**Name generation**

- **Names** are unique identifiers like GUIDs, memory locations, cluster names
- `gensym` creates a fresh name
- Often treated as a probability distribution, e.g. as a prior for a Dirichlet process
  - [e.g. Roy & al.'08]
- Can we use random samples as fresh names?
  - ✓ Continuous samples have 0 collision probability
  - Subtlety: Names can stay private inside functions
    - Example: the following function name - ```> bool always return false``` (A*)
- Stark’s ν-calculus [Stark’93] = higher-order programming with names
- We can translate ν-calculus into a higher-order PPL with continuous distributions where
  - names become real numbers
  - fresh name generation becomes sampling
- Privacy can be very subtle, e.g. compare Higher-Order Probabilistic Programming and Name Generation
  - [Marcin Sabok, Sam Staton, Dario Stein, Michael Wolman]
  - tl;dr `gensym = rnd`
- Name generation and probabilistic programming have interesting connections
  - Random samples are a good semantics for fresh names
    - Was folklore for ground programs (various gensym implementations, randomized GUIDs)
    - We proved that the semantics is abstract even when higher-order functions are involved
    - Unified semantics of probability + names, more refined than traditional semantics (Nominal sets)
- Name-generation ideas like privacy naturally appear when reasoning about probabilistic programs
  - Randomization = anonymization
  - Bayesian inference on function types must be limited, as not to leak private information

**Main result**

- Stark’s ν-calculus [Stark’93] = higher-order programming with names
- We can translate ν-calculus into a higher-order PPL with continuous distributions where
  - names become real numbers
  - fresh name generation becomes sampling
- Privacy can be very subtle, e.g. compare
  - ```let a = normal(0,1) in fun y -> if x == y then a else 0``` (A)
  - ```let a = normal(0,1) in fun y -> 0``` (B)

  - a is revealed
  - b remains private
  - a can then be used to reveal b
  - for this, the function must be called twice

**A theory of random functions**

- Can the following random function real -> bool
  - ```let x = normal(0,1) in fun y -> x == y``` (A)
  - Yes: We prove a ‘privacy equation’ – randomizing a value anonymizes it (x is private)
  - We cannot condition on functions being equal, as this would distinguish (A) and (B)

**Conclusions**

**Theorem:**

**Quasi-Borel space semantics is fully abstract for ν-calculus up to first-order function types**

- Full abstraction: Two name generating programs are equivalent if and only if the corresponding probabilistic programs are equivalent, e.g. (A*) and (A)

**Denotational Semantics: Quasi-Borel spaces**

Which mathematical framework can analyze higher-order functions + continuous distributions?
- Measure theory is insufficient [Aumann’61]; quasi-Borel spaces [Heunen’17] are a convenient tool
- Ours is the first work to analyze quasi-Borel function spaces in detail.

**Proof sketch:** Measurable collections $\mathcal{U} \subseteq \text{Meas}(\mathbb{R}, 2)$ are known as Borel-on-Borel [Kechris’87]
- Using descriptive set theory: If $\mathcal{U} \subseteq \text{Meas}(\mathbb{R}, 2)$ is Borel-on-Borel and $\beta \in \mathcal{U}$
  then $\{x : \{x\} \notin \mathcal{U}\}$ is at most countable.