Introduction to higher-order model-checking

Charles Grellois

LIS — équipe LIRICA

20 mai 2019
What is model-checking?
The halting problem

A natural question: does a program always terminate?

Undecidable problem (Turing 1936): a machine can not always determine the answer.

What if we use approximations?
Model-checking

Approximate the program \( \rightarrow \) build a model \( \mathcal{M} \).

Then, formulate a logical specification \( \varphi \) over the model.

Aim: design a program which checks whether

\[ \mathcal{M} \models \varphi. \]

That is, whether the model \( \mathcal{M} \) meets the specification \( \varphi \).
An example

\[
\begin{align*}
\text{Main} & = \text{Listen Nil} \\
\text{Listen } x & = \text{if end\_signal()} \text{ then } x \\
& \text{else Listen received\_data()} :: x
\end{align*}
\]
An example

\[
\begin{align*}
\text{Main} & = \text{Listen Nil} \\
\text{Listen } x & = \begin{cases} \\
\text{if } \text{end_signal()} \text{ then } x \\
\text{else } \text{Listen received_data()} : x
\end{cases}
\end{align*}
\]

A tree model:

We abstracted **conditionals** and **datatypes**.
The approximation contains a non-terminating branch.
Finite representations of infinite trees

is not regular: it is not the unfolding of a finite graph as
Finite representations of infinite trees

but it is represented by a higher-order recursion scheme (HORS).
Higher-order recursion schemes

Some regularity for infinite trees
Higher-order recursion schemes

\[
\begin{align*}
\text{Main} & = \text{Listen Nil} \\
\text{Listen } x & = \begin{cases} 
\text{if end_signal()} \text{ then } x \\
\text{else Listen received_data()}::x 
\end{cases}
\end{align*}
\]

is abstracted as

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

which represents the higher-order tree of actions

\[
\text{if} \\
\text{Nil if} \\
\text{data :} \\
\text{Nil}
\]
Higher-order recursion schemes

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

Rewriting starts from the start symbol $S$:
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

\[ G = \left\{ \begin{array}{ll}
S & = L \text{ Nil} \\
L \times & = \text{if } x (L \text{ (data } x))
\end{array} \right. \]
Higher-order recursion schemes

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

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

HORS can alternatively be seen as simply-typed \( \lambda \)-terms with simply-typed recursion operators \( Y_\sigma : (\sigma \rightarrow \sigma) \rightarrow \sigma \).
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 λ-terms with simply-typed recursion operators \( Y_\sigma : (\sigma \to \sigma) \to \sigma \).
Alternating parity tree automata

Checking specifications over trees

(see Chapter 2)
Monadic second order logic

MSO is a common logic in verification, allowing to express properties as:

« all executions halt »

« a given operation is executed infinitely often in some execution »

« every time data is added to a buffer, it is eventually processed »
Alternating parity tree automata

Checking whether a formula holds can be performed using an automaton.

For an MSO formula $\varphi$, there exists an equivalent APT $A_\varphi$ s.t.

$$\langle G \rangle \models \varphi \iff A_\varphi \text{ has a run over } \langle G \rangle.$$ 

$$\text{APT} = \text{alternating tree automata (ATA) + parity condition.}$$
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: $\delta(q_0, \text{if}) = (2, q_0) \land (2, q_1)$. 

\[
\begin{array}{c}
\text{if } q_0 \\
\text{Nil} \\
\text{data} \\
\text{Nil} \\
\end{array}
\quad \rightarrow_{A_\varphi}
\begin{array}{c}
\text{data} \\
\text{if} \\
\text{Nil} \\
\text{data} \\
\text{Nil} \\
\end{array}
\]

\[
\begin{array}{c}
\text{if } q_0 \\
\text{Nil} \\
\text{data} \\
\text{Nil} \\
\end{array}
\quad \rightarrow_{A_\varphi}
\begin{array}{c}
\text{data} \\
\text{if} \\
\text{Nil} \\
\text{data} \\
\text{Nil} \\
\end{array}
\]
Alternating parity tree automata

Each state of an APT is attributed 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.
Alternating parity tree automata

Each state of an APT is attributed a color

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

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 MSO formula \( \varphi \):

\[ \mathcal{A}_\varphi \text{ has a winning run-tree over } \langle G \rangle \quad \text{iff} \quad \langle G \rangle \models \varphi. \]
The higher-order model-checking problems
The (local) HOMC problem

Input: HORS $\mathcal{G}$, formula $\varphi$.

Output: true if and only if $\langle \mathcal{G} \rangle \models \varphi$.

Example: $\varphi = \langle \text{there is an infinite execution} \rangle$

Output: true.
The (local) HOMC problem

**Input:** HORS \( G \), formula \( \varphi \).

**Output:** true if and only if \( \langle G \rangle \models \varphi \).

Example: \( \varphi = " \) there is an infinite execution \( " \)

```
    if
   /   \
  Nil   if
         /   \
        data   if
               /     \
              Nil   data
                    /   \
                   data   Nil
```

Output: true.
The global HOMC problem

**Input:** HORS $G$, formula $\varphi$.

**Output:** a HORS $G^\bullet$ producing a marking of $\langle G \rangle$.

Example: $\varphi = \langle$ there is an infinite execution $\rangle$

Output: $G^\bullet$ of value tree:

```
        if^\bullet
       /   \
      /     \
    Nil     if^\bullet
    / \
   /   \
  data  if^\bullet
     /   \
    /     \
   Nil   data
    / \
   /   \
  data  Nil
```
The selection problem

**Input:** HORS $\mathcal{G}$, APT $\mathcal{A}$, state $q \in Q$.

**Output:** false if there is no winning run of $\mathcal{A}$ over $\langle \mathcal{G} \rangle$. Else, a HORS $\mathcal{G}^q$ producing a such a winning run.

Example: $\varphi = \langle$ there is an infinite execution $\rangle$, $q_0$ corresponding to $\varphi$.

Output: $\mathcal{G}^{q_0}$ producing

```
if^{q_0}
|   if^{q_0}
|   |   if^{q_0}
|   |   |   ...
```
Purpose of my thesis

These three problems are **decidable**, with elaborate proofs (often) relying on **semantics**.

**Our contribution:** an excavation of the semantic roots of HOMC, at the light of **linear logic**, leading to refined and clarified proofs.
Recognition by homomorphism

Where semantics comes into play
Automata and recognition

For the usual finite automata on words: given a regular language $L \subseteq A^*$, there exists a finite automaton $A$ recognizing $L$ if and only if...

there exists a finite monoid $M$, a subset $K \subseteq M$ and a homomorphism $\varphi : A^* \to M$ such that $L = \varphi^{-1}(K)$. 
Automata and recognition

The picture we want:

(after Aehlig 2006, Salvati 2009)

but with recursion and w.r.t. an APT.
Intersection types and alternation

A first connection with linear logic
Alternating tree automata and intersection types

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 the tree \( T_1 \) \( T_2 \):

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

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

**Theorem (Kobayashi 2009)**

\( \vdash G : q_0 \) iff the ATA \( A_\varphi \) has a run-tree over \( \langle G \rangle \).
A closer look at the Application rule

In the intersection type system:

\[
\Delta, \Delta_1, \ldots, \Delta_n \vdash t u : \theta \\
\Delta \vdash t : (\theta_1 \land \cdots \land \theta_n) \rightarrow \theta \\
\Delta_i \vdash u : \theta_i
\]

This rule could be decomposed as:

\[
\Delta \vdash t : (\land_{i=1}^{n} \theta_i) \rightarrow \theta' \\
\Delta_i \vdash u : \theta_i \\
\forall i \in \{1, \ldots, n\}
\]

\[
\Delta_1, \ldots, \Delta_n \vdash u : \land_{i=1}^{n} \theta_i \\
\Delta, \Delta_1, \ldots, \Delta_n \vdash t u : \theta'
\]
A closer look at the Application rule

In the intersection type system:

\[ \Delta \vdash t : (\theta_1 \land \cdots \land \theta_n) \to \theta \quad \Delta_i \vdash u : \theta_i \]
\[ \Delta, \Delta_1, \ldots, \Delta_n \vdash t \, u : \theta \]

This rule could be decomposed as:

\[ \Delta \vdash t : (\bigwedge_{i=1}^{n} \theta_i) \to \theta' \]
\[ \Delta_i \vdash u : \theta_i \quad \forall i \in \{1, \ldots, n\} \]
\[ \Delta_1, \ldots, \Delta_n \vdash u : \bigwedge_{i=1}^{n} \theta_i \]
\[ \Delta, \Delta_1, \ldots, \Delta_n \vdash t \, u : \theta' \]
A closer look at the Application rule

$$\Delta \vdash t : (\bigwedge_{i=1}^n \theta_i ) \rightarrow \theta'$$

$$\frac{\Delta_i \vdash u : \theta_i \quad \forall i \in \{1, \ldots, n\}}{\Delta_1, \ldots, \Delta_n \vdash u : \bigwedge_{i=1}^n \theta_i}$$

$$\Delta, \Delta_1, \ldots, \Delta_n \vdash t \ u : \theta'$$

Linear decomposition of the intuitionistic arrow:

$$A \Rightarrow B = ! A \multimap B$$

Two steps: duplication / erasure, then linear use.

Right \( \land \) corresponds to the Promotion rule of indexed linear logic.
(see G.-Melliès, ITRS 2014)
Intersection types and semantics of 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
An example of interpretation

In $\text{Rel}$, one denotation:

$$([q_0, q_1, q_1], [q_1], q_0)$$

In $\text{ScottL}$, a set containing the principal type

$$([\{q_0, q_1\}, \{q_1\}, q_0)$$

but also

$$([\{q_0, q_1, q_2\}, \{q_1\}, q_0)$$

and

$$([\{q_0, q_1\}, \{q_0, q_1\}, q_0)$$

and \ldots
Intersection types and semantics of linear logic


Fundamental idea:

\[
\llbracket t \rrbracket \cong \{ \theta \mid \emptyset \vdash t : \theta \}
\]

for a closed term.
Intersection types and semantics of linear logic

Let $t$ be a term normalizing to a tree $\langle t \rangle$ and $\mathcal{A}$ be an alternating automaton.

$$\mathcal{A} \text{ accepts } \langle t \rangle \text{ from } q \iff q \in \llbracket t \rrbracket \iff \emptyset \vdash t : q :: o$$

(see Chapter 5)

Extension with recursion and parity condition?
Adding parity conditions to the type system
Alternating parity tree automata

We add coloring annotations to intersection types:

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

now corresponds to

$$\text{if} : \emptyset \rightarrow (\square_{\Omega(q_0)} q_0 \land \square_{\Omega(q_1)} q_1) \rightarrow q_0$$

Idea: if is a run-tree with two holes:

$$\text{if}$$

$$[\,]q_0 \quad []q_1$$

A new neutral (least) color: $\epsilon$.

We refine the approach of Kobayashi and Ong in a modal way (see Chapter 6).
An example of colored intersection type

Set $\Omega(q_0) = 0$ and $\Omega(q_1) = 1$.

\[
\lambda x \lambda y \ a \ q_1
\]

\[
\lambda x
\]

\[
\lambda y
\]

\[
a \ q_1
\]

\[
a \ q_0 \quad a \ q_1
\]

\[
x \ q_0 \quad y \ q_1 \quad x \ q_1 \quad x \ q_1
\]

has now type

\[
\Box_0 q_0 \land \Box_1 q_1 \rightarrow \Box_1 q_1 \rightarrow q_1
\]

Note the color 0 on $q_0$…
A type-system for verification (Grellois-Melliès 2014)

Axiom

\[ \frac{}{x : \Box_\epsilon \theta_i \vdash x : \theta_i} \]

\[ \{ (i, q_{ij}) \mid 1 \leq i \leq n, 1 \leq j \leq k_i \} \] satisfies \( \delta_A(q, a) \)

\[ \emptyset \vdash a : \bigwedge_{j=1}^{k_1} \Box_\Omega(q_{1j}) q_{1j} \to \cdots \to \bigwedge_{j=1}^{k_n} \Box_\Omega(q_{nj}) q_{nj} \to q \]

App

\[ \frac{\Delta \vdash t : (\Box_{m_1} \theta_1 \land \cdots \land \Box_{m_k} \theta_k) \to \theta \quad \Delta_i \vdash u : \theta_i}{\Delta + \Box_{m_1} \Delta_1 + \cdots + \Box_{m_k} \Delta_k \vdash t \ u : \theta} \]

\[ \frac{}{\Delta, x : \bigwedge_{i \in I} \Box_{m_i} \theta_i \vdash t : \theta} \]

\[ \frac{\Gamma \vdash R(F) : \theta}{\Gamma \vdash \Box_\epsilon \theta \vdash F : \theta} \]

\[ \frac{}{\Delta \vdash \lambda x. t : (\bigwedge_{i \in I} \Box_{m_i} \theta_i) \to \theta} \]

\[ \frac{}{\Gamma \vdash \mathcal{R}(F) : \theta} \]
A type-system for verification

A colored Application rule:

\[
\begin{align*}
\Delta \vdash t : (\square m_1 \theta_1 \land \cdots \land \square m_k \theta_k) \to \theta & \quad \Delta_i \vdash u : \theta_i \\
\Delta + \square m_1 \Delta_1 + \cdots + \square m_k \Delta_k \vdash t \ u : \theta
\end{align*}
\]

(Leaves colored on the tree are not used in the rule.)

\[
\frac{\Delta \vdash t : (\square m_1 \theta_1 \land \cdots \land \square m_k \theta_k) \to \theta \quad \Delta_i \vdash u : \theta_i}{\Delta + \square m_1 \Delta_1 + \cdots + \square m_k \Delta_k \vdash t \ u : \theta}
\]
A type-system for verification

A colored Application rule:

\[
\begin{array}{c}
\Delta \vdash t : (\square m_1 \theta_1 \land \cdots \land \square m_k \theta_k) \to \theta \\
\Delta + \square m_1 \Delta_1 + \cdots + \square m_k \Delta_k \vdash t \ u : \theta
\end{array}
\]

inducing a winning condition on infinite proofs: the node

\[
\Delta_i \vdash u : \theta_i
\]

has color \( m_i \), others have color \( \epsilon \), and we use the parity condition.
A type-system for verification

We now capture all MSO (see Chapter 6-8):

**Theorem (G.-Melliès 2014, from Kobayashi-Ong 2009)**

\[ S : q_0 \vdash S : q_0 \text{ admits a winning typing derivation iff the alternating parity automaton } A \text{ has a winning run-tree over } \langle G \rangle. \]

We obtain decidability by considering idempotent types.

Our reformulation

- shows the modal nature of \( \Box \) (in the sense of S4),
- internalizes the parity condition,
- paves the way for semantic constructions.
Colored models of linear logic
A closer look at the Application rule

\[
\Delta \vdash t : (\Box m_1 \theta_1 \land \cdots \land \Box m_k \theta_k) \rightarrow \theta \quad \Delta_i \vdash u : \theta_i \\
\Delta + \Box m_1 \Delta_1 + \ldots + \Box m_k \Delta_k \vdash t u : \theta
\]

could be decomposed as:

\[
\Delta \vdash t : \left(\bigwedge_{i=1}^k \Box m_i \theta_i\right) \rightarrow \theta \\
\Box m_1 \Delta_1 \vdash u : \Box m_1 \theta_1 \quad \ldots \quad \Box m_k \Delta_k \vdash u : \bigwedge_{i=1}^k \Box m_i \theta_i \\
\Delta, \Box m_1 \Delta_1, \ldots, \Box m_k \Delta_k \vdash t u : \theta
\]

Right \Box \text{ looks like a promotion. In linear logic:}

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

We show that the modality \Box \text{ distributes over the exponential in the semantics.}
Colored semantics

We extend:

- $Rel$ with **countable** multiplicities, **coloring** and an **inductive-coinductive** fixpoint (Chapter 9)
- $ScottL$ with **coloring** and an **inductive-coinductive** fixpoint (Chapter 10).

Methodology: think in the relational semantics, and adapt to the Scott semantics using Ehrhard’s 2012 result:

the **finitary** model $ScottL$ is the extensional collapse of $Rel$. 
Infinitary relational semantics

Extension of \( \text{Rel} \) with infinite multiplicities:

\[
\downarrow A = \mathcal{M}_{\text{count}}(A)
\]

and coloring modality (parametric comonad)

\[
\Box A = \text{Col} \times A
\]

Composite comonad: \( \downarrow = \downarrow \Box \) is an exponential.

Induces a colored CCC \( \text{Rel}_\downarrow \) (\( \rightarrow \) model of the \( \lambda \)-calculus).
An example of interpretation

Set $\Omega(q_i) = i$.

A tree diagram illustrating the interpretation of $\lambda x \lambda y a q_1$.

The denotation is:

$$([(0, q_0), (1, q_1), (1, q_1)], [(1, q_1)], q_1)$$

(corresponding to the type $\Box_0 q_0 \land \Box_1 q_1 \rightarrow \Box_1 q_1 \rightarrow q_1$)
Model-checking and infinitary semantics

**Inductive-coinductive fixpoint operator**: composes denotations w.r.t. the parity condition.

**Theorem**

An APT $A$ has a winning run from $q_0$ over $\langle G \rangle$ if and only if

$$q_0 \in \llbracket \lambda(G) \rrbracket_A$$

where $\lambda(G)$ is a $\lambda Y$-term corresponding to $G$.

**Conjecture**

An APT $A$ has a winning run from $q_0$ over $\langle G \rangle$ if and only if

$$q_0 \in \llbracket \lambda(G)^\Sigma \rrbracket \circ \llbracket \delta^\dagger \rrbracket$$

where $\lambda(G)^\Sigma$ is a Church encoding of a $\lambda Y$-term corresponding to $G$. 
Finitary semantics

In ScottL, we define $\Box$, $\lambda$ and $\mathbf{Y}$ similarly (using downward-closures). $\mathit{ScottL}$ is a model of the $\lambda\mathbf{Y}$-calculus.

**Theorem**

An APT $A$ has a winning run from $q_0$ over $\langle G \rangle$ if and only if

$$q_0 \in [\lambda(G)].$$

**Corollary**

The local higher-order model-checking problem is decidable (and is $n$-EXPTIME complete).

**Theorem**

The selection problem is decidable.
Perspectives

- A purely coinductive proof of the soundness-and-completeness theorem
- Accommodating the modal approach to other classes of automata
- Understanding the infinitary semantics
- Logical aspects: colored tensorial logic, fixpoints...
- Game semantics interpretations?
- Is the complexity related to light linear logics?
- Extensional collapse between the two colored models?