Applicative bisimulations for lambda-calculus with continuous probabilities

Raphaëlle Crubillé
joint work with Gilles Barthe, Francesco Gavazzo, Ugo Dal Lago

Inria Grand Est

Scalp 2021

Discrete randomized programs

The program can make probabilistic choices at any point during its execution.

Example (Random Walk)

$$n-1 \xleftarrow{(1-p)} n \xrightarrow{p} n+1$$

Faster algorithms

e.g. Randomised sorting algorithms:

Quick-sort:
worst case time complexity $= O(n^2)$.

Randomised Quick-sort:
Expected worst case time complexity $= O(nlogn)$.

In computational cryptography

Can be a necessity in order to achieve security
(e.g. secure encryption in an asymmetric setting.)

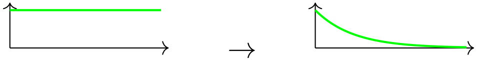
Continuous statistical programming

The program can:

- sample from standard probability distributions (Gaussian distribution, uniform distribution...);
- modify them using a push-forward operation

Example

Build an exponential distribution from a uniform one
let $x =$ sample in $-\log(x)$.



Density functions.

To describe the behaviour of systems:

- with inherent uncertainty
- of whom we have incomplete knowledge

Higher-order Probabilistic Programming

Higher-order languages extended with:

- Discrete randomized algorithms (e.g. randomized sorting);
- continuous probability distributions (e.g. to model physical systems);

$$
\begin{aligned}
M \in \Lambda ::= \; & x \;\big|\; \lambda x^A \cdot M \;\big|\; (MN) \;\big|\; (YN) \\
& \big|\; \text{ifz } (M, N, L) \;\big|\; \text{let } x = M \text{ in } N) \\
& \big|\; M \oplus N \;\big|\; \underline{n} \;\big|\; \text{succ } (M) \;\big|\; \text{pred } (M), \qquad n \in \mathbb{N} \\
& \big|\; \text{sample} \;\big|\; \underline{r} \;\big|\; \underline{f}, \qquad r \in \mathbb{R}, f : \mathbb{R} \to \mathbb{R} \text{ mesureable}
\end{aligned}
$$

Operational Semantics.

Discrete case

$[\![M]\!]$: Values $\to [0,1]$ a discrete distribution

Continuous case

$[\![M]\!] : \Sigma_{\text{Values}} \to [0,1]$ a continuous distribution where:

$$\Sigma_{\text{Values}} \subseteq \text{Parts(Values) a } \sigma\text{-algebra;}$$

Example

- $[\![\mathtt{t} \oplus \mathtt{f}]\!] = \mathscr{D}$ with $\mathscr{D}(\mathtt{t}) = \frac{1}{2}$, $\mathscr{D}(\mathtt{f}) = \frac{1}{2}$;
- $[\![\text{let } x = \text{ sample } \text{ in } (\lambda y.x + y)]\!] = \mathscr{D}$ with

$$\mathscr{D} : A \in \Sigma_{\text{Values}} \mapsto \mu_{\text{Borel}}(\{z \in [0,1] \mid (\lambda y.z + y) \in A\})$$

e.g. $\mathscr{D}(\{\lambda y.r + y \mid r \in [\frac{1}{3}, \frac{2}{3}]\}) = \frac{1}{3}$.

Morris Context Equivalence (1969)

Comparing Two Programs

check whether two programs behaves *the same* no matter how the environment interacts with them (compiler optimisation, verification of a specification...);

When are two programs context equivalent?

- Environments are modelled as **contexts**–i.e. terms with a hole–thus by way of the underlying language
- Two terms are context equivalent if their **observable behaviour** is the same in **any** context.

Definition (Context Equivalence)

$M \ \equiv^{\mathrm{ctx}} N$ when $\forall$ context $\mathcal{C}$ that returns a ground type,

$$[\![\mathcal{C}[M]]\!] = [\![\mathcal{C}[N]]\!] \qquad \text{as distributions.}$$

Contextual Reasoning– Challenges

- $M \not\equiv^{\text{ctx}} N$?
  Find **one** context $\mathcal{C}$ able to distinguish them.
- $M \equiv^{\text{ctx}} N$?
  all contexts should be considered.

Objective:

A characterisation of context equivalence that gets rid of the *universal quantification* on contexts.

Definition

A binary relation R on programs is:

- *sound* when $M\,R\,N \Rightarrow M \equiv^{\text{ctx}} N$
- *complete* when $M \equiv^{\text{ctx}} N \Leftrightarrow M\,R\,N$.

Applicative Bisimilarity [Abramsky93](1)

General idea:

- Expressing interactively the semantics as a Labelled Transition System.
- Obtain an equivalence on programs from the bisimulation on this system.

Programs	**Values**
M	
N	
L	
$\vdots$	

Applicative Bisimilarity [Abramsky93](1)

General idea:

- Expressing interactively the semantics as a Labelled Transition System.
- Obtain an equivalence on programs from the bisimulation on this system.

Programs	**Values**
M	V
N	W
L	U
$\vdots$	$\vdots$

Applicative Bisimilarity [Abramsky93](1)

General idea:

- Expressing interactively the semantics as a Labelled Transition System.
- Obtain an equivalence on programs from the bisimulation on this system.

Programs **Values**

M

Applicative Bisimilarity [Abramsky93](1)

General idea:

- Expressing interactively the semantics as a Labelled Transition System.
- Obtain an equivalence on programs from the bisimulation on this system.

Programs **Values**

$$M \xrightarrow{eval} V$$

Applicative Bisimilarity [Abramsky93](1)

General idea:

- Expressing interactively the semantics as a Labelled Transition System.
- Obtain an equivalence on programs from the bisimulation on this system.

Programs **Values**

$$M \xrightarrow{eval} V$$

$$\lambda x.N$$

Applicative Bisimilarity [Abramsky93](1)

General idea:

- Expressing interactively the semantics as a Labelled Transition System.
- Obtain an equivalence on programs from the bisimulation on this system.

Programs **Values**

$$M \xrightarrow{eval} V$$

$$N\{L/x\} \xleftarrow{L} \lambda x.N$$

Applicative Bisimilarity [Abramsky93] (2)

$$
\begin{tikzcd}
 & & & \lambda x.L \arrow[dr, bend left, "T"] & & & \\
M \arrow[urrr, bend left, "eval"] & R & N \arrow[dr, bend right, "eval"'] & & L\{T/x\} & R & K\{T/x\} \\
 & & & \lambda x.K \arrow[urrr, bend right, "T"'] & & &
\end{tikzcd}
$$

- **Similarity**: union of all simulations, denoted $\precsim$;
- **Bisimilarity**: union of all bisimulations, denoted $\sim$.

Full Abstraction results (deterministic case)

Untyped pure λ-calculus, where we observe **convergence**

	$\sim \subseteq \equiv^{ctx}$	$\sim = \equiv^{ctx}$	$\precsim \subseteq \leq^{ctx}$	$\precsim = \leq^{ctx}$
CBN	✓	✓	✓	✓
CBV	✓	✓	✓	✓

[Abramsky1990,Howe1993]

Bisimilarity for discrete probabilistic systems (LMC)

Definition (Labelled Markov Chain)

a triple $\mathscr{M} = (\mathcal{S}, \mathcal{A}, \{h_a \mid a \in \mathcal{A}\})$, where

- $\mathcal{S}$ is a countable set of *states*, $\mathcal{A}$ a countable set of *labels*;
- h_a is a *transition probability matrix*, i.e., a function $h_a : \mathcal{S} \times \mathcal{S} \to [0,1]$ such that $\forall s, a,\ h_a(s,\cdot)$ is a (sub)-distribution;

Definition ([Larsen-Skou'91])

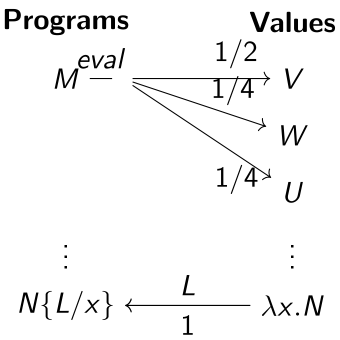
A symmetric relation R on $\mathcal{S}$ is a bisimulation when:

$$s\,R\,t \quad \Rightarrow \quad \forall a \in \mathcal{L}, \forall X \subseteq \mathcal{S}\ R\text{-closed}, h_a(s,X) = h_a(t,X).$$

Theorem (Logical caracterisation[Breugel et al'05])

$$\phi ::= \top \;\Big|\; < a >_p \cdot\phi \;\Big|\; \phi \wedge \phi$$

A Labelled Markov Chain for $\Lambda_\oplus$ [Dal Lago et al'14]



Theorem (Dal Lago et al'14,Crubillé et al'14)

	$\sim \subseteq \equiv^{ctx}$	$\equiv^{ctx} \subseteq \sim$	$\precsim \subseteq \leq^{ctx}$	$\leq^{ctx} \subseteq \precsim$
CBN	✓	✗	✓	✗
CBV	✓	✓	✓	✗

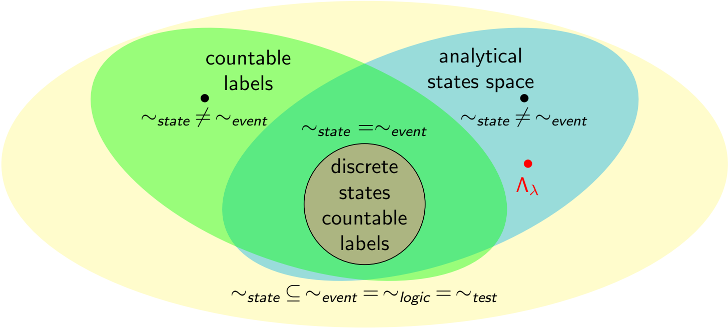
The subject of this talk: the continuous case

Definition (Labelled Markov Process)

A triple $(\mathcal{S}, \mathcal{A}, \{h_a \mid a \in \mathcal{A}\})$, where $\mathcal{S}$ is measurable, $\mathcal{A}$ is an arbitrary set, and for every $a \in \mathcal{A}$ the map $h_a : \mathcal{S} \times \Sigma_{\mathcal{S}} \to [0, 1]$ is a sub-probability kernel.

From the literature

Two distincts notions of bisimulations exist for LMPs:



State bisimulation

Definition (for Labelled Markov Chain)

A symmetric relation R on $\mathcal{S}$ is a bisimulation when:

$$s \, R \, t \quad \Rightarrow \quad \forall a \in \mathcal{L}, \forall X \; R\text{-closed}, h_a(s, X) = h_a(t, X).$$

Proposition (Dal Lago-Gavazzo'19)

Applicative state bisimulation is sound (w.r.t. context equivalence) for Λ_λ.

State bisimulation

Definition (for Labelled Markov Process)

A symmetric relation R on $\mathcal{S}$ is a bisimulation when:

$$s\,R\,t \quad \Rightarrow \quad \forall a \in \mathcal{L}, \forall X \in \Sigma_{\mathcal{S}}\ R\text{-closed}, h_a(s,X) = h_a(t,X).$$

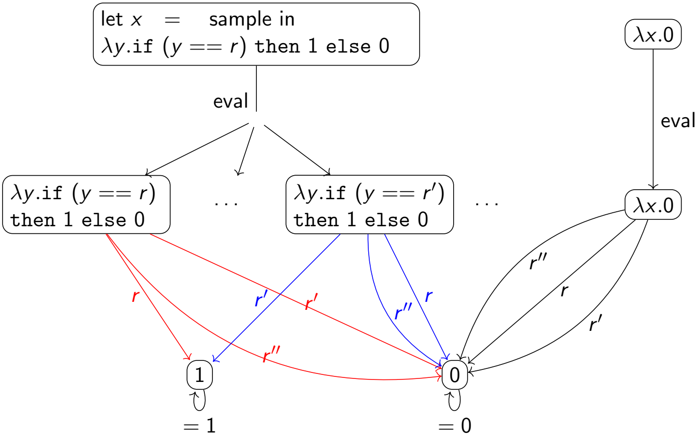
Proposition (Dal Lago-Gavazzo'19)

Applicative state bisimulation is sound (w.r.t. context equivalence) for Λ_λ.

Theorem

Applicative state bisimulation is not complete for Λ_λ.

Proof.



M and N are context equivalent [Staton et al'21], but not bisimilar.

Event bisimulation for Λ_λ

Definition

An event bisimulation on a LMP $(\mathscr{M}, \Sigma, \{h_a :| a \in \mathcal{A}\})$ is a sub-σ-algebra Λ of Σ,such that $(\mathscr{M}, \Lambda, \{h_a \mid a \in \mathcal{A}\})$ is a LMP.

Proposition

logical caracterisation [Breugel et al'05]

Theorem

Applicative event bisimulation is complete, but not sound.

Proof.

- completeness proof: uses the logical caracterisation
- counter-example for soundness:

$$M := \text{let } x = \text{sample in } (\lambda y.((\text{if } x == y \text{ then } 1 \text{ else } 0) \oplus x)),$$
$$N := \text{let } x = \text{sample in } (\lambda y.(0 \oplus x)),$$
$$C = (\text{ let } z = [\,] \text{ in } z(z1)).$$

M and N are event bisimilar, but not context equivalent.

Λ_λ with only continuous primitive functions

$$
\begin{aligned}
M \in \Lambda_{\lambda,c} ::= \; & x \;\big|\; \lambda x^A \cdot M \;\big|\; (MN) \;\big|\; (YN) \\
& \big|\; \text{ifz } (M, N, L) \;\big|\; \text{let}(x, M, N) \\
& \big|\; M \oplus N \;\big|\; \underline{n} \;\big|\; \text{succ } (M) \;\big|\; \text{pred } (M), \qquad n \in \mathbb{N} \\
& \big|\; \text{sample} \;\big|\; \underline{r} \;\big|\; \underline{f}, \qquad r \in \mathbb{R}, f : \mathbb{R} \to \mathbb{R} \textbf{ continuous}
\end{aligned}
$$

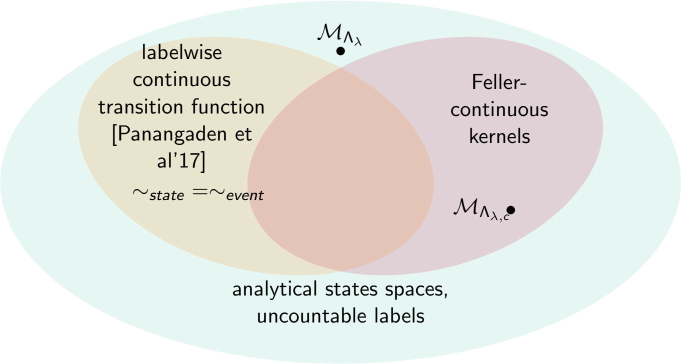
Previous counter-examples cannot be written in this language.

Question:

Can we recover $\sim_{event} = \sim_{context} = \sim_{state}$?

Our demarch

Build a class of LMP with **uncountable labels** such that the two bisimulations coincide.



Feller Continuous LMPs

Definition

X a polish space, $(\mu_n)_{n\in\mathbb{N}}$ a sequence of measures over X.
$(\mu_n)_{n\in\mathbb{N}}$ *converges weakly* toward μ when $\forall f : X \to \mathbb{R}$ bounded and continuous function:

$$\lim_{n\to\infty} \int_X f.d\mu_n = \int_X f.d\mu.$$

Definition

A LMP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \{h_a \mid a \in \mathcal{A}\})$ with $\mathcal{S}, \mathcal{A}$ polish spaces. $\mathcal{M}$ is *Feller continuous* when:

- for every $a \in \mathcal{A}$, the map $h_a : \mathcal{S} \to \mathrm{Distrs}(\mathcal{S})$ is continuous;
- for every $s \in \mathscr{M}$ the map $h_{(.)}(s) : a \in \mathcal{A} \mapsto h_a(s) \in \mathrm{Distrs}(\mathcal{S})$ is continuous.

Bisimulations for Feller continuous LMPs

Theorem

For Feller continuous LMPs, state bisimulation and event bisimulation coincide.

Proof.

Uses a result from optimal transport [Villani'08]. □

Theorem

$\Lambda_{\lambda,c}$ *is Feller continuous, thus* $\sim_{event} = \sim_{context} = \sim_{state}$.

Conclusion:

Contribution

- an extensive picture of the full abstraction problem for applicative similarité on Λ_λ
- the definition of a new class of LMP (Feller-continuous LMPs) where state and event bisimilarity coincide.

Perspectives

- bisimulation for a language with continuous probabilities and bayesian reasonning...
- quantitative reasonning (i.e. distances) for a continuous language.