Semantics of linear logic and higher-order model-checking

Charles Grellois   Paul-André Melliès

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

NII Shonan Meeting on HOMC
March 14, 2016

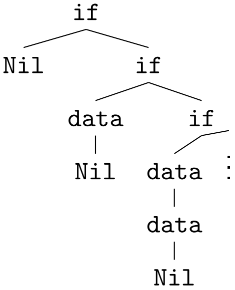
Model-checking higher-order programs

For higher-order programs with recursion, the model $\mathcal{M}$ of interest 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



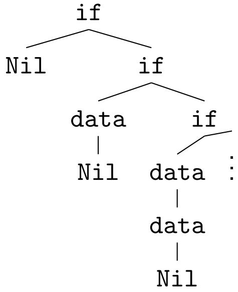
Model-checking higher-order programs

For higher-order programs with recursion, the model $\mathcal{M}$ of interest 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, the model $\mathcal{M}$ of interest 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 φ.

(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

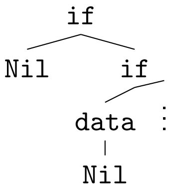
Higher-order recursion schemes

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

is abstracted as

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

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



Higher-order recursion schemes

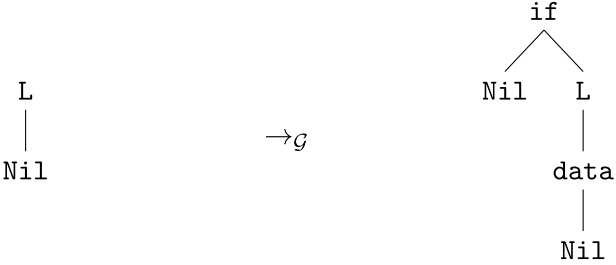
$$\mathcal{G} \;=\; \begin{cases} \mathtt{S} & = \;\; \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L}\ x & = \;\; \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$

Rewriting starts from the start symbol S:

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

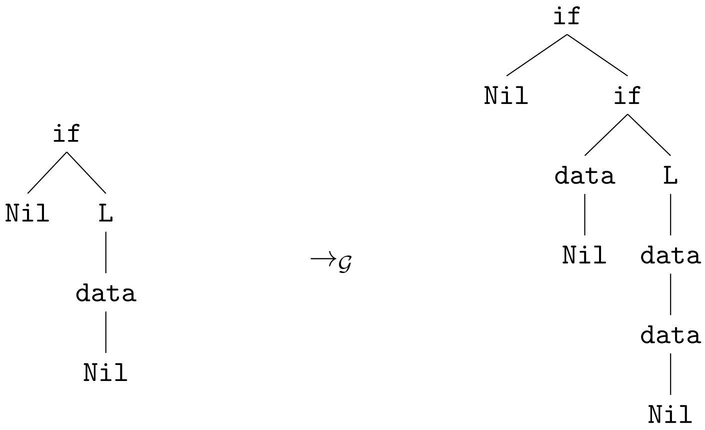
Higher-order recursion schemes

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



Higher-order recursion schemes

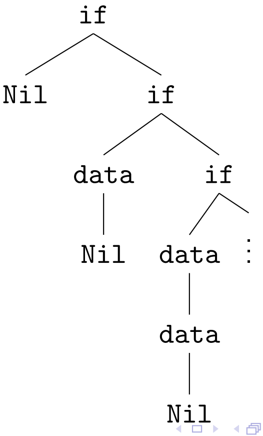
$$\mathcal{G} \;=\; \begin{cases} \mathtt{S} & = \quad \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L}\ x & = \quad \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$



Higher-order recursion schemes

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

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



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

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 $\langle \mathcal{G} \rangle$.

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

Alternating tree automata

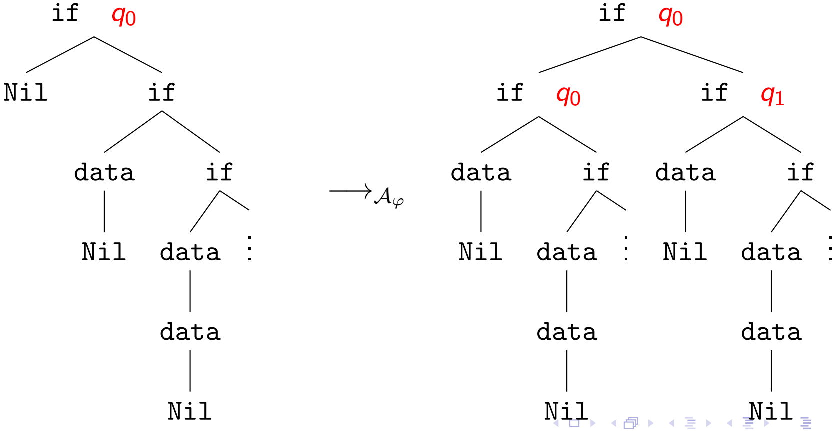
ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

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



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

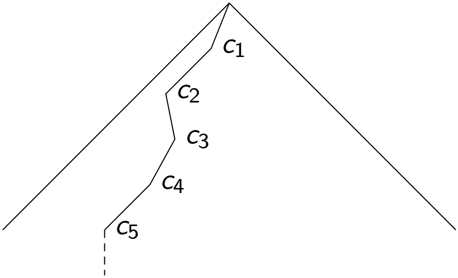
"after a `read` operation, a `write` eventually occurs".

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

$$\Omega(q) \in \mathit{Col} \subseteq \mathbb{N}$$

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:

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

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 type-system for verification: without parity conditions

Axiom $\dfrac{}{x \,:\, \bigwedge_{\{i\}} \;\theta_i \;::\; \kappa \;\vdash\; x \,:\, \theta_i \;::\; \kappa}$

$$\delta \qquad \frac{\{\,(i, q_{ij}) \mid 1 \leq i \leq n, 1 \leq j \leq k_i\,\} \quad \text{satisfies} \quad \delta_A(q, a)}{\emptyset \vdash a \,:\, \bigwedge_{j=1}^{k_1} \; q_{1j} \;\rightarrow\; \ldots \;\rightarrow\; \bigwedge_{j=1}^{k_n} \; q_{nj} \rightarrow q \;::\; o \rightarrow \cdots \rightarrow o}$$

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

$$\lambda \qquad \frac{\Delta\,,\,x \,:\, \bigwedge_{i \in I} \;\theta_i \;::\; \kappa \;\vdash\; t \,:\, \theta \;::\; \kappa'}{\Delta \;\vdash\; \lambda x\,.\,t \,:\, \left(\bigwedge_{i \in I} \;\theta_i\right) \rightarrow \theta :: \kappa \rightarrow \kappa'}$$

$$\mathit{fix} \qquad \frac{\Gamma \vdash \mathcal{R}(F) \,:\, \theta \;::\; \kappa}{\overline{F \,:\; \theta \;::\; \kappa \;\vdash\; F \,:\, \theta \;::\; \kappa}}$$

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

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

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)

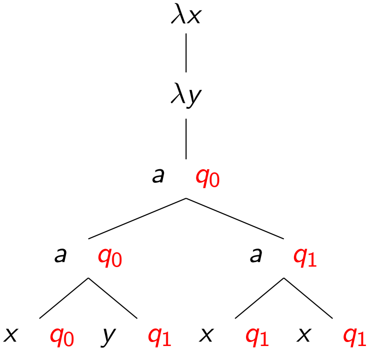
$$!\,A \;\; = \;\; \mathcal{M}_{fin}(A)$$

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

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

Unbounded multiplicities

An example of interpretation



In *Rel*, one denotation:

$$([q_0, q_1, q_1],\ [q_1],\ q_0)$$

In *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

$$\begin{tikzcd}[column sep=huge, row sep=large]
Rel_! \arrow[r, leftrightarrow, "Bucciareli-Ehrhard", "de\ \ Carvalho"'] \arrow[d, "Ehrhard"'] & \text{Non-idempotent types} \arrow[d, "Ehrhard"] \\
ScottL_! \arrow[r, leftrightarrow, "Ehrhard", "Terui"'] & \text{Idempotent types}
\end{tikzcd}$$

Fundamental idea:

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

for a closed term.

Intersection types and semantics of linear logic

$$
\begin{tikzcd}
\mathit{Rel}_! \arrow[r, leftrightarrow, "\mathit{Bucciareli-Ehrhard}", "\mathit{de\ Carvalho}"'] \arrow[d, "\mathit{Ehrhard}"'] & \text{Non-idempotent types} \arrow[d, "\mathit{Ehrhard}"] \\
\mathit{ScottL}_! \arrow[r, leftrightarrow, "\mathit{Ehrhard}", "\mathit{Terui}"'] & \text{Idempotent types}
\end{tikzcd}
$$

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:

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

now corresponds to

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

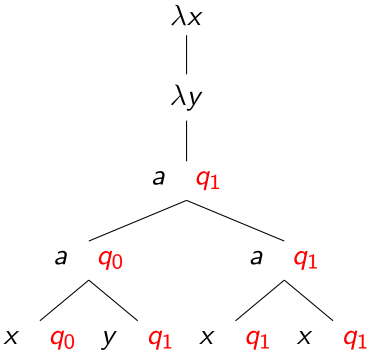
Idea: `if` is a run-tree with two holes:

$$\begin{array}{ccc} & \mathtt{if} & \\ [\,]_{q_0} & & [\,]_{q_1} \end{array}$$

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

An example of colored intersection type

Set $\Omega(q_i) = i$.



has type

$$\Box_0\, q_0 \wedge \Box_1\, q_1 \rightarrow \Box_1\, q_1 \rightarrow q_1$$

Note the color 0 on q_0...

A type-system for verification (Grellois-Melliès 2014)

Axiom $\dfrac{}{x \,:\, \bigwedge_{\{i\}} \square_{\epsilon}\ \theta_i \,::\, \kappa \ \vdash\ x \,:\, \theta_i \,::\, \kappa}$

δ $\dfrac{\{\,(i, q_{ij}) \mid 1 \leq i \leq n, 1 \leq j \leq k_i\,\} \quad \text{satisfies} \quad \delta_A(q, a)}{\emptyset \vdash a \,:\, \bigwedge_{j=1}^{k_1} \square_{\Omega(q_{1j})}\ q_{1j} \,\rightarrow\, \ldots \,\rightarrow\, \bigwedge_{j=1}^{k_n} \square_{\Omega(q_{nj})}\ q_{nj} \rightarrow q \,::\, o \rightarrow \cdots \rightarrow o \rightarrow o}$

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

fix $\dfrac{\Gamma \vdash \mathcal{R}(F) \,:\, \theta \,::\, \kappa}{F \,:\, \square_{\epsilon}\ \theta \,::\, \kappa \ \vdash\ F \,:\, \theta \,::\, \kappa}$

λ $\dfrac{\Delta\,, x \,:\, \bigwedge_{i \in I} \square_{m_i}\ \theta_i \,::\, \kappa \ \vdash\ t \,:\, \theta \,::\, \kappa'}{\Delta \ \vdash\ \lambda x\,.\,t \,:\, \left(\bigwedge_{i \in I} \square_{m_i}\ \theta_i\right) \rightarrow \theta :: \kappa \rightarrow \kappa'}$

A type-system for verification

A colored Application rule:

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

inducing a winning condition on infinite proofs: the node

$$\Delta_i \vdash u\ :\ \theta_i\ ::\ \kappa$$

has color c_i, others have color ϵ, and we use the parity condition.

A type-system for verification

We now capture all MSO:

Theorem (G.-Melliès 2014, from Kobayashi-Ong 2009)

$S : q_0 \vdash S : q_0$ 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

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

Towards sequent calculus:

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

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

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

Our reformulation of the Kobayashi-Ong type system shows that $\Box$ 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*.

Infinitary relational semantics

Extension of *Rel* with infinite multiplicities:

$$\lightning\, A \;=\; \mathcal{M}_{count}(A)$$

and coloring modality

$$\Box\, A \;=\; Col \times A$$

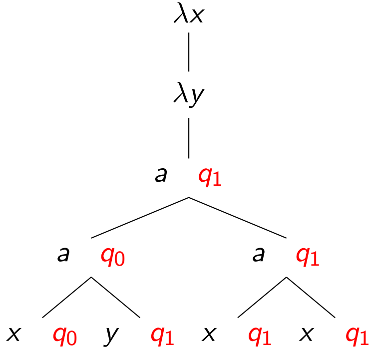
Distributive law:

$$\begin{array}{rcl} \lambda_A & = & :\ \lightning\,\Box\, A \rightarrow \Box\,\lightning\, A \\ & & \{([(c,a_1),\,(c,a_2),\ldots],(c,\,[a_1,\,a_2,\ldots]))\ \mid a_i \in A,\ c \in Col\} \end{array}$$

Allows to compose comonads: $\boldsymbol{\lightning} \;=\; \lightning\,\Box$ is an exponential in the infinitary relational semantics.
This induces a colored CCC $Rel_{\lightning}$ ($\rightarrow$ model of the λ-calculus).

An example of interpretation

Set $\Omega(q_i) \;=\; i$.



has denotation

$$([(0, q_0), (1, q_1), (1, q_1)], [(1, q_1)], q_1)$$

(corresponding to the type $\Box_0\, q_0 \wedge \Box_1\, q_1 \rightarrow \Box_1\, q_1 \rightarrow q_1$)

An inductive-coinductive fixpoint operator

$\mathbf{Y}$ transports

$$f \quad : \quad \lightning X \otimes \lightning A \ \multimap \ A$$

into

$$\mathbf{Y}_{X,A}\,(f) \quad : \quad \lightning X \ \multimap \ A.$$

via

$$\mathbf{Y}_{X,A}\,(f) \quad = \quad \{\, (w, a) \,|\, \exists witness \in \textbf{run-tree}(f, a) \text{ with } w = \textbf{leaves}(witness) \text{ and } witness \text{ is accepting} \,\}$$

An inductive-coinductive fixpoint operator

$$\mathbf{Y}_{X,A}(f) \;=\; \{\, (w, a) \mid \exists witness \in \textbf{run-tree}(f, a) \text{ with } w = \textbf{leaves}(witness) \text{ and } witness \text{ is accepting} \,\}$$

witness is built from finite pieces

$$(c, a')$$

$$(c'_1, x_1) \quad (c'_2, x_2) \quad \cdots \quad (c_1, a_1) \quad (c_2, a_2) \quad \cdots$$

where

$$\left(\left([(c'_1, x_1), (c'_2, x_2), \ldots], [(c_1, a_1), (c_2, a_2), \ldots] \right), a' \right) \in f$$

leaves(*witness*) is the colored multiset of the parameter leaves of *witness*.

$\mathbf{Y}$ is a Conway operator, and $Rel_{\lightning}$ is a model of the λY-calculus.

Model-checking and infinitary semantics

Conjecture

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

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

where $\lambda(\mathcal{G})$ is a λY-term corresponding to $\mathcal{G}$.

Using Church encoding, we can also design an interpretation independent of the automaton of interest.

Finitary semantics

In ScottL, we define $\Box$, λ and $\mathbf{Y}$ similarly (using downward-closures).

$ScottL_{\not\downarrow}$ is a model of the λY-calculus.

Theorem

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

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

Corollary

The higher-order model-checking problem is decidable.

Elements of proof

The proof proceeds by relating

- *ScottL*$_{\lightning}$ to an intersection type system $\mathcal{S}_{\mathit{fix}}(\mathcal{A})$ extending Terui's with recursion and parity conditions,
- $\mathcal{S}_{\mathit{fix}}(\mathcal{A})$ and the colored intersection type system presented earlier, on η-long β-normal forms
- and by using our modified version of the soundness-and-completeness theorem of Kobayashi and Ong.

Decidability

Checking whether $q_0 \in [\![\mathcal{G}]\!] \iff$ solving a parity game on a finite fragment of $ScottL_{\lightning}$, which is decidable.

We also obtain memoryless strategies which correspond to regular typings in $\mathcal{S}_{fix}(\mathcal{A})$:

$$\dfrac{\begin{array}{c}\pi_1 \\ F_1 : \left\{(c_{j_1}, \beta_{j_1}) \mid j_1 \in J_1\right\}, \ldots \vdash \mathcal{R}(F_i) : \alpha\end{array} \qquad \begin{array}{c}\emptyset \vdash \mu F_{k''} : \beta''_{j''_{k''}} \\ \vdots \\ \emptyset \vdash \mu F_k : \beta_{j_k}\end{array} \qquad \begin{array}{c}\vdots \\ \emptyset \vdash \mu F_{k'} : \beta'_{j'_{k'}}\end{array}}{\emptyset \vdash \mu F_1 : q}$$

A key to the selection property.

Conclusion

- Connections between intersection types and linear logic
- Refinement of the Kobayashi-Ong type system: coloring is a modality
- Colored models of the λY-calculus coming from linear logic
- Decidability using the finitary Scott semantics
- Raises interesting questions in semantics: infinitary models, coeffects...
- Towards the model-checking of other classes of properties?

Thank you for your attention!

Conclusion

- Connections between intersection types and linear logic
- Refinement of the Kobayashi-Ong type system: coloring is a modality
- Colored models of the λY-calculus coming from linear logic
- Decidability using the finitary Scott semantics
- Raises interesting questions in semantics: infinitary models, coeffects...
- Towards the model-checking of other classes of properties?

Thank you for your attention!