First steps towards probabilistic higher-order model-checking

Charles Grellois   Ugo dal Lago

FOCUS Team – INRIA & University of Bologna

Institute for Mathematical Sciences, Singapore
   September 22, 2016
1. A quick reminder of higher-order model-checking (HOMC) and an introduction to intersection types for HOMC

2. Automata for probabilistic properties, and quantitative $\mu$-calculus

3. Towards probabilistic HOMC: first steps and main challenges

4. Connections with semantics: what tensorial logic with effect brings us
Higher-order model-checking
Model-checking

\[ \mathcal{T} = \]

\[ \text{if} \]

\[ \text{Nil} \]

\[ \text{if} \]

\[ \text{data} \]

\[ \text{if} \]

\[ \text{Nil} \]

\[ \text{data} \]

\[ \text{data} \]

\[ \text{Nil} \]

\( \phi \) a logical property on trees, e.g. “all executions are finite”.

Model-checking: does \( \mathcal{T} \models \phi \)?
Finite representations of infinite trees

is not regular: it is not the unfolding of a finite graph as
Finite representations of infinite trees

![](diagram.png)

but it is represented by a higher-order recursion scheme (HORS).

\[ G = \begin{cases} 
    S &= L \text{ Nil} \\
    L \ x &= \text{if } x (L \ (\text{data } x)) 
\end{cases} \]
Higher-order recursion schemes

\[ G = \begin{cases} 
S & = L \text{ Nil} \\
L \times & = \text{if } x (L \text{ (data } x )) 
\end{cases} \]

Rewriting starts from the start symbol \( S \):

\[ S \rightarrow_G L \text{ Nil} \]
Higher-order recursion schemes

\[
G = \begin{cases} 
S & = & L \text{ Nil} \\
L \ x & = & \text{if } x (L \ (\text{data } x)) 
\end{cases}
\]

\[
\begin{array}{c}
\text{L} \\
\text{Nil}
\end{array} \rightarrow_G \\
\begin{array}{c}
\text{if} \\
\text{Nil} \\
\text{L}
\end{array} \\
\begin{array}{c}
\text{data} \\
\text{Nil}
\end{array}
\]
Higher-order recursion schemes

\[ G = \begin{cases} 
  S & = L \ NIL \\
  L \ x & = \text{if } \ x \ (L \ \text{(data } x \ ))
\end{cases} \]
Higher-order recursion schemes

\[ G = \begin{cases} 
S & = \text{L Nil} \\
L \times & = \text{if } x (L (\text{data } x )) 
\end{cases} \]

\[ \langle G \rangle = \]

Tree representation:
- **if**
  - **Nil**
  - **if**
    - **data**
    - **if**
      - **Nil**
      - **data**
        - **data**
        - **Nil**
Higher-order recursion schemes

\[ \mathcal{G} = \begin{cases} 
S & = & L \text{ Nil} \\
L \times & = & \text{if } x (L (\text{data } x)) 
\end{cases} \]

HORS can alternatively be seen as simply-typed \( \lambda \)-terms with simply-typed recursion operators \( Y_\sigma : (\sigma \rightarrow \sigma) \rightarrow \sigma \).
Modal $\mu$-calculus

Equivalent to MSO over trees.

$$\phi, \psi ::= X | a | \phi \lor \psi | \phi \land \psi | \Box \phi | \Diamond_i \phi | \mu X. \phi | \nu X. \phi$$

$\Diamond_i \phi$: $\phi$ holds on a successor in direction $i$

$\Diamond \phi$: $\phi$ holds on a successor

$\Box \phi$: $\phi$ holds on all successors
Modal $\mu$-calculus

Equivalent to MSO over trees.

$\phi, \psi ::= X | a | \phi \lor \psi | \phi \land \psi | \square \phi | \diamond_i \phi | \mu X. \phi | \nu X. \phi$

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

$$
\mu X. \phi(X) \quad \rightarrow \quad \phi(\mu X. \phi(X)) \quad \rightarrow \quad \phi(\phi(\mu X. \phi(X)))
$$

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

$$
\nu X. \phi(X) \quad \rightarrow \quad \phi(\nu X. \phi(X)) \quad \rightarrow \quad \phi(\phi(\nu X. \phi(X)))
$$
Modal μ-calculus

Equivalent to MSO over trees.

\[ \phi, \psi ::= X | a | \phi \lor \psi | \phi \land \psi | \square \phi | \Diamond_i \phi | \mu X . \phi | \nu X . \phi \]

Example formula:

\[ \nu X . ( \text{if} \land \Diamond_1 ( \mu Y . ( \text{Nil} \lor \square Y )) \land \Diamond_2 X ) \]

Companion automata model: APT = ATA + parity condition.
Alternating tree automata (ATA)

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

Typically: $\delta(q_0, \text{if}) = (2, q_0) \land (2, q_1)$. 
Alternating tree automata (ATA)

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

Typically: \( \delta(q_0, \text{if}) = (2, q_0) \land (2, q_1) \).

\[
\begin{array}{c}
\text{if} \quad q_0 \\
\text{Nil} \quad \text{if} \\
\text{data} \quad \text{if} \\
\text{Nil} \quad \text{data} : \\
\text{data} \\
\text{Nil}
\end{array}
\quad \rightarrow \quad
\begin{array}{c}
\text{if} \quad q_0 \\
\text{if} \quad q_1 \\
\text{data} \quad \text{if} \\
\text{Nil} \quad \text{data} : \\
\text{data} \\
\text{Nil}
\end{array}
\]
Alternating parity tree automata

Express reachability with ATA: does every branch ends by Nil?

Problem: ATA execute coinductively.

Solution: parity condition.
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 occurring 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 occurring infinitely often along it is even.

A run-tree is winning iff all its infinite branches are.

For a MSO formula \( \varphi \):

\[ A_\varphi \text{ has a winning run-tree over } \langle G \rangle \text{ iff } \langle G \rangle \models \varphi. \]
Alternating parity tree automata

\[
Q = \{q\}
\]

\[
\Omega(q) = 1
\]

\[
\delta(if, q) = (1, q) \land (2, q)
\]

\[
\delta(data, q) = (1, q)
\]

\[
\delta(Nil, q) = \top
\]
HOMC and intersection types
Alternating tree automata and intersection types

A key remark (Kobayashi 2009, Kobayashi-Ong 2009):

\[ \delta(q_0, \text{if}) = (2, q_0) \land (2, q_1) \]

can be seen as the intersection typing

\[ \text{if} : \emptyset \rightarrow (q_0 \land q_1) \rightarrow q_0 \]

refining the simple typing

\[ \text{if} : o \rightarrow o \rightarrow o \]
Alternating tree automata and intersection types

A run-tree over if $T_1 \ T_2$ is a derivation of $\emptyset \vdash \text{if } T_1 \ T_2$:

$$
\begin{align*}
\delta & \qquad \emptyset \vdash \text{if } : \emptyset \rightarrow (q_0 \land q_1) \rightarrow q_0 \\
\text{App} & \qquad \emptyset \vdash \text{if } T_1 : (q_0 \land q_1) \rightarrow q_0 \\
\text{App} & \qquad \emptyset \vdash \text{if } T_1 \ T_2 : q_0
\end{align*}
$$

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

Grellois & dal Lago (INRIA & U. Bologna) Towards probabilistic HOMC Sep 22, 2016 14 / 40
A type-system for verification: without parity conditions

Axiom
\[ x : \bigwedge_{i \in I} \theta_i :: \kappa \vdash x : \theta_i :: \kappa \]

\[ \{ (i, q_{ij}) \mid 1 \leq i \leq n, 1 \leq j \leq k_i \} \text{ satisfies } \delta_A(q, a) \]

\[ \emptyset \vdash a : \bigwedge_{j=1}^{k_1} q_{1j} \to \cdots \to \bigwedge_{j=1}^{k_n} q_{nj} \to q :: o \to \cdots \to o \]

App
\[ \Delta \vdash t : (\theta_1 \land \cdots \land \theta_k) \to \theta :: \kappa \to \kappa' \]
\[ \Delta \vdash u : \theta_i :: \kappa \]
\[ \Delta + \Delta_1 + \cdots + \Delta_k \vdash tu : \theta :: \kappa' \]

\[ \Delta, x : \bigwedge_{i \in I} \theta_i :: \kappa \vdash t : \theta :: \kappa' \]
\[ \Delta \vdash \lambda x. t : (\bigwedge_{i \in I} \theta_i) \to \theta :: \kappa \to \kappa' \]

\[ \Gamma \vdash \mathcal{R}(F) : \theta :: \kappa \]
\[ \frac{F : \theta :: \kappa \vdash F : \theta :: \kappa}{fix} \]
Soundness and completeness: without parity conditions

**Theorem (Kobayashi)**

\[ S : q_0 \vdash S : q_0 \text{ iff } \text{the ATA } A_{\varphi} \text{ has a run-tree over } \langle G \rangle. \]

Additional connection with models of linear logic: intersection acts as the exponential modality.

Bridge: indexed linear logic (Bucciarelli-Ehrhard). We can also use an indexed version of tensorial logic.
Colored intersection types
An example of colored intersection type

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

$$\lambda x \quad \lambda y \quad a \quad q_1$$

has now type

$$\Box_0 q_0 \land \Box_1 q_1 \rightarrow \Box_1 q_1 \rightarrow q_1$$

Note the color 0 on $q_0$...
A type-system for verification

A colored Application rule:

\[
\begin{align*}
\Delta & \vdash t : (\Box_{m_1} \theta_1 \land \cdots \land \Box_{m_k} \theta_k) \to \theta \\
\Delta & + \Box_{m_1} \Delta_1 + \ldots + \Box_{m_k} \Delta_k \vdash u : \theta_i \\
\Delta_i & \vdash u : \theta_i
\end{align*}
\]
A type-system for verification

A colored Application rule:

\[\Delta \vdash t : (\square_{m_1} \theta_1 \land \cdots \land \square_{m_k} \theta_k) \rightarrow \theta \quad \Delta_i \vdash u : \theta_i\]

\[\Delta + \square_{m_1} \Delta_1 + \cdots + \square_{m_k} \Delta_k \vdash t \ u : \theta\]

inducing a winning condition on infinite proofs: the node

\[\Delta_i \vdash u : \theta_i\]

has color \(m_i\), others have color \(\epsilon\), and we use the parity condition.
A type-system for verification (Grellois-Melliès 2014)

\[
\begin{align*}
\text{Axiom} & \quad x : \Box_\epsilon \theta_i \vdash x : \theta_i \\
\text{δ} & \quad \{ (i, q_{ij}) \mid 1 \leq i \leq n, 1 \leq j \leq k_i \} \text{ satisfies } \delta_A(q, a) \\
& \quad \emptyset \vdash a : \bigwedge_{j=1}^{k_1} \Box \Omega(q_{1j}) q_{1j} \rightarrow \cdots \rightarrow \bigwedge_{j=1}^{k_n} \Box \Omega(q_{nj}) q_{nj} \rightarrow q \\
\text{App} & \quad \Delta \vdash t : (\Box_{m_1} \theta_1 \land \cdots \land \Box_{m_k} \theta_k) \rightarrow \theta \quad \Delta_i \vdash u : \theta_i \\
& \quad \Delta + \Box_{m_1} \Delta_1 + \cdots + \Box_{m_k} \Delta_k \vdash t u : \theta \\
\text{λ} & \quad \Delta, x : \bigwedge_{i \in I} \Box_{m_i} \theta_i \vdash t : \theta \\
& \quad \Delta \vdash \lambda x . t : (\bigwedge_{i \in I} \Box_{m_i} \theta_i) \rightarrow \theta \\
\text{fix} & \quad \Gamma \vdash \mathcal{R}(F) : \theta \\
& \quad F : \Box_\epsilon \theta \vdash F : \theta
\end{align*}
\]
A type system for verification

Theorem (G.-Melliès 2014, from Kobayashi-Ong 2009)

\[ S : q_0 \vdash S : q_0 \quad \text{admits a \textit{winning} typing derivation} \]

\[ \iff \]

\textit{the alternating parity automaton} \( \mathcal{A} \) has a \textit{winning} run-tree over \( \langle G \rangle \).

Static analysis: directly on the \textit{finite} HORS \( G \).
**Theorem (G.-Melliès 2014, from Kobayashi-Ong 2009)**

S : q₀ ⊢ S : q₀ admits a **winning** typing derivation

iff

the alternating **parity** automaton \( A \) has a **winning** run-tree over \( \langle G \rangle \).

Again, connection with **models of linear logic**.
We proved that coloring is a modality in the sense of S4.

Connection can be made through **indexed colored logics**.
Probabilistic HOMC
```cpp
IntList random_list() {
    IntList list = Nil;
    while(rand() > 0.1) {
        list := rand_int()::list;
    }
    return l;
}
```
Probabilistic HOMC

Allows to represent probabilistic programs.

And to define higher-order regular Markov Decision Processes: those bisimilar to their encoding represented by a HORS.

(encoding of probabilities + payoffs in symbols)
Probabilistic automata

Idea: no longer verify $\phi$ but $Pr_{\geq p} \phi$.

- Step one: quantitative ATA.
- Step two: deal with colors and parity condition.

Probabilistic automata (PATA):

- ATA on non-probabilistic symbols
- $\oplus_p$ probabilistic behavior on choice symbol

Run-tree: labels $(q, p_n, p_f)$.

The root of a run-tree of probability $p$ is labeled $(q_0, 1, p)$, where $p$ is the probability with which we want the tree to satisfy the formula.
Probabilistic alternating tree automata

Probabilistic behavior:

\[ \oplus_p (q, p_n, p_f) \]

is labeled as

\[ \oplus_p (q, p_n, p_f) \]

\[ b (q, p \times p_n, p'_f) \]

\[ c (q, (1 - p) \times p_n, p_f - p'_f) \]

for some \( p'_f \in [0, p_f] \) such that \( p'_f \leq p \times p_n \) and \( p_f - p'_f \leq (1 - p) \times p_n \).
Example of PATA run

\[ \phi = \text{“all the branches of the tree contain data”} \]

is modeled by the PATA:

- \( \delta_1(q_0, \text{data}) = (1, q_1) \),
- \( \delta_1(q_1, \text{data}) = (1, q_1) \),
- \( \delta_1(q_0, \text{Nil}) = \bot \),
- \( \delta_1(q_1, \text{Nil}) = \top \).
Example of PATA run

\[ \bigoplus \frac{1}{10} \left( q_0, 1, \frac{9}{10} \right) \]

\[ \bigoplus \frac{1}{10} \left( q_0, \frac{9}{10}, \frac{9}{10} \right) \]

\[ \bigoplus \frac{1}{10} \left( q_0, \frac{81}{100}, \frac{81}{100} \right) \]

\[ \bigoplus \frac{1}{10} \left( q_0, \frac{81}{100}, \frac{81}{100} \right) \]

\[ \bigoplus \frac{1}{10} \left( q_0, \frac{81}{100}, \frac{81}{100} \right) \]

\[ \bigoplus \frac{1}{10} \left( q_0, \frac{81}{100}, \frac{81}{100} \right) \]
Another example

\( \phi = \) all the branches of the tree contain an even amount of data.

Associated automaton:

- \( \delta_2(q_0, \text{data}) = (1, q_1) \),
- \( \delta_2(q_1, \text{data}) = (1, q_0) \),
- \( \delta_2(q_0, \text{Nil}) = \top \),
- \( \delta_2(q_1, \text{Nil}) = \bot \).
Another example

\[ \text{Nil} \left( q_0, \frac{1}{10}, \frac{1}{10} \right) \oplus \frac{1}{10} \left( q_0, \frac{10}{19}, \frac{10}{19} \right) \]

\[ \text{data} \left( q_0, \frac{9}{100}, 0 \right) \oplus \frac{1}{10} \left( q_0, \frac{9}{10}, \frac{81}{190} \right) \]

\[ \text{data} \left( q_0, \frac{81}{1000}, \frac{81}{1000} \right) \oplus \frac{1}{10} \left( q_0, \frac{81}{1000}, \frac{81}{1900} \right) \]

\[ \text{data} \left( q_1, \frac{81}{1000}, \frac{81}{1000} \right) \]

\[ \text{data} \left( q_0, \frac{729}{1000}, \frac{6561}{19000} \right) \]

\[ \text{Nil} \left( q_0, \frac{81}{1000}, \frac{81}{1000} \right) \]
Intersection types for PATA

As for ATA, except for tree constructors:

\[ \{ (i, q_{ij}) \mid 1 \leq i \leq n, 1 \leq j \leq k_i \} \text{ satisfies } \delta_A(q, a) \]

\[ \emptyset \vdash a : \wedge_{j=1}^{k_1} (q_{1j}, p_n, p_f) \rightarrow \ldots \rightarrow \wedge_{j=1}^{k_n} (q_{nj}, p_n, p_f) \rightarrow (q, p_n, p_f) \]

\[ p'_f \in ]0, p_f[ \text{ and } p'_f \leq p \times p_n \text{ and } p_f - p'_f \leq (1 - p) \times p_n \]

\[ \emptyset \vdash \oplus_p : (q, p \times p_n, p'_f) \rightarrow (q, (1 - p) \times p_n, p_f - p'_f) \rightarrow (q, p_n, p_f) \]

\[ q \in Q \text{ and } p \times p_n \geq p_f \]

\[ \emptyset \vdash \oplus_p : (q, p \times p_n, p_f) \rightarrow \emptyset \rightarrow (q, p_n, p_f) \]

\[ q \in Q \text{ and } (1 - p) \times p_n \geq p_f \]

\[ \emptyset \vdash \oplus_p : \emptyset \rightarrow (q, (1 - p) \times p_n, p_f) \rightarrow (q, p_n, p_f) \]
Intersection types for PATA

Theorem

\[ \emptyset \vdash S : (q_0, 1, p) \]

iff

the PATA \( A \) has a run-tree of probability \( p \) over the tree \( \langle G \rangle \) generated by \( G \).

Under connection Rel/non-idempotent types, we obtain a similar denotational theorem.

Note that \( \llbracket o \rrbracket = Q \times [0, 1] \times [0, 1] \).
The probabilistic $\mu$-calculi zoo

- $q\mu = \text{quantitative interpretation of } \mu$-calculus \cite{HK97,MM97}
  - $\cup = \max$, $\cap = \min$, no PCTL, game characterization on finite models
- $GPL = \text{extension with finite nesting of } [.]_{>p}$ quantifications \cite{CPN99}
  - expresses PCTL* but neither $\exists a$ nor $L_\mu$ over Kripke structures
  - no game characterization, alternation-free fragment
- $pL\mu^\oplus$ is $L\mu + \text{Lukasiewicz-operators} + \text{more}$ \cite{MS13}
  - probabilistic quantification = fixed point and multiplication
  - (tree) game characterization over all models, encodes PCTL
- $\mu^p$ and $\muPCTL$ \cite{CKP15}
  - distinguishes between qualitative and quantitative formulas
  - model checking $\mu^p$-calculus is as hard as solving parity games
  - poly-time model checking of $\muPCTL$ for bounded alternation depth
- $P\muTL = L\mu + [\cdot]_{>p}$ for next-modalities \cite{LSWZ15}
  - satisfiability by emptiness in prob. alt. parity automata (in 2EXPTIME)
PATA and quantitative $\mu$-calculus

What we seem to capture: $[[\phi]]_{\emptyset}(\epsilon) \geq p$ for safety formulas, with:

- $[[a]]_{\rho}(s) = 1$ iff $\text{label}(s) = a$, 0 else
- $[[X]]_{\rho}(s) = \rho(X)(s)$
- $[[\phi \land \psi]]_{\rho}(s) = \min([[\phi]]_{\rho}(s), [[\psi]]_{\rho}(s))$
- $[[\phi \lor \psi]]_{\rho}(s) = \max([[\phi]]_{\rho}(s), [[\psi]]_{\rho}(s))$
- $[[\Box \phi]]_{\rho}(s) = \min \{[[\phi]]_{\rho}(s') | s' \text{ successor of } s\}$
- $[[\Diamond \phi]]_{\rho}(s) = \max \{[[\phi]]_{\rho}(s') | s' \text{ successor of } s\}$
- $[[\nu X. \phi]]_{\rho}(s) = \text{gfp}(f \mapsto [[\phi]]_{\rho[f/X]})(s)$

We did not consider the quantitative operator $\odot \phi$ but could add it, with

$$[[\odot \phi]]_{\rho}(s) = \sum_{s' \text{ succ } s} Pr(s, s')[[\phi]]_{\rho}(s')$$
Why only safety?

Safety conditions → all infinite branches are accepted.

Problem with automata: can not detect a priori sets of loosing branches.

That’s why there is an a posteriori parity condition.

To capture it: a colored run-tree of probability

\[ p - p_{bad} \]

is

- a run-tree of probability \( p \),
- where \( p_{bad} \) is the measure of the set of rejecting (= odd-colored) branches in the run-tree.

But how to reflect that size in the typing?
Tensorial logic with effects and PATA
Automata are counter-programs with effects

Grellois-Melliès, CSL 2015:

With a linear logic point of view: HOMC is a dual process between

a program: the recursion scheme $\mathcal{G}$,

and

a counter-program with (co)effects: the APT $\mathcal{A}$. 
Tensorial logic

- A refinement of linear logic
- A logic of tensor, sum and negation where $A \not\cong \neg\neg A$
- Purpose: conciliate linear logic with algebraic effects
- Deeply related to game semantics: it is the syntax of dialogue games...
- ...and more generally related to dialogue categories

Tensorial logic with effects (Melliès) connects with semantics (dialogue categories with effects)
States in tensorial logic

\[ \text{Lookup} \quad \frac{\Gamma \vdash \bot}{\Gamma} \]

\[ \text{Update}_{val} \quad \frac{\Gamma \vdash \bot}{\Gamma} \]

and equations such as

\[ \pi \]

\[ \vdots \]

\[ \Gamma \vdash \bot \]

\[ = \]

\[ \text{Update}_{val_1} \quad \frac{\Gamma \vdash \bot}{\Gamma} \]

\[ \text{Lookup} \quad \frac{\Gamma \vdash \bot}{\Gamma} \]

\[ \vdots \]

\[ \text{Update}_{val_n} \quad \frac{\Gamma \vdash \bot}{\Gamma} \]
Tensorial logic and PATA

Update\(_{q_0,p \times p_b,p'_f}\) \[\frac{\Gamma \vdash t_1 : \bot}{\Gamma \vdash t_1 : \bot} \quad \frac{\Gamma \vdash t_2 : \bot}{\Gamma \vdash t_2 : \bot}\]  
Update\(_{q_0,(1-p) \times p_b,p_f-p'_f}\)  
Choice\(_{p'_f}\) \[\frac{\Gamma \vdash \bigoplus_p t_1 \ t_2 : \bot}{\Gamma \vdash \bigoplus_p t_1 \ t_2 : \bot} \quad \vdots \]  
Lookup\(_{p_b,p_f}\) \[\frac{\Gamma \vdash \bigoplus_p t_1 \ t_2 : \bot}{\Gamma \vdash \bigoplus_p t_1 \ t_2 : \bot} \]  

where \(\bigoplus_p : \bot \longrightarrow \bot \longrightarrow \bot \subset \Gamma\)

**Fundamental idea:** the state of the automaton is a state in the sense of the state monad. Non-determinism is handled by a monadic effect as well.
Tensorial logic and PATA

\[ \delta(a, q_0) = (1, q_0) \land (1, q_1) \quad \delta(a, q_1) = \bot \]

 Exceptions when \( \delta \) is not defined.

Automata are counter-programs with effects
What’s next

- Have a look at other $\mu$-calculi, there seems to be some connection with obligation games

- Investigate decidability for safety (reachability is already undecidable)

- Can we obtain approximations in undecidable cases?

- Connection with denotational models and semantics: Rel, dialogue categories with effects. . .
  Automata theory opens interesting questions in semantics!

Thank you for your attention!
What’s next

- Have a look at other $\mu$-calculi, there seems to be some connection with obligation games
- Investigate decidability for safety (reachability is already undecidable)
- Can we obtain approximations in undecidable cases?
- Connection with denotational models and semantics: Rel, dialogue categories with effects. . .
  Automata theory opens interesting questions in semantics!

Thank you for your attention!