A semantic study of higher-order model-checking

Charles Grellois   Paul-André Melliès

PPS & LIAFA — Université Paris 7
University of Dundee

University of Aarhus
November 2nd, 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

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

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

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

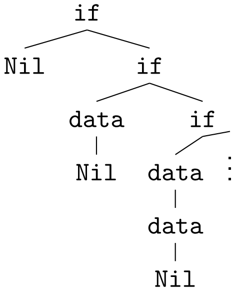
Model-checking higher-order programs

For higher-order programs with recursion, $\mathcal{M}$ is a higher-order tree.

Example:

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

modelled as



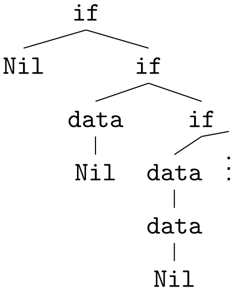
Model-checking higher-order programs

For higher-order programs with recursion, $\mathcal{M}$ is a higher-order tree.

Example:

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

modelled as



How to represent this tree finitely?

Model-checking higher-order programs

For higher-order programs with recursion, $\mathcal{M}$ 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

Some regularity for infinite trees

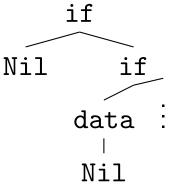
Higher-order recursion schemes

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

is abstracted as

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

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



Higher-order recursion schemes

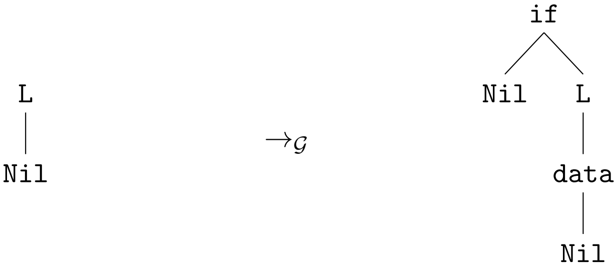
$$\mathcal{G} \;=\; \left\{ \begin{array}{lcl} \mathtt{S} & = & \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L}\ x & = & \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{array} \right.$$

Rewriting starts from the start symbol S:

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

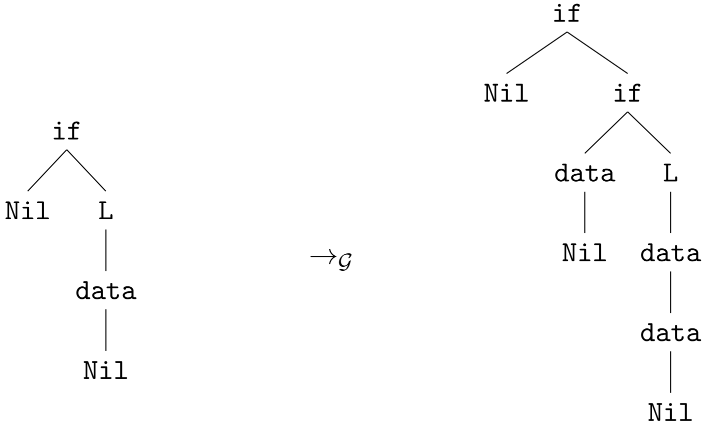
Higher-order recursion schemes

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



Higher-order recursion schemes

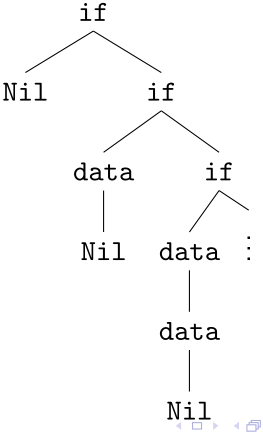
$$\mathcal{G} = \begin{cases} \mathtt{S} & = \quad \mathtt{L}\ \mathtt{Nil} \\ \mathtt{L}\ x & = \quad \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$



Higher-order recursion schemes

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

$\langle \mathcal{G} \rangle \quad =$



Higher-order recursion schemes

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

"Everything" is simply-typed, and

well-typed programs can't go too wrong:

we can detect productivity, and enforce it (replace divergence by outputing a distinguished symbol Ω in one step).

Higher-order recursion schemes

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

"Everything" is simply-typed, and

well-typed programs can't go too wrong:

we can detect productivity, and enforce it (replace divergence by outputing a distinguished symbol Ω in one step).

HORS can alternatively be seen as simply-typed λ-terms with

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

Higher-order recursion schemes

We can adapt to HORS the fact that coinductive parallel head reduction computes the normal form of infinite λ-terms:

$$\frac{}{(\,\lambda x.\,s\,)\ t \quad \rightarrow_{\mathcal{G}w} \quad s[x \leftarrow t]} \qquad \frac{s \quad \rightarrow_{\mathcal{G}w} \quad s'}{s\ t \quad \rightarrow_{\mathcal{G}w} \quad s'\ t}$$

$$\frac{}{F \quad \rightarrow_{\mathcal{G}w} \quad \mathcal{R}(F)}$$

$$\frac{t \quad \rightarrow_{\mathcal{G}w}^{*} \quad a\ t_1 \cdots t_n \qquad t_i \quad \rightarrow_{\mathcal{G}}^{\infty} \quad t_i' \quad (\forall i)}{t \quad \rightarrow_{\mathcal{G}}^{\infty} \quad a\ t_1' \cdots t_n'}$$

This reduction computes $\langle \mathcal{G} \rangle$ whenever it exists (a decidable question).

This presentation allows coinductive reasoning on rewriting.

Modal μ-calculus and alternating parity tree automata

Modal μ-calculus

Over trees we may use several logics: CTL, MSO,. . .

In this work we use modal μ-calculus. It is equivalent to MSO over trees.

Grammar: $\phi, \psi \ ::= \ X \mid a \mid \phi \vee \psi \mid \phi \wedge \psi \mid \Box \phi \mid \ \diamond_i \phi \mid \mu X.\, \phi \mid \nu X.\, \phi$

Modal μ-calculus

Grammar: $\phi, \psi ::= X \mid a \mid \phi \vee \psi \mid \phi \wedge \psi \mid \Box \phi \mid \diamond_i \phi \mid \mu X.\, \phi \mid \nu X.\, \phi$

X is a variable

a is a predicate corresponding to a symbol of Σ

$\Box \phi$ means that ϕ should hold on every successor of the current node

$\diamond_i \phi$ means that ϕ should hold on one successor of the current node (in direction i)

We can also define (variant) $\diamond = \bigvee_i \diamond_i$.

Modal μ-calculus

Grammar: $\phi, \psi \ ::= \ X \mid a \mid \phi \vee \psi \mid \phi \wedge \psi \mid \Box\,\phi \mid \ \diamond_i\,\phi \mid \mu X.\,\phi \mid \nu X.\,\phi$

$\mu X.\,\phi$ is the least fixpoint of $\phi(X)$. It is computed by expanding finitely the formula:

$$\mu X.\,\phi(X) \quad \longrightarrow \quad \phi(\mu X.\,\phi(X)) \quad \longrightarrow \quad \phi(\phi(\mu X.\,\phi(X)))$$

Modal μ-calculus

Grammar: $\phi, \psi \ ::= \ X \mid a \mid \phi \vee \psi \mid \phi \wedge \psi \mid \Box \phi \mid \ \diamond_i \phi \mid \mu X.\, \phi \mid \nu X.\, \phi$

$\nu X.\, \phi$ is the greatest fixpoint of $\phi(X)$. It is computed by expanding infinitely the formula:

$$\nu X.\, \phi(X) \quad \longrightarrow \quad \phi(\nu X.\, \phi(X)) \quad \longrightarrow \quad \phi(\phi(\nu X.\, \phi(X)))$$

Modal μ-calculus

Grammar: $\phi, \psi \ ::= \ X \mid a \mid \phi \vee \psi \mid \phi \wedge \psi \mid \Box \phi \mid \ \diamond_i \phi \mid \mu X.\phi \mid \nu X.\phi$

What does:

$$\phi \ = \ \nu X. \left(\texttt{if} \wedge \diamond_1 \left(\mu Y. \left(\texttt{Nil} \vee \Box Y \right) \right) \wedge \diamond_2 X \right)$$

mean ?

And how does it interact with a tree?

$\rightarrow$ tree automata

Modal μ-calculus

Grammar: $\phi, \psi ::= X \mid a \mid \phi \vee \psi \mid \phi \wedge \psi \mid \Box \phi \mid \diamond_i \phi \mid \mu X.\, \phi \mid \nu X.\, \phi$

What does:

$$\phi = \nu X.\, (\, \texttt{if} \wedge \diamond_1 \, (\, \mu Y.\, (\, \texttt{Nil} \vee \Box Y \,)\,) \wedge \diamond_2 \, X \,)$$

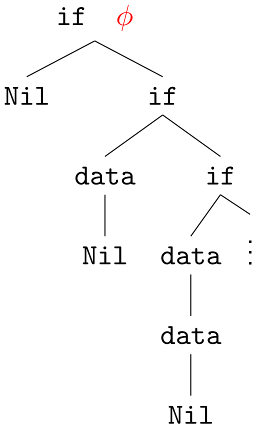
mean ?

And how does it interact with a tree?

$\rightarrow$ tree automata

Alternating parity tree automata

Idea: the formula "starts" on the root

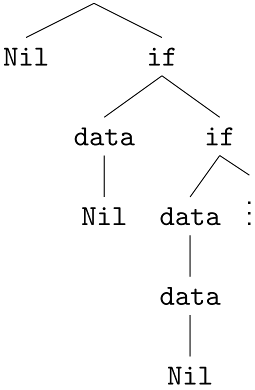


where $\phi \;=\; \nu X.\,(\ \texttt{if} \wedge \diamond_1\ (\ \mu Y.\,(\ \texttt{Nil} \vee \Box Y\ )\ ) \wedge \diamond_2\ X\ )$

Alternating parity tree automata

Idea: the formula "starts" on the root

$\texttt{if}$ $\quad \texttt{if} \wedge \diamond_1 \; (\; \mu Y. \; (\; \texttt{Nil} \vee \Box Y \;)\;) \wedge \diamond_2 \; \phi$

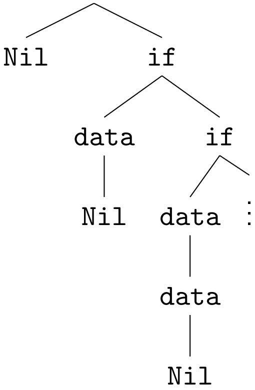


where $\phi \;=\; \nu X. \; (\; \texttt{if} \wedge \diamond_1 \; (\; \mu Y. \; (\; \texttt{Nil} \vee \Box Y \;)\;) \wedge \diamond_2 \; X \;)$

Alternating parity tree automata

Idea: the formula "starts" on the root

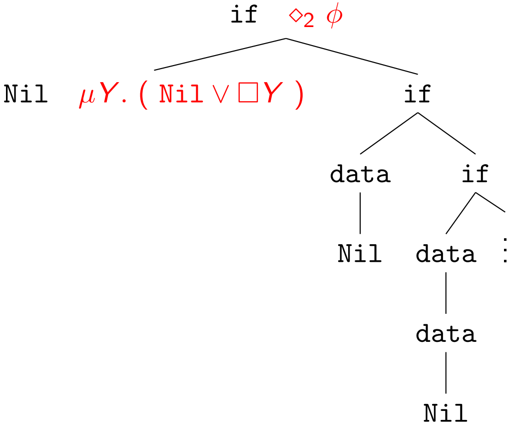
if $\diamond_1 \,(\, \mu Y.\, (\, \texttt{Nil} \vee \Box Y \,)\,) \wedge \diamond_2 \, \phi$



where $\phi \;=\; \nu X.\, (\, \texttt{if} \wedge \diamond_1 \,(\, \mu Y.\, (\, \texttt{Nil} \vee \Box Y \,)\,) \wedge \diamond_2 \, X \,)$

Alternating parity tree automata

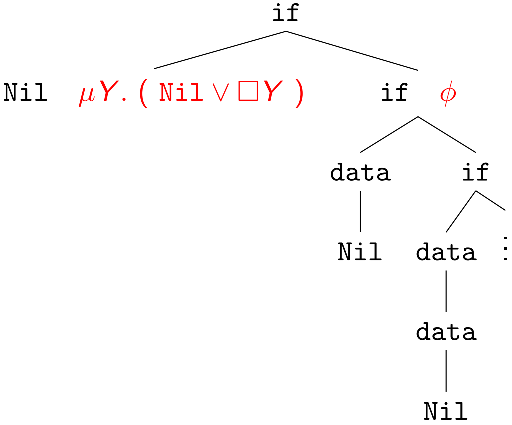
Idea: the formula "starts" on the root



where $\phi\ =\ \nu X.\ (\ \mathtt{if} \wedge \diamond_1\ (\ \mu Y.\ (\ \mathtt{Nil} \vee \square Y\ )\ ) \wedge \diamond_2\ X\ )$

Alternating parity tree automata

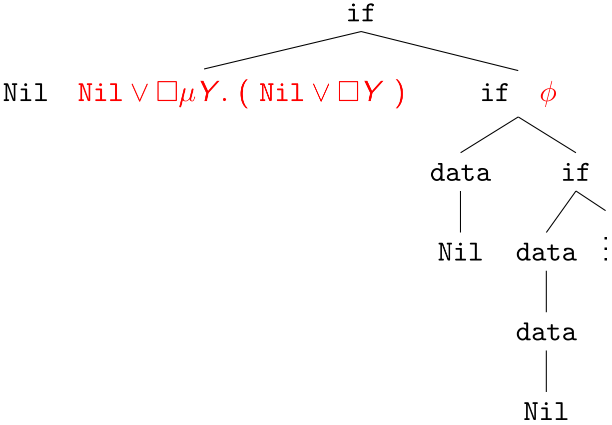
Idea: the formula "starts" on the root



where $\phi\ =\ \nu X.\ (\ \texttt{if} \wedge \diamond_1\ (\ \mu Y.\ (\ \texttt{Nil} \vee \Box Y\ )\ ) \wedge \diamond_2\ X\ )$

Alternating parity tree automata

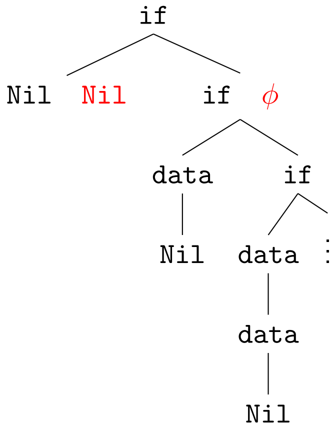
Idea: the formula "starts" on the root



where $\phi \;=\; \nu X.\, (\, \texttt{if} \land \diamond_1\, (\, \mu Y.\, (\, \texttt{Nil} \lor \Box Y\,)\,) \land \diamond_2\, X\,)$

Alternating parity tree automata

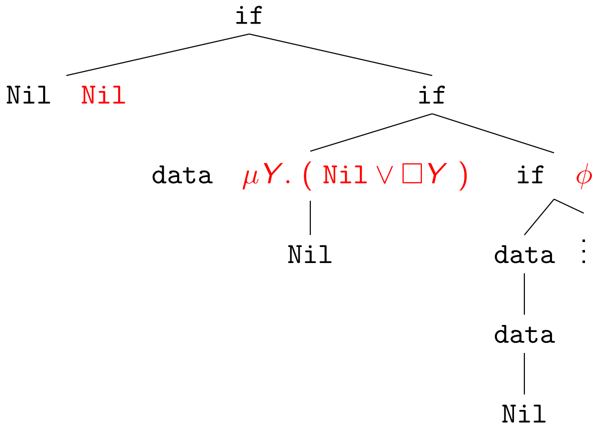
Idea: the formula "starts" on the root



where $\phi = \nu X.\ (\ \mathtt{if} \wedge \diamond_1\ (\ \mu Y.\ (\ \mathtt{Nil} \vee \square Y\ )\ ) \wedge \diamond_2\ X\ )$

Alternating parity tree automata

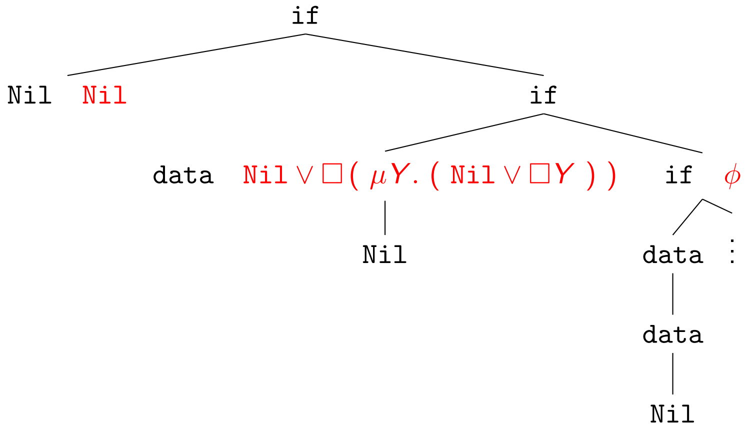
Idea: the formula "starts" on the root



where $\phi \;=\; \nu X.\,(\,\mathtt{if} \wedge \diamond_1\;(\,\mu Y.\,(\,\mathtt{Nil} \vee \Box Y\,)\,) \wedge \diamond_2\, X\,)$

Alternating parity tree automata

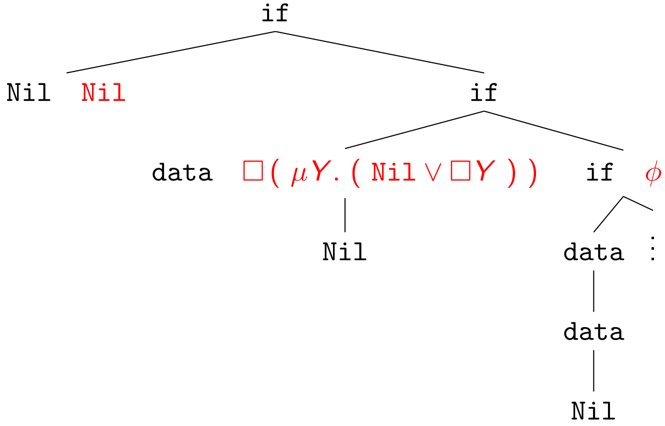
Idea: the formula "starts" on the root



where $\phi \; = \; \nu X. \,(\, \texttt{if} \wedge \diamond_1 \;(\, \mu Y. \,(\, \texttt{Nil} \vee \Box Y \,)\,) \wedge \diamond_2 \, X \,)$

Alternating parity tree automata

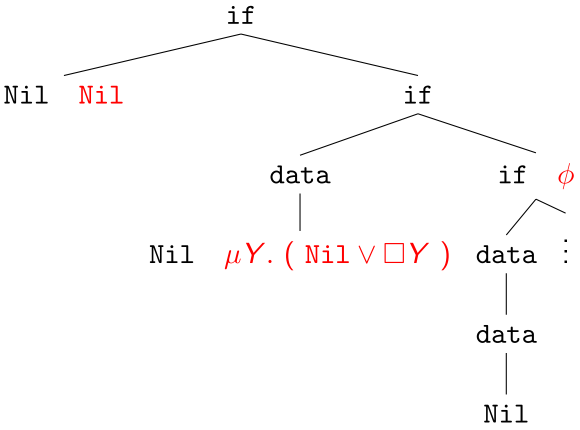
Idea: the formula "starts" on the root



where $\phi = \nu X.\,(\,\texttt{if} \wedge \diamond_1\,(\,\mu Y.\,(\,\texttt{Nil} \vee \Box Y\,)\,) \wedge \diamond_2\,X\,)$

Alternating parity tree automata

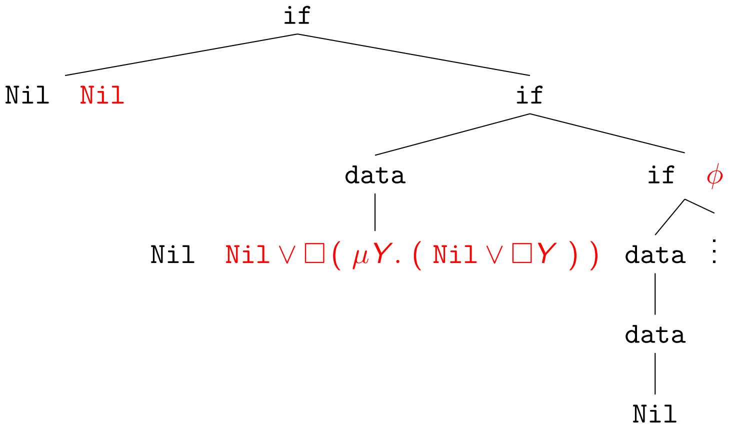
Idea: the formula "starts" on the root



where $\phi \;=\; \nu X.\,(\,\mathtt{if} \wedge \diamond_1\,(\,\mu Y.\,(\,\mathtt{Nil} \vee \Box Y\,)\,) \wedge \diamond_2\,X\,)$

Alternating parity tree automata

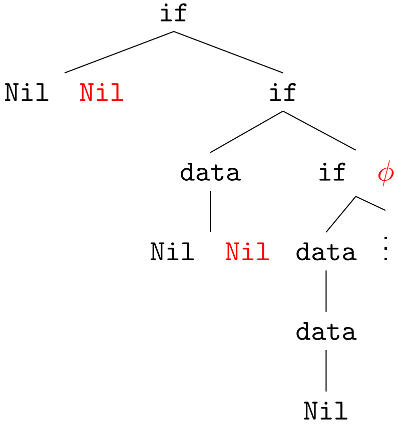
Idea: the formula "starts" on the root



where $\phi \;=\; \nu X.\, (\, \texttt{if} \wedge \Diamond_1 \;(\, \mu Y.\, (\, \texttt{Nil} \vee \Box Y \,)\,) \wedge \Diamond_2\, X \,)$

Alternating parity tree automata

Idea: the formula "starts" on the root



where $\phi \ = \ \nu X. \ (\ \texttt{if} \wedge \diamond_1 \ (\ \mu Y. \ (\ \texttt{Nil} \vee \Box Y \ ) \ ) \wedge \diamond_2 X \ )$

Alternating parity tree automata

Conversion to an automaton ?

- Unfold the formula over the tree, but always by reading a letter: synchronization with the tree.
- States $\Leftrightarrow$ subformulas
- Needs non-determinism for $\vee$ and alternation for $\wedge$
- Needs a parity condition for distinguishing μ and ν

Alternating parity tree automata

For a MSO formula φ,

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

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

$$\text{APT} \quad = \quad \underbrace{\underbrace{\text{alternating tree automata (ATA)}}_{\text{weak MSO}} + \text{parity condition.}}_{\text{MSO}}$$

Alternating tree automata

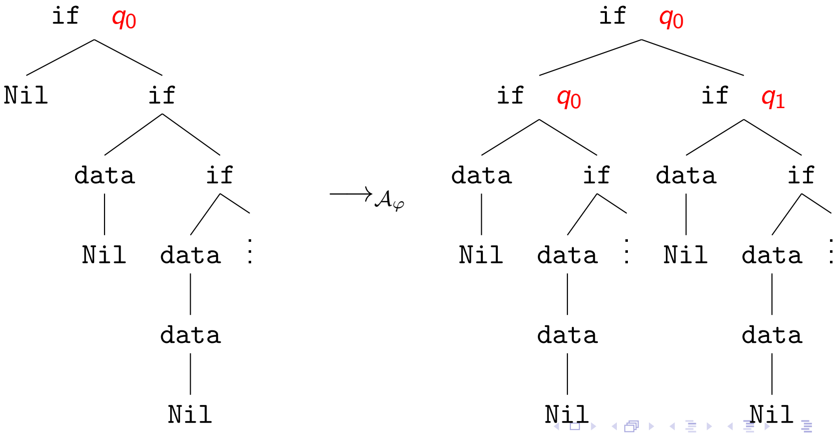
ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

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 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

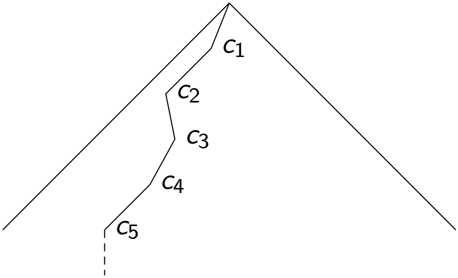
"after a `read` operation, a `write` eventually occurs".

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.



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:

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

Intersection types and alternation

ATA vs. HORS

$$\frac{}{(\,\lambda x.\, s\,)\ t \quad \rightarrow_{\mathcal{G}^w} \quad s[x \leftarrow t]} \qquad \frac{s \quad \rightarrow_{\mathcal{G}^w} \quad s'}{s\ t \quad \rightarrow_{\mathcal{G}^w} \quad s'\ t}$$

$$\frac{}{F \quad \rightarrow_{\mathcal{G}^w} \quad \mathcal{R}(F)}$$

$$\frac{t \quad \rightarrow^*_{\mathcal{G}^w} \quad a\ t_1\ \cdots\ t_n \qquad t_i\ :\ q_{ij} \quad \rightarrow^\infty_{\mathcal{G},\mathcal{A}} \quad t'_i\ :\ q_{ij}}{t\ :\ q \quad \rightarrow^\infty_{\mathcal{G},\mathcal{A}} \quad (a\ (t'_{11}\ :\ (1, q_{11}))\ \cdots\ (t'_{nk_n}\ :\ (n, q_{nk_n})))\ :\ q}$$

where the duplication "conforms to δ" (there is non-determinism).

Starting from $S\ :\ q_0$, this computes run-trees of an ATA $\mathcal{A}$ over $\langle \mathcal{G} \rangle$.

We get closer to type theory. . .

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

$$\texttt{if} \;:\; o \rightarrow o \rightarrow o$$

(this talk is NOT about filter models!)

Alternating tree automata and intersection types

In a derivation typing $\texttt{if} \quad T_1 \quad T_2$:

$$\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \texttt{if} : \emptyset \rightarrow (q_0 \wedge q_1) \rightarrow q_0}\delta \qquad \color{red}{\emptyset}}{\emptyset \vdash \texttt{if}\ T_1 : (q_0 \wedge q_1) \rightarrow q_0}\text{App} \qquad \dfrac{\vdots}{\color{red}{\Gamma_{21} \vdash T_2 : q_0}} \qquad \dfrac{\vdots}{\color{red}{\Gamma_{22} \vdash T_2 : q_1}}}{\Gamma_{21}, \Gamma_{22} \vdash \texttt{if}\ T_1\ T_2 : q_0}\text{App}$$

Intersection types naturally lift to higher-order – and thus to $\mathcal{G}$, which finitely represents $\langle \mathcal{G} \rangle$.

Theorem (Kobayashi)

$S : q_0 \vdash S : q_0$ *iff* *the ATA* $\mathcal{A}_\varphi$ *has a run-tree over* $\langle \mathcal{G} \rangle$.

A type-system for verification: without parity conditions

$$\text{Axiom} \qquad \frac{}{x : \bigwedge_{\{i\}} \theta_i :: \kappa \ \vdash \ x : \theta_i :: \kappa}$$

$$\delta \qquad \frac{\{ (i, q_{ij}) \mid 1 \leq i \leq n, 1 \leq j \leq k_i \} \quad \text{satisfies} \quad \delta_A(q, a)}{\emptyset \vdash a : \bigwedge_{j=1}^{k_1} q_{1j} \rightarrow \ldots \rightarrow \bigwedge_{j=1}^{k_n} q_{nj} \rightarrow q :: o \rightarrow \cdots \rightarrow o}$$

$$\text{App} \qquad \frac{\Delta \vdash t : (\theta_1 \wedge \cdots \wedge \theta_k) \rightarrow \theta :: \kappa \rightarrow \kappa' \qquad \Delta_i \vdash u : \theta_i :: \kappa}{\Delta + \Delta_1 + \ldots + \Delta_k \ \vdash \ t\, u : \theta :: \kappa'}$$

$$\lambda \qquad \frac{\Delta, x : \bigwedge_{i \in I} \theta_i :: \kappa \ \vdash \ t : \theta :: \kappa'}{\Delta \ \vdash \ \lambda x . t : \left(\bigwedge_{i \in I} \theta_i\right) \rightarrow \theta :: \kappa \rightarrow \kappa'}$$

$$\mathit{fix} \qquad \frac{\Gamma \vdash \mathcal{R}(F) : \theta :: \kappa}{\overline{F : \theta :: \kappa \ \vdash \ F : \theta :: \kappa}}$$

An alternate proof

Theorem

$S : q_0 \vdash S : q_0$ iff the ATA $\mathcal{A}_\phi$ has a run-tree over $\langle \mathcal{G} \rangle$.

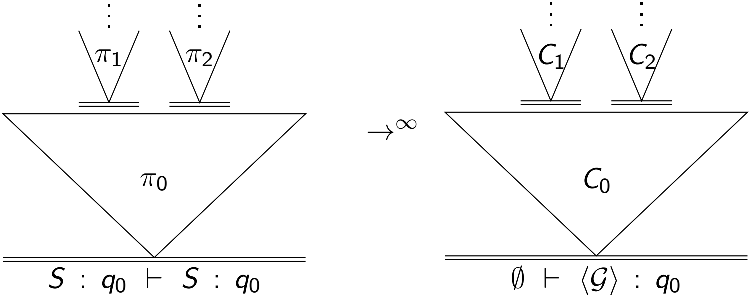
Proof: coinductive subject reduction/expansion along the head reduction of derivations with non-idempotent intersection types.

$$\dfrac{\begin{array}{c}\pi \\ \vdots\end{array}}{S : q_0 \vdash S : q_0} \quad \longleftrightarrow \quad \dfrac{\begin{array}{c}\pi' \\ \vdots\end{array}}{\emptyset \vdash \langle \mathcal{G} \rangle : q_0} \quad \Longleftrightarrow$$

$\langle \mathcal{G} \rangle$ is accepted by $\mathcal{A}$.

Adding parity conditions to the type system

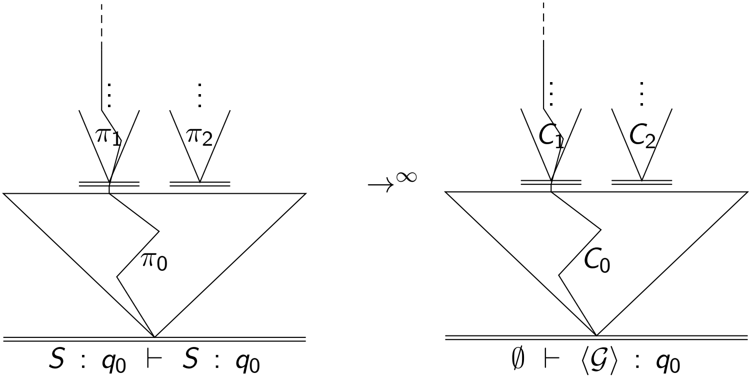
One more word on proof rewriting



where the C_i are the tree contexts obtained by normalizing each π_i.

$C_0[C_1[], C_2[]]$ is a prefix of a run-tree of $\mathcal{A}$ over $\langle \mathcal{G} \rangle$.

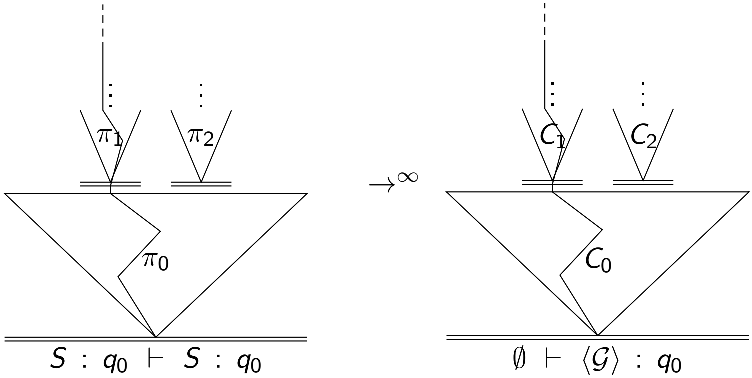
One more word on proof rewriting



Theorem

In this quantitative setting, there is a correspondence between infinite branches of the typing of $\mathcal{G}$ and of the run-tree over $\langle \mathcal{G} \rangle$ obtained by normalization.

One more word on proof rewriting



The goal now: add information in π_i about the maximal color seen in C_i.

One extra color: ϵ for the case $C_i = []$.

Alternating parity tree automata

We add coloring informations to intersection types:

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

now corresponds to

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

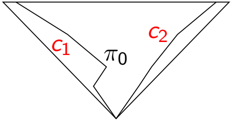
Application computes the “local” maximum of colors, and the fixpoint deals with the acceptance condition.

A type-system for verification (Grellois-Melliès 2014)

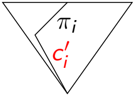
$$\text{App} \qquad \frac{\Delta \vdash t : (\Box_{c_1}\ \theta_1\ \wedge \cdots \wedge \Box_{c_k}\ \theta_k) \to \theta\ ::\ \kappa \to \kappa' \qquad \Delta_i \vdash u\ :\ \theta_i\ ::\ \kappa}{\Delta\ +\ \Box_{c_1}\Delta_1\ +\ \ldots\ +\ \Box_{c_k}\Delta_k\ \vdash\ t\,u\ :\ \theta :: \kappa'}$$

Subject reduction: the contraction of a redex

$$x\ :\ \Box_\epsilon\,\theta_1\ \vdash\ x\ :\ \theta_1 \qquad x\ :\ \Box_\epsilon\,\theta_2\ \vdash\ x\ :\ \theta_2$$



$$y\ :\ \Box_\epsilon\,\sigma_i\ \vdash\ y\ :\ \sigma_i$$



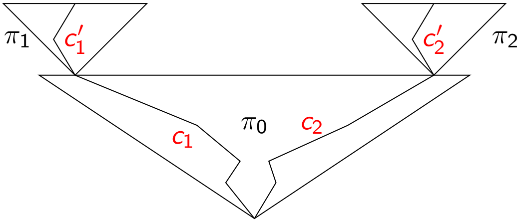
$$\frac{\dfrac{\Delta, x\ :\ \Box_{c_1}\ \theta_1\ \wedge \cdots \wedge \Box_{c_k}\ \theta_k\ \vdash\ t\ :\ \theta}{\Delta\ \vdash\ \lambda x.\,t\ :\ (\Box_{c_1}\ \theta_1\ \wedge \cdots \wedge \Box_{c_k}\ \theta_k) \to \theta} \qquad \Delta_i\ \vdash\ u\ :\ \theta_i}{\Delta\ +\ \Box_{c_1}\Delta_1\ +\ \ldots\ +\ \Box_{c_k}\Delta_k \vdash\ (\,\lambda x.\,t\,)\ \ u\ :\ \theta}$$

A type-system for verification (Grellois-Melliès 2014)

$$\text{App} \quad \frac{\Delta \vdash t : (\Box_{c_1}\ \theta_1\ \wedge \cdots \wedge \Box_{c_k}\ \theta_k) \to \theta\ ::\ \kappa \to \kappa' \qquad \Delta_i \vdash u\ :\ \theta_i\ ::\ \kappa}{\Delta + \Box_{c_1}\Delta_1 + \ldots + \Box_{c_k}\Delta_k\ \vdash\ t\,u\ :\ \theta :: \kappa'}$$

gives a proof of the same sequent:

$y\ :\ \Box_\epsilon\,\sigma_1\ \vdash\ y\ :\ \sigma_1 \qquad\qquad y\ :\ \Box_\epsilon\,\sigma_2\ \vdash\ y\ :\ \sigma_2$



$$\Delta + \Box_{c_1}\Delta_1 + \ldots + \Box_{c_k}\Delta_k \vdash\ t[x \leftarrow u]\ :\ \theta$$

A type-system for verification (Grellois-Melliès 2014)

Axiom $$\frac{}{x\ :\ \bigwedge_{\{i\}} \square_{\epsilon}\ \theta_i\ ::\ \kappa\ \ \vdash\ \ x\ :\ \theta_i\ ::\ \kappa}$$

δ $$\frac{\{\,(i, q_{ij}) \mid 1 \leq i \leq n, 1 \leq j \leq k_i\,\}\quad \text{satisfies}\quad \delta_A(q, a)}{\emptyset \vdash a\ :\ \bigwedge_{j=1}^{k_1} \square_{\Omega(q_{1j})}\ q_{1j}\ \rightarrow\ \ldots\ \rightarrow\ \bigwedge_{j=1}^{k_n} \square_{\Omega(q_{nj})}\ q_{nj} \rightarrow q\ ::\ o \rightarrow \cdots \rightarrow o \rightarrow o}$$

App $$\frac{\Delta \vdash t : (\square_{m_1}\ \theta_1\ \wedge \cdots \wedge \square_{m_k}\ \theta_k) \rightarrow \theta\ ::\ \kappa \rightarrow \kappa' \qquad \Delta_i \vdash u\ :\ \theta_i\ ::\ \kappa}{\Delta\ +\ \square_{m_1}\Delta_1\ +\ \ldots\ +\ \square_{m_k}\Delta_k\ \ \vdash\ \ t\,u\ :\ \theta :: \kappa'}$$

fix $$\frac{\Gamma \vdash \mathcal{R}(F)\ :\ \theta\ ::\ \kappa}{\frac{}{F\ :\ \square_{\epsilon}\ \theta\ ::\ \kappa\ \vdash\ \ F\ :\ \theta\ ::\ \kappa}}$$

λ $$\frac{\Delta\,,\, x\ :\ \bigwedge_{i \in I} \square_{m_i}\ \theta_i\ ::\ \kappa\ \ \vdash\ \ t\ :\ \theta\ ::\ \kappa'}{\Delta\ \ \vdash\ \ \lambda x\,.\,t\ :\ \left(\bigwedge_{i \in I} \square_{m_i}\ \theta_i\right) \rightarrow \theta :: \kappa \rightarrow \kappa'}$$

A type-system for verification (Grellois-Melliès 2014)

We rephrase the parity condition to typing trees, and now capture all MSO:

Theorem (G.-Melliès 2014)

$S : q_0 \vdash S : q_0$ admits a winning typing derivation iff the alternating parity automaton $\mathcal{A}$ has a winning run-tree over $\langle \mathcal{G} \rangle$.

We obtain decidability by collapsing to idempotent types.

Non-idempotency is very helpful for proofs, but leads to infinitary constructions.

Models of linear logic

It was linear logic all the way!

Linear logic very naturally handles alternation via

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

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

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

so that

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

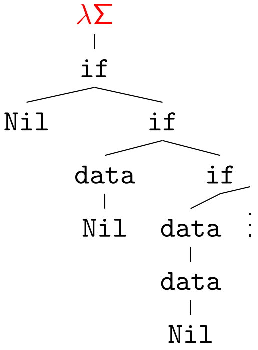
and

$$([\,], [q_0, q_1], q_0) \;\in\; [\![\texttt{if}]\!] \;\subseteq\; \mathcal{M}_{fin}(Q) \times \mathcal{M}_{fin}(Q) \times Q$$

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

Duality between trees and automata

Church encoding of trees:



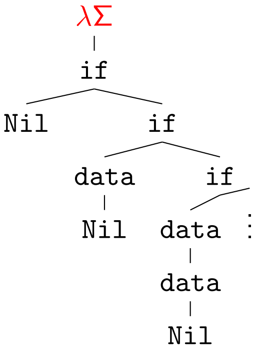
$: \quad o(\Sigma) \to o$

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

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

Duality between trees and automata

Church encoding of trees:



$: \quad o(\Sigma) \to o$

Now, a term of type $o(\Sigma) \to o$ normalizes to a Σ-labelled tree.

Model-checking I

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

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

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

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

Relational composition:

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

Interactive interpretation?

Model-checking I

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

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

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:

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

Interactive interpretation?

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

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

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

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

the set of states q from which

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

accepts the tree generated by the normalization of t is

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

Static analysis, directly on the term.

Model-checking II

Generalizing to trees generated by HORS? $\rightarrow$ add a fixpoint.

Finite iteration $\rightarrow$ inductive fixpoint operator on *Rel*.

Theorem

The infinitary normal form of a λY-term

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

is accepted by

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

from the set of states

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

Model-checking II

Generalizing to trees generated by HORS? $\rightarrow$ add a fixpoint.

Finite iteration $\rightarrow$ inductive fixpoint operator on *Rel*.

Theorem

The infinitary normal form of a λY-term

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

is accepted by

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

from the set of states

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

after a finite execution of the automaton.

On finiteness

Infinite trees need infinite multisets: tree constructors may be used countably.

Defining a new exponential

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

gives a relational model of linear logic with a

coinductive fixpoint operator

(infinite fixpoint unfolding).

New interpretation of terms: $[\![\, t \,]\!]_{gfp}$.

Model-checking III

Theorem

The infinitary normal form of a λY*-term*

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

is accepted by

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

from the set of states

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

The coloring comonad

The coloring modality of the type system corresponds to a comonad in the semantics:

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

Structural morphisms:

$$\begin{array}{rcl} (\max(m_1, m_2),\ a) \ \multimap\ (m_1,\ (m_2,\ a)) & : & \Box A \multimap \Box\,\Box\, A \\ (\epsilon,\ a) \ \multimap\ a & : & \Box A \multimap A \end{array}$$

Parity conditions

The modality $\Box$ distributes over the exponential $\lightning$: there is a natural transformation

$$\lightning\,\Box \;\to \Box\,\lightning$$

satisfying some coherence diagram.

It follows that the composite

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

is an exponential, so that we automatically obtain a model of the λ-calculus associated to the coloured typings.

Colored interpretation: $[\![\, t \,]\!]_{col}$.

Linear decomposition of the intuitionnistic arrow

Kleisli composition: consider

$$f \;:\; \lightning \,\Box\, A \to B$$

and

$$g \;:\; \lightning \,\Box\, B \to C$$

Their composite is defined as

$$\lightning \,\Box\, B \xrightarrow{g} C$$

where λ is the distributivity law between ! and $\Box$.

Linear decomposition of the intuitionnistic arrow

Kleisli composition: consider

$$f \ : \ \lightning \Box A \rightarrow B$$

and

$$g \ : \ \lightning \Box B \rightarrow C$$

Their composite is defined as

$$\lightning \Box \lightning \Box A \xrightarrow{\lightning \Box f} \lightning \Box B \xrightarrow{g} C$$

where λ is the distributivity law between ! and $\Box$.

Linear decomposition of the intuitionnistic arrow

Kleisli composition: consider

$$f \ : \ \lightning \, \square \, A \rightarrow B$$

and

$$g \ : \ \lightning \, \square \, B \rightarrow C$$

Their composite is defined as

$$\lightning \, \lightning \, \square \square \, A \ \xrightarrow{\lambda} \ \lightning \, \square \, \lightning \, \square \, A \ \xrightarrow{\lightning \, \square \, f} \ \lightning \, \square \, B \xrightarrow{g} \ C$$

where λ is the distributivity law between ! and $\square$.

Linear decomposition of the intuitionnistic arrow

Kleisli composition: consider

$$f \ : \ \lightning \square A \to B$$

and

$$g \ : \ \lightning \square B \to C$$

Their composite is defined as

$$\lightning \square\square A \ \to \ \lightning \lightning \square\square A \ \xrightarrow{\lambda} \ \lightning \square \lightning \square A \ \xrightarrow{\lightning \square f} \ \lightning \square B \xrightarrow{g} \ C$$

where λ is the distributivity law between ! and $\square$.

Linear decomposition of the intuitionnistic arrow

Kleisli composition: consider

$$f \; : \; \lightning \, \Box \, A \to B$$

and

$$g \; : \; \lightning \, \Box \, B \to C$$

Their composite is defined as

$$\lightning \, \Box \, A \;\to\; \lightning \, \Box\Box \, A \;\to\; \lightning \, \lightning \, \Box\Box \, A \;\xrightarrow{\lambda}\; \lightning \, \Box \, \lightning \, \Box \, A \;\xrightarrow{\lightning \, \Box \, f}\; \lightning \, \Box \, B \;\xrightarrow{g}\; C$$

where λ is the distributivity law between ! and $\Box$.

Parity conditions

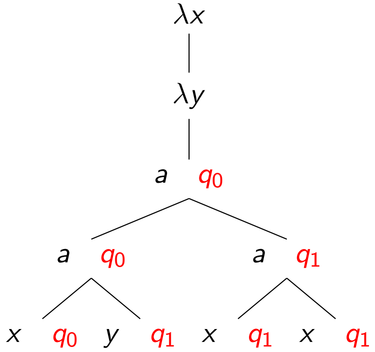
We obtain a very natural colored interpretation of types:

$$[\![A \Rightarrow B]\!] \; = \mathcal{M}_{count}(Col \times [\![A]\!]) \times [\![B]\!]$$

and we can relate the typing derivations in the colored intersection type system with the construction of denotations in the resulting model.

An example of coloured interpretation

Suppose $\Omega(q_0) = 0$ and $\Omega(q_1) = 1$.



This rule will be interpreted in the model as

$$([(0, q_0), (1, q_1), (1, q_1)],\ [(1, q_1)],\ q_0)$$

Connection with the coloured relational model

To obtain the acceptance theorem for alternating parity automata, we need a fixpoint which reflects the parity condition.

This operator composes denotations infinitely, and only keeps the result if it comes from a winning composition tree.

Model-checking IV

Theorem

The infinitary normal form of a λY-term

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

is accepted by the parity automaton

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

from the set of states

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

Model-checking V

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

Two essential differences:

- $[\![!\, A]\!] \;=\; \mathcal{P}_{fin}(A)$
- necessity of "subtyping"

We adapted to *ScottL* the theoretical approach of this work.

Corollary

The higher-order model-checking problem is decidable.

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 idempotent type systems / finitary semantics, we obtain decidability of higher-order model-checking.

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 idempotent type systems / finitary semantics, we obtain decidability of higher-order model-checking.

Thank you for your attention!