Name Generation and Higher-order Probabilistic Programming
(Or is new=rnd?)

LAFI 2020, New Orleans

Dario Stein\textsuperscript{1}, Sam Staton\textsuperscript{1}, Michael Wolman\textsuperscript{2}

January 21, 2020

\textsuperscript{1}University of Oxford \textsuperscript{2}McGill University
Aim: Unifying name generation & probabilistic programming

1. Names and their relation to PPLs
2. $\nu$-calculus: A higher-order language for name generation
3. Denotation semantics
   3.1 Classical semantics (nominal sets)
   3.2 Probabilistic semantics (higher-order probability)
4. Quasi-Borel spaces and the full abstraction problem
What are names?

Examples

1. $\alpha$-equivalence
   
   \[ \lambda x.x \, z \equiv_\alpha \lambda y.y \, z \]

2. memory locations
   
   ```
   int* x = new int;
   let y = ref() : unit ref
   ```

3. metaprogramming
   
   ```
   (defmacro (while condition . body)
     (let ((loop (gensym)))
       `(let ,loop ()
         (cond ,condition (begin ,body) ,loop )))))
   ```

Key properties

- *atomic* – only comparable for identity
- freshly generated
- usually: stateful effect
Names in PPL

gensym: exchangeable random primitive (XRP) in Bayesian nonparametrics.

<table>
<thead>
<tr>
<th>Base distribution for clustering with Dirichlet process</th>
</tr>
</thead>
<tbody>
<tr>
<td>(define draw-class</td>
</tr>
<tr>
<td>(DPmem 1.0 gensym))</td>
</tr>
<tr>
<td>(define class</td>
</tr>
<tr>
<td>(mem ((\lambda) (obj) (draw-class))))</td>
</tr>
<tr>
<td>(define class-weight</td>
</tr>
<tr>
<td>(mem ((\lambda) (obj-class feature) (beta 1.0 1.0))))</td>
</tr>
</tbody>
</table>

In practice, gensym is used like a probability distribution
Names vs. random numbers

Names

<table>
<thead>
<tr>
<th>Names</th>
<th>Random samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>let x : name  = new() in</td>
<td>let x : real = rnd() in</td>
</tr>
<tr>
<td>let y : name  = new() in</td>
<td>let y : real = rnd() in</td>
</tr>
<tr>
<td>x == y</td>
<td>x == y</td>
</tr>
<tr>
<td>≡ false</td>
<td>≡ false</td>
</tr>
</tbody>
</table>

Formal analogy (program equations)

1. commutative and discardable effects
2. fresh samples are almost surely distinct (continuous distributions: uniform, gaussian)
Question: “Is name generation just random sampling?”

If so:

- probability theory includes name generation
- prove things about name-generating programs using probability

Difficulty: Interaction of names & higher-order functions
Names & higher-order functions

Names & Closures

val f, g : name → bool

let f = (let x = new() in fun y → (y == x))

let g = fun y → false

The functions $f$ and $g$ are contextually equivalent.

- $x$ is private inside the closure $f$
- garbage collection, escape analysis
- how to prove such equivalences?
Privacy equation:

\[
(\text{let } x = \text{new()} \text{ in fun } y \rightarrow (y == x)) \equiv \text{fun } y \rightarrow \text{false}
\]

What about the analogous statement for random numbers?

\[
\text{let } x = \text{rnd()} \text{ in fun } y \rightarrow (y == x) \equiv \text{fun } y \rightarrow \text{false}
\]

This is a statement about random functions \( \mathbb{R} \rightarrow 2 \).

Later

- Make sense of this statement (requires a model of probability w/ higher-order functions)
- Prove that it’s true
Stark’s $\nu$-calculus ['93]: Simply-typed call-by-value $\lambda$-calculus

1. types $\nu$ (name), $o$ (bool) and $\tau_1 \to \tau_2$
2. equality tests & conditionals
3. construct $\nu a. M$ to allocate a fresh name $a$

Used to study observational equivalence $\approx$

**Freshness:**

$$\nu x. \nu y. (x = y) \approx false.$$ 

**Privacy equation:**

$$\nu x. \lambda y. (x = y) \approx \lambda y. false.$$
Name generation is subtle

\[ \nu x. \lambda y. x \not\equiv \lambda y. \nu x. x \]

\[ \nu a. \nu b. \lambda x. \text{if } (x = a) \text{ then } a \text{ else } b \approx \nu b. \lambda x. b \]

\[ \nu a. \nu b. \lambda x. \text{if } (x = b) \text{ then } a \text{ else } b \not\equiv \nu b. \lambda x. b \]
Classical semantics: Nominal sets [Pitts]

- set theory with atoms $\mathbb{A}$, all constructions equivariant under renaming
- name abstraction monad $T$

$$\langle a \rangle (a, b) = \langle c \rangle (c, b) \in T(\mathbb{A} \times \mathbb{A}).$$

- **Full abstraction**: do observationally equivalent programs have the same semantics?
  - No
Failure of full abstraction

Privacy equation does not hold in **Nom**

\[
[[\nu x. \lambda y. (y = x)]] = \langle x \rangle \{x\} \in T(2^A)
\]

\[
[[\lambda y. \text{false}]] = \langle \rangle \emptyset
\]

are distinct because the *nonemptyness check*

\[
\exists : 2^A \to 2
\]

is equivariant.

⇒ Logical relations to remedy this
Theorem (Probabilistic privacy equation)

It holds that

\[ [[\nu x.\lambda y.(y = x)]] = [[\lambda y.\text{false}]] \in P(2^\mathbb{R}) \]

In stats terms, if

\[ X \sim \mathcal{U}[0, 1] \]
\[ A = \{X\} \]
\[ B = \emptyset \]

then \textbf{A and B have the same distribution!}
1. **Continuous distributions** ⇒ Measure theory. Which \( \sigma \)-algebra to put on \( 2^\mathbb{R} \)?

2. Equality checks are **discontinuous maps**; spaces of continuous functions not sufficient

3. General **higher-order functions** don’t combine with measure theory [Aumann ’61]

Quasi-Borel spaces [Staton & al, ’17] are a model of all of the above.
Theorem

Quasi-Borel spaces are a sound and correct probabilistic model of the $\nu$-calculus.

Next

Qbs semantics is fully abstract up to first-order function types $\tau_1 \rightarrow \cdots \rightarrow \tau_n$, $\tau_i \in \{o, \nu\}$.

Proof for $\tau_n = o$. All names are private, eliminate sampling (privacy equation only).

Sketch for $\tau_n = \nu$. Normalize to see which names are private. Eliminate those (few more equations).
Let $X \sim \mathcal{U}[0, 1], A = \{X\}$.

1. for any $x_0 \in \mathbb{R}$

   $x_0 \in A \iff x_0 \in \emptyset \quad \text{a.s.}$

2. if $\mu$ $\sigma$-finite then

   $\mu(A) = 0 = \mu(\emptyset) \quad \text{a.s.}$
Let $X \sim \mathcal{U}[0, 1]$, $A = \{X\}$. Assume that

$$\exists : 2^\mathbb{R} \to 2$$

was a morphism in $\text{Qbs}$.

1. Let $B \subseteq \mathbb{R}^2$ be any Borel set
2. $\chi_B : \mathbb{R} \to \mathbb{R} \to 2$ is a morphism
3. $\lambda x. \exists (\chi_B(x)) : \mathbb{R} \to 2$ is a morphism
4. that is, the projection $\pi(B) \subseteq \mathbb{R}$ is Borel. \( \text{QED} \)
Proof of the privacy equation

Let \( X \sim \mathcal{U}[0, 1], A = \{X\} \).

1. The law of \( A \) is a measure on the space \( 2^\mathbb{R} = \Sigma_\mathbb{R} \) of Borel sets. \( \sigma \)-algebra \( \Sigma_{2^\mathbb{R}} \) induced from qbs structure; \( \mathcal{U} \in \Sigma_{2^\mathbb{R}} \) iff “Borel on Borel” [Kechris ’87]

\[
\forall B \subseteq \mathbb{R}^2 \text{ Borel }, \{x : B_x \in \mathcal{U}\} \in \Sigma_\mathbb{R}.
\]

2. **Thm** For any Borel on Borel \( \mathcal{U} \)

\[
\emptyset \in \mathcal{U} \iff \{x\} \in \mathcal{U} \text{ for almost all } x.
\]

Elegant proof using descriptive set theory.

3. **We cannot measurably distinguish** \( A \) and \( \emptyset \)!
Takeaway

- Are names random numbers? Yes (in a precise way)
  - Out-of-the-box probabilistic semantics is more abstract than Nom
  - Unify PPL and name generation
  - Justify program equations about name generation using probability
- New understanding of Qbs function spaces
  - Tool: Descriptive set theory
- Measurability as abstraction: Randomization is anonymization (differential privacy)
Future directions

- Descriptive set theory $\iff$ computability theory
  - Borel & Turing inseparability
- Connections to logical relations
  - Qbs structure $M_X \subseteq X^\mathbb{R}$ is an $\mathbb{R}$-ary relation