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 φ of a decidable logic.
3. Exhaustively check if φ 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.

Outline

Higher-Order Model Checking is the model checking of infinite structures, such as trees, that are defined by recursion schemes (equivalently $\lambda\mathbf{Y}$-calculus).

1. Higher-Order Recursion Schemes (HORS) as Grammars of Infinite Trees, and the MSO Model Checking Problem
2. Decidability, Expressivity and Automata Characterisations
3. Compositional Higher-Order Model Checking, and Model Checking of Higher-Type Böhm Trees
4. Some Open Problems and Conclusions

Simple Types (Church JSL 1940)

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

o is the type of ranked trees.

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

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

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 $\quad e : A \quad$ 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 Σ (i.e. a set of tree constructors).

Order-n recursion schemes over Σ = programs of the order-n fragment of simply-typed λ-calculus + recursion + order-1 symbols from Σ.

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

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

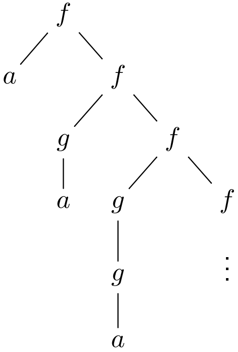
$$\mathcal{G} \ : \left\{ \begin{array}{rcl} S & \rightarrow & F\,a \\ F\,x & \rightarrow & f\,x\,(F\,(g\,x)) \end{array} \right.$$

Example (order 1)

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

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

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



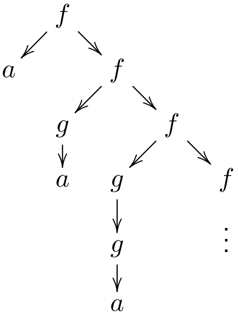
The tree generated, $[\![\, \mathcal{G}\,]\!]$, is the abstract syntax tree underlying $f\, a\, (f\, (g\, a)\, (f\, (g\, (g\, a))(\cdots)))$.

Many ways of defining $[\![\, \mathcal{G}\,]\!]$ (as least fixpoint, least solution, initial algebra semantics, etc.).

A Problem in Verification

E.g. Consider properties of nodes of $[\![\,\mathcal{G}\,]\!]$:

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



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

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

MSO Model-Checking Problem for Trees generated by HORS

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

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):
  $\textbf{PushdownTree}_n\Sigma$ = Trees generated by order-n pushdown automata.
  $\textbf{SafeRecSchTree}_n\Sigma$ = Trees generated by order-n safe rec. schemes.
- Subsuming all the above:
  Caucal (MFCS02). $\textbf{CaucalTree}_n\Sigma$ and $\textbf{CaucalGraph}_n\Sigma$.

Theorem (KNU-Caucal 2002)

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

What is the Safety Constraint on Recursion Schemes / λ-Calculus?

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

$$\mathbf{d}_0 \;:=\; \{\,\text{ranked trees}\,\} \qquad \mathbf{d}_{i+1} \;:=\; \bigcup_{k \geq 0} [\underbrace{\mathbf{d}_i \times \cdots \times \mathbf{d}_i}_{k} \to \mathbf{d}_i]$$

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

$$\begin{array}{lcl} \lambda F.\lambda f.\lambda x.f\,x & : & \mathbf{d}_2 \to (\mathbf{d}_1 \to (\mathbf{d}_0 \to \mathbf{d}_0)) \quad \subseteq \quad \mathbf{d}_3 \quad \text{safe} \\ \lambda F.\lambda x.\underline{\lambda f.f\,x} & : & \mathbf{d}_2 \to (\mathbf{d}_0 \to (\mathbf{d}_1 \to \mathbf{d}_0)) \quad \not\subseteq \quad \mathbf{d}_3 \quad \text{unsafe} \end{array}$$

Safe λ-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 β-reduction) in "safe λ-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

Safe Types (Damm TCS 82) $\mathbf{d}_{i+1} \ := \ \bigcup_{k \geq 0} [\underbrace{\mathbf{d}_i \times \cdots \times \mathbf{d}_i}_{k} \to \mathbf{d}_i]$ Safey: awkward constraint but yields elegant and strong algorithmic results	**Simple Types** (Church JSL 40) $\kappa := o \ \mid \ \kappa \to \kappa'$ Natural, clean and standard, in semantics and in programming
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)	?

A Tale of Two Hierarchies of Finite Types

Syntactically, Safe Types $\subset$ Simple Types

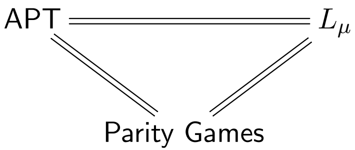
Safe Types	Simple Types
Safe Types (Damm TCS 82) $\mathbf{d}_{i+1} := \bigcup_{k \geq 0} [\underbrace{\mathbf{d}_i \times \cdots \times \mathbf{d}_i}_{k} \to \mathbf{d}_i]$ Safey: awkward constraint but yields elegant and strong algorithmic result	**Simple Types** (Church JSL 40) $\kappa := o \mid \kappa \to \kappa'$ Natural and standard, semantically and in programming
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 characteraction: 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)	?

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.

On trees: $L_\mu \equiv$ MSOL



- Mu-Calculus (L_μ) and Alternating Parity Automata (APT) are effectively equi-expressive for tree languages [Emerson & Jutla, FoCS 91]
- L_μ (Mu-Calculus) Model Checking Problem and PARITY are inter-reducible [Streett and Emerson, Info & Comp 1989]

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:

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

$\iff$ { I. Traversal-Path Correspondence}

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

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

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

which is decidable because unfold$(\mathcal{G})$ 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

Safe Types (Damm TCS 82) $\mathbf{d}_{i+1} \ := \ \bigcup_{k \geq 1} [\underbrace{\mathbf{d}_i \times \cdots \times \mathbf{d}_i}_{k} \to \mathbf{d}_i]$	**Types** (Church JSL 40) $\kappa := o \ \mid \ \kappa \to \kappa'$
MSO model checking of safe RS is decidable [KNU FoSSaCS02]	**Q1**: MSO model checking of recursion schemes is decidable [O. LICS06]
Order-n safe RS $\equiv$ order-n PDA [KNU TLCA01]	**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)

Compositional Higher-Order Model Checking? ... Several Obstacles

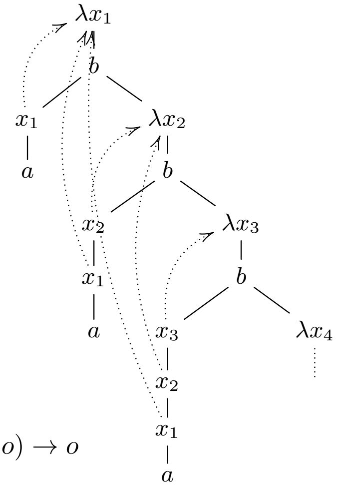
1. Like standard model checking, higher-order model checking is a whole program analysis. This can seem surprising: higher order is supposed to aid modular structuring of programs!
2. Hitherto HOMC is about computation trees of ground-type functional programs.
   Aim: Model check the computation trees of higher-type functional programs (= Böhm trees i.e. trees with binders).
3. **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.
4. Unfortunately the elegant theorems of "Rabin's Heaven" fail in the world of Böhm trees.

Example Böhm Tree ("Semi-infinite Grid"): u_∞

u_∞ uses infinitely many variable names, and each variable occurs infinitely often.

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

u_∞ is $\lambda\mathbf{Y}$-definable of order 4:



$u_\infty = \mathrm{BT}(M)$ where

$$\Gamma \vdash \underbrace{\mathbf{Y}\,(\lambda f.\lambda y^o.\lambda x^{o\to o}.b\,(x\,y)\,(f\,(x\,y)))\,a}_{M} \;:\; (o \to o) \to o$$

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

An expressive yet decidable logic for higher-type Böhm trees?

Take property $\Phi :=$
"*There are only finitely many occurrences of bound variables in each branch*."

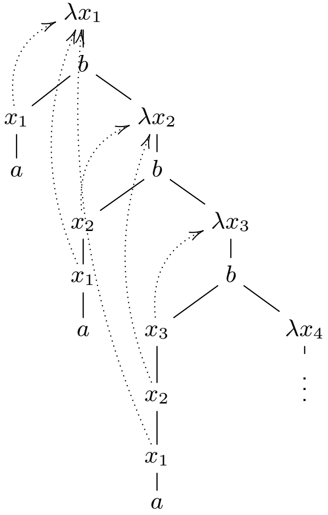
Questions:

1. Is there an expressive "logic" that can describe properties such as Φ?
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 τ
2. Develop decidable proof system for $\Gamma \vdash M : \tau$, and aim for "completeness":

$$\Gamma \vdash M : \tau \iff \Gamma \models \mathrm{BT}(M) : \tau$$



Type-Checking Game

$$\models U : \tau$$

means "Verifier has a winning strategy in the game that checks Böhm tree U has type τ"

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

$$\begin{array}{lrcl} \text{prime types} & \tau & ::= & q \mid \alpha \to \tau \\ \text{intersection types} & \alpha & ::= & \bigwedge_{i \in I} (\tau_i, e_i) \end{array}$$

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

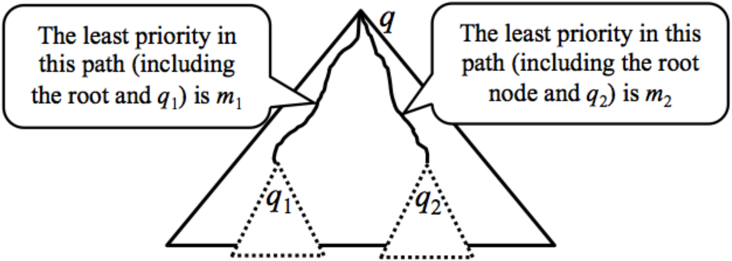
E.g. Alternating parity tree automaton yields a winning condition $(\mathbb{E}, \mathbb{F}, \Omega)$, whereby Q are automaton states, and $\mathbb{E}$ are priorities.

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
- $q_1 \wedge q_2$ is the type of trees accepted from both q_1 and q_2
- $\tau \to q$ is the type of functions that take a tree of type τ and return a tree of type q

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



(The above is a tree context of a run-tree, not the generated tree.)

Read $\Gamma \vdash M : \tau$ as "In Γ, $\mathrm{BT}(M)$ has type τ"

(Below: 5 of 7 rules)

$$\frac{\theta = \theta_i \ \ \& \ \ \epsilon \preceq_{\mathbb{E}} e_i \quad \text{for some } i \text{ where } \Gamma(x) = \bigwedge_{i \in I} (\theta_i, e_i)}{\Gamma \vdash x : \theta}$$

$$\frac{\Gamma \vdash M : \tau \to \theta \quad \Gamma \vdash N : \tau}{\Gamma \vdash M\ N : \theta} \qquad \frac{\Gamma, x : \tau \vdash M : \theta}{\Gamma \vdash \lambda x.M : \tau \to \theta}$$

$$\frac{\Gamma' \preceq \Gamma \quad \Gamma \vdash M : \theta \quad \theta \preceq \theta'}{\Gamma' \vdash M : \theta'} \qquad \frac{\models \mathrm{BT}(\mathbf{Y}) : \theta}{\Gamma \vdash \mathbf{Y} : \theta}$$

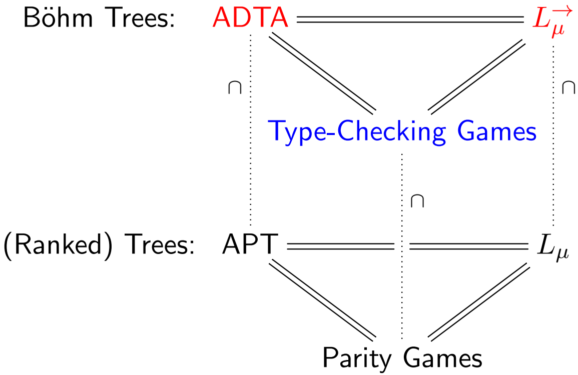
Theorem (Transfer)

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

$\Gamma \vdash M : \tau$ is decidable: syntax-directed rules reduce the problem to solving parity games, $\models \mathrm{BT}(\mathbf{Y}) : \tau$, 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 ($L_\mu^{\to}$)



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 $\lambda\mathbf{Y}$-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 samantics 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 ω-regular games.