Linear logic, duality, and higher-order model-checking

Charles Grellois (joint work with Paul-André Melliès)

PPS & LIAFA — Université Paris 7
University of Dundee

Scottish Programming Languages Seminar
University of Edinburgh
October 21, 2015
Model-checking higher-order programs

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

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

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

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

For higher-order programs with recursion, $\mathcal{M}$ is a higher-order tree.

Example:

$$\begin{align*}
\text{Main} & = \text{Listen Nil} \\
\text{Listen } x & = \text{if } end \text{ then } x \text{ else Listen (data } x) \\
\end{align*}$$

modelled as

```
if
  if
    data
      if
        Nil
          if
            data
              if
                Nil
                  data
                  Nil
```
Model-checking higher-order programs

For higher-order programs with recursion, $\mathcal{M}$ is a higher-order tree.

Example:

\[
\begin{align*}
\text{Main} & \quad = \quad \text{Listen Nil} \\
\text{Listen } x & \quad = \quad \text{if end then } x \text{ else Listen (data } x) \\
\end{align*}
\]

modelled as

How to represent this tree finitely?
Model-checking higher-order programs

For higher-order programs with recursion, $\mathcal{M}$ is a higher-order tree over which we run

an alternating parity tree automaton (APT) $A_\varphi$

corresponding to a

monadic second-order logic (MSO) formula $\varphi$.

(safety, liveness properties, etc)

Can we decide whether a higher-order tree satisfies a MSO formula?
Trees vs. tree automata
Trees and types

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

Ground type: o is the type of trees (and more generally of terms over \( \Sigma \) reducing to a tree).
Applying it to \texttt{Nil} gives the previous tree.
Trees and types

Church encoding of trees:

\[
\lambda \Sigma \\
\quad \text{if} \\
\quad \text{Nil} \\
\quad \text{if} \\
\quad \text{data} \\
\quad \text{if} \\
\quad \text{Nil} \\
\quad \text{if} \\
\quad \text{data} \\
\quad \text{Nil} \\
\quad : o(\Sigma) \rightarrow o
\]

where “\(\lambda \Sigma\)” stands for \(\lambda \text{if}. \lambda \text{data}. \lambda \text{Nil}.\), and

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

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

and

\[[A \rightarrow B] = [A] \times [B]\]

For instance,

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

What does this mean for Church encoding of trees?
Linear decomposition of the intuitionnistic arrow

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

translates as

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

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 $\langle G \rangle$.

$$\text{AP\text{T} }= \text{ alternating tree automata (ATA) }+ \text{ parity condition.}$$
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: \( \delta(q_0, \text{if}) = (2, q_0) \wedge (2, q_1) \).
Alternating tree automata

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

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

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

$$\text{if} : !o \multimap !o \multimap o$$

so that in the relational semantics of linear logic

$$(\emptyset, [q_0, q_1], q_0) \in [\text{if}] \subseteq M_{\text{fin}}(Q) \times M_{\text{fin}}(Q) \times Q$$
An alternating tree automaton over $\Sigma$, with set of states $Q$, of transition function $\delta$, provides

$$\llbracket \delta \rrbracket = \llbracket \text{if} \rrbracket \cup \llbracket \text{data} \rrbracket \cup \llbracket \text{Nil} \rrbracket \subseteq \llbracket o(\Sigma) \rrbracket$$

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

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

Relational composition:

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

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

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

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:

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

Interactive interpretation?
Model-checking I

Relational composition:

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

**Proposition**

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

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

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

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

Rel is a denotational model:

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

Corollary

For a term \( t : o(\Sigma) \rightarrow o \)

the set of states \( q \) from which

\( A = \langle \Sigma, Q, \delta \rangle \)

accepts the tree generated by the normalization of \( t \) is

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

Static analysis, directly on the term.
Higher-order model-checking

We want to model-check

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

```
  if
  \hspace{2em} Nil if
  \hspace{2em} data if
  \hspace{4em} Nil data :
  \hspace{6em} data
  \hspace{8em} Nil
```

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

Some regularity for infinite trees
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

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

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

\[ S \rightarrow_{G} \]

\[ L \]

\[ \text{Nil} \]
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

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

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

“Everything” is simply-typed, and

well-typed programs can’t go too wrong:

we can detect productivity, and enforce it (replace divergence by outputing a distinguished symbol Ω in one step).
Higher-order recursion schemes

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

“Everything” is simply-typed, and

well-typed programs can’t go too wrong:

we can detect productivity, and enforce it (replace divergence by outputing a distinguished symbol \(\Omega\) in one step).

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

\(Y_\sigma : (\sigma \rightarrow \sigma) \rightarrow \sigma\).

\(\rightarrow\) add fixpoints to the model.
Finite iteration $\rightarrow$ inductive fixpoint operator on $Rel$.

**Theorem**

The infinitary normal form of a $\lambda Y$-term $t : o(\Sigma) \rightarrow o$ is accepted by $A = \langle \Sigma, Q, \delta \rangle$ from the set of states $[t] \circ M_{fin}([\delta])$. 
Finite iteration $\rightarrow$ inductive fixpoint operator on $Rel$.

**Theorem**

The infinitary normal form of a $\lambda Y$-term

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

is accepted by

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

from the set of states

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

**after a finite execution of the automaton.**
On finiteness

Infinite trees need infinite multisets: tree constructors may be used countably.

Defining a new exponential

\[ \triangleright : A \mapsto M_{\text{count}}(A) \]

gives a relational model of linear logic with a

coinductive fixpoint operator

(infinite fixpoint unfolding).

New interpretation of terms: \([t]_{\text{gfp}}\).
Theorem

The infinitary normal form of a $\lambda Y$-term

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

is accepted by

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

from the set of states

$$\llbracket t \rrbracket_{gfp} \circ \mathcal{M}_{\text{count}}(\llbracket \delta \rrbracket)$$
Parity conditions
Alternating parity tree automata

MSO discriminates inductive from coinductive behaviour.

Typical properties:

“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.
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 $\varphi$:

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

We disclose that coloring is a modality – or a coeffect. 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.
Coloring and rewriting

The semantics of a finite term of type $o$ characterizes the colors of its finite branches.

This extends to higher-order.

**Colored** fixpoint operator: compose denotations in a winning way – inductively or coinductively, according to the coloring coeffect.

This operator has good properties (Conway operator).

New interpretation $\llbracket t \rrbracket_{col}$. 
Theorem

The infinitary normal form of a $\lambda Y$-term

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

is accepted by the parity automaton

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

from the set of states

$$\llbracket t \rrbracket_{col} \circ M_{col}(\llbracket \delta \rrbracket)$$
Ehrhard 2012: the finitary modal $ScottL$ is the extensional collapse of $Rel$.

Two essential differences:

- $\llbracket ! A \rrbracket = \mathcal{P}_{\text{fin}}(A)$
- necessity of “subtyping”

We adapted to $ScottL$ the theoretical approach of this work.

**Corollary**

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

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.

Thank you for your attention!