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 $\varphi$ in an appropriate logic
- Make them interact: the result is whether

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

When the model is a word, a tree... of actions: translate $\varphi$ to an equivalent automaton:

$$\varphi \mapsto A\varphi$$
Model-checking higher-order programs

For higher-order programs with recursion:

\[ \mathcal{M} \text{ is a higher-order tree:} \]
\[ \text{a tree produced by a higher-order recursion schemes (HORS)} \]

over which we run

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

corresponding to a

monadic second-order logic (MSO) formula \( \varphi \).
Higher-order recursion schemes

\[ \mathcal{G} = \begin{cases} S &= L \text{ Nil} \\ L \ x &= \text{if } x (L (\text{data } x )) \end{cases} \]

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

\[ G = \begin{cases} 
S & = \text{L Nil} \\
\text{L } x & = \text{if } x \left( \text{L } (\text{data } x) \right) 
\end{cases} \]

Rewriting starts from the start symbol \( S \):

\[ S \rightarrow_G \text{L Nil} \]
Higher-order recursion schemes

\[ G = \begin{cases} 
  S & = & L \ Nil \\
  L \ x & = & \text{if } x (L \ (\text{data } x)) 
\end{cases} \]
Higher-order recursion schemes

\[ G = \begin{cases} 
S &= L \text{ Nil} \\
L \times &= \text{if } x (L (\text{data } x)) 
\end{cases} \]

\[ \text{if} \]

\[ \text{Nil} \]

\[ L \]

\[ \text{data} \]

\[ \text{Nil} \]

\[ \rightarrow G \]

\[ \text{Nil} \]

\[ L \]

\[ \text{data} \]

\[ \text{Nil} \]
Higher-order recursion schemes

\[
\mathcal{G} = \begin{cases} 
  S &= L \text{ Nil} \\
  L \ x &= \text{if } x (L \ (\text{data } x)) 
\end{cases}
\]

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

It is our model \(\mathcal{M}\).
Higher-order recursion schemes

\[ G = \begin{cases} 
S &= L \text{ Nil} \\
L \times &= \text{if } x (L \text{ (data } x \text{ )})
\end{cases} \]

HORS can alternatively be seen as simply-typed \(\lambda\)-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

\[ G = \begin{cases} 
S & = L \text{ Nil} \\
L \times & = \text{if } x (L (\text{data } x)) 
\end{cases} \]

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

free variables of order at most 1 ($= \text{tree constructors}$)

and

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

Here: \[ G \leftrightarrow (Y_\sigma \Rightarrow_\sigma (\lambda L. \lambda x. \text{if } x (L (\text{data } x)))) \text{ Nil} \]
Alternating parity tree automata

For a MSO formula $\varphi$,

$$\langle G \rangle \models \varphi$$

iff an equivalent APT $A_\varphi$ has a run over $\langle G \rangle$.

\[\text{APT} = \text{alternating tree automata (ATA) + parity condition}.\]
Alternating tree automata

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

Typically: $\delta(q_0, \text{if}) = (2, q_0) \land (2, q_1)$. 
Alternating tree automata

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

Typically: $\delta(q_0, \text{if}) = (2, q_0) \land (2, q_1)$.
Alternating tree automata

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

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

This infinite process produces a run-tree of $A_\varphi$ over $\langle G \rangle$.

It is an infinite, unranked tree.
Alternating tree automata and intersection types

A key remark (Kobayashi 2009):

\[ \delta(q_0, \text{if}) = (2, q_0) \land (2, q_1) \]

can be seen as the intersection typing

\[ \text{if} : \emptyset \Rightarrow (q_0 \land q_1) \Rightarrow q_0 \]

refining the simple typing

\[ \text{if} : o \Rightarrow o \Rightarrow o \]
Alternating tree automata and intersection types

In a derivation typing \( \text{if } T_1 \ T_2 : \)

\[
\begin{align*}
\delta & \quad \frac{\emptyset \vdash \text{if} : \emptyset \Rightarrow (q_0 \land q_1) \Rightarrow q_0}{\emptyset} \quad \emptyset \\
\text{App} & \quad \frac{\emptyset \vdash \text{if} \ T_1 : (q_0 \land q_1) \Rightarrow q_0}{\emptyset \vdash \text{if} \ T_1 \ T_2 : q_0} \\
\text{App} & \quad \frac{\Gamma_1 \vdash T_2 : q_0}{\Gamma_1 \vdash T_2 : q_1} \\
\end{align*}
\]

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 \( A_\varphi \) has a run-tree over \( \langle \mathcal{G} \rangle \).

A step towards decidability...
Intersection types and linear logic

\[ A \Rightarrow B = ! A \multimap B \]

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

\[ A \Rightarrow B = !A \rightarrow B \]

Two interpretations of the exponential modality:

**Qualitative models**
(Scott semantics)

\[ !A = \mathcal{P}_{\text{fin}}(A) \]

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

\[ \{q_0, q_0, q_1\} = \{q_0, q_1\} \]

**Order closure**

**Quantitative models**
(Relational semantics)

\[ !A = \mathcal{M}_{\text{fin}}(A) \]

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

\[ [q_0, q_0, q_1] \neq [q_0, q_1] \]

**Unbounded multiplicities**
Intersection types and linear logic

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

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):

\[ \text{Rel} & \xleftarrow{\text{Ehrhard}} \xleftarrow{\text{Bucciareli–Ehrhard}} \xleftarrow{\text{de Carvalho}} \text{Non-idempotent types} \\
\text{Scott} & \xleftarrow{\text{Ehrhard, } G–M} \xleftarrow{\text{Terui}} \text{Idempotent types} \]

\[ [q_0, q_0, q_1] \twoheadrightarrow q_0 \xrightarrow{q_0 \land q_0 \land q_1} q_0 \]

\[ \{q_0, q_1\} \twoheadrightarrow q_0 \xrightarrow{q_0 \land q_1} q_0 \]
Intersection types and linear logic

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

\[
\text{Rel} \leftarrow \text{Rel}_! \leftarrow \text{Rel} \leftarrow \text{Rel}_! \leftarrow \text{Non-idempotent types}
\]

\[
\text{Ehrhard} \downarrow \quad \text{Ehrhard} \quad \text{Bucciareli–Ehrhard} \quad \text{de Carvalho} \quad \text{Ehrhard, G–M}
\]

\[
\text{Scott} \leftarrow \text{Scott}_! \leftarrow \text{Scott} \leftarrow \text{Terui} \rightarrow \text{Idempotent types}
\]

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 unbounded multiplicities...
Four theorems: inductive version

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

**Theorem**

In the relational semantics,

\[ q_0 \in [G] \iff \text{the ATA } A_\phi \text{ has a finite run-tree over } \langle G \rangle. \]

**Theorem**

With non-idempotent intersection types,

\[ \vdash G : q_0 \iff \text{the ATA } A_\phi \text{ has a finite run-tree over } \langle G \rangle. \]
Four theorems: inductive version

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

**Theorem**

In the *Scott semantics*,

\[ q_0 \in \llbracket G \rrbracket \iff \text{the ATA } A_\phi \text{ has a finite run-tree over } \langle G \rangle. \]

**Theorem**

With *idempotent intersection types* (+ subtyping),

\[ \vdash G : q_0 \iff \text{the ATA } A_\phi \text{ has a finite run-tree over } \langle G \rangle. \]
An infinitary model of linear logic

Restrictions to finiteness:

- for \( \text{Rel} \) and non-idempotent types: lack of a countable multiplicity \( \omega \).
  Recall that tree constructors are free variables.

- for idempotent types: just need to allow infinite (or circular) derivations.

- for \textit{Scott}: interpret \( Y \) as the gfp.

In \( \text{Rel} \), we introduce a new exponential \( A \mapsto \otimes A \) s.t.

\[
[\otimes A] = \mathcal{M}_{\text{count}}([A])
\]

\text{(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 occurring infinitely often along it is even.

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

For a MSO formula \( \varphi \):

\[ A_\varphi \text{ has a winning run-tree over } \langle G \rangle \text{ iff } \langle G \rangle \models \varphi \]
Alternating parity tree automata

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

\[ \delta(q_0, \text{if}) = (2, q_0) \land (2, q_1) \]

now corresponds to

\[ \text{if} : \emptyset \Rightarrow (\Box_{\Omega(q_0)} q_0 \land \Box_{\Omega(q_1)} q_1) \Rightarrow q_0 \]

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:

\[ \square A = \text{Col} \times A \]

which can be composed with \( \triangledown \), so that

\[
\text{if} : \emptyset \Rightarrow (\square_{\Omega(q_0)} q_0 \land \square_{\Omega(q_1)} q_1) \Rightarrow q_0
\]

corresponds to

\[
[ ] \leadsto [(\Omega(q_0), q_0), (\Omega(q_1), q_1)] \leadsto q_0 \in [\text{if}]
\]

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 \( \rightarrow \) 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

\[ \text{Open question: are the dotted lines an extensional collapse again?} \]
Four theorems: full version

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

**Theorem**

In the colored relational semantics,

\[ q_0 \in [G] \iff \text{the APT } A_\phi \text{ has a winning run-tree over } \langle G \rangle. \]

**Theorem**

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

\[ \vdash G : q_0 \iff \text{the APT } A_\phi \text{ has a winning run-tree over } \langle G \rangle. \]
Four theorems: full version

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

**Theorem**

*In the colored Scott semantics,*

\[ q_0 \in \lbrack \mathcal{G} \rbrack \quad \text{iff} \quad \text{the APT } A_\phi \text{ has a winning run-tree over } \langle \mathcal{G} \rangle. \]

**Theorem**

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

\[ \vdash \mathcal{G} : q_0 \quad \text{iff} \quad \text{the APT } A_\phi \text{ has a winning run-tree over } \langle \mathcal{G} \rangle. \]
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 $A_{\phi}$ accepts $\langle G \rangle$, we can compute effectively a new scheme $G'$ such that $\langle G' \rangle$ is a winning run-tree of $A_{\phi}$ over $\langle G \rangle$.

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

(they 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 $A_\phi$ accepts $\langle G \rangle$, we can compute effectively a new scheme $G'$ such that $\langle G' \rangle$ is a winning run-tree of $A_\phi$ over $\langle G \rangle$.

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!