Colored intersection types: a bridge between linear logic and higher-order model-checking

Charles Grellois (joint work with Paul-André Melliès)

PPS & LIAFA — Université Paris 7

TYPES conference — May 18th, 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

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

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 schemes (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

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

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

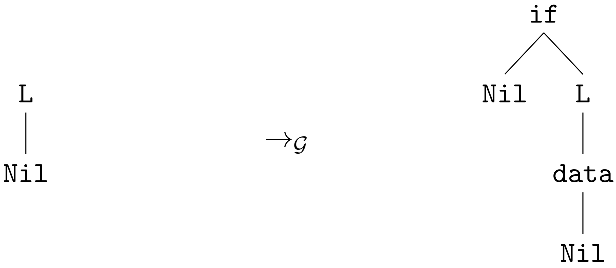
$$\mathcal{G} \;=\; \left\{ \begin{array}{lcl} \mathtt{S} & = & \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L}\ x & = & \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{array} \right.$$

Rewriting starts from the start symbol S:

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

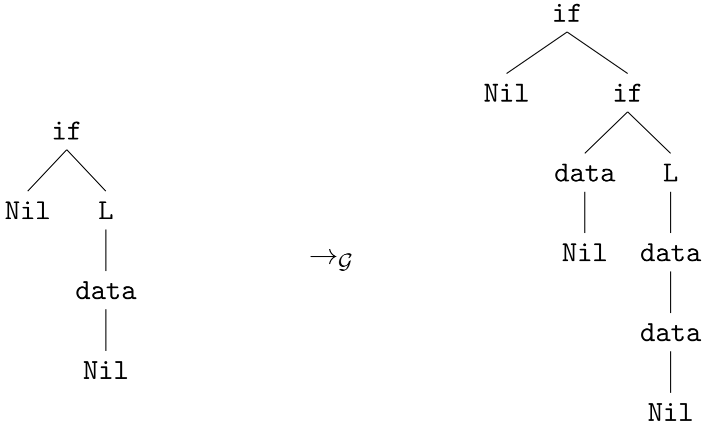
Higher-order recursion schemes

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



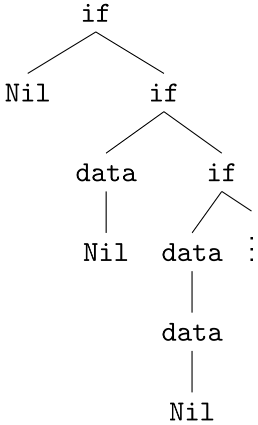
Higher-order recursion schemes

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



Higher-order recursion schemes

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



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

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

Higher-order recursion schemes

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

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

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

and

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

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

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

and

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

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

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

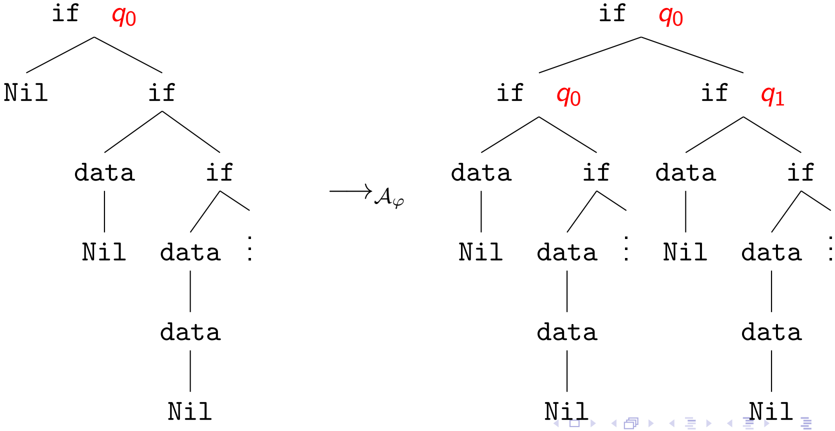
ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

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

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

It is an infinite, unranked tree.

Alternating 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

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

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

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

Theorem (Kobayashi)

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

A step towards decidability. . .

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.

Intersection types and 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 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.

Intersection types and linear logic

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

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

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

Intersection types and linear logic

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

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

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 semantics,

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

Theorem

With non-idempotent intersection types,

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

Four theorems: inductive version

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

Theorem

In the 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 idempotent intersection types (+ subtyping),

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

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.

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

(finite-or-countable multisets)

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

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:

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

Alternating parity tree automata

We reformulate Kobayashi and Ong's colored intersection type system in a very simple way:

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

now corresponds to

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

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

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

corresponds to

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

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.

The final picture

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

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

In the colored relational 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

With colored non-idempotent intersection types, there is a winning derivation of

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

Four theorems: full version

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

Theorem

In the 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

With colored idempotent 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).

Thank you for your attention!

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).

Thank you for your attention!