Structures Defined by Higher-Order Recursion Schemes

Arnaud Carayol

LIGM – Université Paris-Est & CNRS

FICS 2010 – Brno

In a nutshell

(Higher-order) recursion schemes are rewriting systems for simply typed terms.

Simply typed lambda calculus + Recursion
(similar to Statman λY-calculus).

We use schemes to generate infinite trees. For these infinite trees, monadic-second order (MSO) logic or (equivalently the modal μ-calculus) is decidable.

Brief and incomplete historical overview

- Simple recursion schemes [Nivat72,Nivat+Courcelle78,Guessarian81]
  syntax / semantic of recursive programs
  $\Rightarrow$ prove program equivalences and program transformations

Brief and incomplete historical overview

- Simple recursion schemes [Nivat72,Nivat+Courcelle78,Guessarian81]
  syntax / semantic of recursive programs
  $\Rightarrow$ prove program equivalences and program transformations
- Higher-order recursion schemes [Damm82,Damm+Goerdt86]
  generators for languages of finite trees or words
  + only subfamily called the safe schemes

Brief and incomplete historical overview

- Simple recursion schemes [Nivat72,Nivat+Courcelle78,Guessarian81]
  syntax / semantic of recursive programs
  $\Rightarrow$ prove program equivalences and program transformations
- Higher-order recursion schemes [Damm82,Damm+Goerdt86]
  generators for languages of finite trees or words
  + only subfamily called the safe schemes
- Higher-order recursion schemes as infinite tree generators
  starting point [Knapik+Niwinski+Urzyczyn01] with only decidability for safe schemes, full decidability [Ong06] ...

Frontiers of decidability

Quest for more an more expressive countable graphs with decidable MSO-theories.

Frontiers of decidability

Quest for more an more expressive countable graphs with decidable MSO-theories.

Linear orders

- Naturals with the order relation [Buchi63]
- Countable ordinals [Buchi73,Shelah75]
- Morphic ω-words [Carton+Thomas02]
- Necessary and sufficient conditions for decidability on ω-words [Rabinovich05]

Frontiers of decidability

Quest for more an more expressive countable graphs with decidable MSO-theories.

Linear orders

- Naturals with the order relation [Buchi63]
- Countable ordinals [Buchi73,Shelah75]
- Morphic ω-words [Carton+Thomas02]
- Necessary and sufficient conditions for decidability on ω-words [Rabinovich05]

For uncountable orders, the MSO-theory of the reals is undecidable [Shelah75].

Frontiers of decidability

Quest for more an more expressive countable graphs with decidable MSO-theories.

Linear orders

- Naturals with the order relation [Buchi63]
- Countable ordinals [Buchi73,Shelah75]
- Morphic ω-words [Carton+Thomas02]
- Necessary and sufficient conditions for decidability on ω-words [Rabinovich05]

Around the full binary tree

Full binary tree [Rabin69]

- Configurations graphs of pushdown automata [Muller+Schupp85]
- VR and HR-graphs [Courcelle89,90]
- Prefix-recognizable graphs [Caucal96]

Frontiers of decidability

Graph transformations preserving decidability of MSO

- MSO-interpretations
- Unfolding [Courcelle+Walukiewicz98]
- Iteration of structures [Muchnick84,Walukiewicz96]

Frontiers of decidability

Graph transformations preserving decidability of MSO

- MSO-interpretations
- Unfolding [Courcelle+Walukiewicz98]
- Iteration of structures [Muchnick84,Walukiewicz96]

Building graphs using these transformations starting from finite graphs [Caucal98]

Frontiers of decidability

Graph transformations preserving decidability of MSO

- MSO-interpretations
- Unfolding [Courcelle+Walukiewicz98]
- Iteration of structures [Muchnick84,Walukiewicz96]

Building graphs using these transformations starting from finite graphs [Caucal98]

All these* graphs can be defined in MSO (MSO-interpreted) in trees defined by recursion schemes.

($*$) The only exceptions are the ordinals between ϵ_0 and ω_0.

Outline

- Recursions schemes and further motivation
- Automata models for recursion schemes

Typed terms

Simple types $\quad \mathrm{T} ::= \mathbf{o} \mid \mathrm{T} \to \mathrm{T}$

$\mathbf{o}$ is the base / ground type

Typed terms

Simple types $\quad \mathrm{T} ::= \mathbf{o} \mid \mathrm{T} \to \mathrm{T}$

$\mathbf{o}$ is the base / ground type

Every type τ is written uniquely as:
$\tau_1 \to (\ldots \to (\tau_n \to \mathbf{o}) \ldots)$

Typed terms

Simple types $\quad \mathrm{T} ::= \mathbf{o} \mid \mathrm{T} \to \mathrm{T}$

$\mathbf{o}$ is the base / ground type

Every type τ is written uniquely as:

$\tau_1 \to \ldots \to \tau_n \to \mathbf{o}$ $\quad$ ($\to$ is associative on the left)

Typed terms

Simple types $\quad \mathrm{T} ::= \mathbf{o} \mid \mathrm{T} \to \mathrm{T}$

$\mathbf{o}$ is the base / ground type

Every type τ is written uniquely as:

$\tau_1 \to \ldots \to \tau_n \to \mathbf{o}$ ($\to$ is associative on the left)

Order of a type : measure of the "nesting" of the type

$\mathrm{Ord}(\mathbf{o}) = 0$

$\mathrm{Ord}(\tau_1 \to \ldots \to \tau_n \to \mathbf{o}) = \left(\max_{i \in [1,n]} \mathrm{Ord}(\tau_i)\right) + 1$

Typed terms

Simple types $\quad \mathrm{T} ::= \mathbf{o} \mid \mathrm{T} \to \mathrm{T}$

$\mathbf{o}$ is the base / ground type

Every type τ is written uniquely as:

$\tau_1 \to \ldots \to \tau_n \to \mathbf{o}$ $\quad$ ($\to$ is associative on the left)

Order of a type : measure of the "nesting" of the type

$\mathrm{Ord}(\mathbf{o}) = 0$

$\mathrm{Ord}(\tau_1 \to \ldots \to \tau_n \to \mathbf{o}) = \left(\max_{i \in [1,n]} \mathrm{Ord}(\tau_i)\right) + 1$

Examples $\mathbf{o} \to \mathbf{o}$ and $\mathbf{o} \to \mathbf{o} \to \mathbf{o}$ are of order 1 and $(\mathbf{o} \to \mathbf{o}) \to \mathbf{o} \to \mathbf{o}$ is of order 2.

Applicative terms

Typed alphabet N Every $f \in N$ has a type τ (denoted $f : \tau$).

Applicative terms

Typed alphabet N Every $f \in N$ has a type τ (denoted $f : \tau$).

The set $\mathrm{Terms}(\tau)$ of applicative terms of type τ is defined by:

$$\frac{f \in N \wedge f : \tau}{f \in \mathrm{Terms}(\tau)}\ \mathrm{Const} \qquad \frac{f \in \mathrm{Terms}(\tau \to \nu) \wedge g \in \mathrm{Terms}(\tau)}{fg \in \mathrm{Terms}(\nu)}\ \mathrm{Apply}$$

Applicative terms

Typed alphabet N Every $f \in N$ has a type τ (denoted $f : \tau$).

The set $\mathrm{Terms}(\tau)$ of applicative terms of type τ is defined by:

$$\frac{f \in N \wedge f : \tau}{f \in \mathrm{Terms}(\tau)}\ \mathrm{Const} \qquad \frac{f \in \mathrm{Terms}(\tau \to \nu) \wedge g \in \mathrm{Terms}(\tau)}{fg \in \mathrm{Terms}(\nu)}\ \mathrm{Apply}$$

Notations

- Application is right-associative: $fgt = (fg)t$.
- Terms of type $\mathbf{o}$ are called ground terms are can uniquely be written as: $F\ t_1 \ldots t_n$ with $F \in N$ of arity n

Applicative terms

Typed alphabet N Every $f \in N$ has a type τ (denoted $f : \tau$).

The set $\mathrm{Terms}(\tau)$ of applicative terms of type τ is defined by:

$$\frac{f \in N \wedge f : \tau}{f \in \mathrm{Terms}(\tau)}\ \mathrm{Const} \qquad \frac{f \in \mathrm{Terms}(\tau \to \nu) \wedge g \in \mathrm{Terms}(\tau)}{fg \in \mathrm{Terms}(\nu)}\ \mathrm{Apply}$$

Notations

- Application is right-associative: $fgt = (fg)t$.
- Terms of type $\mathbf{o}$ are called ground terms are can uniquely be written as: $F\ t_1 \ldots t_n$ with $F \in N$ of arity n

Examples $f : \mathbf{o} \to \mathbf{o} \to \mathbf{o}$, $h : (\mathbf{o} \to \mathbf{o}) \to \mathbf{o}$ and $c : \mathbf{o}$

$$\underbrace{f}_{\mathbf{o}\to(\mathbf{o}\to\mathbf{o})}\ \underbrace{c}_{\mathbf{o}} : \mathbf{o} \to \mathbf{o}$$

Applicative terms

Typed alphabet N Every $f \in N$ has a type τ (denoted $f : \tau$).

The set $\mathrm{Terms}(\tau)$ of applicative terms of type τ is defined by:

$$\frac{f \in N \wedge f : \tau}{f \in \mathrm{Terms}(\tau)}\ \mathrm{Const} \qquad \frac{f \in \mathrm{Terms}(\tau \to \nu) \wedge g \in \mathrm{Terms}(\tau)}{fg \in \mathrm{Terms}(\nu)}\ \mathrm{Apply}$$

Notations

- Application is right-associative: $fgt = (fg)t$.
- Terms of type $\mathbf{o}$ are called ground terms are can uniquely be written as: $F\ t_1 \ldots t_n$ with $F \in N$ of arity n

Examples $f : \mathbf{o} \to \mathbf{o} \to \mathbf{o}$, $h : (\mathbf{o} \to \mathbf{o}) \to \mathbf{o}$ and $c : \mathbf{o}$

$$\underbrace{f}_{\mathbf{o}\to(\mathbf{o}\to\mathbf{o})}\ \underbrace{c}_{\mathbf{o}} : \mathbf{o} \to \mathbf{o} \qquad fcc = (\underbrace{fc}_{\mathbf{o}\to\mathbf{o}})\ \underbrace{c}_{\mathbf{o}} : \mathbf{o}$$

Applicative terms

Typed alphabet N Every $f \in N$ has a type τ (denoted $f : \tau$).

The set $\mathrm{Terms}(\tau)$ of applicative terms of type τ is defined by:

$$\frac{f \in N \wedge f : \tau}{f \in \mathrm{Terms}(\tau)}\ \mathrm{Const} \qquad \frac{f \in \mathrm{Terms}(\tau \to \nu) \wedge g \in \mathrm{Terms}(\tau)}{fg \in \mathrm{Terms}(\nu)}\ \mathrm{Apply}$$

Notations

- Application is right-associative: $fgt = (fg)t$.
- Terms of type $\mathbf{o}$ are called ground terms are can uniquely be written as: $F\ t_1 \ldots t_n$ with $F \in N$ of arity n

Examples $f : \mathbf{o} \to \mathbf{o} \to \mathbf{o}$, $h : (\mathbf{o} \to \mathbf{o}) \to \mathbf{o}$ and $c : \mathbf{o}$

$$\underbrace{f}_{\mathbf{o}\to(\mathbf{o}\to\mathbf{o})}\ \underbrace{c}_{\mathbf{o}} : \mathbf{o} \to \mathbf{o} \qquad fcc = (\underbrace{fc}_{\mathbf{o}\to\mathbf{o}})\ \underbrace{c}_{\mathbf{o}} : \mathbf{o}$$

(Labelled) Recursion schemes

- Typed alphabet non-terminals N (usually $A, B \ldots$)
  Axiom $S \in N$ of type $\mathbf{o}$
- Typed alphabet of variables V (usually $x, y, \phi, \psi \ldots$)

(Labelled) Recursion schemes

- Typed alphabet non-terminals N (usually $A, B \ldots$)
  Axiom $S \in N$ of type $\mathbf{o}$
- Typed alphabet of variables V (usually $x, y, \phi, \psi \ldots$)
- Finite set of productions of the form:

$$F\, x_1 \ldots x_n : \mathbf{o} \xrightarrow{a} t$$

with $F \in N, x_1, \ldots, x_n \in V$ and t is a ground term over $N \cup V$.

(Labelled) Recursion schemes

- Typed alphabet non-terminals N (usually $A, B \ldots$)
  Axiom $S \in N$ of type $\mathbf{o}$
- Typed alphabet of variables V (usually $x, y, \phi, \psi \ldots$)
- Finite set of productions of the form:

$$\underbrace{F}_{\tau_1 \to \ldots \to \tau_n \to \mathbf{o}} \underbrace{x_1}_{\tau_1} \ldots \underbrace{x_n}_{\tau_n} : \mathbf{o} \xrightarrow{a} t$$

with $F \in N, x_1, \ldots, x_n \in V$ and t is a ground term over $N \cup V$.

(Labelled) Recursion schemes

- Typed alphabet non-terminals N (usually $A, B \ldots$)
  Axiom $S \in N$ of type $\mathbf{o}$
- Typed alphabet of variables V (usually $x, y, \phi, \psi \ldots$)
- Finite set of productions of the form:

$$F\, x_1 \ldots x_n : \mathbf{o} \xrightarrow{a} t$$

with $F \in N, x_1, \ldots, x_n \in V$ and t is a ground term over $N \cup V$.

The order of scheme is the maximal order of its non-terminals.

(Labelled) Recursion schemes

- Typed alphabet non-terminals N (usually $A, B \ldots$)
  Axiom $S \in N$ of type $\mathbf{o}$
- Typed alphabet of variables V (usually $x, y, \phi, \psi \ldots$)
- Finite set of productions of the form:

$$F\, x_1 \ldots x_n : \mathbf{o} \xrightarrow{a} t$$

with $F \in N, x_1, \ldots, x_n \in V$ and t is a ground term over $N \cup V$.

The order of scheme is the maximal order of its non-terminals.

Rewriting relation $\xrightarrow{a}$ for $a \in \Sigma$ over ground terms over N

$$F\, t_1 \ldots t_n \xrightarrow{a} t[x_1/t_1, \ldots, x_n/t_n] \quad \text{if } F\, x_1 \ldots x_n \xrightarrow{a} t \in P$$

Order-1 example

Non-terminals: $S, C, V : \mathbf{o}, F : \mathbf{o} \to \mathbf{o}$

$$\begin{cases} S & \xrightarrow{a} & FC \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,x & \xrightarrow{b} & x \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ C & \xrightarrow{c} & V \end{array}$$

with $x : \mathbf{o}$.

Order-1 example

Non-terminals: $S, C, V : \mathbf{o}, F : \mathbf{o} \to \mathbf{o}$

$$\begin{cases} S & \xrightarrow{a} & FC \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,x & \xrightarrow{b} & x \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ C & \xrightarrow{c} & V \end{array}$$

with $x : \mathbf{o}$.

$$S$$

Order-1 example

Non-terminals: $S, C, V : \mathbf{o}, F : \mathbf{o} \to \mathbf{o}$

$$\begin{cases} S & \xrightarrow{a} & FC \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,x & \xrightarrow{b} & x \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ C & \xrightarrow{c} & V \end{array}$$

with $x : \mathbf{o}$.

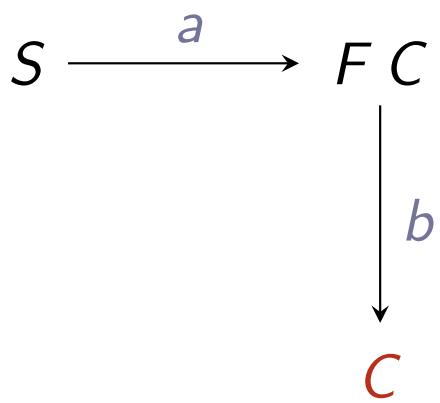
$$S \xrightarrow{a} F\,C$$

Order-1 example

Non-terminals: $S, C, V : \mathbf{o}, F : \mathbf{o} \to \mathbf{o}$

$$\begin{cases} S & \xrightarrow{a} & FC \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,x & \xrightarrow{b} & x \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ C & \xrightarrow{c} & V \end{array}$$

with $x : \mathbf{o}$.

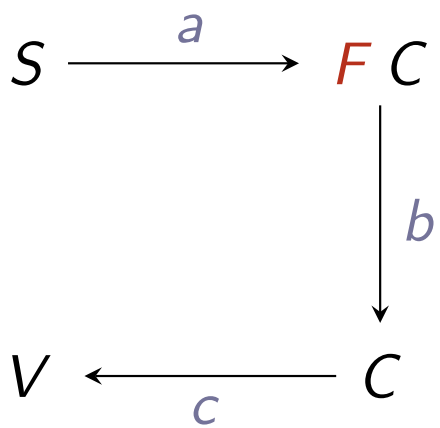


Order-1 example

Non-terminals: $S, C, V : \mathbf{o}, F : \mathbf{o} \to \mathbf{o}$

$$\begin{cases} S & \xrightarrow{a} & FC \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,x & \xrightarrow{b} & x \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ C & \xrightarrow{c} & V \end{array}$$

with $x : \mathbf{o}$.

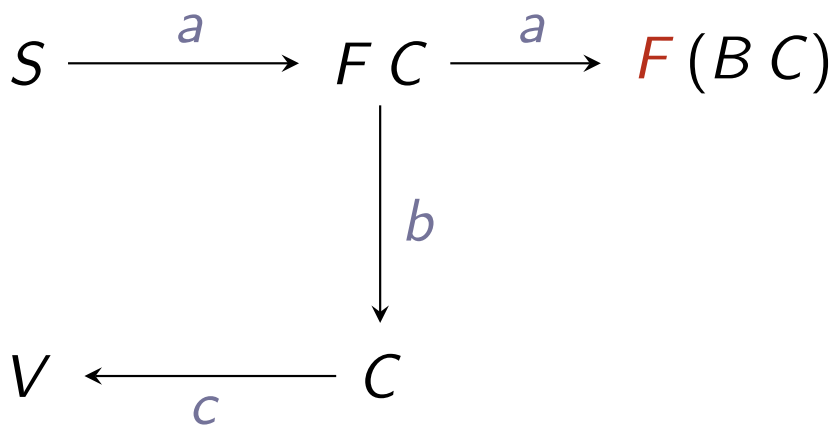


Order-1 example

Non-terminals: $S, C, V : \mathbf{o}, F : \mathbf{o} \to \mathbf{o}$

$$\begin{cases} S & \xrightarrow{a} & FC \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,x & \xrightarrow{b} & x \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ C & \xrightarrow{c} & V \end{array}$$

with $x : \mathbf{o}$.

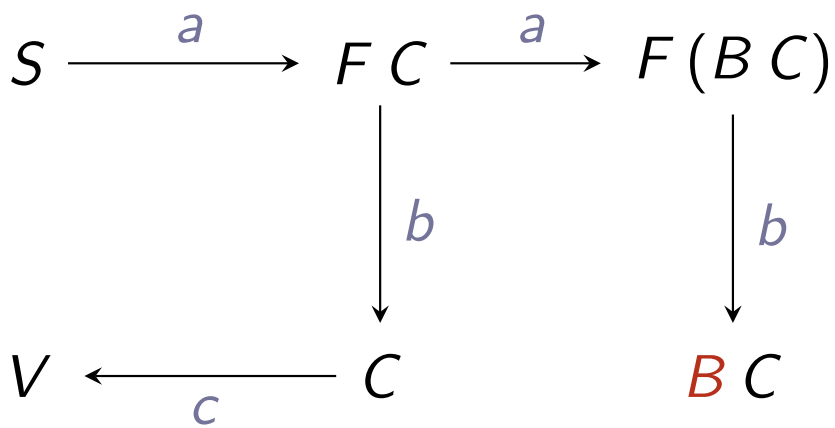


Order-1 example

Non-terminals: $S, C, V : \mathbf{o}, F : \mathbf{o} \to \mathbf{o}$

$$\begin{cases} S & \xrightarrow{a} & FC \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,x & \xrightarrow{b} & x \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ C & \xrightarrow{c} & V \end{array}$$

with $x : \mathbf{o}$.

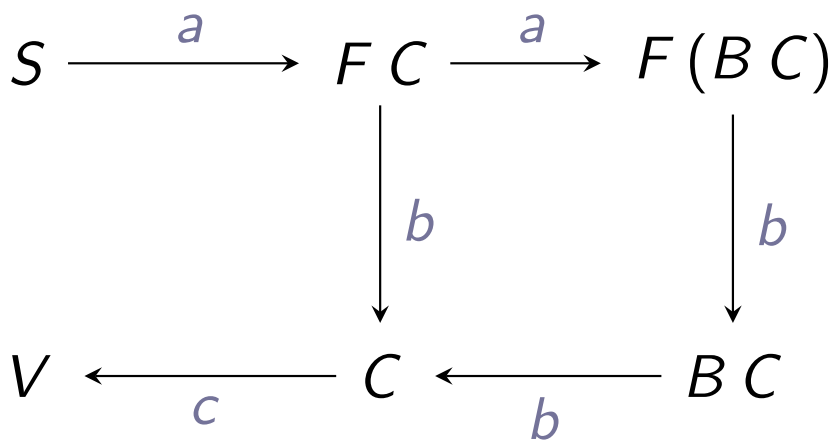


Order-1 example

Non-terminals: $S, C, V : \mathbf{o}, F : \mathbf{o} \to \mathbf{o}$

$$\begin{cases} S & \xrightarrow{a} & FC \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,x & \xrightarrow{b} & x \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ C & \xrightarrow{c} & V \end{array}$$

with $x : \mathbf{o}$.

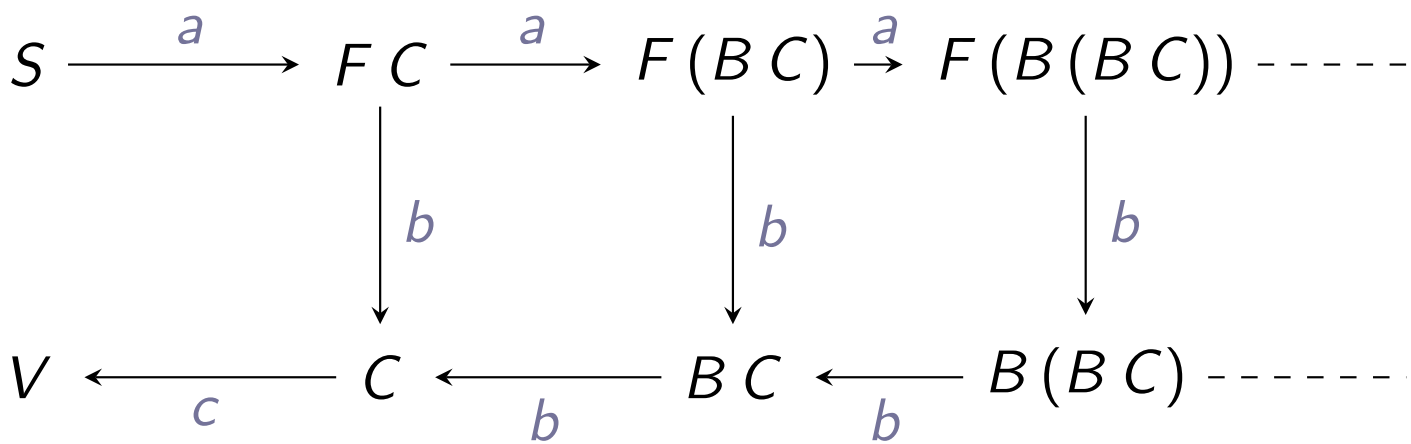


Order-1 example

Non-terminals: $S, C, V : \mathbf{o}, F : \mathbf{o} \to \mathbf{o}$

$$\begin{cases} S & \xrightarrow{a} & FC \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,x & \xrightarrow{b} & x \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ C & \xrightarrow{c} & V \end{array}$$

with $x : \mathbf{o}$.



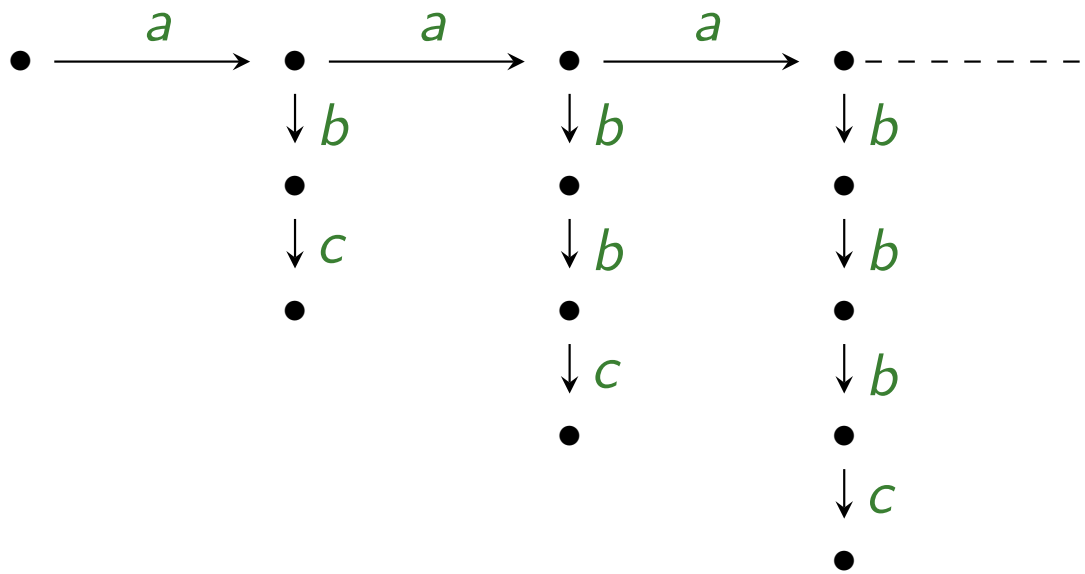
Unfold from the axiom

Order-1 example

Non-terminals: $S, C, V : \mathbf{o}, F : \mathbf{o} \to \mathbf{o}$

$$\begin{cases} S & \xrightarrow{a} & FC \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,x & \xrightarrow{b} & x \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ C & \xrightarrow{c} & V \end{array}$$

with $x : \mathbf{o}$.

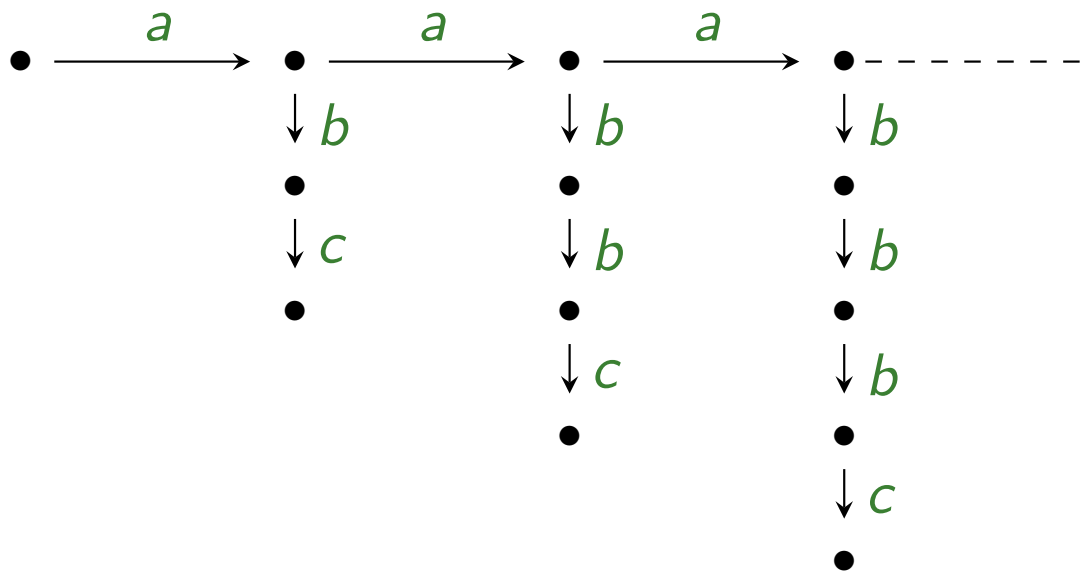


Order-1 example

Non-terminals: $S, C, V : \mathbf{o}, F : \mathbf{o} \to \mathbf{o}$

$$\begin{cases} S & \xrightarrow{a} & FC \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,x & \xrightarrow{b} & x \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ C & \xrightarrow{c} & V \end{array}$$

with $x : \mathbf{o}$.



Tree of (the prefixes of) $\{a^n b^n c \mid n \geq 1\}$.

Tree generated by a recursion scheme

Silent productions labelled by ε and "contracted" in the infinite tree

Tree generated by a recursion scheme

Silent productions labelled by ε and "contracted" in the infinite tree

Syntactic restrictions ensuring that we obtain a deterministic tree.
The productions starting with $F \in N$ are:

- either $F\, x_1 \ldots x_n \xrightarrow{a_1} t_1, \ldots, F\, x_1 \ldots x_n \xrightarrow{a_m} t_1$ with $a_i \neq a_j \neq \varepsilon$
- or $F\, x_1 \ldots x_n \xrightarrow{\varepsilon} t$.

Tree generated by a recursion scheme

Silent productions labelled by ε and "contracted" in the infinite tree

Syntactic restrictions ensuring that we obtain a deterministic tree.
The productions starting with $F \in N$ are:

- either $F\, x_1 \ldots x_n \xrightarrow{a_1} t_1, \ldots, F\, x_1 \ldots x_n \xrightarrow{a_m} t_1$ with $a_i \neq a_j \neq \varepsilon$
- or $F\, x_1 \ldots x_n \xrightarrow{\varepsilon} t$.

The infinite tree of a scheme G is obtained by:

unfolding the graph of G from S

$+$

contracting the ε-labelled edges

Tree generated by a recursion scheme

Silent productions labelled by ε and "contracted" in the infinite tree

Syntactic restrictions ensuring that we obtain a deterministic tree.
The productions starting with $F \in N$ are:

- either $F\, x_1 \ldots x_n \xrightarrow{a_1} t_1, \ldots, F\, x_1 \ldots x_n \xrightarrow{a_m} t_1$ with $a_i \neq a_j \neq \varepsilon$
- or $F\, x_1 \ldots x_n \xrightarrow{\varepsilon} t$.

The infinite tree of a scheme G is obtained by:

unfolding the graph of G from S

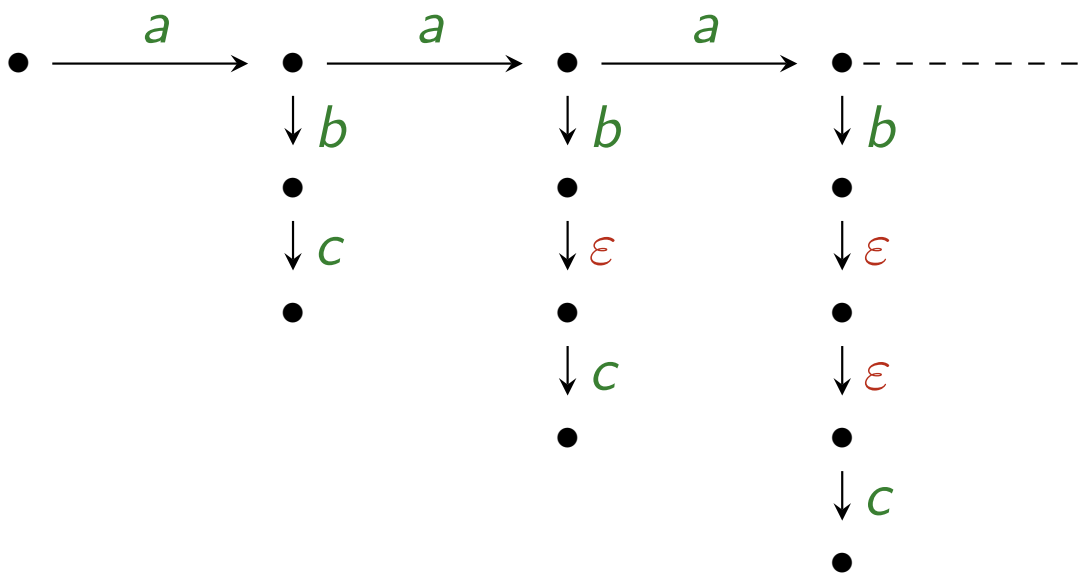
$+$

contracting the ε-labelled edges

The class of infinite trees generated in this way coincide with the terms generated in the sense of [Knapik-Niwinski-Urzyczyn01].

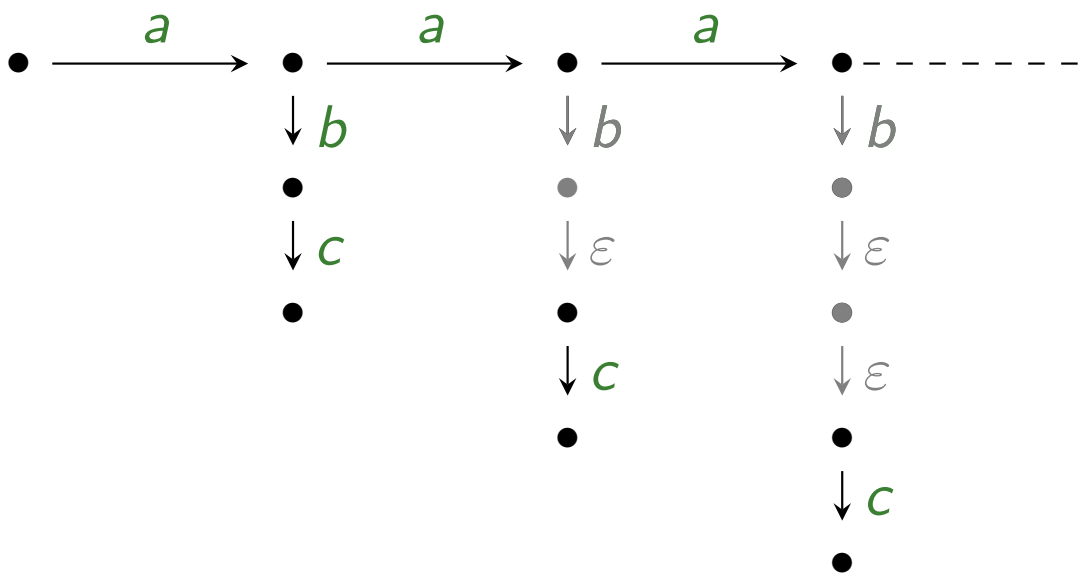
ε-contraction

$$\begin{cases} S \xrightarrow{a} FC & \qquad Bx \xrightarrow{\varepsilon} x \\ Fx \xrightarrow{a} F(Bx) & \qquad C \xrightarrow{c} V \\ Fx \xrightarrow{b} x \end{cases}$$



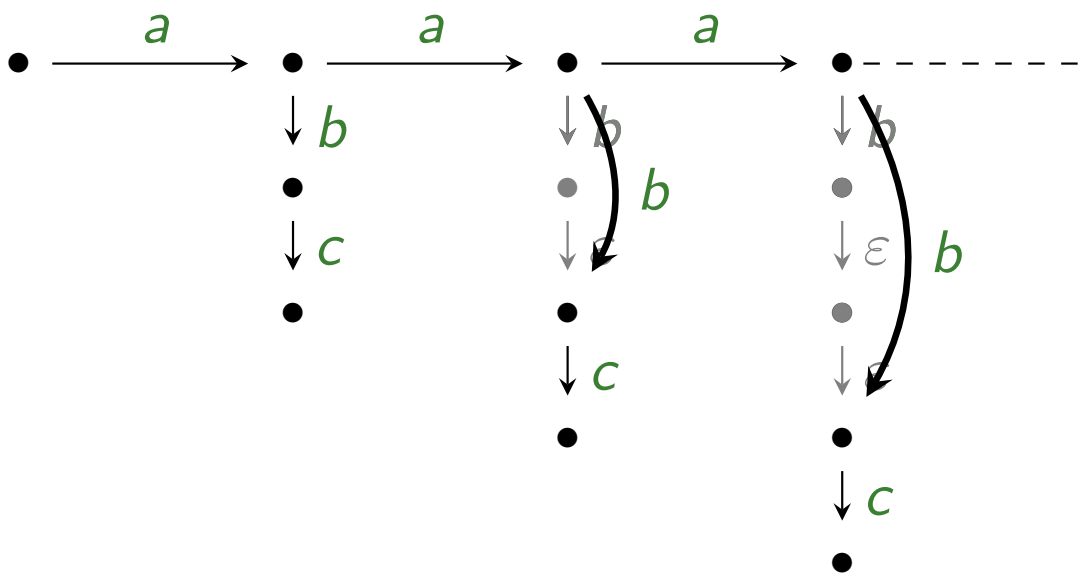
ε-contraction

$$\begin{cases} S \xrightarrow{a} FC & \qquad Bx \xrightarrow{\varepsilon} x \\ Fx \xrightarrow{a} F(Bx) & \qquad C \xrightarrow{c} V \\ Fx \xrightarrow{b} x \end{cases}$$



ε-contraction

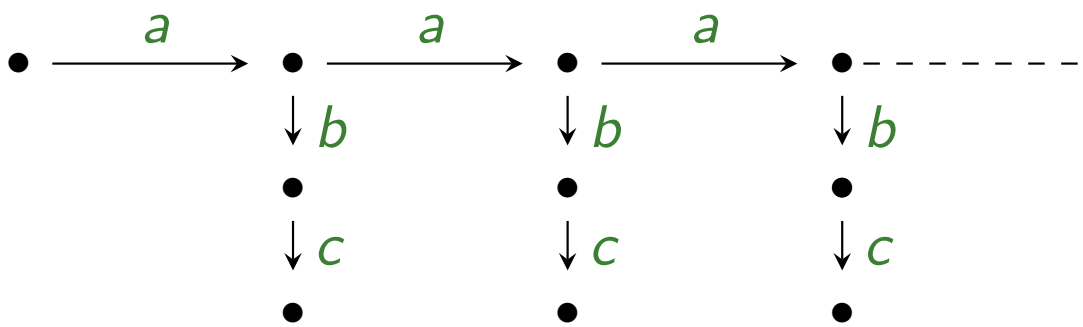
$$\begin{cases} S \xrightarrow{a} FC \\ Fx \xrightarrow{a} F(Bx) \\ Fx \xrightarrow{b} x \end{cases} \qquad \begin{array}{lll} Bx & \xrightarrow{\varepsilon} & x \\ C & \xrightarrow{c} & V \end{array}$$



add b-edge when there was a path in $b\varepsilon^*$

ε-contraction

$$\begin{cases} S \xrightarrow{a} FC & Bx \xrightarrow{\varepsilon} x \\ Fx \xrightarrow{a} F(Bx) & C \xrightarrow{c} V \\ Fx \xrightarrow{b} x \end{cases}$$



Examples of order 2

Aim Tree of $\{a^n d b^n c^n \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C \\ F\,\phi\,\psi & \xrightarrow{a} & F\,(Cp\,B\,\phi)\,(Cp\,C\,\psi) \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) \end{cases} \qquad \begin{cases} B\,x & \xrightarrow{b} & x \\ C\,x & \xrightarrow{b} & x \\ Cp\,\phi\,\psi\,x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

Examples of order 2

Aim Tree of $\{a^n d\, b^n c^n \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\, B\, C & \qquad B\, x & \xrightarrow{b} & x \\ F\, \phi\, \psi & \xrightarrow{a} & F\, (Cp\, B\, \phi)\, (Cp\, C\, \psi) & \qquad C\, x & \xrightarrow{b} & x \\ F\, \phi\, \psi & \xrightarrow{d} & \phi\, (\psi\, V) & \qquad Cp\, \phi\, \psi\, x & \xrightarrow{\varepsilon} & \phi\, (\psi\, x) \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.
$Cp : (\mathbf{o} \to \mathbf{o}) \to (\mathbf{o} \to \mathbf{o}) \to (\mathbf{o} \to \mathbf{o})$
Cp corresponds to the composition of functions
$Cp\, B\, B = B^2$, $Cp\, B\, (Cp\, B\, B) = B^3$, ...
For all $n \geq 0$ and term $t : \mathbf{o}$, $B^n t$ reduces in n b's to t.

Examples of order 2

Aim Tree of $\{a^n d\, b^n c^n \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\, B\, C \\ F\, \phi\, \psi & \xrightarrow{a} & F\, (Cp\, B\, \phi)\, (Cp\, C\, \psi) \\ F\, \phi\, \psi & \xrightarrow{d} & \phi\, (\psi\, V) \end{cases} \qquad \begin{cases} B\, x & \xrightarrow{b} & x \\ C\, x & \xrightarrow{b} & x \\ Cp\, \phi\, \psi\, x & \xrightarrow{\varepsilon} & \phi\, (\psi\, x) \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

S

Examples of order 2

Aim Tree of $\{a^n d b^n c^n \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C \\ F\,\phi\,\psi & \xrightarrow{a} & F\,(Cp\,B\,\phi)\,(Cp\,C\,\psi) \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) \end{cases} \qquad \begin{cases} B\,x & \xrightarrow{b} & x \\ C\,x & \xrightarrow{b} & x \\ Cp\,\phi\,\psi\,x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

$$S \xrightarrow{a} F\,BC$$

Examples of order 2

Aim Tree of $\{a^n d b^n c^n \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C \\ F\,\phi\,\psi & \xrightarrow{a} & F\,(Cp\,B\,\phi)\,(Cp\,C\,\psi) \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) \end{cases} \qquad \begin{cases} B\,x & \xrightarrow{b} & x \\ C\,x & \xrightarrow{b} & x \\ Cp\,\phi\,\psi\,x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

$$S \xrightarrow{a} F\,BC \xrightarrow{a} F\,B^2\,C^2$$

Examples of order 2

Aim Tree of $\{a^n d b^n c^n \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C & \qquad B\,x & \xrightarrow{b} & x \\ F\,\phi\,\psi & \xrightarrow{a} & F\,(Cp\,B\,\phi)\,(Cp\,C\,\psi) & \qquad C\,x & \xrightarrow{b} & x \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) & \qquad Cp\,\phi\,\psi\,x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

$$S \xrightarrow{a} F\,BC \xrightarrow{a} F\,B^2\,C^2 \xrightarrow{a} F\,B^3\,C^3 \text{ ------}$$

Examples of order 2

Aim Tree of $\{a^n d b^n c^n \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C & \qquad B\,x & \xrightarrow{b} & x \\ F\,\phi\,\psi & \xrightarrow{a} & F\,(Cp\,B\,\phi)\,(Cp\,C\,\psi) & \qquad C\,x & \xrightarrow{b} & x \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) & \qquad Cp\,\phi\,\psi\,x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

$$S \xrightarrow{a} F\,BC \xrightarrow{a} F\,B^2\,C^2 \xrightarrow{a} F\,B^3\,C^3 \dashrightarrow$$
$$\downarrow d$$
$$B\,(C\,V)$$

Examples of order 2

Aim Tree of $\{a^n d b^n c^n \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C & \qquad B\,x & \xrightarrow{b} & x \\ F\,\phi\,\psi & \xrightarrow{a} & F\,(Cp\,B\,\phi)\,(Cp\,C\,\psi) & \qquad C\,x & \xrightarrow{b} & x \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) & \qquad Cp\,\phi\,\psi\,x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

$$S \xrightarrow{a} F\,BC \xrightarrow{a} F\,B^2\,C^2 \xrightarrow{a} F\,B^3\,C^3 \text{ ------}$$

$$F\,BC \downarrow d \quad B\,(C\,V) \downarrow b \bullet \downarrow c \bullet$$

Examples of order 2

Aim Tree of $\{a^n d b^n c^n \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C & \qquad B\,x & \xrightarrow{b} & x \\ F\,\phi\,\psi & \xrightarrow{a} & F\,(Cp\,B\,\phi)\,(Cp\,C\,\psi) & \qquad C\,x & \xrightarrow{b} & x \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) & \qquad Cp\,\phi\,\psi\,x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

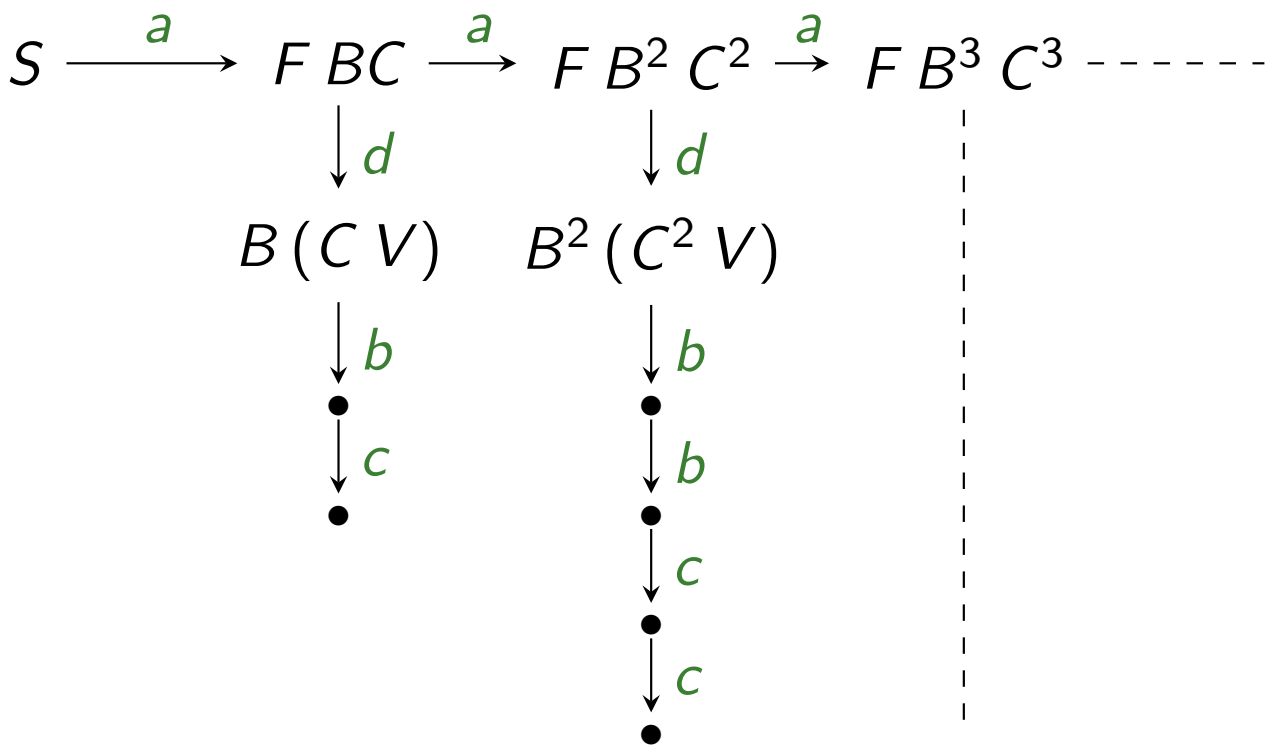
$$\begin{array}{ccccccc} S & \xrightarrow{a} & F\,BC & \xrightarrow{a} & F\,B^2\,C^2 & \xrightarrow{a} & F\,B^3\,C^3 \text{ - - - - - -} \\ & & \downarrow d & & \downarrow d & & \\ & & B\,(C\,V) & & B^2\,(C^2\,V) & & \\ & & \downarrow b & & & & \\ & & \bullet & & & & \\ & & \downarrow c & & & & \\ & & \bullet & & & & \end{array}$$

Examples of order 2

Aim Tree of $\{a^n d b^n c^n \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C & \qquad B\,x & \xrightarrow{b} & x \\ F\,\phi\,\psi & \xrightarrow{a} & F\,(Cp\,B\,\phi)\,(Cp\,C\,\psi) & \qquad C\,x & \xrightarrow{b} & x \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) & \qquad Cp\,\phi\,\psi\,x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.



Examples of order-2

Aim Tree of the language $\{a^n d\, b^{2^n} \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,(Cp\,B\,B) \\ F\,\phi & \xrightarrow{a} & F(Cp\,\phi\,\phi) \\ F\,\phi & \xrightarrow{d} & \phi\,V \end{cases} \qquad \begin{array}{lcl} Bx & \xrightarrow{b} & x \\ Cp\,\phi\,\psi x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{array}$$

with $x : \mathbf{o}$, $\phi : \mathbf{o} \to \mathbf{o}$.

S

Examples of order-2

Aim Tree of the language $\{a^n d\, b^{2^n} \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,(Cp\,B\,B) \\ F\,\phi & \xrightarrow{a} & F(Cp\,\phi\,\phi) \\ F\,\phi & \xrightarrow{d} & \phi\,V \end{cases} \qquad \begin{array}{lcl} Bx & \xrightarrow{b} & x \\ Cp\,\phi\,\psi x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{array}$$

with $x : \mathbf{o}$, $\phi : \mathbf{o} \to \mathbf{o}$.

$$S \xrightarrow{a} F\,B^2$$

Examples of order-2

Aim Tree of the language $\{a^n d\, b^{2^n} \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,(Cp\,B\,B) \\ F\,\phi & \xrightarrow{a} & F(Cp\,\phi\,\phi) \\ F\,\phi & \xrightarrow{d} & \phi\,V \end{cases} \qquad \begin{array}{lcl} Bx & \xrightarrow{b} & x \\ Cp\,\phi\,\psi x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{array}$$

with $x : \mathbf{o}$, $\phi : \mathbf{o} \to \mathbf{o}$.

$$S \xrightarrow{a} F\,B^2 \xrightarrow{a} F\,(B^2 \circ B^2)$$

Examples of order-2

Aim Tree of the language $\{a^n d\, b^{2^n} \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,(Cp\,B\,B) \\ F\,\phi & \xrightarrow{a} & F(Cp\,\phi\,\phi) \\ F\,\phi & \xrightarrow{d} & \phi\,V \end{cases} \qquad \begin{array}{lcl} Bx & \xrightarrow{b} & x \\ Cp\,\phi\,\psi x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{array}$$

with $x : \mathbf{o}$, $\phi : \mathbf{o} \to \mathbf{o}$.

$$S \xrightarrow{a} F\,B^2 \xrightarrow{a} F\,B^4$$

Examples of order-2

Aim Tree of the language $\{a^n d\, b^{2^n} \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,(Cp\,B\,B) \\ F\,\phi & \xrightarrow{a} & F(Cp\,\phi\,\phi) \\ F\,\phi & \xrightarrow{d} & \phi\,V \end{cases} \qquad \begin{array}{lcl} Bx & \xrightarrow{b} & x \\ Cp\,\phi\,\psi x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{array}$$

with $x : \mathbf{o}$, $\phi : \mathbf{o} \to \mathbf{o}$.

$$S \xrightarrow{a} F\,B^2 \xrightarrow{a} F\,B^4 \xrightarrow{a} F\,B^{2^3} \dashrightarrow$$

Examples of order-2

Aim Tree of the language $\{a^n d\, b^{2^n} \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,(Cp\,B\,B) \\ F\,\phi & \xrightarrow{a} & F(Cp\,\phi\,\phi) \\ F\,\phi & \xrightarrow{d} & \phi\,V \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ Cp\,\phi\,\psi\,x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{array}$$

with $x : \mathbf{o}$, $\phi : \mathbf{o} \to \mathbf{o}$.

$$S \xrightarrow{a} F\,B^2 \xrightarrow{a} F\,B^4 \xrightarrow{a} F\,B^{2^3} \cdots\cdots\cdots$$

$$F\,B^2 \downarrow d\ B^2\,V$$

Examples of order-2

Aim Tree of the language $\{a^n d\, b^{2^n} \mid n \geq 0\}$

$$\begin{cases} S \xrightarrow{a} F\,(Cp\,B\,B) & B x \xrightarrow{b} x \\ F\,\phi \xrightarrow{a} F(Cp\,\phi\,\phi) & Cp\,\phi\,\psi x \xrightarrow{\varepsilon} \phi\,(\psi\,x) \\ F\,\phi \xrightarrow{d} \phi\,V \end{cases}$$

with $x : \mathbf{o}$, $\phi : \mathbf{o} \to \mathbf{o}$.

$$S \xrightarrow{a} F\,B^2 \xrightarrow{a} F\,B^4 \xrightarrow{a} F\,B^{2^3} \cdots\cdots\cdots$$

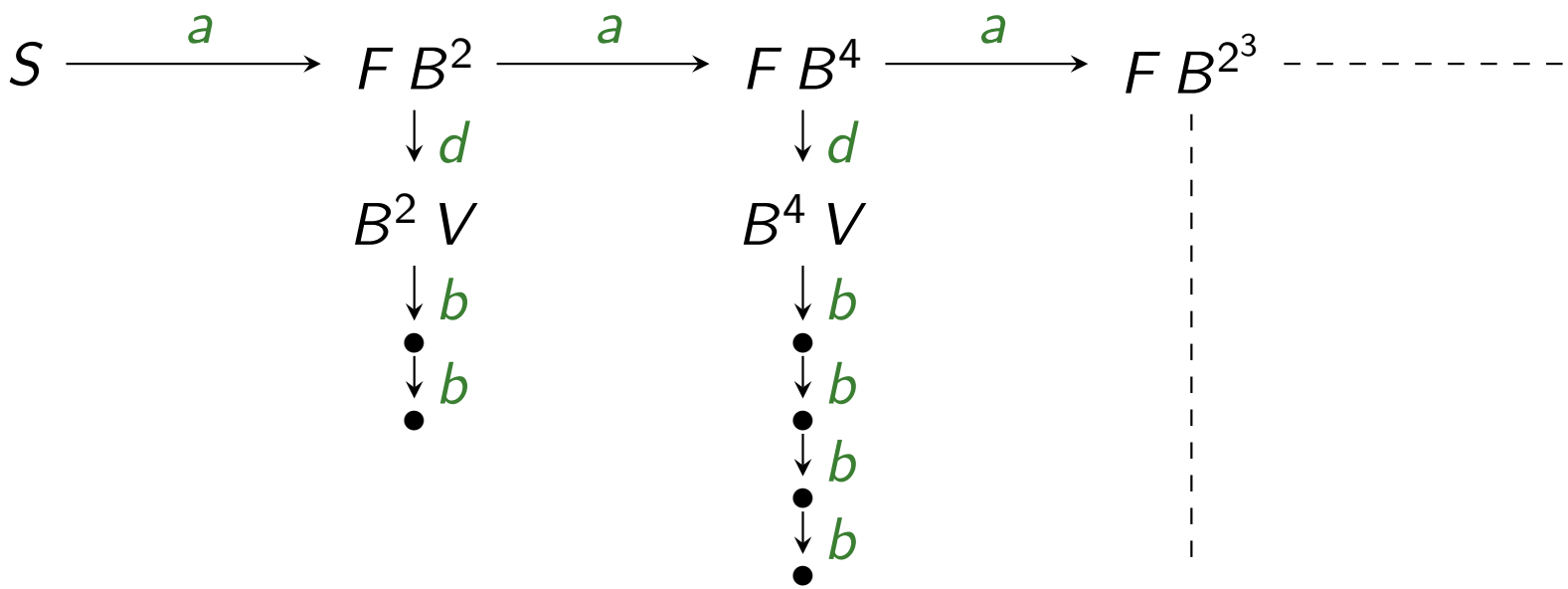
$$F\,B^2 \downarrow d\; B^2\,V \downarrow b \bullet \downarrow b \bullet$$

Examples of order-2

Aim Tree of the language $\{a^n d\, b^{2^n} \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,(Cp\,B\,B) \\ F\,\phi & \xrightarrow{a} & F(Cp\,\phi\,\phi) \\ F\,\phi & \xrightarrow{d} & \phi\,V \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ Cp\,\phi\,\psi\,x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{array}$$

with $x : \mathbf{o}$, $\phi : \mathbf{o} \to \mathbf{o}$.

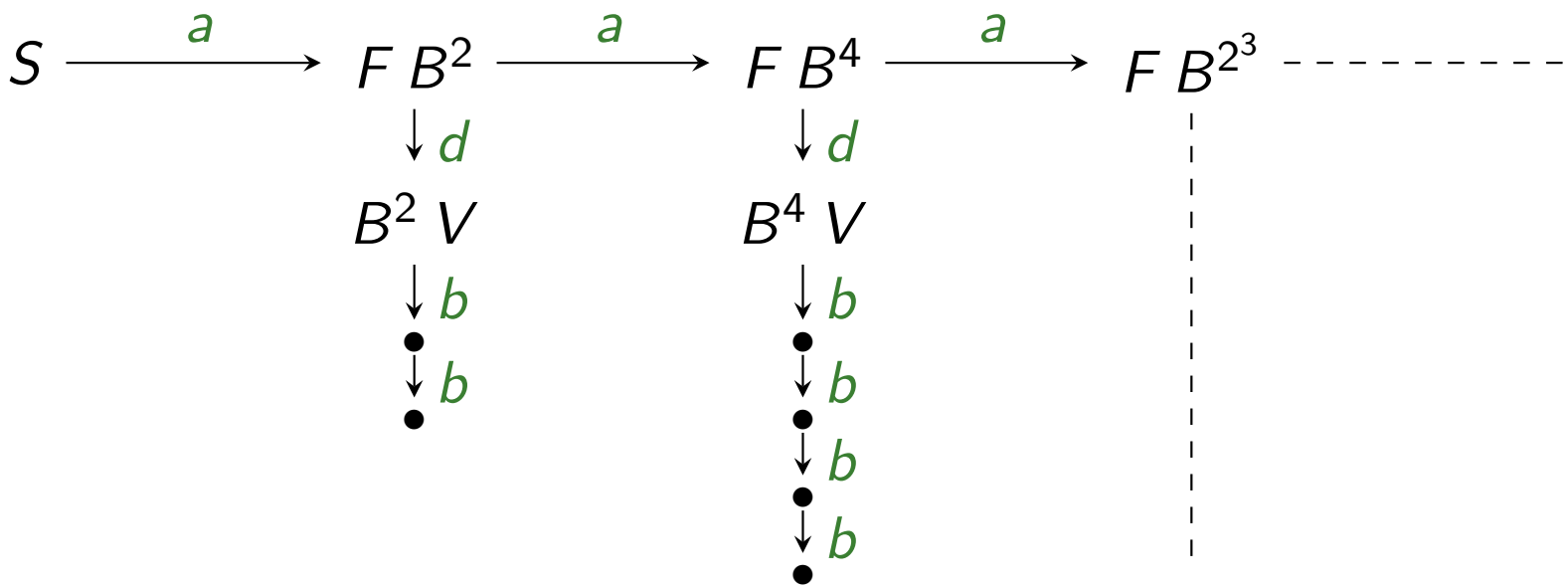


Examples of order-2

Aim Tree of the language $\{a^n d\, b^{2^n} \mid n \geq 0\}$

$$\begin{cases} S & \xrightarrow{a} & F\,(Cp\,B\,B) \\ F\,\phi & \xrightarrow{a} & F(Cp\,\phi\,\phi) \\ F\,\phi & \xrightarrow{d} & \phi\,V \end{cases} \qquad \begin{array}{lcl} B\,x & \xrightarrow{b} & x \\ Cp\,\phi\,\psi\,x & \xrightarrow{\varepsilon} & \phi\,(\psi\,x) \end{array}$$

with $x : \mathbf{o}$, $\phi : \mathbf{o} \to \mathbf{o}$.



Similarly at order 3, we have the tree of $\{a^n d\, b^{2^{(2^n)}}\}$.

Algorithmic properties

Algorithmic properties

Theorem ([Ong06])

For all $n \geq 1$, the model-checking problem for modal mu-calculus is $n - \mathrm{EXPTIME}$ complete for trees generated by order-n recursion schemes.
Hence the trees defined by higher-order recursion schemes have a decidable MSO-theory.

Three proofs

Starting problem

Given a scheme G and an APT $\mathcal{A}$, does $\mathcal{A}$ accept the tree generated by G ?

Three proofs

Starting problem

Given a scheme G and an APT $\mathcal{A}$, does $\mathcal{A}$ accept the tree generated by G ?

- [Ong06] direct reduction to solving a finite parity game
  $\Rightarrow$ heavily based on notions from game semantic

Three proofs

Starting problem

Given a scheme G and an APT $\mathcal{A}$, does $\mathcal{A}$ accept the tree generated by G ?

- [Ong06] direct reduction to solving a finite parity game
  $\Rightarrow$ heavily based on notions from game semantic
- [Hague+Murawski+Ong+Serre08]
  generalising [Knapik+Niwinski+Urzyczyn+Walukiewicz05] and [Aeligh+deMiranda+Ong05] at order 2.
  $\Rightarrow$ characterisation of schemes via collapsible automata + induction on the order

Three proofs

Starting problem

Given a scheme G and an APT $\mathcal{A}$, does $\mathcal{A}$ accept the tree generated by G ?

- [Ong06] direct reduction to solving a finite parity game
  $\Rightarrow$ heavily based on notions from game semantic
- [Hague+Murawski+Ong+Serre08]
  generalising [Knapik+Niwinski+Urzyczyn+Walukiewicz05] and [Aeligh+deMiranda+Ong05] at order 2.
  $\Rightarrow$ characterisation of schemes via collapsible automata + induction on the order
- [Kobayashi+Ong09] reduction to a typing problem for G
  $\Rightarrow$ solving the typing problem amount to solving a finite parity game

Software verification by model-checking (borrowed from Kobayashi)

```
Let F x =
      if |x|==0 then close(x)
           else read(x); F(x)
Let S = open("txt"); F("txt")
```

⇓ Abstract the data

```
Let F =
      if * then close
           else read; F
Let S = open; F
```

Software verification by model-checking

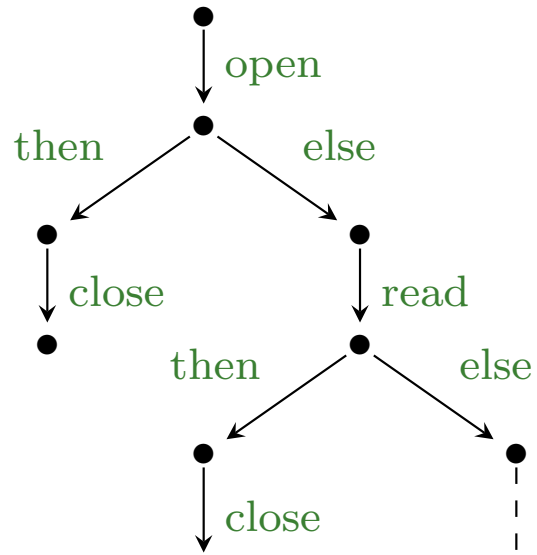
$$\begin{cases} O\,k & \xrightarrow{\text{open}} & k \\ R\,k & \xrightarrow{\text{read}} & k \\ C\,k & \xrightarrow{\text{close}} & k \\ F\,k & \xrightarrow{\text{then}} & C\,k \\ F\,k & \xrightarrow{\text{else}} & R\,(F\,k) \\ S & \xrightarrow{\varepsilon} & O\,(F\,V) \end{cases}$$

$\Uparrow$ CPS transformation

```
Let F =
      if * then close
           else read; F
Let S = open; F
```

Software verification by model-checking

$$\begin{cases} O\,k & \xrightarrow{\text{open}} & k \\ R\,k & \xrightarrow{\text{read}} & k \\ C\,k & \xrightarrow{\text{close}} & k \\ F\,k & \xrightarrow{\text{then}} & C\,k \\ F\,k & \xrightarrow{\text{else}} & R\,(F\,k) \\ S & \xrightarrow{\varepsilon} & O\,(F\,V) \end{cases}$$



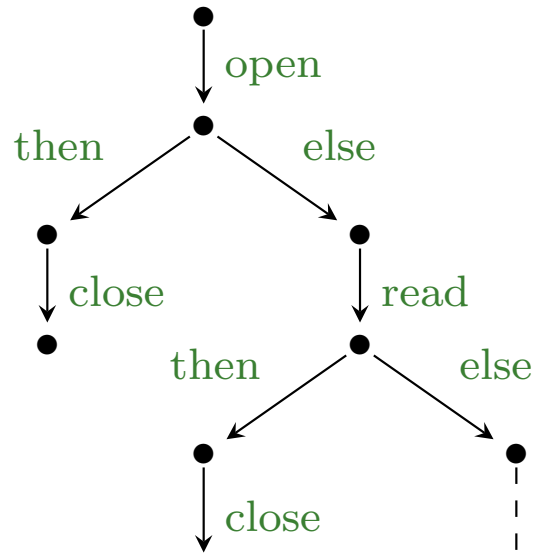
⇑ CPS transformation

```
Let F =
      if * then close
           else read; F
Let S = open; F
```

Software verification by model-checking

We can check on the tree that every read follows an open.

$$\begin{cases} O\,k & \xrightarrow{\text{open}} & k \\ R\,k & \xrightarrow{\text{read}} & k \\ C\,k & \xrightarrow{\text{close}} & k \\ F\,k & \xrightarrow{\text{then}} & C\,k \\ F\,k & \xrightarrow{\text{else}} & R\,(F\,k) \\ S & \xrightarrow{\varepsilon} & O\,(F\,V) \end{cases}$$



$\Uparrow$ CPS transformation

```
Let F =
      if * then close
           else read; F
Let S = open; F
```

Software verification by model-checking (borrowed from Kobayashi)

Complexity (n – EXPTIME complete at order n) is the main issue.

Software verification by model-checking (borrowed from Kobayashi)

Complexity (n – EXPTIME complete at order n) is the main issue.

Kobayashi [Kobayashi09] used the typing approach to build a model-checker for so-called trivial properties.
TReCS `http://www.kb.ecei.tohoku.ac.jp/~koba/trecs/`

Software verification by model-checking (borrowed from Kobayashi)

Complexity ($n - \mathrm{EXPTIME}$ complete at order n) is the main issue.

Kobayashi [Kobayashi09] used the typing approach to build a model-checker for so-called trivial properties.
TReCS `http://www.kb.ecei.tohoku.ac.jp/~koba/trecs/`

- still $(n-1) - \mathrm{EXPTIME}$ complete
- yet: 4 states property on an order-4 scheme with 23 productions in 5ms.

Outline

- Recursions schemes and further motivation
- Automata models for recursion schemes

Model for order 1

Theorem ([folklore])

The infinite tree of a word language L is definable by an order 1 recursion scheme iff L is accepted by a deterministic pushdown automaton.

Model for order 1

Theorem ([folklore])

The infinite tree of a word language L is definable by an order 1 recursion scheme iff L is accepted by a deterministic pushdown automaton.

Theorem ([Sénizergues00])

The equivalence problem for deterministic pushdown automata is decidable.
Isomorphism is decidable for order 1 schemes.

Model for order 2

Non-determinism is not the way to describe the trees of order-2 schemes.

$\Rightarrow$ There exists a language accepted by non-deterministic pushdown automata (i.e. a context-free language) whose tree has an undecidable MSO-theory.

Model for order 2

Independently introduced by
[Knapik+Niwinski+Urzyczyn+Walukiewicz05] and
[Aeligh+deMiranda+Ong05] for order-2.

Theorem

The infinite tree of a word language L is definable by an order 2 recursion scheme iff L is accepted by a deterministic collapsible automaton of order 2.

Model for order 2

Independently introduced by
[Knapik+Niwinski+Urzyczyn+Walukiewicz05] and
[Aeligh+deMiranda+Ong05] for order-2.

Theorem

The infinite tree of a word language L is definable by an order 2 recursion scheme iff L is accepted by a deterministic collapsible automaton of order 2.

Generalised to all orders by [Hague+Murawski+Ong+Serre08] using notions from game semantic.

Model for order 2

Independently introduced by
[Knapik+Niwinski+Urzyczyn+Walukiewicz05] and
[Aeligh+deMiranda+Ong05] for order-2.

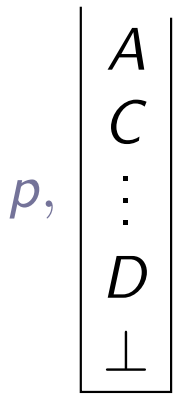
Theorem
The infinite tree of a word language L is definable by an order 2 recursion scheme iff L is accepted by a deterministic collapsible automaton of order 2.

Generalised to all orders by [Hague+Murawski+Ong+Serre08] using notions from game semantic.

We will focus on level 2.

Pushdown automata

Finite control + stack

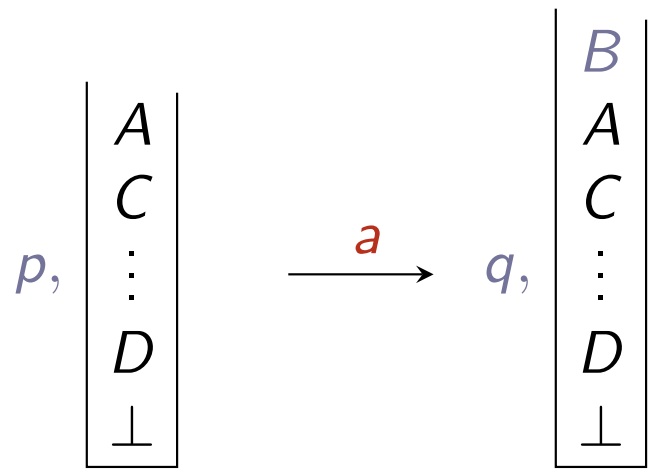


Transitions of the form: $p, A \xrightarrow{a} q, \rho$ with $\rho \in \{\mathrm{push}_B, \mathrm{pop}\}$.

Pushdown automata

Finite control + stack

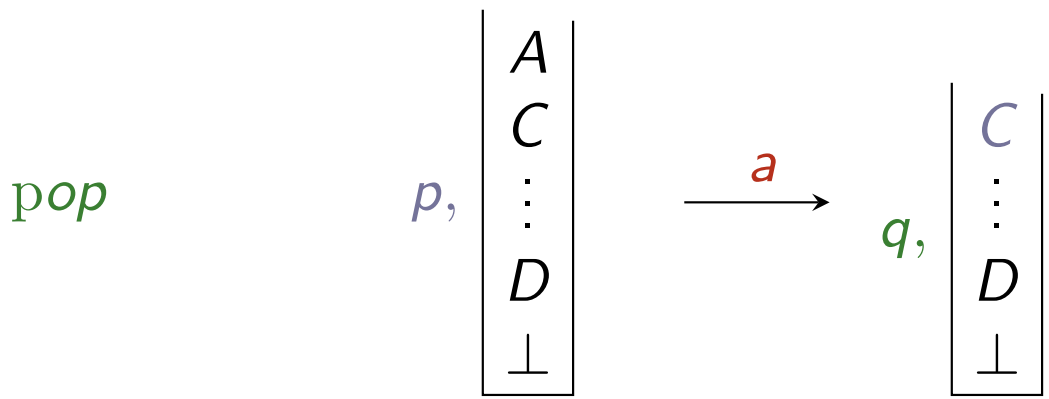
push_B



Transitions of the form: $p, A \xrightarrow{a} q, \rho$ with $\rho \in \{\mathrm{push}_B, \mathrm{pop}\}$.

Pushdown automata

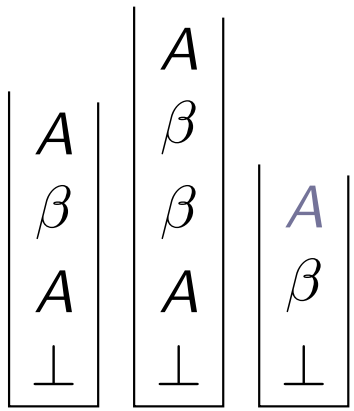
Finite control + stack



Transitions of the form: $p, A \xrightarrow{a} q, \rho$ with $\rho \in \{push_B, pop\}$.

Order-2 pushdown [Maslov76]

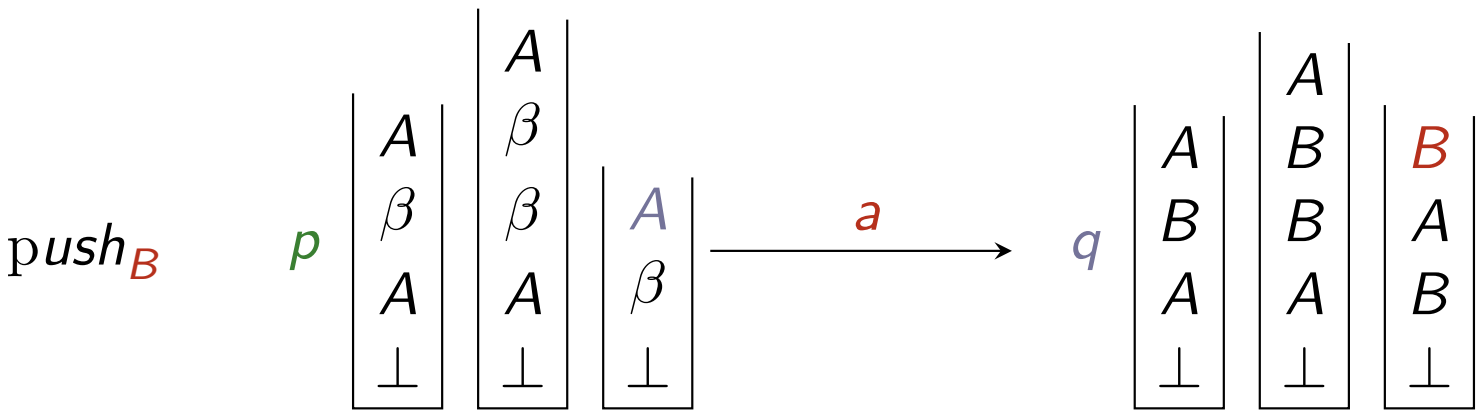
Finite control + stack of (ordinary) stacks



Transitions of the form: $p, A \xrightarrow{a} q, \rho$ with $\rho \in \{push_A, pop, pop_2, push_2\}$.

Order-2 pushdown [Maslov76]

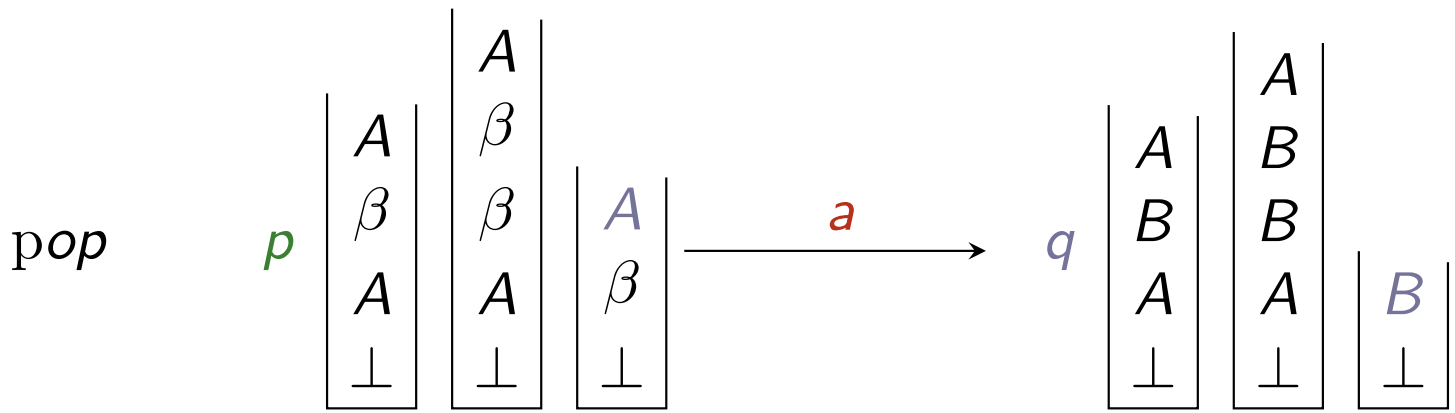
Finite control + stack of (ordinary) stacks



Transitions of the form: $p, A \xrightarrow{a} q, \rho$ with $\rho \in \{push_A, pop, pop_2, push_2\}$.

Order-2 pushdown [Maslov76]

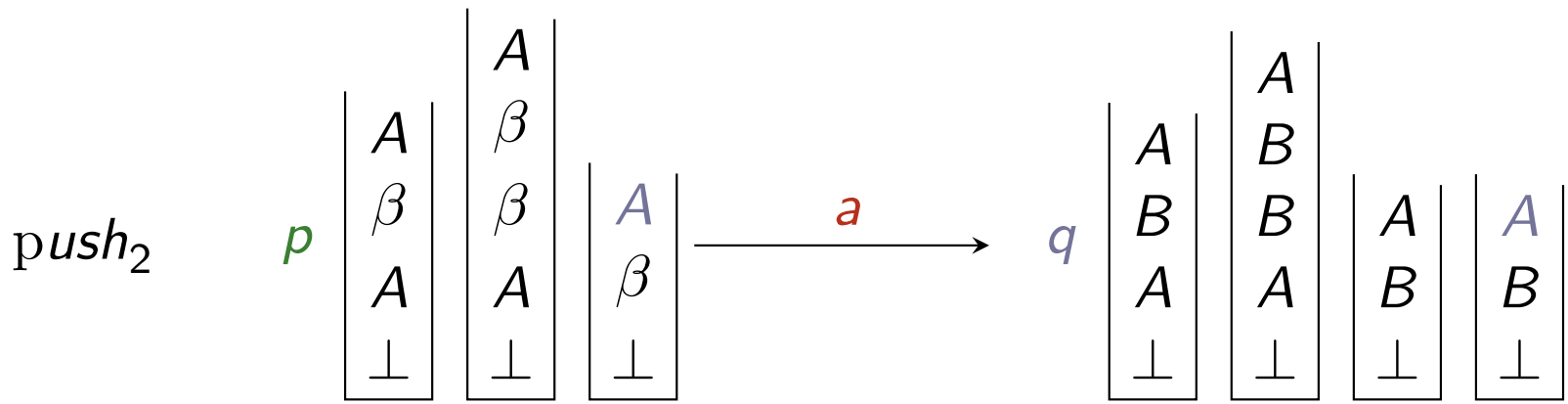
Finite control + stack of (ordinary) stacks



Transitions of the form: $p, A \xrightarrow{a} q, \rho$ with $\rho \in \{\mathrm{push}_A, \mathrm{pop}, \mathrm{pop}_2, \mathrm{push}_2\}$.

Order-2 pushdown [Maslov76]

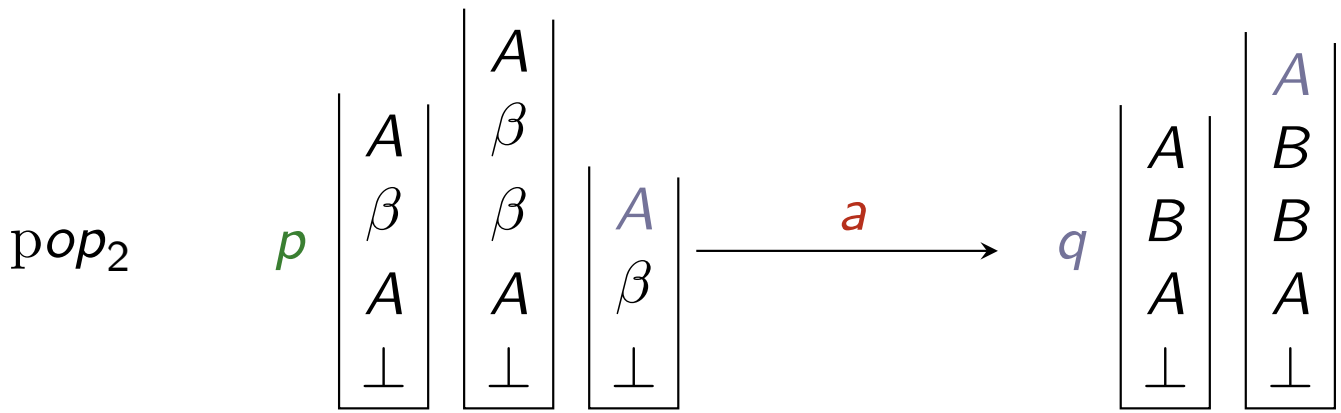
Finite control + stack of (ordinary) stacks



Transitions of the form: $p, A \xrightarrow{a} q, \rho$ with $\rho \in \{push_A, pop, pop_2, push_2\}$.

Order-2 pushdown [Maslov76]

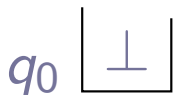
Finite control + stack of (ordinary) stacks



Transitions of the form: $p, A \xrightarrow{a} q, \rho$ with $\rho \in \{push_A, pop, pop_2, push_2\}$.

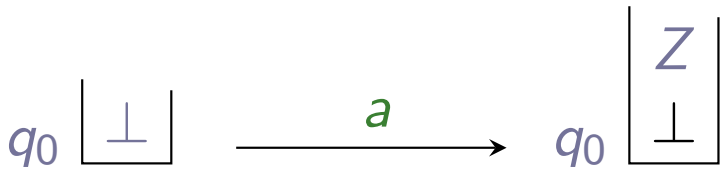
Example $\{a^n b^n c^n | \mid n \geq 0\}$

Accepting *aabbcc*



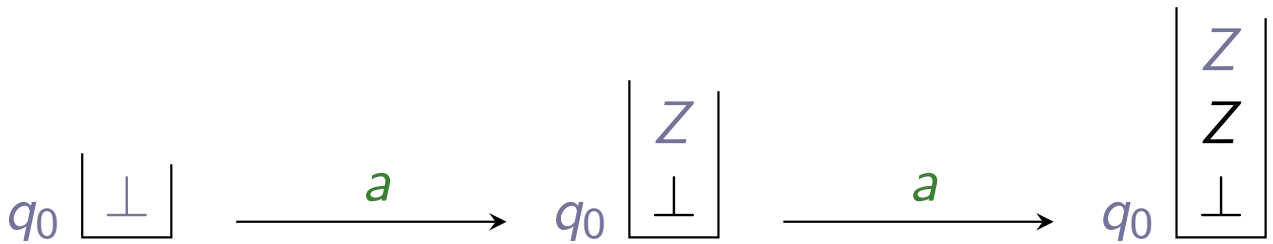
Example $\{a^n b^n c^n | \mid n \geq 0\}$

Accepting *aabbcc*



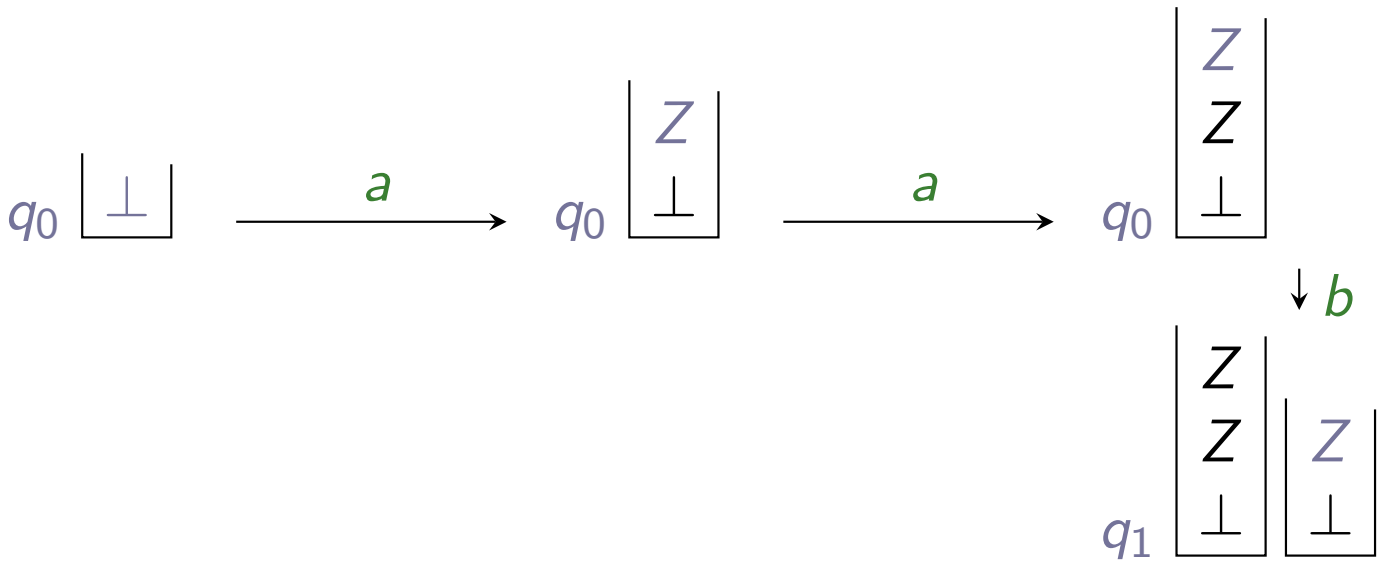
Example $\{a^n b^n c^n | \mid n \geq 0\}$

Accepting *aabbcc*



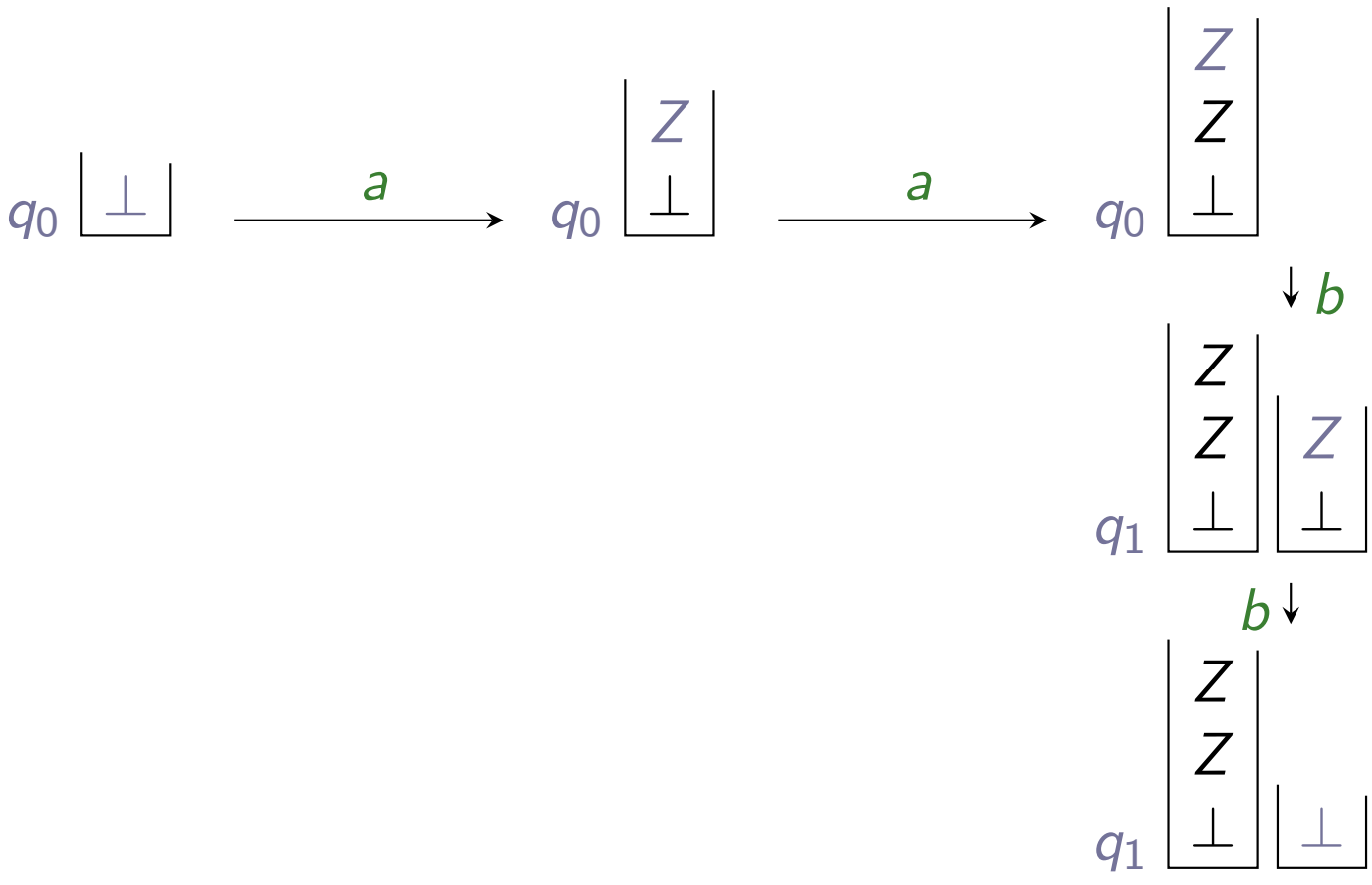
Example $\{a^n b^n c^n | \mid n \geq 0\}$

Accepting *aabbcc*



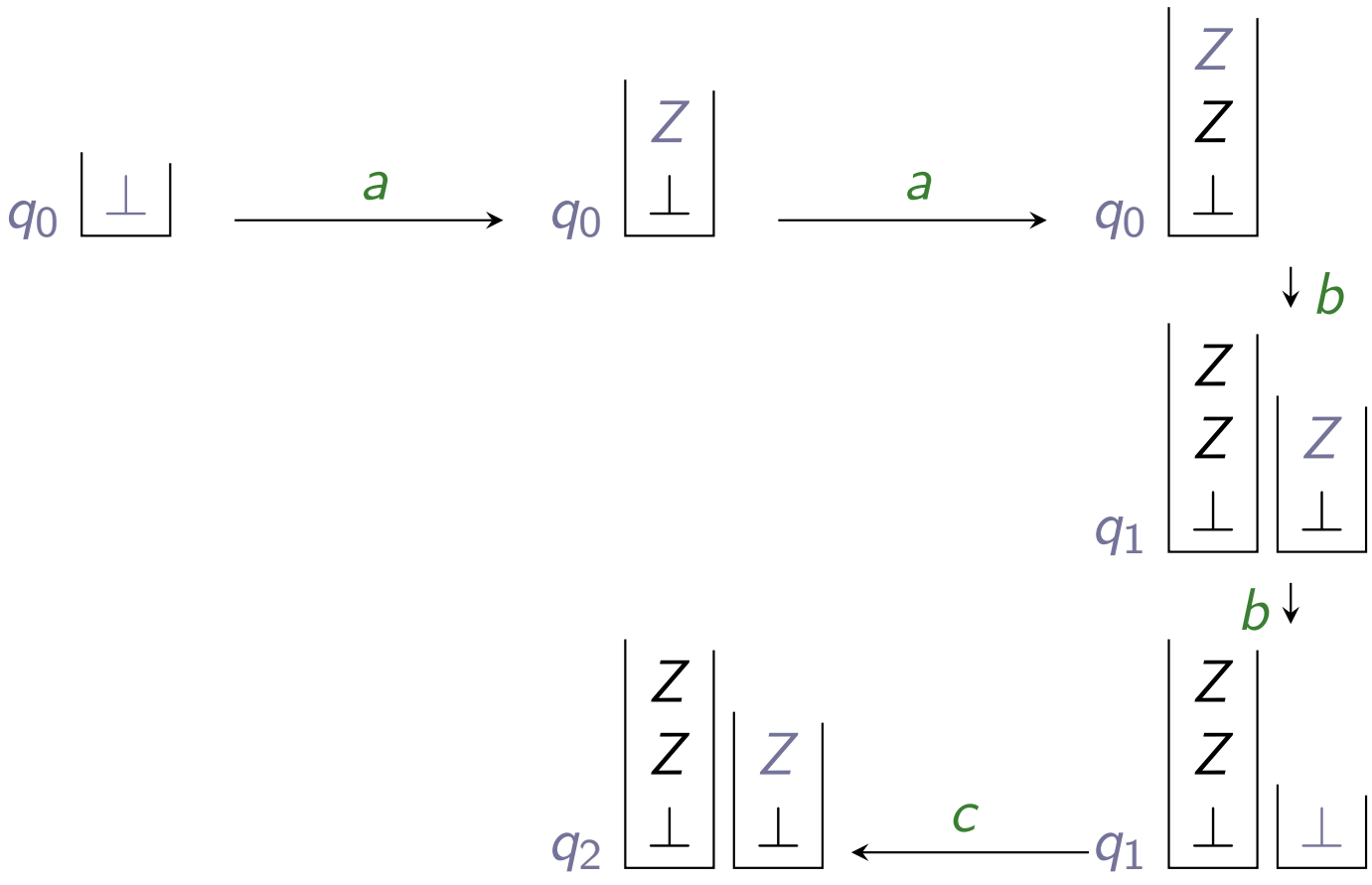
Example $\{a^n b^n c^n \mid \mid n \geq 0\}$

Accepting *aabbcc*



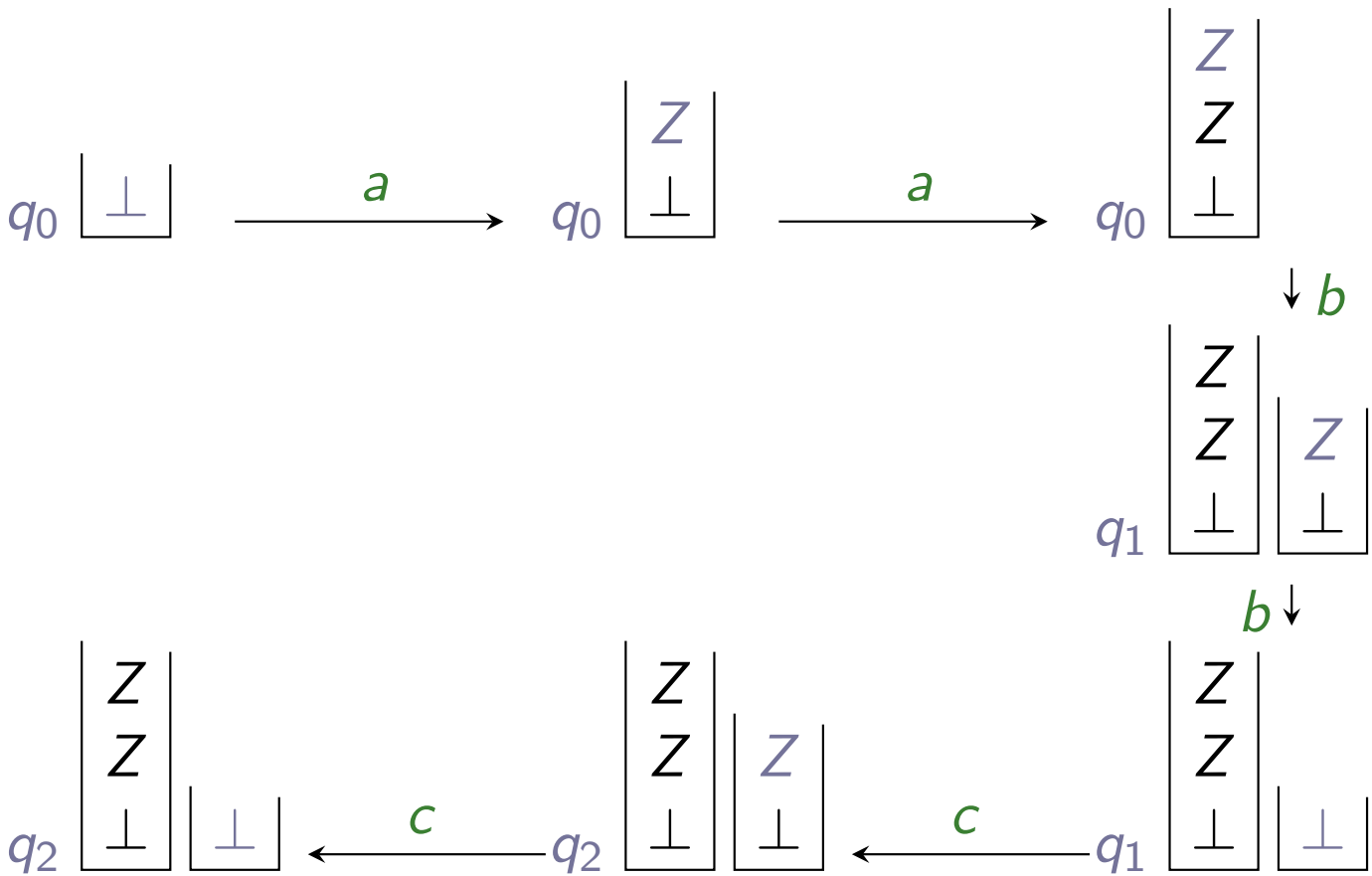
Example $\{a^n b^n c^n \mid \mid n \geq 0\}$

Accepting *aabbcc*



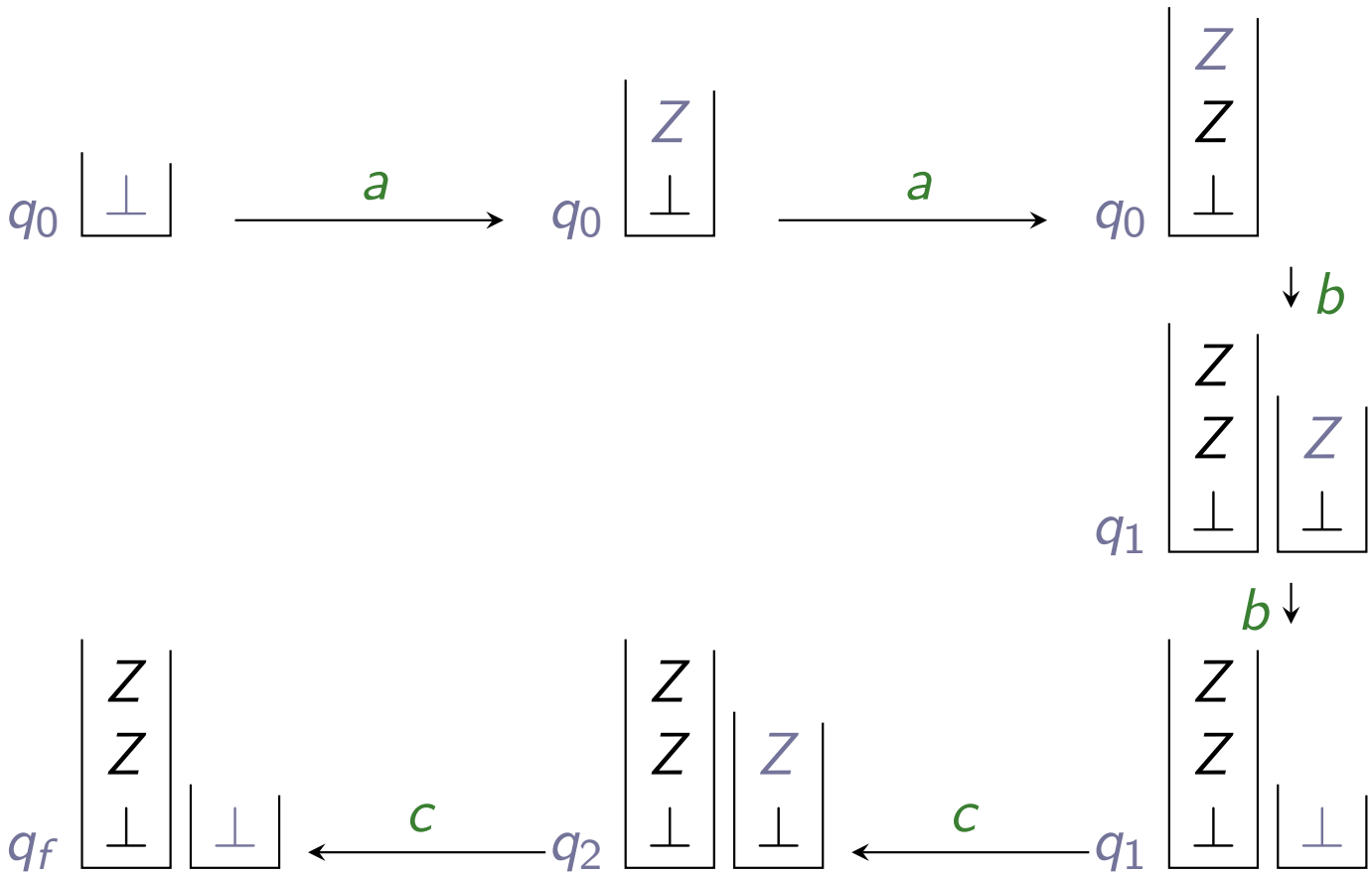
Example $\{a^n b^n c^n \mid \mid n \geq 0\}$

Accepting *aabbcc*



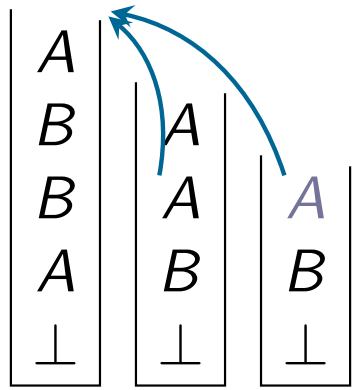
Example $\{a^n b^n c^n \mid \mid n \geq 0\}$

Accepting *aabbcc*



Order-2 collapsible automata

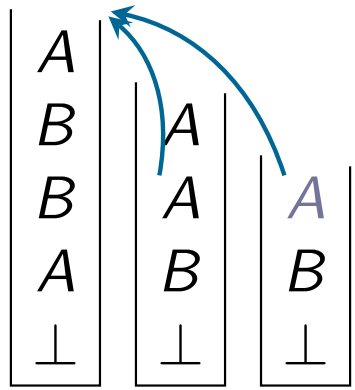
Finite control + stack of (ordinary) stacks with links



Two new operations : $\mathrm{p}ushlk_B + \mathrm{c}ollapse$

Order-2 collapsible automata

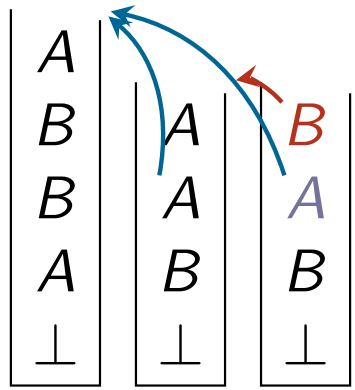
Finite control + stack of (ordinary) stacks with links



$\mathrm{p}ushlk_B$

Order-2 collapsible automata

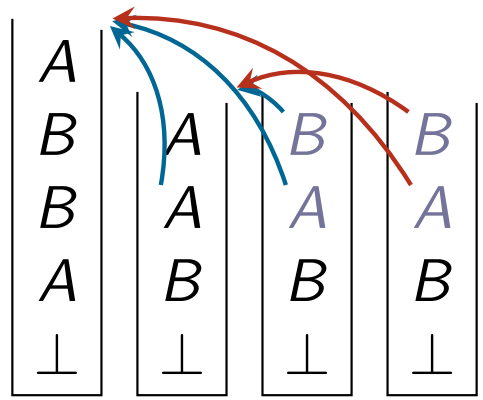
Finite control + stack of (ordinary) stacks with links



$\mathrm{p}ush_2$

Order-2 collapsible automata

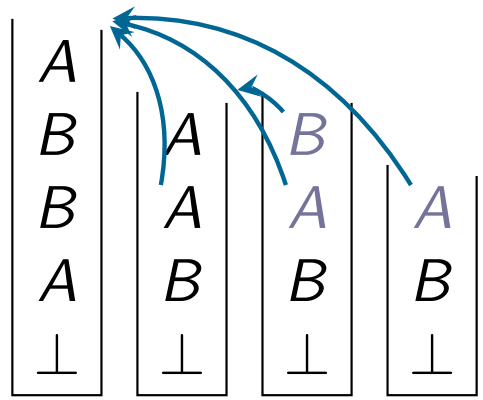
Finite control + stack of (ordinary) stacks with links



pop

Order-2 collapsible automata

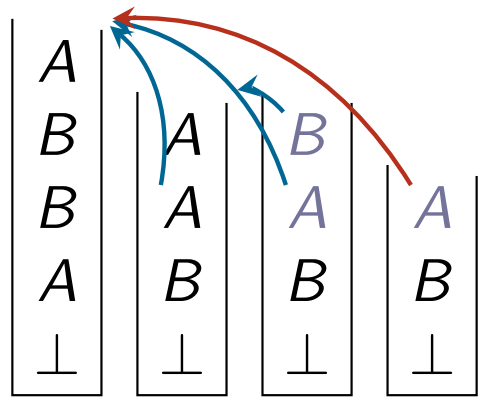
Finite control + stack of (ordinary) stacks with links



collapse

Order-2 collapsible automata

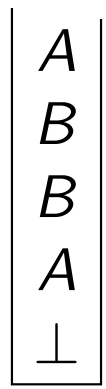
Finite control + stack of (ordinary) stacks with links



collapse

Order-2 collapsible automata

Finite control + stack of (ordinary) stacks with links



The Urzyczyn language U

Alphabet: $\{+, -, \$\}$.

To a word w over $+, -$, we associate a integer $\| w \|$ defined by:

$$\| w \| = (\text{number of } + \text{ in } w) - (\text{number of } - \text{ in } w).$$

E.g. $\| \varepsilon \| = 0$, $\| + + - - ++ \| = 2$ and $\| + - -- \| = -2$.

The Urzyczyn language U

Alphabet: $\{+, -, \$\}$.

To a word w over $+, -$, we associate a integer $\| w \|$ defined by:

$$\| w \| = (\text{number of } + \text{ in } w) - (\text{number of } - \text{ in } w).$$

E.g. $\| \varepsilon \| = 0$, $\| + + - - ++ \| = 2$ and $\| + - -- \| = -2$.

A word is positive if the value of all its prefixes (except ε) is strictly positive.

	+		+		−		+		+		−	
0		1		2		1		2		3		2

The Urzyczyn language U

Alphabet: $\{+, -, \$\}$.

To a word w over $+, -$, we associate a integer $\| w \|$ defined by:

$$\| w \| = (\text{number of } + \text{ in } w) - (\text{number of } - \text{ in } w).$$

E.g. $\| \varepsilon \| = 0$, $\| + + - - ++ \| = 2$ and $\| + - -- \| = -2$.

A word is positive if the value of all its prefixes (except ε) is strictly positive.

	+		+		−		+		+		−	
0		1		2		1		2		3		2

Definition

The language U is the set of all words of the form $w\$^n$ where:

- $w \in \{+, -\}^*$ is a positive word
- n is the number of occurrences of $+$ in the longest prefix of w with value $\| w \| - 1$.

The Urzyczyn language U

Alphabet: $\{+, -, \$\}$.

To a word w over $+, -$, we associate a integer $\| w \|$ defined by:

$$\| w \| = (\text{number of } + \text{ in } w) - (\text{number of } - \text{ in } w).$$

E.g. $\| \varepsilon \| = 0$, $\| ++--++ \| = 2$ and $\| +--- \| = -2$.

A word is positive if the value of all its prefixes (except ε) is strictly positive.

$$\begin{array}{ccccccccccc} & + & & + & & - & & + & & + & & - & \\ 0 & & 1 & & 2 & & 1 & & 2 & & 3 & & 2 \end{array}$$

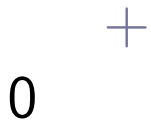
Definition

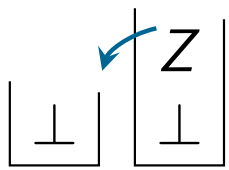
The language U is the set of all words of the form $w\$^n$ where:

- $w \in \{+, -\}^*$ is a positive word
- n is the number of occurrences of $+$ in the longest prefix of w with value $\| w \| - 1$.

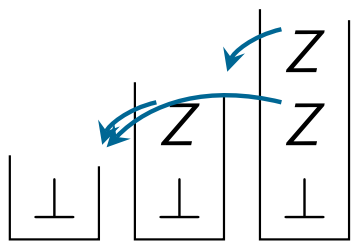
E.g. $++-++-\$\$$ belongs to U.

Order-2 collapsible aut. for U

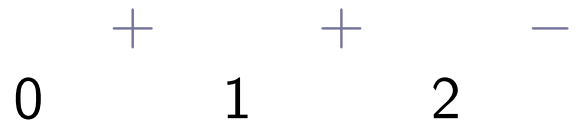


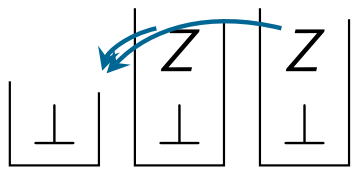


Order-2 collapsible aut. for U

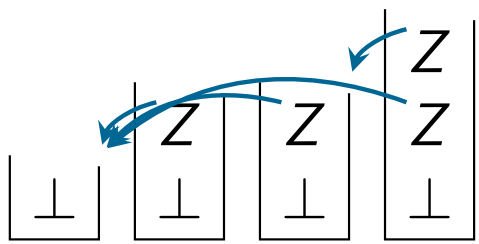


Order-2 collapsible aut. for U

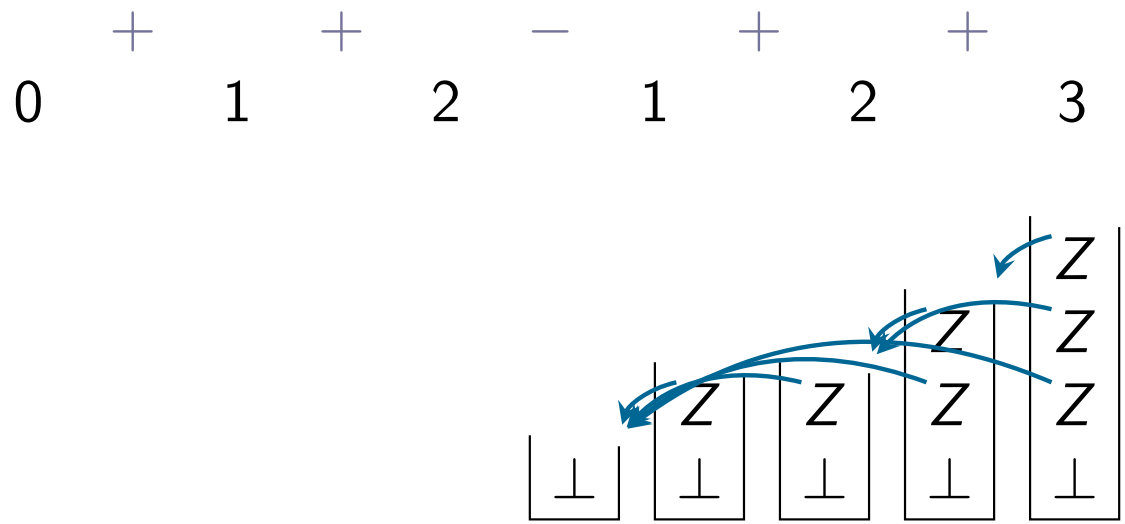




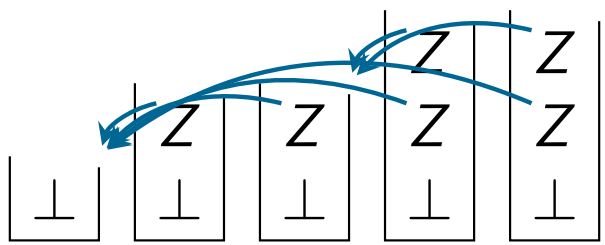
Order-2 collapsible aut. for U



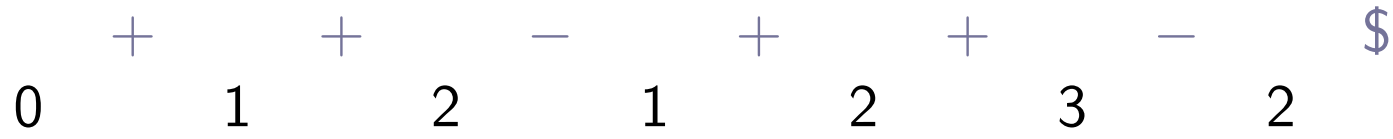
Order-2 collapsible aut. for U

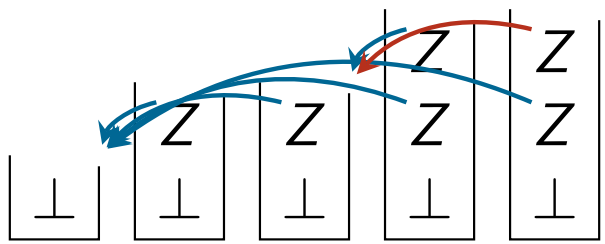


Order-2 collapsible aut. for U

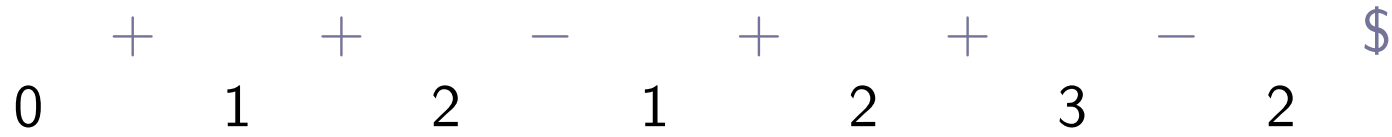


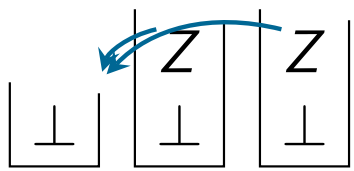
Order-2 collapsible aut. for U



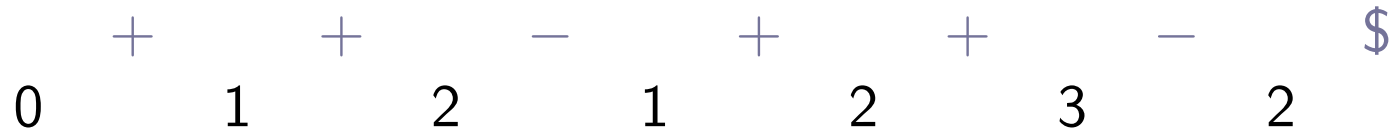


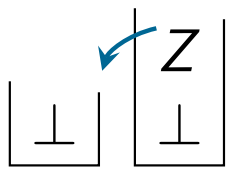
Order-2 collapsible aut. for U



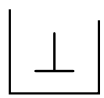


Order-2 collapsible aut. for U





Order-2 collapsible aut. for U



Coming back to the main theorem

Theorem

The infinite tree of a word language L is definable by an order 2 recursion scheme iff L is accepted by a deterministic collapsible automaton of order 2.

Simulating schemes by automata

$$\begin{cases} S & \xrightarrow{a} & FC \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,y & \xrightarrow{b} & y \end{cases} \qquad \begin{array}{lcl} B\,z & \xrightarrow{b} & z \\ C & \xrightarrow{c} & V \end{array}$$

with $x, y, z : \mathbf{o}$.

Alphabet : $\{S, F\,C, z, V, F(B\,x), B\,x, y\}$.

Simulating schemes by automata

$$\begin{cases} S & \xrightarrow{a} & FC & \qquad & B\,z & \xrightarrow{b} & z \\ F\,x & \xrightarrow{a} & F\,(B\,x) & & C & \xrightarrow{c} & V \\ F\,y & \xrightarrow{b} & y & & & & \end{cases}$$

with $x, y, z : \mathbf{o}$.

Alphabet : $\{S, F\,C, z, V, F(B\,x), B\,x, y\}$.

$$S \xrightarrow{a} F\,C \xrightarrow{a} F\,(B\,C) \xrightarrow{b} B\,C \xrightarrow{b} C \xrightarrow{c} V$$

$$\boxed{S}$$

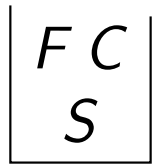
Simulating schemes by automata

$$\begin{cases} S & \xrightarrow{a} & FC & \qquad & B\,z & \xrightarrow{b} & z \\ F\,x & \xrightarrow{a} & F\,(B\,x) & & C & \xrightarrow{c} & V \\ F\,y & \xrightarrow{b} & y & & & & \end{cases}$$

with $x, y, z : \mathbf{o}$.

Alphabet : $\{S, F\,C, z, V, F(B\,x), B\,x, y\}$.

$$S \xrightarrow{a} F\,C \xrightarrow{a} F\,(B\,C) \xrightarrow{b} B\,C \xrightarrow{b} C \xrightarrow{c} V$$



Simulating schemes by automata

$$\begin{cases} S & \xrightarrow{a} & F\,C \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,y & \xrightarrow{b} & y \end{cases} \qquad \begin{array}{lcl} B\,z & \xrightarrow{b} & z \\ C & \xrightarrow{c} & V \end{array}$$

with $x, y, z : \mathbf{o}$.

Alphabet : $\{S, F\,C, z, V, F(B\,x), B\,x, y\}$.

$$S \xrightarrow{a} F\,C \xrightarrow{a} F\,(B\,C) \xrightarrow{b} B\,C \xrightarrow{b} C \xrightarrow{c} V$$

$$\boxed{\begin{array}{c} F\,(B\,x) \\ F\,C \\ S \end{array}}$$

Simulating schemes by automata

$$\begin{cases} S & \xrightarrow{a} & F\,C & \qquad & B\,z & \xrightarrow{b} & z \\ F\,x & \xrightarrow{a} & F\,(B\,x) & & C & \xrightarrow{c} & V \\ F\,y & \xrightarrow{b} & y & & & & \end{cases}$$

with $x, y, z : \mathbf{o}$.

Alphabet : $\{S, F\,C, z, V, F(B\,x), B\,x, y\}$.

$$S \xrightarrow{a} F\,C \xrightarrow{a} F\,(B\,C) \xrightarrow{b} B\,C \xrightarrow{b} C \xrightarrow{c} V$$

$$\left|\begin{array}{c} y \\ F\,(B\,x) \\ F\,C \\ S \end{array}\right|$$

Simulating schemes by automata

$$\begin{cases} S & \xrightarrow{a} & F\,C \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,y & \xrightarrow{b} & y \end{cases} \qquad \begin{array}{lcl} B\,z & \xrightarrow{b} & z \\ C & \xrightarrow{c} & V \end{array}$$

with $x, y, z : \mathbf{o}$.

Alphabet : $\{S, F\,C, z, V, F(B\,x), B\,x, y\}$.

$$S \xrightarrow{a} F\,C \xrightarrow{a} F\,(B\,C) \xrightarrow{b} B\,C \xrightarrow{b} C \xrightarrow{c} V$$

$$\left|\begin{array}{c} B\,x \\ F\,C \\ S \end{array}\right|$$

Simulating schemes by automata

$$\begin{cases} S & \xrightarrow{a} & F\,C \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,y & \xrightarrow{b} & y \end{cases} \qquad \begin{array}{lcl} B\,z & \xrightarrow{b} & z \\ C & \xrightarrow{c} & V \end{array}$$

with $x, y, z : \mathbf{o}$.

Alphabet : $\{S, F\,C, z, V, F(B\,x), B\,x, y\}$.

$$S \xrightarrow{a} F\,C \xrightarrow{a} F\,(B\,C) \xrightarrow{b} B\,C \xrightarrow{b} C \xrightarrow{c} V$$

$$\left|\begin{array}{c} z \\ B\,x \\ F\,C \\ S \end{array}\right.$$

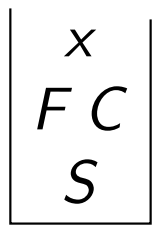
Simulating schemes by automata

$$\begin{cases} S & \xrightarrow{a} & F\,C & \qquad & B\,z & \xrightarrow{b} & z \\ F\,x & \xrightarrow{a} & F\,(B\,x) & & C & \xrightarrow{c} & V \\ F\,y & \xrightarrow{b} & y & & & & \end{cases}$$

with $x, y, z : \mathbf{o}$.

Alphabet : $\{S, F\,C, z, V, F(B\,x), B\,x, y\}$.

$$S \xrightarrow{a} F\,C \xrightarrow{a} F\,(B\,C) \xrightarrow{b} B\,C \xrightarrow{b} C \xrightarrow{c} V$$



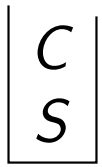
Simulating schemes by automata

$$\begin{cases} S & \xrightarrow{a} & F\,C \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,y & \xrightarrow{b} & y \end{cases} \qquad \begin{array}{lcl} B\,z & \xrightarrow{b} & z \\ C & \xrightarrow{c} & V \end{array}$$

with $x, y, z : \mathbf{o}$.

Alphabet : $\{S, F\,C, z, V, F(B\,x), B\,x, y\}$.

$$S \xrightarrow{a} F\,C \xrightarrow{a} F\,(B\,C) \xrightarrow{b} B\,C \xrightarrow{b} C \xrightarrow{c} V$$



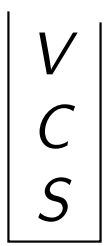
Simulating schemes by automata

$$\begin{cases} S & \xrightarrow{a} & F\,C \\ F\,x & \xrightarrow{a} & F\,(B\,x) \\ F\,y & \xrightarrow{b} & y \end{cases} \qquad \begin{array}{lcl} B\,z & \xrightarrow{b} & z \\ C & \xrightarrow{c} & V \end{array}$$

with $x, y, z : \mathbf{o}$.

Alphabet : $\{S, F\,C, z, V, F(B\,x), B\,x, y\}$.

$$S \xrightarrow{a} F\,C \xrightarrow{a} F\,(B\,C) \xrightarrow{b} B\,C \xrightarrow{b} C \xrightarrow{c} V$$



... at order 2

$$\begin{cases} S & \xrightarrow{a} & F\ B\ C \\ F\ \phi\ \psi & \xrightarrow{d} & \phi\ (\psi\ V) \\ B\ x & \xrightarrow{b} & x \\ C\ x & \xrightarrow{b} & x \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

... at order 2

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) \\ B\,x & \xrightarrow{b} & x \\ C\,x & \xrightarrow{b} & x \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

Alphabet : $\{S, F\,B\,C, B, C, \phi\,(\psi\,V), (\psi\,V), x\}$

$$S \xrightarrow{a} F\,B\,C \xrightarrow{a} B(CV) \xrightarrow{b} C\,V \xrightarrow{c} V$$

$\boxed{S}$

... at order 2

$$\begin{cases} S & \xrightarrow{a} & F\ B\ C \\ F\ \phi\ \psi & \xrightarrow{d} & \phi\ (\psi\ V) \\ B\ x & \xrightarrow{b} & x \\ C\ x & \xrightarrow{b} & x \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

Alphabet : $\{S, F\ B\ C, B, C, \phi\ (\psi\ V), (\psi\ V), x\}$

$$S \xrightarrow{a} F\ B\ C \xrightarrow{a} B(CV) \xrightarrow{b} C\ V \xrightarrow{c} V$$

$$\boxed{\begin{array}{c} F\ B\ C \\ S \end{array}}$$

... at order 2

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) \\ B\,x & \xrightarrow{b} & x \\ C\,x & \xrightarrow{b} & x \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

Alphabet : $\{S, F\,B\,C, B, C, \phi\,(\psi\,V), (\psi\,V), x\}$

$$S \xrightarrow{a} F\,B\,C \xrightarrow{a} B(CV) \xrightarrow{b} C\,V \xrightarrow{c} V$$

$$\boxed{\begin{array}{c} \phi\,(\psi\,V) \\ F\,B\,C \\ S \end{array}}$$

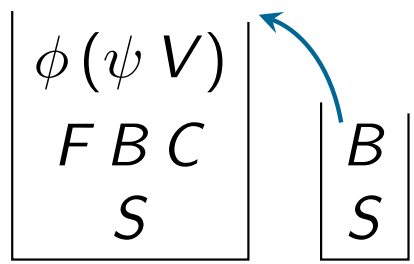
... at order 2

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) \\ B\,x & \xrightarrow{b} & x \\ C\,x & \xrightarrow{b} & x \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

Alphabet : $\{S, F\,B\,C, B, C, \phi\,(\psi\,V), (\psi\,V), x\}$

$$S \xrightarrow{a} F\,B\,C \xrightarrow{a} B(CV) \xrightarrow{b} C\,V \xrightarrow{c} V$$



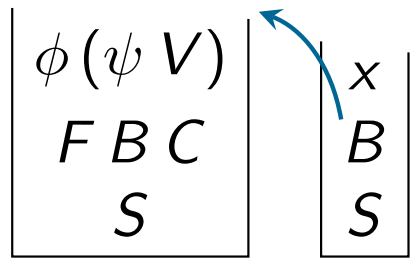
... at order 2

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) \\ B\,x & \xrightarrow{b} & x \\ C\,x & \xrightarrow{b} & x \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

Alphabet : $\{S, F\,B\,C, B, C, \phi\,(\psi\,V), (\psi\,V), x\}$

$$S \xrightarrow{a} F\,B\,C \xrightarrow{a} B(CV) \xrightarrow{b} C\,V \xrightarrow{c} V$$



... at order 2

$$\begin{cases} S & \xrightarrow{a} & F\ B\ C \\ F\ \phi\ \psi & \xrightarrow{d} & \phi\ (\psi\ V) \\ B\ x & \xrightarrow{b} & x \\ C\ x & \xrightarrow{b} & x \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

Alphabet : $\{S, F\ B\ C, B, C, \phi\ (\psi\ V), (\psi\ V), x\}$

$$S \xrightarrow{a} F\ B\ C \xrightarrow{a} B(CV) \xrightarrow{b} C\ V \xrightarrow{c} V$$

$$\boxed{\begin{array}{c} \psi\ V \\ F\ B\ C \\ S \end{array}}$$

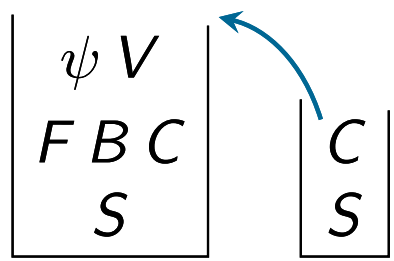
... at order 2

$$\begin{cases} S & \xrightarrow{a} & F\,B\,C \\ F\,\phi\,\psi & \xrightarrow{d} & \phi\,(\psi\,V) \\ B\,x & \xrightarrow{b} & x \\ C\,x & \xrightarrow{b} & x \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

Alphabet : $\{S, F\,B\,C, B, C, \phi\,(\psi\,V), (\psi\,V), x\}$

$$S \xrightarrow{a} F\,B\,C \xrightarrow{a} B(CV) \xrightarrow{b} C\,V \xrightarrow{c} V$$



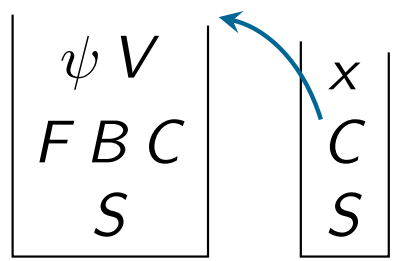
... at order 2

$$\begin{cases} S & \xrightarrow{a} & F\ B\ C \\ F\ \phi\ \psi & \xrightarrow{d} & \phi\ (\psi\ V) \\ B\ x & \xrightarrow{b} & x \\ C\ x & \xrightarrow{b} & x \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

Alphabet : $\{S, F\ B\ C, B, C, \phi\ (\psi\ V), (\psi\ V), x\}$

$$S \xrightarrow{a} F\ B\ C \xrightarrow{a} B(CV) \xrightarrow{b} C\ V \xrightarrow{c} V$$



... at order 2

$$\begin{cases} S & \xrightarrow{a} & F\ B\ C \\ F\ \phi\ \psi & \xrightarrow{d} & \phi\ (\psi\ V) \\ B\ x & \xrightarrow{b} & x \\ C\ x & \xrightarrow{b} & x \end{cases}$$

with $x : \mathbf{o}$, $\phi, \psi : \mathbf{o} \to \mathbf{o}$.

Alphabet : $\{S, F\ B\ C, B, C, \phi\ (\psi\ V), (\psi\ V), x\}$

$$S \xrightarrow{a} F\ B\ C \xrightarrow{a} B(CV) \xrightarrow{b} C\ V \xrightarrow{c} V$$

$$\boxed{\begin{array}{c} V \\ F\ B\ C \\ S \end{array}}$$

Open questions

Expressivity

- What graphs, linear orders can be MSO-defined on schemes ?
- Strictness of the hierarchy
- Pumping lemma for collapsible automata
- Decidability of isomorphism

Connection with programming languages

- What ML-like language and with which evaluation policy can be "compiled" into collapsible automata
- Tools
  $\Rightarrow$ efficient algorithms for collapsible automata