# Semantics of linear logic and higher-order model-checking

Charles Grellois    Paul-André Melliès

IRIF — Université Paris 7
FOCUS Team – INRIA & University of Bologna

GDRI-LL Meeting – University of Bologna
February 2, 2016

# 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
- Interaction: the result is whether

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

Typically: translate $\varphi$ to an equivalent automaton running over $\mathcal{M}$:

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

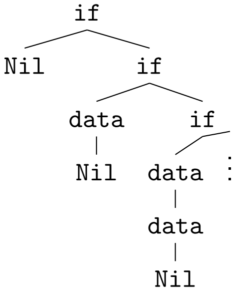
# Model-checking higher-order programs

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

Example:

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

modelled as



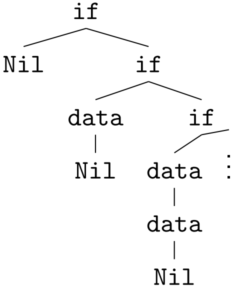
# Model-checking higher-order programs

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

over which we run

an alternating parity tree automaton (APT) $\mathcal{A}_\varphi$

corresponding to a

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

(safety, liveness properties, etc)

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

# Higher-order recursion schemes

Some regularity for infinite trees

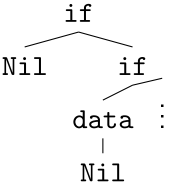
# Higher-order recursion schemes

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

is abstracted as

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

which produces (how ?) the higher-order tree of actions



# Higher-order recursion schemes

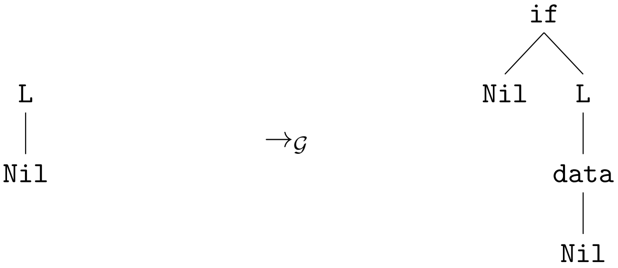
$$\mathcal{G} \;=\; \begin{cases} \texttt{S} & = \;\; \texttt{L Nil} \\ \texttt{L}\ x & = \;\; \texttt{if}\ x\ (\texttt{L}\ (\texttt{data}\ x\ )\ ) \end{cases}$$

Rewriting starts from the start symbol $\texttt{S}$:

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

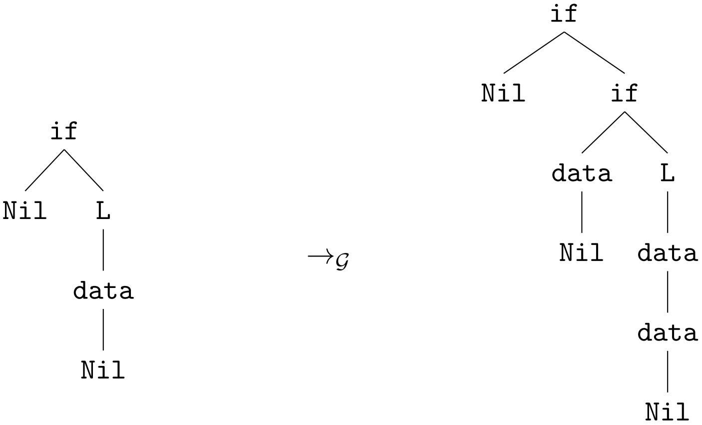
# Higher-order recursion schemes

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



# Higher-order recursion schemes

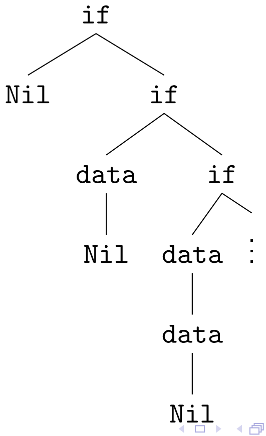
$$\mathcal{G} = \begin{cases} \mathtt{S} & = \quad \mathtt{L\ Nil} \\ \mathtt{L}\ x & = \quad \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$



# Higher-order recursion schemes

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

$\langle \mathcal{G} \rangle \quad =$



# Higher-order recursion schemes

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

"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 $\Omega$ in one step).

# Higher-order recursion schemes

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

"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 $\Omega$ in one step).

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

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

# Alternating parity tree automata

# Alternating parity tree automata

For a MSO formula $\varphi$,

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

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

APT $=$ alternating tree automata (ATA) $+$ parity condition.

# Alternating tree automata

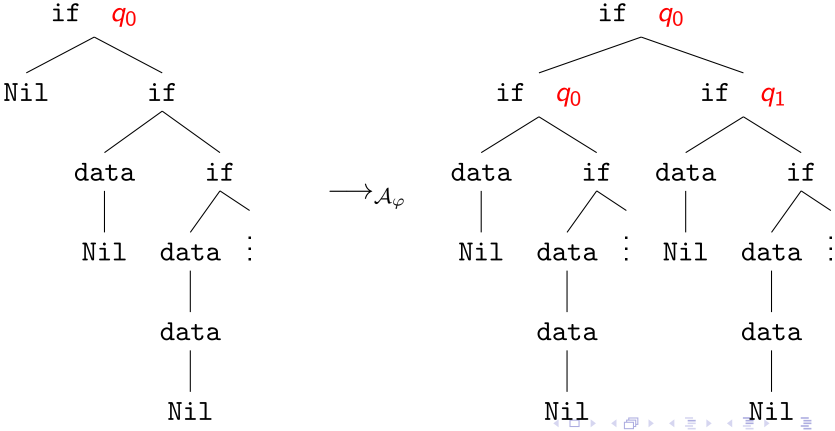
ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

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

# Alternating tree automata

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

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



# Alternating parity tree automata

MSO discriminates inductive from coinductive behaviour.

This allows to express properties as

> "a given operation is executed infinitely often in some execution"

or

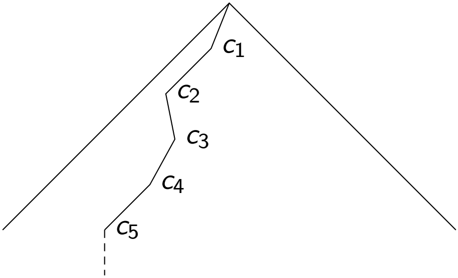
> "after a `read` operation, a `write` eventually occurs".

# 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 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 $\varphi$:

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

# Recognition by homomorphism

# Automata and recognition

For the usual finite automata on words: given a regular language $L \subseteq A^*$,

there exists a finite automaton $\mathcal{A}$ recognizing $L$

if and only if

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

Roughly speaking: there exists a finite algebraic structure in which the language is interpreted.

# Automata and recognition

Let's extend this to:

- higher-order recursion schemes
- alternating parity automata

using domains (Aehlig 2006, Salvati 2009).

How to model...

- Alternation?
- Recursion?
- Parity condition?

# Intersection types and alternation

# Alternating tree automata and intersection types

A key remark (Kobayashi 2009):

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

can be seen as the intersection typing

$$\texttt{if} \ : \ \emptyset \rightarrow (q_0 \wedge q_1) \rightarrow q_0$$

refining the simple typing

$$\texttt{if} \ : \ o \rightarrow o \rightarrow o$$

(this talk is NOT about filter models!)

# Alternating tree automata and intersection types

In a derivation typing $\texttt{if}\ \ T_1\ \ T_2$ :

$$
\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \texttt{if} : \emptyset \to (q_0 \wedge q_1) \to q_0}\delta \qquad \color{red}{\emptyset}}{\emptyset \vdash \texttt{if}\ T_1 : (q_0 \wedge q_1) \to q_0}\text{App} \qquad \dfrac{\vdots}{\color{red}{\Gamma_{21} \vdash T_2 : q_0}} \qquad \dfrac{\vdots}{\color{red}{\Gamma_{22} \vdash T_2 : q_1}}}{\Gamma_{21}, \Gamma_{22} \vdash \texttt{if}\ T_1\ T_2 : q_0}\text{App}
$$

Intersection types naturally lift to higher-order – and thus to $\mathcal{G}$, which finitely represents $\langle \mathcal{G} \rangle$.

**Theorem (Kobayashi)**

$S : q_0 \vdash S : q_0$ *iff* *the ATA $\mathcal{A}_\varphi$ has a run-tree over $\langle \mathcal{G} \rangle$.*

# A closer look at the Application rule

$$\text{App} \qquad \frac{\Delta \vdash t : (\ \theta_1 \ \wedge \cdots \wedge \ \theta_k) \rightarrow \theta \ :: \ \kappa \rightarrow \kappa' \qquad \Delta_i \vdash u \ : \ \theta_i \ :: \ \kappa}{\Delta\, ,\, \Delta_1\, ,\, \ldots\, ,\, \Delta_k \ \vdash \ t\, u \, : \, \theta :: \kappa'}$$

Towards sequent calculus:

$$\dfrac{\Delta \ \vdash \ t \ : \ (\ \bigwedge_{i=1}^{n} \ \theta_i\,) \rightarrow \theta' \qquad \dfrac{\Delta_i \ \vdash \ u \ : \ \theta_i \qquad \forall i \in \{1, \ldots\ n\}}{\Delta_1, \ldots, \Delta_n \ \vdash \ u \ : \ \bigwedge_{i=1}^{n} \ \theta_i} \ \text{Right} \bigwedge}{\Delta, \ \Delta_1, \ldots, \ \Delta_n \ \vdash \ t \ u \ : \ \theta'}$$

# A closer look at the Application rule

$$\dfrac{\Delta \ \vdash \ t \, : \, (\, \bigwedge_{i=1}^{n} \ \theta_i \,) \to \theta' \qquad \dfrac{\Delta_i \ \vdash \ u \, : \, \theta_i \qquad \forall i \in \{1, \ldots n\}}{\Delta_1, \ldots, \Delta_n \ \vdash \ u \, : \, \bigwedge_{i=1}^{n} \ \theta_i} \ \text{Right} \bigwedge}{\Delta, \, \Delta_1, \ldots, \Delta_n \ \vdash \ t \ u \, : \, \theta'}$$

Linear decomposition of the intuitionnistic arrow:

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

Two steps: duplication / erasure, then linear use.

Right $\bigwedge$ corresponds to the Promotion rule of indexed linear logic.

# Intersection types and semantics of linear logic

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

Two interpretations of the exponential modality:

Qualitative models
(Scott semantics)

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

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

$$\{q_0,\, q_0,\, q_1\} \;\; = \;\; \{q_0,\, q_1\}$$

Order closure

Quantitative models
(Relational semantics)

$$!\, A \;\; = \;\; \mathcal{M}_{fin}(A)$$

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

$$[q_0,\, q_0,\, q_1] \;\; \neq \;\; [q_0,\, q_1]$$

Unbounded multiplicities

# Intersection types and semantics of linear logic

$$\begin{tikzcd}
Rel_! \arrow[rr, leftrightarrow, "Bucciareli-Ehrhard", "de\ \ Carvalho"'] \arrow[d, "Ehrhard"'] & & \text{Non-idempotent types} \arrow[d, "Ehrhard"] \\
ScottL_! \arrow[rr, leftrightarrow, "Ehrhard", "Terui"'] & & \text{Idempotent types}
\end{tikzcd}$$

Fundamental idea:

$$[\![t]\!] \cong \{\theta \mid \emptyset \vdash t : \theta\}$$

and similarly for open terms.

# Intersection types and semantics of linear logic

$$\begin{tikzcd}
Rel_! \arrow[rr, leftrightarrow, "Bucciareli-Ehrhard", "de\ \ Carvalho"'] \arrow[d, "Ehrhard"'] & & \text{Non-idempotent types} \arrow[d, "Ehrhard"] \\
ScottL_! \arrow[rr, leftrightarrow, "Ehrhard", "Terui"'] & & \text{Idempotent types}
\end{tikzcd}$$

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

$$\mathcal{A} \text{ accepts } \langle t \rangle \text{ from } q \quad \Leftrightarrow \quad q \in [\![ t ]\!] \quad \Leftrightarrow \quad \emptyset \vdash t : q :: o$$

Extension with recursion and parity condition?

# Adding parity conditions to the type system

# Alternating parity tree automata

We add coloring annotations to intersection types:

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

now corresponds to

$$\texttt{if} \;:\; \emptyset \rightarrow \left( \Box_{\Omega(q_0)}\, q_0 \wedge \Box_{\Omega(q_1)}\, q_1 \right) \rightarrow q_0$$

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

$$\begin{array}{c} \texttt{if} \\ [\,]_{q_0} \quad [\,]_{q_1} \end{array}$$

A new neutral color: $\epsilon$ for an empty run-tree context $[\,]_q$.

# A type-system for verification

A colored Application rule:

$$\text{App} \quad \frac{\Delta \vdash t : (\square_{c_1}\ \theta_1\ \wedge \cdots \wedge \square_{c_k}\ \theta_k) \rightarrow \theta\ ::\ \kappa \rightarrow \kappa' \qquad \Delta_i \vdash u\ :\ \theta_i\ ::\ \kappa}{\Delta\ +\ \square_{c_1}\Delta_1\ +\ \ldots\ +\ \square_{c_k}\Delta_k\ \ \vdash\ \ t\, u\ :\ \theta :: \kappa'}$$

# A type-system for verification

We rephrase the parity condition to typing trees, and now capture all MSO:

**Theorem (G.-Melliès 2014)**

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

We obtain decidability by considering idempotent types.

Non-idempotency is very helpful for proofs, but leads to infinitary constructions.

# Colored models of linear logic

# A closer look at the Application rule

$$\frac{\Delta \vdash t : (\square_{m_1}\ \theta_1\ \wedge \cdots \wedge \square_{m_k}\ \theta_k) \to \theta\ ::\ \kappa \to \kappa' \qquad \Delta_i \vdash u\ :\ \theta_i\ ::\ \kappa}{\Delta\ +\ \square_{m_1}\Delta_1\ +\ \ldots\ +\ \square_{m_k}\Delta_k\ \ \vdash\ \ t\,u\ :\ \theta :: \kappa'}$$

Towards sequent calculus:

$$\frac{\Delta \vdash t : (\bigwedge_{i=1}^{n}\ \square_{m_i}\ \theta_i) \to \theta \qquad \dfrac{\dfrac{\Delta_1 \vdash u\ :\ \theta_1}{\square_{m_1}\ \Delta_1\ \vdash\ u\ :\ \square_{m_1}\ \theta_1} \quad \ldots \quad \dfrac{\Delta_n \vdash u\ :\ \theta_n}{\square_{m_n}\ \Delta_n\ \vdash\ u\ :\ \square_{m_n}\ \theta_1}\ \text{Right}\ \square}{\square_{m_1}\Delta_1,\ \ldots,\ \square_{m_n}\Delta_n\ \vdash\ u\ :\ \bigwedge_{i=1}^{n}\ \square_{m_i}\ \theta_i}\ \text{Right}\ \bigwedge}{\Delta,\ \square_{m_1}\Delta_1,\ \ldots,\ \square_{m_n}\Delta_n\ \ \vdash\ \ t\,u\ :\ \theta}$$

Right $\square$ looks like a promotion. In linear logic:

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

Our reformulation of the Kobayashi-Ong type system shows that $\square$ is a modality (in the sense of S4) which distributes with the exponential in the semantics.

# Colored semantics

We extend:

- *Rel* with countable multiplicites, coloring and an inductive-coinductive fixpoint
- *ScottL* with coloring and an inductive-coinductive fixpoint.

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*.

# Model-checking and finitary semantics

Let $\mathcal{G}$ be a HORS representing the tree $\langle \mathcal{G} \rangle$ and $\mathcal{A}$ be an alternating parity automaton.

Conjecture in infinitary *Rel*, but theorem in colored *ScottL*:

$$\mathcal{A} \text{ accepts } \langle \mathcal{G} \rangle \text{ from } q \quad \Leftrightarrow \quad q \in [\![ t ]\!]$$

A similar theorem holds for a companion intersection type system to colored ScottL. Since the semantics are finitary:

**Corollary**

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

Thank you for your attention!

# Model-checking and finitary semantics

Let $\mathcal{G}$ be a HORS representing the tree $\langle \mathcal{G} \rangle$ and $\mathcal{A}$ be an alternating parity automaton.

Conjecture in infinitary *Rel*, but theorem in colored *ScottL*:

$$\mathcal{A} \text{ accepts } \langle \mathcal{G} \rangle \text{ from } q \quad \Leftrightarrow \quad q \in [\![ t ]\!]$$

A similar theorem holds for a companion intersection type system to colored ScottL. Since the semantics are finitary:

**Corollary**

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

Thank you for your attention!