Higher-Order Recursion Schemes and Collapsible Pushdown Automata

A Survey and a Tutorial

Luke Ong

University of Oxford

24 September 2011, NII Shonan Meeting

Collaborators: Christopher Broadbent, Arnaud Carayol, Matthew Hague, Naoki Kobayashi, Andrzej Murawski, Olivier Serre

A fundamental question in verification

Fix an logic $\mathcal{L}$. Find a family $\mathcal{C}$ of infinite structures for which the $\mathcal{L}$-Model Checking Problem is decidable.

$\mathcal{L}$-Model Checking Problem for $\mathcal{C}$ "Given $M \in \mathcal{C}$ and an $\mathcal{L}$-formula φ, does $M \vDash \varphi$?"

Examples of infinite structures:
Infinite trees and graphs generated by classes of rewrite systems, abstract machines, automata, Petri nets, etc.

Logics for describing properties:

- Reachability: EF (plain), EGF (recurrent) and AF (universal), etc.
- Temporal logics: LTL, CTL, CTL*, modal mu-calculus, etc.
- FO, FO+Reachability, MSO, FO(TC), etc.

Outline

1. Two families of generators
    - Higher-order pushdown automata
    - Higher-order recursion schemes
    - Relating the two generator-families: word languages case

2. Verifying hierarchies of ranked trees
    - Infinite structures with decidable MSO theories
    - Deciding MSO theories of trees generated by recursion schemes
    - Machine characterization: collapsible pushdown automata (CPDA)
    - Characterising expressivity

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.

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

An order-n PDA has an order-n stack, and has $push_i$ and pop_i for each $1 \leq i \leq n$.

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

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

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

$$
\begin{tikzcd}
q_1 \arrow[loop above, "* \xrightarrow{a} push_2\,;\ push_1 z"] \arrow[rr, "z \xrightarrow{b} pop_1 z"'] & & q_2 \arrow[loop above, "z \xrightarrow{b} pop_1"] \arrow[rr, "\perp \xrightarrow{c} pop_2"'] & & q_3 \arrow[loop above, "z \xrightarrow{c} pop_2"]
\end{tikzcd}
$$

'read a' 'read b' 'read c'

Some Properties of the Maslov Hierarchy (Maslov'76, Engelfriet'91)

(i) Higher-order pushdown automata define an infinite hierarchy of word languages; for each n, the order-n languages form an AFL.

(ii) For $k \geq 1$, the emptiness problem of non-deterministic order-k pushdown automata 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) Let $s(n) \geq \log(n)$. 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.

There are no similar complexity characterisations of languages recognisable by higher-order deterministic pushdown automata.

Order-n recursion scheme $G = (\mathcal{N}, \Sigma, \mathcal{R}, S)$

[Park'68, Nivat'72, NC'78, Damm'82,...] Fix a ranked alphabet Σ.

Recursion schemes are a simply-typed grammar for generating possibly-infinite, Σ-labelled ranked trees.

"Recursion schemes = (a version of) PCF"

Example: An order-1 recursion scheme. $\Sigma = \{ f : 2, g : 1, a : 0 \}$. Take

$$G_1 \ : \ \begin{cases} S & = & F\,a \\ F\,x & = & f\,x\,(F\,(g\,x)) \end{cases}$$

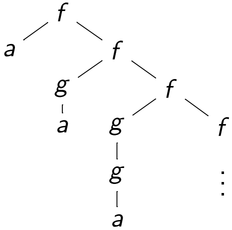
We have

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

The tree $[\![\, G_1 \,]\!]$ thus generated is $f\,a\,(f\,(g\,a)\,(f\,(g\,(g\,a))(\cdots)))$.

Representing $[\![G]\!]$ as a Σ-labelled tree

$[\![G]\!] = f\, a\, (f\, (g\, a)\, (f\, (g\, (g\, a))(\cdots)))$ is the (term-)tree



We view the infinite term $[\![G]\!]$ as a Σ-labelled tree.

To generate trees, we assume recursion schemes are deterministic.
Non-deterministic recursion schemes generate tree languages.

Relating the two generator-families: word-language case

Theorem (Equi-expressivity)

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

1. *order-n pushdown automata (Maslov 76)*
2. *order-n safe recursion schemes (Damm 82, Damm + Goerdt 86)*
3. *order-n indexed grammars (Maslov 76)*

generate the same class of word languages.

What is safety? (See later.)

RecSchTree$_n$: Σ-labelled trees generated by order-n recursion schemes.

MSO Model-Checking Problem for **RecSchTree**$_n$

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

Represent Σ-labelled trees t as structures of the vocabulary:

- $\mathbf{d}_i(x, y) \equiv$ "y is i-child of x"
- $\mathbf{p}_f(x) \equiv$ "x has label f" where f is a Σ-symbol

Monadic second-order (MSO) logic

First-order variables: x, y, z, etc. (ranging over *nodes*)
Second-order variables: X, Y, Z, etc. (ranging over *sets* of nodes)
MSO formulas are built up from atomic formulas—namely, $\mathbf{d}_i(x, y), \mathbf{p}_f(x)$ and set-membership $x \in X$—and closed under boolean connectives, 1st-order quantification ($\forall x.-, \exists x.-$) and 2nd-order quantifications ($\forall X.-, \exists X.-$).

A Survey of MSO-decidable structures: time-line 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 ($= \epsilon$-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:
  Caucal (MFCS 2002). $\mathbf{CaucalTree}_n\Sigma$ and $\mathbf{CaucalGraph}_n\Sigma$.

Theorem (KNU-Caucal 2002)

For $n \geq 0$, $\mathbf{PushdownTree}_n\Sigma = \mathbf{SafeRecSchTree}_n\Sigma = \mathbf{CaucalTree}_n\Sigma$.

What is the safety constraint on recursion schemes?

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 (of application)
3. P contains an order-0 parameter.

Examples (unsafe eqns): $F : (o \to o) \to o \to o \to o$, $\underline{f} : o \to o \to o$.

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

Though syntactically "awkward", safety does has an algorithmic value:

Proposition

Substitution (and hence β-reduction) in safe λ-calculus can be safely implemented without renaming bound variables! Hence no need for fresh name.

Infinite structures generated by recursion schemes: key questions

1. **MSO decidability**: Is safety a genuine constraint for decidability? I.e. do Σ-labelled trees generated by (arbitrary) recursion schemes have decidable MSO theories?
2. **Machine characterization**: 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 (temporal-) logical theories (e.g. modal-mu calculus, MSO, FO, FO+reachability, FO(TC_1), etc.) of the graph families?

Q1. MSO model-checking problem for $\mathsf{RecSchTree}_n\Sigma$

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

Σ-labelled trees generated by order-2 recursion schemes (whether safe or not) have decidable MSO theories.

Theorem (Knapik, Niwinski, Urczyczn + Walukiewicz, ICALP 2005)

Modal mu-calculus model checking problem for homogenously-typed order-2 schemes (whether safe or not) is 2-EXPTIME complete.

Question. What about higher orders?

Yes: MSO decidability extends to all orders (O. LICS06).

Theorem (O. LICS06)

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

Two Key Steps:

$\llbracket G \rrbracket$ satisifes modal mu-calculus formula φ

$\Longleftrightarrow$ { Emerson + Jutla 1991}

APT $\mathcal{B}_\varphi$ has accepting run-tree over value tree $\llbracket G \rrbracket$

$\Longleftrightarrow$ { **I. Transference Principle**: Correspondence Theorem }

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

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

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

which is decidable, since the computation tree $\lambda(G)$ is regular, and the APT (= Alternating Parity Tree automaton) acceptance problem of regular trees is decidable.

Two other proofs (via CPDA and type theory respectively) are known.

Q2: Machine characterisation of HORS

Order-n collapsible pushdown automata (CPDA): "2PDA with links" [AdMO05]; "panic automata" [KNUW05].

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

Two new stack operations:

- $push_1\ a$: 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 up to the point as remembered by (i.e. pointed to) by the top_1-element of the 2-stack.

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.*

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

Definition (AdMO05) ***U*-words** are composed of 3 segments:

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

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

Note: Each U-word has a unique decomposition.
E.g. $\mathbf{(\,(\,)\,(\,(\,)\,(\,(\,)\,)}\,*\,*\,* \quad \in \quad U$
and for each $n \geq 0$, $\mathbf{(\,(}^n\,\mathbf{)}^n\,\mathbf{(}\;*^n\;**\,\in\,U$.

Lemma

U is recognizable by a non-deterministic 2PDA (need to guess the transition from segment A to segment B).

Surprisingly, U is also recognizable by a deterministic 2CPDA!

Recognizing U by a 2CPDA. E.g. **(( ) (( )** $* * * \in U$

Upon reading	Do
(	$push_2$; $push_1 a$
)	pop_1
first $*$	*collapse*
subsequent $*$	pop_2

	$[[\,]]$
(	$[[\,][a]]$
(	$[[\,][a][a\ a]]$
)	$[[\,][a][a]]$
(	$[[\,][a][a][a\ a]]$
(	$[[\,][a][a][a\ a][a\ a\ a]]$
)	$[[\,][a][a][a\ a][a\ a]]$ Collapse!
$*$	$[[\,][a][a]]$
$*$	$[[\,][a]]$
$*$	$[[\,]]$

Is order-n CPDA strictly more expressive than order-n PDA?

Does the *collapse* operation add anything?

Urzyczyn's language U is quite telling!

Fact

1. U is not recognized by a 1PDA - so it is *not* context free.
2. U is recognized by a non-deterministic 2PDA.
3. U is recognized by a deterministic 2CPDA.

Theorem (Parys STACS'11)

U is not recognizable by deterministic 2PDA.

Theorem (Equi-expressivity, HMOS LICS 2008)

For each $n \geq 0$, order-n collapsible PDA and order-n recursion schemes are equi-expressive for Σ-labelled trees.

Proof idea

- From recursion scheme G to CPDA $\mathcal{A}_G$: Use game semantics.
  Code traversals as n-stacks.
  Invariant: The top 1-stack is the P-view of the encoded traversal.
- From CPDA $\mathcal{A}$ to recursion scheme $G_{\mathcal{A}}$:
  Code configurations c as Σ-term M_c, so that $c \to c'$ implies M_c rewrites to $M_{c'}$.

CPDA are a machine characterization of simply-typed lambda calculus with recursions.

Q3: Does safety constrain expressivity as generators of:

Word languages? Conjecture: Yes, in general; but note

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

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

Trees? Conjecture: Yes, and proved(?)

Pawel Parys. "Collapse Operation Cannot Be Simulated Even By Using Higher Levels", preprint, 2011.

Graphs? Yes.

Theorem (Hague, Murawski, O. +Serre 2007)

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

Conclusion

Higher-order recursion schemes (HORS) are a robust and highly expressive language for infinite trees (and other structures).

They have rich algorithmic properties.

Recent progress in the theory has been made possible by semantic methods (e.g. game semantics and type theory), enabling the extraction of new (but necessarily highly complex) algorithms.

HORS/CPDA can serve as a logical foundation for the verification of higher-order computation.