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

Charles Grellois   Paul-André Melliès

PPS & LIAFA — Université Paris 7
University of Dundee

CSL — September 8, 2015
TU Berlin

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

A well-known approach in verification: model-checking.

- Construct a model $\mathcal{M}$ of a program $\rightarrow$ higher-order trees
- Specify a property φ 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 φ to an equivalent automaton:

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

Model-checking higher-order programs

A well-known approach in verification: model-checking.

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

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

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

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

Model-checking higher-order programs

A well-known approach in verification: model-checking.

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

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

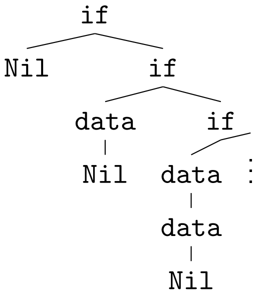
When the model is a word, a tree. . . of actions: translate φ to an equivalent automaton:

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

$\rightarrow$ alternating parity tree automata (APT)

Trees and types

Model-checking of infinite trees of actions:

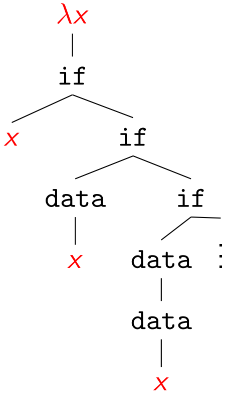


Three actions here: $\Sigma = \{ \texttt{if} : 2, \texttt{data} : 1, \texttt{Nil} : 0 \}$.

Call o the type of trees (and more generally of terms with free variables of order ≤ 1, given by Σ)

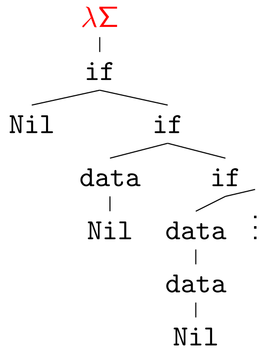
Trees and types

An element of type $o \rightarrow o$:



Applying it to Nil gives the previous tree.

Trees and types



where "$\lambda\Sigma$" stands for $\lambda \texttt{if}.\, \lambda \texttt{data}.\, \lambda \texttt{Nil}.$, has type:

$$o(\Sigma) \rightarrow o \quad = \quad (o \rightarrow o \rightarrow o) \rightarrow (o \rightarrow o) \rightarrow o \rightarrow o$$

Church encoding of trees.

Linear decomposition of the intuitionnistic arrow

In linear logic,

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

$!A$ allows to duplicate or to drop A

$\multimap$ uses linearly (once) each copy

Linear decomposition of the intuitionnistic arrow

$$(o \rightarrow o \rightarrow o) \rightarrow (o \rightarrow o) \rightarrow o \rightarrow o$$

translates as

$$!\,(!\,o \multimap !\,o \multimap o) \multimap !\,(!\,o \multimap o) \multimap !\,o \multimap o$$

In the relational semantics of linear logic, with $[\![o]\!] = Q$,

$$[\![!\,A]\!] = \mathcal{M}_{fin}([\![A]\!]) \quad \text{and} \quad [\![A \multimap B]\!] = [\![A]\!] \times [\![B]\!]$$

For instance,

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

Linear decomposition of the intuitionnistic arrow

$$(o \to o \to o) \to (o \to o) \to o \to o$$

translates as

$$!\,(!\,o \multimap !\,o \multimap o) \multimap !\,(!\,o \multimap o) \multimap !\,o \multimap o$$

Complain: where is model-checking?

We mentioned alternating parity tree automata...

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 $\langle \mathcal{G} \rangle$.

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

Alternating tree automata

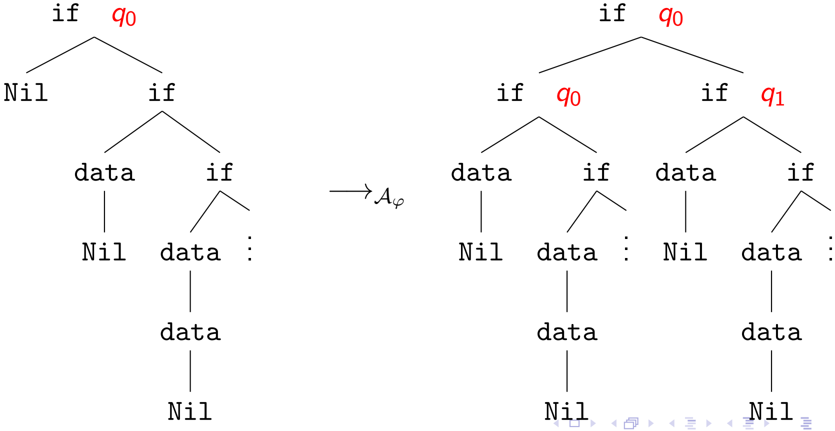
ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

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

In fact, `if` has the linear type

$$\texttt{if} \;:\, !\, o \multimap !\, o \multimap \; o$$

so that in the relational semantics of linear logic, setting $[\![o]\!] \;=\; Q$,

$$[\![\texttt{if}]\!] \subseteq \mathcal{M}_{fin}(Q) \times \mathcal{M}_{fin}(Q) \times Q$$

and

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

Model-checking I

An alternating tree automaton over Σ, with set of states Q, of transition function δ, provides

$$[\![\delta]\!] \quad = \quad [\![\mathtt{if}]\!] \times [\![\mathtt{data}]\!] \times [\![\mathtt{Nil}]\!] \quad \subseteq \quad [\![o(\Sigma)]\!]$$

while a tree t over Σ gives, under Church encoding:

$$[\![t]\!] \quad \subseteq \quad [\![o(\Sigma) \rightarrow o]\!] \quad = \quad \mathcal{M}_{fin}([\![o(\Sigma)]\!]) \times Q$$

Relational composition:

$$[\![t]\!] \circ \mathcal{M}_{fin}([\![\delta]\!]) \quad \subseteq \quad Q$$

Model-checking I

An alternating tree automaton over Σ, with set of states Q, of transition function δ, provides

$$[\![\delta]\!] \quad = \quad [\![\mathtt{if}]\!] \times [\![\mathtt{data}]\!] \times [\![\mathtt{Nil}]\!] \quad \subseteq \quad [\![o(\Sigma)]\!]$$

while a tree t over Σ gives, under Church encoding:

$$[\![t]\!] \quad \subseteq \quad [\![o(\Sigma) \rightarrow o]\!] \quad = \quad \mathcal{M}_{fin}\,([\![o(\Sigma)]\!]) \times Q$$

Relational composition:

$$[\![t]\!] \,\circ\, \mathcal{M}_{fin}([\![\delta]\!]) \quad \subseteq \quad Q$$

Model-checking I

Relational composition:

$$[\![t]\!] \circ \mathcal{M}_{fin}([\![\delta]\!]) \quad \subseteq \quad Q$$

Proposition

$$[\![t]\!] \circ \mathcal{M}_{fin}([\![\delta]\!])$$

is the set of states q from which

$$\mathcal{A} = \langle \Sigma, Q, \delta, q \rangle$$

accepts the tree t.

Model-checking I

Rel is a denotational model:

$$t \rightarrow_\beta t' \qquad \Longrightarrow \qquad [\![t]\!] = [\![t']\!]$$

Corollary

For a term

$$t \; : \; o(\Sigma) \rightarrow o$$

(= normalizing to a finite Σ-labelled ranked tree),

$$[\![t]\!] \circ \mathcal{M}_{fin}([\![\delta]\!])$$

is the set of states q from which

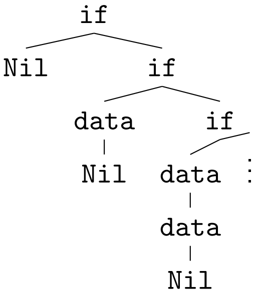
$$\mathcal{A} = \langle \Sigma, Q, \delta, q \rangle$$

accepts the tree $< t >$ *generated by the normalization of* t.

Higher-order model-checking

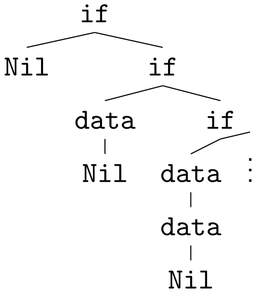
We want to model-check

- higher-order trees ("non-regular, yet of finite representation"), as



- and to account for parity conditions.

Higher-order recursion schemes



is represented as the higher-order recursion scheme (HORS)

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

Higher-order recursion schemes

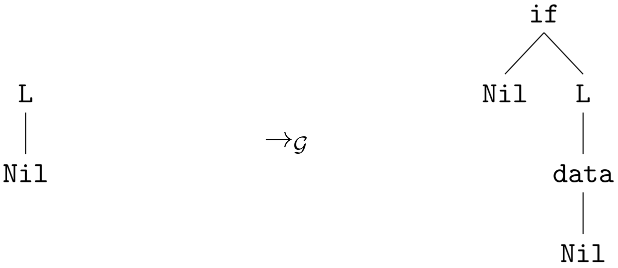
$$\mathcal{G} = \begin{cases} \texttt{S} & = \texttt{L Nil} \\ \texttt{L } x & = \texttt{if } x \, (\texttt{L } (\texttt{data } x\,)\,) \end{cases}$$

Rewriting starts from the start symbol S:

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

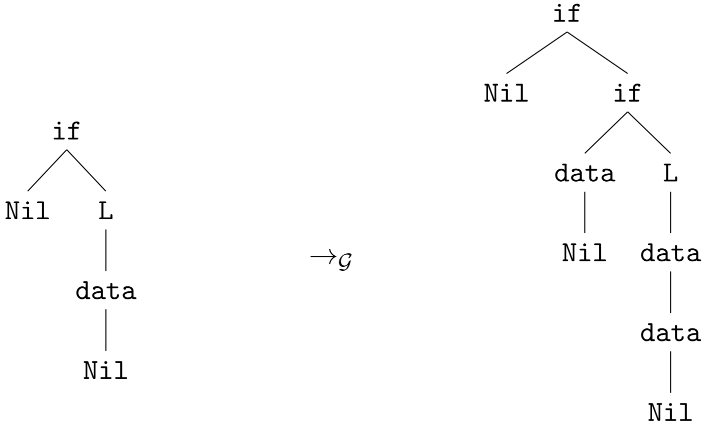
Higher-order recursion schemes

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



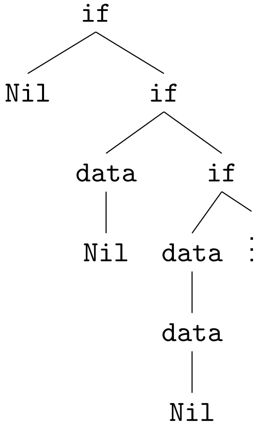
Higher-order recursion schemes

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



Higher-order recursion schemes

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



$\langle \mathcal{G} \rangle$ 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 an extension of the simply-typed λ-terms we considered so far, with

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 need to add fixpoints to the relational model.

Model-checking II

Rel has an inductive fixpoint operator (finite iteration). We obtain:

Theorem

For a λY-term

$$t \ : \ o(\Sigma) \rightarrow o$$

(= normalizing to an infinite Σ-labelled ranked tree),

$$[\![t]\!] \ \circ \ \mathcal{M}_{fin}([\![\delta]\!])$$

is the set of states q from which

$$\mathcal{A} \ = \ \langle \, \Sigma, \ Q, \ \delta, \ q \, \rangle$$

accepts the tree $< t >$ generated by the coinductive normalization of t

during a finite execution

On finiteness

Why a finite execution?

Because constructors = free variables.

Infinite trees need infinite multisets.

So we define a new exponential

$$\lightning \ : \ A \mapsto \mathcal{M}_{count}(A)$$

The resulting model has a coinductive operator ($\approx$ infinite fixpoint unfolding).

(see G.-Melliès, Fossacs 2015)

Model-checking III

With the coinductive fixpoint of this infinitary model:

Theorem

For a λY-term

$$t \ : \ o(\Sigma) \rightarrow o$$

($=$ normalizing to an infinite Σ-labelled ranked tree),

$$[\![t]\!] \ \circ \ \mathcal{M}_{fin}([\![\delta]\!])$$

is the set of states q from which

$$\mathcal{A} \ = \ \langle \, \Sigma, \ Q, \ \delta, \ q \, \rangle$$

accepts the tree $< t >$ generated by the coinductive normalization of t

during a finite or infinite execution

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

In the proceedings paper, we show that coloring is a modality.
It defines a comonad in the semantics:

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

which can be composed with $\lightning$, giving an infinitary, colored model of linear logic in which

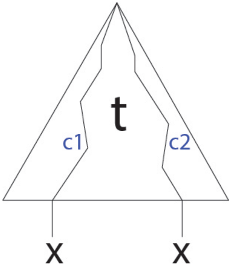
$$\delta(q_0, \texttt{if}) \ = \ (2, q_0) \wedge (2, q_1)$$

corresponds to

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

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).

Model-checking IV

We define an inductive-coinductive fixpoint operator on denotations, which iterates finitely or infinitely depending on the current color.
It is a Conway operator (Bloom-Esik).

Theorem

For a λY-term

$$t \;:\; o(\Sigma) \rightarrow o$$

($=$ normalizing to an infinite Σ-labelled ranked tree),

$$[\![t]\!] \;\circ\; \mathcal{M}_{fin}([\![\delta]\!])$$

is the set of states q from which the alternating parity automaton

$$\mathcal{A} = \langle\, \Sigma,\, Q,\, \delta,\, q \,\rangle$$

accepts the tree $< t >$ generated by the coinductive normalization of t.

Model-checking V

Ehrhard 2012: *ScottL* is the extensional collapse of *Rel*.

G.-Melliès, MFCS 2015: adaptation to *ScottL* of the theoretical approach of this work.

Corollary

The higher-order model-checking problem is decidable.

The resulting model is similar in the spirit to the one of Salvati and Walukiewicz, with subtle differences, notably on color handling and composition of morphisms.

Conclusion

- Linear logic reveals a very natural duality between terms and (alternating) automata.
- Models can be extended to handle additional conditions on automata (parity. . .)
- Relational semantics are infinitary, but their simplicity eases theoretical reasoning on problems.

In the proceedings:

- More on the duality aspects, and on the extended relational semantics.
- Discussion on the modal nature of coloring, and its relations with prior work of Kobayashi and Ong.
- Technical work is based on an equivalent intersection type system.

Thank you for your attention!

Conclusion

- Linear logic reveals a very natural duality between terms and (alternating) automata.
- Models can be extended to handle additional conditions on automata (parity. . .)
- Relational semantics are infinitary, but their simplicity eases theoretical reasoning on problems.

In the proceedings:

- More on the duality aspects, and on the extended relational semantics.
- Discussion on the modal nature of coloring, and its relations with prior work of Kobayashi and Ong.
- Technical work is based on an equivalent intersection type system.

Thank you for your attention!