Higher-order model-checking, categorical semantics, and linear logic

Charles Grellois (joint work with Paul-André Melliès)

PPS & LIAFA — Université Paris 7

TACL conference — June 21, 2015
Model-checking higher-order programs

- Construct a model $\mathcal{M}$ of a program
- Specify a property $\varphi$ in an appropriate logic
- Make them interact: the result is whether

$$\mathcal{M} \models \varphi$$

If $\mathcal{M}$ is a word, a tree... of actions: translate $\varphi$ to an equivalent automaton:

$$\varphi \mapsto A\varphi$$
Model-checking higher-order programs

For higher-order programs with recursion:

\[ \mathcal{M} \text{ is a higher-order tree:} \]
\[ \text{a tree produced by a higher-order recursion schemes (HORS)} \]

over which we run

\[ \text{an alternating parity tree automaton (APT) } \mathcal{A}_\varphi \]

corresponding to a

\[ \text{modal } \mu\text{-calculus formula } \varphi. \]

(NB: here modal \( \mu \)-calculus is equivalent to MSO)
Higher-order recursion schemes

\[ G = \begin{cases} 
S &= L \text{ Nil} \\
L \times &= \text{if } x (L \text{ (data } x \text{)}) 
\end{cases} \]

A HORS is a kind of deterministic higher-order grammar.

Rewrite rules have (higher-order) parameters.

“Everything” is simply-typed.

Rewriting produces a tree \( \langle G \rangle \).
Higher-order recursion schemes

\[ G = \begin{cases} 
  S & = & L \text{ Nil} \\
  L \; x & = & \text{if} \; x \; (L \; (\text{data} \; x)) 
\end{cases} \]

Rewriting starts from the start symbol \( S \):

\[ S \rightarrow_G L \quad \text{Nil} \]
Higher-order recursion schemes

\[
G = \begin{cases} 
  S & = & L \text{ Nil} \\
  L \ x & = & \text{if} \ x (L \ (\text{data} \ x))
\end{cases}
\]
Higher-order recursion schemes

\[
G = \begin{cases} 
S & = \text{L \ Nil} \\
L \ x & = \text{if \ } x \ (\text{L} \ (\text{data} \ x)) 
\end{cases}
\]
Higher-order recursion schemes

\[ \mathcal{G} = \left\{ \begin{array}{c}
S &=& L \text{ Nil} \\
L \ x &=& \text{if} \ x \ (L \ (\text{data} \ x))
\end{array} \right. \]

\[ \langle \mathcal{G} \rangle \text{ is an infinite non-regular tree.} \]

It is our model \( \mathcal{M} \).

How to model-check a non-regular tree?
Higher-order recursion schemes

\[ G = \begin{cases} 
S & = \ L \ \text{Nil} \\
L \ x & = \ \text{if} \ x \ (L \ (\text{data} \ x)) 
\end{cases} \]

HORS can alternatively be seen as simply-typed \( \lambda \)-terms with

free variables of order at most 1 (= tree constructors)

and

simply-typed recursion operators \( Y_\sigma : (\sigma \Rightarrow \sigma) \Rightarrow \sigma \).

Here: \( G \leftrightarrow (Y_o \Rightarrow_o (\lambda L. \lambda x. \text{if} \ x \ (L (\text{data} \ x)))) \ \text{Nil} \)

So, we can interpret HORS in models of the \( \lambda Y \)-calculus.
Alternating parity tree automata

For a modal $\mu$-calculus formula $\varphi$,

$$\langle G \rangle \models \varphi$$

iff an equivalent APT $A_\varphi$ has a run over $\langle G \rangle$.

$$\text{APT} = \text{alternating tree automata (ATA) + parity condition.}$$

Our goal: interpret HORS in a model reflecting the behavior of $A_\varphi$. 
Alternating tree automata

ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

Typically: $\delta(q_0, \text{if}) = (2, q_0) \land (2, q_1)$. 
Alternating tree automata

ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

Typically: $\delta(q_0, \text{if}) = (2, q_0) \land (2, q_1)$. 

\[
\begin{aligned}
&\text{if } q_0 \\
&\text{Nil } \quad \text{if } \quad \text{Nil} \\
&\text{data } \quad \text{if } \quad \text{Nil} \\
&\quad \text{data } \quad \text{:} \\
&\quad \text{Nil} \\
\end{aligned}
\begin{aligned}
&\text{if } q_0 \\
&\text{Nil } \quad \text{if } \quad \text{Nil} \\
&\text{data } \quad \text{if } \quad \text{Nil} \\
&\quad \text{data } \quad \text{:} \\
&\quad \text{Nil} \\
\end{aligned}
\]
Alternating tree automata

ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

Typically: \[ \delta(q_0, \text{if}) = (2, q_0) \land (2, q_1). \]

This infinite process produces a run-tree of \( A_\varphi \) over \( \langle G \rangle \).

It is an infinite, unranked tree.
A key remark (Kobayashi 2009):

\[
\delta(q_0, \text{if}) = (2, q_0) \land (2, q_1)
\]

can be seen as the intersection typing

\[
\text{if} : \emptyset \Rightarrow (q_0 \land q_1) \Rightarrow q_0
\]

refining the simple typing

\[
\text{if} : o \Rightarrow o \Rightarrow o
\]
Alternating tree automata and intersection types

In a derivation typing if $T_1 \ T_2$:

\[
\begin{align*}
\delta & \quad \emptyset \vdash \text{if} : \emptyset \Rightarrow (q_0 \land q_1) \Rightarrow q_0 \\
\text{App} & \quad \emptyset \vdash \text{if} \ T_1 : (q_0 \land q_1) \Rightarrow q_0 \\
\text{App} & \quad \Gamma_1 \vdash T_2 : q_0 \\
& \quad \Gamma_1 \vdash T_2 : q_1
\end{align*}
\]

\[
\emptyset \vdash \text{if} \ T_1 \ T_2 : q_0
\]

Intersection types naturally lift to higher-order – and thus to $G$, which finitely represents $\langle G \rangle$.

**Theorem (Kobayashi)**

\[
\emptyset \vdash G : q_0 \iff \text{the ATA } A_\varphi \text{ has a run-tree over } \langle G \rangle.
\]

A step towards decidability...
Intersection types and linear logic

\[ A \Rightarrow B = !A \multimap B \]

A program of type \( A \Rightarrow B \)

duplicates or drops elements of \( A \)

and then

uses linearly (= once) each copy

Just as intersection types and APT.
Intersection types and linear logic

\[ A \Rightarrow B = !A \multimap B \]

Two interpretations of the exponential modality:

**Qualitative models** (Scott semantics)

\[ !A = P_{\text{fin}}(A) \]

\[ [o \Rightarrow o] = P_{\text{fin}}(Q) \times Q \]

\[ \{q_0, q_0, q_1\} = \{q_0, q_1\} \]

**Order closure**

**Quantitative models** (Relational semantics)

\[ !A = M_{\text{fin}}(A) \]

\[ [o \Rightarrow o] = M_{\text{fin}}(Q) \times Q \]

\[ \{q_0, q_0, q_1\} \neq \{q_0, q_1\} \]

**Unbounded multiplicities**
Intersection types and linear logic

Models of linear logic and intersection types (refining simple types):

\[
\begin{array}{cccc}
\text{Rel} & \leftarrow & \text{Rel}_! & \leftarrow \nonumber \\
\text{Ehrhard} & \downarrow & \text{Ehrhard, } G-M & \downarrow \\
\text{Scott} & \leftarrow & \text{Scott}_! & \leftarrow \text{Terui} \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{Non-idempotent types} & \nonumber \nonumber \\
\text{Idempotent types} & \nonumber \\
\end{array}
\]

\[
\begin{array}{c}
[q_0, q_0, q_1] \rightarrow q_0 \rightarrow q_0 \land q_0 \land q_1 \rightarrow q_0 \\
\{q_0, q_1\} \rightarrow q_0 \rightarrow q_0 \land q_1 \rightarrow q_0 \\
\end{array}
\]
Intersection types and linear logic

Models of linear logic and intersection types (refining simple types):

\[ Rel \leftarrow Rel^! \leftarrow Non-\text{idempotent types} \]

\[ Ehrhard \]

\[ Scott \leftarrow Scott^! \leftarrow Idempotent types \]

\[ Terui \]

Fundamental idea: derivations of the intersection type systems compute denotations in the associated model.
Four theorems: inductive version

We obtain a theorem for every corner of our “equivalence square”:

**Theorem**

*In the relational (resp. Scott) semantics,*

\[ q_0 \in [G] \iff \text{the ATA } A_\phi \text{ has a finite run-tree over } \langle G \rangle. \]

**Theorem**

*With non-idempotent (resp. idempotent with subtyping) intersection types,*

\[ \vdash G : q_0 \iff \text{the ATA } A_\phi \text{ has a finite run-tree over } \langle G \rangle. \]
An infinitary model of linear logic

Rel and non-idempotent types lack of a countable multiplicity $\omega$. Recall that tree constructors are free variables...

In Rel, we introduce a new exponential $A \mapsto \# A$ s.t.

$$\llbracket \# A \rrbracket = M_{count}(\llbracket A \rrbracket)$$

(finite-or-countable multisets), so that

$$\llbracket A \Rightarrow B \rrbracket = \# \llbracket A \rrbracket \rightarrow \llbracket B \rrbracket = M_{count}(\llbracket A \rrbracket) \times \llbracket B \rrbracket$$
An infinitary model of linear logic

This defines an infinitary model of linear logic, which corresponds to non-idempotent intersection types with countable multiplicities and derivations of countable depth.

It admits a coinductive fixpoint, which we use to interpret $Y$.

The four theorems generalize to all ATA ($\rightarrow$ infinite runs).

And the parity condition ?
Alternating parity tree automata

MSO allows to discriminate inductive from coinductive behaviour.

This allows to express properties as

“a given operation is executed infinitely often in some execution”

or

“after a read operation, a write eventually occurs”.
Alternating parity tree automata

Each state of an APT receives a color

\[ \Omega(q) \in \text{Col} \subseteq \mathbb{N} \]

An infinite branch of a run-tree is winning iff the maximal color among the ones occurring infinitely often along it is even.

A run-tree is winning iff all its infinite branches are.

For a modal \( \mu \)-calculus formula \( \varphi \):

\[ \mathcal{A}_\varphi \text{ has a winning run-tree over } \langle G \rangle \text{ iff } \langle G \rangle \models \varphi \]
Alternating parity tree automata

Kobayashi and Ong’s type system has a quite complex handling of colors.

We reformulate it in a very simple way:

\[ \delta(q_0, \text{if}) = (2, q_0) \land (2, q_1) \]

now corresponds to

\[ \text{if} : \emptyset \Rightarrow (\Box_{\Omega(q_0)} q_0 \land \Box_{\Omega(q_1)} q_1) \Rightarrow q_0 \]

Application computes the “local” maximum of colors, and the fixpoint deals with the acceptance condition.

In this reformulation, the colors behave as a family of modalities.
The coloring comonad

Since coloring is a modality, it defines a comonad in the semantics:

$$\Box A = Col \times A$$

which can be composed with \(\not\otimes\) thanks to a distributive law.

Now:

$$[A \Rightarrow B] = \not\otimes \Box [A] \rightarrow [B] = M_{\text{count}}(Col \times [A]) \times [B]$$

We obtain a model of the \(\lambda\)-calculus which reflects the coloring by \(A_\phi\).
An inductive-coinductive fixpoint operator

We define a fixpoint operator:

- On typing derivations: rephrasal of the parity condition over derivations \( \rightarrow \) winning derivations.
- On denotations: it composes inductively or coinductively elements of the semantics, according to the current color.

The key here: parity conditions can be lifted to higher-order.

The fixpoint can also be defined using \( \mu \) and \( \nu \).
Open question: are the dotted lines an extensional collapse again?
Four theorems: full version

We obtain a theorem for every corner of our “colored equivalence square”:

Theorem (G-Melliès 2015)

In the colored relational (resp. colored Scott) semantics,

\[ q_0 \in \llbracket G \rrbracket \iff \text{the APT } A_\phi \text{ has a winning run-tree over } \langle G \rangle. \]

Theorem (G-Melliès 2015)

With colored non-idempotent (resp. colored idempotent with subtyping) intersection types, there is a winning derivation of

\[ \vdash G : q_0 \iff \text{the APT } A_\phi \text{ has a winning run-tree over } \langle G \rangle. \]
The selection problem

In the Scott/idempotent case, finiteness $\Rightarrow$ decidability of the higher-order model-checking problem.

Even better: the selection problem is decidable.
The selection problem

\[
\begin{cases}
S &= L \text{ Nil} \\
L &= \lambda x. \text{ if } x (L (\text{data } x))
\end{cases}
\]

becomes e.g.

\[
\begin{cases}
S_{q_0} &= L \{q_0, q_1\} \rightarrow q_0 \text{ Nil } q_0 \text{ Nil } q_1 \\
&\quad \text{if} \emptyset \rightarrow \{q_0, q_1\} \rightarrow q_0 \\
L \{q_0\} \rightarrow q_0 &= L \{q_0\} \rightarrow q_1 \\
L \{q_0, q_1\} \rightarrow q_0 &= \lambda x \{q_0, q_1\}.
\end{cases}
\]
Conclusion

- Higher-order model-checking $\rightarrow$ verification of non-regular trees.
- Semantic methods allow to study the term generating them.
- Models of linear logic can be extended to capture parity conditions.
- The semantic of a term reflects whether it satisfies a given property.
- Decidability $\rightarrow$ existence of a finite model.

Thank you for your attention!
Conclusion

- Higher-order model-checking → verification of non-regular trees.
- Semantic methods allow to study the term generating them.
- Models of linear logic can be extended to capture parity conditions.
- The semantic of a term reflects whether it satisfies a given property.
- Decidability → existence of a finite model.

Thank you for your attention!