An Introduction to Higher-Order Recursion Schemes and Pushdown Automata

Luke Ong

Oxford University Computing Laboratory

Workshop on Higher-Order Recursion Schemes and Pushdown Automata, 10-12 March 2010, Paris

Aim: survey and introduction

Two families of generators of infinite structures (word and tree languages, and graphs): higher-order recursion schemes and pushdown automata.

Three questions:

1. Expressivity of the generator families
2. Relationship between generator families
3. Algorithmic properties of the generator families

Motivations

1. Understand connexions between semantics (structures) and verification (algorithmics).

   Fully abstract semantics of PCF and recursion schemes. Algorithmics of game semantics. Type theory.

2. Computer-aided verification of higher-order computation: a challenge.

   What are the models of higher-order computation amenable to verification by model checking?

Aim: survey and introduction

Two families of generators of infinite structures (word and tree languages, and graphs): higher-order recursion schemes and pushdown automata.

Three questions:

1. Expressivity of the generator families
2. Relationship between generator families
3. Algorithmic properties of the generator families

Motivations

1. Understand connexions between semantics (structures) and verification (algorithmics).

   Fully abstract semantics of PCF and recursion schemes. Algorithmics of game semantics. Type theory.

2. Computer-aided verification of higher-order computation: a challenge.

   What are the models of higher-order computation amenable to verification by model checking?

Outline I

1. Relating Generator Families: The Maslov Hierarchy
    - Higher-Order Pushdown Automata
    - Higher-Order Recursion Schemes
    - Relating Expressivity of the Generator Families

2. Recursion Schemes, CPDA and their Algorithmics
    - Q1: Decidability of MSO / Modal Mu-Calculus Theories
    - Q2: Machine Characterization by Collapsible Pushdown Automata
    - Q3: Expressivity: *The Safety Conjecture*
    - Q4: Infinite Graphs Generated by Recursion Schemes / CPDA

Outline

1. Relating Generator Families: The Maslov Hierarchy
   - Higher-Order Pushdown Automata
   - Higher-Order Recursion Schemes
   - Relating Expressivity of the Generator Families

2. Recursion Schemes, CPDA and their Algorithmics
   - Q1: Decidability of MSO / Modal Mu-Calculus Theories
   - Q2: Machine Characterization by Collapsible Pushdown Automata
   - Q3: Expressivity: *The Safety Conjecture*
   - Q4: Infinite Graphs Generated by Recursion Schemes / CPDA

Higher-order pushdown automata (HOPDA) [Maslov 74, 76]

Order-2 pushdown automata. A 1-stack is an ordinary stack. A 2-stack (resp. $n+1$-stack) is a stack of 1-stacks (resp. n-stack).

Operations on 2-stacks: s_i ranges over 1-stacks. Top of stack is at the righthand end.

$$
\begin{array}{rcll}
push_2 & : & [s_1 \cdots s_{i-1} \underbrace{[a_1 \cdots a_n]}_{s_i}] & \mapsto [s_1 \cdots s_{i-1}\, s_i\, s_i] \\
pop_2 & : & [s_1 \cdots s_{i-1} [a_1 \cdots a_n]] & \mapsto [s_1 \cdots s_{i-1}] \\
push_1\, a & : & [s_1 \cdots s_{i-1} [a_1 \cdots a_n]] & \mapsto [s_1 \cdots s_{i-1} [a_1 \cdots a_n\, a]] \\
pop_1 & : & [s_1 \cdots s_{i-1} [a_1 \cdots a_n\, a_{n+1}]] & \mapsto [s_1 \cdots s_{i-1} [a_1 \cdots a_n]]
\end{array}
$$

Idea extends to all finite orders: an order-n PDA has an order-n stack, and has $push_i$ and pop_i for each $1 \leq i \leq n$.

N.B. Several equivalent versions: Multilevel stack automata (Maslov); Iterated pushdown automata (Engelfriet); copy + $\overline{\text{copy}}$ (Arnaud)

Higher-order pushdown automata (HOPDA) [Maslov 74, 76]

Order-2 pushdown automata. A 1-stack is an ordinary stack. A 2-stack (resp. $n+1$-stack) is a stack of 1-stacks (resp. n-stack).

Operations on 2-stacks: s_i ranges over 1-stacks. Top of stack is at the righthand end.

$$
\begin{array}{rcll}
push_2 & : & [s_1 \cdots s_{i-1} \underbrace{[a_1 \cdots a_n]}_{s_i}] & \mapsto \quad [s_1 \cdots s_{i-1}\, s_i\, s_i] \\
pop_2 & : & [s_1 \cdots s_{i-1} [a_1 \cdots a_n]] & \mapsto \quad [s_1 \cdots s_{i-1}] \\
push_1\, a & : & [s_1 \cdots s_{i-1} [a_1 \cdots a_n]] & \mapsto \quad [s_1 \cdots s_{i-1} [a_1 \cdots a_n\, a]] \\
pop_1 & : & [s_1 \cdots s_{i-1} [a_1 \cdots a_n\, a_{n+1}]] & \mapsto \quad [s_1 \cdots s_{i-1} [a_1 \cdots a_n]]
\end{array}
$$

Idea extends to all finite orders: an order-n PDA has an order-n stack, and has $push_i$ and pop_i for each $1 \leq i \leq n$.

N.B. Several equivalent versions: Multilevel stack automata (Maslov); Iterated pushdown automata (Engelfriet); copy + $\overline{\text{copy}}$ (Arnaud)

Order-2 pushdown automata. A 1-stack is an ordinary stack. A 2-stack (resp. $n+1$-stack) is a stack of 1-stacks (resp. n-stack).

Operations on 2-stacks: s_i ranges over 1-stacks. Top of stack is at the righthand end.

$$
\begin{array}{rcll}
push_2 & : & [s_1 \cdots s_{i-1} \underbrace{[a_1 \cdots a_n]}_{s_i}] & \mapsto \quad [s_1 \cdots s_{i-1}\, s_i\, s_i] \\
pop_2 & : & [s_1 \cdots s_{i-1} [a_1 \cdots a_n]] & \mapsto \quad [s_1 \cdots s_{i-1}] \\
push_1\, a & : & [s_1 \cdots s_{i-1} [a_1 \cdots a_n]] & \mapsto \quad [s_1 \cdots s_{i-1} [a_1 \cdots a_n\, a]] \\
pop_1 & : & [s_1 \cdots s_{i-1} [a_1 \cdots a_n\, a_{n+1}]] & \mapsto \quad [s_1 \cdots s_{i-1} [a_1 \cdots a_n]]
\end{array}
$$

Idea extends to all finite orders: an order-n PDA has an order-n stack, and has $push_i$ and pop_i for each $1 \leq i \leq n$.

N.B. Several equivalent versions: Multilevel stack automata (Maslov); Iterated pushdown automata (Engelfriet); copy + $\overline{\text{copy}}$ (Arnaud)

Example: $L_3 := \{\, a^n\, b^n\, c^n \mid n \geq 0 \,\}$ is recognizable by an order-2 PDA

(L is not context free. Use the "*uvwxy* Lemma".)

Idea: Use top 1-stack to process $a^n\, b^n$, and height of 2-stack to remember n.

$$
\begin{tikzcd}
q_1\,[[\,]] \arrow[r, "a"] & q_1\,[[\,]\,[z]] \arrow[r, "a"] & q_1\,[[\,]\,[z]\,[zz]] \arrow[d, "b"] \\
 & & q_2\,[[\,]\,[z]\,[z]] \arrow[d, "b"] \\
q_3\,[[\,]] & q_3\,[[\,]\,[z]] \arrow[l, "c"'] & q_2\,[[\,]\,[z]\,[\,]] \arrow[l, "c"']
\end{tikzcd}
$$

Similarly, for every $m \geq 0$, $L_m := \{\, \underbrace{a_1^n\, a_2^n\, a_3^n \cdots a_m^n}_{m} \mid n \geq 0 \,\}$, is recognizable by order-2 PDA.

Example: $L_3 := \{\, a^n\, b^n\, c^n \mid n \geq 0 \,\}$ is recognizable by an order-2 PDA

(L is not context free. Use the "*uvwxy* Lemma".)

Idea: Use top 1-stack to process $a^n\, b^n$, and height of 2-stack to remember n.

$$
\begin{tikzcd}
q_1\,[[\,]] \arrow[r, "a"] & q_1\,[[\,]\,[z]] \arrow[r, "a"] & q_1\,[[\,]\,[z]\,[zz]] \arrow[d, "b"] \\
& & q_2\,[[\,]\,[z]\,[z]] \arrow[d, "b"] \\
q_3\,[[\,]] & q_3\,[[\,]\,[z]] \arrow[l, "c"'] & q_2\,[[\,]\,[z]\,[\,]] \arrow[l, "c"']
\end{tikzcd}
$$

Similarly, for every $m \geq 0$, $L_m := \{\, \underbrace{a_1^n\, a_2^n\, a_3^n \cdots a_m^n}_{m} \mid n \geq 0 \,\}$, is recognizable by order-2 PDA.

Example: $L_3 := \{ a^n \, b^n \, c^n \mid n \geq 0 \}$ is recognizable by an order-2 PDA

(L is not context free. Use the "*uvwxy* Lemma".)

Idea: Use top 1-stack to process $a^n \, b^n$, and height of 2-stack to remember n.

$$
\begin{tikzcd}
q_1 \, [[\,]] \arrow[r, "a"] & q_1 \, [[\,]\,[z]] \arrow[r, "a"] & q_1 \, [[\,]\,[z]\,[zz]] \arrow[d, "b"] \\
 & & q_2 \, [[\,]\,[z]\,[z]] \arrow[d, "b"] \\
q_3 \, [[\,]] & q_3 \, [[\,]\,[z]] \arrow[l, "c"'] & q_2 \, [[\,]\,[z]\,[\,]] \arrow[l, "c"']
\end{tikzcd}
$$

Similarly, for every $m \geq 0$, $L_m := \{ \underbrace{a_1^n \, a_2^n \, a_3^n \cdots a_m^n}_{m} \mid n \geq 0 \}$, is recognizable by order-2 PDA.

Example: $L_3 := \{\, a^n\, b^n\, c^n \mid n \geq 0 \,\}$ is recognizable by an order-2 PDA

(L is not context free. Use the "*uvwxy* Lemma".)

Idea: Use top 1-stack to process $a^n\, b^n$, and height of 2-stack to remember n.

$$\begin{tikzcd}
q_1\,[\,[\,]\,] \arrow[r, "a"] & q_1\,[\,[\,]\,[z]\,] \arrow[r, "a"] & q_1\,[\,[\,]\,[z]\,[zz]\,] \arrow[d, "b"] \\
 & & q_2\,[\,[\,]\,[z]\,[z]\,] \arrow[d, "b"] \\
q_3\,[\,[\,]\,] & q_3\,[\,[\,]\,[z]\,] \arrow[l, "c"'] & q_2\,[\,[\,]\,[z]\,[\,]\,] \arrow[l, "c"']
\end{tikzcd}$$

Similarly, for every $m \geq 0$, $L_m := \{\, \underbrace{a_1^n\, a_2^n\, a_3^n \cdots a_m^n}_{m} \mid n \geq 0 \,\}$, is recognizable by order-2 PDA.

HOPDA as recognizers of word languages

Some old results (Maslov 74, 76):

1. HOPDA define an infinite hierarchy of word languages.
2. Low orders are well-known: orders 0, 1 and 2 are the regular, context free, and indexed languages (Aho 68).
3. For each $n \geq 0$, the order-n languages form an abstract family of languages.
4. For each $n \geq 0$, the emptiness problem for order-n PDA is decidable.

HOPDA can also be used as a recognize / generate

1. ranked trees (KNU01, KNU02), and tree languages
2. graphs (Muller+Schupp 86, Courcelle 95, Cachat 03, etc.)

HOPDA as recognizers of word languages

Some old results (Maslov 74, 76):

1. HOPDA define an infinite hierarchy of word languages.
2. Low orders are well-known: orders 0, 1 and 2 are the regular, context free, and indexed languages (Aho 68).
3. For each $n \geq 0$, the order-n languages form an abstract family of languages.
4. For each $n \geq 0$, the emptiness problem for order-n PDA is decidable.

HOPDA can also be used as a recognize / generate

1. ranked trees (KNU01, KNU02), and tree languages
2. graphs (Muller+Schupp 86, Courcelle 95, Cachat 03, etc.)

Simple types: a review

Types $\qquad A \quad ::= \quad o \quad | \quad (A \to B)$

Every type can be written uniquely as

$$A_1 \to (A_2 \cdots \to (A_n \to o) \cdots), \quad n \geq 0$$

often abbreviated to $A_1 \to A_2 \cdots \to A_n \to o$.

Order of a type: measures "nestedness" on LHS of $\to$.

$$\begin{aligned} \mathrm{order}(o) &:= 0 \\ \mathrm{order}(A \to B) &:= \max(\mathrm{order}(A) + 1, \mathrm{order}(B)) \end{aligned}$$

Examples. $\mathbb{N} \to \mathbb{N}$ and $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$ both have order 1; $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ has order 2.

Notation. $\quad e : A \quad$ means "expression e has type A".

Simple types: a review

Types $\quad A \quad ::= \quad \mathsf{o} \quad | \quad (A \to B)$

Every type can be written uniquely as

$$A_1 \to (A_2 \cdots \to (A_n \to \mathsf{o}) \cdots), \quad n \geq 0$$

often abbreviated to $A_1 \to A_2 \cdots \to A_n \to \mathsf{o}$.

Order of a type: measures "nestedness" on LHS of $\to$.

$$\begin{array}{rcl} \mathrm{order}(\mathsf{o}) & := & 0 \\ \mathrm{order}(A \to B) & := & \max(\mathrm{order}(A) + 1, \mathrm{order}(B)) \end{array}$$

Examples. $\mathbb{N} \to \mathbb{N}$ and $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$ both have order 1;
$(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ has order 2.

Notation. $\quad e : A \quad$ means "expression e has type A".

Recursive program schemes

- Park 68(?); Nivat 72, Nivat+Courcelle 78, Guessarian 81, etc.
- A calculus of first-order recursive procedures that separates control structures from operations on data; a framework for analysing expressivity of control structures and program transformations.
- A large literature on the semantics and transformation of program schemes (Courcelle MIT Handbook 1990).

Higher-order recursion schemes (and precursors)

- Extended to derived types, as generators of trees and tree languages (Damm 77, DFI78) and word languages (Damm 82).
- Comparative schematology and expressivity of dynamic logic with higher-order procedures (KNT89); simulating higher-order stacks by higher-order recursion (KTU92).
- An order-n recursion scheme = "closed ground-type term definable in order-n fragment of simply-typed λ-calculus with recursion and uninterpreted order-1 constant symbols". (Statman's λY-calculus)

Program schemes and higher-order recursion schemes: some history

Recursive program schemes

- Park 68(?); Nivat 72, Nivat+Courcelle 78, Guessarian 81, etc.
- A calculus of first-order recursive procedures that separates control structures from operations on data; a framework for analysing expressivity of control structures and program transformations.
- A large literature on the semantics and transformation of program schemes (Courcelle MIT Handbook 1990).

Higher-order recursion schemes (and precursors)

- Extended to derived types, as generators of trees and tree languages (Damm 77, DFI78) and word languages (Damm 82).
- Comparative schematology and expressivity of dynamic logic with higher-order procedures (KNT89); simulating higher-order stacks by higher-order recursion (KTU92).
- An order-n recursion scheme = "closed ground-type term definable in order-n fragment of simply-typed λ-calculus with recursion and uninterpreted order-1 constant symbols". (Statman's λY-calculus)

Example: An order-1 recursion scheme. Ranked alphabet (i.e. each symbol has an arity) $\Sigma = \{ f : 2, g : 1, a : 0 \}$.

$$G \; : \; \begin{cases} S &= F\,a \\ F\,x &= f\,x\,(F\,(g\,x)) \end{cases}$$

Unfolding from the start symbol S:

$$\begin{array}{rcl} S & \to & F\,a \\ & \to & f\,a\,(F\,(g\,a)) \\ & \to & f\,a\,(f\,(g\,a)\,(F\,(g\,(g\,a)))) \\ & \to & \cdots \end{array}$$

The (term-)tree generated, $[\![\, G \,]\!]$, is $f\,a\,(f\,(g\,a)\,(f\,(g\,(g\,a))(\cdots)))$.

Term-trees such as $[\![\, G \,]\!]$ are ranked and ordered.

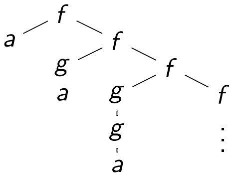
Example: An order-1 recursion scheme. Ranked alphabet (i.e. each symbol has an arity) $\Sigma = \{ f : 2, g : 1, a : 0 \}$.

$$G \ : \ \begin{cases} S &= F\,a \\ F\,x &= f\,x\,(F\,(g\,x)) \end{cases}$$

Unfolding from the start symbol S:

$$\begin{array}{rcl} S & \rightarrow & F\,a \\ & \rightarrow & f\,a\,(F\,(g\,a)) \\ & \rightarrow & f\,a\,(f\,(g\,a)\,(F\,(g\,(g\,a)))) \\ & \rightarrow & \cdots \end{array}$$

The (term-)tree generated, $[\![\, G \,]\!]$, is $f\,a\,(f\,(g\,a)\,(f\,(g\,(g\,a))(\cdots)))$.



Term-trees such as $[\![\, G \,]\!]$ are ranked and ordered.

Assume deterministic schemes. Redex is a term of shape $F\, s_1 \cdots s_{\mathsf{ar}(F)} : o$.

Examples of reduction strategy $\to_{\mathcal{S}}$:

1. Unrestricted: $\to_{\mathrm{unr}}$
2. Outside-In (only contract outermost redexes): $\to_{OI}$
3. Inside-Out (only contract innermost redexes): $\to_{IO}$
4. Others: "square reduction" (Paolini + O. 2010), etc.

For a term t, define a tree $t^{\perp} := \begin{cases} f & \text{if } t \text{ is a terminal } f \\ t_1^{\perp}\, t_2^{\perp} & \text{if } t = t_1\, t_2 \text{ and } t_1^{\perp} \neq \perp \\ \perp & \text{otherwise} \end{cases}$

Define $t \leq t'$ if "t' obtainable from t by replacing some $\perp$ by terms".

For G a recursion scheme, define the $\mathcal{S}$-tree generated by G by

$$[\![\, G \,]\!]^{\mathcal{S}} := \bigsqcup \{\, t^{\perp} \mid S \to_{\mathcal{S}}^{*} t \,\}.$$

Lemma. $[\![-]\!]^{\mathrm{unr}} = [\![-]\!]^{OI} \neq [\![-]\!]^{IO}$. (Henceforth assume $[\![-]\!]^{\mathrm{unr}}$.)

Tree generated by a recursion scheme (in accord with strategy $\to_{\mathcal{S}}$)

Assume deterministic schemes. Redex is a term of shape $F\, s_1 \cdots s_{\mathrm{ar}(F)} : o$.
Examples of reduction strategy $\to_{\mathcal{S}}$:

1. Unrestricted: $\to_{\mathrm{unr}}$
2. Outside-In (only contract outermost redexes): $\to_{OI}$
3. Inside-Out (only contract innermost redexes): $\to_{IO}$
4. Others: "square reduction" (Paolini + O. 2010), etc.

For a term t, define a tree $t^{\perp} := \begin{cases} f & \text{if } t \text{ is a terminal } f \\ t_1^{\perp}\, t_2^{\perp} & \text{if } t = t_1\, t_2 \text{ and } t_1^{\perp} \neq \perp \\ \perp & \text{otherwise} \end{cases}$
Define $t \leq t'$ if "t' obtainable from t by replacing some $\perp$ by terms".

For G a recursion scheme, define the $\mathcal{S}$-tree generated by G by

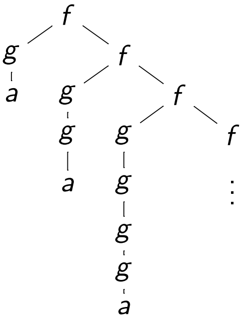
$$[\![\, G \,]\!]^{\mathcal{S}} := \bigsqcup \{\, t^{\perp} \mid S \to_{\mathcal{S}}^{*} t \,\}.$$

Lemma. $[\![-]\!]^{\mathrm{unr}} = [\![-]\!]^{OI} \neq [\![-]\!]^{IO}$. (Henceforth assume $[\![-]\!]^{\mathrm{unr}}$.)

$\Sigma = \{ f : 2, g : 1, a : 0 \}$.
$S : o, \quad B : (o \to o) \to (o \to o) \to o \to o, \quad F : (o \to o) \to o$

$$G_2 \;:\; \left\{ \begin{array}{rcl} S & = & F\, g \\ B\,\varphi\,\psi\,x & = & \varphi\,(\psi\,x) \\ F\,\varphi & = & f\,(\varphi\,a)\,(F\,(B\,\varphi\,\varphi)) \end{array} \right.$$

The generated tree, $[\![\, G_2 \,]\!] : \{ 1, 2 \}^* \longrightarrow \Sigma$, is:



Using recursion schemes as generators of word languages

Represent a finite word "$a\,b\,c$" (say) as the applicative term $a\,(b\,(c\,e)) : o$, where e is a distinguished nullary end-of-word marker.

Example. $\{\, a^n\, b^n \mid n \geq 0 \,\}$ is generated by order-1 recursion scheme:

$$\left\{ \begin{array}{rcl} S & \to & F\,e \\ F\,x & \to & a\,(F\,(b\,x)) \quad | \quad x \end{array} \right.$$

$\{\, a^n b^n c^n \mid n \geq 0 \,\}$ is generated by order-2 scheme:

$$\left\{ \begin{array}{rcl} S & \to & F\,I\,e \\ F\,\varphi\,x & \to & \varphi\,x \quad | \quad F\,(H\,\varphi)\,(c\,x) \\ H\,\varphi\,y & \to & a\,(\varphi\,(b\,y)) \\ I\,x & \to & x \end{array} \right.$$

Both languages can be generated by deterministic schemes.

The Maslov Hierarchy of Word Languages

Theorem (Equi-expressivity)

For each $n \geq 0$, the three formalisms

1. *order-n pushdown automata (Maslov 76)*
2. *order-n safe recursion schemes (or equivalently, satisfying the constraint of derived types) (Damm 82, Damm + Goerdt 86)*
3. *order-n indexed grammars (Maslov 76)*

generate the same family of word languages.

What is safety? (See later.)

Engelfriet's complexity results

Virtually all complexity results of higher-order pushdown systems have been obtained by reduction to one of the following.

Theorem (Engelfriet 1991)

Let $s(n) \geq \log(n)$.

(i) *For* $k \geq 0$, *the word acceptance problem of non-deterministic order-k pushdown automata augmented with a two-way work-tape with* $s(n)$ *space is k-EXPTIME complete.*

(ii) *For* $k \geq 1$, *the word acceptance problem of alternating order-k pushdown automata augmented with a two-way work-tape with* $s(n)$ *space is* $(k-1)$*-EXPTIME complete.*

(iii) *For* $k \geq 0$, *the word acceptance problem of alternating order-k pushdown automata is k-EXPTIME complete.*

(iv) *For* $k \geq 1$, *the emptiness problem of non-deterministic order-k pushdown automata is* $(k-1)$*-EXPTIME complete.*

Maslov Hierarchy: Some Open Problems

1. Pumping Lemma, Myhill-Nerode, and Parikh Theorems.
   Weak "pumping lemmas" for levels 1 and 2 (Hayashi 73, Gilman 96).
   Pace (Blumensath 08, and his talk) for Maslov Hierarchy – runs (not plays) are pumpable, conditions given as lengths of runs and configuration size.
2. Logical Characterizations.
   E.g. MSOL for regular languages (Büchi 60). Characterization of CFL using quantification over matchings (LST 94).
3. Complexity-Theoretic Characterizations.
   Pace (Engelfriet 91): characterizations of languages accepted by alternating / two-way / multi-head / space-auxiliary order-n PDA as time-complexity classes.
   E.g. What is the power (complexity class) of the deterministic Maslov Hierarchy?
4. Relationship with Chomsky Hierachy.
   E.g. Is level 3 context-sensitive?

Outline

1. Relating Generator Families: The Maslov Hierarchy
   - Higher-Order Pushdown Automata
   - Higher-Order Recursion Schemes
   - Relating Expressivity of the Generator Families

2. Recursion Schemes, CPDA and their Algorithmics
   - Q1: Decidability of MSO / Modal Mu-Calculus Theories
   - Q2: Machine Characterization by Collapsible Pushdown Automata
   - Q3: Expressivity: *The Safety Conjecture*
   - Q4: Infinite Graphs Generated by Recursion Schemes / CPDA

A challenge problem in higher-order verification

Let **RecSchTree**$_n$ be the class of Σ-labelled trees generated by order-n recursion schemes.

Is the "MSO Model-Checking Problem for **RecSchTree**$_n$" decidable?

- INSTANCE: An order-n recursion scheme G, and an MSO formula φ
- QUESTION: Does the Σ-labelled tree $[\![G]\!]$ satisfy φ?

A challenge problem in higher-order verification

Let $\mathbf{RecSchTree}_n$ be the class of Σ-labelled trees generated by order-n recursion schemes.

Is the "MSO Model-Checking Problem for $\mathbf{RecSchTree}_n$" decidable?

- INSTANCE: An order-n recursion scheme G, and an MSO formula φ
- QUESTION: Does the Σ-labelled tree $[\![G]\!]$ satisfy φ?

A (selective) survey of related MSO-decidable structures: up to 2002

- Rabin 1969: Regular trees. "Mother of all decidability results in Verification."
- Muller and Schupp 1985: Configuration graphs of PDA.
- Caucal 1996 Prefix-recognizable graphs (ϵ-closures of configuration graphs of pushdown automata, Stirling 2000).
- Knapik, Niwiński and Urzyczyn (TLCA 2001, FoSSaCS 2002): $\mathbf{PushdownTree}_n\Sigma$ = Trees generated by order-n pushdown automata. $\mathbf{SafeRecSchTree}_n\Sigma$ = Trees generated by order-n safe rec. schemes.
- Subsuming all the above: The Caucal Hierarchies (MFCS 2002). $\mathbf{CaucalTree}_n\Sigma$ and $\mathbf{CaucalGraph}_n\Sigma$.

Theorem (KNU-Caucal 2002)

For $n \geq 0$, $\mathbf{PushdownTree}_n\Sigma = \mathbf{SafeRecSchTree}_n\Sigma = \mathbf{CaucalTree}_n\Sigma$; and they have decidable MSO theories.

A (selective) survey of related MSO-decidable structures: up to 2002

- Rabin 1969: Regular trees. "Mother of all decidability results in Verification."
- Muller and Schupp 1985: Configuration graphs of PDA.
- Caucal 1996 Prefix-recognizable graphs (ϵ-closures of configuration graphs of pushdown automata, Stirling 2000).
- Knapik, Niwiński and Urzyczyn (TLCA 2001, FoSSaCS 2002):
  $\mathbf{PushdownTree}_n\Sigma$ = Trees generated by order-n pushdown automata.
  $\mathbf{SafeRecSchTree}_n\Sigma$ = Trees generated by order-n safe rec. schemes.
- Subsuming all the above:
  The Caucal Hierarchies (MFCS 2002). $\mathbf{CaucalTree}_n\Sigma$ and $\mathbf{CaucalGraph}_n\Sigma$.

Theorem (KNU-Caucal 2002)

For $n \geq 0$, $\mathbf{PushdownTree}_n\Sigma = \mathbf{SafeRecSchTree}_n\Sigma = \mathbf{CaucalTree}_n\Sigma$; and they have decidable MSO theories.

A (selective) survey of related MSO-decidable structures: up to 2002

- Rabin 1969: Regular trees. "Mother of all decidability results in Verification."
- Muller and Schupp 1985: Configuration graphs of PDA.
- Caucal 1996 Prefix-recognizable graphs (ϵ-closures of configuration graphs of pushdown automata, Stirling 2000).
- Knapik, Niwiński and Urzyczyn (TLCA 2001, FoSSaCS 2002): $\mathbf{PushdownTree}_n\Sigma$ = Trees generated by order-n pushdown automata. $\mathbf{SafeRecSchTree}_n\Sigma$ = Trees generated by order-n safe rec. schemes.
- Subsuming all the above: The Caucal Hierarchies (MFCS 2002). $\mathbf{CaucalTree}_n\Sigma$ and $\mathbf{CaucalGraph}_n\Sigma$.

Theorem (KNU-Caucal 2002)

For $n \geq 0$, $\mathbf{PushdownTree}_n\Sigma = \mathbf{SafeRecSchTree}_n\Sigma = \mathbf{CaucalTree}_n\Sigma$; and they have decidable MSO theories.

A (selective) survey of related MSO-decidable structures: up to 2002

- Rabin 1969: Regular trees. "Mother of all decidability results in Verification."
- Muller and Schupp 1985: Configuration graphs of PDA.
- Caucal 1996 Prefix-recognizable graphs (ϵ-closures of configuration graphs of pushdown automata, Stirling 2000).
- Knapik, Niwiński and Urzyczyn (TLCA 2001, FoSSaCS 2002):
  $\mathbf{PushdownTree}_n\Sigma$ = Trees generated by order-n pushdown automata.
  $\mathbf{SafeRecSchTree}_n\Sigma$ = Trees generated by order-n safe rec. schemes.
- Subsuming all the above:
  The Caucal Hierarchies (MFCS 2002). $\mathbf{CaucalTree}_n\Sigma$ and $\mathbf{CaucalGraph}_n\Sigma$.

Theorem (KNU-Caucal 2002)

For $n \geq 0$, $\mathbf{PushdownTree}_n\Sigma = \mathbf{SafeRecSchTree}_n\Sigma = \mathbf{CaucalTree}_n\Sigma$; and they have decidable MSO theories.

A (selective) survey of related MSO-decidable structures: up to 2002

- Rabin 1969: Regular trees. "Mother of all decidability results in Verification."
- Muller and Schupp 1985: Configuration graphs of PDA.
- Caucal 1996 Prefix-recognizable graphs (ϵ-closures of configuration graphs of pushdown automata, Stirling 2000).
- Knapik, Niwiński and Urzyczyn (TLCA 2001, FoSSaCS 2002):
  $\mathbf{PushdownTree}_n\Sigma$ = Trees generated by order-n pushdown automata.
  $\mathbf{SafeRecSchTree}_n\Sigma$ = Trees generated by order-n safe rec. schemes.
- Subsuming all the above:
  The Caucal Hierarchies (MFCS 2002). $\mathbf{CaucalTree}_n\Sigma$ and $\mathbf{CaucalGraph}_n\Sigma$.

Theorem (KNU-Caucal 2002)

For $n \geq 0$, $\mathbf{PushdownTree}_n\Sigma = \mathbf{SafeRecSchTree}_n\Sigma = \mathbf{CaucalTree}_n\Sigma$; and they have decidable MSO theories.

A (selective) survey of related MSO-decidable structures: up to 2002

- Rabin 1969: Regular trees. "Mother of all decidability results in Verification."
- Muller and Schupp 1985: Configuration graphs of PDA.
- Caucal 1996 Prefix-recognizable graphs (ϵ-closures of configuration graphs of pushdown automata, Stirling 2000).
- Knapik, Niwiński and Urzyczyn (TLCA 2001, FoSSaCS 2002):
  $\mathbf{PushdownTree}_n\Sigma$ = Trees generated by order-n pushdown automata.
  $\mathbf{SafeRecSchTree}_n\Sigma$ = Trees generated by order-n safe rec. schemes.
- Subsuming all the above:
  The Caucal Hierarchies (MFCS 2002). $\mathbf{CaucalTree}_n\Sigma$ and $\mathbf{CaucalGraph}_n\Sigma$.

Theorem (KNU-Caucal 2002)

For $n \geq 0$, $\mathbf{PushdownTree}_n\Sigma = \mathbf{SafeRecSchTree}_n\Sigma = \mathbf{CaucalTree}_n\Sigma$; and they have decidable MSO theories.

What is the safety constraint on recursion schemes?

Assume that types are homogeneous[1]. Safety is a set of constraints governing where variables may occur in a term.

Definition (Damm TCS 82, KNU FoSSaCS'02)

An order-2 equation is unsafe if the RHS has a subterm P s.t.

1. P is order 1
2. P occurs in an operand position (i.e. as 2nd argument of application)
3. P contains an order-0 parameter.

Consequence: An order-i subterm of a safe term can only have free variables of order at least i.

Example (unsafe eqn): $F : (o \to o) \to o \to o \to o$, $f : o^2 \to o$, $x, y : o$.

$$F\,\varphi\,x\,y \quad = \quad f\,(F\,\underline{(F\,\varphi\,y)}\,y\,(\varphi\,x))\,\underline{a}$$

[1] o is homogeneous; and $(A_1 \to \cdots \to A_n \to o)$ is homogeneous just if $\mathrm{order}(A_1) \geq \mathrm{order}(A_2) \geq \cdots \geq \mathrm{order}(A_n)$, and each A_i is homogeneous.

Assume that types are homogeneous[1]. Safety is a set of constraints governing where variables may occur in a term.

Definition (Damm TCS 82, KNU FoSSaCS'02)

An order-2 equation is unsafe if the RHS has a subterm P s.t.

1. P is order 1
2. P occurs in an operand position (i.e. as 2nd argument of application)
3. P contains an order-0 parameter.

Consequence: An order-i subterm of a safe term can only have free variables of order at least i.

Example (unsafe eqn): $F : (o \to o) \to o \to o \to o,\ f : o^2 \to o,\ x, y : o$.

$$F\,\varphi\,x\,y \quad = \quad f\,(F\,\underline{(F\,\varphi\,y)}\,y\,(\varphi\,x))\,\underline{a}$$

[1] o is homogeneous; and $(A_1 \to \cdots \to A_n \to o)$ is homogeneous just if $\mathrm{order}(A_1) \geq \mathrm{order}(A_2) \geq \cdots \geq \mathrm{order}(A_n)$, and each A_i is homogeneous.

Safe terms enjoy an important algorithmic advantage!

Lemma (KNU 2002, Blum+O. TLCA 2007)

Substitution (hence β-red.) in safe λ-calculus can be safely implemented without renaming bound variables! Hence no fresh names needed.

Expressivity of safety: a characterization

Theorem

1. *(Schwichtenberg 1976) The numeric functions representable by simply-typed λ-terms are multivariate polynomials with conditional.*
2. *(Blum + O. LMCS 09) The numeric functions representable by simply-typed safe λ-terms are the multivariate polynomials.*

(See Blum's thesis for a study on the safe lambda calculus.)

Safe terms enjoy an important algorithmic advantage!

Lemma (KNU 2002, Blum+O. TLCA 2007)

Substitution (hence β-red.) in safe λ-calculus can be safely implemented without renaming bound variables! Hence no fresh names needed.

Expressivity of safety: a characterization

Theorem

1. *(Schwichtenberg 1976) The numeric functions representable by simply-typed λ-terms are multivariate polynomials with conditional.*
2. *(Blum + O. LMCS 09) The numeric functions representable by simply-typed safe λ-terms are the multivariate polynomials.*

(See Blum's thesis for a study on the safe lambda calculus.)

1. **MSO decidability**: Is safety a genuine constraint for decidability?
   I.e. do trees generated by (arbitrary) recursion schemes have decidable MSO theories?
2. **Machine characterization**: Find a hierarchy of automata that characterize the expressive power of recursion schemes.
   I.e. how should the power of higher-order pushdown automata be augmented to achieve equi-expressivity with (arbitrary) recursion schemes?
3. **Expressivity**: Is safety a genuine constraint for expressivity?
   I.e. are there inherently unsafe word languages / trees / graphs?
4. **Graph families**:
   1. **Definition**: What is a good definition of "graphs generated by recursion schemes"?
   2. **Model-checking properties**: What are the decidable (modal-) logical theories of the graph families?

Infinite structures generated by recursion schemes: some questions

1. **MSO decidability**: Is safety a genuine constraint for decidability?
   I.e. do trees generated by (arbitrary) recursion schemes have decidable MSO theories?
2. **Machine characterization**: Find a hierarchy of automata that characterize the expressive power of recursion schemes.
   I.e. how should the power of higher-order pushdown automata be augmented to achieve equi-expressivity with (arbitrary) recursion schemes?
3. **Expressivity**: Is safety a genuine constraint for expressivity?
   I.e. are there inherently unsafe word languages / trees / graphs?
4. **Graph families**:
   1. **Definition**: What is a good definition of "graphs generated by recursion schemes"?
   2. **Model-checking properties**: What are the decidable (modal-) logical theories of the graph families?

Infinite structures generated by recursion schemes: some questions

1. **MSO decidability**: Is safety a genuine constraint for decidability?
   I.e. do trees generated by (arbitrary) recursion schemes have decidable MSO theories?
2. **Machine characterization**: Find a hierarchy of automata that characterize the expressive power of recursion schemes.
   I.e. how should the power of higher-order pushdown automata be augmented to achieve equi-expressivity with (arbitrary) recursion schemes?
3. **Expressivity**: Is safety a genuine constraint for expressivity?
   I.e. are there inherently unsafe word languages / trees / graphs?
4. **Graph families**:
   1. **Definition**: What is a good definition of "graphs generated by recursion schemes"?
   2. **Model-checking properties**: What are the decidable (modal-) logical theories of the graph families?

Infinite structures generated by recursion schemes: some questions

1. **MSO decidability**: Is safety a genuine constraint for decidability?
   I.e. do trees generated by (arbitrary) recursion schemes have decidable MSO theories?
2. **Machine characterization**: Find a hierarchy of automata that characterize the expressive power of recursion schemes.
   I.e. how should the power of higher-order pushdown automata be augmented to achieve equi-expressivity with (arbitrary) recursion schemes?
3. **Expressivity**: Is safety a genuine constraint for expressivity?
   I.e. are there inherently unsafe word languages / trees / graphs?
4. **Graph families**:
   1. **Definition**: What is a good definition of "graphs generated by recursion schemes"?
   2. **Model-checking properties**: What are the decidable (modal-) logical theories of the graph families?

Q1. Do trees in $\mathbf{RecSchTree}_n\Sigma$ have decidable MSO theories? Yes

Theorem (O. LiCS 2006)

For $n \geq 0$, the modal mu-calculus model-checking problem for $\mathbf{RecSchTree}_n\Sigma$ *(i.e. trees generated by order-n recursion schemes) is n-EXPTIME complete. Thus these trees have decidable MSO theories.*

Two key ingredients of the proof:

$[\![G]\!]$ satisifes modal mu-calculus formula φ

$\iff$ { Emerson + Jutla 1991}

APT $\mathcal{B}_\varphi$ has accepting run-tree over generated tree $[\![G]\!]$

$\iff$ { **I. Transference Principle: Traversal-Path Correspondence**}

APT $\mathcal{B}_\varphi$ has accepting traversal-tree over computation tree $\lambda(G)$

$\iff$ { **II. Simulation of traversals by paths** }

APT $\mathcal{C}_\varphi$ has an accepting run-tree over computation tree $\lambda(G)$

which is decidable.

Q1. Do trees in $\mathbf{RecSchTree}_n\Sigma$ have decidable MSO theories? Yes

Theorem (O. LiCS 2006)

For $n \geq 0$, the modal mu-calculus model-checking problem for $\mathbf{RecSchTree}_n\Sigma$ *(i.e. trees generated by order-n recursion schemes) is n-EXPTIME complete. Thus these trees have decidable MSO theories.*

Two key ingredients of the proof:

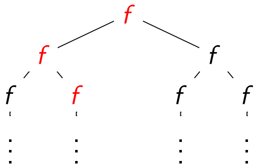
$$
\begin{array}{ll}
 & [\![\, G \,]\!] \text{ satisifes modal mu-calculus formula } \varphi \\
\iff & \{ \text{ Emerson + Jutla 1991} \} \\
 & \text{APT } \mathcal{B}_\varphi \text{ has accepting run-tree over generated tree } [\![\, G \,]\!] \\
\iff & \{ \textbf{ I. Transference Principle: Traversal-Path Correspondence} \} \\
 & \text{APT } \mathcal{B}_\varphi \text{ has accepting traversal-tree over computation tree } \lambda(G) \\
\iff & \{ \textbf{ II. Simulation of traversals by paths } \} \\
 & \text{APT } \mathcal{C}_\varphi \text{ has an accepting run-tree over computation tree } \lambda(G)
\end{array}
$$

which is decidable.

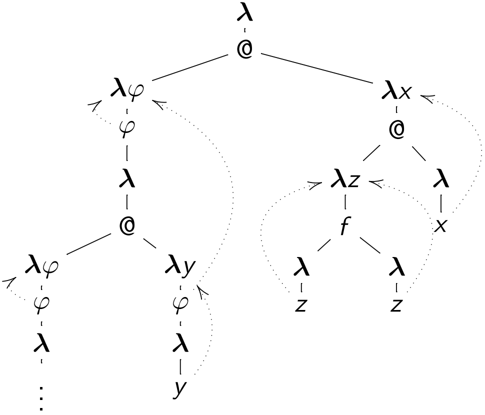
Transference principle, based on a theory of traversals

$$G : \begin{cases} S &= F\,H \\ F\,\varphi &= \varphi\,(F\,\varphi) \\ H\,z &= fzz \end{cases} \quad \mapsto \quad \overline{G} : \begin{cases} S &= \boldsymbol{\lambda}.@\,F\,(\boldsymbol{\lambda}x.@\,H\,\boldsymbol{\lambda}.x) \\ F &= \boldsymbol{\lambda}\varphi.\varphi(\boldsymbol{\lambda}.@\,F\,(\boldsymbol{\lambda}y.\varphi(\boldsymbol{\lambda}.y))) \\ H &= \boldsymbol{\lambda}z.f(\boldsymbol{\lambda}.z)(\boldsymbol{\lambda}.z) \end{cases}$$

$[\![\, G \,]\!]$



$\lambda(G)$



Idea: β-reduction is global (i.e. substitution changes the term being evaluated); game semantics gives an equivalent but local view.
A traversal (over the computation tree $\lambda(G)$) is a trace of the local computation that produces a path (over $[\![G]\!]$).

Theorem (Path-traversal correspondence)

Let G be an order-n recursion scheme.

(i) *There is a 1-1 correspondence between maximal paths p in (Σ-labelled) generated tree $[\![G]\!]$ and maximal traversals t_p over computation tree $\lambda(G)$.*

(ii) *Further for each p, we have $p \upharpoonright \Sigma = t_p \upharpoonright \Sigma$.*

Proof is by game semantics.

Explanation (for game semanticists):

- Term-tree $[\![G]\!]$ is (a representation of) the game semantics of G.
- Paths in $[\![G]\!]$ correspond to plays in the strategy-denotation.
- Traversals t_p over computation tree $\lambda(G)$ are just (representations of) the uncoverings of the plays ($=$ path) p in the game semantics of G.

Idea: β-reduction is global (i.e. substitution changes the term being evaluated); game semantics gives an equivalent but local view.
A traversal (over the computation tree $\lambda(G)$) is a trace of the local computation that produces a path (over $[\![G]\!]$).

Theorem (Path-traversal correspondence)

Let G be an order-n recursion scheme.

(i) *There is a 1-1 correspondence between maximal paths p in (Σ-labelled) generated tree $[\![G]\!]$ and maximal traversals t_p over computation tree $\lambda(G)$.*

(ii) *Further for each p, we have $p \upharpoonright \Sigma = t_p \upharpoonright \Sigma$.*

Proof is by game semantics.

Explanation (for game semanticists):

- Term-tree $[\![G]\!]$ is (a representation of) the game semantics of G.
- Paths in $[\![G]\!]$ correspond to plays in the strategy-denotation.
- Traversals t_p over computation tree $\lambda(G)$ are just (representations of) the uncoverings of the plays (= path) p in the game semantics of G.

Idea: β-reduction is global (i.e. substitution changes the term being evaluated); game semantics gives an equivalent but local view.
A traversal (over the computation tree $\lambda(G)$) is a trace of the local computation that produces a path (over $[\![G]\!]$).

Theorem (Path-traversal correspondence)

Let G be an order-n recursion scheme.

(i) *There is a 1-1 correspondence between maximal paths p in (Σ-labelled) generated tree $[\![G]\!]$ and maximal traversals t_p over computation tree $\lambda(G)$.*

(ii) *Further for each p, we have $p \upharpoonright \Sigma = t_p \upharpoonright \Sigma$.*

Proof is by game semantics.

Explanation (for game semanticists):

- Term-tree $[\![G]\!]$ is (a representation of) the game semantics of G.
- Paths in $[\![G]\!]$ correspond to plays in the strategy-denotation.
- Traversals t_p over computation tree $\lambda(G)$ are just (representations of) the uncoverings of the plays ($=$ path) p in the game semantics of G.

A new proof: Walukiewicz's talk on a model-theoretic approach.

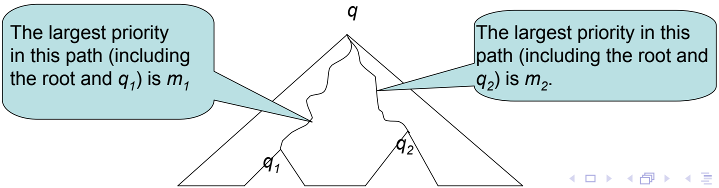
Theorem (Type-Theoretic Characterization. Kobayashi+O. LiCS09)

Given an APT A there is a typing system $\mathcal{K}_A$ such that for every recursion scheme G, the APT A accepts $[\![\, G \,]\!]$ iff G is $\mathcal{K}_A$-typable. Further there is a type-inference algorithm polynomial in size of recursion scheme (assuming other parameters are fixed).

Refine intersection types with states q and priorities m_i of a given APT.

$$\text{Types} \qquad \begin{array}{rcl} \theta & ::= & q \quad | \quad \tau \to \theta \\ \tau & ::= & \bigwedge \{ (\theta_1, m_1), \cdots, (\theta_k, m_k) \} \end{array}$$

Intuition. A tree function described by $(q_1, m_1) \wedge (q_2, m_2) \to q$.



Q2: Machine characterization: collapsible pushdown automata

Order-2 collapsible pushdown automata [HOMS, LiCS 08a] are essentially the same as 2PDA with links [AdMO 05], and panic automata [KNUW 05].

Idea: Each stack symbol in 2-stack "remembers" the stack content at the point it was first created (i.e. $push_1$ed onto the stack), by way of a pointer to some 1-stack underneath it (if there is one such).

Two new stack operations: $a \in \Gamma$ (stack alphabet)

- $push_1\ a$: pushes a onto the top of the top 1-stack, together with a pointer to the 1-stack immediately below the top 1-stack.
- *collapse* (= panic) collapses the 2-stack down to the prefix pointed to by the top_1-element of the 2-stack.

Pointers are created by $push_1^a$'s; they may be replicated by $push_2$'s (the pointer-relation is preserved by $push_2$).

Q2: Machine characterization: collapsible pushdown automata

Order-2 collapsible pushdown automata [HOMS, LiCS 08a] are essentially the same as 2PDA with links [AdMO 05], and panic automata [KNUW 05].

Idea: Each stack symbol in 2-stack "remembers" the stack content at the point it was first created (i.e. $push_1$ed onto the stack), by way of a pointer to some 1-stack underneath it (if there is one such).

Two new stack operations: $a \in \Gamma$ (stack alphabet)

- $push_1\ a$: pushes a onto the top of the top 1-stack, together with a pointer to the 1-stack immediately below the top 1-stack.
- *collapse* (= panic) collapses the 2-stack down to the prefix pointed to by the top_1-element of the 2-stack.

Pointers are created by $push_1^a$'s; they may be replicated by $push_2$'s (the pointer-relation is preserved by $push_2$).

Q2: Machine characterization: collapsible pushdown automata

Order-2 collapsible pushdown automata [HOMS, LiCS 08a] are essentially the same as 2PDA with links [AdMO 05], and panic automata [KNUW 05].

Idea: Each stack symbol in 2-stack "remembers" the stack content at the point it was first created (i.e. $push_1$ed onto the stack), by way of a pointer to some 1-stack underneath it (if there is one such).

Two new stack operations: $a \in \Gamma$ (stack alphabet)

- $push_1\ a$: pushes a onto the top of the top 1-stack, together with a pointer to the 1-stack immediately below the top 1-stack.
- *collapse* (= panic) collapses the 2-stack down to the prefix pointed to by the top_1-element of the 2-stack.

Pointers are created by $push_1^a$'s; they may be replicated by $push_2$'s (the pointer-relation is preserved by $push_2$).

Collapsible pushdown automata: extending to all finite orders

In **order-n CPDA**, there are $n-1$ versions of $push_1$, namely, $push_1^j\ a$, with $1 \leq j \leq n-1$:

> $push_1^j\ a$: *pushes* a *onto the top of the top 1-stack, together with a pointer to the* j*-stack immediately below the top* j*-stack.*

Definition (Aehlig, de Miranda + O. FoSSaCS 05) A U-**word** has 3 segments:

$$\underbrace{(\cdots(\cdots(}_{A}\ \underbrace{(\cdots)\cdots(\cdots)}_{B}\ \underbrace{*\cdots*}_{C}$$

- Segment A is a prefix of a well-bracketed word that ends in **(**, and the opening **(** is not matched in the entire word.
- Segment B is a well-bracketed word.
- Segment C has length equal to the number of **(** in segment A.

Examples

1. $(\,(\,)\,(\,(\,)\,(\,(\,)\,)\,*\,*\,*\ \in\ U$
2. For each $n \geq 0$, we have $(\,(^n\,)^n\,(\,*^n\,*\,*\ \in\ U$. (Hence by "*uvwxy* Lemma", U is not context-free.)

Definition (Aehlig, de Miranda + O. FoSSaCS 05) A *U*-**word** has 3 segments:

$$\underbrace{(\cdots(\cdots(}_{A}\;\underbrace{(\cdots)\cdots(\cdots)}_{B}\;\underbrace{*\cdots*}_{C}$$

- Segment A is a prefix of a well-bracketed word that ends in **(**, and the opening **(** is not matched in the entire word.
- Segment B is a well-bracketed word.
- Segment C has length equal to the number of **(** in segment A.

Examples

1. $(\,(\,)\,(\,(\,)\,(\,(\,)\,)\,*\;*\;*\;\in\;U$
2. For each $n \geq 0$, we have $(\,(^n\,)^n\,(\,*^n\,*\,*\;\in\;U$. (Hence by "*uvwxy* Lemma", U is not context-free.)

Example: Urzyczyn's Language U over alphabet $\{\,(,), *\,\}$

Definition (Aehlig, de Miranda + O. FoSSaCS 05) A **U-word** has 3 segments:

$$\underbrace{(\cdots(\cdots(}_{A}\ \underbrace{(\cdots)\cdots(\cdots)}_{B}\ \underbrace{*\cdots*}_{C}$$

- Segment A is a prefix of a well-bracketed word that ends in **(**, and the opening **(** is not matched in the entire word.
- Segment B is a well-bracketed word.
- Segment C has length equal to the number of **(** in segment A.

Examples

1. $(\,(\,)\,(\,(\,)\,(\,(\,)\,)\,*\ *\ * \in U$
2. For each $n \geq 0$, we have $(\,(^n\,)^n\,(\,*^n\ *\ * \in U$. (Hence by "*uvwxy* Lemma", U is not context-free.)

U is recognizable by a **deterministic** 2CPDA.

E.g. $(\,(\,)\,(\,(\,)\; * \; * \, * \;\in\; U$

Q	Σ	Γ	Op_2^*	Q
q_0	(	$\perp$	$push_2\,;\,push_1^Z$	q_1
q_1	(	Z	$push_2\,;\,push_1^Z$	q_1
q_1	)	Z	pop_1	q_1
q_1	$*$	Z	*collapse*	q_2
q_2	$*$	Z	pop_2	q_2

$q_0, [[\,]]$

$\xrightarrow{(}\quad q_1, [[\,][Z]]$

$\xrightarrow{(}\quad q_1, [[\,][Z][Z\,Z]]$

$\xrightarrow{)}\quad q_1, [[\,][Z][Z]]$

$\xrightarrow{(}\quad q_1, [[\,][Z][Z][Z\,Z]]$

$\xrightarrow{(}\quad q_1, [[\,][Z][Z][Z\,Z][Z\,Z\,Z]]$

$\xrightarrow{)}\quad q_1, [[\,][Z][Z][Z\,Z][Z\,Z]]$ collapse!

$\xrightarrow{*}\quad q_2, [[\,][Z][Z]]$

$\xrightarrow{*}\quad q_2, [[\,][Z]]$

$\xrightarrow{*}\quad q_2, [[\,]]$

U **is recognizable by a deterministic 2CPDA.**

E.g. $(\,(\,)\,(\,(\,)\,*\,*\,*\;\in\;U$

Q	Σ	Γ	Op_2^*	Q
q_0	(	$\perp$	$push_2\,;\,push_1^Z$	q_1
q_1	(	Z	$push_2\,;\,push_1^Z$	q_1
q_1	)	Z	pop_1	q_1
q_1	$*$	Z	$collapse$	q_2
q_2	$*$	Z	pop_2	q_2

$$
\begin{array}{ll}
 & q_0, [[]] \\
\xrightarrow{(} & q_1, [[][Z]] \\
\xrightarrow{(} & q_1, [[][Z][ZZ]] \\
\xrightarrow{)} & q_1, [[][Z][Z]] \\
\xrightarrow{(} & q_1, [[][Z][Z][ZZ]] \\
\xrightarrow{(} & q_1, [[][Z][Z][ZZ][ZZZ]] \\
\xrightarrow{)} & q_1, [[][Z][Z][ZZ][ZZ]] \\
\xrightarrow{*} & q_2, [[][Z][Z]] \\
\xrightarrow{*} & q_2, [[][Z]] \\
\xrightarrow{*} & q_2, [[]]
\end{array}
$$

collapse!

U is recognizable by a **deterministic** 2CPDA.

E.g. $(\,(\,)\,(\,(\,)\; * \; * * \;\in\; U$

Q	Σ	Γ	Op_2^*	Q
q_0	(	$\perp$	$push_2\,;\,push_1^Z$	q_1
q_1	(	Z	$push_2\,;\,push_1^Z$	q_1
q_1	)	Z	pop_1	q_1
q_1	$*$	Z	*collapse*	q_2
q_2	$*$	Z	pop_2	q_2

$$
\begin{array}{ll}
 & q_0, [\,[\,]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{)} & q_1, [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z\,Z]\,] \\
\xrightarrow{)} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z]\,] \qquad \text{collapse!} \\
\xrightarrow{*} & q_2, [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,[Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,]
\end{array}
$$

U is recognizable by a **deterministic** 2CPDA.

E.g. $(\,(\,)\,(\,(\,)\; *\; *\, *\; \in\; U$

Q	Σ	Γ	Op_2^*	Q
q_0	(	$\perp$	$push_2\,;\,push_1^Z$	q_1
q_1	(	Z	$push_2\,;\,push_1^Z$	q_1
q_1	)	Z	pop_1	q_1
q_1	$*$	Z	$collapse$	q_2
q_2	$*$	Z	pop_2	q_2

$$
\begin{array}{ll}
 & q_0, [\,[\,]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{)} & q_1, [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z\,Z]\,] \\
\xrightarrow{)} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z]\,] \qquad \text{collapse!} \\
\xrightarrow{*} & q_2, [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,[Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,]
\end{array}
$$

U is recognizable by a **deterministic** 2CPDA.

E.g. $(\, (\,) \, (\, (\,) \; * \; * \, * \; \in \; U$

Q	Σ	Γ	Op_2^*	Q
q_0	(	$\perp$	$push_2$; $push_1^Z$	q_1
q_1	(	Z	$push_2$; $push_1^Z$	q_1
q_1	)	Z	pop_1	q_1
q_1	$*$	Z	*collapse*	q_2
q_2	$*$	Z	pop_2	q_2

$$q_0, [\,[\,]\,]$$
$$\xrightarrow{(} \quad q_1, [\,[\,]\,[Z]\,]$$
$$\xrightarrow{(} \quad q_1, [\,[\,]\,[Z]\,[Z\,Z]\,]$$
$$\xrightarrow{)} \quad q_1, [\,[\,]\,[Z]\,[Z]\,]$$
$$\xrightarrow{(} \quad q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,]$$
$$\xrightarrow{(} \quad q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z\,Z]\,]$$
$$\xrightarrow{)} \quad q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z]\,]$$

collapse!

$$\xrightarrow{*} \quad q_2, [\,[\,]\,[Z]\,[Z]\,]$$
$$\xrightarrow{*} \quad q_2, [\,[\,]\,[Z]\,]$$
$$\xrightarrow{*} \quad q_2, [\,[\,]\,]$$

U **is recognizable by a deterministic 2CPDA.**

E.g. $(\, (\,) \, (\, (\,) \, * \; * \, * \; \in \; U$

Q	Σ	Γ	Op_2^*	Q
q_0	(	$\perp$	$push_2$; $push_1^Z$	q_1
q_1	(	Z	$push_2$; $push_1^Z$	q_1
q_1	)	Z	pop_1	q_1
q_1	$*$	Z	*collapse*	q_2
q_2	$*$	Z	pop_2	q_2

$$
\begin{array}{ll}
 & q_0, [\,[\,]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{)} & q_1, [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z\,Z]\,] \\
\xrightarrow{)} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z]\,] \qquad \text{collapse!} \\
\xrightarrow{*} & q_2, [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,[Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,]
\end{array}
$$

U is recognizable by a **deterministic** 2CPDA.

E.g. $(( ) (( ) * * * \in U$

Q	Σ	Γ	Op_2^*	Q
q_0	(	$\perp$	$push_2 ; push_1^Z$	q_1
q_1	(	Z	$push_2 ; push_1^Z$	q_1
q_1	)	Z	pop_1	q_1
q_1	$*$	Z	*collapse*	q_2
q_2	$*$	Z	pop_2	q_2

$$
\begin{array}{ll}
 & q_0, [\,[\,]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{)} & q_1, [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z\,Z]\,] \\
\xrightarrow{)} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,[Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,]
\end{array}
$$

collapse!

U is recognizable by a **deterministic** 2CPDA.

E.g. (( ) (( ) $* \; * * \; \in \; U$

Q	Σ	Γ	Op_2^*	Q
q_0	(	$\perp$	$push_2 \,;\, push_1^Z$	q_1
q_1	(	Z	$push_2 \,;\, push_1^Z$	q_1
q_1	)	Z	pop_1	q_1
q_1	$*$	Z	$collapse$	q_2
q_2	$*$	Z	pop_2	q_2

$$
\begin{array}{ll}
 & q_0, [[\,]] \\
\xrightarrow{(} & q_1, [[\,]\,[Z]] \\
\xrightarrow{(} & q_1, [[\,]\,[Z]\,[Z\,Z]] \\
\xrightarrow{)} & q_1, [[\,]\,[Z]\,[Z]] \\
\xrightarrow{(} & q_1, [[\,]\,[Z]\,[Z]\,[Z\,Z]] \\
\xrightarrow{(} & q_1, [[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z\,Z]] \\
\xrightarrow{)} & q_1, [[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z]] \\
\xrightarrow{*} & q_2, [[\,]\,[Z]\,[Z]] \\
\xrightarrow{*} & q_2, [[\,]\,[Z]] \\
\xrightarrow{*} & q_2, [[\,]]
\end{array}
$$

collapse!

U **is recognizable by a deterministic 2CPDA.**

E.g. $(\,(\,)\,(\,(\,)\ *\ **\ \in\ U$

Q	Σ	Γ	Op_2^*	Q
q_0	(	$\perp$	$push_2\,;\,push_1^Z$	q_1
q_1	(	Z	$push_2\,;\,push_1^Z$	q_1
q_1	)	Z	pop_1	q_1
q_1	$*$	Z	$collapse$	q_2
q_2	$*$	Z	pop_2	q_2

$$
\begin{array}{ll}
 & q_0,\ [\,[\,]\,] \\
\xrightarrow{(} & q_1,\ [\,[\,]\,[Z]\,] \\
\xrightarrow{(} & q_1,\ [\,[\,]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{)} & q_1,\ [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{(} & q_1,\ [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{(} & q_1,\ [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z\,Z]\,] \\
\xrightarrow{)} & q_1,\ [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z]\,] \\
\xrightarrow{*} & q_2,\ [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{*} & q_2,\ [\,[\,]\,[Z]\,] \\
\xrightarrow{*} & q_2,\ [\,[\,]\,]
\end{array}
$$

collapse!

U is recognizable by a **deterministic** 2CPDA.

E.g. $(\,(\,)\,(\,(\,)\ *\ *\ *\ \in\ U$

Q	Σ	Γ	Op_2^*	Q
q_0	(	$\perp$	$push_2\,;\,push_1^Z$	q_1
q_1	(	Z	$push_2\,;\,push_1^Z$	q_1
q_1	)	Z	pop_1	q_1
q_1	$*$	Z	*collapse*	q_2
q_2	$*$	Z	pop_2	q_2

$$
\begin{array}{ll}
 & q_0, [\,[\,]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{)} & q_1, [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z\,Z]\,] \\
\xrightarrow{)} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z]\,] \quad \text{collapse!} \\
\xrightarrow{*} & q_2, [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,[Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,]
\end{array}
$$

U **is recognizable**
by a deterministic 2CPDA.

E.g. $(\,(\,)\,(\,(\,)\; * \; * \, * \;\in\; U$

Q	Σ	Γ	Op_2^*	Q
q_0	(	$\perp$	$push_2\,;\,push_1^Z$	q_1
q_1	(	Z	$push_2\,;\,push_1^Z$	q_1
q_1	)	Z	pop_1	q_1
q_1	$*$	Z	$collapse$	q_2
q_2	$*$	Z	pop_2	q_2

$$
\begin{array}{ll}
 & q_0, [\,[\,]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{)} & q_1, [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,] \\
\xrightarrow{(} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z\,Z]\,] \\
\xrightarrow{)} & q_1, [\,[\,]\,[Z]\,[Z]\,[Z\,Z]\,[Z\,Z]\,] \qquad \text{collapse!} \\
\xrightarrow{*} & q_2, [\,[\,]\,[Z]\,[Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,[Z]\,] \\
\xrightarrow{*} & q_2, [\,[\,]\,]
\end{array}
$$

Observation

1. U is recognizable by a deterministic order-2 CPDA.
2. Equivalently (AdMO 05) U is recognizable by a non-deterministic order-2 PDA — power of non-determinacy is needed to guess the transition from segment A to segment B.

Conjecture

U is not recognizable by a deterministic order-2 PDA.

(Related to the Safety Conjecture - more anon.)

But see Paweł Parys' talk: Collapse Operation Increases Expressive Power of Deterministic Higher Order Pushdown Automata

Q2: Recursion schemes are equi-expressive with CPDA

Theorem (Equi-Expressivity, Hague, Murawski, O. + Serre LiCS'08)

For each $n \geq 0$, order-n recursion schemes and order-n collapsible PDA are equi-expressive for Σ-labelled trees. I.e. $\mathbf{RecSchTree}_n\Sigma = \mathbf{CPDATree}_n\Sigma$

(Translation "RS $\to$ CPDA" uses traversals, based on game semantics.)

Consequences:

1. **Kleene's Problem:** What computing power (originally, in terms of game) is required to compute order-n lambda-definable functionals?

   The Theorem gives a syntax-independent automata-theoretic characterization of pure simply-typed lambda-calculus with recursion.

2. A **new proof** of the MSO decidability of trees generated by order-n recursion schemes.

Open Problem. Find a new proof of "RS $\to$ CPDA" without using game semantics. Simulation in [KNU02] does not appear to generalize to unsafe recursion schemes.

Theorem (Equi-Expressivity, Hague, Murawski, O. + Serre LiCS'08)

For each $n \geq 0$, order-n recursion schemes and order-n collapsible PDA are equi-expressive for Σ-labelled trees. I.e. $\mathbf{RecSchTree}_n\Sigma \;=\; \mathbf{CPDATree}_n\Sigma$

(Translation "RS $\rightarrow$ CPDA" uses traversals, based on game semantics.)

Consequences:

1. **Kleene's Problem:** What computing power (originally, in terms of game) is required to compute order-n lambda-definable functionals?

   The Theorem gives a syntax-independent automata-theoretic characterization of pure simply-typed lambda-calculus with recursion.

2. A **new proof** of the MSO decidability of trees generated by order-n recursion schemes.

Open Problem. Find a new proof of "RS $\rightarrow$ CPDA" without using game semantics. Simulation in [KNU02] does not appear to generalize to unsafe recursion schemes.

Theorem (Equi-Expressivity, Hague, Murawski, O. + Serre LiCS'08)

For each $n \geq 0$, order-n recursion schemes and order-n collapsible PDA are equi-expressive for Σ-labelled trees. I.e. $\mathbf{RecSchTree}_n\Sigma \ = \ \mathbf{CPDATree}_n\Sigma$

(Translation "RS $\to$ CPDA" uses traversals, based on game semantics.)

Consequences:

1. **Kleene's Problem:** What computing power (originally, in terms of game) is required to compute order-n lambda-definable functionals?

   The Theorem gives a syntax-independent automata-theoretic characterization of pure simply-typed lambda-calculus with recursion.

2. A **new proof** of the MSO decidability of trees generated by order-n recursion schemes.

Open Problem. Find a new proof of "RS $\to$ CPDA" without using game semantics. Simulation in [KNU02] does not appear to generalize to unsafe recursion schemes.

Q2: Recursion schemes are equi-expressive with CPDA

Theorem (Equi-Expressivity, Hague, Murawski, O. + Serre LiCS'08)

For each $n \geq 0$, order-n recursion schemes and order-n collapsible PDA are equi-expressive for Σ-labelled trees. I.e. $\mathbf{RecSchTree}_n\Sigma \ = \ \mathbf{CPDATree}_n\Sigma$

(Translation "RS $\to$ CPDA" uses traversals, based on game semantics.)

Consequences:

1. **Kleene's Problem:** What computing power (originally, in terms of game) is required to compute order-n lambda-definable functionals?

   The Theorem gives a syntax-independent automata-theoretic characterization of pure simply-typed lambda-calculus with recursion.
2. A **new proof** of the MSO decidability of trees generated by order-n recursion schemes.

Open Problem. Find a new proof of "RS $\to$ CPDA" without using game semantics. Simulation in [KNU02] does not appear to generalize to unsafe recursion schemes.

Q3: Does safety constrain expressivity?

Case 1: Word languages. Conjecture: Yes; but note

Theorem (Aehlig, de Miranda + O., FoSSaCS 2005)

At order 2, there are no inherently unsafe word languages. I.e. for every unsafe order-2 recursion scheme, there is a safe (non-deterministic) order-2 recursion scheme that generates the same language.

Case 2: Trees. Conjecture: Yes.

The Safety Conjecture (many versions)

For each $n \geq 2$, there is a tree generated by an unsafe order-n recursion scheme but not by any safe order-n recursion scheme.

Pace Paweł Parys' recent result.

Q3: Does safety constrain expressivity?

Case 1: Word languages. Conjecture: Yes; but note

Theorem (Aehlig, de Miranda + O., FoSSaCS 2005)

At order 2, there are no inherently unsafe word languages. I.e. for every unsafe order-2 recursion scheme, there is a safe (non-deterministic) order-2 recursion scheme that generates the same language.

Case 2: Trees. Conjecture: Yes.

The Safety Conjecture (many versions)

For each $n \geq 2$, there is a tree generated by an unsafe order-n recursion scheme but not by any safe order-n recursion scheme.

Pace Paweł Parys' recent result.

Q3: Does safety constrain expressivity?

Case 3: Graphs. Yes.

Theorem (Hague, Murawski, O. + Serre LiCS 2008a)

There is an order-2 CPDA graph that is not generated by any order-2 PDA.

(See example graph later.)

Model checking properties of some graph families

	Decidable?			
	MSO	μ	FO(R)	FO
Caucal's Graph Hierarchy	yes	yes	yes	yes
C	no	yes	?	?
Ground-term tree rewriting (Löding 02)	no	no	yes	yes
Automatic graphs (Hodgson 76, KN 94)	no	no	no	yes
Rational graphs	no	no	no	no

Question

Is there a generically-defined family **C** of graphs that have decidable modal-mu calculus theories but undecidable MSO theories?

Yes. See construction on next slide (HMOS, LiCS 08a).
Recent progress on decidability of first-order theories (with / without reachabiity) of classes of CPDA graphs by Kartzow and Broadbent.

Model checking properties of some graph families

	Decidable?			
	MSO	μ	FO(R)	FO
Caucal's Graph Hierarchy	yes	yes	yes	yes
C	no	yes	?	?
Ground-term tree rewriting (Löding 02)	no	no	yes	yes
Automatic graphs (Hodgson 76, KN 94)	no	no	no	yes
Rational graphs	no	no	no	no

Question

Is there a generically-defined family **C** of graphs that have decidable modal-mu calculus theories but undecidable MSO theories?

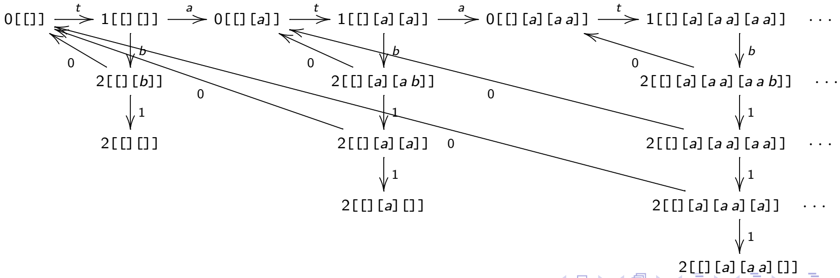
Yes. See construction on next slide (HMOS, LiCS 08a).

Recent progress on decidability of first-order theories (with / without reachabiity) of classes of CPDA graphs by Kartzow and Broadbent.

Theorem (Hague, Murawski, O and Serre, LiCS 2008a)

1. *For each $n \geq 0$, the decidability of modal mu-calculus model-checking problem for configuration graphs of order-n CPDA is n-EXPTIME complete.*
2. *Equivalently solvability of parity games over order-n CPDA graphs is n-EXPTIME complete.*

An order-2 CPDA graph: MSO-interpretable into the infinite half-grid.



Conclusions

Summary

- Higher-order recursion schemes and pushdown automata are robust and highly expressive families of generators of infinite structures. Their algorithmics are rich and interesting.
- Recent progress in the theory have used both *semantic methods* (e.g. game semantics and type theory) as well as more traditional automata-theoretic techniques.

Application (Looking ahead to Kobayashi's talks)

- New (but necessarily highly complex) model-checking algorithms have been obtained.
- The type-inference approach gives rise to a surprisingly efficient implementation.
- Verification of functional programs can be reduced to model checking recursion schemes. The approach is automatic, sound and complete.