An infinitary model of linear logic

Charles Grellois (joint work with Paul-André Melliès)

PPS & LIAFA — Université Paris 7

April 13th, 2015

Model-checking higher-order programs

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

- Construct a model of a program
- Specify a property in an appropriate logic
- Make them interact in order to determine whether the program satisfies the property.

Interaction is often realized by translating the formula into an equivalent automaton, which then runs over the model.

Model-checking higher-order programs

For higher-order programs with recursion, a natural model is

higher-order recursion schemes (HORS)

which generate a tree abstracting the set of potential behaviors of a program, and over which we want to run

alternating parity automata (APT)

in order to check whether some MSO formula holds.

Model-checking higher-order programs

For higher-order programs with recursion, a natural model is

higher-order recursion schemes (HORS)

which generate a tree abstracting the set of potential behaviors of a program, and over which we want to run

alternating parity automata (APT)

in order to check whether some MSO formula holds.

Model-checking higher-order programs

This model-checking problem is decidable:

- Ong 2006 (game semantics)
- Hague-Murawski-Ong-Serre 2008 (game semantics + collapsible higher-order pushdown automata)
- Kobayashi-Ong 2009 (intersection types)
- Salvati-Walukiewicz 2011 (interpretation with Krivine machines)
- Carayol-Serre 2012 (collapsible higher-order pushdown automata)
- Tsukada-Ong 2014 (game semantics)
- Salvati-Walukiewicz 2015 (interpretation in finite models)
- Grellois-Melliès 2015

As we will see, the challenge is to understand how an automaton acts at higher-order, directly on terms.

In this talk: we study of the problem at the light of the relational semantics of linear logic.

Model-checking higher-order programs

This model-checking problem is decidable:

- Ong 2006 (game semantics)
- Hague-Murawski-Ong-Serre 2008 (game semantics + collapsible higher-order pushdown automata)
- Kobayashi-Ong 2009 (intersection types)
- Salvati-Walukiewicz 2011 (interpretation with Krivine machines)
- Carayol-Serre 2012 (collapsible higher-order pushdown automata)
- Tsukada-Ong 2014 (game semantics)
- Salvati-Walukiewicz 2015 (interpretation in finite models)
- Grellois-Melliès 2015

As we will see, the challenge is to understand how an automaton acts at higher-order, directly on terms.

In this talk: we study of the problem at the light of the relational semantics of linear logic.

Higher-order recursion schemes

A very simple functional program

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

With a recursion scheme we can model this program and produce its tree of behaviours.

Note that constants are not interpreted: in particular, a recursion scheme does not evaluate a boolean conditional `if ... then ... else ...`

A very simple functional program

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

is modelled as a recursion scheme:

$$\begin{array}{rcl} \texttt{S} & = & \texttt{L Nil} \\ \texttt{L } x & = & \texttt{if } x \,(\texttt{L } (\texttt{data } x\,)\,) \end{array}$$

Value tree of a recursion scheme

$$\begin{array}{rcl} \mathtt{S} & = & \mathtt{L\ Nil} \\ \mathtt{L}\ x & = & \mathtt{if}\ x\,(\mathtt{L}\ (\mathtt{data}\ x\,)\,) \end{array}$$

generates:

S

Value tree of a recursion scheme

$$\begin{array}{lcl} \texttt{S} & = & \texttt{L Nil} \\ \texttt{L } x & = & \texttt{if } x \,(\texttt{L } (\texttt{data } x\,)\,) \end{array}$$

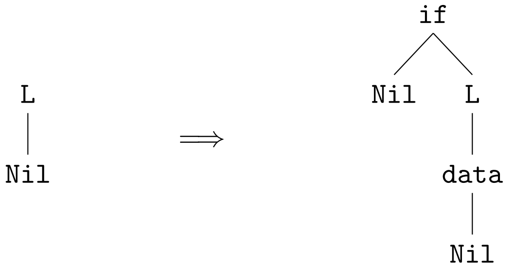
generates:

$\texttt{S} \quad \Longrightarrow$

Value tree of a recursion scheme

$$\begin{array}{lcl} \mathtt{S} & = & \mathtt{L\ Nil} \\ \mathtt{L}\ x & = & \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{array}$$

generates:

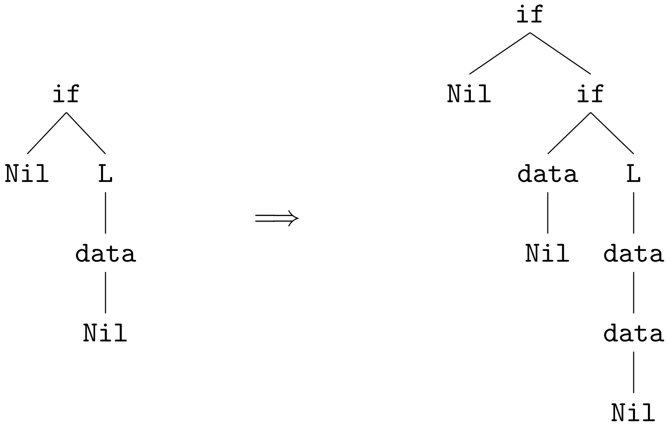


Notice that substitution and expansion occur in one same step.

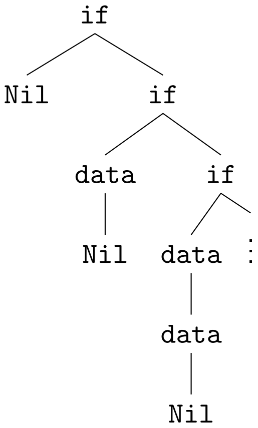
Value tree of a recursion scheme

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

generates:



Value tree of a recursion scheme



Very simple program, yet it produces a tree which is not regular. . .

Representation of recursion schemes

The only finite representation of such a tree is actually the scheme itself — even for this very simple, order-1 recursion scheme.

So, in order to get a decidability proof of the result, we need to analyze the recursion scheme itself, and to predict the behaviour of the automaton directly over it.

In the sequel, we will consider the equivalent formalism of λ-terms with a recursion operator Y.

Representation of recursion schemes

The only finite representation of such a tree is actually the scheme itself — even for this very simple, order-1 recursion scheme.

So, in order to get a decidability proof of the result, we need to analyze the recursion scheme itself, and to predict the behaviour of the automaton directly over it.

In the sequel, we will consider the equivalent formalism of λ-terms with a recursion operator Y.

Alternating tree automata

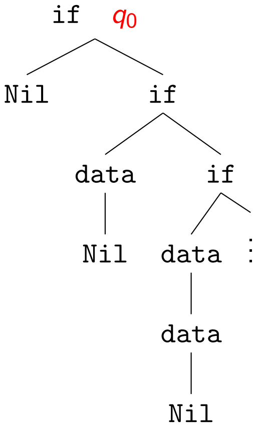
Alternating tree automata (ATA) are non-deterministic tree automata whose transitions may duplicate or drop a subtree.

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

This is reminiscent of the behavior of the exponential modality of linear logic...

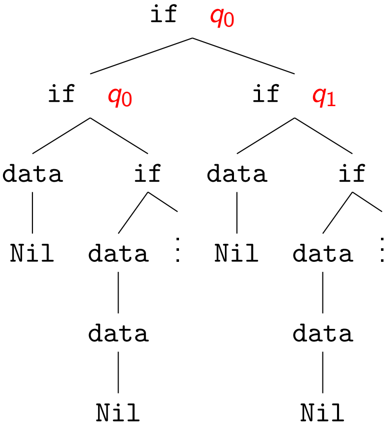
Alternating tree automata

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



Alternating tree automata

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



and so on. This gives the notion of run-tree. They are unranked.

Alternating tree automata and intersection types

A key remark (Kobayashi 2009): if $\delta(q, a) = (1, q_0) \wedge (1, q_1) \wedge (2, q_2) \ldots$

then we may consider that a has a refined intersection type

$$(q_0 \wedge q_1) \Rightarrow q_2 \Rightarrow q$$

In previous work, we studied these intersection types at the light of indexed linear logic, and of its relational semantics.

The intersection operation acts as a uniform exponential: it duplicates resources corresponding to the same term.

Alternating tree automata and intersection types

A key remark (Kobayashi 2009): if $\delta(q, a) \;=\; (1, q_0) \wedge (1, q_1) \wedge (2, q_2)\ldots$

then we may consider that a has a refined intersection type

$$(q_0 \wedge q_1) \Rightarrow q_2 \Rightarrow q$$

In previous work, we studied these intersection types at the light of indexed linear logic, and of its relational semantics.

The intersection operation acts as a uniform exponential: it duplicates resources corresponding to the same term.

Alternating tree automata and intersection types

A key remark (Kobayashi 2009): if $\delta(q, a) \; = \; (1, q_0) \wedge (1, q_1) \wedge (2, q_2) \ldots$

then we may consider that a has a refined intersection type

$$(q_0 \wedge q_1) \Rightarrow q_2 \Rightarrow q$$

In previous work, we studied these intersection types at the light of indexed linear logic, and of its relational semantics.

The intersection operation acts as a uniform exponential: it duplicates resources corresponding to the same term.

Relational semantics and alternating tree automata

An inductive fixpoint in relational semantics

To interpret the recursion of schemes in the semantics, we need to interpret the rule

$$\frac{!\,X \otimes !\,A \vdash A}{!\,X \vdash A} \quad \textit{fix}$$

by a suitable collection of morphisms

$$\mathbf{Y}_{X,A} \;:\; \mathscr{C}(!\,X \otimes !A\,,\,A\,) \quad \longrightarrow \quad \mathscr{C}(!\,X, A)$$

X is a set of parameters (used to collect the free variables, which correspond here to the tree constructors).

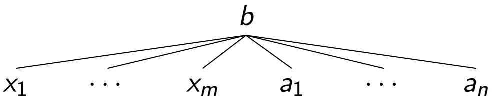
Here we focus on the case $\mathscr{C} \;=\; \mathit{Rel}$.

An inductive fixpoint in relational semantics

The semantics of

$$f \quad : \quad !\,X \otimes !\,A \quad \multimap \quad A$$

can be seen as a set of tree constructors



We call such a tree a witness tree. If it is finite, we say that it is winning.

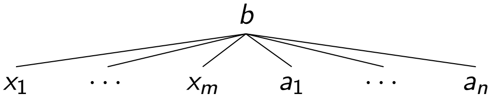
Then the fixpoint $Y\,f \; : \, !\,X \multimap A$ maps multisets of parameters $[y_1, \ldots, y_k]$ to elements $b \in A$ such that there is a winning witness-tree with root labelled b and multiset of leaves $[y_1, \ldots, y_k]$

An inductive fixpoint in relational semantics

The semantics of

$$f \quad : \quad !X \otimes !A \quad \multimap \quad A$$

can be seen as a set of tree constructors



We call such a tree a witness tree. If it is finite, we say that it is winning.

Then the fixpoint $Y\,f \;:\; !X \multimap A$ maps multisets of parameters $[y_1, \ldots, y_k]$ to elements $b \in A$ such that there is a winning witness-tree with root labelled b and multiset of leaves $[y_1, \ldots, y_k]$

An inductive fixpoint in relational semantics

This fixpoint is "suitable", meaning that it satisfies a series of equations introduced by Bloom and Esik (see also Simpson and Plotkin).

These equations may be understood as

- a branching version of the equations of Wilke algebras
- with additional conditions on the handling of the parameters.

They come from a categorical study of the properties of usual fixpoint operators on domains.

We obtain that Y is a Conway operator.

Relational interpretation and automata acceptance

In the resulting model of λY-calculus, we obtain a semantic model-checking theorem:

Theorem (G.-Melliès 2014)

Consider an alternating tree automaton $\mathcal{A}$ and a λY-term t reducing to (the Church encoding of) a tree T.

Then $\mathcal{A}$ has a finite run-tree over T if and only if

$$q_0 \ \in \ [\![t]\!] \circ [\![\delta]\!]$$

where the interpretation is computed in the relational model, the base type (of trees) being interpreted as Q.

In other words: the dual interpretations of a term and of an automaton interact to compute the set of accepting states of the automaton over the tree generated by the term.

Relational interpretation and automata acceptance

In the resulting model of λY-calculus, we obtain a semantic model-checking theorem:

Theorem (G.-Melliès 2014)

Consider an alternating tree automaton $\mathcal{A}$ and a λY-term t reducing to (the Church encoding of) a tree T.

Then $\mathcal{A}$ has a finite run-tree over T if and only if

$$q_0 \in [\![t]\!] \circ [\![\delta]\!]$$

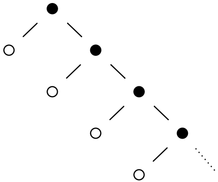
where the interpretation is computed in the relational model, the base type (of trees) being interpreted as Q.

In other words: the dual interpretations of a term and of an automaton interact to compute the set of accepting states of the automaton over the tree generated by the term.

Fixed point and alternating tree automata

A morphism f induces an alternating automaton which we run over

comb =



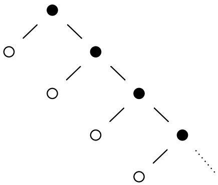
$$\delta(b, \bullet) = \bigvee \; (\,(1, x_1) \wedge \cdots \wedge (1, x_n) \wedge (2, a_1) \wedge \cdots \wedge (2, a_m))$$

where $(([x_1, \cdots, x_n]), [a_1, \cdots, a_m]), b) \in f$.

Finite run-trees $\Longleftrightarrow$ fixed point computations

Fixed point and alternating tree automata

A morphism f induces an alternating automaton which we run over

comb $=$ 

$$\delta(b, \bullet) = \bigvee \; (\,(1, x_1) \wedge \cdots \wedge (1, x_n) \wedge (2, a_1) \wedge \cdots \wedge (2, a_m))$$

where $(([x_1, \cdots, x_n]), [a_1, \cdots, a_m]), b) \in f$.

Finite run-trees $\Longleftrightarrow$ fixed point computations

An infinitary model of linear logic

An infinitary relational semantics

An infinite run-tree uses countably some elements of the signature.

We therefore need to introduce a variant of the relational semantics of linear logic, in which objects are set of cardinality at most the reals, and we define a new exponential modality $\lightning$:

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

(finite-or-countable multisets)

This comonad $\lightning$ satisfies the axioms of an exponential, and thus gives immediately an infinitary model of the λ-calculus by the Kleisli construction.

An infinitary relational semantics

An infinite run-tree uses countably some elements of the signature.

We therefore need to introduce a variant of the relational semantics of linear logic, in which objects are set of cardinality at most the reals, and we define a new exponential modality $\lightning$:

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

(finite-or-countable multisets)

This comonad $\lightning$ satisfies the axioms of an exponential, and thus gives immediately an infinitary model of the λ-calculus by the Kleisli construction.

An infinitary relational semantics

Now that we can interpret infinite trees, all we need is to change the winning condition of witness trees.

Let us consider all witness trees as accepting. We obtain the gfp (coinductive interpretation), mapping a morphism

$$f \; : \; \lightning A \otimes \lightning X \multimap A$$

to a morphism

$$Y\, f \; : \; \lightning X \multimap A$$

It is again a Conway operator.

An infinitary relational semantics

The Theorem then extends:

Theorem (G.-Melliès 2014)

Consider an alternating tree automaton $\mathcal{A}$ and a λY-term t producing (the Church encoding of) a tree T.

Then $\mathcal{A}$ has a possibly infinite run-tree over T if and only if

$$q_0 \in [\![t]\!] \circ [\![\delta]\!]$$

where the recursion operator of the λY-calculus is computed using this coinductive fixed point operator.

An infinitary relational semantics

The Theorem then extends:

Theorem (G.-Melliès 2014)

Consider an alternating tree automaton $\mathcal{A}$ and a λY-term t producing (the Church encoding of) a tree T.

Then $\mathcal{A}$ has a possibly infinite run-tree over T if and only if

$$q_0 \in [\![t]\!] \circ [\![\delta]\!]$$

where the recursion operator of the λY-calculus is computed using this coinductive fixed point operator.

Specifying inductive and coinductive behaviours: 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

In the APT, this inductive-coinductive policy is encoded using parity conditions. Every state receives a colour

$$\Omega(q) \in Col \subseteq \mathbb{N}$$

Say that an infinite branch of a run-tree is winning iff the maximal colour among the ones occuring infinitely often along it is even.

Say that a run-tree is winning iff all of its infinite branches are.

Then an APT has a winning run-tree over a tree T iff the root of T satisfies the corresponding MSO formula ϕ.

Alternating parity tree automata

In the APT, this inductive-coinductive policy is encoded using parity conditions. Every state receives a colour

$$\Omega(q) \in Col \subseteq \mathbb{N}$$

Say that an infinite branch of a run-tree is winning iff the maximal colour among the ones occuring infinitely often along it is even.

Say that a run-tree is winning iff all of its infinite branches are.

Then an APT has a winning run-tree over a tree T iff the root of T satisfies the corresponding MSO formula ϕ.

The coloring parametric comonad

From a study of the Kobayashi-Ong 2009 approach, we discovered that the coloring operation behaves as a

parametric comonad

Informally, there is

- a neutral color ϵ, corresponding to the absence of box,
- and a composition mechanism which computes the maximum color of (finitely many) boxes.

It can be incorporated to the infinitary relational model, setting

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

(and adding an appropriate collection of structural morphisms).

Colored relational semantics

A distributivity law between $\lightning$ and $\Box$ allows to consider their composition as a new exponential.

$$\boldsymbol{\lightning} \quad = \quad \lightning\,\Box$$

Elements of the semantics are now colored:

We can thus recast the parity condition on such trees. This adaptation of the winning condition of run-trees defines an inductive-coinductive fixpoint operator.

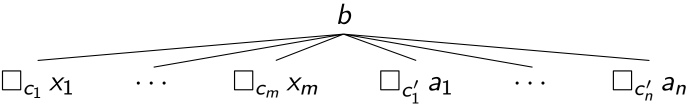
It is a Conway operator.

Colored relational semantics

A distributivity law between $\lightning$ and $\Box$ allows to consider their composition as a new exponential.

$$\boldsymbol{\lightning} \quad = \quad \lightning \, \Box$$

Elements of the semantics are now colored:



We can thus recast the parity condition on such trees. This adaptation of the winning condition of run-trees defines an inductive-coinductive fixpoint operator.

It is a Conway operator.

Colored relational semantics

Theorem (G.-Melliès 2015)

Consider an alternating parity tree automaton $\mathcal{A}$ and a λY-term t producing (the Church encoding of) a tree T.

Then $\mathcal{A}$ has a winning run-tree over T if and only if

$$q_0 \;\in\; [\![t]\!]_{col} \circ [\![\delta]\!]_{col}$$

A finitary colored model of the λY-calculus

The Scott model of linear logic

In order to get a decidability proof, we need to recast our approach in a finitary setting.

If the exponential modality ! is interpreted with finite sets, we obtain the poset-based model of linear logic (a.k.a. its Scott model).

Ehrhard proved in 2012 that it is the extensional collapse of the relational model.

We could make this adaptation and obtain a new decidability proof.

The Scott model of linear logic

In order to get a decidability proof, we need to recast our approach in a finitary setting.

If the exponential modality ! is interpreted with finite sets, we obtain the poset-based model of linear logic (a.k.a. its Scott model).

Ehrhard proved in 2012 that it is the extensional collapse of the relational model.

We could make this adaptation and obtain a new decidability proof.

Conclusions and perspectives

- The behavior of an alternating parity tree automaton can be adapted to the semantics of the term computing the tree over which it runs.
- We introduce an infinitary, colored relational semantics to interpret winning APT run-trees.
- The fixpoint operator combines inductive and coinductive interpretations depending on the color.
- Current work: relate this automata-theoretic definition of the fixpoint with one interleaving μ and ν.

$$Y f \quad = \quad \nu x_n \mu x_{n-1} \cdots \nu x_2 \,.\, \mu x_1 \,.\, \nu x_0 . f(x_0, \ldots, x_n)$$

Conclusions and perspectives

- The behavior of an alternating parity tree automaton can be adapted to the semantics of the term computing the tree over which it runs.
- We introduce an infinitary, colored relational semantics to interpret winning APT run-trees.
- The fixpoint operator combines inductive and coinductive interpretations depending on the color.
- Current work: relate this automata-theoretic definition of the fixpoint with one interleaving μ and ν.

$$Yf \quad = \quad \nu x_n \mu x_{n-1} \cdots \nu x_2 \, . \, \mu x_1 \, . \, \nu x_0 . f(x_0, \ldots, x_n)$$