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

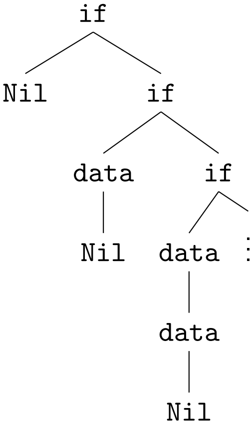
Roadmap

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 μ-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

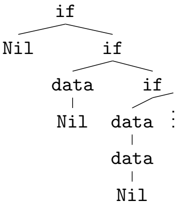
$\mathcal{T}$ $=$



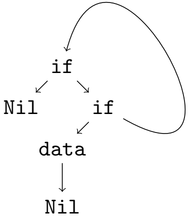
ϕ a logical property on trees, e.g. "all executions are finite".

Model-checking: does $\mathcal{T} \vDash \phi$?

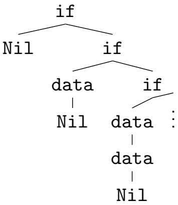
Finite representations of infinite trees



is not regular: it is not the unfolding of a finite graph as



Finite representations of infinite trees



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

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

Higher-order recursion schemes

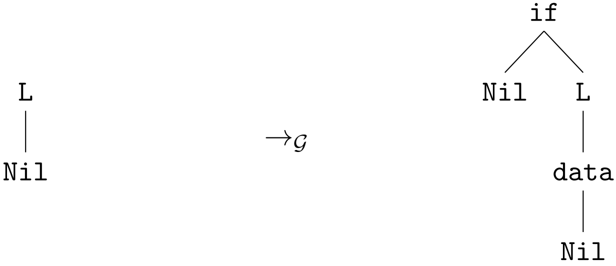
$$\mathcal{G} = \begin{cases} \mathtt{S} & = \mathtt{L\ Nil} \\ \mathtt{L}\ x & = \mathtt{if}\ x\ (\mathtt{L}\ (\mathtt{data}\ x\ )\ ) \end{cases}$$

Rewriting starts from the start symbol S:

$$\mathtt{S} \quad \rightarrow_{\mathcal{G}} \quad \mathtt{L}(\mathtt{Nil})$$

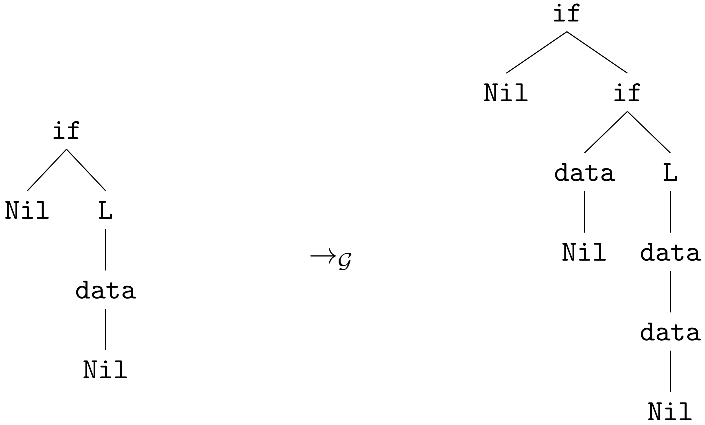
Higher-order recursion schemes

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



Higher-order recursion schemes

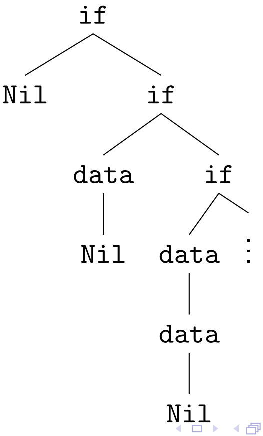
$$\mathcal{G} = \begin{cases} \texttt{S} & = \texttt{L Nil} \\ \texttt{L}\ x & = \texttt{if}\ x\ (\texttt{L}\ (\texttt{data}\ x\ )\ ) \end{cases}$$



Higher-order recursion schemes

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

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



Higher-order recursion schemes

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

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

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

Modal μ-calculus

Equivalent to MSO over trees.

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

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

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

$\Box\,\phi$: ϕ holds on all successors

Modal μ-calculus

Equivalent to MSO over trees.

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

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

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

$\nu X.\,\phi$ 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

Equivalent to MSO over trees.

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

Example formula:

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

Companion automata model: APT = ATA + parity condition.

Alternating tree automata (ATA)

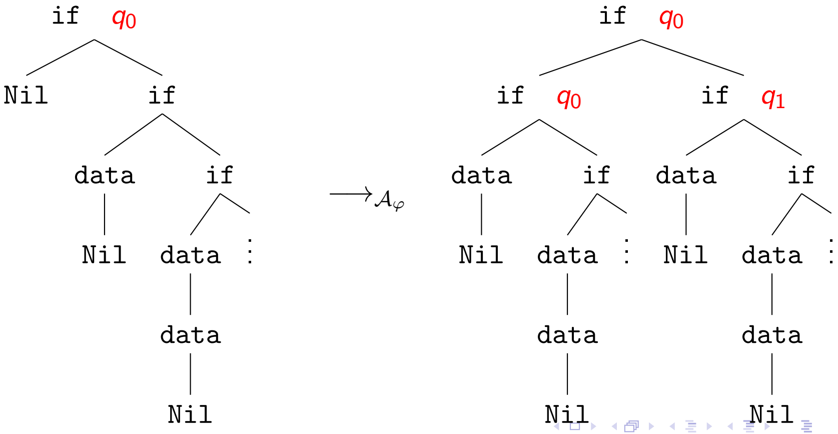
ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

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

Alternating tree automata (ATA)

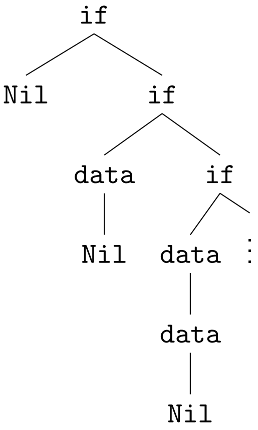
ATA: non-deterministic tree automata whose transitions may duplicate or drop a subtree.

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



Alternating parity tree automata

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



Problem: ATA execute coinductively.

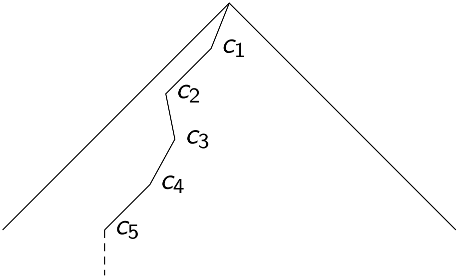
Solution: parity condition.

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 occuring infinitely often along it is even.



Alternating parity tree automata

Each state of an APT is attributed a color

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

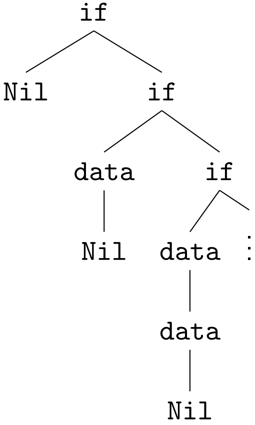
An infinite branch of a run-tree is winning iff the maximal color among the ones occuring infinitely often along it is even.

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

For a MSO formula φ:

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

Alternating parity tree automata



$$Q = \{q\}$$

$$\Omega(q) = 1$$

$$\delta(\texttt{if}, q) = (1, q) \wedge (2, q)$$

$$\delta(\texttt{data}, q) = (1, q)$$

$$\delta(\texttt{Nil}, q) = \top$$

HOMC and intersection types

Alternating tree automata and intersection types

A key remark (Kobayashi 2009, Kobayashi-Ong 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

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

Alternating tree automata and intersection types

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

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

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

A type-system for verification: without parity conditions

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

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

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

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

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

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

Soundness and completeness: without parity conditions

Theorem (Kobayashi)

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

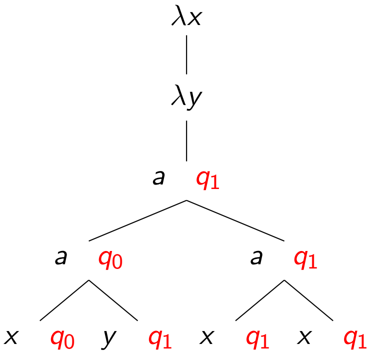
Additional connection with models of linear logic: intersection acts as the exponential modality.

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 $\Omega(q_1) = 1$.



has now type

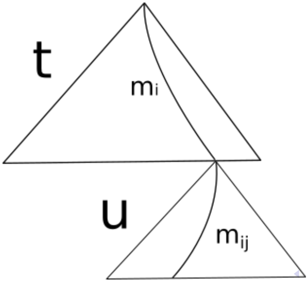
$$\square_0\, q_0 \wedge \square_1\, q_1 \rightarrow \square_1\, q_1 \rightarrow q_1$$

Note the color 0 on q_0...

A type-system for verification

A colored Application rule:

$$\text{App} \quad \frac{\Delta \vdash t : (\square_{m_1}\ \theta_1\ \wedge \cdots \wedge \square_{m_k}\ \theta_k) \to \theta \qquad \Delta_i \vdash u\ :\ \theta_i}{\Delta\ +\ \square_{m_1}\Delta_1\ +\ \ldots\ +\ \square_{m_k}\Delta_k\ \vdash\ t\ u\ :\ \theta}$$



A type-system for verification

A colored Application rule:

$$\text{App} \quad \frac{\Delta \vdash t : (\square_{m_1}\ \theta_1\ \wedge \cdots \wedge \square_{m_k}\ \theta_k) \to \theta \qquad \Delta_i \vdash u\ :\ \theta_i}{\Delta\ +\ \square_{m_1}\Delta_1\ +\ \ldots\ +\ \square_{m_k}\Delta_k\ \ \vdash\ \ t\ u\ :\ \theta}$$

inducing a winning condition on infinite proofs: the node

$$\Delta_i \vdash u\ :\ \theta_i$$

has color m_i, others have color ϵ, and we use the parity condition.

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

Axiom $$\frac{}{x \,:\, \Box_{\epsilon}\ \theta_i \ \vdash\ x \,:\, \theta_i}$$

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

App $$\frac{\Delta \vdash t : (\Box_{m_1}\ \theta_1\ \wedge \cdots \wedge \Box_{m_k}\ \theta_k) \to \theta \qquad \Delta_i \vdash u \,:\, \theta_i}{\Delta\ +\ \Box_{m_1}\Delta_1\ +\ \ldots\ +\ \Box_{m_k}\Delta_k\ \vdash\ t\ u\ :\ \theta}$$

$$\lambda \qquad \frac{\Delta\,,\, x \,:\, \bigwedge_{i \in I} \Box_{m_i}\ \theta_i\ \vdash\ t \,:\, \theta}{\Delta\ \vdash\ \lambda x\,.\,t \,:\, \left(\bigwedge_{i \in I} \Box_{m_i}\ \theta_i\right) \to \theta}$$

$$\mathit{fix} \qquad \frac{\Gamma \vdash \mathcal{R}(F) \,:\, \theta}{F \,:\, \Box_{\epsilon}\ \theta\ \vdash\ F \,:\, \theta}$$

A type system for verification

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

$S : q_0 \vdash S : q_0$ *admits a winning typing derivation*

iff

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

Static analysis: directly on the finite HORS $\mathcal{G}$.

A type system for verification

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

$S : q_0 \vdash S : q_0$ admits a winning typing derivation

iff

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

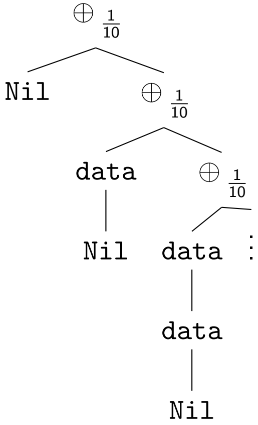
Again, connection with models of linear logic.
We proved that coloring is a modality in the sense of S4.

Connection can be made through indexed colored logics.

Probabilistic HOMC

Probabilistic HOMC

```
IntList random_list() {
  IntList list = Nil;
  while(rand() > 0.1) {
    list := rand_int()::list;
  }
  return l;
}
```

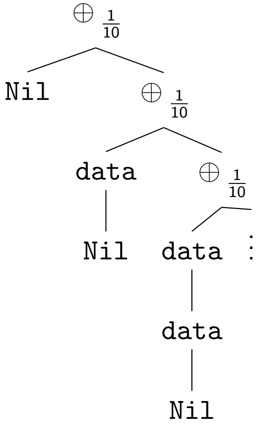


Probabilistic HOMC

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 ϕ 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
- $+$ probabilistic behavior on choice symbol $\oplus_p$

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:

$$\oplus_p \quad (q,\, p_n,\, p_f)$$

$$b \qquad\qquad c$$

is labeled as

$$\oplus_p \quad (q,\, p_n,\, p_f)$$

$$b \quad (q,\, p \times p_n,\, p'_f) \qquad\qquad c \quad (q,\, (1-p) \times p_n,\, p_f - p'_f)$$

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

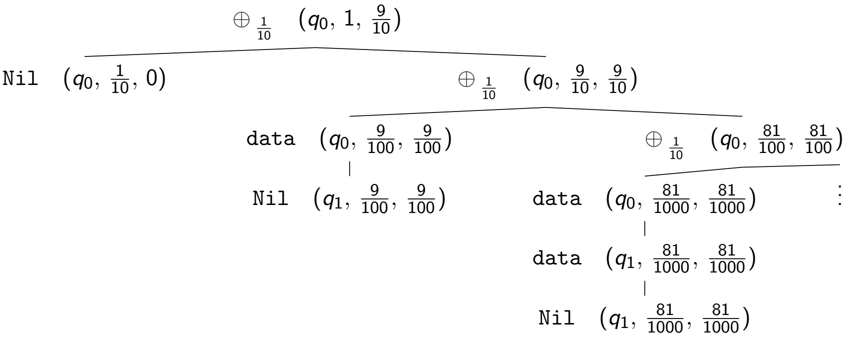
Example of PATA run

$\phi \quad = \quad$ "all the branches of the tree contain $\texttt{data}$"

is modeled by the PATA:

- $\delta_1(q_0, \texttt{data}) \; = \; (1, q_1)$,
- $\delta_1(q_1, \texttt{data}) \; = \; (1, q_1)$,
- $\delta_1(q_0, \texttt{Nil}) \; = \; \perp$,
- $\delta_1(q_1, \texttt{Nil}) \; = \; \top$.

Example of PATA run



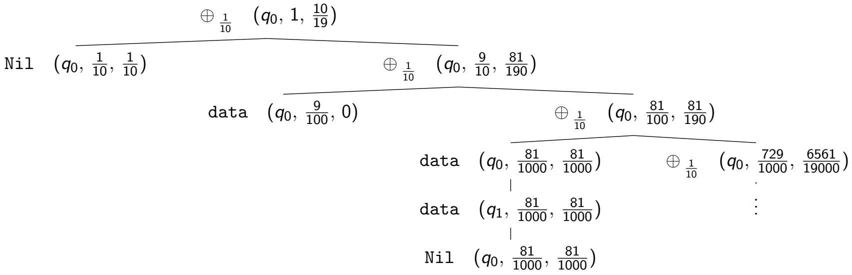
Another example

ϕ = all the branches of the tree contain an even amount of `data`.

Associated automaton:

- $\delta_2(q_0, \texttt{data}) = (1, q_1)$,
- $\delta_2(q_1, \texttt{data}) = (1, q_0)$,
- $\delta_2(q_0, \texttt{Nil}) = \top$,
- $\delta_2(q_1, \texttt{Nil}) = \bot$.

Another example



Intersection types for PATA

As for ATA, except for tree constructors:

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

$$\frac{p'_f \in]0, p_f[\quad \text{and} \quad p'_f \leq p \times p_n \quad \text{and} \quad p_f - p'_f \leq (1-p) \times p_n}{\emptyset \vdash \oplus_p \,:\, (q, p \times p_n, p'_f) \to (q, (1-p) \times p_n, p_f - p'_f) \to (q, p_n, p_f)}$$

$$\frac{q \in Q \quad \text{and} \quad p \times p_n \geq p_f}{\emptyset \vdash \oplus_p \,:\, (q, p \times p_n, p_f) \to \emptyset \to (q, p_n, p_f)}$$

$$\frac{q \in Q \quad \text{and} \quad (1-p) \times p_n \geq p_f}{\emptyset \vdash \oplus_p \,:\, \emptyset \to (q, (1-p) \times p_n,\ p_f) \to (q, p_n, p_f)}$$

Intersection types for PATA

Theorem

$$\emptyset \vdash S : (q_0, 1, p)$$

iff

the PATA $\mathcal{A}$ has a run-tree of probability p over the tree $\langle \mathcal{G} \rangle$ generated by $\mathcal{G}$.

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

Note that $[\![o]\!] \;=\; Q \times [0,1] \times [0,1]$.

PATA and quantitative μ-calculus

The probabilistic μ-calculi zoo

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

PATA and quantitative μ-calculus

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

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

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

$$[\![\circledcirc\phi]\!]_{\rho}(s) \; = \; \sum_{s' \text{ succ } s} Pr(s, s')[\![\phi]\!]_{\rho}(s')$$

Why only safety?

Safety conditions $\rightarrow$ 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

$$\mathit{Lookup} \quad \frac{\Gamma \vdash \bot \qquad \cdots \qquad \Gamma \vdash \bot}{\Gamma \vdash \bot}$$

$$\mathit{Update}_{val} \quad \frac{\Gamma \vdash \bot}{\Gamma \vdash \bot}$$

and equations such as

$$\dfrac{\begin{array}{c}\pi \\ \vdots\end{array}}{\Gamma \vdash \bot} \quad = \quad \begin{array}{r}\mathit{Update}_{val_1} \\ \mathit{Lookup}\end{array} \dfrac{\dfrac{\begin{array}{c}\pi \\ \vdots \\ \Gamma \vdash \bot\end{array}}{\Gamma \vdash \bot} \qquad \cdots \qquad \dfrac{\begin{array}{c}\pi \\ \vdots \\ \Gamma \vdash \bot\end{array}}{\Gamma \vdash \bot}\ \mathit{Update}_{val_n}}{\Gamma \vdash \bot}$$

Tensorial logic and PATA

$$
\dfrac{\dfrac{Update_{q_0, p \times p_b, p'_f} \quad \dfrac{\Gamma \vdash t_1 : \bot}{\Gamma \vdash t_1 : \bot} \qquad \dfrac{\Gamma \vdash t_2 : \bot}{\Gamma \vdash t_2 : \bot} \quad Update_{q_0, (1-p) \times p_b, p_f - p'_f}}{Choice_{p'_f} \quad \dfrac{\Gamma \vdash \oplus_p \ t_1 \ t_2 : \bot}{\Gamma \vdash \oplus_p \ t_1 \ t_2 : \bot}} \qquad \cdots}{Lookup_{p_b, p_f} \quad \Gamma \vdash \oplus_p \ t_1 \ t_2 : \bot}
$$

where $\oplus_p : \bot \multimap \bot \multimap \bot \ \in \ \Gamma$

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

$$\delta(a, q_0) = (1, q_0) \wedge (1, q_1) \qquad \delta(a, q_1) = \bot$$

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash t : \bot}{\Gamma \vdash t : \bot}\ Update_{q_0,p_b,p_f} \qquad \dfrac{\Gamma \vdash t : \bot}{\Gamma \vdash t : \bot}\ Update_{q_1,p_b,p_f}}{\Gamma \vdash t :! \bot}\ Promotion \qquad \dfrac{fail}{\Gamma \vdash a\ t : \bot}}{\Gamma \vdash a\ t : \bot}\ Lookup_{p_b,p_f}$$

where $a :! \bot \multimap \bot \ \in \ \Gamma$

Exceptions when δ is not defined.

Automata are counter-programs with effects

What's next

- Have a look at other μ-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 μ-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!