characterisations by patterns

Given $L \subseteq A^*$, is L definable in FO?

Given $L \subseteq A^*$, is L definable in FO?

iff $L = \mathrm{L}(e)$ for a star-free regexp: $e ::= \varnothing \mid A \mid ee \mid e \cup e \mid \sim e$

Given $L \subseteq A^*$, is L definable in FO?

iff $L = \mathrm{L}(e)$ for a star-free regexp: $\quad e ::= \varnothing \mid A \mid ee \mid e{\cup}e \mid {\sim}e$

$$A^* = {\sim}\varnothing$$
$$(ab)^* = {\sim}\big[\; bA^* \cup A^*a \cup A^*aaA^* \cup A^*bbA^* \big]$$

Given $L \subseteq A^*$, is L definable in FO?

iff $L = \mathrm{L}(e)$ for a star-free regexp: $e ::= \varnothing \mid A \mid ee \mid e \cup e \mid \sim e$

Given $L \subseteq A^*$, is L definable in FO?

iff $L = \mathrm{L}(e)$ for a star-free regexp: $e ::= \varnothing \mid A \mid ee \mid e \cup e \mid \sim e$

By Ehrenfeucht-Fraïssé

$$a^{2^k} \equiv_k a^{2^k+1}$$

Given $L \subseteq A^*$, is L definable in FO?

iff $L = \mathrm{L}(e)$ for a star-free regexp: $e ::= \varnothing \mid A \mid ee \mid e{\cup}e \mid {\sim}e$

By Ehrenfeucht-Fraïssé

$$a^{2^k} \equiv_k a^{2^k+1}$$

$$\Big[\text{If } \mathrm{QD}(\varphi) = k \text{ then } a^{2^k} \models \varphi \text{ iff } a^{2^k+1} \models \varphi\Big]$$

Given $L \subseteq A^*$, is L definable in FO?

iff $L = \mathrm{L}(e)$ for a star-free regexp: $e ::= \varnothing \mid A \mid ee \mid e{\cup}e \mid {\sim}e$

By Ehrenfeucht-Fraïssé

$$a^{2^k} \equiv_k a^{2^k+1}$$

Given $L \subseteq A^*$, is L definable in FO?

iff $L = \mathrm{L}(e)$ for a star-free regexp: $e ::= \varnothing \mid A \mid ee \mid e{\cup}e \mid {\sim}e$

By Ehrenfeucht-Fraïssé

$$a^{2^k} \equiv_k a^{2^k+1}$$

Theorem (Schutzenberger [1965]; McNaughton, Papert [1971])

TFAE for $L \subseteq A^*$:

Given $L \subseteq A^*$, is L definable in FO?

iff $L = \mathrm{L}(e)$ for a star-free regexp: $e ::= \varnothing \mid A \mid ee \mid e{\cup}e \mid {\sim}e$

By Ehrenfeucht-Fraïssé

$$a^{2^k} \equiv_k a^{2^k+1}$$

Theorem (Schutzenberger [1965]; McNaughton, Papert [1971])

TFAE for $L \subseteq A^*$:

– L is definable in FO

Given $L \subseteq A^*$, is L definable in FO?

iff $L = \mathrm{L}(e)$ for a star-free regexp: $e ::= \varnothing \mid A \mid ee \mid e{\cup}e \mid {\sim}e$

By Ehrenfeucht-Fraïssé

$$a^{2^k} \equiv_k a^{2^k+1}$$

Theorem (Schutzenberger [1965]; McNaughton, Papert [1971])

TFAE for $L \subseteq A^*$:

– L is definable in FO

– the minimal automaton $\mathcal{A}_L$ for L is counter-free

Given $L \subseteq A^*$, is L definable in FO?

iff $L = \mathrm{L}(e)$ for a star-free regexp: $e ::= \varnothing \mid A \mid ee \mid e \cup e \mid \sim e$

By Ehrenfeucht-Fraïssé

$$a^{2^k} \equiv_k a^{2^k+1}$$

Theorem (Schutzenberger [1965]; McNaughton, Papert [1971])

TFAE for $L \subseteq A^*$:

- L is definable in FO
- the minimal automaton $\mathcal{A}_L$ for L is counter-free
- the syntactic monoid M_L for L is group-free

Given $L \subseteq A^*$, is L definable in FO?

iff $L = \mathrm{L}(e)$ for a star-free regexp: $e ::= \varnothing \mid A \mid ee \mid e \cup e \mid \sim e$

By Ehrenfeucht-Fraïssé

$$a^{2^k} \equiv_k a^{2^k+1}$$

Theorem (Schutzenberger [1965]; McNaughton, Papert [1971])

TFAE for $L \subseteq A^*$:

– L is definable in FO

– the minimal automaton $\mathcal{A}_L$ for L is counter-free

– the syntactic monoid M_L for L is group-free

– the syntactic monoid M_L for L satisfies $s^\sharp = s \cdot s^\sharp$

Given $L \subseteq A^*$, is L definable in FO?

iff $L = \mathrm{L}(e)$ for a star-free regexp: $e ::= \varnothing \mid A \mid ee \mid e \cup e \mid \sim e$

By Ehrenfeucht-Fraïssé

$$a^{2^k} \equiv_k a^{2^k+1}$$

Theorem (Schutzenberger [1965]; McNaughton, Papert [1971])

TFAE for $L \subseteq A^*$:

- L is definable in FO
- the minimal automaton $\mathcal{A}_L$ for L is counter-free
- the syntactic monoid M_L for L is group-free
- the syntactic monoid M_L for L satisfies $s^\sharp = s \cdot s^\sharp$

"pattern method for rigid representations"

L is definable in FO

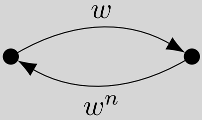
iff

the minimal automaton $\mathcal{A}_L$ for L is counter-free

L is definable in FO

iff

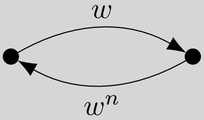
the minimal automaton $\mathcal{A}_L$ for L is counter-free



L is definable in FO

iff

the minimal automaton $\mathcal{A}_L$ for L is counter-free

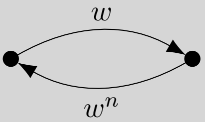


1. Let $L = \mathrm{L}(\mathcal{A})$ for a counter-free $\mathcal{A}$

L is definable in FO

iff

the minimal automaton $\mathcal{A}_L$ for L is counter-free



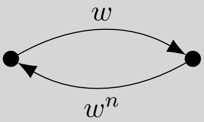
1. Let $L = \mathrm{L}(\mathcal{A})$ for a counter-free $\mathcal{A}$

$\rightsquigarrow$ write φ in FO such that $L = \mathrm{L}(\varphi)$

L is definable in FO

iff

the minimal automaton $\mathcal{A}_L$ for L is counter-free



1. Let $L = \mathrm{L}(\mathcal{A})$ for a counter-free $\mathcal{A}$

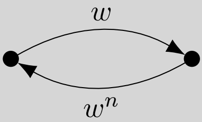
$\rightsquigarrow$ write φ in FO such that $L = \mathrm{L}(\varphi)$

$\rightsquigarrow$ L is definable in FO

L is definable in FO

iff

the minimal automaton $\mathcal{A}_L$ for L is counter-free



1. Let $L = \mathrm{L}(\mathcal{A})$ for a counter-free $\mathcal{A}$

⇝ write φ in FO such that $L = \mathrm{L}(\varphi)$

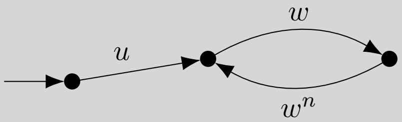
⇝ L is definable in FO

2. Let $\mathcal{A}_L$ contain a counter

L is definable in FO

iff

the minimal automaton $\mathcal{A}_L$ for L is counter-free



1. Let $L = \mathrm{L}(\mathcal{A})$ for a counter-free $\mathcal{A}$

$\rightsquigarrow$ write φ in FO such that $L = \mathrm{L}(\varphi)$

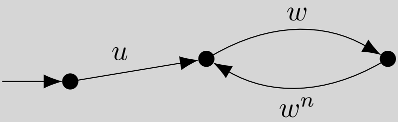
$\rightsquigarrow$ L is definable in FO

2. Let $\mathcal{A}_L$ contain a counter

L is definable in FO

iff

the minimal automaton $\mathcal{A}_L$ for L is counter-free



1. Let $L = \mathrm{L}(\mathcal{A})$ for a counter-free $\mathcal{A}$

⇝ write φ in FO such that $L = \mathrm{L}(\varphi)$

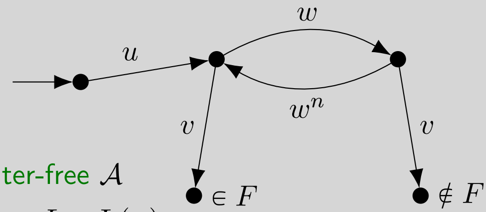
⇝ L is definable in FO

2. Let $\mathcal{A}_L$ contain a counter ($\mathcal{A}_L$ is minimal!)

L is definable in FO

iff

the minimal automaton $\mathcal{A}_L$ for L is counter-free



1. Let $L = \mathrm{L}(\mathcal{A})$ for a counter-free $\mathcal{A}$

⇝ write φ in FO such that $L = \mathrm{L}(\varphi)$

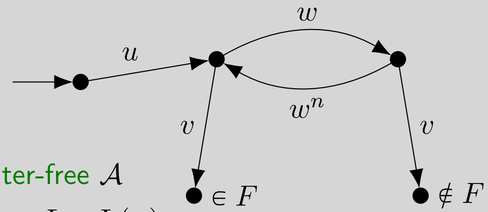
⇝ L is definable in FO

2. Let $\mathcal{A}_L$ contain a counter ($\mathcal{A}_L$ is minimal!)

L is definable in FO

iff

the minimal automaton $\mathcal{A}_L$ for L is counter-free



1. Let $L = \mathrm{L}(\mathcal{A})$ for a counter-free $\mathcal{A}$

⇝ write φ in FO such that $L = \mathrm{L}(\varphi)$

⇝ L is definable in FO

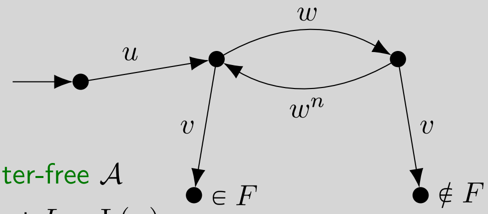
2. Let $\mathcal{A}_L$ contain a counter ($\mathcal{A}_L$ is minimal!)

⇝ $uw^{(n+1)\cdot 2^k}v \in L$ and $uw^{(n+1)\cdot\left(2^k+1\right)}v \notin L$

L is definable in FO

iff

the minimal automaton $\mathcal{A}_L$ for L is counter-free



1. Let $L = \mathrm{L}(\mathcal{A})$ for a counter-free $\mathcal{A}$

$\rightsquigarrow$ write φ in FO such that $L = \mathrm{L}(\varphi)$

$\rightsquigarrow$ L is definable in FO

2. Let $\mathcal{A}_L$ contain a counter ($\mathcal{A}_L$ is minimal!)

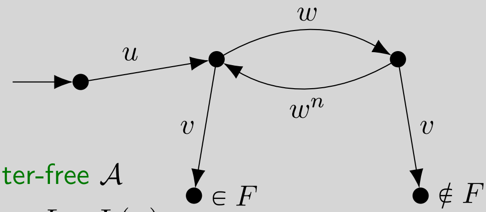
$\rightsquigarrow$ $uw^{(n+1)\cdot 2^k} v \in L$ and $uw^{(n+1)\cdot \left(2^k+1\right)} v \notin L$

$\equiv_k$

L is definable in FO

iff

the minimal automaton $\mathcal{A}_L$ for L is counter-free



1. Let $L = \mathrm{L}(\mathcal{A})$ for a counter-free $\mathcal{A}$

$\rightsquigarrow$ write φ in FO such that $L = \mathrm{L}(\varphi)$

$\rightsquigarrow$ L is definable in FO

2. Let $\mathcal{A}_L$ contain a counter ($\mathcal{A}_L$ is minimal!)

$\rightsquigarrow$ $uw^{(n+1)\cdot 2^k} v \in L$ and $uw^{(n+1)\cdot \left(2^k+1\right)} v \notin L$

$\equiv_k$

$\rightsquigarrow$ $L \notin \mathrm{FO}_k$ $\rightsquigarrow$ $L \notin \mathrm{FO}$

Proof strategy overview

Proof strategy overview

1. Take a rigid representation of L

Proof strategy overview

1. Take a rigid representation of L

(e.g. minimal automaton, syntactic algebra, ...)

Proof strategy overview

1. Take a rigid representation of L

(e.g. minimal automaton, syntactic algebra, ...)

2. Look for a **pattern**

Proof strategy overview

1. Take a rigid representation of L

(e.g. minimal automaton, syntactic algebra, ...)

2. Look for a **pattern**

(e.g. graph gadget, violation of equation, ...)

Proof strategy overview

1. Take a rigid representation of L

(e.g. minimal automaton, syntactic algebra, ...)

2. Look for a **pattern**

(e.g. graph gadget, violation of equation, ...)

2.a If no **pattern** found, construct a simple representation of L

Proof strategy overview

1. Take a rigid representation of L

(e.g. minimal automaton, syntactic algebra, ...)

2. Look for a **pattern**

(e.g. graph gadget, violation of equation, ...)

2.a If no **pattern** found, construct a simple representation of L

2.b If **pattern** found, pump it to show that L is hard

Proof strategy overview

1. Take a rigid representation of L

(e.g. minimal automaton, syntactic algebra, ...)

2. Look for a **pattern**

(e.g. graph gadget, violation of equation, ...)

2.a If no **pattern** found, construct a simple representation of L

2.b If **pattern** found, pump it to show that L is hard

Examples:

Proof strategy overview

1. Take a rigid representation of L

   (e.g. minimal automaton, syntactic algebra, ...)

2. Look for a **pattern**

   (e.g. graph gadget, violation of equation, ...)

   2.a If no **pattern** found, construct a simple representation of L

   2.b If **pattern** found, pump it to show that L is hard

Examples:

Benedikt, Blumensath, Bojańczyk, Colcombet, Facchini, Idziaszek, Murlak, Niwiński, Pin, Place, Schutzenberger, Segoufin, Straubing, Thérien, Thomas, Walukiewicz, Wilke, Zeitoun, ...

Proof strategy overview

1. Take a rigid representation of L

(e.g. minimal automaton, syntactic algebra, ...)

2. Look for a **pattern**

(e.g. graph gadget, violation of equation, ...)

2.a If no **pattern** found, construct a simple representation of L

2.b If **pattern** found, pump it to show that L is hard

Proof strategy overview

1. Take a rigid representation of L

(e.g. minimal automaton, syntactic algebra, ...)

2. Look for a **pattern**

(e.g. graph gadget, violation of equation, ...)

2.a If no **pattern** found, construct a simple representation of L

2.b If **pattern** found, pump it to show that L is hard

Limitations:

Proof strategy overview

1. Take a rigid representation of L

(e.g. minimal automaton, syntactic algebra, ...)

2. Look for a **pattern**

(e.g. graph gadget, violation of equation, ...)

2.a If no **pattern** found, construct a simple representation of L

2.b If **pattern** found, pump it to show that L is hard

Limitations:

- **2.a** works under assumption of lack of obstruction

Proof strategy overview

1. Take a rigid representation of L

(e.g. minimal automaton, syntactic algebra, ...)

2. Look for a **pattern**

(e.g. graph gadget, violation of equation, ...)

2.a If no **pattern** found, construct a simple representation of L

2.b If **pattern** found, pump it to show that L is hard

Limitations:

- **2.a** works under assumption of lack of obstruction
- algebraic methods limited to varieties or lattices of languages

Proof strategy overview

1. Take a rigid representation of L

(e.g. minimal automaton, syntactic algebra, ...)

2. Look for a **pattern**

(e.g. graph gadget, violation of equation, ...)

2.a If no **pattern** found, construct a simple representation of L

2.b If **pattern** found, pump it to show that L is hard

Limitations:

- **2.a** works under assumption of lack of obstruction
- algebraic methods limited to varieties or lattices of languages

(Birkhoff)

Proof strategy overview

1. Take a rigid representation of L

(e.g. minimal automaton, syntactic algebra, ...)

2. Look for a **pattern**

(e.g. graph gadget, violation of equation, ...)

2.a If no **pattern** found, construct a simple representation of L

2.b If **pattern** found, pump it to show that L is hard

Limitations:

- **2.a** works under assumption of lack of obstruction
- algebraic methods limited to varieties or lattices of languages
- rigid representations needed (Birkhoff)

Rigid representations

Rigid representations

	logic	automata	algebra	expressions	···
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Det. lang of inf. trees	—	det. top-down	—	—	
Inf. trees	MSO	nondet. parity	ω-clones	—	

Rigid representations

	logic	automata	algebra	expressions	···
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Det. lang of inf. trees	—	det. top-down	—	—	
Inf. trees	MSO	nondet. parity	ω-clones	—	

Rigid representations

	logic	automata	algebra	expressions	...
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Det. lang of inf. trees	—	det. top-down	—	—	
Inf. trees	MSO	nondet. parity	ω-clones	—	

Not mentioned: thin trees, Boolean combinations of open sets, ...

Deterministic parity automata

Deterministic parity automata

$\mathcal{A}$ reads $\alpha = a_0 a_1 \cdots$ and produces $\rho = q_0 q_1 \cdots$

Deterministic parity automata

$\mathcal{A}$ reads $\alpha = a_0 a_1 \cdots$ and produces $\rho = q_0 q_1 \cdots$

$$\alpha = \quad a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad \cdots$$

Deterministic parity automata

$\mathcal{A}$ reads $\alpha = a_0 a_1 \cdots$ and produces $\rho = q_0 q_1 \cdots$

$$\begin{array}{lcccccccccc} \alpha = & & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & & \cdots \\ \rho = & q_0 & \to q_1 & \to q_2 & \to q_3 & \to q_4 & \to q_5 & \to q_6 & \to q_7 & & \cdots \end{array}$$

Deterministic parity automata

$\mathcal{A}$ reads $\alpha = a_0 a_1 \cdots$ and produces $\rho = q_0 q_1 \cdots$

$$
\begin{array}{rccccccccccccccccc}
\alpha = & & a_0 & & a_1 & & a_2 & & a_3 & & a_4 & & a_5 & & a_6 & & & \cdots \\
\rho = & q_0 & \rightarrow & q_1 & \rightarrow & q_2 & \rightarrow & q_3 & \rightarrow & q_4 & \rightarrow & q_5 & \rightarrow & q_6 & \rightarrow & q_7 & & \cdots \\
 & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \\
\Omega: & 1 & & 2 & & 0 & & 2 & & 1 & & 0 & & 2 & & 1 & & \cdots
\end{array}
$$

Deterministic parity automata

$\mathcal{A}$ reads $\alpha = a_0 a_1 \cdots$ and produces $\rho = q_0 q_1 \cdots$

$$
\begin{array}{rcccccccccccccccccc}
\alpha = & & a_0 & & a_1 & & a_2 & & a_3 & & a_4 & & a_5 & & a_6 & & \cdots \\
\rho = & q_0 & \to & q_1 & \to & q_2 & \to & q_3 & \to & q_4 & \to & q_5 & \to & q_6 & \to & q_7 & \cdots \\
 & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \\
\Omega: & 1 & & 2 & & 0 & & 2 & & 1 & & 0 & & 2 & & 1 & \cdots
\end{array}
$$

$\mathcal{A}$ accepts α iff $\limsup_{n\to\infty} \Omega(q_n) \equiv 0 \pmod 2$

Deterministic parity automata

$\mathcal{A}$ reads $\alpha = a_0 a_1 \cdots$ and produces $\rho = q_0 q_1 \cdots$

$$
\begin{array}{rccccccccc}
\alpha = & & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & \cdots \\
\rho = & q_0 \to & q_1 \to & q_2 \to & q_3 \to & q_4 \to & q_5 \to & q_6 \to & q_7 & \cdots \\
& \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\Omega : & 1 & 2 & 0 & 2 & 1 & 0 & 2 & 1 & \cdots
\end{array}
$$

$\mathcal{A}$ accepts α iff $\underbrace{\limsup_{n\to\infty} \Omega(q_n) \equiv 0 \pmod 2}_{\text{parity condition}}$

Deterministic parity automata

$\mathcal{A}$ reads $\alpha = a_0 a_1 \cdots$ and produces $\rho = q_0 q_1 \cdots$

$$\begin{array}{rcccccccccc} \alpha = & & a_0 & & a_1 & & a_2 & & a_3 & & a_4 & & a_5 & & a_6 & & \cdots \\ \rho = & q_0 & \to & q_1 & \to & q_2 & \to & q_3 & \to & q_4 & \to & q_5 & \to & q_6 & \to & q_7 & \cdots \\ & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \\ \Omega : & 1 & & 2 & & 0 & & 2 & & 1 & & 0 & & 2 & & 1 & \cdots \end{array}$$

$\mathcal{A}$ accepts α iff $\underbrace{\limsup_{n\to\infty} \Omega(q_n) \equiv 0 \pmod 2}_{\text{parity condition}}$

$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{\alpha \in A^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$$

Deterministic parity automata

$\mathcal{A}$ reads $\alpha = a_0 a_1 \cdots$ and produces $\rho = q_0 q_1 \cdots$

$$\begin{array}{lcccccccccc}
\alpha = & & a_0 & & a_1 & & a_2 & & a_3 & & a_4 \quad a_5 \quad a_6 \quad \cdots \\
\rho = & q_0 \to & q_1 \to & q_2 \to & q_3 \to & q_4 \to & q_5 \to & q_6 \to & q_7 & \cdots \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\Omega: & 1 & 2 & 0 & 2 & 1 & 0 & 2 & 1 & \cdots
\end{array}$$

$\mathcal{A}$ accepts α iff $\underbrace{\limsup_{n\to\infty} \Omega(q_n) \equiv 0 \pmod 2}_{\text{parity condition}}$

$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{\alpha \in A^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$$

[For later use: the index of $\mathcal{A}$ is $\mathrm{rg}(\Omega) = \{i, i+1, \ldots, j\} \subseteq \mathbb{N}$]

Deterministic parity automata

$\mathcal{A}$ reads $\alpha = a_0 a_1 \cdots$ and produces $\rho = q_0 q_1 \cdots$

$$
\begin{array}{rcccccccccc}
\alpha = & & a_0 & & a_1 & & a_2 & & a_3 & & a_4 & & a_5 & & a_6 & & \cdots \\
\rho = & q_0 & \to & q_1 & \to & q_2 & \to & q_3 & \to & q_4 & \to & q_5 & \to & q_6 & \to & q_7 & \cdots \\
& \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \\
\Omega : & 1 & & 2 & & 0 & & 2 & & 1 & & 0 & & 2 & & 1 & \cdots
\end{array}
$$

$\mathcal{A}$ accepts α iff $\underbrace{\limsup_{n\to\infty} \Omega(q_n) \equiv 0 \pmod 2}_{\text{parity condition}}$

$$\mathrm{L}(\mathcal{A}) \overset{\text{def}}{=} \{\alpha \in A^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$$

[For later use: the index of $\mathcal{A}$ is $\mathrm{rg}(\Omega) = \underline{\{i, i+1, \ldots, j\}} \subseteq \mathbb{N}$]

Part 3

Games on graphs

Games of: infinite duration, perfect information, finite arena

Games of: infinite duration, perfect information, finite arena

$\mathcal{G}$ has: vertices $V = V_\exists \sqcup V_\forall$,

Games of: infinite duration, perfect information, finite arena

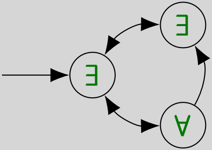
$\mathcal{G}$ has: vertices $V = V_\exists \sqcup V_\forall$, initial vertex $v_0 \in V$,

Games of: infinite duration, perfect information, finite arena

$\mathcal{G}$ has: vertices $V = V_{\exists} \sqcup V_{\forall}$, initial vertex $v_0 \in V$, edges $E \subseteq V \times V$

Games of: infinite duration, perfect information, finite arena

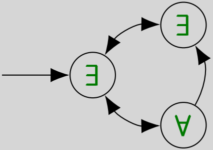
$\mathcal{G}$ has: vertices $V = V_{\exists} \sqcup V_{\forall}$, initial vertex $v_0 \in V$, edges $E \subseteq V \times V$



Games of: infinite duration, perfect information, finite arena

$\mathcal{G}$ has: vertices $V = V_{\exists} \sqcup V_{\forall}$, initial vertex $v_0 \in V$, edges $E \subseteq V \times V$

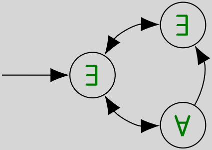
+ winning condition $W \subseteq V^{\omega}$



Games of: infinite duration, perfect information, finite arena

$\mathcal{G}$ has: vertices $V = V_{\exists} \sqcup V_{\forall}$, initial vertex $v_0 \in V$, edges $E \subseteq V \times V$

+ winning condition $W \subseteq V^{\omega}$

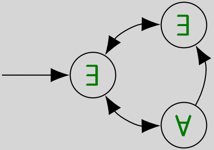


$\rightsquigarrow$ a play $\pi = v_0 v_1 v_2 \cdots$

Games of: infinite duration, perfect information, finite arena

$\mathcal{G}$ has: vertices $V = V_\exists \sqcup V_\forall$, initial vertex $v_0 \in V$, edges $E \subseteq V \times V$

+ winning condition $W \subseteq V^\omega$



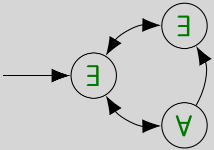
$\rightsquigarrow$ a play $\pi = v_0 v_1 v_2 \cdots$

$\exists$ wins π iff $\pi \in W$

Games of: infinite duration, perfect information, finite arena

$\mathcal{G}$ has: vertices $V = V_\exists \sqcup V_\forall$, initial vertex $v_0 \in V$, edges $E \subseteq V \times V$

$+$ winning condition $W \subseteq V^\omega$



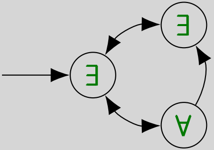
$\rightsquigarrow$ a play $\pi = v_0 v_1 v_2 \cdots$

$\exists$ wins π iff $\pi \in W$ $\forall$ wins π iff $\pi \notin W$

Games of: infinite duration, perfect information, finite arena

$\mathcal{G}$ has: vertices $V = V_\exists \sqcup V_\forall$, initial vertex $v_0 \in V$, edges $E \subseteq V \times V$

+ winning condition $W \subseteq V^\omega$



$\rightsquigarrow$ a play $\pi = v_0 v_1 v_2 \cdots$

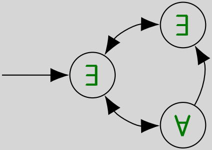
$\exists$ wins π iff $\pi \in W$ $\qquad$ $\forall$ wins π iff $\pi \notin W$

$\rightsquigarrow$ each play has a winner

Games of: infinite duration, perfect information, finite arena

$\mathcal{G}$ has: vertices $V = V_\exists \sqcup V_\forall$, initial vertex $v_0 \in V$, edges $E \subseteq V \times V$

$+$ winning condition $W \subseteq V^\omega$



$\rightsquigarrow$ a play $\pi = v_0 v_1 v_2 \cdots$

$\exists$ wins π iff $\pi \in W$ $\qquad$ $\forall$ wins π iff $\pi \notin W$

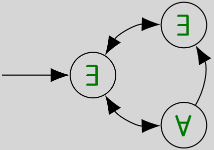
$\rightsquigarrow$ each play has a winner

But: for certain $W \subseteq V^\omega$ none of the players has a winning strategy!

Games of: infinite duration, perfect information, finite arena

$\mathcal{G}$ has: vertices $V = V_\exists \sqcup V_\forall$, initial vertex $v_0 \in V$, edges $E \subseteq V \times V$

+ winning condition $W \subseteq V^\omega$



$\rightsquigarrow$ a play $\pi = v_0 v_1 v_2 \cdots$

$\exists$ wins π iff $\pi \in W$ $\qquad$ $\forall$ wins π iff $\pi \notin W$

$\rightsquigarrow$ each play has a winner

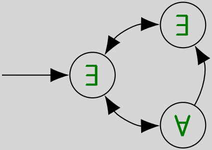
But: for certain $W \subseteq V^\omega$ none of the players has a winning strategy!

E.g. for infinite XOR, see (Kopczyński, Niwiński [2014])

Games of: infinite duration, perfect information, finite arena

$\mathcal{G}$ has: vertices $V = V_\exists \sqcup V_\forall$, initial vertex $v_0 \in V$, edges $E \subseteq V \times V$

+ winning condition $W \subseteq V^\omega$



$\rightsquigarrow$ a play $\pi = v_0 v_1 v_2 \cdots$

$\exists$ wins π iff $\pi \in W$ $\forall$ wins π iff $\pi \notin W$

$\rightsquigarrow$ each play has a winner

But: for certain $W \subseteq V^\omega$ none of the players has a winning strategy!

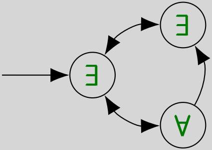
E.g. for infinite XOR, see (Kopczyński, Niwiński [2014])

Also: a winning strategy may require unlimitted memory!

Games of: infinite duration, perfect information, finite arena

$\mathcal{G}$ has: vertices $V = V_\exists \sqcup V_\forall$, initial vertex $v_0 \in V$, edges $E \subseteq V \times V$

+ winning condition $W \subseteq V^\omega$



$\rightsquigarrow$ a play $\pi = v_0 v_1 v_2 \cdots$

$\exists$ wins π iff $\pi \in W$ $\forall$ wins π iff $\pi \notin W$

$\rightsquigarrow$ each play has a winner

But: for certain $W \subseteq V^\omega$ none of the players has a winning strategy!

E.g. for infinite XOR, see (Kopczyński, Niwiński [2014])

Also: a winning strategy may require unlimitted memory!

And: it may not be decidable which of the players wins. . .

Theorem (Büchi, Landweber [1969])

If $W \subseteq V^\omega$ is regular then:

Theorem (Büchi, Landweber [1969])

If $W \subseteq V^\omega$ is regular then:

- it is decidable who wins the game

Theorem (Büchi, Landweber [1969])

If $W \subseteq V^\omega$ is regular then:

- it is decidable who wins the game
- there exists a finite memory winning strategy

Theorem (Büchi, Landweber [1969])

If $W \subseteq V^\omega$ is regular then:

- it is decidable who wins the game
- there exists a finite memory winning strategy
- such a strategy can be effectively constructed

Theorem (Büchi, Landweber [1969])

If $W \subseteq V^\omega$ is regular then:

- it is decidable who wins the game
- there exists a finite memory winning strategy
- such a strategy can be effectively constructed

Proof

Theorem (Büchi, Landweber [1969])

If $W \subseteq V^\omega$ is regular then:

- it is decidable who wins the game
- there exists a finite memory winning strategy
- such a strategy can be effectively constructed

Proof

1. $W = \mathrm{L}(\mathcal{A})$

Theorem (Büchi, Landweber [1969])

If $W \subseteq V^\omega$ is regular then:

- it is decidable who wins the game
- there exists a finite memory winning strategy
- such a strategy can be effectively constructed

Proof

1. $W = \mathrm{L}(\mathcal{A})$

2. Consider $\mathcal{G} \times \mathcal{A}$:

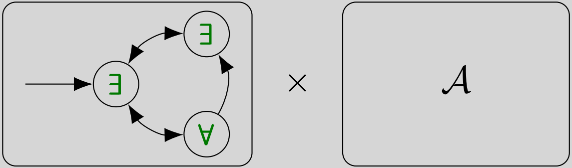
Theorem (Büchi, Landweber [1969])

If $W \subseteq V^\omega$ is regular then:

- it is decidable who wins the game
- there exists a finite memory winning strategy
- such a strategy can be effectively constructed

Proof

1. $W = \mathrm{L}(\mathcal{A})$
2. Consider $\mathcal{G} \times \mathcal{A}$:



Theorem (Büchi, Landweber [1969])

If $W \subseteq V^{\omega}$ is regular then:

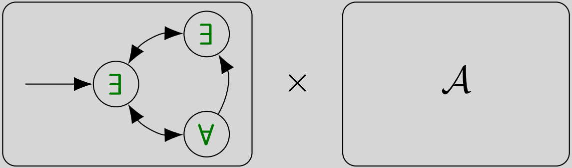
- it is decidable who wins the game
- there exists a finite memory winning strategy
- such a strategy can be effectively constructed

Proof

1. $W = \mathrm{L}(\mathcal{A})$

2. Consider $\mathcal{G} \times \mathcal{A}$:

3. Apply:



Theorem (Mostowski [1991]; Emmerson, Jutla [1991])

Parity games are effectively, positionally determined.

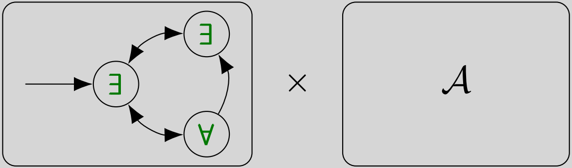
Theorem (Büchi, Landweber [1969])

If $W \subseteq V^\omega$ is regular then:

- it is decidable who wins the game
- there exists a finite memory winning strategy
- such a strategy can be effectively constructed

Proof

1. $W = \mathrm{L}(\mathcal{A})$
2. Consider $\mathcal{G} \times \mathcal{A}$:
3. Apply:



Theorem (Mostowski [1991]; Emmerson, Jutla [1991])

Parity games are effectively, positionally determined.

$\rightsquigarrow$ the winner of $\mathcal{G}$ can use $\mathcal{A}$ as a memory structure

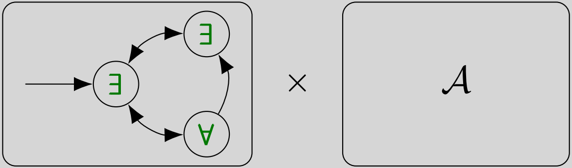
Theorem (Büchi, Landweber [1969])

If $W \subseteq V^\omega$ is regular then:

- it is decidable who wins the game
- there exists a finite memory winning strategy
- such a strategy can be effectively constructed

Proof

1. $W = \mathrm{L}(\mathcal{A})$
2. Consider $\mathcal{G} \times \mathcal{A}$:
3. Apply:



Theorem (Mostowski [1991]; Emmerson, Jutla [1991])

Parity games are effectively, positionally determined.

⇝ the winner of $\mathcal{G}$ can use $\mathcal{A}$ as a memory structure

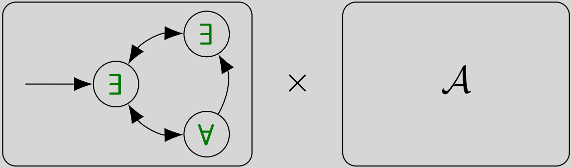
Theorem (Büchi, Landweber [1969])

If $W \subseteq V^\omega$ is regular then:

- it is decidable who wins the game
- there exists a finite memory winning strategy
- such a strategy can be effectively constructed

Proof

1. $W = \mathrm{L}(\mathcal{A})$
2. Consider $\mathcal{G} \times \mathcal{A}$:
3. Apply:



Theorem (Mostowski [1991]; Emmerson, Jutla [1991])

Parity games are effectively, positionally determined.

$\rightsquigarrow$ the winner of $\mathcal{G}$ can use $\mathcal{A}$ as a memory structure

---

(Chatterjee, Henzinger, Kupferman, Piterman, Vardi, ...)

Theorem (Büchi, Landweber [1969])

If $W \subseteq V^\omega$ is regular then:

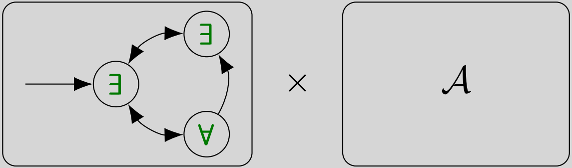
- it is decidable who wins the game
- there exists a finite memory winning strategy
- such a strategy can be effectively constructed

Proof

1. $W = \mathrm{L}(\mathcal{A})$

2. Consider $\mathcal{G} \times \mathcal{A}$:

3. Apply:



Theorem (Mostowski [1991]; Emmerson, Jutla [1991])

Parity games are effectively, positionally determined.

$\rightsquigarrow$ the winner of $\mathcal{G}$ can use $\mathcal{A}$ as a memory structure

---

(Chatterjee, Henzinger, Kupferman, Piterman, Vardi, . . .)

(Kopczyński [2006]; Zimmermann [2016]; Colcombet, Göller [2016])

Part 4

First examples

Task:

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$ with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$\forall$:

$\exists$:

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$\forall$: a_0

$\exists$:

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$\forall$: $a_0 \in A$

$\exists$:

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$\forall$: $a_0 \in A$

$\exists$: p_0

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$\forall$: $a_0 \in A$

$\exists$: $p_0 \in \{i, \ldots, j\}$

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$\forall$: $a_0 \in A$ $\quad a_1$

$\exists$: $p_0 \in \{i, \ldots, j\}$

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$\forall$: $a_0 \in A$ $\quad a_1$

$\exists$: $p_0 \in \{i, \ldots, j\}$ $\quad p_1$

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$ with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$\forall$:	$a_0 \in A$	a_1	a_2	a_3	a_4	a_5	a_6
$\exists$:	$p_0 \in \{i, \ldots, j\}$	p_1	p_2	p_3	p_4	p_5	p_6

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$ with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$\forall$:	$a_0 \in A$	a_1	a_2	a_3	a_4	a_5	a_6	$\cdots$
$\exists$:	$p_0 \in \{i, \ldots, j\}$	p_1	p_2	p_3	p_4	p_5	p_6	$\cdots$

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$ with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$$
\begin{array}{lllllllll}
\forall: & a_0 \in A & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & \cdots \\
\exists: & p_0 \in \{i, \ldots, j\} & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & \cdots
\end{array}
$$

$$W \stackrel{\text{def}}{=} \Big\{a_0 p_0 a_1 p_1 \cdots \mid \big(a_0 a_1 \cdots \in L\big) \iff \big(\limsup_{n\to\infty} p_n \equiv 0 \pmod 2\big)\Big\}$$

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$$
\begin{array}{llllllll}
\forall: & \overset{\in A}{a_0} & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \; \cdots \\
\exists: & \underset{\in \{i,\ldots,j\}}{p_0} & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \; \cdots
\end{array}
$$

$$W \stackrel{\text{def}}{=} \Big\{ a_0 p_0 a_1 p_1 \cdots \mid \big(a_0 a_1 \cdots \in L\big) \iff \underbrace{\big(\limsup_{n\to\infty} p_n \equiv 0 \pmod 2\big)}_{\text{parity condition}} \Big\}$$

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$ with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$$
\begin{array}{llllllllll}
\forall: & a_0 \in A & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & \cdots \\
\exists: & p_0 \in \{i, \ldots, j\} & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & \cdots
\end{array}
$$

$$W \stackrel{\text{def}}{=} \Big\{ a_0 p_0 a_1 p_1 \cdots \mid \big(a_0 a_1 \cdots \in L\big) \iff \underbrace{\big(\limsup_{n\to\infty} p_n \equiv 0 \pmod 2\big)}_{\text{parity condition}} \Big\}$$

1. W is regular

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$$\begin{array}{lccccccccc} & \overset{\in A}{} & & & & & & & \\ \forall: & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & \cdots \\ \exists: & p_0 & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & \cdots \\ & \underset{\in \{i,\ldots,j\}}{} & & & & & & & \end{array}$$

$$W \stackrel{\text{def}}{=} \Big\{ a_0 p_0 a_1 p_1 \cdots \mid \big(a_0 a_1 \cdots \in L\big) \iff \underbrace{\big(\limsup_{n\to\infty} p_n \equiv 0 \pmod 2\big)}_{\text{parity condition}} \Big\}$$

1. W is regular

2. $\exists$ wins $\Rightarrow$ her strategy is a det. parity aut. for L

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$ with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$\forall$:	$a_0 \in A$	a_1	a_2	a_3	a_4	a_5	a_6	$\cdots$
$\exists$:	$p_0 \in \{i, \ldots, j\}$	p_1	p_2	p_3	p_4	p_5	p_6	$\cdots$

$$W \stackrel{\text{def}}{=} \Big\{ a_0 p_0 a_1 p_1 \cdots \mid \big(a_0 a_1 \cdots \in L\big) \iff \underbrace{\big(\limsup_{n\to\infty} p_n \equiv 0 \pmod 2\big)}_{\text{parity condition}} \Big\}$$

1. W is regular

2. $\exists$ wins $\Rightarrow$ her strategy is a det. parity aut. for L

   $\rightsquigarrow$ a representation for L

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$\forall$:	$a_0 \in A$	a_1	a_2	a_3	a_4	a_5	a_6	$\cdots$
$\exists$:	$p_0 \in \{i, \ldots, j\}$	p_1	p_2	p_3	p_4	p_5	p_6	$\cdots$

$$W \stackrel{\text{def}}{=} \Big\{ a_0 p_0 a_1 p_1 \cdots \mid \big(a_0 a_1 \cdots \in L\big) \iff \underbrace{\big(\limsup_{n\to\infty} p_n \equiv 0 \pmod 2\big)}_{\text{parity condition}} \Big\}$$

1. W is regular
2. $\exists$ wins $\Rightarrow$ her strategy is a det. parity aut. for L
   $\rightsquigarrow$ a representation for L
3. $\forall$ wins $\Rightarrow$ his strategy shows that no such automaton exists

Task:

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$ with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$$
\begin{array}{llllllll}
\forall: & \underset{}{a_0}^{\in A} & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & \cdots \\
\exists: & p_0{}_{\in \{i,\ldots,j\}} & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & \cdots
\end{array}
$$

$$W \stackrel{\text{def}}{=} \Big\{ a_0 p_0 a_1 p_1 \cdots \mid \big(a_0 a_1 \cdots \in L\big) \iff \underbrace{\big(\limsup_{n\to\infty} p_n \equiv 0 \pmod 2\big)}_{\text{parity condition}} \Big\}$$

1. W is regular

2. $\exists$ wins $\Rightarrow$ her strategy is a det. parity aut. for L

$\rightsquigarrow$ a representation for L

3. $\forall$ wins $\Rightarrow$ his strategy shows that no such automaton exists

$\rightsquigarrow$ a witness of hardness for L

Task: **(Wadge+Wagner hierarchy)**

Input: Regular $L \subseteq A^\omega$ and $i < j$

Output: Can L be recognised by a det. parity aut. $\mathcal{A}$ with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Game:

$\forall$:	$a_0 \in A$	a_1	a_2	a_3	a_4	a_5	a_6	$\cdots$
$\exists$:	$p_0 \in \{i, \ldots, j\}$	p_1	p_2	p_3	p_4	p_5	p_6	$\cdots$

$$W \stackrel{\text{def}}{=} \Big\{ a_0 p_0 a_1 p_1 \cdots \mid \big(a_0 a_1 \cdots \in L\big) \iff \underbrace{\big(\limsup_{n\to\infty} p_n \equiv 0 \pmod 2\big)}_{\text{parity condition}} \Big\}$$

1. W is regular

2. $\exists$ wins $\Rightarrow$ her strategy is a det. parity aut. for L

⇝ a representation for L

3. $\forall$ wins $\Rightarrow$ his strategy shows that no such automaton exists

⇝ a witness of hardness for L

Task:

Task:

Input: Regular $L \subseteq A^*$ and k

Task:

Input: Regular $L \subseteq A^*$ and k

Output: Can L be defined by a regexp

Task:

Input: Regular $L \subseteq A^*$ and k

Output: Can L be defined by a regexp of star-height $\leqslant k$ (no complementation here!)

Task:

Input: Regular $L \subseteq A^*$ and k

Output: Can L be defined by a regexp of star-height $\leqslant k$ (no complementation here!)

Solution 1 (Hashiguchi [1988]): complicated

Task:

Input: Regular $L \subseteq A^*$ and k

Output: Can L be defined by a regexp

of star-height $\leqslant k$ (no complementation here!)

Solution 1 (Hashiguchi [1988]): complicated

Solution 2 (Kirsten [2005]):

Task:

Input: Regular $L \subseteq A^*$ and k

Output: Can L be defined by a regexp of star-height $\leqslant k$ (no complementation here!)

Solution 1 (Hashiguchi [1988]): complicated

Solution 2 (Kirsten [2005]):

2.a: reduce to limitedness of some counter-automata (rather easy)

Task:

Input: Regular $L \subseteq A^*$ and k

Output: Can L be defined by a regexp of star-height $\leqslant k$ (no complementation here!)

Solution 1 (Hashiguchi [1988]): complicated

Solution 2 (Kirsten [2005]):

2.a: reduce to limitedness of some counter-automata (rather easy)

2.b: solve limitedness by hand

Task:

Input: Regular $L \subseteq A^*$ and k

Output: Can L be defined by a regexp of star-height $\leqslant k$ (no complementation here!)

Solution 1 (Hashiguchi [1988]): complicated

Solution 2 (Kirsten [2005]):

2.a: reduce to limitedness of some counter-automata (rather easy)

2.b: solve limitedness by hand

Solutions 2', ... (Colcombet [2009]; Toruńczyk [2011]): understand **2.b**

Task:

Input: Regular $L \subseteq A^*$ and k

Output: Can L be defined by a regexp of star-height $\leqslant k$ (no complementation here!)

Solution 1 (Hashiguchi [1988]): complicated

Solution 2 (Kirsten [2005]):

2.a: reduce to limitedness of some counter-automata (rather easy)

2.b: solve limitedness by hand

Solutions 2', ... (Colcombet [2009]; Toruńczyk [2011]): understand **2.b**

Solution 3 (Bojańczyk [2015]): solve **2.b** directly by a game!

Task:

Input: Regular $L \subseteq A^*$ and k

Output: Can L be defined by a regexp of star-height $\leqslant k$ (no complementation here!)

Solution 1 (Hashiguchi [1988]): complicated

Solution 2 (Kirsten [2005]):

2.a: reduce to limitedness of some counter-automata (rather easy)

2.b: solve limitedness by hand

Solutions 2', ... (Colcombet [2009]; Toruńczyk [2011]): understand **2.b**

Solution 3 (Bojańczyk [2015]): solve **2.b** directly by a game!

- construct a game $\mathcal{G}$ with regular W

Task:

Input: Regular $L \subseteq A^*$ and k

Output: Can L be defined by a regexp of star-height $\leqslant k$ (no complementation here!)

Solution 1 (Hashiguchi [1988]): complicated

Solution 2 (Kirsten [2005]):

2.a: reduce to limitedness of some counter-automata (rather easy)

2.b: solve limitedness by hand

Solutions 2', ... (Colcombet [2009]; Toruńczyk [2011]): understand **2.b**

Solution 3 (Bojańczyk [2015]): solve **2.b** directly by a game!

- construct a game $\mathcal{G}$ with regular W
- if $\exists$ wins $\mathcal{G}$ then her memory gives limitedness

Task:

Input: Regular $L \subseteq A^*$ and k

Output: Can L be defined by a regexp of star-height $\leqslant k$ (no complementation here!)

Solution 1 (Hashiguchi [1988]): complicated

Solution 2 (Kirsten [2005]):

2.a: reduce to limitedness of some counter-automata (rather easy)

2.b: solve limitedness by hand

Solutions 2', ... (Colcombet [2009]; Toruńczyk [2011]): understand **2.b**

Solution 3 (Bojańczyk [2015]): solve **2.b** directly by a game!

- construct a game $\mathcal{G}$ with regular W
- if $\exists$ wins $\mathcal{G}$ then her memory gives limitedness
- if $\forall$ wins $\mathcal{G}$ then there is **no** limitedness

Part 5

More examples (infinite trees)

Rigid representations

	logic	**automata**	**algebra**	**expressions**	...
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Det. lang of inf. trees	—	det. top-down	—	—	
Inf. trees	MSO	nondet. parity	ω-clones	—	

Rigid representations

	logic	**automata**	**algebra**	**expressions**	⋯
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Det. lang of inf. trees	—	det. top-down	—	—	
Inf. trees	MSO	nondet. parity	ω-clones	—	

⇝ infinite trees inherently require non-determinism

Rigid representations

	logic	automata	algebra	expressions	⋯
Fin. words	MSO	DFA	monoids	regexp	
Inf. words	MSO	det. parity	Wilke alg.	ω-regexp	
Fin. trees	MSO	det. bottom-up	forest alg.	tree regexp	
Det. lang of inf. trees	—	det. top-down	—	—	
Inf. trees	MSO	nondet. parity	ω-clones	—	

⇝ infinite trees inherently require non-determinism

(Niwiński, Walukiewicz [1996]; Carayol, Löding [2010])

(Bilkowski, S. [2013]; Blumensath [2013])

Task (Rabin-Mostowski index problem):

Task (Rabin-Mostowski index problem):

Input: Regular language of inf. trees L and $i < j$

Task (Rabin-Mostowski index problem):

Input: Regular language of inf. trees L and $i < j$

Output: Can L be recognised by a non-det. parity tree aut.

Task (Rabin-Mostowski index problem):

Input: Regular language of inf. trees L and $i < j$

Output: Can L be recognised by a non-det. parity tree aut. with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Task (Rabin-Mostowski index problem):

Input: Regular language of inf. trees L and $i < j$

Output: Can L be recognised by a non-det. parity tree aut. with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Theorem (Colcombet, Löding [2008])

Reduction to domination of cost functions

Task (Rabin-Mostowski index problem):

Input: Regular language of inf. trees L and $i < j$

Output: Can L be recognised by a non-det. parity tree aut. with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Theorem (Colcombet, Löding [2008])

Reduction to domination of cost functions

[not known to be decidable]

Task (Rabin-Mostowski index problem):

Input: Regular language of inf. trees L and $i < j$

Output: Can L be recognised by a non-det. parity tree aut.

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Theorem (Colcombet, Löding [2008])

Reduction to domination of cost functions

$$\big[\underbrace{\text{not known to be decidable}}_{\text{finite memory determinacy???}}\big]$$

(Fijalkow, Horn, Kuperberg, S. [2015])

Task (Rabin-Mostowski index problem):

Input: Regular language of inf. trees L and $i < j$

Output: Can L be recognised by a non-det. parity tree aut. with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Theorem (Colcombet, Löding [2008])

Reduction to domination of cost functions

$$\underbrace{\left[\text{ not known to be decidable }\right]}_{\text{finite memory determinacy???}}$$

(Fijalkow, Horn, Kuperberg, S. [2015])

Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Task (Rabin-Mostowski index problem):

Input: Regular language of inf. trees L and $i < j$

Output: Can L be recognised by a non-det. parity tree aut.

with $\mathrm{rg}(\Omega) \subseteq \{i, \ldots, j\}$ (i.e. index $\{i, \ldots, j\}$)?

Theorem (Colcombet, Löding [2008])

Reduction to domination of cost functions

[not known to be decidable]

finite memory determinacy???

(Fijalkow, Horn, Kuperberg, S. [2015])

Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

[framework of domination games]

Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Theorem (S., Walukiewicz 2014)

The same, directly by a game $\mathcal{F}$

Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Theorem (S., Walukiewicz 2014)

The same, directly by a game $\mathcal{F}$ $\quad W \equiv A \vee \big(B \wedge C\big)$

Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Theorem (S., Walukiewicz 2014)

The same, directly by a game $\mathcal{F}$ $\quad W \equiv A \vee \big(B \wedge C\big)$

$\exists$ wins $\Rightarrow L$ is $(1, 2)$-definable

Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Theorem (S., Walukiewicz 2014)

The same, directly by a game $\mathcal{F}$ $\quad W \equiv A \vee \big(B \wedge C\big)$

$\exists$ wins $\Rightarrow L$ is $(1, 2)$-definable $\quad$ $\forall$ wins $\Rightarrow L$ is **not** $(1, 2)$-definable

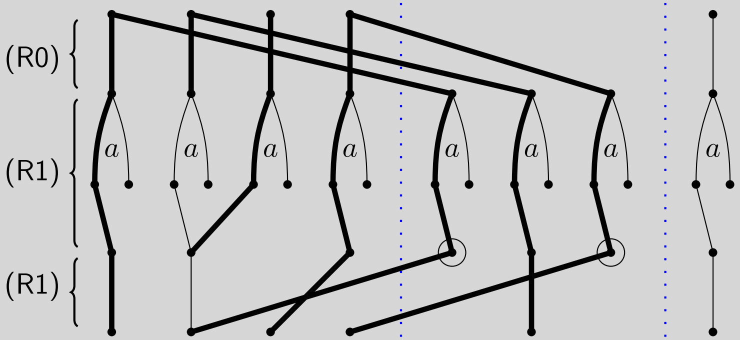
Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Theorem (S., Walukiewicz 2014)

The same, directly by a game $\mathcal{F}$ $\quad W \equiv A \vee \big(B \wedge C\big)$

$\exists$ wins $\Rightarrow$ L is $(1, 2)$-definable $\quad$ $\forall$ wins $\Rightarrow$ L is **not** $(1, 2)$-definable



Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Theorem (S., Walukiewicz 2014)

The same, directly by a game $\mathcal{F}$ $\quad W \equiv A \vee \big(B \wedge C\big)$

∃ wins ⇒ L is $(1, 2)$-definable

∀ wins ⇒ L is **not** $(1, 2)$-definable

Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Theorem (S., Walukiewicz 2014)

The same, directly by a game $\mathcal{F}$ $\quad W \equiv A \vee \big(B \wedge C\big)$

$\exists$ wins $\Rightarrow L$ is $(1, 2)$-definable $\quad$ $\forall$ wins $\Rightarrow L$ is **not** $(1, 2)$-definable

But it seemed that we can get more (ranks)!

Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Theorem (S., Walukiewicz 2014)

The same, directly by a game $\mathcal{F}$ $\quad W \equiv A \vee \big(B \wedge C\big)$

$\exists$ wins $\Rightarrow L$ is $(1, 2)$-definable $\quad$ $\forall$ wins $\Rightarrow L$ is **not** $(1, 2)$-definable

But it seemed that we can get more (ranks)!

Theorem (S., Walukiewicz [2016])

More, directly by a game $\mathcal{F}'$

Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Theorem (S., Walukiewicz 2014)

The same, directly by a game $\mathcal{F}$ $\quad W \equiv A \vee \big(B \wedge C\big)$

$\exists$ wins $\Rightarrow L$ is $(1,2)$-definable $\quad\quad$ $\forall$ wins $\Rightarrow L$ is **not** $(1,2)$-definable

But it seemed that we can get more (ranks)!

Theorem (S., Walukiewicz [2016])

More, directly by a game $\mathcal{F}'$ $\quad W \equiv \big(A \vee B\big) \wedge C'$

Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Theorem (S., Walukiewicz 2014)

The same, directly by a game $\mathcal{F}$ $\qquad W \equiv A \vee \big(B \wedge C\big)$

$\exists$ wins $\Rightarrow L$ is $(1,2)$-definable $\qquad \forall$ wins $\Rightarrow L$ is **not** $(1,2)$-definable

But it seemed that we can get more (ranks)!

Theorem (S., Walukiewicz [2016])

More, directly by a game $\mathcal{F}'$ $\qquad W \equiv \big(A \vee B\big) \wedge C'$

$\exists$ wins $\Rightarrow L$ is $(1,2)$-definable

and **Borel**

Theorem (Colcombet, Kuperberg, Löding, Vanden Boom [2013])

Solution of the Büchi case: $L = \mathrm{L}(\mathcal{B})^{\mathrm{c}}$, $i = 1$, $j = 2$, $\mathrm{rg}(\Omega^{\mathcal{B}}) = \{1, 2\}$

Theorem (S., Walukiewicz 2014)

The same, directly by a game $\mathcal{F}$ $\quad W \equiv A \vee \big(B \wedge C\big)$

$\exists$ wins $\Rightarrow L$ is $(1,2)$-definable $\quad$ $\forall$ wins $\Rightarrow L$ is **not** $(1,2)$-definable

But it seemed that we can get more (ranks)!

Theorem (S., Walukiewicz [2016])

More, directly by a game $\mathcal{F}'$ $\quad W \equiv \big(A \vee B\big) \wedge C'$

$\exists$ wins $\Rightarrow L$ is $(1,2)$-definable and **Borel** $\quad$ $\forall$ wins $\Rightarrow L$ is **not** $(1,2)$-definable and **non-Borel**

Part 6

Last example(s)

Theorem (Cavallari, Michalewski, S. [2017])

Let L be regular lang. of inf. trees. Then effectively either:

Theorem (Cavallari, Michalewski, S. [2017])

Let L be regular lang. of inf. trees. Then effectively either:

1. L is $\text{weak}-\text{alt}(0,2)$-definable and $L \in \mathbf{\Pi}^0_2$

Theorem (Cavallari, Michalewski, S. [2017])

Let L be regular lang. of inf. trees. Then effectively either:

1. L is $\mathrm{weak{-}alt}(0,2)$-definable and $L \in \mathbf{\Pi}^0_2$

2. L isn't $\mathrm{weak{-}alt}(0,2)$-definable and $L \notin \mathbf{\Pi}^0_2$

Theorem (Cavallari, Michalewski, S. [2017])

Let L be regular lang. of inf. trees. Then effectively either:

1. L is $\mathrm{weak{-}alt}(0,2)$-definable and $L \in \mathbf{\Pi}^0_2$

2. L isn't $\underbrace{\mathrm{weak{-}alt}(0,2)}_{\text{weak index}}$-definable and $L \notin \mathbf{\Pi}^0_2$

Theorem (Cavallari, Michalewski, S. [2017])

Let L be regular lang. of inf. trees. Then effectively either:

1. L is $\mathrm{weak{-}alt}(0,2)$-definable and $L \in \mathbf{\Pi}^0_2$
2. L isn't $\underbrace{\mathrm{weak{-}alt}(0,2)}_{\text{weak index}}$-definable and $\underbrace{L \notin \mathbf{\Pi}^0_2}_{\text{topological complexity}}$

Theorem (Cavallari, Michalewski, S. [2017])

Let L be regular lang. of inf. trees. Then effectively either:

1. L is $\mathrm{weak{-}alt}(0,2)$-definable and $L \in \mathbf{\Pi}^0_2$

2. L isn't $\underbrace{\mathrm{weak{-}alt}(0,2)}_{\text{weak index}}$-definable and $\underbrace{L \notin \mathbf{\Pi}^0_2}_{\text{topological complexity}}$

Proof

Take two non-det. parity tree automata: $\mathcal{A}$ for L and $\mathcal{B}$ for L^c.

Theorem (Cavallari, Michalewski, S. [2017])

Let L be regular lang. of inf. trees. Then effectively either:

1. L is $\mathrm{weak{-}alt}(0,2)$-definable and $L \in \mathbf{\Pi}^0_2$

2. L isn't $\underbrace{\mathrm{weak{-}alt}(0,2)}_{\text{weak index}}$-definable and $\underbrace{L \notin \mathbf{\Pi}^0_2}_{\text{topological complexity}}$

Proof

Take two non-det. parity tree automata: $\mathcal{A}$ for L and $\mathcal{B}$ for L^{c}.

Consider a game $\mathcal{F}$ on $\mathcal{B} \times \mathcal{A} \times \mathcal{A}$

Theorem (Cavallari, Michalewski, S. [2017])

Let L be regular lang. of inf. trees. Then effectively either:

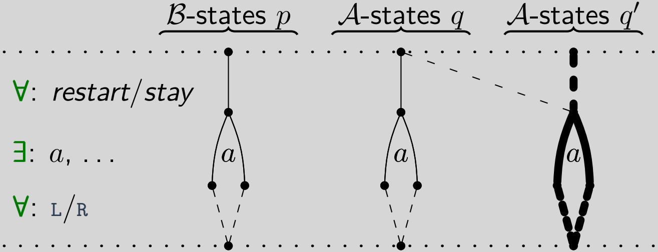
1. L is $\mathrm{weak{-}alt}(0,2)$-definable and $L \in \mathbf{\Pi}^0_2$

2. L isn't $\underbrace{\mathrm{weak{-}alt}(0,2)}_{\text{weak index}}$-definable and $\underbrace{L \notin \mathbf{\Pi}^0_2}_{\text{topological complexity}}$

Proof

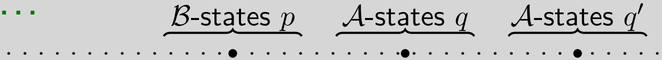
Take two non-det. parity tree automata: $\mathcal{A}$ for L and $\mathcal{B}$ for L^{c}.

Consider a game $\mathcal{F}$ on $\mathcal{B} \times \mathcal{A} \times \mathcal{A}$



Gameplay. . .

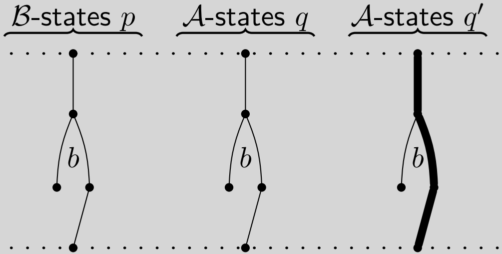
Gameplay. . .



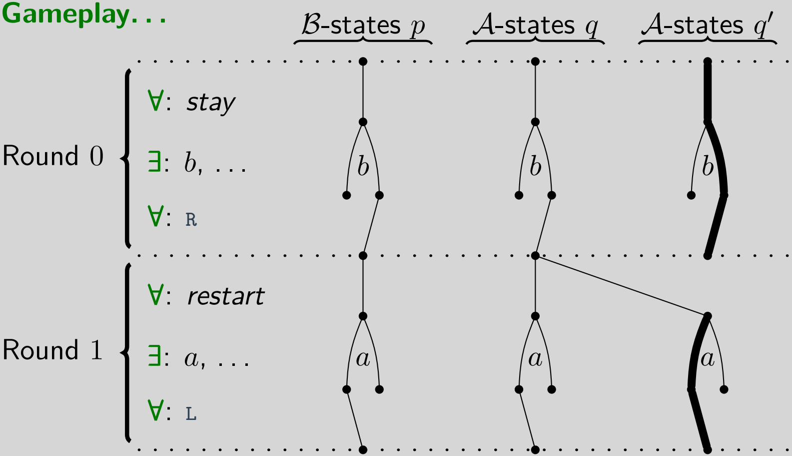
Gameplay...

Round 0

- $\forall$: *stay*
- $\exists$: b, ...
- $\forall$: R



Gameplay...



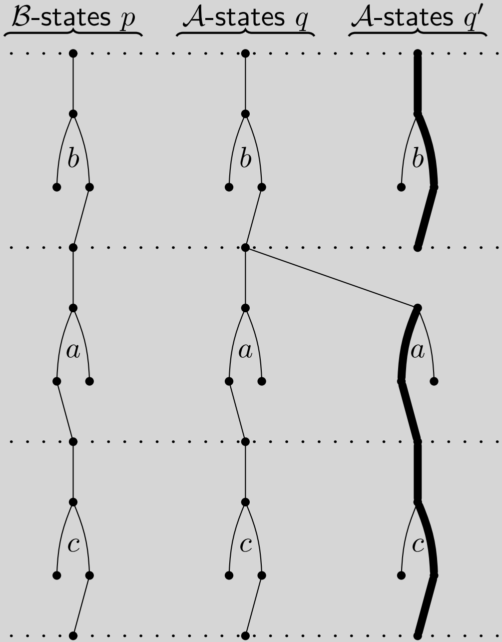
$\mathcal{B}$-states p
$\mathcal{A}$-states q
$\mathcal{A}$-states q'
Round 0
∀: *stay*
∃: b, ...
∀: R
Round 1
∀: *restart*
∃: a, ...
∀: L
b
b
b
a
a
a

Gameplay...

Round 0
- $\forall$: *stay*
- $\exists$: b, ...
- $\forall$: R

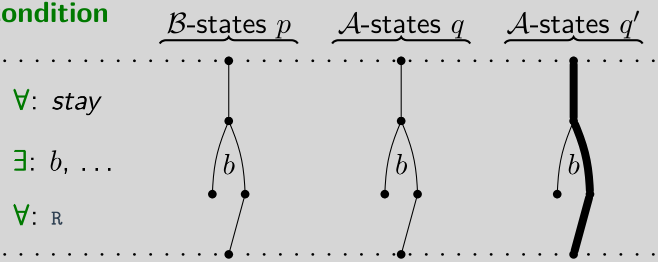
Round 1
- $\forall$: *restart*
- $\exists$: a, ...
- $\forall$: L

Round 2
- $\forall$: *stay*
- $\exists$: c, ...
- $\forall$: R

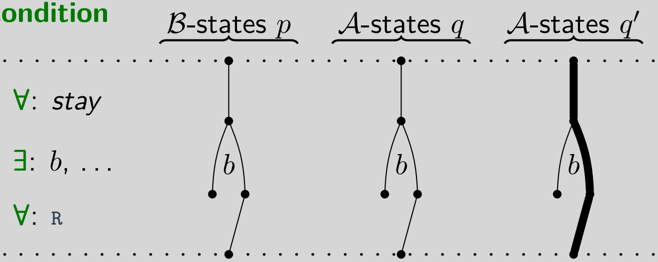


Winning condition

Winning condition

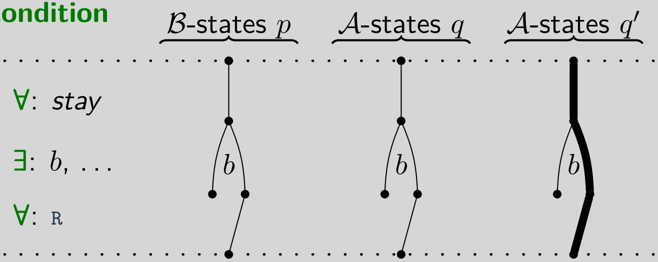


Winning condition



(WR) ∀ restarted infinitely many times

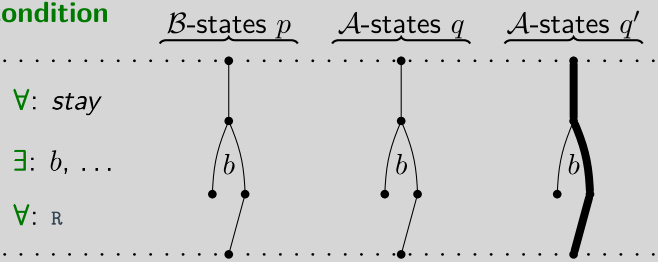
Winning condition



(WR) ∀ restarted infinitely many times

(WB) $\mathcal{B}$-states p are accepting

Winning condition

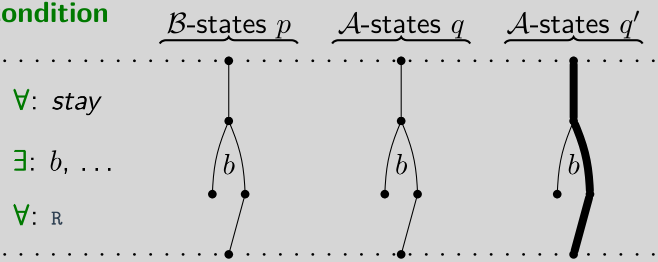


(WR) ∀ restarted infinitely many times

(WB) $\mathcal{B}$-states p are accepting

(WA) $\mathcal{A}$-states q' are accepting

Winning condition



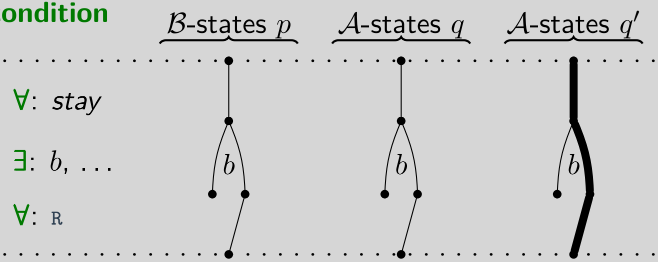
(WR) ∀ restarted infinitely many times

(WB) $\mathcal{B}$-states p are accepting

(WA) $\mathcal{A}$-states q' are accepting

$$W \equiv \big((\mathrm{WR}) \wedge (\mathrm{WB})\big) \vee \big(\neg(\mathrm{WR}) \wedge (\mathrm{WA})\big)$$

Winning condition



(WR) ∀ restarted infinitely many times

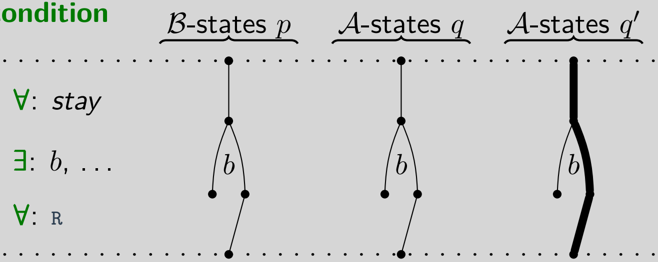
(WB) $\mathcal{B}$-states p are accepting

(WA) $\mathcal{A}$-states q' are accepting

$$W \equiv \big((\mathrm{WR}) \wedge (\mathrm{WB})\big) \vee \big(\neg(\mathrm{WR}) \wedge (\mathrm{WA})\big)$$

$\rightsquigarrow$ regular condition over infinite words

Winning condition



(WR) ∀ restarted infinitely many times

(WB) $\mathcal{B}$-states p are accepting

(WA) $\mathcal{A}$-states q' are accepting

$$W \equiv \big((\mathrm{WR}) \wedge (\mathrm{WB})\big) \vee \big(\neg(\mathrm{WR}) \wedge (\mathrm{WA})\big)$$

⇝ regular condition over infinite words

⇝ we can solve $\mathcal{F}$

Two lemmata:

Two lemmata:

1. If ∀ wins $\mathcal{F}$ then L is $\mathrm{weak{-}alt}(0,2)$-definable

Two lemmata:

1. If ∀ wins $\mathcal{F}$ then L is $\mathrm{weak{-}alt}(0,2)$-definable

Proof

Take a finite memory strategy of ∀ in $\mathcal{F}$

Two lemmata:

1. If ∀ wins $\mathcal{F}$ then L is $\mathrm{weak{-}alt}(0,2)$-definable

Proof

Take a finite memory strategy of ∀ in $\mathcal{F}$

Add some pumping

Two lemmata:

1. If ∀ wins $\mathcal{F}$ then L is $\mathrm{weak}{-}\mathrm{alt}(0,2)$-definable

Proof

Take a finite memory strategy of ∀ in $\mathcal{F}$

Add some pumping

$\rightsquigarrow$ a weak alternating $(0,2)$ automaton for L

Two lemmata:

1. If ∀ wins $\mathcal{F}$ then L is $\mathrm{weak{-}alt}(0,2)$-definable

Proof

Take a finite memory strategy of ∀ in $\mathcal{F}$

Add some pumping

$\rightsquigarrow$ a weak alternating $(0,2)$ automaton for L

$\rightsquigarrow$ $L \in \mathbf{\Pi}^0_2$

Two lemmata:

1. If ∀ wins $\mathcal{F}$ then L is $\mathrm{weak{-}alt}(0,2)$-definable

Proof

Take a finite memory strategy of ∀ in $\mathcal{F}$

Add some pumping

$\rightsquigarrow$ a weak alternating $(0,2)$ automaton for L

$\rightsquigarrow$ $L \in \mathbf{\Pi}^0_2$

2. If ∃ wins $\mathcal{F}$ then L is **not** $\mathbf{\Pi}^0_2$

Two lemmata:

1. If ∀ wins $\mathcal{F}$ then L is $\mathrm{weak{-}alt}(0,2)$-definable

Proof

Take a finite memory strategy of ∀ in $\mathcal{F}$

Add some pumping

$\rightsquigarrow$ a weak alternating $(0,2)$ automaton for L

$\rightsquigarrow$ $L \in \mathbf{\Pi}^0_2$

2. If ∃ wins $\mathcal{F}$ then L is **not** $\mathbf{\Pi}^0_2$

Proof

Take a strategy of ∃ in $\mathcal{F}$

Two lemmata:

1. If $\forall$ wins $\mathcal{F}$ then L is $\mathrm{weak{-}alt}(0,2)$-definable

Proof

Take a finite memory strategy of $\forall$ in $\mathcal{F}$

Add some pumping

$\rightsquigarrow$ a weak alternating $(0,2)$ automaton for L

$\rightsquigarrow$ $L \in \mathbf{\Pi}^0_2$

2. If $\exists$ wins $\mathcal{F}$ then L is **not** $\mathbf{\Pi}^0_2$

Proof

Take a strategy of $\exists$ in $\mathcal{F}$

Confront it with a family of quasi-strategies of $\forall$

Two lemmata:

1. If ∀ wins $\mathcal{F}$ then L is $\mathrm{weak{-}alt}(0,2)$-definable

Proof

Take a finite memory strategy of ∀ in $\mathcal{F}$

Add some pumping

$\rightsquigarrow$ a weak alternating $(0,2)$ automaton for L

$\rightsquigarrow$ $L \in \mathbf{\Pi}^0_2$

2. If ∃ wins $\mathcal{F}$ then L is **not** $\mathbf{\Pi}^0_2$

Proof

Take a strategy of ∃ in $\mathcal{F}$

Confront it with a family of quasi-strategies of ∀

$\rightsquigarrow$ a reduction proving that $L \notin \mathbf{\Pi}^0_2$

Two lemmata:

1. If ∀ wins $\mathcal{F}$ then L is $\mathrm{weak{-}alt}(0,2)$-definable

Proof

Take a finite memory strategy of ∀ in $\mathcal{F}$

Add some pumping

$\rightsquigarrow$ a weak alternating $(0,2)$ automaton for L

$\rightsquigarrow$ $L \in \mathbf{\Pi}^0_2$

2. If ∃ wins $\mathcal{F}$ then L is **not** $\mathbf{\Pi}^0_2$

Proof

Take a strategy of ∃ in $\mathcal{F}$

Confront it with a family of quasi-strategies of ∀

$\rightsquigarrow$ a reduction proving that $L \notin \mathbf{\Pi}^0_2$

$\rightsquigarrow$ L is **not** $\mathrm{weak{-}alt}(0,2)$-definable

Two lemmata:

1. If ∀ wins $\mathcal{F}$ then L is $\mathrm{weak{-}alt}(0,2)$-definable

Proof

Take a finite memory strategy of ∀ in $\mathcal{F}$

Add some pumping

$\rightsquigarrow$ a weak alternating $(0,2)$ automaton for L

$\rightsquigarrow$ $L \in \mathbf{\Pi}^0_2$

2. If ∃ wins $\mathcal{F}$ then L is **not** $\mathbf{\Pi}^0_2$

Proof

Take a strategy of ∃ in $\mathcal{F}$

Confront it with a family of quasi-strategies of ∀

$\rightsquigarrow$ a reduction proving that $L \notin \mathbf{\Pi}^0_2$

$\rightsquigarrow$ L is **not** $\mathrm{weak{-}alt}(0,2)$-definable

} A complete proof

Two lemmata:

1. If ∀ wins $\mathcal{F}$ then L is $\mathrm{weak{-}alt}(0,2)$-definable

Proof

Take a finite memory strategy of ∀ in $\mathcal{F}$

Add some pumping

$\rightsquigarrow$ a weak alternating $(0,2)$ automaton for L

$\rightsquigarrow$ $L \in \mathbf{\Pi}^0_2$

2. If ∃ wins $\mathcal{F}$ then L is **not** $\mathbf{\Pi}^0_2$

Proof

Take a strategy of ∃ in $\mathcal{F}$

Confront it with a family of quasi-strategies of ∀

$\rightsquigarrow$ a reduction proving that $L \notin \mathbf{\Pi}^0_2$

$\rightsquigarrow$ L is **not** $\mathrm{weak{-}alt}(0,2)$-definable

A complete proof **not** using properties on which the game $\mathcal{F}$ is based

Two lemmata:

1. If ∀ wins $\mathcal{F}$ then L is $\mathrm{weak{-}alt}(0,2)$-definable

Proof

Take a finite memory strategy of ∀ in $\mathcal{F}$

Add some pumping

$\rightsquigarrow$ a weak alternating $(0,2)$ automaton for L

$\rightsquigarrow$ $L \in \mathbf{\Pi}^0_2$

2. If ∃ wins $\mathcal{F}$ then L is **not** $\mathbf{\Pi}^0_2$

Proof

Take a strategy of ∃ in $\mathcal{F}$

Confront it with a family of quasi-strategies of ∀

$\rightsquigarrow$ a reduction proving that $L \notin \mathbf{\Pi}^0_2$

$\rightsquigarrow$ L is **not** $\mathrm{weak{-}alt}(0,2)$-definable

A complete proof **not** using properties on which the game $\mathcal{F}$ is based

[dealternation]

Another example

Another example

Theorem (Michalewski, Mio, S. [2017])

Given a game automaton $\mathcal{A}$ over infinite trees,

Another example

Theorem (Michalewski, Mio, S. [2017])

Given a game automaton $\mathcal{A}$ over infinite trees, one can decide if $\mathrm{L}(\mathcal{A})$ is meager.

Another example

Theorem (Michalewski, Mio, S. [2017])

Given a game automaton $\mathcal{A}$ over infinite trees,
one can decide if $\mathrm{L}(\mathcal{A})$ is meager.

Proof

Making the Banach-Mazur game regular

Another example

Theorem (Michalewski, Mio, S. [2017])

Given a game automaton $\mathcal{A}$ over infinite trees,
one can decide if $\mathrm{L}(\mathcal{A})$ is meager.

Proof

Making the Banach-Mazur game regular ■

Another example

Theorem (Michalewski, Mio, S. [2017])

Given a game automaton $\mathcal{A}$ over infinite trees,
one can decide if $\mathrm{L}(\mathcal{A})$ is meager.

Proof

Making the Banach-Mazur game regular ■

Open problem: what about general regular tree languages?

Summary

Summary

→ characterising which languages are simple

Summary

$\longrightarrow$ characterising which languages are simple

$\longrightarrow$ pattern method (rigid representatons: det. aut. / algebra)

Summary

→ characterising which languages are simple

→ pattern method (rigid representatons: det. aut. / algebra)

pattern found

⇝ L is hard

Summary

⟶ characterising which languages are simple

⟶ pattern method (rigid representatons: det. aut. / algebra)

pattern found

$\rightsquigarrow L$ is hard

pattern missing

$\rightsquigarrow L$ is simple

Summary

→ characterising which languages are simple

→ pattern method (rigid representatons: det. aut. / algebra)

pattern found

⇝ L is hard

pattern missing

⇝ L is simple

→ games (may deal with non-determinism)

Summary

→ characterising which languages are simple

→ pattern method (rigid representatons: det. aut. / algebra)

pattern found

⇝ L is hard

pattern missing

⇝ L is simple

→ games (may deal with non-determinism)

strategy of ∃

⇝ L is hard

Summary

→ characterising which languages are simple

→ pattern method (rigid representatons: det. aut. / algebra)

- pattern found
  ⇝ L is hard
- pattern missing
  ⇝ L is simple

→ games (may deal with non-determinism)

- strategy of ∃
  ⇝ L is hard
- strategy of ∀
  ⇝ L is simple

Summary

→ characterising which languages are simple

→ pattern method (rigid representatons: det. aut. / algebra)

↙ ↘

pattern found $\rightsquigarrow$ L is hard

pattern missing $\rightsquigarrow$ L is simple

→ games (may deal with non-determinism)

↙ ↘

strategy of $\exists$ $\rightsquigarrow$ L is hard

strategy of $\forall$ $\rightsquigarrow$ L is simple

→ **no** general recipe for design

Summary

→ characterising which languages are simple

→ pattern method (rigid representatons: det. aut. / algebra)

pattern found
⇝ L is hard

pattern missing
⇝ L is simple

→ games (may deal with non-determinism)

strategy of ∃
⇝ L is hard

strategy of ∀
⇝ L is simple

→ **no** general recipe for design

Conjecture: Every class of languages has a game characterisation