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 φ in an appropriate logic
- Make them interact: the result is whether

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

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

$$\varphi \quad \mapsto \quad \mathcal{A}_{\varphi}$$

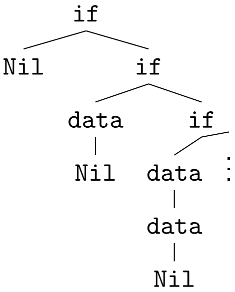
Model-checking higher-order programs

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

Example:

$$\begin{array}{rcl} \texttt{Main} & = & \texttt{Listen Nil} \\ \texttt{Listen } x & = & \texttt{if } \textit{end} \texttt{ then } x \texttt{ else Listen (data } x\texttt{)} \end{array}$$

modelled as



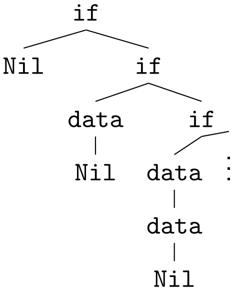
Model-checking higher-order programs

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

Example:

$$\begin{array}{rcl} \texttt{Main} & = & \texttt{Listen Nil} \\ \texttt{Listen } x & = & \texttt{if } \textit{end} \texttt{ then } x \texttt{ else Listen (data } x\texttt{)} \end{array}$$

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) $\mathcal{A}_\varphi$

corresponding to a

monadic second-order logic (MSO) formula φ.

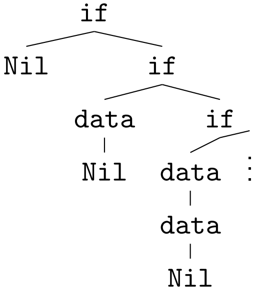
(safety, liveness properties, etc)

Can we decide whether a higher-order tree satisfies a MSO formula?

Trees vs. tree automata

Trees and types

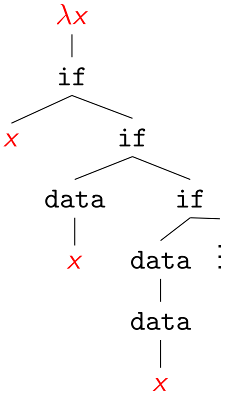
Three actions here: $\Sigma = \{\, \texttt{if} : 2, \texttt{data} : 1, \texttt{Nil} : 0 \,\}$.



$: \quad o$

Ground type: o is the type of trees
(and more generally of terms over Σ reducing to a tree).

Trees and types

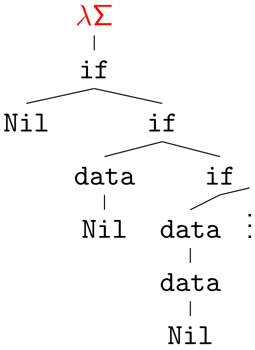


$: \quad o \rightarrow o$

Applying it to `Nil` gives the previous tree.

Trees and types

Church encoding of trees:



$: \quad o(\Sigma) \to o$

where "$\lambda\Sigma$" stands for $\lambda \texttt{if}.\, \lambda \texttt{data}.\, \lambda \texttt{Nil}.$, and

$$o(\Sigma) \to o \quad = \quad (o \to o \to o) \to (o \to o) \to o \to o$$

Linear decomposition of the intuitionnistic arrow

In linear logic,

$$A \rightarrow B \quad = \quad !A \multimap B$$

$!A$ allows to duplicate or to drop A

$\multimap$ uses linearly (once) each copy

Linear decomposition of the intuitionnistic arrow

$$(o \to o \to o) \to (o \to o) \to o \to o$$

translates as

$$!\,(!\,o \multimap !\,o \multimap o) \multimap !\,(!\,o \multimap o) \multimap !\,o \multimap o$$

In the relational semantics of linear logic, with $[\![o]\!] \;=\; Q$,

$$[\![!\,A]\!] \;=\; \mathcal{M}_{fin}([\![A]\!]) \quad \text{and} \quad [\![A \multimap B]\!] \;=\; [\![A]\!] \times [\![B]\!]$$

For instance,

$$[\![o \to o \to o]\!] \;=\; \mathcal{M}_{fin}(Q) \times \mathcal{M}_{fin}(Q) \times Q$$

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

$$!\,(!\,o \multimap !\,o \multimap o) \multimap !\,(!\,o \multimap o) \multimap !\,o \multimap o$$

Complain: where is model-checking?

We mentioned alternating parity tree automata...

Alternating parity tree automata

For a MSO formula φ,

$$\langle \mathcal{G} \rangle \quad \vDash \quad \varphi$$

iff an equivalent APT $\mathcal{A}_\varphi$ has a run over $\langle \mathcal{G} \rangle$.

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

Alternating tree automata

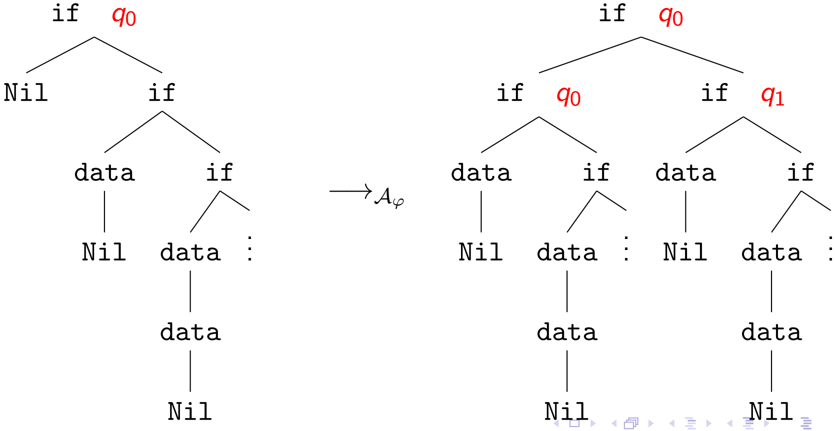
ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

Typically: $\delta(q_0, \texttt{if}) \; = \; (2, q_0) \wedge (2, q_1)$.

Alternating tree automata

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

Typically: $\delta(q_0, \texttt{if}) \;=\; (2, q_0) \wedge (2, q_1)$.



Alternating tree automata

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

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

In fact, `if` has the linear type

$$\mathtt{if} \ : !\, o \multimap !\, o \multimap \ o$$

so that in the relational semantics of linear logic

$$([], [q_0, q_1], q_0) \ \in \ [\![\mathtt{if}]\!] \ \subseteq \ \mathcal{M}_{fin}(Q) \times \mathcal{M}_{fin}(Q) \times Q$$

Model-checking I

An alternating tree automaton over Σ, with set of states Q, of transition function δ, provides

$$[\![\delta]\!] \quad = \quad [\![\texttt{if}]\!] \uplus [\![\texttt{data}]\!] \uplus [\![\texttt{Nil}]\!] \quad \subseteq \quad [\![o(\Sigma)]\!]$$

while a tree t over Σ gives, under Church encoding:

$$[\![t]\!] \quad \subseteq \quad [\![o(\Sigma) \rightarrow o]\!] \quad = \quad \mathcal{M}_{fin}([\![o(\Sigma)]\!]) \times Q$$

Relational composition:

$$[\![t]\!] \circ \mathcal{M}_{fin}([\![\delta]\!]) \quad \subseteq \quad Q$$

Interactive interpretation?

Model-checking I

An alternating tree automaton over Σ, with set of states Q, of transition function δ, provides

$$[\![\delta]\!] \quad = \quad [\![\mathtt{if}]\!] \uplus [\![\mathtt{data}]\!] \uplus [\![\mathtt{Nil}]\!] \quad \subseteq \quad [\![o(\Sigma)]\!]$$

while a tree t over Σ gives, under Church encoding:

$$[\![t]\!] \quad \subseteq \quad [\![o(\Sigma) \rightarrow o]\!] \quad = \quad \mathcal{M}_{fin}([\![o(\Sigma)]\!]) \times Q$$

Relational composition:

$$[\![t]\!] \circ \mathcal{M}_{fin}([\![\delta]\!]) \quad \subseteq \quad Q$$

Interactive interpretation?

Model-checking I

Relational composition:

$$[\![t]\!] \circ \mathcal{M}_{fin}([\![\delta]\!]) \quad \subseteq \quad Q$$

Proposition

$$[\![t]\!] \circ \mathcal{M}_{fin}([\![\delta]\!])$$

is the set of states q from which

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

accepts the tree t.

Model-checking I

Rel is a denotational model:

$$t \;\rightarrow_\beta\; t' \qquad \Longrightarrow \qquad [\![t]\!] \;=\; [\![t']\!]$$

Corollary

For a term

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

the set of states q from which

$$\mathcal{A} \;=\; \langle\, \Sigma,\; Q,\; \delta \,\rangle$$

accepts the tree generated by the normalization of t is

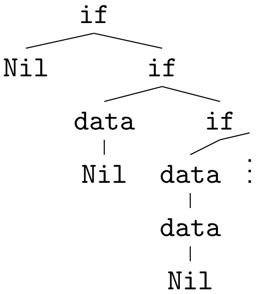
$$[\![t]\!] \;\circ\; \mathcal{M}_{fin}([\![\delta]\!])$$

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

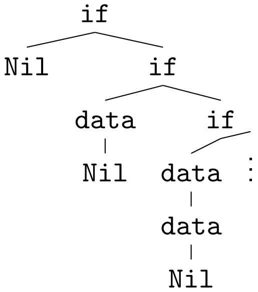


- and to account for parity conditions.

Higher-order recursion schemes

Some regularity for infinite trees

Higher-order recursion schemes



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

$$\mathcal{G} \;=\; \begin{cases} \mathtt{S} & = \quad \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L}\ x & = \quad \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$

Higher-order recursion schemes

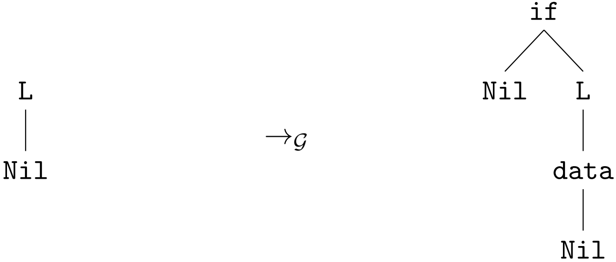
$$\mathcal{G} \;=\; \left\{ \begin{array}{lcl} \texttt{S} & = & \texttt{L Nil} \\ \texttt{L } x & = & \texttt{if } x \,(\texttt{L (data } x\texttt{))} \end{array} \right.$$

Rewriting starts from the start symbol S:

$$\texttt{S} \qquad \rightarrow_{\mathcal{G}} \qquad \begin{array}{c} \texttt{L} \\ | \\ \texttt{Nil} \end{array}$$

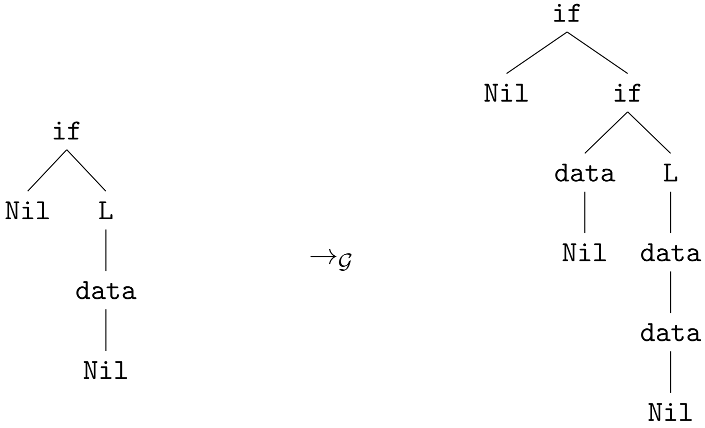
Higher-order recursion schemes

$$\mathcal{G} \quad = \quad \begin{cases} \mathtt{S} & = \quad \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L}\ x & = \quad \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$



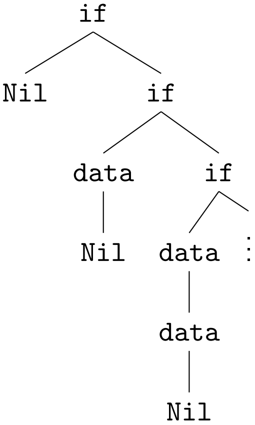
Higher-order recursion schemes

$$\mathcal{G} = \begin{cases} \mathtt{S} & = \quad \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L}\ x & = \quad \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$



Higher-order recursion schemes

$$\mathcal{G} = \begin{cases} \mathtt{S} & = \mathtt{L\ Nil} \\ \mathtt{L}\ x & = \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$



$\langle \mathcal{G} \rangle$ is an infinite non-regular tree.

It is our model $\mathcal{M}$.

Higher-order recursion schemes

$$\mathcal{G} = \left\{ \begin{array}{lcl} \mathtt{S} & = & \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L}\ x & = & \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{array} \right.$$

"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

$$\mathcal{G} \;=\; \begin{cases} \mathtt{S} & = \quad \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L}\ x & = \quad \mathtt{if}\ x\,(\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$

"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).

HORS can alternatively be seen as simply-typed λ-terms with

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

$\rightarrow$ add fixpoints to the model.

Model-checking II

Finite iteration $\rightarrow$ inductive fixpoint operator on *Rel*.

Theorem

The infinitary normal form of a λY-term

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

is accepted by

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

from the set of states

$$[\![t]\!] \ \circ \ \mathcal{M}_{fin}([\![\delta]\!])$$

Model-checking II

Finite iteration $\rightarrow$ inductive fixpoint operator on *Rel*.

Theorem

The infinitary normal form of a λY*-term*

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

is accepted by

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

from the set of states

$$[\![t]\!] \ \circ \ \mathcal{M}_{fin}([\![\delta]\!])$$

after a finite execution of the automaton.

On finiteness

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

Defining a new exponential

$$\lightning \; : \; A \mapsto \mathcal{M}_{count}(A)$$

gives a relational model of linear logic with a

coinductive fixpoint operator

(infinite fixpoint unfolding).

New interpretation of terms: $[\![\, t \,]\!]_{gfp}$.

Model-checking III

Theorem

The infinitary normal form of a λY*-term*

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

is accepted by

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

from the set of states

$$[\![\, t \,]\!]_{gfp} \ \circ \ \mathcal{M}_{count}([\![\delta]\!])$$

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

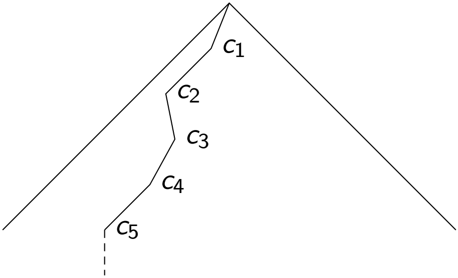
"after a `read` operation, a `write` eventually occurs".

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 occuring 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 occuring infinitely often along it is even.

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

For a MSO formula φ:

$$\mathcal{A}_\varphi \text{ has a winning run-tree over } \langle \mathcal{G} \rangle \text{ iff } \langle \mathcal{G} \rangle \ \vDash \ \phi$$

The coloring comonad

We disclose that coloring is a modality – or a coeffect.
It defines a comonad in the semantics:

$$\square\, A \quad = \quad Col \times A$$

which can be composed with $\lightning$, giving an infinitary, colored model of linear logic in which

$$\delta(q_0, \mathtt{if}) \;=\; (2, q_0) \wedge (2, q_1)$$

corresponds to

$$([],\ [(\Omega(q_0),\ q_0), (\Omega(q_1),\ q_1)],\ q_0) \quad \in \quad [\![\mathtt{if}]\!]$$

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 $[\![\, t \,]\!]_{col}$.

Model-checking IV

Theorem

The infinitary normal form of a λY-term

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

is accepted by the parity automaton

$$\mathcal{A} \ = \ \langle\, \Sigma,\ Q,\ \delta,\ \Omega\, \rangle$$

from the set of states

$$[\![\, t \,]\!]_{col} \ \circ \ \mathcal{M}_{col}([\![\delta]\!])$$

Model-checking V

Ehrhard 2012: the finitary modal *ScottL* is the extensional collapse of *Rel*.

Two essential differences:

- $[\![!\, A]\!] \;=\; \mathcal{P}_{fin}(A)$
- 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!