On Randomization in (Higher-Order) Programming

Part V

Ugo Dal Lago

Escuela de Ciencias Informáticas, Buenos Aires, July 2023

Part I

From Randomized to Bayesian

From Randomised Algorithms to Bayesian Programming

- Binary probabilistic choice is perfectly adequate to model randomised computation.

From Randomised Algorithms to Bayesian Programming

- Binary probabilistic choice is perfectly adequate to model randomised computation.

Theorem

The class of computable probabilistic functions coincides with the class of probabilistic functions computable by $\mathsf{PCF}_{\oplus}$.

From Randomised Algorithms to Bayesian Programming

- Binary probabilistic choice is perfectly adequate to model randomised computation.

Theorem

The class of computable probabilistic functions coincides with the class of probabilistic functions computable by $\mathsf{PCF}_{\oplus}$.

- In recent years, starting from the pioneering works on languages like CHURCH, ANGLICAN, or HANSEI, functional programs have been also employed as means to represent probabilistic *models* rather than *algorithms*.
- The languages above can be modeled [Staton2017] as λ-calculi endowed with two new operators:
  - `sample`, modeling sampling from the uniform distribution on $[0, 1]$.
  - `score`, which takes a positive real number r as a parameter, and modify the *weight* of the current probabilistic branch by multiplying it by r.

Beyond Randomized Algorithms: The Grass Problem

- You get up in the morning, and notice that the grass in your garden is wet.

Beyond Randomized Algorithms: The Grass Problem

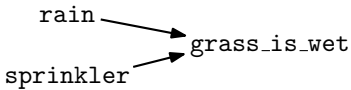
- You get up in the morning, and notice that the grass in your garden is wet.
- You know that there is 30% probability that it rains during the night.

Beyond Randomized Algorithms: The Grass Problem

- You get up in the morning, and notice that the grass in your garden is wet.
- You know that there is 30% probability that it rains during the night.
- The sprinkler is activated once every two days.

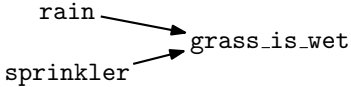
Beyond Randomized Algorithms: The Grass Problem

- You get up in the morning, and notice that the grass in your garden is wet.
- You know that there is 30% probability that it rains during the night.
- The sprinkler is activated once every two days.
- You further know that:
  - *If it rains*, there is 90% probability that the grass is wet the following morning.
  - *If the sprinkler is activated*, there is 80% probability that the grass is wet the following morning.
  - *In any other case*, there is anyway a 10% probability that the grass is wet.
- In other words, you are in a situation like the following:



Beyond Randomized Algorithms: The Grass Problem

- You get up in the morning, and notice that the grass in your garden is wet.
- You know that there is 30% probability that it rains during the night.
- The sprinkler is activated once every two days.
- You further know that:
  - *If it rains*, there is 90% probability that the grass is wet the following morning.
  - *If the sprinkler is activated*, there is 80% probability that the grass is wet the following morning.
  - *In any other case*, there is anyway a 10% probability that the grass is wet.
- In other words, you are in a situation like the following:



- Now: how likely it is that it rained?

The Grass Model

```
let grass_model () =    (* unit -> bool *)
  let rain = flip 0.3 and sprinkler = flip 0.5 in
  let grass_is_wet =
    (flip 0.9 && rain) || (flip 0.8 && sprinkler) || flip 0.1 in
  if not grass_is_wet then fail ();
  rain
```

$\mathsf{PCF}_{\texttt{sample},\texttt{score}}$: Terms, Typing Rules, and Reduction

Terms $\quad M, N ::= \texttt{sample} \mid \texttt{score}(V).$

$\mathsf{PCF}_{\texttt{sample,score}}$: Terms, Typing Rules, and Reduction

Terms $\quad M, N ::= \texttt{sample} \mid \texttt{score}(V).$

Typing Rules $\quad \dfrac{}{\Gamma \vdash \texttt{sample} : \textsc{Num}}\ \mathsf{A} \qquad \dfrac{\Gamma \vdash V : \textsc{Num}}{\Gamma \vdash \texttt{score}(V) : \textsc{Unit}}\ \mathsf{C}$

$\mathsf{PCF}_{\mathtt{sample},\mathtt{score}}$: Terms, Typing Rules, and Reduction

Terms $\quad M, N ::= \mathtt{sample} \mid \mathtt{score}(V).$

Typing Rules $\quad \dfrac{}{\Gamma \vdash \mathtt{sample} : \textsc{Num}}\ \mathsf{A} \qquad \dfrac{\Gamma \vdash V : \textsc{Num}}{\Gamma \vdash \mathtt{score}(V) : \textsc{Unit}}\ \mathsf{C}$

- One needs to switch from distributions to *measures*, and assume the underlying set, namely $\mathbb{R}$ to have the structure of a measurable space
- Adapting the rule for `let`-terms naturally leads to

$$\frac{M \to \mu}{\mathtt{let}\ M = x\ \mathtt{in}\ N \to \mathtt{let}\ \mu = x\ \mathtt{in}\ N}$$

  where $\mathtt{let}\ \mu = x\ \mathtt{in}\ N$ should itself be a measure.
- The key rule in step-indexed reduction needs to be adapted:

$$\frac{M \to \mu \quad \forall N \in \mathsf{SUPP}(\mu).N \Rightarrow_n \sigma_N}{M \Rightarrow_{n+1} A \mapsto \int \sigma_N(A)\mu(dN)}$$

The Operational Meaning of Bayesian Terms

- It can well be that $\langle M \rangle = \mu$, where μ sums to something *strictly higher* than 1.

The Operational Meaning of Bayesian Terms

- It can well be that $\langle M \rangle = \mu$, where μ sums to something *strictly higher* than 1.
- How should we interpret $\mu(A)$ for a measurable set of terms A?
- We need to *normalize* μ!

The Operational Meaning of Bayesian Terms

- It can well be that $\langle M \rangle = \mu$, where μ sums to something *strictly higher* than 1.
- How should we interpret $\mu(A)$ for a measurable set of terms A?
- We need to *normalize* μ!
- Explicitly building a normalized version of μ (if it exists) is the goal of so-called *inference algorithms*.

The Operational Meaning of Bayesian Terms

- It can well be that $\langle M \rangle = \mu$, where μ sums to something *strictly higher* than 1.
- How should we interpret $\mu(A)$ for a measurable set of terms A?
- We need to *normalize* μ!
- Explicitly building a normalized version of μ (if it exists) is the goal of so-called *inference algorithms*.
- The operational semantics we have just introduced, called **distribution-based** is thus just an *idealized* form of semantics.
- An *executable* semantics can be given in the form of **sampling-based semantics**.

Sampling-Based Semantics (1)

$$\langle (\lambda x.M)V, s\rangle \overset{1}{\twoheadrightarrow} \langle M[V/x], s\rangle$$

$$\langle \texttt{let } V = x \texttt{ in } M, s\rangle \overset{1}{\twoheadrightarrow} \langle M[V/x], s\rangle$$

$$\langle \texttt{if } 0 \texttt{ then } M \texttt{ else } N, s\rangle \overset{1}{\twoheadrightarrow} \langle M, s\rangle$$

$$\langle \texttt{if } r \texttt{ then } M \texttt{ else } N, s\rangle \overset{1}{\twoheadrightarrow} \langle N, s\rangle \text{ if } r \neq 0$$

$$\langle \texttt{sample}, r :: s\rangle \overset{1}{\twoheadrightarrow} \langle r, s\rangle$$

$$\langle \texttt{score}(r), s\rangle \overset{r}{\twoheadrightarrow} \langle \star, s\rangle$$

$$\langle f(r_1, \cdots, r_n), s\rangle \overset{1}{\twoheadrightarrow} \langle f^*(r_1 \ldots, r_n), s\rangle$$

$$\frac{\langle M, s\rangle \overset{r}{\twoheadrightarrow} \langle L, t\rangle}{\langle \texttt{let } M = x \texttt{ in } N, s\rangle \overset{r}{\twoheadrightarrow} \langle \texttt{let } L = x \texttt{ in } N, s\rangle}$$

Sampling-Based Semantics (2)

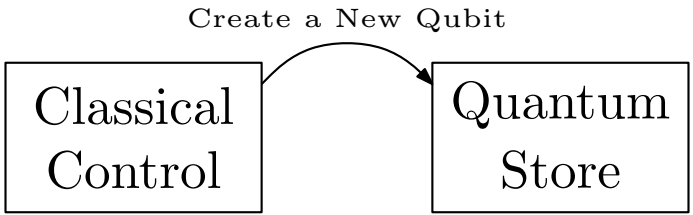
$$\frac{}{\langle V, s\rangle \overset{1}{\Rrightarrow} \langle V, s\rangle} \qquad \frac{\langle M, s\rangle \overset{r}{\twoheadrightarrow} \langle N, t\rangle \quad \langle N, t\rangle \overset{s}{\Rrightarrow} \langle L, u\rangle}{\langle M, s\rangle \overset{r\cdot s}{\Rrightarrow} \langle L, u\rangle}$$

Part II

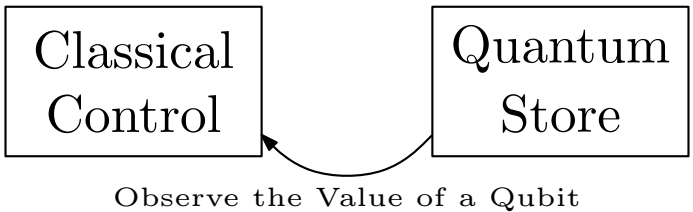
From Randomized to Quandum Higher-Order Programming

Quantum Data and Classical Control

Quantum Data and Classical Control



Quantum Data and Classical Control



Quantum Data and Classical Control

A Naïve Attempt

- While looking for a quantum generalization of the λ-calculus, one could be tempted to proceed merely by enriching its syntax of **terms** as follows:

$$M, N \ ::= \ x \ \mid \ \lambda x.M \ \mid \ MN \ \mid \ r \ \mid \ \mathsf{U} \ \mid \ \texttt{new} \ \mid \ \texttt{meas}$$

  where:
  - r is a **quantum variable** pointing to the quantum store;
  - the operator `new` **creates** a new qubit;
  - the operator `meas` **measures** a qubit from the quantum store;
  - U applies a unitary transform to one (or more) qubits from the quantum store.
- This would be enough to express, e.g., some simple quantum circuits:

$$\lambda x.\lambda y.\mathsf{CNOT}\langle \mathsf{H}\ x, y\rangle \qquad\qquad \lambda x.\texttt{meas}(\mathsf{H}(\texttt{new}\ x))$$

]

A Naïve Attempt

$$[\emptyset, (\lambda x.\lambda y.\mathsf{CNOT}\langle \mathsf{H}\ x, y\rangle)\langle \mathtt{new}\ \overline{0}, \mathtt{new}\ \overline{0}\rangle]$$

A Naïve Attempt

$$[\emptyset, (\lambda x.\lambda y.\mathsf{CNOT}\langle \mathsf{H}\ x, y\rangle)\langle \mathtt{new}\ \overline{0}, \mathtt{new}\ \overline{0}\rangle]$$

$$\to^* [|r \to 0\rangle \otimes |q \to 0\rangle, (\lambda x.\lambda y.\mathsf{CNOT}\langle \mathsf{H}\ x, y\rangle)\langle r, q\rangle]$$

A Naïve Attempt

$$[\emptyset, (\lambda x.\lambda y.\mathsf{CNOT}\langle \mathsf{H}\ x, y\rangle)\langle \mathtt{new}\ \overline{0}, \mathtt{new}\ \overline{0}\rangle]$$

$$\to^* [|r \to 0\rangle \otimes |q \to 0\rangle, (\lambda x.\lambda y.\mathsf{CNOT}\langle \mathsf{H}\ x, y\rangle)\langle r, q\rangle]$$

$$\to [|r \to 0\rangle \otimes |q \to 0\rangle, \mathsf{CNOT}\langle \mathsf{H}\ r, q\rangle]$$

A Naïve Attempt

$$[\emptyset, (\lambda x.\lambda y.\mathsf{CNOT}\langle \mathsf{H}\ x, y\rangle)\langle \mathtt{new}\ \overline{0}, \mathtt{new}\ \overline{0}\rangle]$$

$$\to^* [|r \to 0\rangle \otimes |q \to 0\rangle, (\lambda x.\lambda y.\mathsf{CNOT}\langle \mathsf{H}\ x, y\rangle)\langle r, q\rangle]$$

$$\to [|r \to 0\rangle \otimes |q \to 0\rangle, \mathsf{CNOT}\langle \mathsf{H}\ r, q\rangle]$$

$$\to [\frac{1}{\sqrt{2}}|r \to 0, q \to 0\rangle + \frac{1}{\sqrt{2}}|r \to 1, q \to 0\rangle, \mathsf{CNOT}\langle r, q\rangle]$$

A Naïve Attempt

$$[\emptyset, (\lambda x.\lambda y.\mathsf{CNOT}\langle \mathsf{H}\ x, y\rangle)\langle \mathtt{new}\ \overline{0}, \mathtt{new}\ \overline{0}\rangle]$$

$$\to^* [|r \to 0\rangle \otimes |q \to 0\rangle, (\lambda x.\lambda y.\mathsf{CNOT}\langle \mathsf{H}\ x, y\rangle)\langle r, q\rangle]$$

$$\to [|r \to 0\rangle \otimes |q \to 0\rangle, \mathsf{CNOT}\langle \mathsf{H}\ r, q\rangle]$$

$$\to [\frac{1}{\sqrt{2}}|r \to 0, q \to 0\rangle + \frac{1}{\sqrt{2}}|r \to 1, q \to 0\rangle, \mathsf{CNOT}\langle r, q\rangle]$$

$$\to [\frac{1}{\sqrt{2}}|r \to 0, q \to 0\rangle + \frac{1}{\sqrt{2}}|r \to 1, q \to 1\rangle, \langle r, q\rangle]$$

A Naïve Attempt

- This approach has some **fundamental problems**, however.

A Naïve Attempt

- This approach has some **fundamental problems**, however.
- Take the term $(\lambda x.\mathsf{CNOT}\langle x, x\rangle)\mathtt{new}$ and suppose, as in the previous example, to reduce it in CBV:

$$[\emptyset, (\lambda x.\mathsf{CNOT}\langle x, x\rangle)(\mathtt{new}\overline{0})]$$

A Naïve Attempt

- This approach has some **fundamental problems**, however.
- Take the term $(\lambda x.\mathsf{CNOT}\langle x, x\rangle)\mathtt{new}$ and suppose, as in the previous example, to reduce it in CBV:

$$\begin{aligned}&[\emptyset, (\lambda x.\mathsf{CNOT}\langle x, x\rangle)(\mathtt{new}\overline{0})]\\ \to\ &[|r \to 0\rangle, (\lambda x.\mathsf{CNOT}\langle x, x\rangle)r]\end{aligned}$$

A Naïve Attempt

- This approach has some **fundamental problems**, however.
- Take the term $(\lambda x.\mathsf{CNOT}\langle x, x\rangle)\texttt{new}$ and suppose, as in the previous example, to reduce it in CBV:

$$\begin{aligned} &[\emptyset, (\lambda x.\mathsf{CNOT}\langle x, x\rangle)(\texttt{new}\overline{0})] \\ \to\ &[|r \to 0\rangle, (\lambda x.\mathsf{CNOT}\langle x, x\rangle)r] \\ \to\ &[|r \to 0\rangle, \mathsf{CNOT}\langle r, r\rangle] \end{aligned}$$

- Qubits can be freely duplicated or erased!
  - And this goes against the principles of quantum mechanics.
- We need a way to force qubits to be used **exactly** once when passed to functions.

Linearity and the λ-Calculus

- First of all, let us impose that in any abstraction $\lambda x.M$, the variable x occurs **exactly** once in M.
  - Copying and erasure **not** possible, anymore.
  - But the calculus **loses** much of its expressive power.

Linearity and the λ-Calculus

- First of all, let us impose that in any abstraction $\lambda x.M$, the variable x occurs **exactly** once in M.
  - Copying and erasure **not** possible, anymore.
  - But the calculus **loses** much of its expressive power.
- The idea is to **refine** the calculus in such a way as to reintroduce erasure and copying, but in a controlled way.
  - We mark as $!M$ all those subterms which can be (potentially) copied or erased;
  - Besides the usual linear abstraction $\lambda x.M$, there is a non-linear abstraction $\lambda !x.M$.
- Altogether, then, the language of terms becomes:

$$M, N ::= x \mid \lambda x.M \mid \lambda !x.M \mid MN \mid !M$$
$$r \mid \mathsf{U} \mid \texttt{new} \mid \texttt{meas}$$

where we insist that any term in the form $!M$ does not contain any quantum variable.

Operational Semantics

Evaluation Contexts

$$E ::= [\cdot] \mid EM \mid ME \mid \lambda x.E \mid \lambda !x.E$$

Operational Semantics

Evaluation Contexts

$$E ::= [\cdot] \mid EM \mid ME \mid \lambda x.E \mid \lambda !x.E$$

Small Step Rules

$$\begin{aligned}
[\mathcal{Q}, E[(\lambda x.M)N]] &\Rightarrow \{[\mathcal{Q}, E[M[N/x]]]^1\} \\
[\mathcal{Q}, E[(\lambda !x.M)!N]] &\Rightarrow \{[\mathcal{Q}, E[M[N/x]]]^1\} \\
[\mathcal{Q}, E[\texttt{meas}\ r]] &\Rightarrow \{[\Pi_r^0(\mathcal{Q}), E[0]]^{\Xi_r^0(\mathcal{Q})}, [\Pi_r^1(\mathcal{Q}), E[1]]^{\Xi_r^1(\mathcal{Q})}\} \\
[\mathcal{Q}, E[\mathsf{U}\ r]] &\Rightarrow \{[\mathsf{U}_r(\mathcal{Q}), E[r]]^1\} \\
[\mathcal{Q}, E[\texttt{new}\ 0]] &\Rightarrow \{[\mathcal{Q} \otimes |r \to 0\rangle, E[r]]^1\} \\
[\mathcal{Q}, E[\texttt{new}\ 1]] &\Rightarrow \{[\mathcal{Q} \otimes |r \to 1\rangle, E[r]]^1\}
\end{aligned}$$

Expressive Power

Theorem

Any uniform quantum circuit family $\{C_n\}_{n\in\mathbb{N}}$ *is simulated by a* `meas`*-free term* M*.*

Expressive Power

Theorem

Any uniform quantum circuit family $\{C_n\}_{n\in\mathbb{N}}$ *is simulated by a* `meas`*-free term* M*.*

- Given s, the term M computes a *code* for $C_{|s|}$
- Then, it applies $C_{|s|}$ to s.

Expressive Power

Theorem

Any uniform quantum circuit family $\{C_n\}_{n\in\mathbb{N}}$ *is simulated by a* `meas`*-free term* M*.*

- Given s, the term M computes a *code* for $C_{|s|}$
- Then, it applies $C_{|s|}$ to s.

Theorem

Any `meas`*-free term* M *computes the same function as a uniform quantum circuit family* $\{C_n\}_{n\in\mathbb{N}}$*.*

Expressive Power

Theorem

Any uniform quantum circuit family $\{C_n\}_{n\in\mathbb{N}}$ *is simulated by a* `meas`*-free term* M*.*

- Given s, the term M computes a *code* for $C_{|s|}$
- Then, it applies $C_{|s|}$ to s.

Theorem

Any `meas`*-free term* M *computes the same function as a uniform quantum circuit family* $\{C_n\}_{n\in\mathbb{N}}$*.*

- A term M can be evaluated by first reducing all the "classical" redexes, and the reducing the "quantum" ones.
- This is possible due to a standardization result.