Higher-order model-checking, categorical semantics, and linear logic

Charles Grellois (joint work with Paul-André Melliès)

PPS & LIAFA — Université Paris 7

TACL conference — June 21, 2015

Model-checking higher-order programs

- Construct a model $\mathcal{M}$ of a program
- Specify a property φ in an appropriate logic
- Make them interact: the result is whether

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

If $\mathcal{M}$ 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

modal μ-calculus formula φ.

(NB: here modal μ-calculus is equivalent to MSO)

Higher-order recursion schemes

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

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

Rewrite rules have (higher-order) parameters.

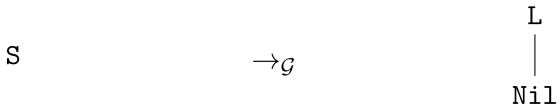
"Everything" is simply-typed.

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

Higher-order recursion schemes

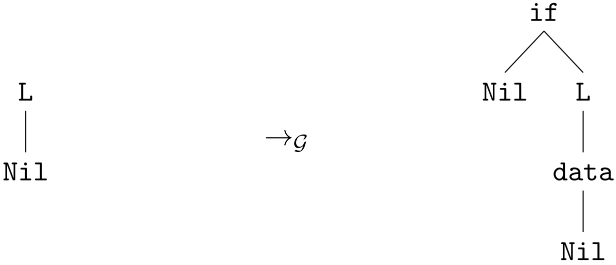
$$\mathcal{G} \;=\; \begin{cases} \mathtt{S} & = \;\; \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L}\ x & = \;\; \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$

Rewriting starts from the start symbol S:



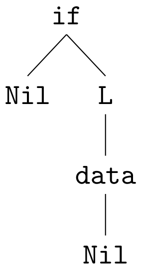
Higher-order recursion schemes

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

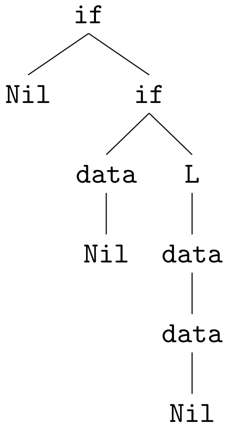


Higher-order recursion schemes

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

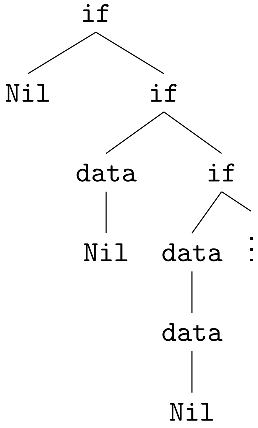


$\to_{\mathcal{G}}$



Higher-order recursion schemes

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



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

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

How to model-check a non-regular tree?

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 $Y_\sigma \; : \; (\sigma \Rightarrow \sigma) \Rightarrow \sigma$.

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

So, we can interpret HORS in models of the λY-calculus.

Alternating parity tree automata

For a modal μ-calculus formula φ,

$$\langle \mathcal{G} \rangle \quad \models \quad \varphi$$

iff an equivalent APT $\mathcal{A}_\varphi$ has a run over $\langle \mathcal{G} \rangle$.

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

Our goal: interpret HORS in a model reflecting the behavior of $\mathcal{A}_\phi$.

Alternating tree automata

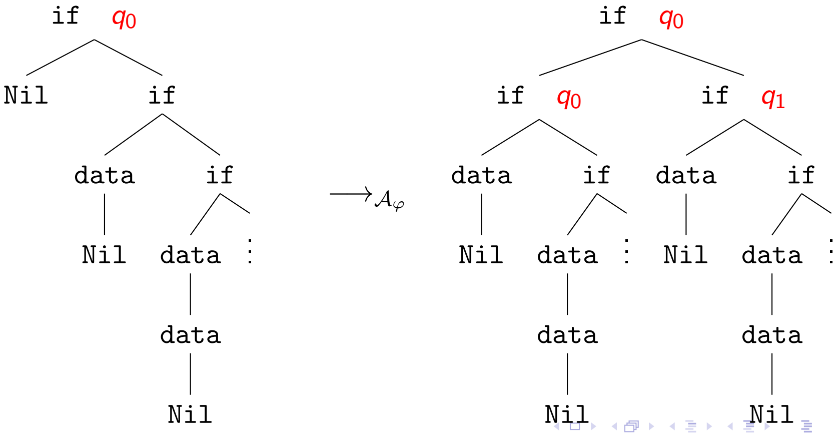
ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

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



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 $\texttt{if}\ \ T_1\ \ T_2$:

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

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 and APT.

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)

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

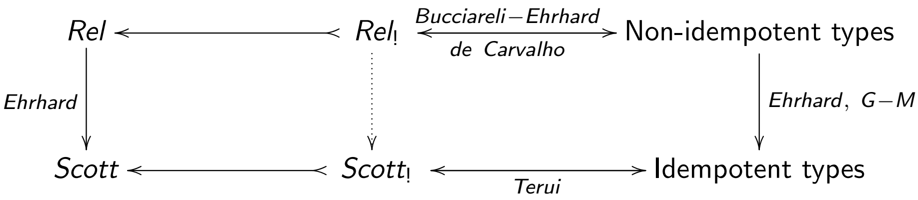
$$[\![o \Rightarrow o]\!] \;=\; \mathcal{M}_{fin}(Q) \times Q$$

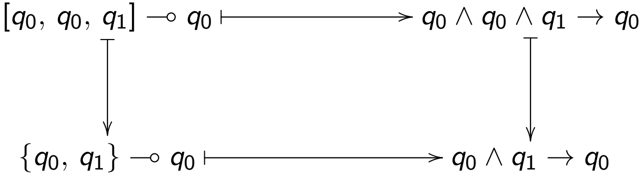
$$[q_0,\, q_0,\, q_1] \;\neq\; [q_0,\, q_1]$$

Unbounded multiplicities

Intersection types and linear logic

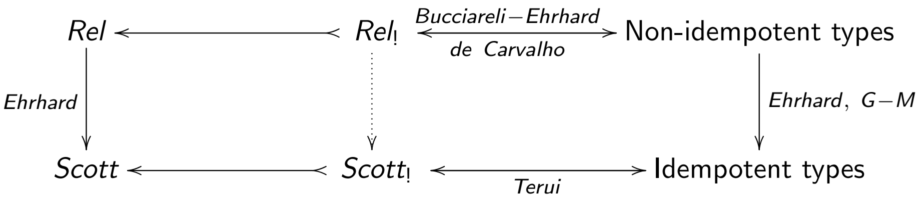
Models of linear logic and intersection types (refining simple types):





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.

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

Rel and non-idempotent types lack of a countable multiplicity ω.
Recall that tree constructors are free variables. . .

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), so that

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

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 modal μ-calculus 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

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

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

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 $\lightning$ thanks to a distributive law.

Now:

$$[\![A \Rightarrow B]\!] \quad = \quad \lightning\, \Box\, [\![A]\!] \multimap [\![B]\!] \quad = \quad \mathcal{M}_{count}(Col \times [\![A]\!]) \times [\![B]\!]$$

We obtain a model of the λ-calculus which reflects the coloring by $\mathcal{A}_\phi$.

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.

The key here: parity conditions can be lifted to higher-order.

The fixpoint can also be defined using μ and ν.

The final picture

$$\begin{array}{ccc} Rel_{\lightning} + \Box + Y & \longleftrightarrow & \text{Non-idempotent types} + \Box + Y \\ \vdots\downarrow & & \vdots\downarrow \\ Scott_{\lightning} + \Box + Y & \longleftrightarrow & \text{Idempotent types} + \Box + Y \end{array}$$

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 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 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.

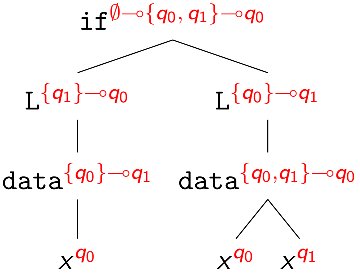
Even better: the selection problem is decidable.

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.

$$\mathtt{S}^{q_0} = \mathtt{L}^{\{q_0,\,q_1\}\multimap q_0}\ \mathtt{Nil}^{q_0}\ \mathtt{Nil}^{q_1}$$



$$\begin{cases} \mathtt{L}^{\{q_0,\,q_1\}\multimap q_0} & = & \lambda x^{\{q_0,\,q_1\}}. \\ \mathtt{L}^{\{q_0\}\multimap q_1} & = & \dots \\ \mathtt{L}^{\{q_1\}\multimap q_0} & = & \dots \end{cases}$$

Conclusion

- Higher-order model-checking $\rightarrow$ verification of non-regular trees.
- Semantic methods allow to study the term generating them.
- Models of linear logic can be extended to capture parity conditions.
- The semantic of a term reflects whether it satisfies a given property.
- Decidability $\rightarrow$ existence of a finite model.

Thank you for your attention!

Conclusion

- Higher-order model-checking $\rightarrow$ verification of non-regular trees.
- Semantic methods allow to study the term generating them.
- Models of linear logic can be extended to capture parity conditions.
- The semantic of a term reflects whether it satisfies a given property.
- Decidability $\rightarrow$ existence of a finite model.

Thank you for your attention!