Varietal membership problem and alternation

Marcin Kozik

Eduard Čech Center
Charles University
Czech Republic

Algorithmics Research Group
Jagiellonian University
Poland

Algorithmic Complexity and Universal Algebra, Szeged 2007

Two problems

The universal membership problem

INPUT	two finite algebras $\mathbf{A}$ and $\mathbf{B}$
PROBLEM	decide whether $\mathbf{B} \in \mathrm{HSP}(\mathbf{A})$.

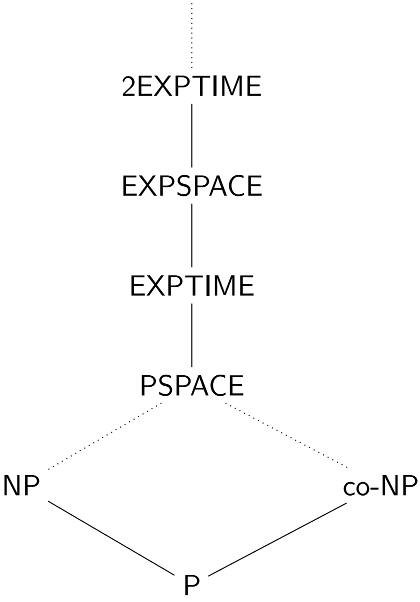
Two problems

The universal membership problem

INPUT	two finite algebras $\mathbf{A}$ and $\mathbf{B}$
PROBLEM	decide whether $\mathbf{B} \in \mathsf{HSP}(\mathbf{A})$.

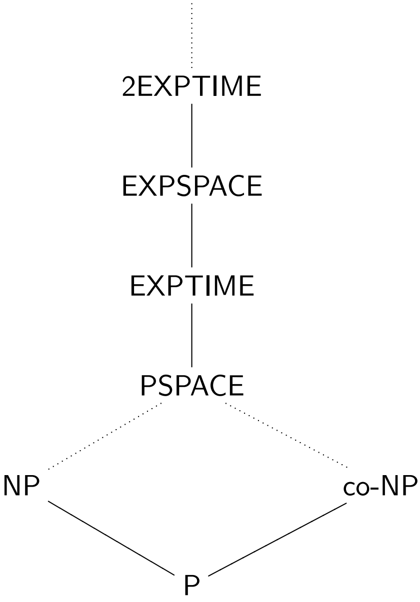
The membership problem for a fixed algebra $\mathbf{A}$

INPUT	a finite algebra $\mathbf{B}$
PROBLEM	decide whether $\mathbf{B} \in \mathsf{HSP}(\mathbf{A})$.



TIME

The amount of steps performed before accepting or rejecting the input.

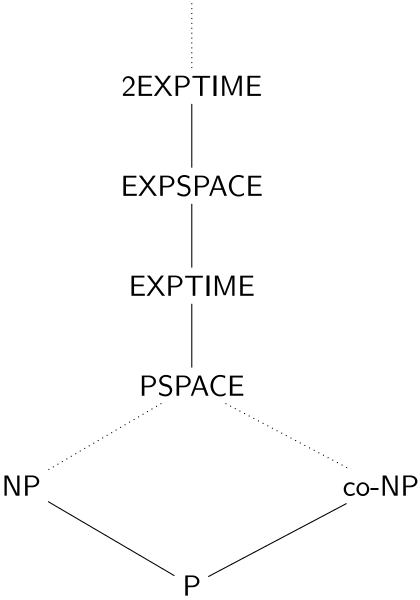


TIME

The amount of steps performed before accepting or rejecting the input.

SPACE

The amount of tape required to finish the computations.



TIME

The amount of steps performed before accepting or rejecting the input.

SPACE

The amount of tape required to finish the computations.

The rate of growth

The function bounding the amount of a resource.

The first question

Problem 2.5 [Margolis, Sapir 1995]

For every finite universal algebra $\mathbf{A}$ find the computational complexity of the membership problem for the variety generated by $\mathbf{A}$. In particular is there a finite algebra $\mathbf{A}$ for which the problem cannot be solved in polynomial time or in polynomial space?

The first question

Problem 2.5 [Margolis, Sapir 1995]

For every finite universal algebra $\mathbf{A}$ find the computational complexity of the membership problem for the variety generated by $\mathbf{A}$. In particular is there a finite algebra $\mathbf{A}$ for which the problem cannot be solved in polynomial time or in polynomial space?

Answer

We know “How hard can it be”, which answers the second part of the question as well.

The second question

The equivalence problem

INPUT	two finite algebras $\mathbf{A}$ and $\mathbf{B}$
PROBLEM	decide whether $\mathrm{HSP}(\mathbf{B}) = \mathrm{HSP}(\mathbf{A})$.

The second question

The equivalence problem

INPUT	two finite algebras $\mathbf{A}$ and $\mathbf{B}$
PROBLEM	decide whether $\mathrm{HSP}(\mathbf{B}) = \mathrm{HSP}(\mathbf{A})$.

Problem 6.8 [Bergman, Slutzki 2000]

Is the equivalence problem complete for 2EXPTIME? Is it in EXPSPACE?

The second question

The equivalence problem

INPUT	two finite algebras $\mathbf{A}$ and $\mathbf{B}$
PROBLEM	decide whether $\mathsf{HSP}(\mathbf{B}) = \mathsf{HSP}(\mathbf{A})$.

Problem 6.8 [Bergman, Slutzki 2000]

Is the equivalence problem complete for 2EXPTIME? Is it in EXPSPACE?

Answer

It is complete for 2EXPTIME.

The connected problem

The equivalence problem

INPUT	two finite algebras $\mathbf{A}$ and $\mathbf{B}$
PROBLEM	decide whether $\mathsf{HSP}(\mathbf{B}) = \mathsf{HSP}(\mathbf{A})$.

The connected problem

The equivalence problem

INPUT	two finite algebras $\mathbf{A}$ and $\mathbf{B}$
PROBLEM	decide whether $\mathrm{HSP}(\mathbf{B}) = \mathrm{HSP}(\mathbf{A})$.

The reduction

A problem $\beth$ is easier than problem $\gimel$ if every instance of problem $\beth$ can be *easily* interpreted as an instance of problem $\gimel$ having the same answer.

The connected problem

The equivalence problem

INPUT	two finite algebras $\mathbf{A}$ and $\mathbf{B}$
PROBLEM	decide whether $\mathrm{HSP}(\mathbf{B}) = \mathrm{HSP}(\mathbf{A})$.

The reduction

A problem $\beth$ is easier than problem $\gimel$ if every instance of problem $\beth$ can be *easily* interpreted as an instance of problem $\gimel$ having the same answer.

A reduction

Instead of asking whether $\mathbf{B} \in \mathrm{HSP}(\mathbf{A})$ ask whether

$$\mathrm{HSP}(\mathbf{B} \times \mathbf{A}) = \mathrm{HSP}(\mathbf{A}).$$

The upper bound

An algorithm [Bergman, Slutzki 2000]

For algebras **A** and **B** given on input let $p_0, \ldots, p_{|B|-1} \subseteq A^{A^{|B|}}$ denote the projections

- let φ be a function from $p_0, \ldots, p_{|B|-1}$ onto B

The upper bound

An algorithm [Bergman, Slutzki 2000]

For algebras **A** and **B** given on input let $p_0, \ldots, p_{|B|-1} \subseteq A^{A^{|B|}}$ denote the projections

- let φ be a function from $p_0, \ldots, p_{|B|-1}$ onto B
- REPEAT
  for a basic operation $t(\bar{x})$ and elements $\bar{a}$ from the domain of φ

  UNTIL φ is correctly extended to the subalgebra of $\mathbf{A}^{A^{|B|}}$

The upper bound

An algorithm [Bergman, Slutzki 2000]

For algebras **A** and **B** given on input let $p_0, \ldots, p_{|B|-1} \subseteq A^{A^{|B|}}$ denote the projections

- let φ be a function from $p_0, \ldots, p_{|B|-1}$ onto B
- REPEAT
  for a basic operation $t(\bar{x})$ and elements $\bar{a}$ from the domain of φ
  - if $t(\bar{a})$ is not in a domain of φ then extend φ,

  UNTIL φ is correctly extended to the subalgebra of $\mathbf{A}^{A^{|B|}}$

The upper bound

An algorithm [Bergman, Slutzki 2000]

For algebras $\mathbf{A}$ and $\mathbf{B}$ given on input let $p_0, \ldots, p_{|B|-1} \subseteq A^{A^{|B|}}$ denote the projections

- let φ be a function from $p_0, \ldots, p_{|B|-1}$ onto B
- REPEAT
  for a basic operation $t(\bar{x})$ and elements $\bar{a}$ from the domain of φ
  - if $t(\bar{a})$ is not in a domain of φ then extend φ,
  - if $\varphi(t(\bar{a})) \neq t(\varphi(\bar{a}))$ RETURN $\mathbf{B} \notin \mathrm{HSP}(\mathbf{A})$.

  UNTIL φ is correctly extended to the subalgebra of $\mathbf{A}^{A^{|B|}}$

The upper bound

An algorithm [Bergman, Slutzki 2000]

For algebras $\mathbf{A}$ and $\mathbf{B}$ given on input let $p_0, \ldots, p_{|B|-1} \subseteq A^{A^{|B|}}$ denote the projections

- let φ be a function from $p_0, \ldots, p_{|B|-1}$ onto B
- REPEAT
  for a basic operation $t(\bar{x})$ and elements $\bar{a}$ from the domain of φ
  - if $t(\bar{a})$ is not in a domain of φ then extend φ,
  - if $\varphi(t(\bar{a})) \neq t(\varphi(\bar{a}))$ RETURN $\mathbf{B} \notin \mathrm{HSP}(\mathbf{A})$.

  UNTIL φ is correctly extended to the subalgebra of $\mathbf{A}^{A^{|B|}}$
- RETURN $\mathbf{B} \in \mathrm{HSP}(\mathbf{A})$.

The upper bound

An algorithm [Bergman, Slutzki 2000]

For algebras **A** and **B** given on input let $p_0, \ldots, p_{|B|-1} \subseteq A^{A^{|B|}}$ denote the projections

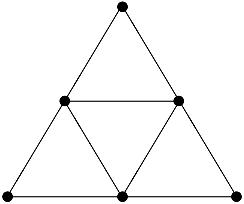
- let φ be a function from $p_0, \ldots, p_{|B|-1}$ onto B
- REPEAT
  for a basic operation $t(\bar{x})$ and elements $\bar{a}$ from the domain of φ
  - if $t(\bar{a})$ is not in a domain of φ then extend φ,
  - if $\varphi(t(\bar{a})) \neq t(\varphi(\bar{a}))$ RETURN $\mathbf{B} \notin \mathrm{HSP}(\mathbf{A})$.

  UNTIL φ is correctly extended to the subalgebra of $\mathbf{A}^{A^{|B|}}$
- RETURN $\mathbf{B} \in \mathrm{HSP}(\mathbf{A})$.

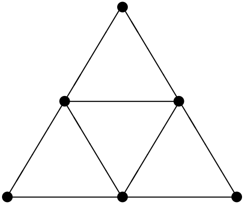
The complexity

The algorithm works in doubly exponential time.

Some lower bounds



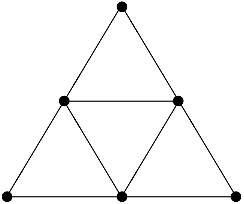
Some lower bounds



Székely 1998

The flat graph algebra of this graph generates a variety with NP-complete membership problem.

Some lower bounds



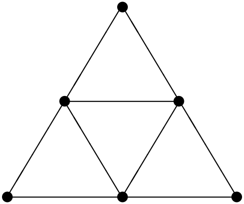
Székely 1998

The flat graph algebra of this graph generates a variety with NP-complete membership problem.

Kozik, Kun 2005

Same for graph algebra.

Some lower bounds



Székely 1998

The flat graph algebra of this graph generates a variety with NP-complete membership problem.

Kozik, Kun 2005

Same for graph algebra.

Jackson, McKenzie 2005

A semigroup (based on this graph) generating a variety with NP-hard membership problem.

Turing machines

A simple Turing machine

- states of the Turing machine are α and β

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

An instruction $\alpha\,0\,1\,\beta\,R$

- in state α

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

An instruction $\alpha\,0\,1\,\beta\,R$

- in state α
- reading 0

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

An instruction $\alpha\,0\,1\,\beta\,R$

- in state α
- reading 0
- write 1

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

An instruction $\alpha\,0\,1\,\beta\,R$

- in state α
- reading 0
- write 1
- change state to β

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

An instruction $\alpha\,0\,1\,\beta\,R$

- in state α
- reading 0
- write 1
- change state to β
- and move R right.

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

0	0	0	0	0	0	0	0	0	0
				$\uparrow$					
				α					

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

0	0	0	0	1	0	0	0	0	0
					$\uparrow$				
					β				

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

0	0	0	0	1	1	0	0	0	0
				$\uparrow$					
				α					

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\, 0\, 1\, \beta\, R$, $\beta\, 1\, 1\, \beta\, R$, $\beta\, 0\, 1\, \alpha\, L$, $\alpha\, 1\, 1\, \alpha\, L$.

0	0	0	0	1	1	0	0	0	0

$$\begin{array}{cccccccccc} & & & \uparrow & & & & & & \\ & & & \alpha & & & & & & \end{array}$$

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\, 0\, 1\, \beta\, R$, $\beta\, 1\, 1\, \beta\, R$, $\beta\, 0\, 1\, \alpha\, L$, $\alpha\, 1\, 1\, \alpha\, L$.

0	0	0	1	1	1	0	0	0	0
				$\uparrow$					
				β					

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\, 0\, 1\, \beta\, R$, $\beta\, 1\, 1\, \beta\, R$, $\beta\, 0\, 1\, \alpha\, L$, $\alpha\, 1\, 1\, \alpha\, L$.

0	0	0	1	1	1	0	0	0	0
					$\uparrow$				
					β				

Turing machines

A simple Turing machine

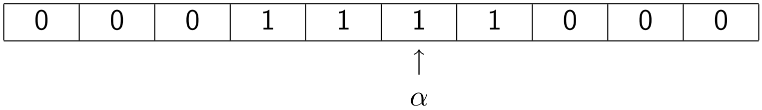
- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

0	0	0	1	1	1	0	0	0	0
						$\uparrow$			
						β			

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.



Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

0	0	0	1	1	1	1	0	0	0
				$\uparrow$					
				α					

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

0	0	0	1	1	1	1	0	0	0

$\uparrow$

α

Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$.

0	0	0	1	1	1	1	0	0	0

$\uparrow$

α

McKenzie's $\mathbf{A}(\mathbf{T})$ (part I)

Ideas

McKenzie's $\mathbf{A}(\mathbf{T})$ (part I)

Ideas

- An element of $\mathbf{A}(\mathbf{T})^n$ can encode
  - a *tape* of length n.
  - a position of a head on a tape.
  - an internal state of the machine.

McKenzie's $\mathbf{A}(\mathbf{T})$ (part I)

Ideas

- An element of $\mathbf{A}(\mathbf{T})^n$ can encode a configuration of the machine.

McKenzie's $\mathbf{A}(\mathbf{T})$ (part I)

Ideas

- An element of $\mathbf{A}(\mathbf{T})^n$ can encode a configuration of the machine.

Problems

- The coordinates are indistinguishable.

McKenzie's $\mathbf{A}(\mathbf{T})$ (part I)

Ideas

- An element of $\mathbf{A}(\mathbf{T})^n$ can encode a configuration of the machine.
- Introduce special elements of $\mathbf{A}(\mathbf{T})^n$ acting as *markers* for different coordinates
$$(0, \ldots, 0, H, 0, \ldots, 0) \in A(\mathbf{T})$$

Problems

- The coordinates are indistinguishable.

McKenzie's $\mathbf{A}(\mathbf{T})$ (part I)

Ideas

- An element of $\mathbf{A}(\mathbf{T})^n$ can encode a configuration of the machine.
- Introduce special elements of $\mathbf{A}(\mathbf{T})^n$ acting as *markers* for different coordinates

$$(0, \ldots, 0, H, 0, \ldots, 0) \in A(\mathbf{T})$$

Problems

- The coordinates are indistinguishable.
- The coordinates are unordered.

McKenzie's $\mathbf{A}(\mathbf{T})$ (part I)

Ideas

- An element of $\mathbf{A}(\mathbf{T})^n$ can encode a configuration of the machine.
- Introduce special elements of $\mathbf{A}(\mathbf{T})^n$ acting as *markers* for different coordinates
- We need to know "left" from "right" among markers.

Problems

- The coordinates are indistinguishable.
- The coordinates are unordered.

McKenzie's $\mathbf{A}(\mathbf{T})$ (part I)

Ideas

- An element of $\mathbf{A}(\mathbf{T})^n$ can encode a configuration of the machine.
- Introduce special elements of $\mathbf{A}(\mathbf{T})^n$ acting as *markers* for different coordinates
- We need to know "left" from "right" among markers.

Problems

- The coordinates are indistinguishable.
- The coordinates are unordered.
- It cannot be done.

McKenzie's $\mathbf{A}(\mathbf{T})$ (part I)

Ideas

- An element of $\mathbf{A}(\mathbf{T})^n$ can encode a configuration of the machine.
- Introduce special elements of $\mathbf{A}(\mathbf{T})^n$ acting as *markers* for different coordinates
- We need to know "left" from "right" among markers.

Solution

The *markers* are linearly ordered in subalgebras of $\mathbf{A}(\mathbf{T})^n$ having subdirectly irreducible homomorphic images.

McKenzie's $\mathbf{A}(\mathbf{T})$ (part I)

Ideas

- An element of $\mathbf{A}(\mathbf{T})^n$ can encode a configuration of the machine.
- Introduce special elements of $\mathbf{A}(\mathbf{T})^n$ acting as *markers* for different coordinates
- We need to know "left" from "right" among markers.
- A configuration and markers produce the next configuration.

Solution

The *markers* are linearly ordered in subalgebras of $\mathbf{A}(\mathbf{T})^n$ having subdirectly irreducible homomorphic images.

McKenzie's $\mathbf{A}(\mathbf{T})$ (part II)

The construction

- Start with $\mathbf{A}(\mathbf{T})^n \geq \mathbf{B} \xrightarrow{\text{onto}} \mathbf{S}$ where $\mathbf{S}$ is subdirectly irreducible.

McKenzie's $\mathbf{A}(\mathbf{T})$ (part II)

The construction

- Start with $\mathbf{A}(\mathbf{T})^n \geq \mathbf{B} \xrightarrow{\text{onto}} \mathbf{S}$ where $\mathbf{S}$ is subdirectly irreducible.
- The algebra $\mathbf{B}$ contains *markers* and some configuration of the Turing machine $\mathbf{T}$ (determined by the algebra $\mathbf{S}$).

McKenzie's $\mathbf{A}(\mathbf{T})$ (part II)

The construction

- Start with $\mathbf{A}(\mathbf{T})^n \geq \mathbf{B} \xrightarrow{\text{onto}} \mathbf{S}$ where $\mathbf{S}$ is subdirectly irreducible.
- The algebra $\mathbf{B}$ contains *markers* and some configuration of the Turing machine $\mathbf{T}$ (determined by the algebra $\mathbf{S}$).
- The algebra $\mathbf{B}$ contains the whole computation starting at this configuration.

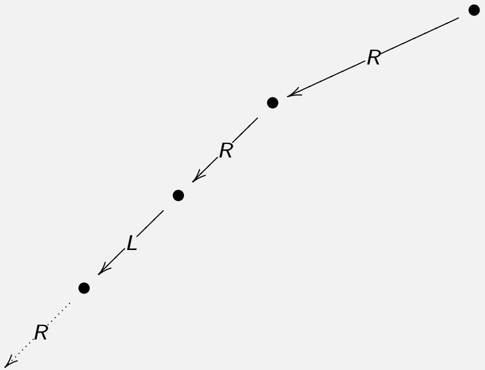
McKenzie's $\mathbf{A}(\mathbf{T})$ (part II)

The construction

- Start with $\mathbf{A}(\mathbf{T})^n \geq \mathbf{B} \xrightarrow{\text{onto}} \mathbf{S}$ where $\mathbf{S}$ is subdirectly irreducible.
- The algebra $\mathbf{B}$ contains *markers* and some configuration of the Turing machine $\mathbf{T}$ (determined by the algebra $\mathbf{S}$).
- The algebra $\mathbf{B}$ contains the whole computation starting at this configuration.
- If the computation ends in an accepting state then $\mathbf{S}$ is not subdirectly irreducible.

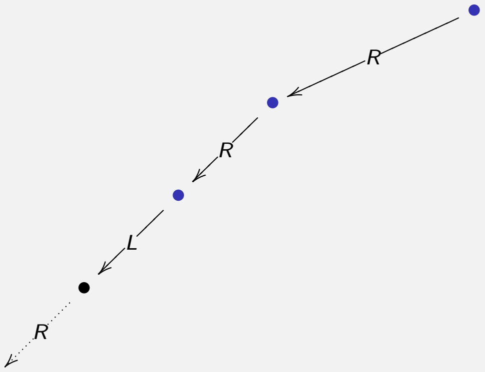
The McKenzie's $\mathbf{A}(\mathbf{T})$ and beyond

In the algebra $\mathbf{B}$ such that $\mathbf{A}(\mathbf{T})^n \geq \mathbf{B} \xrightarrow{\text{onto}} \mathbf{S}$



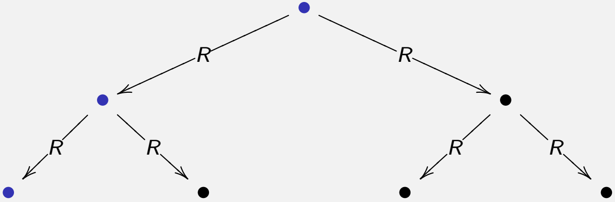
The McKenzie's $\mathbf{A}(\mathbf{T})$ and beyond

In the algebra $\mathbf{B}$ such that $\mathbf{A}(\mathbf{T})^n \geq \mathbf{B} \xrightarrow{\text{onto}} \mathbf{S}$



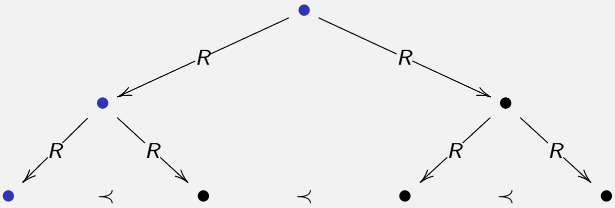
The McKenzie's $\mathbf{A}(\mathbf{T})$ and beyond

In the algebra $\mathbf{B}$ such that $\mathbf{A}(\mathbf{T})^n \geq \mathbf{B} \xrightarrow{\text{onto}} \mathbf{S}$



The McKenzie's $\mathbf{A}(\mathbf{T})$ and beyond

In the algebra $\mathbf{B}$ such that $\mathbf{A}(\mathbf{T})^n \geq \mathbf{B} \xrightarrow{\text{onto}} \mathbf{S}$



Main differences

McKenzie's $\mathbf{A}(\mathbf{T})$	A modification of $\mathbf{A}(\mathbf{T})$
▶ polynomial tape	

Main differences

McKenzie's $\mathbf{A}(\mathbf{T})$

- polynomial tape

A modification of $\mathbf{A}(\mathbf{T})$

- exponential tape

Main differences

McKenzie's $\mathbf{A}(\mathbf{T})$

- polynomial tape
- blank tape

A modification of $\mathbf{A}(\mathbf{T})$

- exponential tape

Main differences

McKenzie's $\mathbf{A}(\mathbf{T})$

- polynomial tape
- blank tape

A modification of $\mathbf{A}(\mathbf{T})$

- exponential tape
- tape with the input word

Main differences

McKenzie's $\mathbf{A}(\mathbf{T})$

- polynomial tape
- blank tape
- control over finite subdirectly irreducible algebras

A modification of $\mathbf{A}(\mathbf{T})$

- exponential tape
- tape with the input word

Main differences

McKenzie's $\mathbf{A}(\mathbf{T})$

- polynomial tape
- blank tape
- control over finite subdirectly irreducible algebras

A modification of $\mathbf{A}(\mathbf{T})$

- exponential tape
- tape with the input word
- control over sufficiently big set of algebras

Non-deterministic Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$,

Facts

Non-deterministic Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$, $\alpha\,1\,1\,(\mathrm{REJECT})\,L$, $\beta\,1\,1\,(\mathrm{ACCEPT})\,R$.

Facts

Non-deterministic Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\,0\,1\,\beta\,R$, $\beta\,1\,1\,\beta\,R$, $\beta\,0\,1\,\alpha\,L$, $\alpha\,1\,1\,\alpha\,L$, $\alpha\,1\,1\,(\mathrm{REJECT})\,L$, $\beta\,1\,1\,(\mathrm{ACCEPT})\,R$.

Facts

- The machine is non-deterministic.

Non-deterministic Turing machines

A simple Turing machine

- states of the Turing machine are α and β
- the operations of the Turing machine are $\alpha\, 0\, 1\, \beta\, R$, $\beta\, 1\, 1\, \beta\, R$, $\beta\, 0\, 1\, \alpha\, L$, $\alpha\, 1\, 1\, \alpha\, L$, $\alpha\, 1\, 1\, (\mathrm{REJECT})\, L$, $\beta\, 1\, 1\, (\mathrm{ACCEPT})\, R$.

Facts

- The machine is non-deterministic.
- The machine can accept and reject the same input.

NP and co-NP

Accepting

The machines are working in polynomial time and

- for NP: “ACCEPT if at least one computation ACCEPTS”.

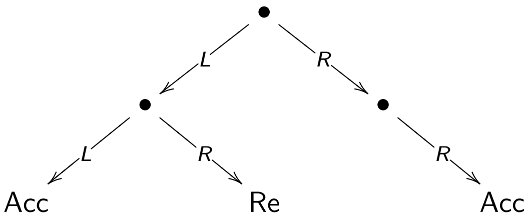
NP and co-NP

Accepting

The machines are working in polynomial time and

- for NP: "ACCEPT if at least one computation ACCEPTS".
- for co-NP: "ACCEPT if all computations ACCEPT".

NP and co-NP

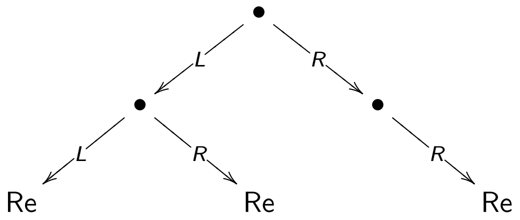


Accepting

The machines are working in polynomial time and

- for NP: “ACCEPT if at least one computation ACCEPTS”.
- for co-NP: “ACCEPT if all computations ACCEPT”.

NP and co-NP

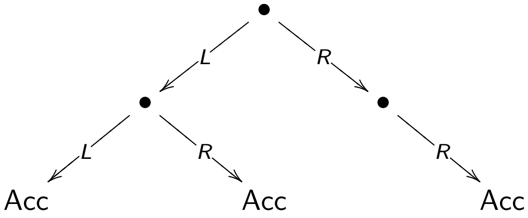


Accepting

The machines are working in polynomial time and

- for NP: "ACCEPT if at least one computation ACCEPTS".
- for co-NP: "ACCEPT if all computations ACCEPT".

NP and co-NP



Accepting

The machines are working in polynomial time and

- for NP: “ACCEPT if at least one computation ACCEPTS”.
- for co-NP: “ACCEPT if all computations ACCEPT”.

Alternating Turing machines

Accepting

A configuration is accepting if

- the state is ACCEPT,

Accepting

A configuration is accepting if

- the state is ACCEPT,
- it is of the type $\exists$ and there is *next* accepting configuration,

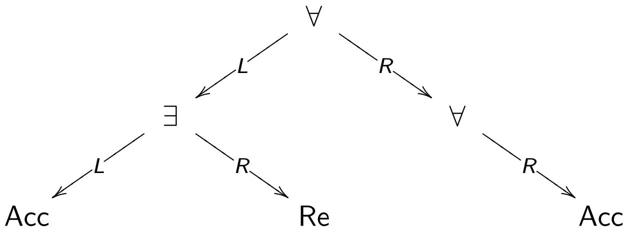
Alternating Turing machines

Accepting

A configuration is accepting if

- the state is ACCEPT,
- it is of the type $\exists$ and there is *next* accepting configuration,
- it is of the type $\forall$ and all *next* configurations are accepting.

Alternating Turing machines

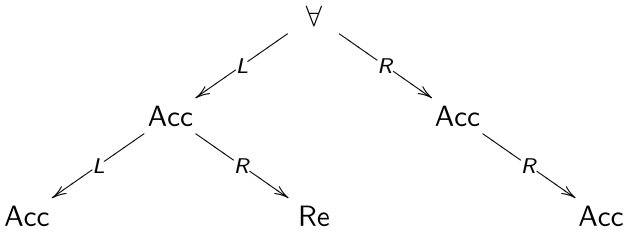


Accepting

A configuration is accepting if

- the state is ACCEPT,
- it is of the type $\exists$ and there is *next* accepting configuration,
- it is of the type $\forall$ and all *next* configurations are accepting.

Alternating Turing machines

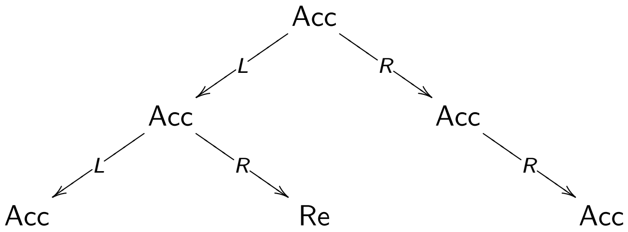


Accepting

A configuration is accepting if

- the state is ACCEPT,
- it is of the type $\exists$ and there is *next* accepting configuration,
- it is of the type $\forall$ and all *next* configurations are accepting.

Alternating Turing machines

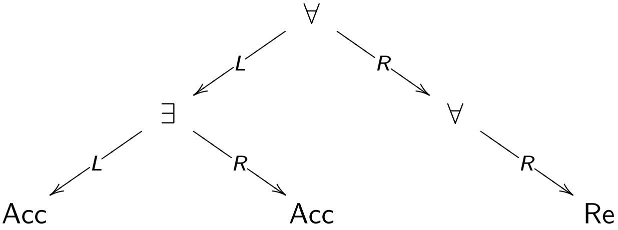


Accepting

A configuration is accepting if

- the state is ACCEPT,
- it is of the type $\exists$ and there is *next* accepting configuration,
- it is of the type $\forall$ and all *next* configurations are accepting.

Alternating Turing machines

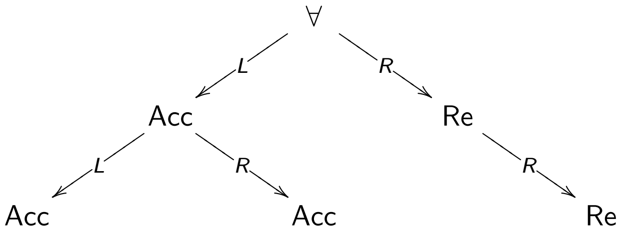


Accepting

A configuration is accepting if

- the state is ACCEPT,
- it is of the type $\exists$ and there is *next* accepting configuration,
- it is of the type $\forall$ and all *next* configurations are accepting.

Alternating Turing machines

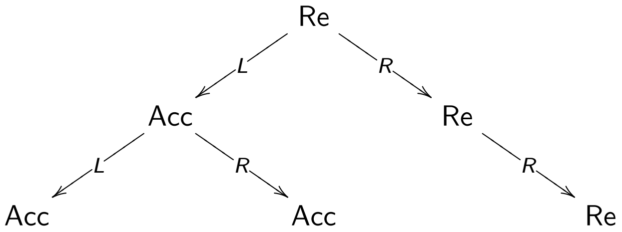


Accepting

A configuration is accepting if

- the state is ACCEPT,
- it is of the type $\exists$ and there is *next* accepting configuration,
- it is of the type $\forall$ and all *next* configurations are accepting.

Alternating Turing machines



Accepting

A configuration is accepting if

- the state is ACCEPT,
- it is of the type $\exists$ and there is *next* accepting configuration,
- it is of the type $\forall$ and all *next* configurations are accepting.

Some news

A bad news [Savitch 1970]

EXPSPACE = NEXPSPACE

Some news

A bad news [Savitch 1970]

EXPSPACE = NEXPSPACE

A good news [Chandra, Kozen, Stockmeyer 1981]

AEXPSPACE = 2EXPTIME

Some news

A bad news [Savitch 1970]

EXPSPACE = NEXPSPACE

A good news [Chandra, Kozen, Stockmeyer 1981]

AEXPSPACE = 2EXPTIME

Theorem

There exists a finite algebra generating a variety with a 2EXPTIME complete membership problem.