Semantics of linear logic and higher-order model-checking

Charles Grellois

Thèse dirigée par Paul-André Melliès et Olivier Serre

IRIF — Université Paris Diderot - Paris 7

8 avril 2016

Semantics of linear logic and higher-order model-checking

Linear logic: a logical system with an emphasis on the notion of *resource*.

Model-checking: a key technique in *verification* — where we want to determine *automatically* whether a program satisfies a specification.

My thesis: linear logic and its semantics can be enriched to obtain new and cleaner proofs of decidability in higher-order model-checking.

What is model-checking?

The halting problem

A natural question: does a program always terminate?

Undecidable problem (Turing 1936): a machine can not always determine the answer.

What if we use approximations?

Model-checking

Approximate the program $\longrightarrow$ build a model $\mathcal{M}$.

Then, formulate a logical specification φ over the model.

Aim: design a program which checks whether

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

That is, whether the model $\mathcal{M}$ meets the specification φ.

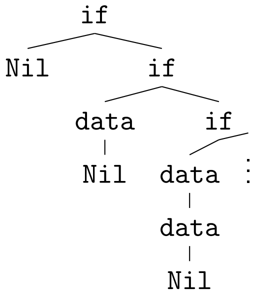
An example

```
  Main   =  Listen Nil
Listen x =  if end_signal() then x
            else Listen received_data() :: x
```

An example

```
Main     =  Listen Nil
Listen x =  if end_signal() then x
            else Listen received_data()::x
```

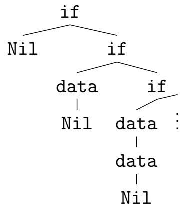
A tree model:



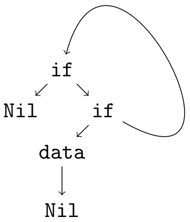
We abstracted conditionals and datatypes.

The approximation contains a non-terminating branch.

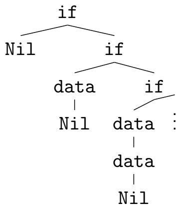
Finite representations of infinite trees



is not regular: it is not the unfolding of a finite graph as



Finite representations of infinite trees



but it is represented by a higher-order recursion scheme (HORS).

Higher-order recursion schemes

Some regularity for infinite trees

(see Chapter 3)

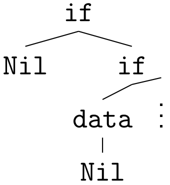
Higher-order recursion schemes

$$\begin{array}{rcl} \texttt{Main} & = & \texttt{Listen Nil} \\ \texttt{Listen } x & = & \texttt{if end\_signal() then } x \\ & & \texttt{else Listen received\_data()} :: x \end{array}$$

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 represents the higher-order tree of actions



Higher-order recursion schemes

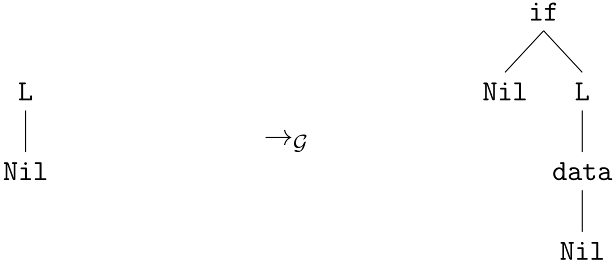
$$\mathcal{G} = \begin{cases} \mathtt{S} & = \quad \mathtt{L\ Nil} \\ \mathtt{L}\ x & = \quad \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$

Rewriting starts from the start symbol S:

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

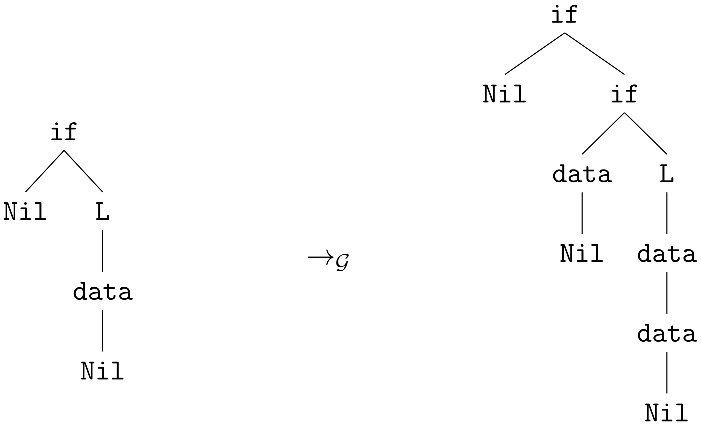
Higher-order recursion schemes

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



Higher-order recursion schemes

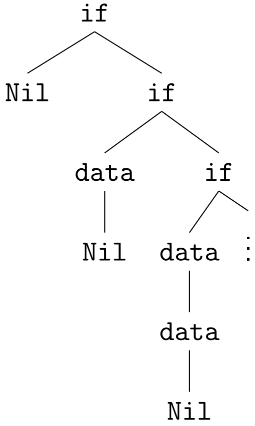
$$\mathcal{G} = \begin{cases} \mathtt{S} & = \quad \mathtt{L\ Nil} \\ \mathtt{L}\ x & = \quad \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$



Higher-order recursion schemes

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

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



Higher-order recursion schemes

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

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

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

Higher-order recursion schemes

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

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

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

The rewriting may be presented coinductively (see Chapter 4).

Alternating parity tree automata

Checking specifications over trees

(see Chapter 2)

Monadic second order logic

MSO is a common logic in verification, allowing to express properties as:

« all executions halt »

« a given operation is executed infinitely often in some execution »

« every time data is added to a buffer, it is eventually processed »

Alternating parity tree automata

Checking whether a formula holds can be performed using an automaton.

For an MSO formula φ, there exists an equivalent APT $\mathcal{A}_\varphi$ s.t.

$$\langle \mathcal{G} \rangle \models \varphi \quad \text{iff} \quad \mathcal{A}_\varphi \text{ has a run over } \langle \mathcal{G} \rangle.$$

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

Alternating tree automata

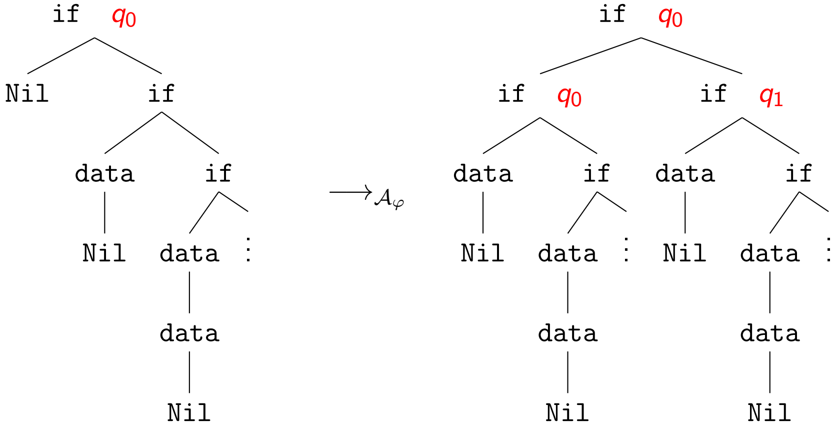
ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

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

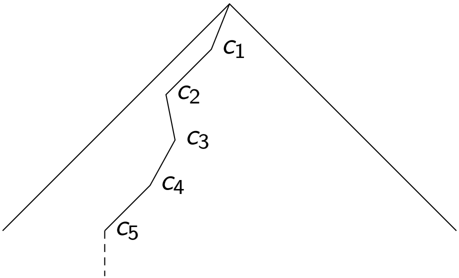


Alternating parity tree automata

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:

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

The higher-order model-checking problems

The (local) HOMC problem

Input: HORS $\mathcal{G}$, formula φ.

Output: `true` if and only if $\langle \mathcal{G} \rangle \vDash \varphi$.

Example: φ = « there is an infinite execution »

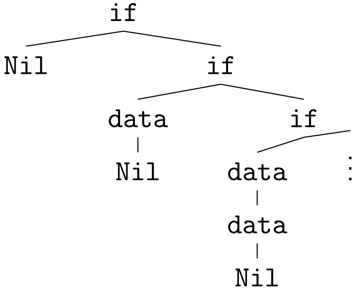
Output: `true`.

The (local) HOMC problem

Input: HORS $\mathcal{G}$, formula φ.

Output: true if and only if $\langle \mathcal{G} \rangle \vDash \varphi$.

Example: φ = « there is an infinite execution »



Output: true.

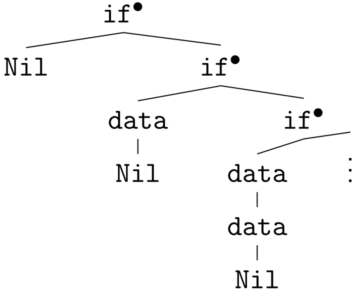
The global HOMC problem

Input: HORS $\mathcal{G}$, formula φ.

Output: a HORS $\mathcal{G}^\bullet$ producing a marking of $\langle \mathcal{G} \rangle$.

Example: φ = « there is an infinite execution »

Output: $\mathcal{G}^\bullet$ of value tree:



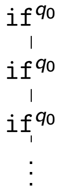
The selection problem

Input: HORS $\mathcal{G}$, APT $\mathcal{A}$, state $q \in Q$.

Output: `false` if there is no winning run of $\mathcal{A}$ over $\langle \mathcal{G} \rangle$.
Else, a HORS $\mathcal{G}^q$ producing a such a winning run.

Example: $\varphi \ = \ $ « there is an infinite execution », q_0 corresponding to φ

Output: $\mathcal{G}^{q_0}$ producing



Purpose of this thesis

These three problems are decidable, with elaborate proofs (often) relying on semantics.

Our contribution: an excavation of the semantic roots of HOMC, at the light of linear logic, leading to refined and clarified proofs.

Recognition by homomorphism

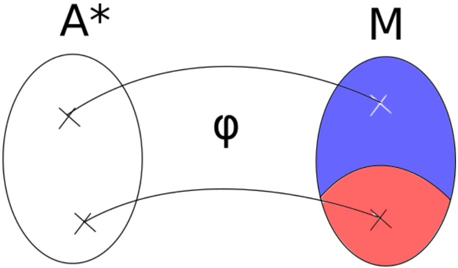
Where semantics comes into play

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

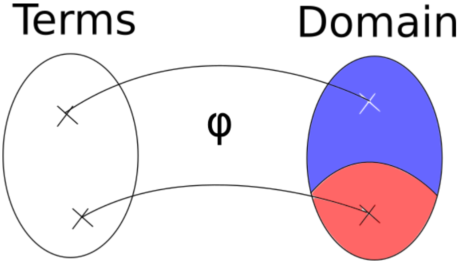
if and only if...



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

Automata and recognition

The picture we want:



(after Aehlig 2006, Salvati 2009)

but with recursion and w.r.t. an APT.

Intersection types and alternation

A first connection with linear logic

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

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

refining the simple typing

$$\texttt{if} \; : \; o \to o \to o$$

Alternating tree automata and intersection types

In a derivation typing the tree `if` T_1 T_2 :

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

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

Theorem (Kobayashi 2009)

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

A closer look at the Application rule

In the intersection type system:

$$\text{App} \qquad \frac{\Delta \vdash t : (\ \theta_1\ \wedge \cdots \wedge\ \theta_n) \to \theta \qquad \Delta_i \vdash u\ :\ \theta_i}{\Delta\ ,\ \Delta_1\ ,\ \ldots\ ,\ \Delta_n\ \vdash\ t\,u\ :\ \theta}$$

This rule could be decomposed as:

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

A closer look at the Application rule

In the intersection type system:

$$\text{App} \qquad \frac{\Delta \vdash t : (\,\theta_1 \;\wedge \cdots \wedge\; \theta_n) \to \theta \qquad \Delta_i \vdash u \,:\, \theta_i}{\Delta\,,\, \Delta_1\,,\, \ldots\,,\, \Delta_n \;\vdash\; t\,u \,:\, \theta}$$

This rule could be decomposed as:

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

A closer look at the Application rule

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

Linear decomposition of the intuitionistic arrow:

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

Two steps: duplication / erasure, then linear use.

Right $\bigwedge$ corresponds to the Promotion rule of indexed linear logic.
(see G.-Melliès, ITRS 2014)

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)

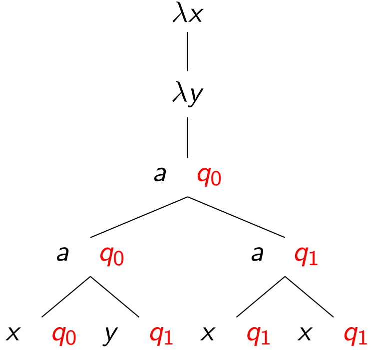
$$!A \;\; = \;\; \mathcal{M}_{fin}(A)$$

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

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

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

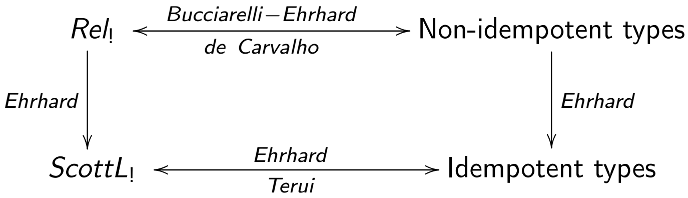
$$(\{q_0, q_1, q_2\}, \{q_1\}, q_0)$$

and

$$(\{q_0, q_1\}, \{q_0, q_1\}, q_0)$$

and ...

Intersection types and semantics of linear logic



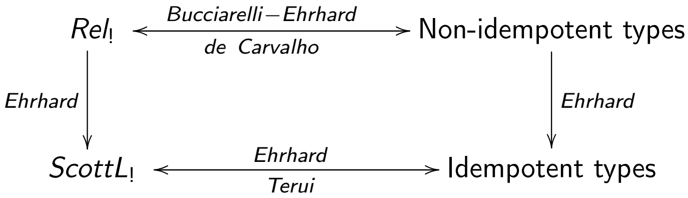
(Bucciarelli-Ehrhard 2001, de Carvalho 2009, Ehrhard 2012, Terui 2012)

Fundamental idea:

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

for a closed term.

Intersection types and semantics of linear logic



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

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

(see Chapter 5)

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:

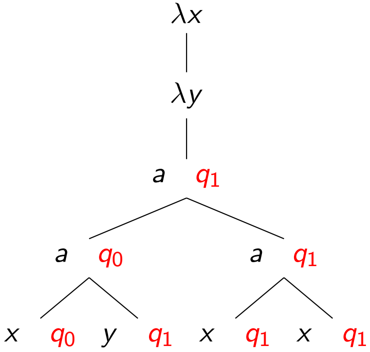
$$\begin{array}{c} \mathtt{if} \\ [\,]_{q_0} \quad [\,]_{q_1} \end{array}$$

A new neutral (least) color: ϵ.

We refine the approach of Kobayashi and Ong in a modal way (see Chapter 6).

An example of colored intersection type

Set $\Omega(q_0) = 0$ and $\Omega(q_1) = 1$.



has now 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 $$\frac{}{x \,:\, \square_{\epsilon}\ \theta_i \ \vdash\ x \,:\, \theta_i}$$

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

App $$\frac{\Delta \vdash t : (\square_{m_1}\ \theta_1\ \wedge \cdots \wedge \square_{m_k}\ \theta_k) \rightarrow \theta \qquad \Delta_i \vdash u \,:\, \theta_i}{\Delta + \square_{m_1}\Delta_1 + \ldots + \square_{m_k}\Delta_k\ \vdash\ t\ u \,:\, \theta}$$

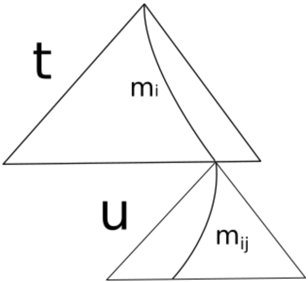
λ $$\frac{\Delta\,,\, x \,:\, \bigwedge_{i \in I} \square_{m_i}\ \theta_i\ \vdash\ t \,:\, \theta}{\Delta\ \vdash\ \lambda\, x \,.\, t \,:\, \left(\bigwedge_{i \in I} \square_{m_i}\ \theta_i\right) \rightarrow \theta}$$

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

A type-system for verification

A colored Application rule:

$$\text{App} \quad \frac{\Delta \vdash t : (\square_{m_1}\ \theta_1\ \wedge \cdots \wedge \square_{m_k}\ \theta_k) \rightarrow \theta \qquad \Delta_i \vdash u\ :\ \theta_i}{\Delta\ +\ \square_{m_1}\Delta_1\ +\ \ldots\ +\ \square_{m_k}\Delta_k\ \vdash\ t\ u\ :\ \theta}$$



A type-system for verification

A colored Application rule:

$$\text{App} \qquad \frac{\Delta \vdash t : (\square_{m_1}\ \theta_1\ \wedge \cdots \wedge \square_{m_k}\ \theta_k) \rightarrow \theta \qquad \Delta_i \vdash u\ :\ \theta_i}{\Delta\ +\ \square_{m_1}\Delta_1\ +\ \ldots\ +\ \square_{m_k}\Delta_k\ \vdash\ t\ u\ :\ \theta}$$

inducing a winning condition on infinite proofs: the node

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

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

A type-system for verification

We now capture all MSO (see Chapter 6-8):

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.

Our reformulation

- shows the modal nature of $\Box$ (in the sense of S4),
- internalizes the parity condition,
- paves the way for semantic constructions.

Colored models of linear logic

A closer look at the Application rule

$$\frac{\Delta \vdash t : (\Box_{m_1}\ \theta_1\ \wedge \cdots \wedge \Box_{m_k}\ \theta_k) \rightarrow \theta \qquad \Delta_i \vdash u : \theta_i}{\Delta + \Box_{m_1}\Delta_1 + \ldots + \Box_{m_k}\Delta_k \ \vdash\ t\,u : \theta}$$

could be decomposed as:

$$\frac{\Delta \vdash t : \left(\bigwedge_{i=1}^{k} \Box_{m_i}\ \theta_i\right) \rightarrow \theta \qquad \dfrac{\dfrac{\Delta_1 \vdash u : \theta_1}{\Box_{m_1}\ \Delta_1\ \vdash\ u : \Box_{m_1}\ \theta_1} \quad \ldots \quad \dfrac{\Delta_k \vdash u : \theta_k}{\Box_{m_k}\ \Delta_k\ \vdash\ u : \Box_{m_k}\ \theta_k}\ \text{Right}\ \Box}{\Box_{m_1}\Delta_1, \ldots, \Box_{m_k}\Delta_k\ \vdash\ u : \bigwedge_{i=1}^{k} \Box_{m_i}\ \theta_i}\ \text{Right}\ \bigwedge}{\Delta, \Box_{m_1}\Delta_1, \ldots, \Box_{m_k}\Delta_k\ \vdash\ t\,u : \theta}$$

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

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

We show that the modality $\Box$ distributes over the exponential in the semantics.

Colored semantics

We extend:

- *Rel* with countable multiplicities, coloring and an inductive-coinductive fixpoint (Chapter 9)
- *ScottL* with coloring and an inductive-coinductive fixpoint (Chapter 10).

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 (parametric comonad)

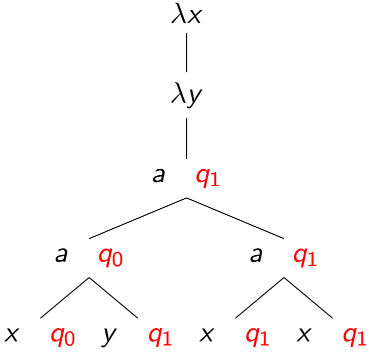
$$\Box\, A \;=\; Col \times A$$

Composite comonad: $\boldsymbol{\lightning} \;=\; \lightning\,\Box$ is an exponential.

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

Model-checking and infinitary semantics (Chapter 9)

Inductive-coinductive fixpoint operator: composes denotations w.r.t. the parity condition.

Theorem

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

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

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

Conjecture

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

$$q_0 \in [\![\lambda(\mathcal{G})^{\Sigma}]\!] \circ [\![\delta^{\dagger}]\!]$$

where $\lambda(\mathcal{G})^{\Sigma}$ is a Church encoding of a λY-term corresponding to $\mathcal{G}$.

Finitary semantics (Chapter 10)

In ScottL, we define $\Box$, λ and $\mathbf{Y}$ similarly (using downward-closures).
$ScottL_{\lightning}$ 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 local higher-order model-checking problem is decidable (and is n-EXPTIME complete).

Theorem

The selection problem is decidable.

Contributions (see Chapter 11)

- A coinductive presentation of the interaction of HORS rewriting and APT execution (Chapter 4)
- A modal and purely type-theoretic reformulation of the Kobayashi-Ong type system (Chapter 6), including a full proof of the soundness-and-completeness theorem (Chapters 7 and 8)
- An infinitary model of linear logic, with a non-continuous interpretation of λY-terms (Chapter 9)
- Colored tensorial logic (Chapter 9)
- A finitary model of linear logic leading to the decidability of the HOMC problems (Chapter 10)

Perspectives (see Chapter 11)

- A purely coinductive proof of the soundness-and-completeness theorem
- Accommodating the modal approach to other classes of automata
- Understanding the infinitary semantics
- Logical aspects: colored tensorial logic, fixpoints...
- Game semantics interpretations?
- Is the complexity related to light linear logics?
- Extensional collapse between the two colored models?