Semantic methods in higher-order model-checking

Charles Grellois (joint work with Paul-André Melliès)

PPS & LIAFA — Université Diderot - Paris 7

RAPIDO meeting — June 18, 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:
a tree produced by a higher-order recursion scheme (HORS)

over which we run

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

corresponding to a

monadic second-order logic (MSO) formula φ.

Higher-order recursion schemes

Higher-order recursion schemes

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

A HORS is a kind of deterministic higher-order grammar.

Rewrite rules have (higher-order) parameters.

"Everything" is simply-typed.

Rewriting produces a tree $\langle \mathcal{G} \rangle$.

Higher-order recursion schemes

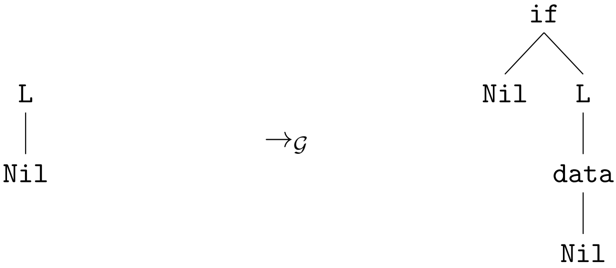
$$\mathcal{G} \;=\; \begin{cases} \mathtt{S} & = \quad \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L}\ x & = \quad \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$

Rewriting starts from the start symbol S:

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

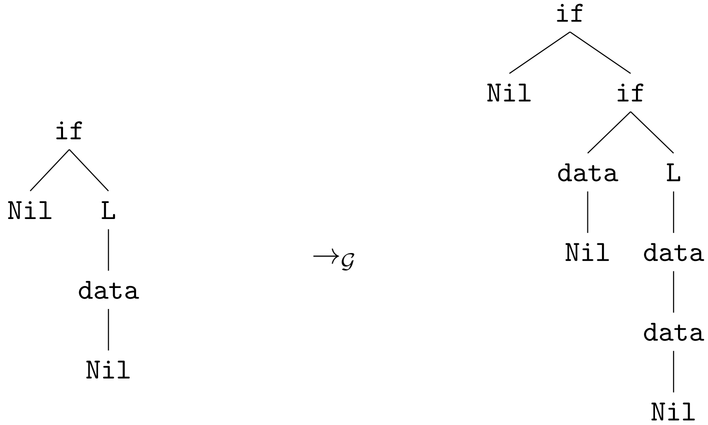
Higher-order recursion schemes

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



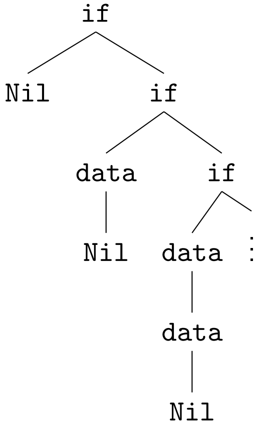
Higher-order recursion schemes

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



Higher-order recursion schemes

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



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

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

Higher-order recursion schemes

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

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

free variables of order at most 1 ($=$ tree constructors)

and

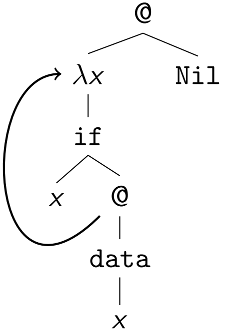
simply-typed recursion operators $Y_\sigma \;:\; (\sigma \Rightarrow \sigma) \Rightarrow \sigma$.

Here : $\mathcal{G} \quad \leftrightsquigarrow \quad (Y_{o \Rightarrow o}\,(\lambda \mathtt{L}.\lambda x.\mathtt{if}\ x\ \ (\mathtt{L}\,(\mathtt{data}\ x))))\ \ \mathtt{Nil}$

Game semantics and HORS

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

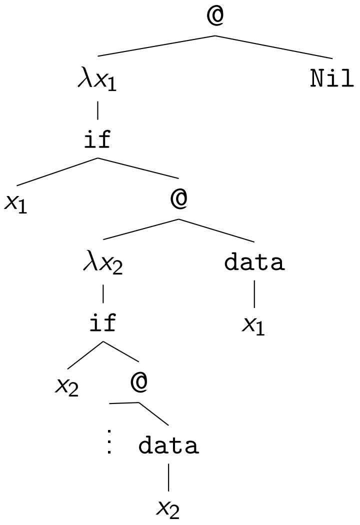
Graph representation of the Y combinator.



Game semantics and HORS

$$\begin{cases} \mathtt{S} & = & \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L} & = & \lambda x.\, \mathtt{if}\ \ x\ (\mathtt{L}\ \ (\mathtt{data}\ \ x)) \end{cases}$$

Unfolding as a regular grammar produces an infinite term.

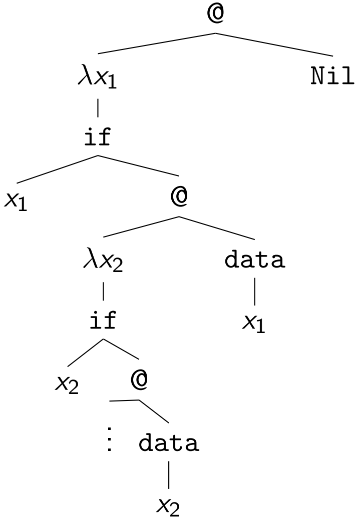


Game semantics and HORS

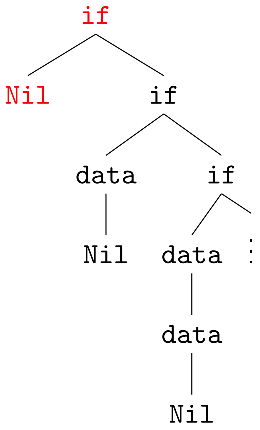
Ong 2006:

Notion of traversal:
trace of the head
reduction along a path.

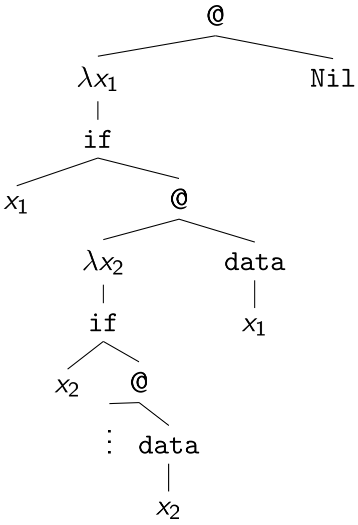
Path-traversal
correspondence.



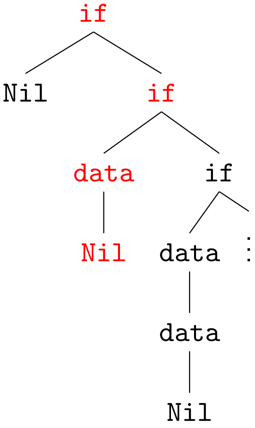
Game semantics and HORS



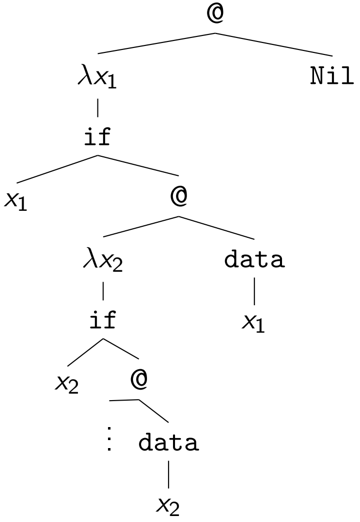
vs.



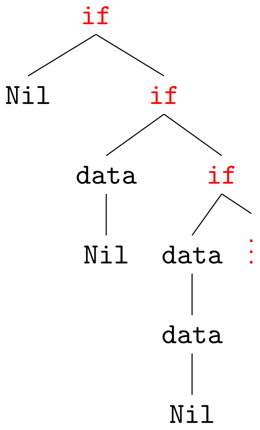
Game semantics and HORS



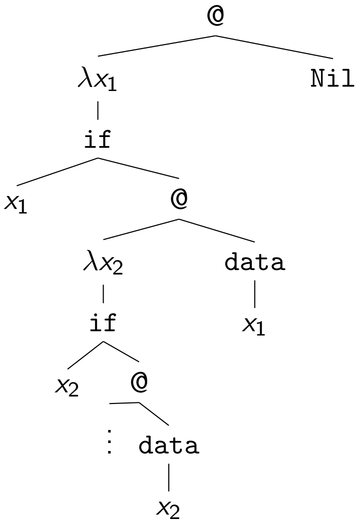
vs.



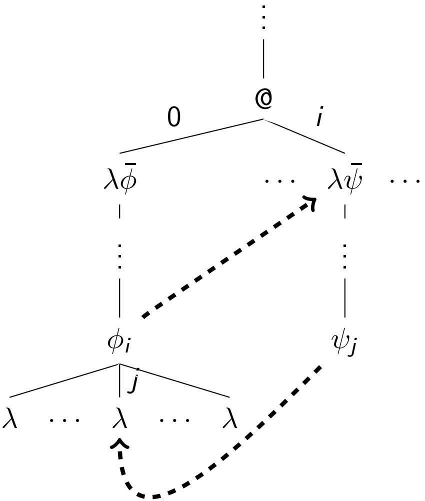
Game semantics and HORS



vs.



Traversals: general shape



Alternating parity tree automata

Alternating parity tree automata

We will use this semantic understanding of HORS to analyze them w.r.t. a MSO formula.

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

Alternating tree automata

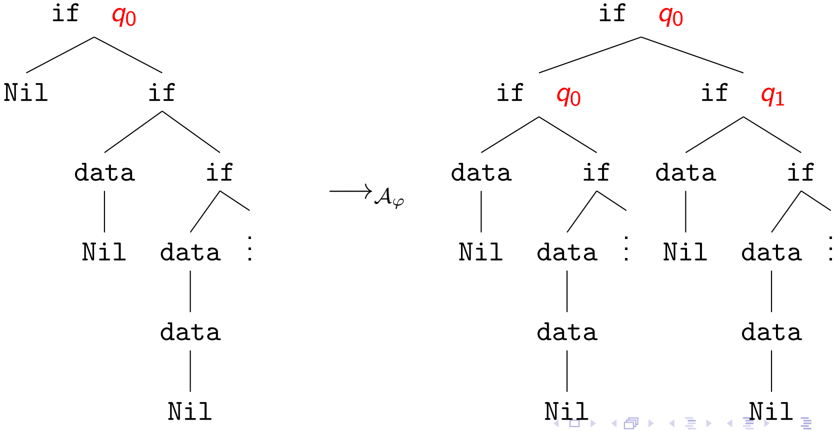
ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

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 tree automata

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

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

This infinite process produces a run-tree of $\mathcal{A}_\varphi$ over $\langle \mathcal{G} \rangle$.

It is an infinite, unranked tree.

Alternating parity tree automata

MSO allows to discriminate 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 receives 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.

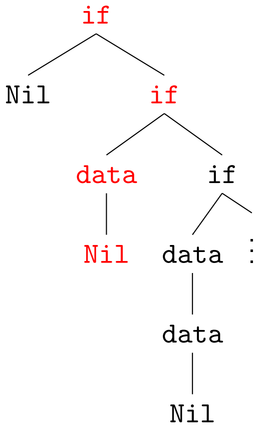
A run-tree is winning iff all its infinite branches are.

For a MSO formula φ:

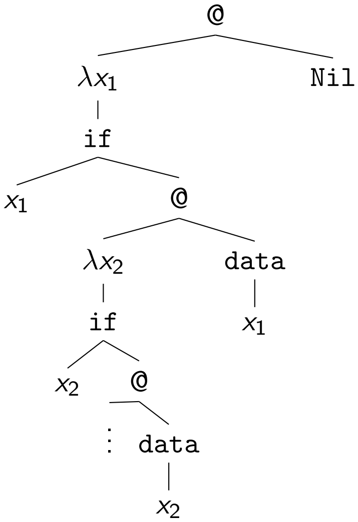
$$\mathcal{A}_\varphi \text{ has a winning run-tree over } \langle \mathcal{G} \rangle \text{ iff } \langle \mathcal{G} \rangle \vDash \phi$$

Traversals and APT

The traversal-simulating APT



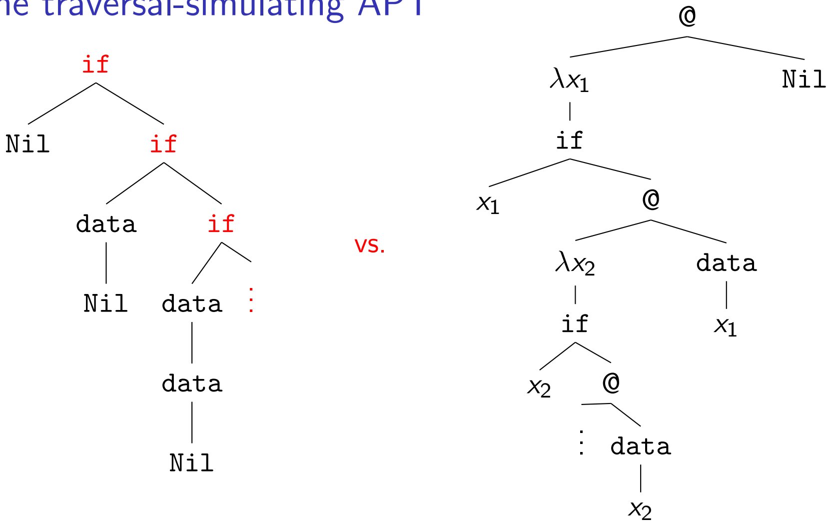
vs.



Necessity to simulate jumps, using non-determinism (heavily).

On Application nodes: guess of some environment (including colors).

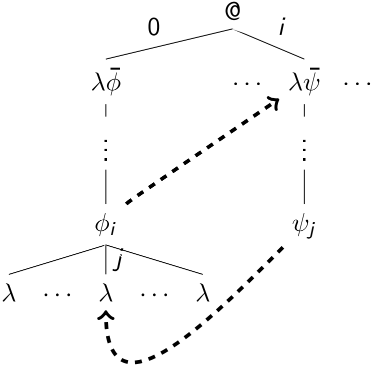
The traversal-simulating APT



Necessity to simulate jumps, using non-determinism (heavily).

On Application nodes: guess of some environment (including colors).

The traversal-simulating APT



Ong relates the runs of this APT with the ones of the original APT using the path-traversal correspondence.

Huge APT, but over a finite graph $\longrightarrow$ decidability.

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

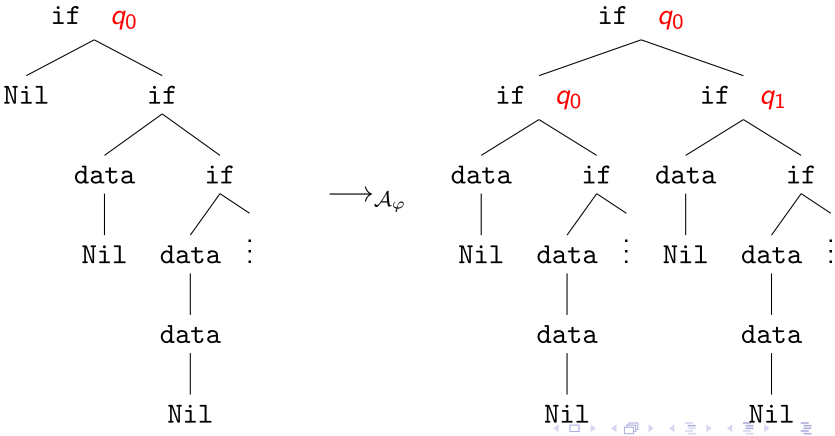
$$\texttt{if} \;:\; o \Rightarrow o \Rightarrow o$$

Alternating tree automata and intersection types

Recall the effect of

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

during an execution of $\mathcal{A}$:



Alternating tree automata and intersection types

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

$$\dfrac{\text{App}\;\dfrac{\delta\;\dfrac{}{\emptyset \vdash \texttt{if} : \emptyset \Rightarrow (q_0 \wedge q_1) \Rightarrow q_0} \qquad \color{red}{\emptyset}}{\emptyset \vdash \texttt{if}\ T_1 : (q_0 \wedge q_1) \Rightarrow q_0} \qquad \color{red}{\dfrac{\vdots}{\Gamma_1 \vdash T_2 : q_0}} \qquad \color{red}{\dfrac{\vdots}{\Gamma_2 \vdash T_2 : q_1}}}{\Gamma_1, \Gamma_2 \vdash \texttt{if}\ T_1\ T_2 : q_0}\;\text{App}$$

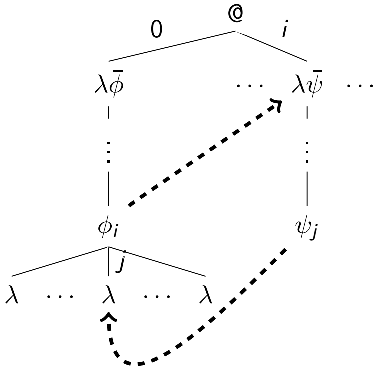
Intersection types naturally lift to higher-order – and thus to $\mathcal{G}$, which finitely represents $\langle \mathcal{G} \rangle$.

Theorem (Kobayashi)

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

A step towards decidability… but what about parity ?

Alternating parity tree automata and intersection types



Kobayashi-Ong (2009): encode the traversal-simulating APT as an intersection type system.

Idempotency + finite graph $\longrightarrow$ decidability.

Intersection types and linear logic

Intersection types and linear logic

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

A program of type $A \Rightarrow B$

duplicates or drops elements of A

and then

uses linearly ($=$ once) each copy

Just as intersection types and APT.

Intersection types and linear logic

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

Two interpretations of the exponential modality:

Qualitative models
(Scott semantics)

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

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

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

Order closure

Quantitative models
(Relational semantics)

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

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

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

Unbounded multiplicities

Intersection types and linear logic

Models of linear logic and intersection types (refining simple types):

$$
\begin{tikzcd}
Rel \arrow[d, "Ehrhard"'] & Rel_{!} \arrow[l, tail] \arrow[d, dotted] \arrow[r, leftrightarrow, "Bucciareli-Ehrhard", "de\ \ Carvalho"'] & \text{Non-idempotent types} \arrow[d, "Ehrhard,\ G-M"] \\
Scott & Scott_{!} \arrow[l, tail] \arrow[r, leftrightarrow, "Terui"'] & \text{Idempotent types}
\end{tikzcd}
$$

Fundamental idea: derivations of the intersection type systems compute denotations in the associated model.

(see also G-Melliès, ITRS 2014)

Intersection types and linear logic

Models of linear logic and intersection types (refining simple types):

$$
\begin{tikzcd}
Rel \arrow[d, "Ehrhard"'] & Rel_! \arrow[l, tail] \arrow[d, dotted] \arrow[r, leftrightarrow, "Bucciareli-Ehrhard", "de\ \ Carvalho"'] & \text{Non-idempotent types} \arrow[d, "{Ehrhard,\ G-M}"] \\
Scott & Scott_! \arrow[l, tail] \arrow[r, leftrightarrow, "Terui"'] & \text{Idempotent types}
\end{tikzcd}
$$

$$
\begin{tikzcd}
{[q_0,\, q_0,\, q_1] \multimap q_0} \arrow[r, mapsto] \arrow[d, mapsto] & q_0 \wedge q_0 \wedge q_1 \rightarrow q_0 \arrow[d, mapsto] \\
\{q_0,\, q_1\} \multimap q_0 \arrow[r, mapsto] & q_0 \wedge q_1 \rightarrow q_0
\end{tikzcd}
$$

Intersection types and linear logic

Models of linear logic and intersection types (refining simple types):

$$\begin{tikzcd}
Rel \arrow[d, "Ehrhard"'] & Rel_! \arrow[l] \arrow[d, dotted] \arrow[r, leftrightarrow, "Bucciareli-Ehrhard", "de\ Carvalho"'] & \text{Non-idempotent types} \arrow[d, "Ehrhard,\ G-M"] \\
Scott & Scott_! \arrow[l] \arrow[r, leftrightarrow, "Terui"'] & \text{Idempotent types}
\end{tikzcd}$$

Important remark: in order to connect idempotent types with a denotational model ($\rightarrow$ invariance modulo $\beta\eta$), one needs subtyping.

Subtyping appears naturally in the Scott model, as the order closure condition.

In the relational semantics/non-idempotent types: no such requirement. But unbouded multiplicities. . .

Four theorems: inductive version

We obtain a theorem for every corner of our "equivalence square":

Theorem

In the relational (resp. Scott) semantics,

$$q_0 \in [\![\mathcal{G}]\!] \quad \textit{iff} \quad \textit{the ATA } \mathcal{A}_\phi \textit{ has a finite run-tree over } \langle \mathcal{G} \rangle.$$

Theorem

With non-idempotent (resp. idempotent with subtyping) intersection types,

$$\vdash \mathcal{G} : q_0 \quad \textit{iff} \quad \textit{the ATA } \mathcal{A}_\phi \textit{ has a finite run-tree over } \langle \mathcal{G} \rangle.$$

An infinitary model of linear logic

Restrictions to finiteness:

- for *Rel* and non-idempotent types: lack of a countable multiplicity ω. Recall that tree constructors are free variables...
- for idempotent types: just need to allow infinite (or circular) derivations.
- for *Scott*: interpret Y as the gfp.

In *Rel*, we introduce a new exponential $A \mapsto \lightning A$ s.t.

$$[\![\lightning A]\!] \quad = \quad \mathcal{M}_{count}([\![A]\!])$$

$\mathcal{M}_{count}$ builds finite-or-countable multisets.

(G-Melliès, FoSSaCS 2015)

An infinitary model of linear logic

This defines an infinitary model of linear logic, which corresponds to

non-idempotent intersection types with countable multiplicities

and derivations of countable depth.

It admits a coinductive fixpoint, which we use to interpret Y.

The four theorems generalize to all ATA ($\rightarrow$ infinite runs).

And the parity condition ?

Alternating parity tree automata

Kobayashi and Ong's type system has a quite complex handling of colors.

We reformulate it in a very simple way:

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

now corresponds to

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

Application computes the “local” maximum of colors, and the fixpoint deals with the acceptance condition.

In this reformulation, the colors behave as a family of modalities.

The coloring comonad

Since coloring is a modality, it defines a comonad in the semantics:

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

which can be composed with $\lightning$, so that

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

corresponds to

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

in the semantics (relational in this example, but it also works for Scott)

An inductive-coinductive fixpoint operator

We define a fixpoint operator:

- On typing derivations: rephrasal of the parity condition over derivations $\longrightarrow$ winning derivations.
- On denotations: it composes inductively or coinductively elements of the semantics, according to the current color.

Work in progress: semantic definition of Y using directly the lfp and gfp (strongly related to the expression of the solution of parity games with lfp and gfp).

The final picture

$$
\begin{tikzcd}
\mathit{Rel}_{\lightning} + \square + Y \arrow[r, leftrightarrow] \arrow[d, dotted] & \text{Non-idempotent types} + \square + Y \arrow[d, dotted] \\
\mathit{Scott}_{\lightning} + \square + Y \arrow[r, leftrightarrow] & \text{Idempotent types} + \square + Y
\end{tikzcd}
$$

Open question: are the dotted lines an extensional collapse again?

Four theorems: full version

We obtain a theorem for every corner of our "colored equivalence square":

Theorem (G-Melliès, CSL 2015)

In the colored relational (resp. colored Scott) semantics,

$$q_0 \in [\![\mathcal{G}]\!] \quad \textit{iff} \quad \textit{the APT } \mathcal{A}_\phi \textit{ has a winning run-tree over } \langle \mathcal{G} \rangle.$$

Theorem (G-Melliès, MFCS 2015)

With colored non-idempotent (resp. colored idempotent with subtyping) intersection types, there is a winning derivation of

$$\vdash \mathcal{G} : q_0 \quad \textit{iff} \quad \textit{the APT } \mathcal{A}_\phi \textit{ has a winning run-tree over } \langle \mathcal{G} \rangle.$$

The selection problem

In the Scott/idempotent case, finiteness $\Rightarrow$ decidability of the higher-order model-checking problem.

Even better: the selection problem is decidable.

If $\mathcal{A}_\phi$ accepts $\langle \mathcal{G} \rangle$, we can compute effectively a new scheme $\mathcal{G}'$ such that $\langle \mathcal{G}' \rangle$ is a winning run-tree of $\mathcal{A}_\phi$ over $\langle \mathcal{G} \rangle$.

In other words: there is a higher-order winning run-tree.

(the key: annotate the rules with their denotation/their types).

The selection problem

$$\begin{cases} \mathtt{S} & = & \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L} & = & \lambda x.\,\mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x)) \end{cases}$$

becomes e.g.

$$\begin{cases} \mathtt{S}^{q_0} & = & \mathtt{L}^{\{q_0,\, q_1\} \multimap q_0}\ \mathtt{Nil}^{q_0}\ \mathtt{Nil}^{q_1} \\ \mathtt{L}^{\{q_0,\, q_1\} \multimap q_0} & = & \lambda x^{\{q_0,\, q_1\}}.\ \mathtt{if}^{\emptyset \multimap \{q_0,\, q_1\} \multimap q_0}\ \left(\mathtt{L}^{\{q_1\} \multimap q_0}\ \left(\mathtt{data}^{\{q_0\} \multimap q_1}\ x^{q_0}\right)\right)\ \left(\mathtt{L}^{\{q_0\} \multimap q_1}\ \left(\mathtt{data}^{\{q_0, q_1\} \multimap q_0}\ x^{q_0}\ x^{q_1}\right)\right) \\ \mathtt{L}^{\{q_0\} \multimap q_1} & = & \cdots \\ \mathtt{L}^{\{q_1\} \multimap q_0} & = & \cdots \end{cases}$$

Other approaches

- Ong 2006 (game semantics)
- Hague-Murawski-Ong-Serre 2008 (game semantics + collapsible higher-order pushdown automata)
- Kobayashi-Ong 2009 (intersection types)
- Salvati-Walukiewicz 2011 (interpretation with Krivine machines)
- Carayol-Serre 2012 (collapsible higher-order pushdown automata)
- Tsukada-Ong 2014 (game semantics)
- Salvati-Walukiewicz 2015 (interpretation in finite models)
- Grellois-Melliès 2015

Thank you for your attention!

Other approaches

- Ong 2006 (game semantics)
- Hague-Murawski-Ong-Serre 2008 (game semantics + collapsible higher-order pushdown automata)
- Kobayashi-Ong 2009 (intersection types)
- Salvati-Walukiewicz 2011 (interpretation with Krivine machines)
- Carayol-Serre 2012 (collapsible higher-order pushdown automata)
- Tsukada-Ong 2014 (game semantics)
- Salvati-Walukiewicz 2015 (interpretation in finite models)
- Grellois-Melliès 2015

Thank you for your attention!