Finitary semantics of linear logic and higher-order model-checking

Charles Grellois   Paul-André Melliès

PPS & LIAFA — Université Paris 7

MFCS 40 — Aug 28, 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

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

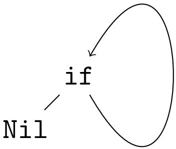
When the model is a word, a tree... of actions: translate φ to an equivalent automaton:

$$\varphi \quad \mapsto \quad \mathcal{A}_{\varphi}$$

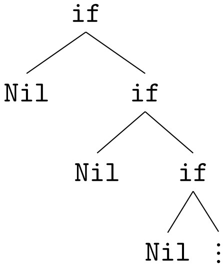
Model-checking higher-order programs

Model-checking of MSO over graphs is well-known: we can decide whether $\mathcal{G} \models \phi$ (amounts to solving a finite parity game).

$\mathcal{G}_0 \quad =$



$\mathcal{T}_0 \quad =$



Graph unfolding $\iff$ regular tree.

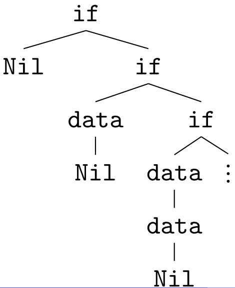
Model-checking higher-order programs

For functional programs (i.e. a function can have a function as input), with recursion (Haskell, OCaml, Javascript, Python...), $\mathcal{M}$ is a higher-order tree.

Example:

$$\begin{array}{rcl} \texttt{Main} & = & \texttt{Listen Nil} \\ \texttt{Listen } x & = & \texttt{if } \textit{end} \texttt{ then } x \texttt{ else Listen (data } x\texttt{)} \end{array}$$

modelled as



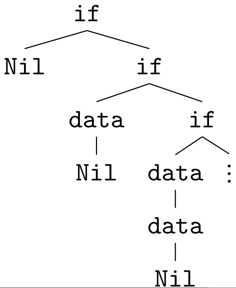
Model-checking higher-order programs

For functional programs (i.e. a function can have a function as input), with recursion (Haskell, OCaml, Javascript, Python...), $\mathcal{M}$ is a higher-order tree.

Example:

$$\begin{array}{rcl} \texttt{Main} & = & \texttt{Listen Nil} \\ \texttt{Listen } x & = & \texttt{if } \textit{end} \texttt{ then } x \texttt{ else Listen (data } x\texttt{)} \end{array}$$

modelled as



How to represent this tree finitely?

Model-checking higher-order programs

For functional programs (i.e. a function can have a function as input), with recursion (Haskell, OCaml, Javascript, Python...), $\mathcal{M}$ is a higher-order tree

over which we run

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

corresponding to a

monadic second-order logic (MSO) formula φ.

(safety, liveness properties, etc)

Model-checking higher-order programs

For functional programs (i.e. a function can have a function as input), with recursion (Haskell, OCaml, Javascript, Python...), $\mathcal{M}$ is a higher-order tree

over which we run

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

corresponding to a

monadic second-order logic (MSO) formula φ.

(safety, liveness properties, etc)

Can we decide whether a higher-order tree satisfies a MSO formula?

Higher-order recursion schemes

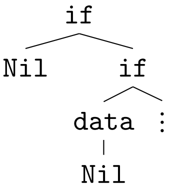
Higher-order recursion schemes

$$\begin{array}{rcl} \texttt{Main} & = & \texttt{Listen Nil} \\ \texttt{Listen}\ x & = & \texttt{if}\ \mathit{end}\ \texttt{then}\ x\ \texttt{else Listen (data}\ x\texttt{)} \end{array}$$

is abstracted as

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

which produces (how ?) the higher-order tree of actions



Higher-order recursion schemes

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

Sort of deterministic higher-order grammar providing a finite representation of higher-order trees.

Rewrite rules have (higher-order) parameters.

"Everything" is simply-typed.

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

Higher-order recursion schemes

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

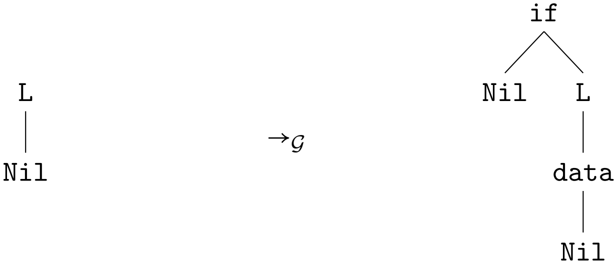
Rewriting starts from the start symbol S:

$$\mathtt{S} \quad \to_{\mathcal{G}} \quad \mathtt{L}(\mathtt{Nil})$$

L
|
Nil

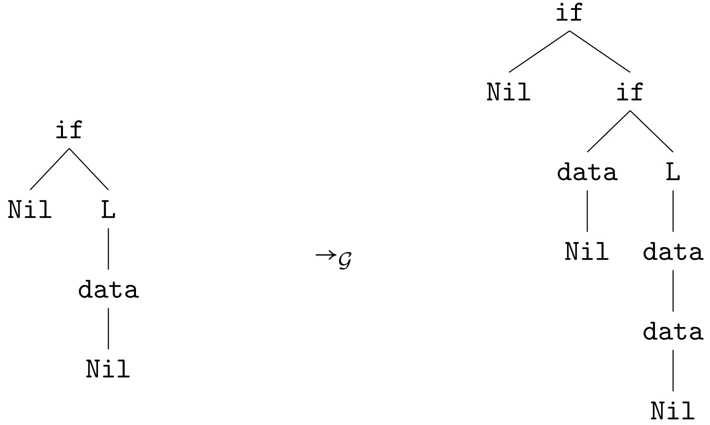
Higher-order recursion schemes

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



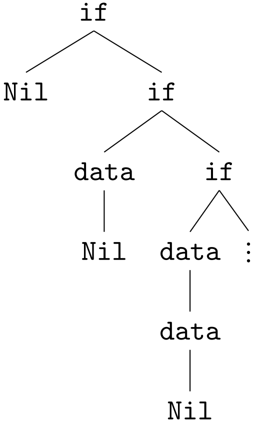
Higher-order recursion schemes

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



Higher-order recursion schemes

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



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

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

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 : $\mathcal{G} \rightsquigarrow (Y_{o \rightarrow o}(\lambda \mathtt{L}.\lambda x.\mathtt{if}\ x\ (\mathtt{L}(\mathtt{data}\ x))))\ \mathtt{Nil}$

Higher-order recursion schemes

In general, many reductions could be used to compute (prefixes of) $\langle \mathcal{G} \rangle$:

$$L\ x\ y \ = \ a\ (M\ (N\ x))\ (P\ y)$$

$$L\ b\ c \qquad \rightarrow_{\mathcal{G}}$$

$\langle \mathcal{G} \rangle$ is computed by the head reduction $\rightarrow_{\mathcal{G}}^{\infty}$, which reduces coinductively the rules.

Higher-order model-checking

$\langle \mathcal{G} \rangle$ is computed using an infinite amount of substitutions and of rule rewritings:

$$S \;\; \rightarrow_\delta \;\; \rightarrow_\beta^* \;\; \rightarrow_\delta \;\; \cdots \;\; \rightarrow_\beta^* \;\; \rightarrow_\delta \;\; \cdots \;\; \langle \mathcal{G} \rangle$$

We want to decide whether $\langle \mathcal{G} \rangle \vDash \phi$: we need to "backtrack" ϕ coinductively along the reduction.

We design denotational models reflecting on terms the action of the automaton $\mathcal{A}_\phi$ corresponding to ϕ:

- the denotation of a term reflects whether it satisfies ϕ,
- usual invariance under β-reduction (inductive backtracking),
- invariance under δ-reduction (coinductive backtracking).

Alternating tree automata

Alternating parity tree automata

For a MSO formula φ,

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

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

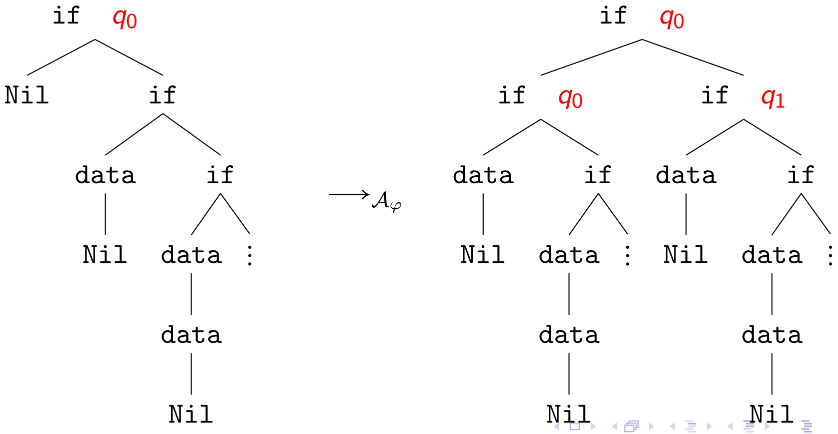
ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

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



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 linear logic

$$A \to B \quad = \quad !A \multimap B$$

A program of type $A \to B$

duplicates or drops elements of A

and then

uses linearly (= once) each copy

Just as alternating automata!

Alternating tree automata and linear logic

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

We obtain finitary semantics (Scott semantics): set $[\![o]\!] = Q$.

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

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

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

Order closure

Alternating tree automata and linear logic

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

We obtain finitary semantics (Scott semantics): set $[\![o]\!] = Q$.

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

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

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

Order closure

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

translates as

$$(\emptyset,\, \{q_0,\, q_1\}\,, q_0) \in [\![\texttt{if}]\!]$$

which notably implies

$$(\{q_0\}\,,\, \{q_0,\, q_1\}\,, q_0) \in [\![\texttt{if}]\!]$$

Scott semantics and tree automata

These semantics are (prime algebraic) lattice semantics, and admit a greatest fixpoint (coinductive), which interprets $\rightarrow_\delta$.

The model is parameterized by $\mathcal{A}_\phi$. We obtain:

Theorem

$\langle \mathcal{G} \rangle \vDash \phi$ *iff* $q_0 \in [\![S]\!]$ *(in the model parameterized by* $\mathcal{A}_\phi$*).*

No parity condition $\Rightarrow$ ϕ is a weak MSO formula.

Corollary: decidability for weak MSO.

Parity conditions

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 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:

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

The coloring comonad

Our work shows that coloring is a modality.
It defines a comonad in the semantics:

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

which can be composed with !, so that

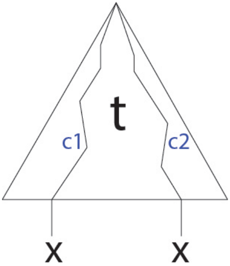
$$\delta(q_0, \texttt{if}) \;=\; (2, q_0) \wedge (2, q_1)$$

now corresponds to

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

in the semantics.

Parity conditions



In this setting, t has some type $\Box_{c_1}\ \sigma_1 \wedge \Box_{c_2}\ \sigma_2 \rightarrow \tau$.

The color labelling each occurence is the maximal color leading to it in the normal form of t.

On applications, the comonad computes the maximal color (inductive treatment).

An inductive-coinductive fixpoint operator

We define an inductive-coinductive fixpoint operator on denotations, which composes inductively or coinductively elements of the semantics, according to the current color.

It is a Conway operator (cf. Z. Esik's talk).

Theorem (G.-Melliès 2015)

For a MSO formula ϕ, $\langle \mathcal{G} \rangle \models \phi$ iff $q_0 \in [[\mathcal{G}]]$ (parameterized by $\mathcal{A}_\phi$).

Corollary

The higher-order model-checking problem is decidable.

(since the semantics of a recursion scheme induce a finite parity game).

The selection problem

Even better: the selection problem is decidable.

If $\mathcal{A}_\phi$ accepts $\langle \mathcal{G} \rangle$,

there is a higher-order accepting run-tree of $\mathcal{A}_\phi$ over $\langle \mathcal{G} \rangle$,

and we can effectively compute a HORS reducing to $\langle \mathcal{G} \rangle$.

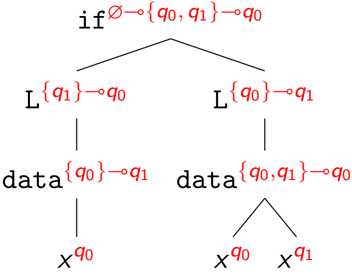
(the key: annotate the rules with their denotation).

The selection problem

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

becomes e.g.

$$\begin{cases} \texttt{S}^{q_0} & = & \texttt{L}^{\{q_0,\,q_1\} \multimap q_0}\ \texttt{Nil}^{q_0}\ \texttt{Nil}^{q_1} \\ \texttt{L}^{\{q_0,\,q_1\} \multimap q_0} & = & \lambda x^{\{q_0,\,q_1\}}. \\ \texttt{L}^{\{q_0\} \multimap q_1} & = & \cdots \\ \texttt{L}^{\{q_1\} \multimap q_0} & = & \cdots \end{cases}$$



Conclusion

- Sort of static analysis of infinitary properties.
- We lift to higher-order the behavior of APT.
- Coloring is a modality, stable by reduction in some sense, and can therefore be added to models and type systems.
- In these finitary semantics, we obtain decidability of HOMC and of the selection problem.
- In the proceedings: the technical aspects, and an equivalent intersection type system.

Thank you for your attention!

Conclusion

- Sort of static analysis of infinitary properties.
- We lift to higher-order the behavior of APT.
- Coloring is a modality, stable by reduction in some sense, and can therefore be added to models and type systems.
- In these finitary semantics, we obtain decidability of HOMC and of the selection problem.
- In the proceedings: the technical aspects, and an equivalent intersection type system.

Thank you for your attention!