Automata, Logic and Games for the Lambda Calculus
Recent Developments in Higher-Order Model Checking

Luke Ong
University of Oxford

ICLA 2017, Kanpur IIT
Model checking is an approach to verification that promises accurate analysis with push-button automation.

2007 ACM Turing Award (Clarke, Emerson and Sifakis) “for their rôle in developing model checking into a highly effective verification technology, widely adopted in hardware and software industries”.

What is Model Checking?

Problem: Given a system $Sys$ (e.g. an OS) and a correctness property $Spec$ (e.g. deadlock freedom), does $Sys$ satisfy $Spec$?

The Model Checking Approach:

1. Find an abstract (e.g. finite-state) model $M$ of the system $Sys$.
2. Describe the property $Spec$ as a formula $\varphi$ of a decidable logic.
3. Exhaustively check if $\varphi$ is violated by $M$.

In recent years, there has been extensive research in the model checking of higher-order computation. Haskell, F#, C++11, Java8, JavaScript, Scala, Perl5, Python, etc.
Higher-Order Model Checking is the model checking of infinite structures, such as trees, that are defined by recursion schemes (equivalently $\lambda^Y$-calculus).

1. Higher-Order Recursion Schemes (HORS) as Grammars of Infinite Trees, and the MSO Model Checking Problem

2. Decidability, Expressivity and Automata Characerisations

3. Compositional Higher-Order Model Checking, and Model Checking of Higher-Type Böhm Trees

4. Some Open Problems and Conclusions
Types \[ A ::= o \mid (A \to B) \]

\( o \) is the type of ranked trees.

Order of a type: measures “nestedness” on LHS of \( \to \).

\[
\begin{align*}
\text{order}(o) & := 0 \\
\text{order}(A \to B) & := \max(\text{order}(A) + 1, \text{order}(B))
\end{align*}
\]

Examples

1. \( \mathbb{N} \to \mathbb{N} \) and \( \mathbb{N} \to (\mathbb{N} \to \mathbb{N}) \) both have order 1;

2. \( (\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \) has order 2.

Notation \( e : A \) means “expression \( e \) has type \( A \)”. 
Higher-Order Recursion Schemes (HORS)

(Park 68, de Roever 72, Nivat 72, Nivat-Courcelle 78, Damm 82, . . .)

HORS are grammars for trees (and tree languages).

Fix a ranked alphabet $\Sigma$ (i.e. a set of tree constructors).

Order-$n$ recursion schemes over $\Sigma = $ programs of the order-$n$ fragment of simply-typed $\lambda$-calculus + recursion + order-1 symbols from $\Sigma$.

Concretely, a HORS is a finite set of simply-typed functions, defined by mutual recursion over $\Sigma$, with a distinguished start function $S : o$.

Example (order 1). $\Sigma = \{ f : 2, \ g : 1, \ a : 0 \}$.

$$G : \left\{ \begin{array}{c} S \rightarrow Fa \\ Fx \rightarrow fx(F(gx)) \end{array} \right\}$$
Example (order 1)

\[ \Sigma = \{ f : 2, \ g : 1, \ a : 0 \}. \]

\[
G : \left\{ \begin{align*}
S & \rightarrow FA \\
Fx & \rightarrow fx(F(gx))
\end{align*} \right.
\]

\[ S \rightarrow FA \]
\[ \rightarrow fa(F(ga)) \]
\[ \rightarrow fa(f(ga)(F(g(ga)))) \]
\[ \rightarrow \ldots \]

The tree generated, \( \llbracket G \rrbracket \), is the abstract syntax tree underlying \( fa(f(ga)(f(g(ga))(\cdots))) \).

Many ways of defining \( \llbracket G \rrbracket \) (as least fixpoint, least solution, initial algebra semantics, etc.).
E.g. Consider properties of nodes of $[G]$:

- $\varphi = \text{"Infinitely many } f\text{-nodes are reachable"}.$
- $\psi = \text{"Only finitely many } G\text{-nodes are reachable"}.$

Every node of the tree satisfies $\varphi \lor \psi$.

Monadic second-order logic (MSO) is an expressive logic that can describe properties such as $\varphi \lor \psi$.

**MSO Model-Checking Problem for Trees generated by HORS**

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

**QUESTION**: Is the above problem decidable?
Some Infinite Structures with Decidable MSO Theories

- **Rabin 1969**: Regular trees.
  “Mother of all decidability results in Verification”

- **Muller and Schupp 1985**: Configuration graphs of pushdown automata.

- **Knapik, Niwiński and Urzyczyn (TLCA01, FoSSaCS02)**:
  \[ \text{PushdownTree}_{n \Sigma} = \text{Trees generated by order-}n \text{ pushdown automata.} \]
  \[ \text{SafeRecSchTree}_{n \Sigma} = \text{Trees generated by order-}n \text{ safe rec. schemes.} \]

- **Subsuming all the above**:
  Caucal (MFCS02). \[ \text{CaucalTree}_{n \Sigma} \text{ and CaucalGraph}_{n \Sigma}. \]

**Theorem (KNU-Caucal 2002)**

For \( n \geq 0 \), \[ \text{PushdownTree}_{n \Sigma} = \text{SafeRecSchTree}_{n \Sigma} = \text{CaucalTree}_{n \Sigma} \]
have decidable MSO theories.
What is the Safety Constraint on Recursion Schemes / \( \lambda \)-Calculus?

There is a “weaker” hierarchy of finite types: safe types (Damm 82)

\[
d_0 := \{ \text{ranked trees} \} \\
d_{i+1} := \bigcup_{k \geq 0} [d_i \times \cdots \times d_i \rightarrow d_i]
\]

Parameters of safe types have non-increasing order. E.g.

\[
\lambda F. \lambda f. \lambda x. f \ x : d_2 \rightarrow (d_1 \rightarrow (d_0 \rightarrow d_0)) \subseteq d_3 \text{ safe}
\]

\[
\lambda F. \lambda x. \lambda f. f \ x : d_2 \rightarrow (d_0 \rightarrow (d_1 \rightarrow d_0)) \nsubseteq d_3 \text{ unsafe}
\]

Safe \( \lambda \)-Terms (KNO01; Blum & O. LMCS 2009)

1. **Safety** – a syntactic constraint: no order-\( k \) subterm can contain free variables of order < \( k \).
2. Substitution (hence \( \beta \)-reduction) in “safe \( \lambda \)-calculus” can be implemented without renaming bound variables: variable capture is guaranteed never to happen! Hence no need for fresh names.

Knapik et al. exploits this algorithmic advantage of safety in MSO-decidability
## A Tale of Two Hierarchies of Finite Types

Syntactically, **Safe Types** $\subset$ **Simple Types**

<table>
<thead>
<tr>
<th><strong>Safe Types</strong> (Damm TCS 82)</th>
<th><strong>Simple Types</strong> (Church JSL 40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{i+1} := \bigcup_{k \geq 0} [d_i \times \cdots \times d_i \to d_i]$</td>
<td>$\kappa := 0 \mid \kappa \to \kappa'$</td>
</tr>
<tr>
<td>Safey: awkward constraint but yields elegant and strong algorithmic results</td>
<td>Natural, clean and standard, in semantics and in programming</td>
</tr>
</tbody>
</table>

- MSO model checking of **safe** recursion scheme is decidable (KNU 02)
- Order-$n$ **safe** RS $\equiv$ order-$n$ pushdown automata (Damm 82, KNU 02)
- Hierarchy is strict (Damm 82)
- Word languages are context-sensitive (Inaba & Maneth 08)

? (Marked sections denote missing data or placeholders.)
A Tale of Two Hierarchies of Finite Types

Syntactically, **Safe Types** ⊂ **Simple Types**

<table>
<thead>
<tr>
<th><strong>Safe Types</strong> (Damm TCS 82)</th>
<th><strong>Simple Types</strong> (Church JSL 40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{i+1} := \bigcup_{k \geq 0} \left[ d_i \times \cdots \times d_i \to d_i \right]$</td>
<td>$\kappa := o \mid \kappa \to \kappa'$</td>
</tr>
<tr>
<td>Safey: awkward constraint but yields elegant and strong algorithmic result</td>
<td>Natural and standard, semantically and in programming</td>
</tr>
</tbody>
</table>

| MSO model checking of safe recursion scheme is decidable (KNU 02) | Q1: Is MSO model checking of arbitrary recursion scheme decidable? |
| Order-$n$ safe RS $\equiv$ order-$n$ pushdown automata (Damm 82, KNU 02) | Q2: Automata characterization: Order-$n$ recursion schemes $\equiv$ ? |
| Q3: Expressivity: Are there inherently unsafe languages / trees / graphs? |

Hierarchy is strict (Damm 82)  
Word languages are context-sensitive (Inaba & Maneth 08)
For $n \geq 0$, the alternating parity tree automaton (APT) model-checking problem for order-$n$ recursion schemes is $n$-EXPTIME complete. Hence the MSO model checking problem is decidable.
Recall: A Standard Automata-Logic-Games Correspondence

On trees: $L_\mu \equiv \text{MSOL}$

- Mu-Calculus ($L_\mu$) and Alternating Parity Automata (APT) are effectively equi-expressive for tree languages [Emerson & Jutla, FoCS 91]

- $L_\mu$ (Mu-Calculus) Model Checking Problem and Parity are inter-reducible [Streett and Emerson, Info & Comp 1989]
Q1. MSO Model-Checking Problem for Trees generated by HORS

Theorem (O. LICS06)

For $n \geq 0$, the alternating parity tree automaton (APT) model-checking problem for order-$n$ recursion schemes is $n$-EXPTIME complete. Hence the MSO model checking problem is decidable.

[Rabin, Emerson & Jutla, etc.: APT equi-expressive with MSO over trees]

Proof Idea. By game semantics. Two key ingredients:

1. Traversal-Path Correspondence

   APT $B$ has accepting run-tree over generated tree $[G]$

   $\iff$

   APT $B$ has accepting traversal-tree over tree-unfolding of $G$, $\text{unfold}(G)$

2. Simulation of traversals by paths

   transformed APT $\hat{B}$ has accepting run-tree over $\text{unfold}(G)$

   which is decidable because $\text{unfold}(G)$ is a regular tree.
Various Proofs of the MSO Decidability Result

1. Game semantics and traversals (O. LICS06)
   - Variable profiles
2. Collapsible pushdown automata (Hague, Murawski, O. & Serre LICS08)
   - Priority-aware automata & equi-expressivity theorem
3. Type-theoretic characterisation of APT (Kobayashi & O. LICS09)
   - Intersection types
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 finite games: respectively variable profiles / automaton control-states / intersection types / residuals.
### Summary: A Tale of Two Hierarchies of Finite Types

Syntactically, **Safe Types** $\subset$ **Simple Types**

<table>
<thead>
<tr>
<th>Safe Types (Damm TCS 82)</th>
<th>Types (Church JSL 40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{i+1} := \bigcup_{k \geq 1} [d_i \times \cdots \times d_i \rightarrow d_i]$</td>
<td>$\kappa := o \mid \kappa \rightarrow \kappa'$</td>
</tr>
</tbody>
</table>

**MSO model checking of safe RS is decidable** [KNU FoSSaCS02]

**Order-$n$ safe RS $\equiv$ order-$n$ PDA** [KNU TLCA01]

**Q1**: MSO model checking of recursion schemes is **decidable** [O. LICS06]

**Q2**: Order-$n$ RS $\equiv$ order-$n$ CPDA [Hague, Murawski, O. & Serre LICS08]

**Q3a**: Inherently unsafe trees exist. [Parys LICS12]

**Q3b**: Inherently unsafe graphs exist. [Hague, Murawski, O. & Serre LICS08]

**Hierarchy is strict** [Damm TCS82]

**Hierarchy is strict** [Kartzow & Parys STACS12]

**Word languages are context-sensitive** [Inaba & Maneth FSTTCS08]

**Order-3 unsafe languages are context-sensitive** (Kobayashi et al. FoSSaCS14)
Like standard model checking, higher-order model checking is a whole program analysis. This can seem surprising: higher order is supposed to aid modular structuring of programs!

Hitherto HOMC is about computation trees of ground-type functional programs. **Aim**: Model check the computation trees of higher-type functional programs (= Böhm trees i.e. trees with binders).

**Seek**: a denotational model to support compositional model checking, which should be strategy aware (i.e. modelling Böhm trees, and witnesses of correctness properties of Böhm trees), and organisable into a cartesian closed category of parity games.

Unfortunately the elegant theorems of “Rabin’s Heaven” fail in the world of Böhm trees.
Example Böhm Tree ("Semi-infinite Grid"): \( u_\infty \)

\( u_\infty \) uses infinitely many variable names, and each variable occurs infinitely often.

\( u_\infty \) has an undecidable MSO theory (Salvati; Clairambault & Murawski FSTTCS13).

\( u_\infty \) is \( \lambda Y \)-definable of order 4:

\( u_\infty = BT(M) \) where

\[
\Gamma \vdash Y (\lambda f. \lambda y^o. \lambda x^{o^{\rightarrow o}}. b (x y) (f (x y))) a : (o \rightarrow o) \rightarrow o
\]

with \( \Gamma = a : o, \ b : o \rightarrow ((o \rightarrow o) \rightarrow o) \rightarrow o \).

\( \lambda X_1 \\
\lambda X_2 \\
\lambda X_3 \\
\lambda X_4 \\
X_1 \\
X_2 \\
X_3 \\
X_4 \\
a \\
b \\
a \\
b \\
a \\
b \\
a \\
b \\
M \\
\Gamma \\
\vdash \\
Y \\
(\lambda f. \lambda y^o. \lambda x^{o^{\rightarrow o}}. b (x y) (f (x y))) a : (o \rightarrow o) \rightarrow o \\
\]

\( \Gamma \vdash Y (\lambda f. \lambda y^o. \lambda x^{o^{\rightarrow o}}. b (x y) (f (x y))) a : (o \rightarrow o) \rightarrow o \\
\]

with \( \Gamma = a : o, \ b : o \rightarrow ((o \rightarrow o) \rightarrow o) \rightarrow o \).
Take property $\Phi :=$

“*There are only finitely many occurrences of bound variables in each branch.*”

Questions:

1. Is there an expressive “logic” that can describe properties such as $\Phi$?
2. Is such a logic decidable for Böhm trees?

Approach:

1. Fix a semantics (satisfaction relation), $\Gamma \models U : \tau$, for Böhm trees $U$ and formulas $\tau$
2. Develop decidable proof system for $\Gamma \vdash M : \tau$, and aim for “completeness”:

$$\Gamma \vdash M : \tau \iff \Gamma \models \text{BT}(M) : \tau$$
Type-Checking Game

\[ \models U : \tau \]

means “Verifier has a winning strategy in the game that checks Böhm tree \( U \) has type \( \tau \)”

Types \( \tau \) (Kobayashi & O. LICS09) are parameterised by base types \( Q \), and a winning condition \((E, F, \Omega)\), which is an algebraic abstraction of the \( \omega \)-regular winning conditions:

- **Prime types**: \( \tau ::= q \mid \alpha \rightarrow \tau \)
- **Intersection types**: \( \alpha ::= \bigwedge_{i \in I}(\tau_i, e_i) \)

where \( q \in Q \) (base types), \( e_i \in E \) (effect set), and \( I \) is a finite indexing set.

E.g. Alternating parity tree automaton yields a winning condition \((E, F, \Omega)\), whereby \( Q \) are automaton states, and \( E \) are priorities.
Intuition

Regard automaton states as the base types i.e. types of trees

- \( q \) is the type of trees accepted by the automaton from state \( q \)
- \( q_1 \land q_2 \) is the type of trees accepted from both \( q_1 \) and \( q_2 \)
- \( \tau \rightarrow q \) is the type of functions that take a tree of type \( \tau \) and return a tree of type \( q \)

A tree function described by \((q_1, m_1) \land (q_2, m_2) \rightarrow q\).

(The above is a tree context of a run-tree, not the generated tree.)
Inference System for Type-Checking Game: $\Gamma \vdash M : \tau$

Read $\Gamma \vdash M : \tau$ as “In $\Gamma$, $\text{BT}(M)$ has type $\tau$”

(Below: 5 of 7 rules)

\[
\theta = \theta_i \quad \& \quad \epsilon \leq \sqsubseteq \epsilon_i \quad \text{for some } i \quad \text{where } \Gamma(x) = \bigwedge_{i \in I} (\theta_i, \epsilon_i)
\]

\[
\frac{\Gamma \vdash M : \tau \rightarrow \theta \quad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \theta}
\]

\[
\frac{\Gamma, x : \tau \vdash M : \theta}{\Gamma \vdash \lambda x. M : \tau \rightarrow \theta}
\]

\[
\frac{\Gamma' \leq \Gamma \quad \Gamma \vdash M : \theta \quad \theta \leq \theta'}{\Gamma' \vdash M : \theta'}
\]

\[
\frac{\Gamma \vdash \text{BT}(Y) : \theta}{\Gamma \vdash Y : \theta}
\]

**Theorem (Transfer)**

For all $\lambda Y$-terms $M$ and types $\tau$: $\Gamma \vdash M : \tau \iff \Gamma \models \text{BT}(M) : \tau$

$\Gamma \vdash M : \tau$ is decidable: syntax-directed rules reduce the problem to solving parity games, $\models \text{BT}(Y) : \tau$, which is decidable.
New Automata-Logic-Games Correspondence for Higher-type Böhm Trees

1. Type-checking Games  [This talk]
2. Alternating Dependency Tree Automata (ADTA)
3. Higher-type Mu-calculus ($L_\mu^\to$)

\[ \text{Böhm Trees: ADTA} \quad \quad \quad \quad L_\mu^\to \]

\[ \text{(Ranked) Trees: APT} \quad \quad \quad \quad L_\mu \]

\[ \text{Type-Checking Games} \]

\[ \text{Parity Games} \]
Some Open Problems

1. **Equivalence of Recursion Schemes** asks whether two given recursion schemes generate the same tree. (Recursively equivalent to Böhm Tree Equivalence of $\lambda Y$-terms.)
   Is the problem decidable?

2. **The Nondeterministic Safety Conjecture**: there is a word language recognisable by a nondeterministic $n$-CPDA, but not by any nondeterministic HOPDA.
   False for $n = 2$; open for $n \geq 3$.

3. **Are Unsafe Word Languages Context Sensitive?**.
   Problem open for 4th order or higher.

4. **Computing Downward Closures of Tree Languages of the Recursion Schemes Hierarchy**.

5. **Extensions of Higher-Order Model Checking**
Numerous Topics Not Covered

1. Design and Implementation of Practical Higher-Order Model Checking Algorithms
2. Effective Denotational Models of / for Higher-Order Model Checking
3. Many Applications
4. etc.

Conclusions

- HORS are a robust and highly expressive grammar for infinite structures. They have rich algorithmic properties.
- Recent progress in the theory have used semantic methods (game semantics and types) as well as the more standard automata-theoretic techniques.
- We have developed a compositional approach to model check higher-type Böhm trees, guided and justified by a new CCC of $\omega$-regular games.