Positive logic is not elementary

Daria Walukiewicz-Chrząszcz
University of Warsaw

(joint work with Aleksy Schubert and Paweł Urzyczyn)

Highlights 2013
Paris, September 19, 2013

First-order intuitionistic logic

First-order intuitionistic logic

In intuitionistic logic:

First-order intuitionistic logic

In intuitionistic logic:

Truth defined by provability

First-order intuitionistic logic

In intuitionistic logic:

Truth defined by provability

Classical laws like: excluded middle, double negation elimination or De Morgans's laws are not valid

First-order intuitionistic logic

In intuitionistic logic:

Truth defined by provability

Classical laws like: excluded middle, double negation elimination or De Morgans's laws are not valid

There is no prenex normal form

First-order intuitionistic logic

In intuitionistic logic:

Truth defined by provability

Classical laws like: excluded middle, double negation elimination or De Morgans's laws are not valid

There is no prenex normal form

In classical logic:

Every formula is classically equivalent to one of the form:

$$Q_1 x_1 \, Q_2 x_2 \dots Q_k x_k . \, Body(x_, x_2, \dots, x_k),$$

where *Body* has no quantifiers

The language we study

First-order formulas

- with $\forall$ and $\rightarrow$
- without function symbols

The language we study

First-order formulas

- with $\forall$ and $\rightarrow$
- without function symbols

This fragment is known to be undecidable

Classification

In classical logic:

Classification

In classical logic:

Formulas may be classified according to the quantifier prefix

"The Classical Decision Problem" by E.Börger , E.Grädel , Y.Gurevich

Classification

In classical logic:

Formulas may be classified according to the quantifier prefix

"The Classical Decision Problem" by E.Börger , E.Grädel , Y.Gurevich

In intuitionistic logic:

Classification

In classical logic:

Formulas may be classified according to the quantifier prefix

"The Classical Decision Problem" by E.Börger , E.Grädel , Y.Gurevich

In intuitionistic logic:

The prenex fragment is much weaker (Pspace-complete)

Classification

In classical logic:

Formulas may be classified according to the quantifier prefix

"The Classical Decision Problem" by E.Börger , E.Grädel , Y.Gurevich

In intuitionistic logic:

The prenex fragment is much weaker (Pspace-complete)

Mints hierarchy (1968): consider the quantifier prefix
a formula *would get*, if classically normalized

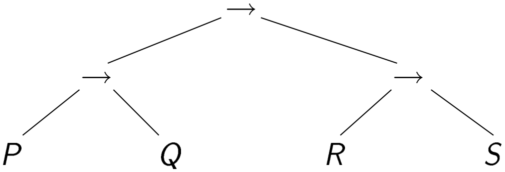
$\circ$

Positive first-order logic

$\forall$ quantifiers occurring at *positive* positions will remain $\forall$ in the prefix

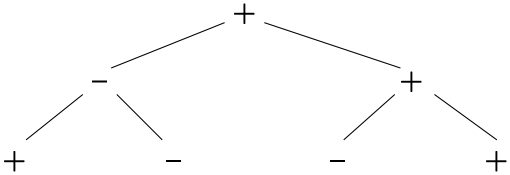
Positive first-order logic

$\forall$ quantifiers occurring at *positive* positions will remain $\forall$ in the prefix



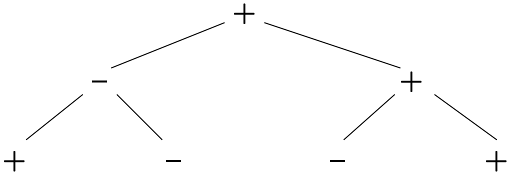
Positive first-order logic

$\forall$ quantifiers occurring at *positive* positions will remain $\forall$ in the prefix



Positive first-order logic

$\forall$ quantifiers occurring at *positive* positions will remain $\forall$ in the prefix

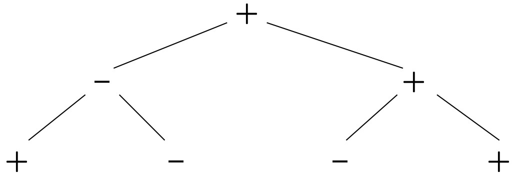


Examples:

$$(\forall x P(x)) \to Q \qquad \text{not positive}$$

Positive first-order logic

$\forall$ quantifiers occurring at *positive* positions will remain $\forall$ in the prefix



Examples:

$$\begin{array}{ll} (\forall x P(x)) \to Q & \text{not positive} \\ ((\forall x P(x)) \to Q) \to R & \text{positive} \end{array}$$

Positive first-order intuitionistic logic is decidable:
Mints (1968), Dowek, Jiang (2006), Rummelhoff (2007)

Proof construction seen as automaton

Proof construction seen as automaton

$$\Gamma \vdash \varphi$$

where $\Gamma \rightarrow \varphi$ is positive.

Proof construction seen as automaton

$$\Gamma \vdash \varphi$$

where $\Gamma \rightarrow \varphi$ is positive.

$$\text{Proofs} \quad \approx \quad \text{Automaton}$$

Proof construction seen as automaton

$$\Gamma \vdash \varphi$$

where $\Gamma \to \varphi$ is positive.

$$\begin{array}{rcl} \text{Proofs} & \approx & \text{Automaton} \\ \varphi & \approx & \text{state} \end{array}$$

Proof construction seen as automaton

$$\Gamma \vdash \varphi$$

where $\Gamma \rightarrow \varphi$ is positive.

$$\begin{array}{rcl} \text{Proofs} & \approx & \text{Automaton} \\ \varphi & \approx & \text{state} \\ \Gamma & \approx & \text{memory} \end{array}$$

Proof construction seen as automaton

$$\Gamma \vdash \varphi$$

where $\Gamma \rightarrow \varphi$ is positive.

$$\begin{aligned} \text{Proofs} \quad &\approx \quad \text{Automaton} \\ \varphi \quad &\approx \quad \text{state} \\ \Gamma \quad &\approx \quad \text{memory} \\ \text{proof-search process} \quad &\approx \quad \text{computation} \end{aligned}$$

Proof construction seen as automaton

$$\Gamma \vdash \varphi$$

where $\Gamma \to \varphi$ is positive.

Proofs	$\approx$	Automaton
φ	$\approx$	state
Γ	$\approx$	memory
proof-search process	$\approx$	computation
proof completed	$\approx$	accept

Eden automata

Memory: tree of knowledge:

Eden automata

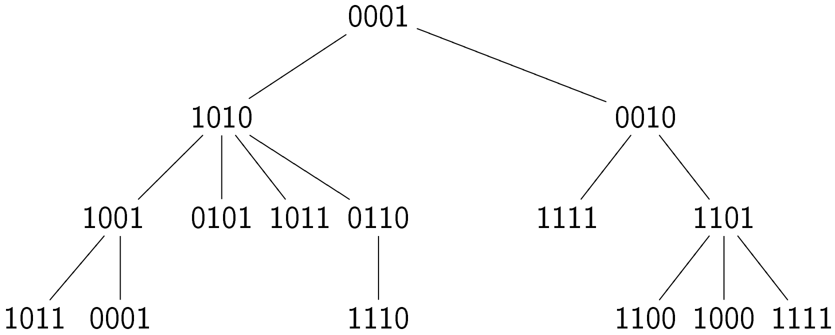
Memory: tree of knowledge:

- bounded depth, unbounded width;
- fixed number of registers in each node;

Eden automata

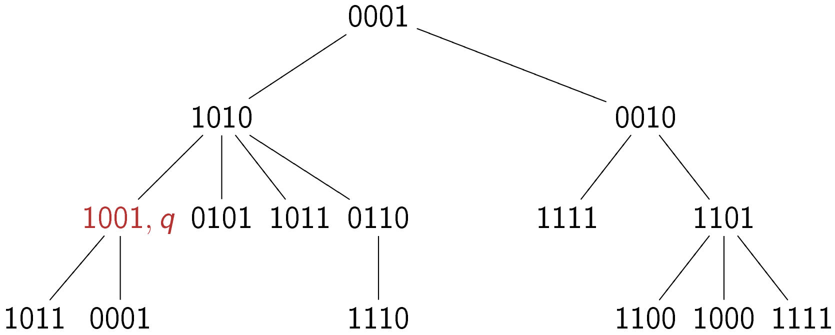
Memory: tree of knowledge:

- bounded depth, unbounded width;
- fixed number of registers in each node;



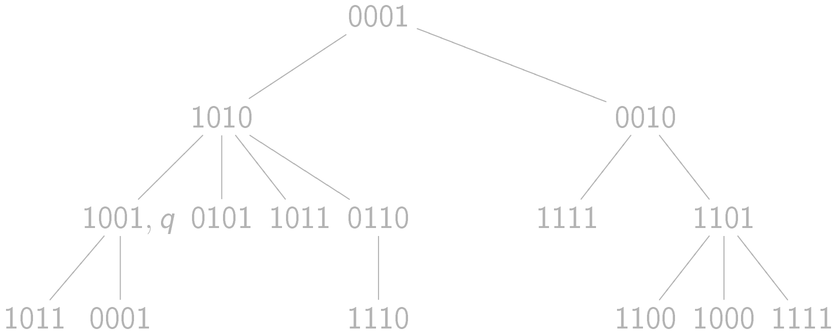
Eden automata

Automata configuration: <tree, node, state>

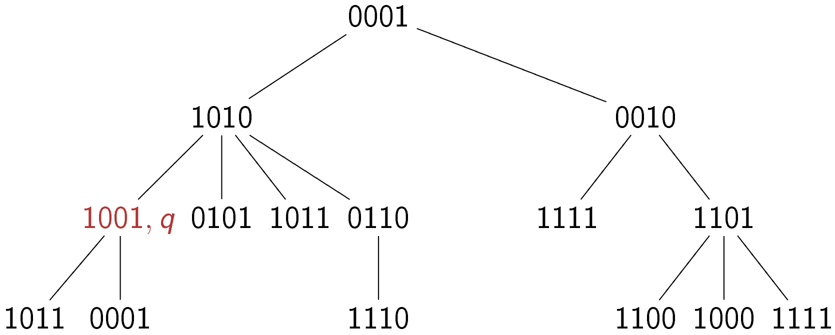


Eden automata

No input, initially one node 0000, q_I

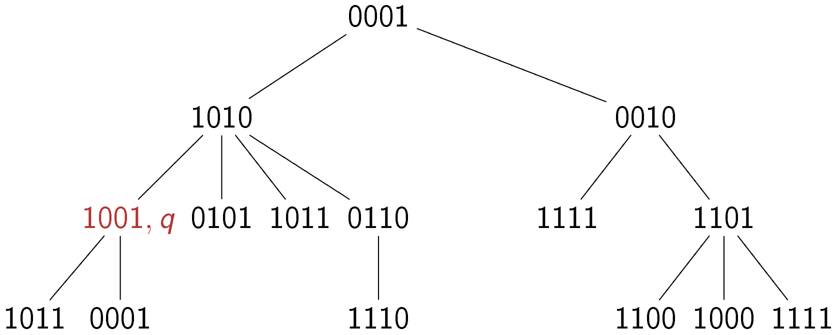


Eden automata—instructions



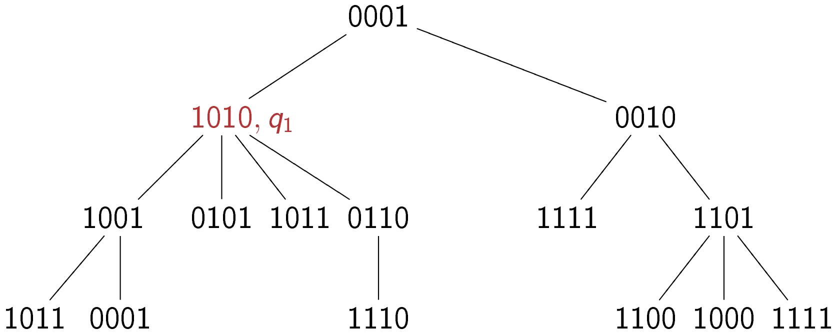
Eden automata—instructions

Move up to the father



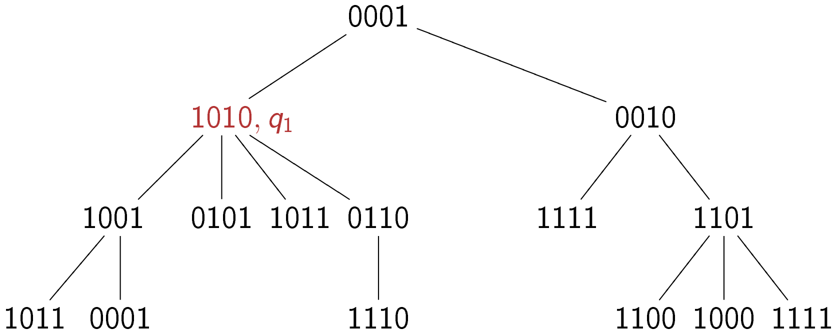
Eden automata—instructions

Move up to the father



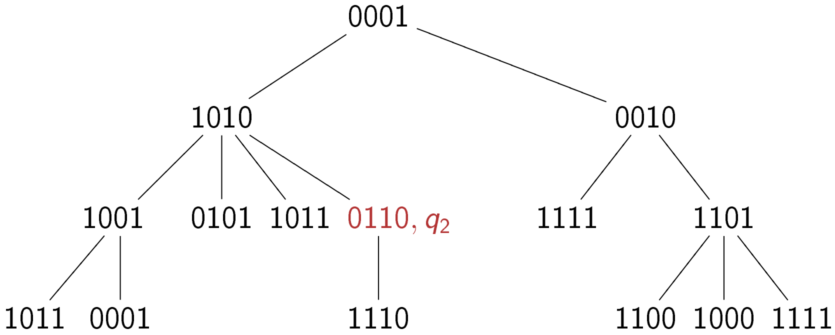
Eden automata—instructions

Move down to a son (chosen nondeterministically)



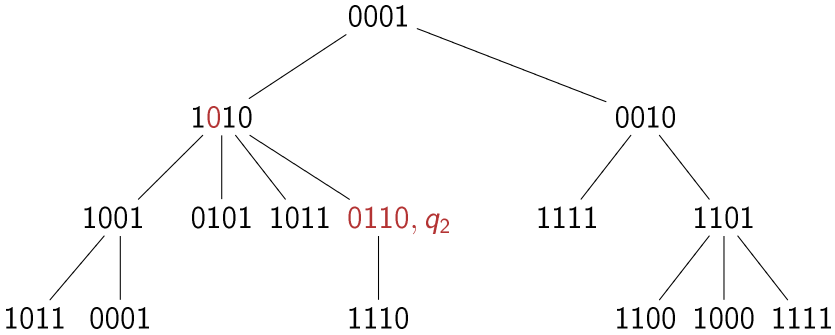
Eden automata—instructions

Move down to a son (chosen nondeterministically)



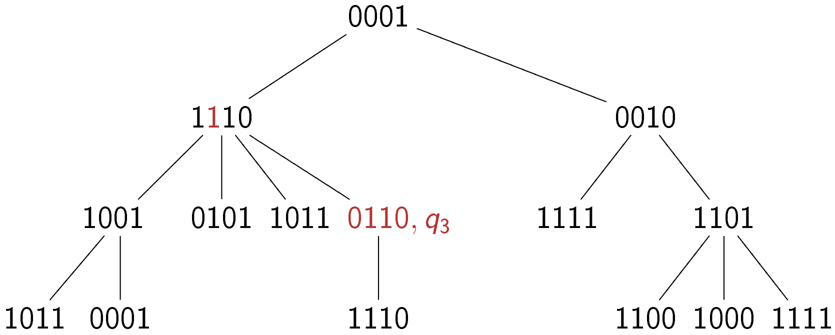
Eden automata—instructions

Write “1” in a given register at a given ancestor



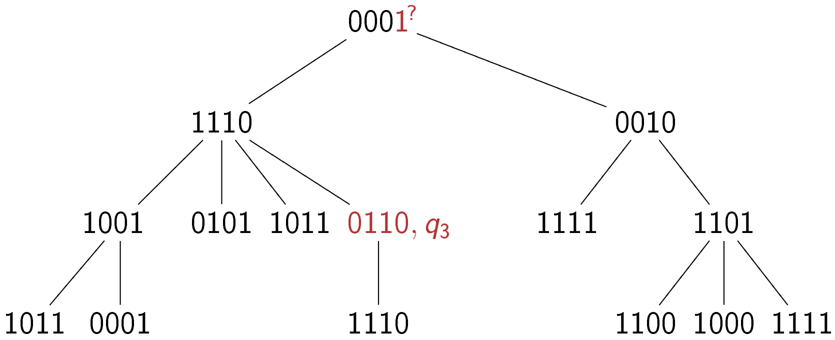
Eden automata—instructions

Write “1” in a given register at a given ancestor



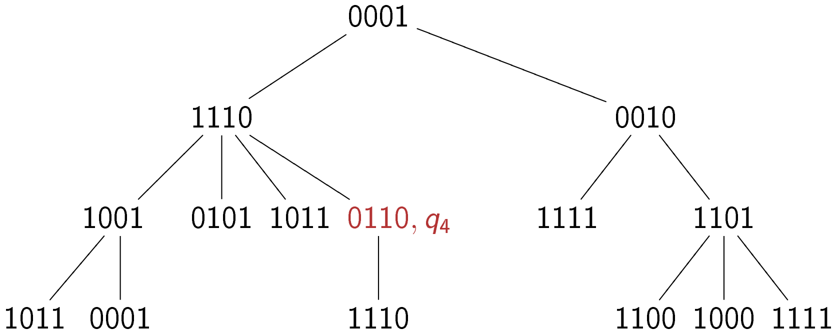
Eden automata—instructions

Check if a given register is "1" at a given ancestor



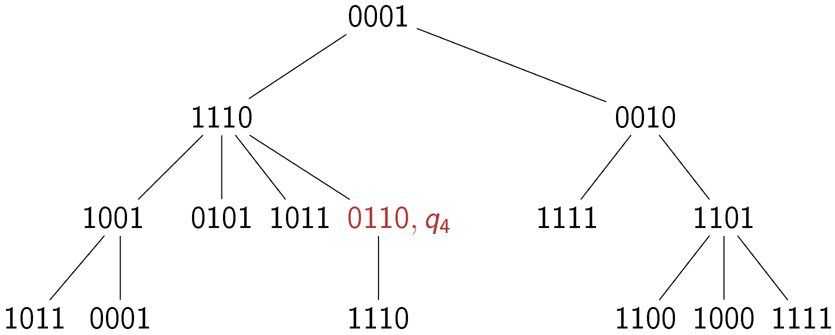
Eden automata—instructions

Check if a given register is “1” at a given ancestor



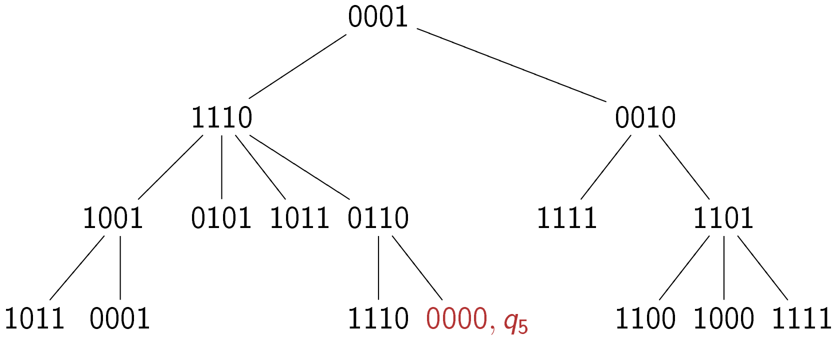
Eden automata—instructions

Create a new child and move there



Eden automata—instructions

Create a new child and move there



Eden automata—limitations

Non-erasing writes:

Eden automata—limitations

Non-erasing writes: one cannot set a register to “0”

Eden automata—limitations

Non-erasing writes: one cannot set a register to “0”

Positive reads:

Eden automata—limitations

Non-erasing writes: one cannot set a register to “0”

Positive reads: one cannot check that a register is “0”

Eden automata—limitations

Non-erasing writes: one cannot set a register to “0”

Positive reads: one cannot check that a register is “0”

Blind-access to children:

Eden automata—limitations

Non-erasing writes: one cannot set a register to “0”

Positive reads: one cannot check that a register is “0”

Blind-access to children: one enters a random child (nondeterminism)

Eden automata—limitations

Non-erasing writes: one cannot set a register to "0"

Positive reads: one cannot check that a register is "0"

Blind-access to children: one enters a random child (nondeterminism)

Bounded memory:

Eden automata—limitations

Non-erasing writes: one cannot set a register to "0"

Positive reads: one cannot check that a register is "0"

Blind-access to children: one enters a random child (nondeterminism)

Bounded memory: one can access only bounded memory, even though the tree is unbounded

Alternation

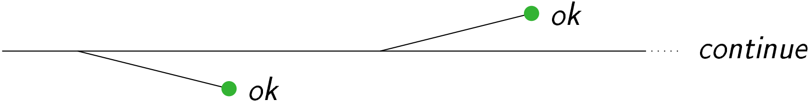
Alternation

Computation is a tree

Alternation

Computation is a tree

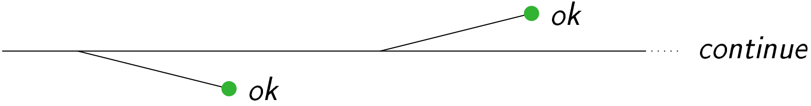
Typical use



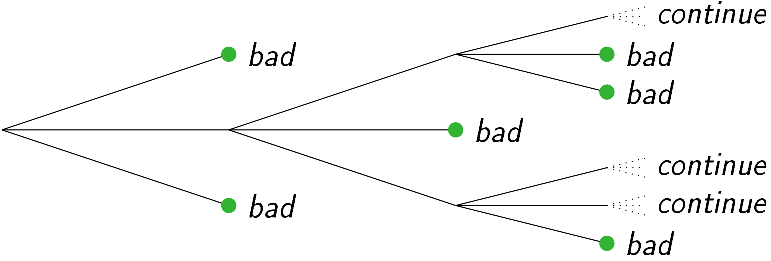
Alternation

Computation is a tree

Typical use



New (for us) use:



Contributions

Thm. 1: *Any deterministic Turing Machine working in elementary time can be simulated by an Eden automaton.*

Contributions

Thm. 1: *Any deterministic Turing Machine working in elementary time can be simulated by an Eden automaton.*

Thm. 2: *The halting problem for Eden Automata is logspace-reducible to positive logic.*

Contributions

Thm. 1: *Any deterministic Turing Machine working in elementary time can be simulated by an Eden automaton.*

Thm. 2: *The halting problem for Eden Automata is logspace-reducible to positive logic.*

Moral: *The positive logic is not elementary.*

Lessons learned

- Automata are good in proof theory

Lessons learned

- Automata are good in proof theory
- Universal loops are useful

Lessons learned

- Automata are good in proof theory
- Universal loops are useful
- Positive logic is extremely strong