First steps towards probabilistic higher-order model-checking

Charles Grellois    Ugo dal Lago

FOCUS Team – INRIA & University of Bologna

GDRI-LL Meeting on Intersection Types
June 14, 2016
Roadmap

1. A reminder of higher-order model-checking (HOMC) and intersection types for HOMC

2. Non-idempotent intersection types and HOMC for almost-sure MSO properties

3. Automata for probabilistic properties (weaker than MSO), intersection types, tensorial logic with effects
Higher-order model-checking
Model-checking

\[ \mathcal{T} = \]

\[
\begin{array}{c}
\text{if} \\
\text{Nil} \\
\text{if} \\
\text{data} \\
\text{if} \\
\text{Nil} \\
\text{data} \\
\text{data} \\
\text{Nil}
\end{array}
\]

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

Model-checking: does \( \mathcal{T} \models \phi \)?
Higher-order model-checking

Infinite trees with a finite representation: a $\lambda Y$-term or a higher-order recursion scheme (HORS).

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

\[ \rightarrow \text{model-checking on } \lambda\text{-terms or HORS.} \]
Alternating tree automata

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

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

```
if q_0
  data
    if
      data
        if
          data
            Nil

if q_0
  data
    if
      data
        if
          data
            Nil

A_\varphi
```
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 \( \varphi \):

\( \mathcal{A}_\varphi \) has a winning run-tree over \( \langle G \rangle \) iff \( \langle G \rangle \models \varphi \).
Alternating parity tree automata

\[
\begin{align*}
Q &= \{q\} \\
\Omega(q) &= 1 \\
\delta(\text{if}, q) &= (1, q) \land (2, q) \\
\delta(\text{data}, q) &= (1, q) \\
\delta(\text{Nil}, q) &= \top
\end{align*}
\]
HOMC and intersection types
Alternating tree automata and intersection types

A key remark (Kobayashi 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

\[ \text{if} : o \rightarrow o \rightarrow o \]

(this talk is NOT about filter models!)
Alternating tree automata and intersection types

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

\[
\delta \quad \emptyset \vdash \text{if : } \emptyset \rightarrow (q_0 \land q_1) \rightarrow q_0 \quad \emptyset \quad \vdots \quad \emptyset \vdash T_2 : q_0 \quad \vdots \\
\text{App} \quad \emptyset \vdash \text{if } T_1 : (q_0 \land q_1) \rightarrow q_0 \\
\text{App} \quad \emptyset \vdash \text{if } T_1 T_2 : q_0
\]

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 \( \langle \mathcal{G} \rangle \).

Here: variant with non-idempotent types.

Under connection Rel/non-idempotent types, we obtain a similar denotational theorem.
A type-system for verification: without parity conditions

Axiom

\[
\begin{align*}
\text{Axiom} & \quad x : \bigwedge_{\{i\}} \theta_i :: \kappa \mid x : \theta_i :: \kappa \\
\end{align*}
\]

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

\[
\begin{align*}
\emptyset \mid a : \bigwedge_{j=1}^{k_1} q_{1j} \rightarrow \cdots \rightarrow \bigwedge_{j=1}^{k_n} q_{nj} \rightarrow q :: o \rightarrow \cdots \rightarrow o \\
\end{align*}
\]

App

\[
\begin{align*}
\Delta \mid t : (\theta_1 \land \cdots \land \theta_k) \rightarrow \theta :: \kappa \rightarrow \kappa' & \quad \Delta_i \mid u : \theta_i :: \kappa \\
+ \Delta_1 + \cdots + \Delta_k \mid t u : \theta :: \kappa' \\
\end{align*}
\]

\[
\begin{align*}
\Delta, x : \bigwedge_{i \in I} \theta_i :: \kappa \mid t : \theta :: \kappa' & \quad \Delta \mid \lambda x . t : (\bigwedge_{i \in I} \theta_i) \rightarrow \theta :: \kappa \rightarrow \kappa' \\
\end{align*}
\]

\[
\begin{align*}
\text{fix} & \quad \Gamma \mid \mathcal{R}(F) : \theta :: \kappa \\
\end{align*}
\]

\[
\begin{align*}
\Gamma \mid F : \theta :: \kappa \mid F : \theta :: \kappa \\
\end{align*}
\]
Idea of the proof

**Theorem**

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

\[ \pi \]
\[ \vdash \]
\[ S : q_0 \vdash S : q_0 \]

\[ \iff \]

\[ \pi' \]
\[ \vdash \]
\[ \emptyset \vdash \langle G \rangle : q_0 \]

\[ \iff \]

\[ \langle G \rangle \text{ is accepted by } A. \]

- **Soundness**: infinitary (in fact, coinductive) subject reduction
- **Completeness**: build a derivation for \( G \) (similar to subject expansion)
Soundness

\[
\vdash S : q_0 \\
\pi_0 \rightarrow^\infty \\
\vdash \emptyset \vdash \langle G \rangle : q_0
\]

where the \( C_i \) are the tree contexts obtained by normalizing each \( \pi_i \).

\( C_0[C_1[], C_2[]] \) is a prefix of a run-tree of \( A \) over \( \langle G \rangle \).
Colored intersection types
A type-system for verification

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

\[ \Delta \vdash t : (\square c_1 \theta_1 \land \cdots \land \square c_k \theta_k) \rightarrow \theta :: \kappa \rightarrow \kappa' \quad \Delta_i \vdash u : \theta_i :: \kappa \]

\[ \Delta + \square c_1 \Delta_1 + \ldots + \square c_k \Delta_k \vdash t u : \theta :: \kappa' \]

Subject reduction: the contraction of a redex

\[ \Delta, x : \square \epsilon \theta_1 \vdash x : \theta_1 \quad x : \square \epsilon \theta_2 \vdash x : \theta_2 \]

\[ \Delta \vdash \lambda x. t : (\square c_1 \theta_1 \land \cdots \land \square c_k \theta_k) \rightarrow \theta \]

\[ \Delta + \square c_1 \Delta_1 + \ldots + \square c_k \Delta_k \vdash (\lambda x. t) u : \theta \]

Charles Grellois (INRIA & U. Bologna)
A type-system for verification

\[
\begin{align*}
\Delta &\vdash t : (\Box_{c_1} \theta_1 \land \cdots \land \Box_{c_k} \theta_k) \to \theta :: \kappa \to \kappa' \quad \Delta_i \vdash u : \theta_i :: \kappa \\
\Delta + \Box_{c_1} \Delta_1 + \cdots + \Box_{c_k} \Delta_k &\vdash t\ u : \theta :: \kappa'
\end{align*}
\]

gives a proof of the same sequent:

\[
\begin{align*}
y : \Box_\epsilon \sigma_i &\vdash y : \sigma_i \\
\pi_i &\vdash \pi_i \\
c_i &\vdash c_i' \\
\pi_0 &\vdash \pi_0 \\
c_1 &\vdash c_1 \\
c_2 &\vdash c_2
\end{align*}
\]

\[
\Delta + \Box_{c_1} \Delta_1 + \cdots + \Box_{c_k} \Delta_k \vdash t[x \leftarrow u] : \theta
\]
A type system for verification

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

<table>
<thead>
<tr>
<th>Theorem (G.-Melliès 2014, from Kobayashi-Ong 2009)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ S : q_0 \vdash S : q_0 \text{ admits a } \textbf{winning} \text{ typing derivation} ]</td>
</tr>
<tr>
<td>[ \text{iff} ]</td>
</tr>
<tr>
<td>[ \text{the alternating } \textit{parity} \text{ automaton } A \text{ has a } \textbf{winning} \text{ run-tree over } \langle G \rangle. ]</td>
</tr>
</tbody>
</table>

- **Soundness**: infinitary (in fact, coinductive) subject reduction + study of color preservation on infinite branches
- **Completeness**: build an \textbf{optimal} derivation for \( G \)
Soundness

Crucial lemma of Kobayashi-Ong (unpublished and reformulated here): every infinite branch of $\emptyset \vdash \langle G \rangle : q_0$ comes from an infinite branch of $S : q_0 \vdash S : q_0$.

Consequence: derivation of $S : q_0 \vdash S : q_0$ is winning $\implies$ the run-tree computed by subject reduction is as well.
Soundness

Reformulated in this non-idempotent setting, this lemma seems to induce:

**Conjecture**

*There is an injection from the infinite branches of the run-tree to these of the derivation of* $S : q_0 \vdash S : q_0$. 
Completeness and optimality

Completeness: build a derivation for $S : q_0 \vdash S : q_0$ from a run-tree (i.e. a derivation for $\emptyset \vdash \langle G \rangle : q_0$).

Main challenge: consider

$$G = \begin{cases} 
S &= F H \\
F &= \lambda x. a (F x) \\
H &= b H 
\end{cases}$$

producing

$$\langle G \rangle = a a a \cdots$$

via finite reductions

$$S \xrightarrow{\ast}_G a \cdots a F (b \cdots b H)$$

(only head reduction is optimal)
Completeness and optimality

Completeness: build a derivation for $S : q_0 \vdash S : q_0$ from a run-tree (i.e. a derivation for $\emptyset \vdash \langle G \rangle : q_0$).

Main challenge: consider

$$G = \begin{cases} 
S & = F H \\
F & = \lambda x. a \ (F \ x) \\
H & = b \ H 
\end{cases}$$

We need to detect that $H$ is never contributes to $\langle G \rangle$, else we can take:

$$F : \Box_0 (\Box_0 \ q_0 \to q_0)$$

and introduce a loosing branch for $H$.

Completeness proof $\to$ define an optimal derivation (relying on the optimality of head reduction).
Almost-sure MSO properties
Almost-sure MSO properties

In the spirit of qualitative tree languages, consider an APT \( A \) with the following acceptance condition:

a run-tree is almost winning

iff

the set of its branches loosing for the parity condition has measure 0.

This allows to check whether a MSO property is almost-surely satisfied.
Almost-sure MSO properties

Consider the same non-idempotent intersection type system as for MSO.

But change the winning condition accordingly.

Conjecture

\[ S : q_0 \vdash S : q_0 \text{ admits an } \textit{almost winning} \text{ typing derivation } \]

iff

the APT \( A \) has an \textit{almost winning} run-tree over \( \langle G \rangle \).
Almost-sure MSO properties

**Soundness:** reduction process which may drop branches (cf. the infinite branch injection conjecture).

So the size of the set of loosing branches decreases.
Almost-sure MSO properties

Completeness: from the proof of Kobayashi and Ong, in a non-idempotent setting:

Conjecture

There is a 1-to-1 correspondence between the infinite branches of the run-tree and these of the optimal derivation of $S : q_0 \vdash S : q_0$ built by the completeness proof.

In other words: in this particular case of an optimal proof, the injection of the previous conjecture becomes a bijection.

It follows that the existence of an almost winning run-tree over $\langle G \rangle$ gives an almost winning derivation of $S : q_0 \vdash S : q_0$. 
Probabilistic automata
IntList random_list() {
    IntList list = Nil;
    while (rand() > 0.1) {
        list := rand_int()::list;
    }
    return l;
}
Probabilistic automata

Idea: check that $\phi$ holds with probability $\geq p$ i.e. that it holds on a subtree of measure $\geq p$.
We extend an ATA $\mathcal{A}$ with some quantitative behavior.

Probabilistic automata (PATA):
- ATA on non-probabilistic symbols
- $\oplus p$ probabilistic behavior on choice symbol

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

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 automata

Probabilistic behavior:

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

is labeled as

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

\[ b (q, p \times p_b, p'_f) \quad c (q, (1 - p) \times p_b, p_f - p'_f) \]

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

\( \phi = \text{“all the branches of the tree contain data”} \)

is modeled by the PATA:

\[
\begin{align*}
\delta_1(q_0, \text{data}) &= (1, q_1), \\
\delta_1(q_1, \text{data}) &= (1, q_1), \\
\delta_1(q_0, \text{Nil}) &= \bot, \\
\delta_1(q_1, \text{Nil}) &= \top.
\end{align*}
\]
Example of PATA run

$\oplus \frac{1}{10} (q_0, 1, \frac{9}{10})$

$\oplus \frac{1}{10} (q_0, \frac{9}{10}, \frac{9}{10})$

$\oplus \frac{1}{10} (q_0, \frac{81}{100}, \frac{81}{100})$

$\oplus \frac{1}{10} (q_0, \frac{81}{1000}, \frac{81}{1000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{10000}, \frac{81}{10000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{100000}, \frac{81}{100000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{1000000}, \frac{81}{1000000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{10000000}, \frac{81}{10000000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{100000000}, \frac{81}{100000000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{1000000000}, \frac{81}{1000000000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{10000000000}, \frac{81}{10000000000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{100000000000}, \frac{81}{100000000000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{1000000000000}, \frac{81}{1000000000000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{10000000000000}, \frac{81}{10000000000000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{100000000000000}, \frac{81}{100000000000000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{1000000000000000}, \frac{81}{1000000000000000})$

$\oplus \frac{1}{10} (q_0, \frac{81}{10000000000000000}, \frac{81}{10000000000000000})$
Intersection types for PATA

As for ATA, except for tree constructors:

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

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

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

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

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

\[
\emptyset \vdash \bigoplus_p : (q, p \times p_b, p_f) \rightarrow \emptyset \rightarrow (q, p_b, p_f)
\]

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

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

Conjecture

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

iff

the PATA \( \mathcal{A} \) has a run-tree of probability \( p \) over the tree \( \langle t \rangle \) generated by \( t \) (which is a term or an unfolded \( \lambda Y \)-term).

To check: that the former proof works with an infinite amount of types refining \( o \).
Intersection types for PATA

Conjecture

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

iff

the PATA $\mathcal{A}$ has a run-tree of probability $p$ over the tree $\langle t \rangle$
generated by $t$ (which is a term or an unfolded $\lambda Y$-term).

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

Note that $\llbracket o \rrbracket = Q \times [0, 1] \times [0, 1]$. 
Automata are counter-programs with effects

Grellois-Mellès, 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 with effects and PATA
Tensorial logic

- A refinement of linear logic
- A logic of tensor, sum and negation where $A \not\equiv \neg\neg A$
- 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

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

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

and equations such as

\[ \pi \quad \vdash \bot \quad \ldots \quad \vdash \bot \quad \pi \]

\[ = \quad \text{Update}_{val_1} \quad \frac{\vdash \bot}{\vdash \bot} \quad \text{Lookup} \quad \frac{\vdash \bot \quad \ldots \quad \vdash \bot}{\vdash \bot} \]

\[ \pi \quad \vdash \bot \quad \ldots \quad \vdash \bot \quad \pi \]

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

\[
\begin{align*}
\text{Update}_{q_0, p \times p_b, p'_f} & : \\
\Gamma \vdash t_1 : \bot & \quad \Gamma \vdash t_2 : \bot \\
\hline
\Gamma \vdash t_1 : \bot & \quad \Gamma \vdash t_2 : \bot \\
\end{align*}
\]

\[
\begin{align*}
\text{Choice}_{p'_f} \quad \Gamma \vdash \bigoplus_p t_1 \cdot t_2 : \bot & \quad \Gamma \vdash \bigoplus_p t_1 \cdot t_2 : \bot \\
\hline
\Gamma \vdash \bigoplus_p t_1 \cdot t_2 : \bot \\
\end{align*}
\]

\[
\begin{align*}
\text{Lookup}_{p_b, p_f} \quad \Gamma \vdash \bigoplus_p t_1 \cdot t_2 : \bot & \quad \Gamma \vdash \bigoplus_p t_1 \cdot t_2 : \bot \\
\hline
\end{align*}
\]

where \( \bigoplus_p : \bot \to \bot \to \bot \to \bot \in \Gamma \)

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

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

\[
\begin{array}{c}
\text{Update}_{q_0,p_b,p_f} \quad \frac{\Gamma \vdash t : \bot}{\Gamma \vdash t : \bot} \quad \frac{\Gamma \vdash t : \bot}{\Gamma \vdash t : \bot} \\
\text{Promotion} \quad \frac{\Gamma \vdash t : !\bot}{\Gamma \vdash t : \bot} \\
\text{Lookup}_{p_b,p_f} \quad \frac{\Gamma \vdash a \, t : \bot}{\Gamma \vdash a \, t : \bot} \\
\end{array}
\]  

where \( a : !\bot \rightarrow \bot \in \Gamma \)

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

Automata are counter-programs with effects
What’s next

- A non-idempotent variant of Kobayashi-Ong’s result (in a coinductive way?)
- Find a less naive type system
- Connection with denotational models: Rel, dialogue categories with effects