Probabilistic Testing Semantics in Coq

Jean-François MONIN

Joint work with Yuxin DENG, ECNU, Shanghai

Introduction

Framework

- Probabilistic concurrency theory (Larsen & Skou)
- Probabilistic bisimilarity
- Testing characterisation generalized to reactive probabilistic processes RPP
- General result by van Breugel et al. on continuous state spaces
- Simplification by Deng and Feng on finite state RPP

This work

- Formal proof in Coq of the result by Deng and Feng
- Central part: a non-trivial (initially imperative) program

Introduction

Framework

- Probabilistic concurrency theory (Larsen & Skou)
- Probabilistic bisimilarity
- Testing characterisation generalized to reactive probabilistic processes RPP
- General result by van Breugel et al. on continuous state spaces
- Simplification by Deng and Feng on finite state RPP

This work

- Formal proof in Coq of the result by Deng and Feng
- Central part: a non-trivial (initially imperative) program

Background

Goal and approach

- **Goal:** To reason about probabilistic behaviors in *randomized*, *distributed* and *fault-tolerant* systems.
- **Approach:** extend existing successful models and techniques.
- **Modeling:** process algebra, labeled transition system, Markov chain, ...
- **Verification:** temporal logic, model checking, ..., interactive theorem proving

Probabilistic modelling

- To represent and quantify unreliable behaviour (e.g. fault-tolerant systems)
- To break symmetry in distributed co-ordination problems (e.g. leader election problem, consensus problem)
- Other forms of uncertainty

Preliminaries

Labelled transition systems

Def. A *labelled transition system* (LTS) is a triple $\langle S, Act, \rightarrow\rangle$, where

1. S is a set of states
2. Act is a set of actions
3. $\rightarrow \ \subseteq \ S \times Act \times S$ is the transition relation
   Notation $s \xrightarrow{\alpha} s'$

Bisimulation

$$\begin{array}{ccc} s & \xrightarrow{a} & s' \\ \mathcal{R} & & \mathcal{R} \\ t & \xrightarrow{a} & t' \end{array}$$

s and t are bisimilar if there exists a bisimulation $\mathcal{R}$ with $s \; \mathcal{R} \; t$.

Notation

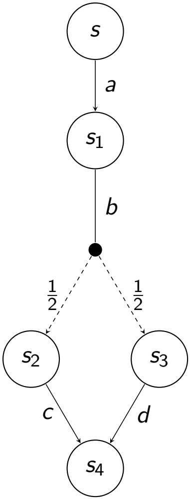
- A discrete probability distribution over a set S is a function $\Delta : S \to [0,1]$ s.t. $\sum_{s \in S} \Delta(s) = 1$
- $\mathcal{D}(S)$: the set of all distributions over S
- $\overline{s}$: the point distribution $\overline{s}(s) = 1$
- Given distributions $\Delta_1, ..., \Delta_n$, we form their linear combination $\sum_{i \in 1..n} p_i \cdot \Delta_i$, where $\forall i : p_i > 0$ and $\sum_{i \in 1..n} p_i = 1$.

Probabilistic labelled transition systems

Def. A *probabilistic labelled transition system* (pLTS) is a triple $\langle S, Act, \rightarrow\rangle$, where

1. S is a set of states
2. Act is a set of actions
3. $\rightarrow \ \subseteq \ S \times Act \times \mathcal{D}(S)$.
   Notation $s \xrightarrow{\alpha} \Delta$

Example



s
a
s_1
b
$\frac{1}{2}$
$\frac{1}{2}$
s_2
s_3
c
d
s_4
t
a
$\frac{1}{2}$
$\frac{1}{2}$
t_1
t_2
b
b
t_3
t_4
c
d
t_5

Probabilistic Bisimulation

$$\begin{array}{ccc} s & \xrightarrow{a} & \Delta \\ \mathcal{R} & & \mathcal{R}^{\dagger} \\ t & \xrightarrow{a} & \Theta \end{array}$$

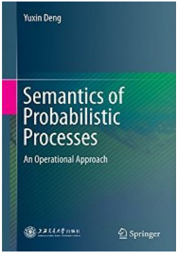
Write $\sim$ for probabilistic bisimilarity.

Lifting relations

Def. Let S, T be two countable sets and $\mathcal{R} \subseteq S \times T$ be a binary relation. The lifted relation $\mathcal{R}^\dagger \subseteq \mathcal{D}(S) \times \mathcal{D}(T)$ is the smallest relation satisfying

1. $s \ \mathcal{R} \ t$ implies $\overline{s}\mathcal{R}^\dagger\overline{t}$
2. $\Delta_i \mathcal{R}^\dagger \Theta_i$ for all $i \in I$ implies $(\sum_{i\in I} p_i \cdot \Delta_i)\mathcal{R}^\dagger(\sum_{i\in I} p_i \cdot \Theta_i)$

There are alternative formulations; related to the Kantorovich metric and the network flow problem. See e.g. http://www.springer.com/978-3-662-45197-7



First modal characterisation

The logic $\mathcal{L}_1$

The language $\mathcal{L}_1$ of formulas:

$$\varphi ::= \quad \top \mid \varphi_1 \wedge \varphi_2 \mid \langle a \rangle_p \varphi.$$

where p is rational number in $[0, 1]$.

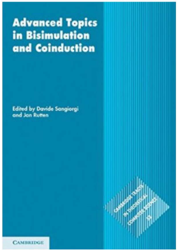
Semantics

- $s \models \top$ always;
- $s \models \varphi_1 \wedge \varphi_2$, if $s \models \varphi_1$ and $s \models \varphi_2$;
- $s \models \langle a \rangle_p \varphi$ iff $s \xrightarrow{a} \Delta$ and $\Delta(\llbracket \varphi \rrbracket) \geq p$, where $\llbracket \varphi \rrbracket = \{s \in S \mid s \models \varphi\}$.

Logical equivalence: $s =_1 t$ if $s \models \varphi \Leftrightarrow t \models \varphi$ for all $\varphi \in \mathcal{L}_1$.

Modal characterisation

Modal characterisation ($s \sim t$ iff $s =_1 t$) for the continuous case given by [Desharnais et al. Inf. Comput. 2003], using the machinery of analytic spaces.



298 *Prakash Panangaden*

There are many variations that one can imagine. Perhaps the simplest is to have negation and dispense with Δ and disjunction. All the variations considered by Larsen and Skou have some negative construct. The striking fact – first discovered in the context of continuous state spaces [DEP98, DEP02] – is that one can get a logical characterisation result with purely positive formulas. The discrete case is covered by these results. Surprisingly no elementary proof for the discrete case – i.e. one that avoids the measure theory machinery – is known.

Second modal characterisation

The logic $\mathcal{L}_2$

The language $\mathcal{L}_2$ of formulas:

$$\varphi ::= \quad \top \mid \varphi_1 \wedge \varphi_2 \mid \langle a \rangle \varphi.$$

Modal characterisation for the continuous case given by [van Breugel et al.]
Again : heavy machinery (probabilistic powerdomains and Banach algebra)

Elementary proof for the discrete case [Deng and Feng].

The logic $\mathcal{L}_2$

The language $\mathcal{L}_2$ of formulas:

$$\varphi ::= \quad \top \mid \varphi_1 \wedge \varphi_2 \mid \langle a \rangle \varphi.$$

Modal characterisation for the continuous case given by [van Breugel et al.]
Again : heavy machinery (probabilistic powerdomains and Banach algebra)

Elementary proof for the discrete case [Deng and Feng].

Semantics

$$
\begin{array}{rcl}
Pr(s, \top) & = & 1 \\
Pr(s, \langle a \rangle \varphi) & = & \begin{cases} \sum_{t \in \lceil \Delta \rceil} \Delta(t) \cdot Pr(t, \varphi) & \text{if } s \xrightarrow{a} \Delta \\ 0 & \text{otherwise.} \end{cases} \\
Pr(s, \varphi_1 \wedge \varphi_2) & = & Pr(s, \varphi_1) \cdot Pr(s, \varphi_2)
\end{array}
$$

Logical equivalence: $s =_2 t$ if $Pr(s, \varphi) = Pr(t, \varphi)$ for all $\varphi \in \mathcal{L}_2$.

Soundness

Thm. If $s \sim t$ then $s =_2 t$.
Proof. Easy by structural induction.

Completeness

Thm. For finite-state reactive pLTSs, if $s =_2 t$ then $s \sim t$.

Key lemma

Lem. For any $I \subseteq \{1, \cdots, n\}$ with $I \neq \emptyset$, there exist a nonempty $I' \subseteq I$ and an enhanced formula t such that

(i) for any $i \neq j \in I'$, $Pr(C_i, t) \neq Pr(C_j, t)$;

(ii) for any $k \in I \backslash I'$, $Pr(C_k, t) = 0$.

30

input : A nonempty $I \subseteq \{1, \cdots, n\}$ with dist. formulae φ_{ij} for all $i \neq j$.
output: A nonempty $I' \subseteq I$ and an enhanced formula φ satisfying (i) and (ii)

begin
 $\mathcal{I}_{pass} \leftarrow \emptyset$; $\mathcal{I}_{rem} \leftarrow \{(i,j) \in I \times I : i < j\}$;
 $I' \leftarrow I$; $\varphi \leftarrow \top$;
 while $\mathcal{I}_{rem} \neq \emptyset$ **do**
  Choose arbitrarily $(i,j) \in \mathcal{I}_{rem}$;
  $I' \leftarrow \{k \in I' : Pr(C_k, \varphi_{ij}) > 0\}$;
  $\mathcal{I}_{dis} \leftarrow \{(k,l) \in \mathcal{I}_{rem} \cap I' \times I' : Pr(C_k, \varphi_{ij}) \neq Pr(C_l, \varphi_{ij})\}$;
  $\mathcal{I}_{rem} \leftarrow (\mathcal{I}_{rem} \cap I' \times I') \backslash \mathcal{I}_{dis}$;
  $\mathcal{I}_{pass} \leftarrow (\mathcal{I}_{pass} \cap I' \times I') \cup \mathcal{I}_{dis}$;
  $\varphi \leftarrow \varphi \wedge \varphi_{ij}$; $\mathcal{I}_{tem} \leftarrow \emptyset$; $\mathcal{I} \leftarrow \mathcal{I}_{pass}$;
  while $\mathcal{I} \neq \emptyset$ **do**
   $\mathcal{I} \leftarrow \{(k,l) \in \mathcal{I}_{pass} \backslash \mathcal{I}_{tem} : Pr(C_k, \varphi) = Pr(C_l, \varphi)\}$;
   if $\mathcal{I} \neq \emptyset$ **then**
    $\varphi \leftarrow \varphi \wedge \varphi_{ij}$;
    $\mathcal{I}_{tem} \leftarrow \mathcal{I}_{tem} \cup \mathcal{I}$;
   end
  end
 end
 return I', φ;
end

Summary

Two logical characterisation of probabilistic bisimilarity for countable and finite-state reactive processes, respectively, with much more elementary proofs than those of Desharnais et al. and van Breugel et al.

Questionable part : correctness of the key algorithm.
Original proof = a few pages of purely technical considerations, without general interest. There are no chances that such a proof could be reused in any way.
Is it reasonable to inflict such proof checking to human reviewers, who have many other important tasks to perform, such as applications for funding?

Fact : many published proofs on computer science are actually wrong.
Proof : by folklore.

Summary

Two logical characterisation of probabilistic bisimilarity for countable and finite-state reactive processes, respectively, with much more elementary proofs than those of Desharnais et al. and van Breugel et al.

Questionable part : correctness of the key algorithm.
Original proof = a few pages of purely technical considerations, without general interest. There are no chances that such a proof could be reused in any way.
Is it reasonable to inflict such proof checking to human reviewers, who have many other important tasks to perform, such as applications for funding?

Fact : many published proofs on computer science are actually wrong.
Proof : by folklore.

Certified formal proof of the key algorithm

Coq in a nutshell

- A highly reliable proof assistant
  - LCF kernel-based architecture
  - proofs as objects
  - based on type theory
- Very expressive logic, including higher-order features, induction, etc.
- Embeds a (functional) programming language,
  which makes it a tool of choice for reasoning about algorithms
- Uniform framework based on Curry-Howard isomorphism
- Human-aided proofs + automation for easy steps :
  tactics and tacticals
- Helpers : notations, abstraction mechanisms such as Haskell-like classes, etc.

Coq for the scientific layperson

- Read the specification
- Check that the script is accepted
- Look for unproved claims, axioms...

Up to the Coq writer: write a clear specification
Common practice : no need to spend time on qualities of the proof: this part is relevant to machines, not humans

- Read the specification
- Check that the script is accepted
- Look for unproved claims, axioms...

Up to the Coq writer: write a clear specification
Common practice : no need to spend time on qualities of the proof: this part is relevant to machines, not humans

Common mistake among early Coq writers:
No need to spend time on qualities of the proof: this part is relevant to machines, not humans

Like programming, proving is not a one-shot activity
Proof : by Curry-Howard

Coq for proof engineers

Common mistake among early Coq writers:
No need to spend time on qualities of the proof: this part is relevant to machines, not humans

Like programming, proving is not a one-shot activity
Proof : by Curry-Howard

Coq in this case study

Imperative algorithm:

- Deep embedding?
- Semantics?
- Hoare logic?
- Translate proof steps of the original paper?

IRRELEVANT!

See statement of the lemma (slide 23)

- use the functional language of Coq
- mimick/adapt/improve the original proof

Coq in this case study

Imperative algorithm:

- Deep embedding?
- Semantics?
- Hoare logic?
- Translate proof steps of the original paper?

IRRELEVANT!

See statement of the lemma (slide 23)

- use the functional language of Coq
- mimick/adapt/improve the original proof

Coq in this case study

Imperative algorithm:

- Deep embedding?
- Semantics?
- Hoare logic?
- Translate proof steps of the original paper?

IRRELEVANT!

See statement of the lemma (slide 23)

- use the functional language of Coq
- mimick/adapt/improve the original proof

Practical issues

Representation of states

Loops, termination

Set notations

Formalisation (main)

```
Variable I0 : set ℕ.

Definition distinguish i j :=
   {t : basic_test | i <> j  ->  ¬ Prb i t == Prb j t}.

Variable oracle : ∀ i j, (i, j) ∈ I0×I0 -> distinguish i j.

Theorem correctness :
  let (fI, ft) := algo_compt_enhanced_test in
  fI <> ∅ ∧
  (∀ k, k ∈ fI -> 0 < Pr k ft) ∧
  (∀ k, k ∈ (I0 \ fI)  -> Pr k ft == 0) ∧
  (∀ i j, i ∈ fI -> j ∈ fI -> i <> j -> ¬ Pr i ft == Pr j ft).

Corollary main_lemma :
  ∃ fI, ∃ ft, fI <> ∅ ∧ etc.
```

Formalisation (main algo)

```
Function out_loop r (di : out_data_invariant0 r)
         measure size_of_out_data r : out_iter_data :=
  match pick di with
  | P_empty _ e => r
  | P_nonempty _ i j s e dis => let (bt, _) := dis in
          out_loop (outer_loop_iter2 r di bt)
                   (outer_loop_iter_invar0 r di bt)
  end.

Definition algo_compt_enhanced_test : final_data :=
  let r := init_data I0 in
  let final_r := out_loop r init_data_invariant0 in
  {| I'f := I' final_r;
     etf := et final_r |}.
```

Formalisation (pick)

```
Inductive resu_pick r : Set :=
| P_empty : Y_rem r = ∅ -> resu_pick r
| P_nonempty : forall i j s,
    Y_rem r = (i, j) :: s -> distinguish i j -> resu_pick r.

Definition pick r (di : out_data_invariant0 r) : resu_pick r.
(* ... using oracle *)
Defined.
```

Thank you!

Former formalisation

I0 created as an initial segment of $\mathbb{N}$ $\{0\ldots\ m\}$

```
Theorem correctness : ∀ m,
  let (fI, ft) := algo_compt_enhanced_test m in
  ((0 < m) -> fI <> ∅) ∧
  (∀ k, k ∈ fI -> 0 < Pr k ft) ∧
  (∀ k, k ∈ (createI m \ fI) -> Pr k ft == 0) ∧
  (∀ i j, i ∈ fI -> j ∈ fI -> i <> j -> Pr i ft <> Pr j ft).
```