Relational semantics of linear logic
and higher-order model-checking

Charles Grellois    Paul-André Melliès

PPS & LIAFA — Université Paris 7
University of Dundee

CSL — September 8, 2015
TU Berlin
Model-checking higher-order programs

A well-known approach in verification: model-checking.

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

$$M \models \varphi$$

When the model is a word, a tree... of actions: translate $\varphi$ to an equivalent automaton:

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

A well-known approach in verification: \textit{model-checking}.

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

$$M \models \varphi$$

When the model is a word, a tree... of actions: translate $\varphi$ to an \textit{equivalent automaton}:

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

A well-known approach in verification: model-checking.

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

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

When the model is a word, a tree... of actions: translate $\varphi$ to an equivalent automaton:

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

A well-known approach in verification: model-checking.

- Construct a model $M$ of a program $\rightarrow$ higher-order trees
- Specify a property $\varphi$ in an appropriate logic $\rightarrow$ MSO
- Make them interact: the result is whether

$$M \models \varphi$$

When the model is a word, a tree... of actions: translate $\varphi$ to an equivalent automaton:

$$\varphi \mapsto A_\varphi$$

$\rightarrow$ alternating parity tree automata (APT)
Trees and types

Model-checking of infinite trees of actions:

Three actions here: $\Sigma = \{ \text{if} : 2, \text{data} : 1, \text{Nil} : 0 \}$.

Call $\sigma$ the type of trees (and more generally of terms with free variables of order $\leq 1$, given by $\Sigma$).
Trees and types

An element of type $o \rightarrow o$:

\[
\lambda x \text{ if } \text{if } \text{data} \text{ if } \text{data} : \text{data} \text{ if } x
\]

Applying it to Nil gives the previous tree.
where “\(\lambda \Sigma\)” stands for \(\lambda \text{if} \cdot \lambda \text{data} \cdot \lambda \text{Nil}.\), has type:

\[
o(\Sigma) \rightarrow o = (o \rightarrow o \rightarrow o) \rightarrow (o \rightarrow o) \rightarrow o \rightarrow o
\]

Church encoding of trees.
Linear decomposition of the intuitionistic arrow

In linear logic,

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

\( ! A \) allows to duplicate or to drop \( A \)

\( \multimap \) uses linearly (once) each copy
Linear decomposition of the intuitionistic arrow

\[(o \rightarrow o \rightarrow o) \rightarrow (o \rightarrow o) \rightarrow o \rightarrow o\]

translates as

\[!(o \rightarrow !o \rightarrow o) \rightarrow !(o \rightarrow o) \rightarrow !o \rightarrow o\]

In the relational semantics of linear logic, with \([o] = Q\),

\[![A] = \mathcal{M}_{fin}([A])\quad\text{and}\quad [A \rightarrow B] = [A] \times [B]\]

For instance,

\[[o \rightarrow o \rightarrow o] = \mathcal{M}_{fin}(Q) \times \mathcal{M}_{fin}(Q) \times Q\]
Linear decomposition of the intuitionistic arrow

\[(o \rightarrow o \rightarrow o) \rightarrow (o \rightarrow o) \rightarrow o \rightarrow o\]

translates as

\[(! (l o \rightarrow! o \rightarrow o) \rightarrow! (l o \rightarrow o) \rightarrow! o \rightarrow o)\]

Complain: where is model-checking?

We mentioned alternating parity tree automata…
Alternating parity tree automata

For a MSO formula \( \varphi \),

\[ \langle G \rangle \models \varphi \]

iff an equivalent APT \( A_{\varphi} \) has a run over \( \langle G \rangle \).

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

\[
\begin{array}{c}
\text{if } q_0 \\
\text{Nil} & \text{if}
\end{array}
\]

\[
\begin{array}{c}
\text{data} & \text{if} \\
\text{Nil} & \text{Nil} & \text{data}
\end{array}
\]

\[
\begin{array}{c}
\text{data} \\
\text{Nil}
\end{array}
\]

\[
\begin{array}{c}
\text{if } q_0 \\
\text{if } q_0 \\
\text{if } q_1
\end{array}
\]

\[
\begin{array}{c}
\text{data} & \text{if} \\
\text{Nil} & \text{Nil}
\end{array}
\]

\[
\begin{array}{c}
\text{data} \\
\text{Nil}
\end{array}
\]

\[
\begin{array}{c}
\text{if} \\
A_\varphi
\end{array}
\]
Alternating tree automata

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

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

In fact, \text{if} has the linear type

\[
\text{if} : \! o \multimap ! o \multimap o
\]

so that in the relational semantics of linear logic, setting \([o] = Q\),

\[
[i] \subseteq M_{\text{fin}}(Q) \times M_{\text{fin}}(Q) \times Q
\]

and

\[
([], [q_0, q_1], q_0) \in [\text{if}]
\]
Model-checking I

An alternating tree automaton over $\Sigma$, with set of states $Q$, of transition function $\delta$, provides

\[
[\delta] = [\text{if}] \times [\text{data}] \times [\text{Nil}] \subseteq [o(\Sigma)]
\]

while a tree $t$ over $\Sigma$ gives, under Church encoding:

\[
[t] \subseteq [o(\Sigma) \rightarrow o] = M_{\text{fin}}([o(\Sigma)]) \times Q
\]

Relational composition:

\[
[t] \circ M_{\text{fin}}([\delta]) \subseteq Q
\]
An alternating tree automaton over $\Sigma$, with set of states $Q$, of transition function $\delta$, provides

$$[[\delta]] = [[\text{if}]] \times [[\text{data}]] \times [[\text{Nil}]] \subseteq [[o(\Sigma)]]$$

while a tree $t$ over $\Sigma$ gives, under Church encoding:

$$[[t]] \subseteq [[o(\Sigma) \rightarrow o]] = M_{\text{fin}}([[o(\Sigma)]) \times Q$$

Relational composition:

$$[[t]] \circ M_{\text{fin}}([[\delta]]) \subseteq Q$$
Relational composition:

\[ [t] \circ M_{\text{fin}}([\delta]) \subseteq Q \]

**Proposition**

\[ [t] \circ M_{\text{fin}}([\delta]) \]

is the set of states \( q \) from which

\[ A = \langle \Sigma, Q, \delta, q \rangle \]

accepts the tree \( t \).
Model-checking I

Rel is a denotational model:

\[ t \rightarrow_{\beta} t' \implies [t] = [t'] \]

Corollary

For a term

\[ t : o(\Sigma) \rightarrow o \]

(= normalizing to a finite \(\Sigma\)-labelled ranked tree),

\[ [t] \circ M_{\text{fin}}([\delta]) \]

is the set of states \(q\) from which

\[ \mathcal{A} = \langle \Sigma, Q, \delta, q \rangle \]

accepts the tree \(< t >\) generated by the normalization of \(t\).
Higher-order model-checking

We want to model-check

- **higher-order trees** ("non-regular, yet of finite representation"), as

\[
\begin{align*}
\text{if} & \quad \text{if} \\
\text{Nil} & \quad \text{data} \\
\text{Nil} & \quad \text{data} \\
\end{align*}
\]

and to account for **parity conditions**.
Higher-order recursion schemes

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

is represented as the higher-order recursion scheme (HORS)
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 \( S \):

\[ \begin{array}{c}
  S \\
  \rightarrow_G \\
  L \\
  \text{Nil}
\end{array} \]
Higher-order recursion schemes

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

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

\[ \langle G \rangle \] is an infinite non-regular tree.

It is our model \( M \).
Higher-order recursion schemes

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

HORS can alternatively be seen as an extension of the simply-typed \( \lambda \)-terms we considered so far, with

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

Here: \[ G \leftrightarrow (Y_{\text{o} \rightarrow \text{o}} (\lambda \text{L.} \lambda x. \text{if } x (\text{L } (\text{data } x ))) ) \text{ Nil} \]

So we need to add fixpoints to the relational model.
Model-checking II

Rel has an inductive fixpoint operator (finite iteration). We obtain:

Theorem

For a λY-term

\[ t : o(\Sigma) \rightarrow o \]

(= normalizing to an infinite \( \Sigma \)-labelled ranked tree),

\[ [t] \circ M_{\text{fin}}([\delta]) \]

is the set of states \( q \) from which

\[ \mathcal{A} = \langle \Sigma, Q, \delta, q \rangle \]

accepts the tree \(< t >\) generated by the coinductive normalization of \( t \)

during a finite execution
On finiteness

Why a finite execution?

Because constructors = free variables.

Infinite trees need infinite multisets.

So we define a new exponential

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

The resulting model has a coinductive operator (\( \approx \) infinite fixpoint unfolding).

(see G.-Melliès, Fossacs 2015)
Model-checking III

With the coinductive fixpoint of this infinitary model:

**Theorem**

For a $\lambda Y$-term

$$t : o(\Sigma) \rightarrow o$$

(= normalizing to an *infinite* $\Sigma$-labelled ranked tree),

$$[t] \circ M_{\text{fin}}([\delta])$$

is the set of states $q$ from which

$$A = \langle \Sigma, Q, \delta, q \rangle$$

accepts the tree $< t >$ generated by the *coinductive normalization of* $t$

*during a finite or infinite execution*
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 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 $\varphi$:

$$\mathcal{A}_\varphi$$ has a winning run-tree over $\langle G \rangle$ iff $\langle G \rangle \models \varphi$
The coloring comonad

In the proceedings paper, we show that coloring is a modality. It defines a comonad in the semantics:

\[ \Box A = Col \times A \]

which can be composed with \( \otimes \), giving an infinitary, colored model of linear logic in which

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

corresponds to

\[ ([], [(\Omega(q_0), q_0), (\Omega(q_1), q_1)], q_0) \in [[\text{if}]] \]

in the semantics.
In this setting, $t$ has some type $\Box c_1 \sigma_1 \land \Box c_2 \sigma_2 \rightarrow \tau$.

The color labelling each occurrence is the maximal color leading to it in the normal form of $t$.

On applications, the comonad computes the maximal color (inductive treatment).
Model-checking IV

We define an inductive-coinductive fixpoint operator on denotations, which iterates finitely or infinitely depending on the current color. It is a Conway operator (Bloom-Esik).

**Theorem**

For a $\lambda Y$-term

$$t : o(\Sigma) \rightarrow o$$

(= normalizing to an infinite $\Sigma$-labelled ranked tree),

$$\llbracket t \rrbracket \circ M_{\text{fin}}(\llbracket \delta \rrbracket)$$

is the set of states $q$ from which the alternating parity automaton

$$A = \langle \Sigma, Q, \delta, q \rangle$$

accepts the tree $\langle t \rangle$ generated by the coinductive normalization of $t$. 
Ehrhard 2012: *ScottL* is the extensional collapse of *Rel*.

G.-Melliès, MFCS 2015: adaptation to *ScottL* of the theoretical approach of this work.

**Corollary**

*The higher-order model-checking problem is decidable.*

The resulting model is similar in the spirit to the one of Salvati and Walukiewicz, with subtle differences, notably on color handling and composition of morphisms.
Conclusion

- **Linear logic** reveals a very natural **duality** between terms and (alternating) automata.
- **Models** can be extended to handle additional conditions on automata (parity . . .)
- Relational semantics are **infinitary**, but their simplicity eases theoretical reasoning on problems.

In the proceedings:

- More on the **duality** aspects, and on the **extended relational semantics**.
- Discussion on the **modal nature** of coloring, and its relations with prior work of Kobayashi and Ong.
- Technical work is based on an equivalent **intersection type system**.

Thank you for your attention!
Conclusion

- **Linear logic** reveals a very natural **duality** between terms and (alternating) automata.
- **Models** can be extended to handle additional conditions on automata (parity...)
- Relational semantics are **infinitary**, but their simplicity eases theoretical reasoning on problems.

In the proceedings:

- More on the **duality** aspects, and on the **extended relational semantics**.
- Discussion on the **modal nature** of coloring, and its relations with prior work of Kobayashi and Ong.
- Technical work is based on an equivalent **intersection type system**.

Thank you for your attention!