Higher-Order Model Checking

II: Recursion Schemes and their Algorithmics

Luke Ong

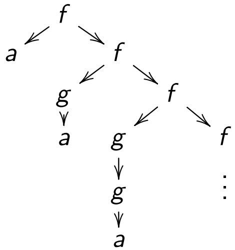
University of Oxford
http://www.cs.ox.ac.uk/people/luke.ong/personal/
http://mjolnir.cs.ox.ac.uk

Estonia Winter School in Computer Science, 3-8 Mar 2013

A challenge problem in higher-order verification

Example: Consider $[\![G]\!]$ on the right

- $\varphi_1 =$ "Infinitely many f-nodes are reachable".
- $\varphi_2 =$ "Only finitely many g-nodes are reachable".



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

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

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

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

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.

Monadic Second-Order Logic (for Σ-labelled trees)

Fix a vocabulary. Three types of predicate symbols:

1. Parent-child relationship between nodes: $\mathbf{d}_i(x, y) \equiv$ "y is i-child of x"
2. Node labelling: $\mathbf{p}_f(x) \equiv$ "x has label f" where f is a Σ-symbol
3. Set-membership: $x \in X$

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 generated from three kinds of atomic formulas:

$$\mathbf{d}_i(x, y), \quad \mathbf{p}_f(x), \quad x \in X$$

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

A Σ-labelled tree $t : dom(t) \longrightarrow \Sigma$ is represented as a structure

$$\langle\, dom(t),\ \langle\, \mathbf{d}_i : 1 \leq i \leq m \,\rangle,\ \langle\, \mathbf{p}_f : f \in \Sigma \,\rangle\,\rangle$$

Examples of MSO-definable properties

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

1. Set inclusion (and hence equality): $X \subseteq Y \;\equiv\; \forall x : x \in X \to x \in Y$.
2. "Is-an-ancestor-of" or prefix ordering $x \leq y$ (and hence $x = y$):

$$\begin{array}{rcl} \mathrm{PrefCl}(X) & \equiv & \forall x, y : y \in X \wedge \bigvee_{i=1}^{m} \mathbf{d}_i(x, y) \;\to\; x \in X \\ x \leq y & \equiv & \forall X : \mathrm{PrefCl}(X) \wedge y \in X \;\to\; x \in X \end{array}$$

3. Reachability property: "X is a path"

$$\begin{array}{rcl} \mathrm{Path}(X) & \equiv & \forall x, y \in X : x \leq y \vee y \leq x \qquad \wedge \\ & & \forall x, y, z : x \in X \wedge z \in X \wedge x \leq y \leq z \;\to\; y \in X \end{array}$$

$$\begin{array}{rcl} \mathrm{MaxPath}(X) & \equiv & \mathrm{Path}(X) \quad \wedge \\ & & \forall Y : \mathrm{Path}(Y) \wedge X \subseteq Y \;\to\; Y \subseteq X. \end{array}$$

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.

$$\begin{array}{rcl} \mathrm{Cut}(X) & \equiv & \forall x, y \in X : \neg(x \leq y \ \vee \ y \leq x) \quad \wedge \\ & & \forall Z : (\mathrm{MaxPath}(Z) \ \rightarrow \ \exists z \in Z : z \in X) \end{array}$$

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.

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

Hence "there are finitely many nodes labelled by f" is expressible in MSOL by

$$\exists X : (\mathrm{Finite}(X) \ \wedge \ \forall x : \mathbf{p}_f(x) \rightarrow x \in X)$$

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.
- Caucal 1996 Prefix-recognisable 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:
  Caucal (MFCS 2002). $\mathbf{CaucalTree}_n\Sigma$ and $\mathbf{CaucalGraph}_n\Sigma$.

Theorem (KNU-Caucal 2002)

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

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

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

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 β-red.) in safe λ-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 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 + 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 $\mathrm{RecSchTree}_n\Sigma$ have decidable MSO theories?

Some progress:

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.

What about higher orders?

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

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.*

Proof Idea. Two key ingredients:

Generated tree $[\![G]\!]$ satisfies mu-calculus formula φ

$\Longleftrightarrow$ { Emerson + Jutla 1991}

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

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

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

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

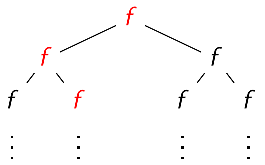
APT $\mathcal{C}_\varphi$ has an accepting run-tree over computation tree $\lambda(G)$

which is decidable because $\lambda(G)$ is regular.

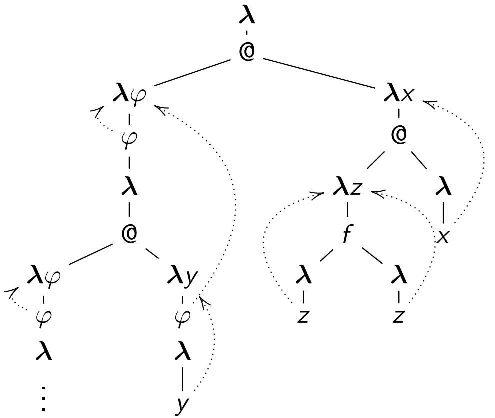
Transference principle, based on a theory of traversals

$$G : \left\{ \begin{array}{rcl} S & = & F\,H \\ F\,\varphi & = & \varphi\,(F\,\varphi) \\ H\,z & = & f z z \end{array} \right. \qquad \mapsto \qquad \overline{G} : \left\{ \begin{array}{rcl} S & = & \boldsymbol{\lambda}.@\,F\,(\boldsymbol{\lambda}x.@\,H\,\boldsymbol{\lambda}.x) \\ F & = & \boldsymbol{\lambda}\varphi.\varphi(\boldsymbol{\lambda}.@\,F\,(\boldsymbol{\lambda}y.\varphi(\boldsymbol{\lambda}.y))) \\ H & = & \boldsymbol{\lambda}z.f(\boldsymbol{\lambda}.z)(\boldsymbol{\lambda}.z) \end{array} \right.$$

$[\![G]\!]$



$\lambda(G)$



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.

Four different proofs of the MSO decidability result

1. Game semantics and traversals (O. LICS06)
   - variable profiles. E.g. a profile of $(o \to o) \to o$ 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. $(q \to q) \wedge (q \wedge q' \to q') \to q$
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.
3. Key ingredient of the game: variable profiles / automaton control-states / intersection types / residuals.

Q2: Machine characterisation: 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.

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, $push_1^j\ a$, 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 (Aehlig, de Miranda + O. FoSSaCS 05) A ***U*-word** has 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 entire word.
- Segment B is a well-bracketed word.
- Segment C has length equal to the number of **(** in segment A.

Examples

1. $\mathbf{(\,(\,)\,(\,(\,)\,(\,(\,)\,)}\; * \; * \; *$ is a U-word
2. For each $n \geq 0$, we have $\mathbf{(\,(}^{n}\,\mathbf{)}^{n}\,\mathbf{(}\; *^{n}\; **$ 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)

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]]
$*$	[[ ]]

What does the depth of the top 1-stack mean?

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

Does the *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

Is U recognisable by a deterministic 2PDA? or by nPDA 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 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 Σ-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 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.

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 nPDA 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 nPDA, for any n.

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.