Higher-Order Model Checking
II: Recursion Schemes and their Algorithmics

Luke Ong

University of Oxford
http://www.cs.ox.ac.uk/people/luke.ong/personal/
http://mjolnir.cs.ox.ac.uk

Estonia Winter School in Computer Science, 3-8 Mar 2013
A challenge problem in higher-order verification

**Example**: Consider $\llbracket G \rrbracket$ on the right

- $\varphi_1 = \text{“Infinitely many } f\text{-nodes are reachable”}$.
- $\varphi_2 = \text{“Only finitely many } g\text{-nodes are reachable”}$.

Every node on the tree satisfies $\varphi_1 \lor \varphi_2$.

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

**Is the “MSO Model-Checking Problem for $\text{RecSchTree}_n$” decidable?**

- INSTANCE: An order-$n$ recursion scheme $G$, and an MSO formula $\varphi$
- QUESTION: Does the $\Sigma$-labelled tree $\llbracket G \rrbracket$ satisfy $\varphi$?
Why study monadic second-order (MSO) logic?

Because it is the gold standard of logics for describing correctness properties.

- **MSO is very expressive.**
  Over graphs, MSO is more expressive than the modal mu-calculus, into which all standard temporal logics (e.g. LTL, CTL, CTL*, etc.) can embed.

- **It is hard to extend MSO meaningfully without sacrificing decidability where it holds.**
Fix a vocabulary. Three types of predicate symbols:

1. **Parent-child relationship between nodes**: \( d_i(x, y) \) \( \equiv \) “\( y \) is \( i \)-child of \( x \)”

2. **Node labelling**: \( p_f(x) \) \( \equiv \) “\( x \) has label \( f \)” where \( f \) is a \( \Sigma \)-symbol

3. **Set-membership**: \( x \in X \)

First-order variables: \( x, y, z, \text{ etc.} \) (ranging over nodes)

Second-order variables: \( X, Y, Z, \text{ etc.} \) (ranging over sets of nodes)

**MSO formulas** are generated from three kinds of atomic formulas:

\[
\begin{align*}
&d_i(x, y), \quad p_f(x), \quad x \in X \\
\end{align*}
\]

and closed under boolean connectives, first-order quantification (\( \forall x. -, \exists x. - \)) and second-order quantifications: (\( \forall X. -, \exists X. - \)).

A \( \Sigma \)-labelled tree \( t : \text{dom}(t) \rightarrow \Sigma \) is represented as a structure

\[
\langle \text{dom}(t), \langle d_i : 1 \leq i \leq m \rangle, \langle p_f : f \in \Sigma \rangle \rangle
\]
Examples of MSO-definable properties

Our version of MSOL is parsimonious. Several useful predicates are definable:

1. **Set inclusion (and hence equality):** $X \subseteq Y \equiv \forall x : x \in X \rightarrow x \in Y$.

2. **“Is-an-ancestor-of” or prefix ordering $x \leq y$ (and hence $x = y$):**

   $$\text{PrefCl}(X) \equiv \forall x, y : y \in X \land \bigvee_{i=1}^m d_i(x, y) \rightarrow x \in X$$

   $$x \leq y \equiv \forall X : \text{PrefCl}(X) \land y \in X \rightarrow x \in X$$

3. **Reachability property:** “$X$ is a path”

   $$\text{Path}(X) \equiv \forall x, y \in X : x \leq y \lor y \leq x \land$$

   $$\forall x, y, z : x \in X \land z \in X \land x \leq y \leq z \rightarrow y \in X$$

   $$\text{MaxPath}(X) \equiv \text{Path}(X) \land$$

   $$\forall Y : \text{Path}(Y) \land X \subseteq Y \rightarrow Y \subseteq X.$$
Example: “A tree has infinitely many $f$-labelled nodes”

A set of nodes is a cut if (i) no two nodes in it are $\leq$-compatible, and (ii) it has a non-empty intersection with every maximal path.

$$\text{Cut}(X) \equiv \forall x, y \in X : \neg(x \leq y \lor y \leq x) \land \forall Z : (\text{MaxPath}(Z) \rightarrow \exists z \in Z : z \in X)$$

**Lemma.** A set $X$ of nodes in a finitely-branching tree is finite iff there is a cut $C$ such that every $X$-node is a prefix of some $C$-node.

$$\text{Finite}(X) \equiv \exists Y : (\text{Cut}(Y) \land \forall x \in X : \exists y \in Y : x \leq y)$$

Hence “there are finitely many nodes labelled by $f$” is expressible in MSOL by

$$\exists X : (\text{Finite}(X) \land \forall x : p_f(x) \rightarrow x \in X)$$

**But** “MSOL cannot count”: E.g. “$X$ has twice as many elements as $Y$” is not expressible in MSO.
Recapitulation

- Two families of generators: HORS and HOPDA

Today’s lecture

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

- **Rabin 1969**: Infinite binary trees and regular trees. “Mother of all decidability results in algorithmic verification.”
- **Muller and Schupp 1985**: Configuration graphs of PDA.
- **Knapik, Niwiński and Urzyczyn (TLCA 2001, FOSSACS 2002)**:
  - $\text{PushdownTree}_n\Sigma = \text{Trees generated by order-}n\text{ pushdown automata.}$
  - $\text{SafeRecSchTree}_n\Sigma = \text{Trees generated by order-}n\text{ safe rec. schemes.}$
- **Subsuming all the above**: Caucal (MFCS 2002). $\text{CaucalTree}_n\Sigma$ and $\text{CaucalGraph}_n\Sigma$.

**Theorem (KNU-Caucal 2002)**

For $n \geq 0$, $\text{PushdownTree}_n\Sigma = \text{SafeRecSchTree}_n\Sigma = \text{CaucalTree}_n\Sigma$; and they have decidable MSO theories.
What is the safety constraint on recursion schemes?

Safety is a set of constraints on 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 rule).**

\[
F : (o \rightarrow o) \rightarrow o \rightarrow o \rightarrow o, \quad f : o^2 \rightarrow o, \quad x, y : o.
\]

\[
F \varphi x y = f (F (F \varphi y) y (\varphi x)) a
\]

The subterm $F \varphi y$ has order 1, but the free variable $y$ has order 0.
What is the point of safety?

Safety does have an important algorithmic advantage!

Theorem (KNU 02, Blum + O. TLCA 07, LMCS 09)

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

Theorem

1. (Schwichtenberg 76) The numeric functions representable by simply-typed $\lambda$-terms are multivariate polynomials with conditional.

2. (Blum + O. LMCS 09) The numeric functions representable by simply-typed safe $\lambda$-terms are the multivariate polynomials.

(See (Blum + O. LMCS 09) for a study on the safe lambda calculus.)
Infinite structures generated by recursion schemes: key 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 characterisation**: Find a hierarchy of automata that characterise 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 $\text{RecSchTree}_n \Sigma$ have decidable MSO theories?

Some progress:

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

$\Sigma$-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.

What about higher orders?

Yes: MSO decidability extends to all orders (O. LICS06).
Q1. Do trees in $\text{RecSchTree}_{n,\Sigma}$ have decidable MSO theories? Yes

**Theorem (O. LICS 2006)**

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

**Proof Idea.** Two key ingredients:

Generated tree $\llbracket G \rrbracket$ satisfies mu-calculus formula $\varphi$

$\iff \{ \text{Emerson + Jutla 1991}\}$

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

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

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

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

APT $\mathcal{C}_\varphi$ has an accepting run-tree over computation tree $\lambda(G)$ which is decidable because $\lambda(G)$ is regular.
Transference principle, based on a theory of traversals

\[ G : \begin{cases} 
S &= FH \\
F \varphi &= \varphi(F \varphi) \\
Hz &= f zz 
\end{cases} \quad \Rightarrow \quad \overline{G} : \begin{cases} 
S &= \lambda. F (\lambda x. H \lambda x) \\
F &= \lambda \varphi. \varphi(\lambda. F (\lambda y. \varphi(\lambda y)))) \\
H &= \lambda z. f(\lambda z)(\lambda z) 
\end{cases} \]

\[ \boxed{G} \]

\[ \lambda(G) \]
Idea: $\beta$-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 $\llbracket G \rrbracket$).

**Theorem (Path-traversal correspondence)**

Let $G$ be an order-$n$ recursion scheme.

(i) There is a 1-1 correspondence between maximal paths $p$ in ($\Sigma$-labelled) generated tree $\llbracket G \rrbracket$ and maximal traversals $t_p$ over computation tree $\lambda(G)$.

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

Proof is by game semantics.

**Explanation (for game semanticists):**

- Term-tree $\llbracket G \rrbracket$ is (a representation of) the game semantics of $G$.
- Paths in $\llbracket G \rrbracket$ correspond to plays in the strategy-denotation.
- Traversals $t_p$ over computation tree $\lambda(G)$ are just (representations of) the uncoverings of the plays ($=\text{path}$) $p$ in the game semantics of $G$. 
Four different proofs of the MSO decidability result

1. Game semantics and traversals (O. LICS06)
   - variable profiles. E.g. a profile of \((o \rightarrow o) \rightarrow o\) is \(\{\{q\}, q\}, \{q, q'\}, q'\}\), \(q\)

2. Collapsible pushdown automata (Hague, Murawski, O. & Serre LICS08)
   - equi-expressivity theorem + rank aware automata

3. Type-theoretic characterisation of APT (Kobayashi & O. LICS09)
   - intersection types. E.g. \((q \rightarrow q) \land (q \land q' \rightarrow q') \rightarrow q\)

4. Krivine machine (Salvati & Walukiewicz ICALP11)
   - residuals

A common pattern

1. Decision problem equivalent to solving an infinite parity game.
2. Simulate the infinite parity game by a finite parity game.
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.

Note that 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, $push_j^1 a$, with $1 \leq j \leq n - 1$:

$push_j^1 a$: 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 $\{ (, ), * \}$

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

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

- 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. $((()))(())(())\ast\ast\ast$ is a $U$-word
2. For each $n \geq 0$, we have $(n^n)(\ast^n\ast\ast$ is a $U$-word.
   Hence by “uvwxy Lemma”, $U$ is not context-free.
Recognising $U$ by a (det.) 2CPDA. E.g. $((())((()))*** \in U$
(Ignoring control states for simplicity)

<table>
<thead>
<tr>
<th>Upon reading</th>
<th>Do</th>
</tr>
</thead>
<tbody>
<tr>
<td>( )</td>
<td>$push_2 ; push_1 a$</td>
</tr>
<tr>
<td>first $*$</td>
<td>$pop_1$</td>
</tr>
<tr>
<td>subsequent $*$</td>
<td>$collapse$</td>
</tr>
<tr>
<td></td>
<td>$pop_2$</td>
</tr>
</tbody>
</table>

What does the depth of the top 1-stack mean?
Is order-$n$ CPDA strictly more expressive than order-$n$ PDA?

Does the \textit{collapse} operation add any expressive power?

Lemma (AdMO FoSSaCS05): Urzyczyn’s language $U$ is quite telling!

1. $U$ is not recognised by a 1PDA.
2. $U$ is recognised by a non-deterministic 2PDA.
3. $U$ is recognised by a deterministic 2CPDA.

Question

\textit{Is $U$ recognisable by a deterministic 2PDA? or by $n$PDA for any $n$?}

If true, there is an associated tree that is generated by an order-2 recursion scheme, but not by any order-2 \textit{safe} recursion scheme.
Theorem (Equi-expressivity [Hague, Murawski, O. & Serre LICS08])

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

Proof idea

- From recursion scheme to CPDA: Use game semantics. Code traversals as \( n \)-stacks. **Invariant**: The top 1-stack is the P-view of the encoded traversal.
- From CPDA to recursion scheme: Code configuration \( c \) as \( \Sigma \)-term \( M_c \), so that \( c \rightarrow c' \) implies \( M_c \) rewrites to \( M_{c'} \).

CPDA are a machine characterization of simply-typed lambda calculus with recursions.
A direct proof (without game semantics) [Carayol & Serre LICS12].
**Q3: Is safety a genuine constraint on expressivity?**

**Question (Safety, KNW FoSSaCS02)**

Are there inherently unsafe word languages / trees / graphs?

**Word languages? Yes**

**Theorem (Parys STACS11, LICS12)**

There is a language (similar to $U$) recognised by a deterministic 2CPDA but not by any deterministic $n$PDA for all $n \geq 0$.

Proof uses a powerful pumping lemma for HOPDA.

(Another pumping lemma for nCPDA is used to prove a hierarchy theorem for collapsible graphs and trees [Kartzow & Parys, MFCS12])

**Trees? Yes**

**Corollary (Parys STACS11, LICS12)**

There is a tree generated by an order-2 recursion scheme but not by any safe HORS.
Graphs? Yes.

**Theorem (Hague, Murawski, O and Serre LICS08)**

1. Solvability of parity games over order-$n$ CPDA graphs is $n$-EXPTIME complete.
2. There is an 2CPDA configuration graph with an undecidable MSO theory.

**Corollary**

There is a 2CPDA whose configuration graph (semi-infinite grid) is not that of any $n$PDA, for any $n$. 
A safety question for non-determinacy

**Question (Safety non-determinacy)**

*Is there a word language recognised by a order-n CPDA which is not recognisable by any non-deterministic HOPDA?*

For order 2, the answer is no.

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

*For every order-2 recursion scheme, there is a safe non-deterministic order-2 recursion scheme that generates the same word language.*