Introduction to higher-order model-checking

Charles Grellois

LIS — équipe LIRICA

Séminaire LDP
20 juin 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

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

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

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

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

is abstracted as

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

which represents the higher-order tree of actions

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

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

Rewriting starts from the start symbol $S$:

\[
\xymatrix{
S \ar[r]^G & L \text{ Nil} \\
& L \\
& \text{Nil}
}
\]
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

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

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

\[ \langle G \rangle = \begin{cases} 
\text{if} \\
\text{Nil} & \text{if} \\
\text{data} & \text{if} \\
\text{Nil} & \text{data} \\
\text{data} & \text{Nil} 
\end{cases} \]
Higher-order recursion schemes

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

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 \( \lambda \)-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.

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

$\text{APT} = \text{alternating tree automata (ATA) } + \text{parity condition.}$
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{center}
\begin{tikzpicture}
\node [state, fill=red] (q0) {$q_0$} [grow = down, sibling distance=3cm, level distance=3cm]
    child {node [state] (nil) {$\text{Nil}$} [grow = left]
        child {node [state] (data) {$\text{data}$} [grow = left]
            child {node [state] (nil) {$\text{Nil}$} [grow = left]}
            child {node [state] (data) {$\text{data}$} [grow = left]}
            child {node [state] (nil) {$\text{Nil}$} [grow = left]}
        }
        child {node [state] (if) {$\text{if}$} [grow = left]}
    }
    child {node [state] (if) {$\text{if}$} [grow = left]}
\end{tikzpicture}
\end{center}
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.

\begin{figure}
\centering
\begin{tikzpicture}
\draw (0,0) -- (1,1) node[midway,above]{$c_1$};
\draw (1,1) -- (2,2) node[midway,above]{$c_2$};
\draw (2,2) -- (3,3) node[midway,above]{$c_3$};
\draw (3,3) -- (4,4) node[midway,above]{$c_4$};
\draw (4,4) -- (5,5) node[midway,above]{$c_5$};
\end{tikzpicture}
\end{figure}
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.

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

For a MSO formula \( \varphi \):

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

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

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

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

Output: $\text{true}$.
The (local) HOMC problem

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

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

Example: $\varphi = \llbracket \text{there is an infinite execution} \rrbracket$

Output: $\text{true}$.
The global HOMC problem

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

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

Example: $\varphi = «$ there is an infinite execution »

Output: $\mathcal{G}^\bullet$ of value tree:
The selection problem

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

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

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

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

\[
\text{if } q_{0} \\
\quad \text{if } q_{0} \\
\quad \quad \text{if } q_{0} \\
\quad \quad \quad \vdots
\]
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 if $T_1 \ T_2$:

\[
\begin{array}{c}
\delta \\
\text{App} \quad \text{App} \\
\text{App}
\end{array}
\]

\[
\begin{array}{c}
\emptyset \vdash \text{if} : \emptyset \rightarrow (q_0 \land q_1) \rightarrow q_0 & \emptyset \\
\emptyset \vdash \text{if } T_1 : (q_0 \land q_1) \rightarrow q_0 & \emptyset \\
\emptyset \vdash \text{if } T_1 \ T_2 : q_0 & \emptyset \\
\end{array}
\]

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:

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

This rule could be decomposed as:

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

In the intersection type system:

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

This rule could be decomposed as:

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

\[ \begin{array}{c}
\Delta \vdash t : (\bigwedge_{i=1}^n \theta_i) \to \theta' \\
\Delta, \Delta_1, \ldots, \Delta_n \vdash t u : \theta'
\end{array} \]

\[ \begin{array}{c}
\Delta_i \vdash u : \theta_i \\
\forall i \in \{1, \ldots, n\} \\
\Delta_1, \ldots, \Delta_n \vdash u : \bigwedge_{i=1}^n \theta_i
\end{array} \]

Linear decomposition of the intuitionistic arrow:

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

Two steps: duplication / erasure, then linear use.

Right \( \bigwedge \) 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)

\[ !A = \mathcal{P}_{\text{fin}}(A) \]

\[ [o \Rightarrow o] = \mathcal{P}_{\text{fin}}(Q) \times Q \]

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

Order closure

**Quantitative models**  
(Relational semantics)

\[ !A = \mathcal{M}_{\text{fin}}(A) \]

\[ [o \Rightarrow o] = \mathcal{M}_{\text{fin}}(Q) \times Q \]

\[ [q_0, q_0, q_1] \neq [q_0, q_1] \]

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 ...
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

\[
\begin{array}{c}
\text{Rel}! & \xleftarrow{\text{Bucciarelli–Ehrhard}} & \text{Non-idempotent types} \\
\downarrow & & \downarrow \text{Ehrhard} \\
\text{Ehrhard} & \xleftarrow{\text{de Carvalho}} & \text{Idempotent types} \\
\text{ScottL}! & \xleftarrow{\text{Ehrhard}} & \text{Terui} \\
\end{array}
\]

Let \( t \) be a term normalizing to a tree \( \langle t \rangle \) and \( A \) be an alternating automaton.

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

(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

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

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

$$\text{if}$$

$$\begin{array}{c}
[]q_0 \\
[]q_1
\end{array}$$

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 $\Omega(q_1) = 1$.

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

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

\[\delta\]

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

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

App

\[
\Delta \vdash t : (\bigwedge_{m_1} \theta_1 \wedge \cdots \wedge \bigwedge_{m_k} \theta_k) \rightarrow \theta \\
\Delta \vdash u : \theta_i
\]

\[
\Delta \vdash t u : \theta
\]

\[
\lambda
\]

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

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

\[
fix
\]

\[
\Gamma \vdash R(F) : \theta
\]

\[
\frac{}{F : \Box_\epsilon \theta \vdash F : \theta}
\]
A type-system for verification

A colored Application rule:

\[
\begin{align*}
\text{App} & \quad \frac{\Delta \vdash t : (\Box m_1 \theta_1 \land \cdots \land \Box m_k \theta_k) \rightarrow \theta \quad \Delta_i \vdash u : \theta_i}{\Delta + \Box m_1 \Delta_1 + \cdots + \Box m_k \Delta_k \vdash t \ u : \theta}
\end{align*}
\]
A type-system for verification

A colored Application rule:

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

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

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

could be decomposed as:

\[
\begin{align*}
\Delta & \vdash t : \bigwedge_{i=1}^k \Box m_i \theta_i \rightarrow \theta & \Box m_1 \Delta_1 & \vdash u : \Box m_1 \theta_1 & \ldots & \Box m_k \Delta_k & \vdash u : \Box m_k \theta_k \\
\Delta, \Box m_1 \Delta_1, \ldots, \Box m_k \Delta_k & \vdash tu : \theta
\end{align*}
\]

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

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

We show that the modality \Box 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 $\Omega(q_i) = i$.

\[
\lambda x \\
\lambda y \\
a \ q_1 \\
a \ q_0 \ x \ q_0 \ y \ q_1 \ x \ q_1 \ x \ q_1
\]

has denotation

\[
(\[(0, q_0), (1, q_1), (1, q_1)\], [(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 $\mathcal{A}$ has a winning run from $q_0$ over $\langle G \rangle$ if and only if

$$q_0 \in [\lambda(G)]_\mathcal{A}$$

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

**Conjecture**

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

$$q_0 \in [\lambda(G)^\Sigma] \circ [\delta^\dagger]$$

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

In ScottL, we define □, λ and Y similarly (using downward-closures). ScottL↓ is a model of the λY-calculus.

Theorem

An APT A has a winning run from q₀ over \( \langle G \rangle \) if and only if

\[ q₀ \in [\lambda(G)]. \]

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?