On Higher-Order Probabilistic Computation:
Relational Reasoning, Termination, and Bayesian Programming

Ugo Dal Lago

(Based on joint work with *Michele Alberti*, *Raphaëlle Crubillé*, *Charles Grellois*, *Davide Sangiorgi*,...)

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Inria informatiques mathématiques

IFIP WG 2.2 Annual Meeting, Brno, September 17th

Probabilistic Models

- The **environment** is supposed not to behave *deterministically*, but *probabilistically*.

Probabilistic Models

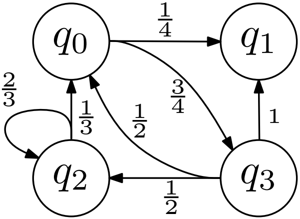
- The **environment** is supposed not to behave *deterministically*, but *probabilistically*.
- Crucial when modeling **uncertainty**.

Probabilistic Models

- The **environment** is supposed not to behave *deterministically*, but *probabilistically*.
- Crucial when modeling **uncertainty**.
- Useful to handle **complex** domains.

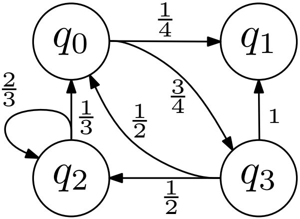
Probabilistic Models

- The **environment** is supposed not to behave *deterministically*, but *probabilistically*.
- Crucial when modeling **uncertainty**.
- Useful to handle **complex** domains.
- **Example:**



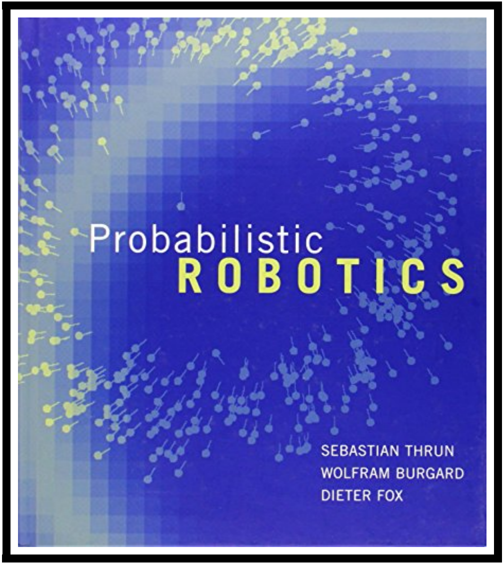
Probabilistic Models

- The **environment** is supposed not to behave *deterministically*, but *probabilistically*.
- Crucial when modeling **uncertainty**.
- Useful to handle **complex** domains.
- **Example:**



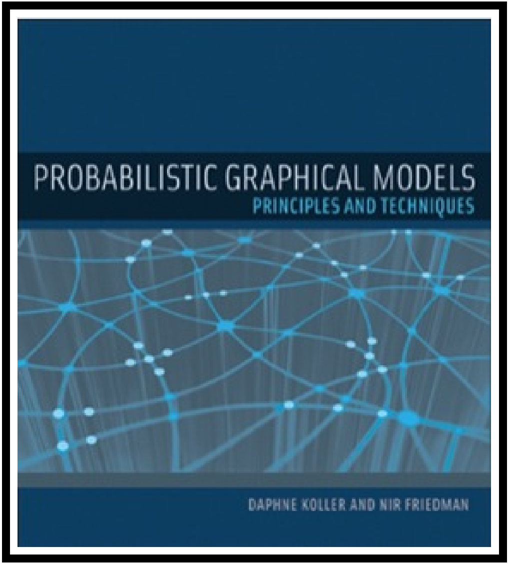
- **Abstractions:**
  - (Labelled) Markov Chains.

Probabilistic Models



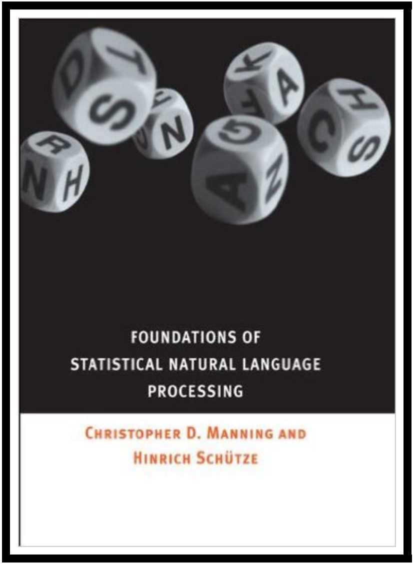
ROBOTICS

Probabilistic Models



ARTIFICIAL INTELLIGENCE

Probabilistic Models



NATURAL LANGUAGE
PROCESSING

Randomized Computation

- Algorithms and automata are assumed to have the ability to **sample** from a distribution [dLMSS1956,R1963].

Randomized Computation

- Algorithms and automata are assumed to have the ability to **sample** from a distribution [dLMSS1956,R1963].
- This is a **powerful tool** when solving computational problems.

Randomized Computation

- Algorithms and automata are assumed to have the ability to **sample** from a distribution [dLMSS1956,R1963].
- This is a **powerful tool** when solving computational problems.
- **Example:**

```
Input: n > 3, an odd integer to be tested for primality;
Input: k, a parameter that determines the accuracy of the test
Output: composite if n is composite, otherwise probably prime
write n − 1 as 2^s·d with d odd by factoring powers of 2 from n − 1
WitnessLoop: repeat k times:
   pick a random integer a in the range [2, n − 2]
   x ← a^d mod n
   if x = 1 or x = n − 1 then do next WitnessLoop
   repeat s − 1 times:
      x ← x^2 mod n
      if x = 1 then return composite
      if x = n − 1 then do next WitnessLoop
   return composite
return probably prime
```

Randomized Computation

- Algorithms and automata are assumed to have the ability to **sample** from a distribution [dLMSS1956,R1963].
- This is a **powerful tool** when solving computational problems.
- **Example:**

```
Input: n > 3, an odd integer to be tested for primality;
Input: k, a parameter that determines the accuracy of the test
Output: composite if n is composite, otherwise probably prime
write n − 1 as 2^s·d with d odd by factoring powers of 2 from n − 1
WitnessLoop: repeat k times:
   pick a random integer a in the range [2, n − 2]
   x ← a^d mod n
   if x = 1 or x = n − 1 then do next WitnessLoop
   repeat s − 1 times:
      x ← x^2 mod n
      if x = 1 then return composite
      if x = n − 1 then do next WitnessLoop
   return composite
return probably prime
```

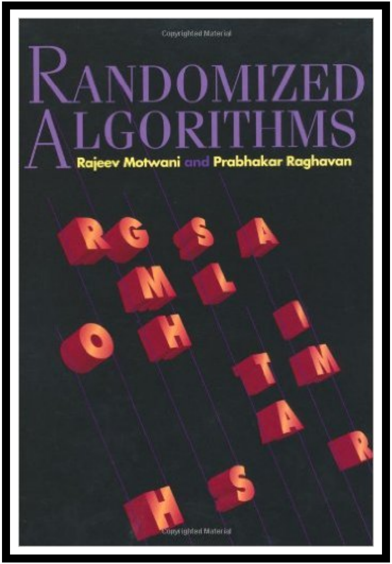
Randomized Computation

- Algorithms and automata are assumed to have the ability to **sample** from a distribution [dLMSS1956,R1963].
- This is a **powerful tool** when solving computational problems.
- **Example:**

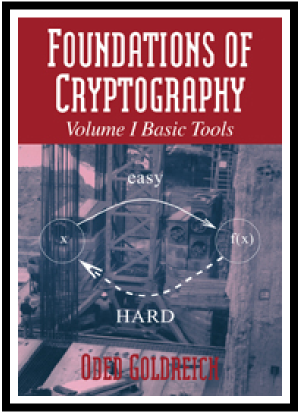
```
Input: n > 3, an odd integer to be tested for primality;
Input: k, a parameter that determines the accuracy of the test
Output: composite if n is composite, otherwise probably prime
write n − 1 as 2^s·d with d odd by factoring powers of 2 from n − 1
WitnessLoop: repeat k times:
   pick a random integer a in the range [2, n − 2]
   x ← a^d mod n
   if x = 1 or x = n − 1 then do next WitnessLoop
   repeat s − 1 times:
      x ← x^2 mod n
      if x = 1 then return composite
      if x = n − 1 then do next WitnessLoop
   return composite
return probably prime
```

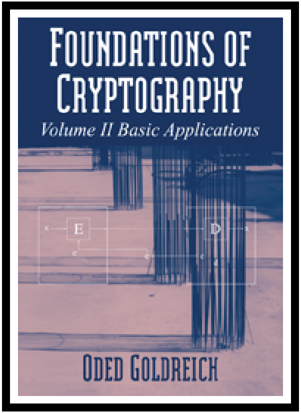
- **Abstractions:**
  - Randomized algorithms;
  - Probabilistic Turing machines.
  - Labelled Markov chains.

Randomized Computation



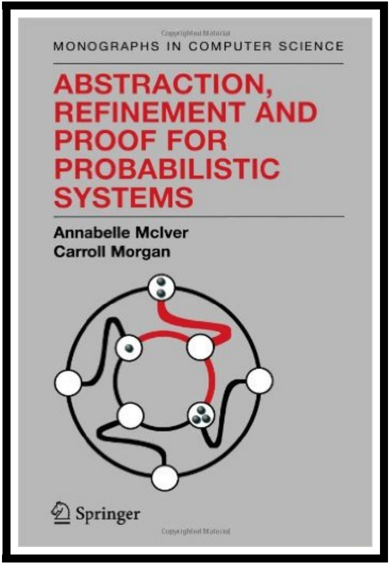
ALGORITHMICS





CRYPTOGRAPHY

Randomized Computation



PROGRAM VERIFICATION

Higher-Order Computation

- Mainly useful in **programming**.

Higher-Order Computation

- Mainly useful in **programming**.
- Functions are **first-class citizens**:
  - They can be passed as *arguments*;
  - They can be obtained as *results*.

Higher-Order Computation

- Mainly useful in **programming**.
- Functions are **first-class citizens**:
  - They can be passed as *arguments*;
  - They can be obtained as *results*.
- **Motivations**:
  - *Modularity*;
  - *Code reuse*;
  - *Conciseness*.

Higher-Order Computation

- Mainly useful in **programming**.
- Functions are **first-class citizens**:
  - They can be passed as *arguments*;
  - They can be obtained as *results*.
- **Motivations**:
  - *Modularity*;
  - *Code reuse*;
  - *Conciseness*.
- **Example**:

```
foldr              :: (a -> b -> b) -> b -> [a] -> b
foldr f acc []      = acc
foldr f acc (x:xs) = f x (foldr f acc xs)
```

Higher-Order Computation

- Mainly useful in **programming**.
- Functions are **first-class citizens**:
  - They can be passed as *arguments*;
  - They can be obtained as *results*.
- **Motivations**:
  - *Modularity*;
  - *Code reuse*;
  - *Conciseness*.
- **Example**:

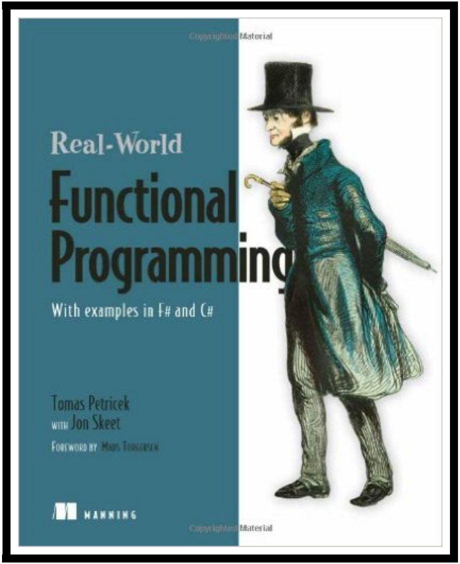
```
foldr              :: (a -> b -> b) -> b -> [a] -> b
foldr f acc []      = acc
foldr f acc (x:xs)  = f x (foldr f acc xs)
```

Higher-Order Computation

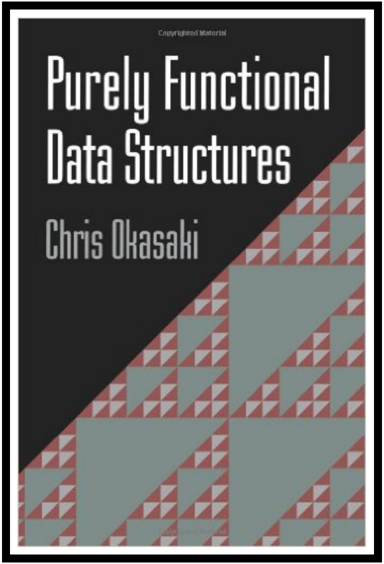
- Mainly useful in **programming**.
- Functions are **first-class citizens**:
  - They can be passed as *arguments*;
  - They can be obtained as *results*.
- **Motivations**:
  - *Modularity*;
  - *Code reuse*;
  - *Conciseness*.
- **Example**:

```
foldr              :: (a -> b -> b) -> b -> [a] -> b
foldr f acc []      = acc
foldr f acc (x:xs) = f x (foldr f acc xs)
```

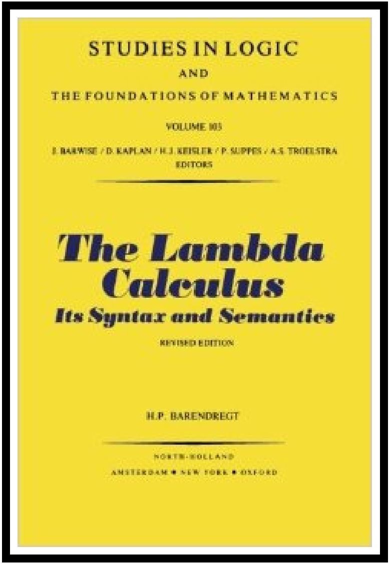
- **Models**:
  - λ-calculus



FUNCTIONAL PROGRAMMING



FUNCTIONAL
DATA STRUCTURES



λ-CALCULUS

Does it Make Sense?

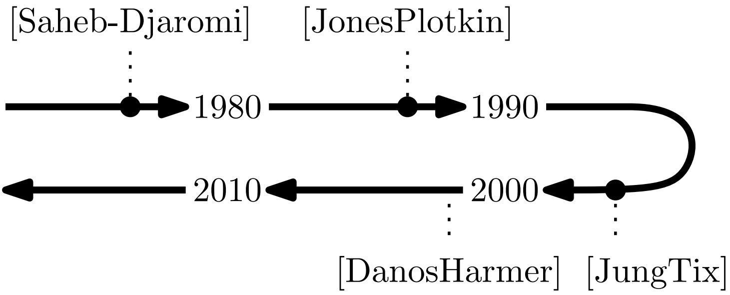
Does it Make Sense?

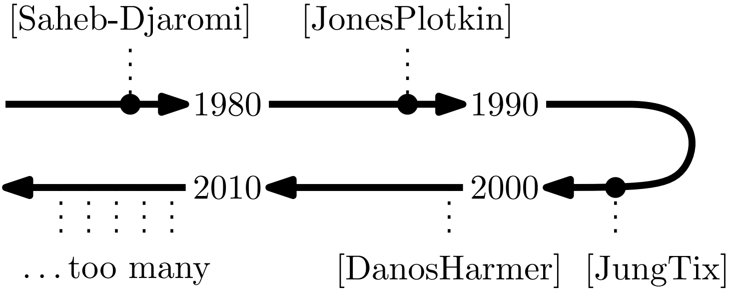
What Kind of Metatheory Does it Have?

Does it Make Sense?

What Kind of Metatheory Does it Have?

Applications?


[Saheb-Djaromi]
[JonesPlotkin]
1980
1990
2010
2000
[DanosHarmer]
[JungTix]


[Saheb-Djaromi]
[JonesPlotkin]
1980
1990
2010
2000
...too many
[DanosHarmer]
[JungTix]

Outline

Part I

Relational Reasoning

Syntax and Operational Semantics of $\Lambda_\oplus$

- **Terms**: $M ::= x \;\Big|\; \lambda x.M \;\Big|\; MM \;\Big|\; M \oplus M;$

Syntax and Operational Semantics of $\Lambda_{\oplus}$

- **Terms**: $M ::= x \;\Big|\; \lambda x.M \;\Big|\; MM \;\Big|\; M \oplus M$;
- **Values**: $V ::= \lambda x.M$;

Syntax and Operational Semantics of $\Lambda_\oplus$

- **Terms**: $M ::= x \;\big|\; \lambda x.M \;\big|\; MM \;\big|\; M \oplus M$;
- **Values**: $V ::= \lambda x.M$;
- **Value Distributions**:

$$V \xrightarrow{\mathcal{D}} \mathcal{D}(V) \in \mathbb{R}_{[0,1]} \qquad\qquad \sum \mathcal{D} = \sum_V \mathcal{D}(V) \leq 1.$$

Syntax and Operational Semantics of $\Lambda_{\oplus}$

- **Terms**: $M ::= x \;\Big|\; \lambda x.M \;\Big|\; MM \;\Big|\; M \oplus M$;
- **Values**: $V ::= \lambda x.M$;
- **Value Distributions**:

$$V \xrightarrow{\mathcal{D}} \mathcal{D}(V) \in \mathbb{R}_{[0,1]} \qquad \sum \mathcal{D} = \sum_{V} \mathcal{D}(V) \leq 1.$$

- **Semantics**: $[\![M]\!] = \sup_{M \Downarrow \mathcal{D}} \mathcal{D}$;

Syntax and Operational Semantics of $\Lambda_\oplus$

$$\frac{}{M \Downarrow \emptyset} \qquad \frac{}{V \Downarrow \{V^1\}} \qquad \frac{M \Downarrow \mathcal{D} \qquad N \Downarrow \mathcal{E}}{M \oplus N \Downarrow \frac{1}{2}\mathcal{D} + \frac{1}{2}\mathcal{E}}$$

$$\frac{M \Downarrow \mathscr{K} \qquad \{P[N/x] \Downarrow \mathscr{E}_P\}_{\lambda x.P \in \mathsf{S}\mathscr{K}}}{MN \Downarrow \sum_{\lambda x.P \in \mathsf{S}\mathscr{K}} \mathscr{K}(\lambda x.P) \cdot \mathscr{E}_P}$$

$$V \mapsto \mathcal{D}(V) \in \mathbb{R}_{[0,1]} \qquad \sum \mathcal{D} = \sum_V \mathcal{D}(V) \leq 1.$$

- **Semantics**: $[\![M]\!] = \sup_{M \Downarrow \mathcal{D}} \mathcal{D}$;

Syntax and Operational Semantics of $\Lambda_{\oplus}$

- **Terms**: $M ::= x \mid \lambda x.M \mid MM \mid M \oplus M$;
- **Values**: $V ::= \lambda x.M$;
- **Value Distributions**:

$$V \xrightarrow{\mathcal{D}} \mathcal{D}(V) \in \mathbb{R}_{[0,1]} \qquad\qquad \sum \mathcal{D} = \sum_{V} \mathcal{D}(V) \leq 1.$$

- **Semantics**: $[\![M]\!] = \sup_{M \Downarrow \mathcal{D}} \mathcal{D}$;
- **Context Equivalence**: $M \equiv N$ iff for every context C it holds that $\sum [\![C[M]]\!] = \sum [\![C[N]]\!]$.

Syntax and Operational Semantics of $\Lambda_\oplus$

- **Terms**: $M ::= x \mid \lambda x.M \mid MM \mid M \oplus M$;
- **Values**: $V ::= \lambda x.M$;
- **Value Distributions**:

$C ::= [\cdot] \mid \lambda x.C \mid CM \mid MC \mid C \oplus M \mid M \oplus C$

$V) \leq 1.$

- **Semantics**: $[\![M]\!] = \sup_{M \Downarrow \mathcal{D}} \mathcal{D}$;
- **Context Equivalence**: $M \equiv N$ iff for every context C it holds that $\sum [\![C[M]]\!] = \sum [\![C[N]]\!]$.

Syntax and Operational Semantics of $\Lambda_\oplus$

- **Terms**: $M ::= x \;\Big|\; \lambda x.M \;\Big|\; MM \;\Big|\; M \oplus M$;
- **Values**: $V ::= \lambda x.M$;
- **Value Distributions**:

$$V \xrightarrow{\mathcal{D}} \mathcal{D}(V) \in \mathbb{R}_{[0,1]} \qquad\qquad \sum \mathcal{D} = \sum_V \mathcal{D}(V) \leq 1.$$

- **Semantics**: $[\![M]\!] = \sup_{M \Downarrow \mathcal{D}} \mathcal{D}$;
- **Context Equivalence**: $M \equiv N$ iff for every context C it holds that $\sum [\![C[M]]\!] = \sum [\![C[N]]\!]$.
- **Context Distance**:
  $\delta^C(M, N) = \sup_C |\sum [\![C[M]]\!] - \sum [\![C[N]]\!]|$.

Examples

$$I \oplus \Omega \qquad \text{vs.} \qquad I$$

Examples

$\lambda x.x$

$$I \oplus \Omega \qquad \text{vs.} \qquad I$$

Examples

$\Delta\Delta = (\lambda x.xx)(\lambda x.xx)$

$$I \oplus \Omega \qquad \text{vs.} \qquad I$$

Exam

Not Context Equivalent: $C = [\cdot]$.

Context Distance? Consider $C_n = (\lambda x.\ \underbrace{x \ldots x}_{n \text{ times}})[\cdot]$.

$$I \oplus \Omega \qquad \text{vs.} \qquad I$$

Examples

$$I \oplus \Omega \qquad \text{vs.} \qquad I$$

$$I \oplus \Omega \qquad \text{vs.} \qquad \Omega$$

Examples

Not Context Equivalent: $C = [\cdot]$.
Context Distance? Cannot Easily Amplify.

$$I \oplus \Omega \qquad \text{vs.} \qquad \Omega$$

Examples

$$I \oplus \Omega \qquad \text{vs.} \qquad I$$

$$I \oplus \Omega \qquad \text{vs.} \qquad \Omega$$

$$(\lambda x.I) \oplus (\lambda x.\Omega) \qquad \text{vs.} \qquad \lambda x.I \oplus \Omega$$

Examples

$$I \oplus \Omega \qquad \text{vs.} \qquad I$$

Not Context Equivalent in CBV: $C = (\lambda x.x(xI))[\cdot]$
Apparently Context Equivalent in CBN.

$$(\lambda x.I) \oplus (\lambda x.\Omega) \qquad \text{vs.} \qquad \lambda x.I \oplus \Omega$$

Examples

$$I \oplus \Omega \qquad \text{vs.} \qquad I$$

$$I \oplus \Omega \qquad \text{vs.} \qquad \Omega$$

$$(\lambda x.I) \oplus (\lambda x.\Omega) \qquad \text{vs.} \qquad \lambda x.I \oplus \Omega$$

$$Y_1 \qquad \text{vs.} \qquad Y_2$$

Examples

$$I \oplus \Omega \qquad \text{vs.} \qquad I$$

$$I \oplus \Omega \qquad \text{vs.} \qquad \Omega$$

$$Y_1 M \to^* M(Y_2 M) \oplus M(Y_3 M)$$
$$Y_2 M \to^* M(Y_1 M) \oplus M(Y_3 M)$$
$$Y_3 M \to^* M(Y_1 M) \oplus M(Y_2 M)$$

$$Y_1 \qquad \text{vs.} \qquad Y_2$$

Terms

A Labelled Markov Chain for $\Lambda_{\oplus}$

Terms **Values**

A Labelled Markov Chain for $\Lambda_{\oplus}$

Terms	**Values**
M	

A Labelled Markov Chain for $\Lambda_\oplus$

Terms **Values**

$$\begin{tikzcd}
M \arrow[r, "{eval,\ \llbracket M \rrbracket(V)}"] \arrow[dr, "{eval,\ \llbracket M \rrbracket(W)}"] \arrow[ddr, "{eval,\ \llbracket M \rrbracket(Z)}"'] & V \\
& W \\
& Z \\
& \vdots
\end{tikzcd}$$

A Labelled Markov Chain for $\Lambda_{\oplus}$

Terms	**Values**
	$\lambda x.N$

A Labelled Markov Chain for $\Lambda_{\oplus}$

Terms		**Values**
$N\{W/x\}$	$\xleftarrow{W,\ 1}$	$\lambda x.N$

Probabilistic Applicative Bisimulation

$$\lambda x.M \;\; \mathcal{R} \;\; \lambda x.N$$

Probabilistic Applicative Bisimulation

$$\lambda x.M \;\; \mathcal{R} \;\; \lambda x.N \qquad\qquad M\{L/x\}$$

Probabilistic Applicative Bisimulation

$$\begin{tikzcd}
\lambda x.M \arrow[rrr, bend left, "L"] & \mathcal{R} & \lambda x.N \arrow[rr, bend right, "L"'] & M\{L/x\} & N\{L/x\}
\end{tikzcd}$$

Probabilistic Applicative Bisimulation

$$
\begin{tikzcd}
\lambda x.M \arrow[r, phantom, "\mathcal{R}"] \arrow[rr, bend left, "L"] & \lambda x.N \arrow[rr, bend right, "L"'] & M\{L/x\} \arrow[r, phantom, "\mathcal{R}"] & N\{L/x\}
\end{tikzcd}
$$

Probabilistic Applicative Bisimulation

$$
\begin{tikzcd}
\lambda x.M \arrow[rrr, bend left, "L"] & \mathcal{R} & \lambda x.N \arrow[rrr, bend right, "L"'] & M\{L/x\} & \mathcal{R} & N\{L/x\}
\end{tikzcd}
$$

$$M \quad \mathcal{R} \quad N$$

Probabilistic Applicative Bisimulation

$$\begin{tikzcd}
\lambda x.M \arrow[rrr, bend left, "L"] & \mathcal{R} & \lambda x.N \arrow[rrr, bend right, "L"'] & M\{L/x\} & \mathcal{R} & N\{L/x\}
\end{tikzcd}$$

$$\begin{tikzcd}
 & & & \llbracket M \rrbracket \\
M \arrow[urrr, bend left, "eval"] & \mathcal{R} & N &
\end{tikzcd}$$

Probabilistic Applicative Bisimulation

$$
\begin{tikzcd}
\lambda x.M \arrow[rrr, bend left, "L"] & \mathcal{R} & \lambda x.N \arrow[rrr, bend right, "L"'] & M\{L/x\} & \mathcal{R} & N\{L/x\}
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & & & \llbracket M \rrbracket \\
M \arrow[urrr, bend left, "eval"] & \mathcal{R} & N \arrow[dr, bend right, "eval"'] & \\
 & & & \llbracket N \rrbracket
\end{tikzcd}
$$

Probabilistic Applicative Bisimulation

$$\begin{tikzcd}
\lambda x.M \arrow[rrr, bend left, "L"] & \mathcal{R} & \lambda x.N \arrow[rrr, bend right, "L"'] & M\{L/x\} & \mathcal{R} & N\{L/x\}
\end{tikzcd}$$

$$\begin{tikzcd}
 & & & \llbracket M \rrbracket \arrow[dr, bend left] & \\
M \arrow[urrr, bend left, "eval"] & \mathcal{R} & N \arrow[dr, bend right, "eval"'] & & \llbracket M \rrbracket(E) \\
 & & & \llbracket N \rrbracket &
\end{tikzcd}$$

Probabilistic Applicative Bisimulation

$$\begin{tikzcd}
\lambda x.M \arrow[rrrr, bend left, "L"] & \mathcal{R} & \lambda x.N \arrow[rrrr, bend right, "L"'] & & M\{L/x\} & \mathcal{R} & N\{L/x\}
\end{tikzcd}$$

$$\begin{tikzcd}
 & & & \llbracket M \rrbracket \arrow[dr] & & \\
M \arrow[urrr, bend left, "eval"] & \mathcal{R} & N \arrow[dr, bend right, "eval"'] & & \llbracket M \rrbracket(E) & \llbracket N \rrbracket(E) \\
 & & & \llbracket N \rrbracket \arrow[urr, bend right] & &
\end{tikzcd}$$

Probabilistic Applicative Bisimulation

$$
\begin{tikzcd}
\lambda x.M \arrow[rrr, bend left, "L"] & \mathcal{R} & \lambda x.N \arrow[rrr, bend right, "L"'] & M\{L/x\} & \mathcal{R} & N\{L/x\}
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & & \llbracket M \rrbracket \arrow[dr, bend left] & \\
M \arrow[urr, bend left, "eval"] & \mathcal{R} & N \arrow[d, bend right, "eval"'] & \llbracket M \rrbracket(E) = \llbracket N \rrbracket(E) \\
 & & \llbracket N \rrbracket \arrow[ur, bend right] &
\end{tikzcd}
$$

Applicative Bisimilarity vs. Context Equivalence

- **Bisimilarity**: the union $\sim$ of all bisimulation relations.
- Is it that $\sim$ is included in $\equiv$? How to prove it?
- Natural strategy: is $\sim$ a congruence?
  - If this is the case:

$$
\begin{aligned}
M \sim N \implies C[M] \sim C[N] &\implies \sum [\![C[M]]\!] = \sum [\![C[N]]\!] \\
&\implies M \equiv N.
\end{aligned}
$$

  - This is a necessary sanity check anyway.
- The naïve proof by induction **fails**, due to application: from $M \sim N$, one cannot directly conclude that $LM \sim LN$.

Howe's Technique

$\mathcal{R}$ $\mathcal{R}^H$

$$\begin{tikzcd} \mathcal{R} \arrow[rr, bend left, "\subseteq"] & & \mathcal{R}^H \end{tikzcd}$$

Howe's Technique

$\mathcal{R}^H$ is a
Congruence
whenever $\mathcal{R}$ is
an equivalence

$$\begin{tikzcd} \mathcal{R} \arrow[r, bend left, "\subseteq"] & \mathcal{R}^H \end{tikzcd}$$

Howe's Technique

$\sim^H$ is a
Congruence

$$\begin{tikzcd} \sim \arrow[rr, bend left, "\subseteq"] & & \sim^H \end{tikzcd}$$

Howe's Technique

$\sim^H$ is a **Congruence**

$$\begin{tikzcd} \sim \arrow[r, bend left, "\subseteq"] & \sim^H \arrow[l, bend left, "\supseteq"] \end{tikzcd}$$

Key Lemma

Our Neighborhood

- Λ, where we observe **convergence**

	$\sim\ \subseteq\ \equiv$	$\equiv\ \subseteq\ \sim$
CBN	✓	✓
CBV	✓	✓

[Abramsky1990, Howe1993]

- $\Lambda_{\oplus}$ with nondeterministic semantics, where we observe **convergence**, in its **may** or **must** flavors.

	$\sim\ \subseteq\ \equiv$	$\equiv\ \subseteq\ \sim$
CBN	✓	×
CBV	✓	×

[Ong1993, Lassen1998]

The Probabilistic Case

- $\Lambda_{\oplus}$ with probabilistic semantics.

	$\sim \subseteq \equiv$	$\equiv \subseteq \sim$
CBN	✓	✗
CBV	✓	✓

The Probabilistic Case

- $\Lambda_{\oplus}$ with probabilistic semantics.

	$\sim \subseteq \equiv$	$\equiv \subseteq \sim$
CBN	✓	×
CBV	✓	✓

- Counterexample for CBN: $(\lambda x.I) \oplus (\lambda x.\Omega) \not\sim \lambda x.I \oplus \Omega$
- **Where** these discrepancies come from?

The Probabilistic Case

- $\Lambda_{\oplus}$ with probabilistic semantics.

	$\sim\ \subseteq\ \equiv$	$\equiv\ \subseteq\ \sim$
CBN	✓	✗
CBV	✓	✓

- Counterexample for CBN: $(\lambda x.I) \oplus (\lambda x.\Omega) \not\sim \lambda x.I \oplus \Omega$
- **Where** these discrepancies come from?
- From **testing**!

The Probabilistic Case

- $\Lambda_{\oplus}$ with probabilistic semantics.

	$\sim \subseteq \equiv$	$\equiv \subseteq \sim$
CBN	✓	✗
CBV	✓	✓

- Counterexample for CBN: $(\lambda x.I) \oplus (\lambda x.\Omega) \not\sim \lambda x.I \oplus \Omega$
- **Where** these discrepancies come from?
- From **testing**!
- Bisimulation can be characterized by testing equivalence as follows:

Calculus	**Testing**
Λ	$T ::= \omega \mid a \cdot T$
$P\Lambda_{\oplus}$	$T ::= \omega \mid a \cdot T \mid \langle T, T \rangle$
$N\Lambda_{\oplus}$	$T ::= \omega \mid a \cdot T \mid \wedge_{i \in I} T_i \mid \ldots$

The Probabilistic Case

- $\Lambda_{\oplus}$ with probabilistic semantics.

	$\precsim \subseteq \leq$	$\leq \subseteq \precsim$
CBN	✓	×
CBV	✓	×

The Probabilistic Case

- $\Lambda_{\oplus}$ with probabilistic semantics.

	$\precsim \subseteq \leq$	$\leq \subseteq \precsim$
CBN	✓	×
CBV	✓	×

- Probabilistic simulation can be characterized by testing as follows:

$$T ::= \omega \;\Big|\; a \cdot T \;\Big|\; \langle T, T \rangle \;\Big|\; T \vee T$$

The Probabilistic Case

- $\Lambda_{\oplus}$ with probabilistic semantics.

	$\precsim \subseteq \leq$	$\leq \subseteq \precsim$
CBN	✓	×
CBV	✓	×

- Probabilistic simulation can be characterized by testing as follows:

$$T ::= \omega \mid a \cdot T \mid \langle T, T \rangle \mid T \vee T$$

- Full abstraction can be recovered if endowing $\Lambda_{\oplus}$ with parallel disjunction [CDLSV2015].

	$\precsim \subseteq \leq$	$\leq \subseteq \precsim$
CBN	✓	×
CBV	✓	✓

Context Distance: the Affine Case [CDL2015]

- Let us consider a simple fragment of $\Lambda_{\oplus}$, first.

Context Distance: the Affine Case [CDL2015]

- Let us consider a simple fragment of $\Lambda_\oplus$, first.
- **Preterms**: $M, N ::= x \;\big|\; \lambda x.M \;\big|\; MM \;\big|\; M \oplus M \;\big|\; \Omega$;

Context Distance: the Affine Case [CDL2015]

- Let us consider a simple fragment of $\Lambda_{\oplus}$, first.
- **Preterms**: $M, N ::= x \;\big|\; \lambda x.M \;\big|\; MM \;\big|\; M \oplus M \;\big|\; \Omega$;
- **Terms**: any preterm M such that $\Gamma \vdash M$.

Con

$$\frac{}{\Gamma, x \vdash x} \qquad \frac{x, \Gamma \vdash M}{\Gamma \vdash \lambda x.M} \qquad \frac{\Gamma \vdash M \qquad \Delta \vdash N}{\Gamma, \Delta \vdash MN} \qquad \frac{\Gamma \vdash M \qquad \Gamma \vdash N}{\Gamma \vdash M \oplus N}$$

- Let us consider a simple fragment of $\Lambda_{\oplus}$, first.
- **Preterms**: $M, N ::= x \;\big|\; \lambda x.M \;\big|\; MM \;\big|\; M \oplus M \;\big|\; \Omega$;
- **Terms**: any preterm M such that $\Gamma \vdash M$.

Context Distance: the Affine Case [CDL2015]

- Let us consider a simple fragment of $\Lambda_{\oplus}$, first.
- **Preterms**: $M, N ::= x \;\Big|\; \lambda x.M \;\Big|\; MM \;\Big|\; M \oplus M \;\Big|\; \Omega$;
- **Terms**: any preterm M such that $\Gamma \vdash M$.
- **Behavioural Distance** δ^b.
  - The metric analogue to bisimilarity.

Context Distance: the Affine Case [CDL2015]

- Let us consider a simple fragment of $\Lambda_\oplus$, first.
- **Preterms**: $M, N ::= x \;\big|\; \lambda x.M \;\big|\; MM \;\big|\; M \oplus M \;\big|\; \Omega$;
- **Terms**: any preterm M such that $\Gamma \vdash M$.
- **Behavioural Distance** δ^b.
  - The metric analogue to bisimilarity.
- **Trace Distance** δ^t.
  - The maximum distance induced by traces, i.e., sequences of actions: $\delta^t(M, N) = \sup_{\mathsf{T}} |Pr(M, \mathsf{T}) - Pr(N, \mathsf{T})|$.

Context Distance: the Affine Case [CDL2015]

- Let us consider a simple fragment of $\Lambda_{\oplus}$, first.
- **Preterms**: $M, N ::= x \mid \lambda x.M \mid MM \mid M \oplus M \mid \Omega;$
- **Terms**: any preterm M such that $\Gamma \vdash M$.
- **Behavioural Distance** δ^b.
  - The metric analogue to bisimilarity.
- **Trace Distance** δ^t.
  - The maximum distance induced by traces, i.e., sequences of actions: $\delta^t(M, N) = \sup_{\mathsf{T}} |Pr(M, \mathsf{T}) - Pr(N, \mathsf{T})|$.
- **Soundness and Completeness Results:**

$\delta^b \leq \delta^c$	$\delta^c \leq \delta^b$	$\delta^t \leq \delta^c$	$\delta^c \leq \delta^t$
✓	×	✓	✓

Context Distance: the Affine Case [CDL2015]

- Let us consider a simple fragment of $\Lambda_{\oplus}$, first.
- **Preterms**: $M, N ::= x \;\big|\; \lambda x.M \;\big|\; MM \;\big|\; M \oplus M \;\big|\; \Omega;$
- **Terms**: any preterm M such that $\Gamma \vdash M$.
- **Behavioural Distance** δ^b.
  - The metric analogue to bisimilarity.
- **Trace Distance** δ^t.
  - The maximum distance induced by traces, i.e., sequences of actions: $\delta^t(M, N) = \sup_{\mathsf{T}} |Pr(M, \mathsf{T}) - Pr(N, \mathsf{T})|$.
- **Soundness and Completeness Results:**

$\delta^b \leq \delta^c$	$\delta^c \leq \delta^b$	$\delta^t \leq \delta^c$	$\delta^c \leq \delta^t$
✓	×	✓	✓

- **Example**: $\delta^t(I, I \oplus \Omega) = \delta^t(I \oplus \Omega, \Omega) = \frac{1}{2}$.

Context Distance: the General Case [CDL2016]

- The LMC we have have worked so far with induces **unsound** metrics for $\Lambda_{\oplus}\ldots$

Context Distance: the General Case [CDL2016]

- The LMC we have have worked so far with induces **unsound** metrics for $\Lambda_{\oplus}$...
- ...because it **does not** adequately model copying.

Context Distance: the General Case [CDL2016]

- The LMC we have have worked so far with induces **unsound** metrics for $\Lambda_{\oplus}\ldots$
- $\ldots$ because it **does not** adequately model copying.
- **A Tuple LMC**.
  - **Preterms**:
    $M ::= x \;\big|\; \lambda x.M \;\big|\; \lambda !x.M \;\big|\; MM \;\big|\; M \oplus M \;\big|\; !M$
  - **Terms**: any preterm M such that $\Gamma \vdash M$.
  - **States**: *sequences* of terms, rather than terms.
  - **Actions** not only model parameter passing, but also *copying* of terms.

Context Distance: the General Case [CDL2016]

$$\frac{}{!\Gamma, x \vdash x} \qquad \frac{}{!\Gamma, !x \vdash x} \qquad \frac{x, \Gamma \vdash M}{\Gamma \vdash \lambda x.M} \qquad \frac{!x, \Gamma \vdash M}{\Gamma \vdash \lambda !x.M}$$

$$\frac{!\Gamma \vdash M}{!\Gamma \vdash !M} \qquad \frac{\Gamma, !\Theta \vdash M \qquad \Delta, !\Theta \vdash N}{\Gamma, \Delta, \Theta \vdash MN} \qquad \frac{\Gamma \vdash M \qquad \Gamma \vdash N}{\Gamma \vdash M \oplus N}$$

- T...
  u...
- ...
- A...
  - **Preterms**:
    $M ::= x \;\big|\; \lambda x.M \;\big|\; \lambda !x.M \;\big|\; MM \;\big|\; M \oplus M \;\big|\; !M$
  - **Terms**: any preterm M such that $\Gamma \vdash M$.
  - **States**: *sequences* of terms, rather than terms.
  - **Actions** not only model parameter passing, but also *copying* of terms.

Context Distance: the General Case [CDL2016]

- The LMC we have have worked so far with induces **unsound** metrics for $\Lambda_{\oplus}$...
- ...because it **does not** adequately model copying.
- **A Tuple LMC**.
  - **Preterms**:
    $M ::= x \;\Big|\; \lambda x.M \;\Big|\; \lambda !x.M \;\Big|\; MM \;\Big|\; M \oplus M \;\Big|\; !M$
  - **Terms**: any preterm M such that $\Gamma \vdash M$.
  - **States**: *sequences* of terms, rather than terms.
  - **Actions** not only model parameter passing, but also *copying* of terms.
- **Soundness and Completeness Results:**

$\delta^t \leq \delta^c$	$\delta^c \leq \delta^t$
✓	✓

Context Distance: the General Case [CDL2016]

- The LMC we have have worked so far with induces **unsound** metrics for $\Lambda_{\oplus}$...
- ... because it **does not** adequately model copying.
- **A Tuple LMC**.
  - **Preterms**:
    $M ::= x \;\Big|\; \lambda x.M \;\Big|\; \lambda !x.M \;\Big|\; MM \;\Big|\; M \oplus M \;\Big|\; !M$
  - **Terms**: any preterm M such that $\Gamma \vdash M$.
  - **States**: *sequences* of terms, rather than terms.
  - **Actions** not only model parameter passing, but also *copying* of terms.
- **Soundness and Completeness Results:**

$\delta^t \leq \delta^c$	$\delta^c \leq \delta^t$
✓	✓

- **Examples**: $\delta^t(!(I \oplus \Omega), !\Omega) = \frac{1}{2}$ $\qquad \delta^t(!(I \oplus \Omega), !I) = 1.$

Context Distance: the General Case [CDL2016]

- The LMC we have have worked so far with induces **unsound** metrics for $\Lambda_{\oplus}\ldots$
- $\ldots$ because it **does not** adequately model copying.
- **A Tuple LMC**.
  - **Preterms**:
    $M ::= x \;\big|\; \lambda x.M \;\big|\; \lambda !x.M \;\big|\; MM \;\big|\; M \oplus M \;\big|\; !M$
  - **Terms**: any preterm M such that $\Gamma \vdash M$.
  - **States**: *sequences* of terms, rather than terms.
  - **Actions** not only model parameter passing, but also *copying* of terms.
- **Soundness and Completeness Results:**

$\delta^t \leq \delta^c$	$\delta^c \leq \delta^t$
✓	✓

- **Examples**: $\delta^t(!(I \oplus \Omega), !\Omega) = \frac{1}{2}$ $\qquad \delta^t(!(I \oplus \Omega), !I) = 1.$
- **Trivialisation**: the context distance *collapses* to an equivalence in *strongly normalising* fragments or in presence of *parellel disjuction*.

Context Distance: the General Case [CDL2016]

- The LMC we have have worked so far with induces **unsound** metrics for $\Lambda_{\oplus}$...
- ... because it **does not** adequately model copying.
- **A Tuple LMC**.
  - **Preterms**:
    $M ::= x \;\Big|\; \lambda x.M \;\Big|\; \lambda !x.M \;\Big|\; MM \;\Big|\; M \oplus M \;\Big|\; !M$
  - **Terms**: any preterm M such that $\Gamma \vdash M$.
  - **States**: *sequences* of terms, rather than terms.
  - **Actions** not only model parameter passing, but also *copying* of terms.
- **Soundness and Comple**

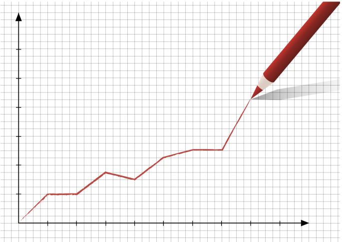
What would a *sensible* notion of distance look like?

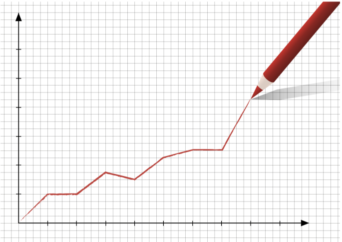
$\delta^t \leq \delta^c$	$\delta^c \leq \delta^t$
✓	✓

- **Examples**: $\delta^t(!(I \oplus \Omega), !\Omega) = \frac{1}{2}$ $\qquad \delta^t(!(I \oplus \Omega), !I) = 1.$
- **Trivialisation**: the context distance *collapses* to an equivalence in *strongly normalising* fragments or in presence of *parellel disjuction*.

Part II

Bayesian Functional Programming




normalize(

  let $x =$ sample($bern\left(\frac{5}{7}\right)$) in

  let $r =$ if x *then* 10 *else* 3 in

    observe 4 from $poisson(r)$;

    return(x))

normalize(

let $x =$ sample($bern\left(\frac{5}{7}\right)$) in

let $r =$ if x *then* 10 *else* 3 in

observe 4 from $poisson(r)$;

return(x))

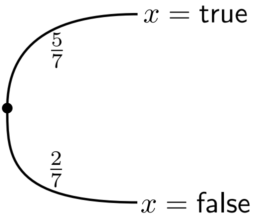
normalize(

let $x =$ sample$(bern\left(\frac{5}{7}\right))$ in

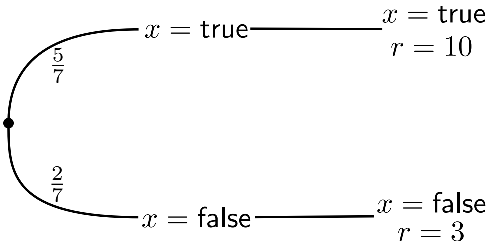
let $r =$ if x *then* 10 *else* 3 in

observe 4 from $poisson(r)$;

return(x))



1. normalize(
2. let $x =$ sample($bern\left(\frac{5}{7}\right)$) in
3. let $r =$ if x *then* 10 *else* 3 in
4. observe 4 from $poisson(r)$;
5. return(x))



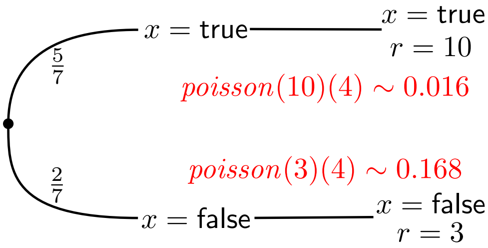
normalize(

  let $x =$ sample($bern\left(\frac{5}{7}\right)$) in

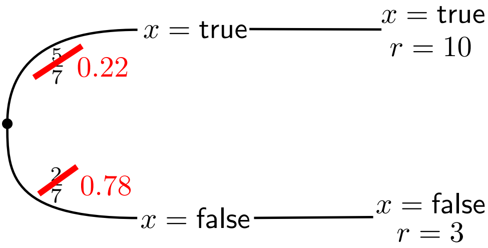
  let $r =$ if x *then* 10 *else* 3 in

    observe 4 from $poisson(r)$;

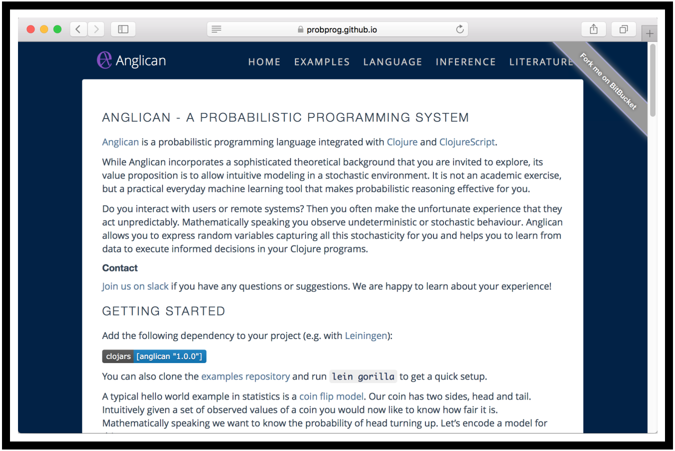
    return(x))



normalize(
    let $x =$ sample($bern\left(\frac{5}{7}\right)$) in
    let $r =$ if x *then* 10 *else* 3 in
        observe 4 from $poisson(r)$;
        return(x))

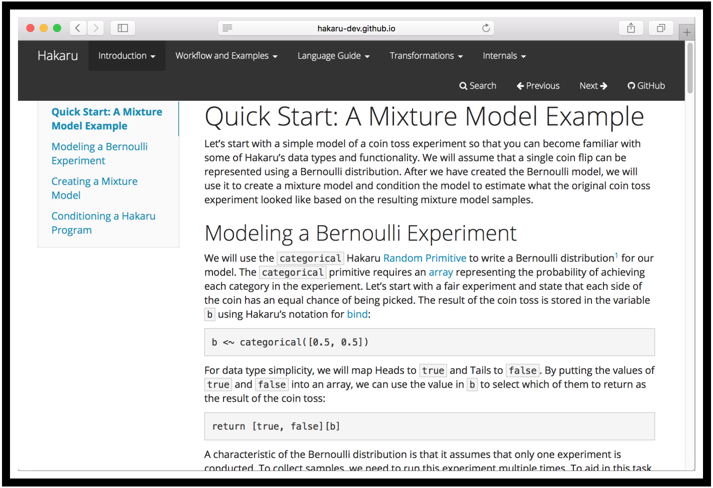


Bayesian Functional Programming



ANGLICAN

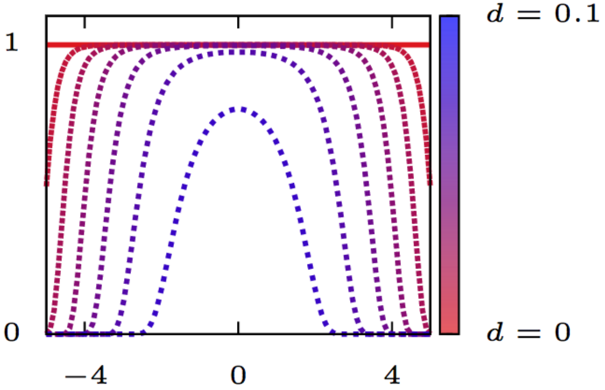
Bayesian Functional Programming



HAKARU

normalize(
    let $x =$ sample($gauss\,(0,1)$) in
        observe d from $exp(1/f(x))$;
        return(x))

1. normalize(
2. let $x =$ sample($gauss\ (0, 1)$) in
4. observe d from $exp(1/f(x))$;
5. return(x))



Bayesian Programming: Semantics

- Giving semantics to programming languages like Anglican or Hakaru is nontrivial:
  - Real numbers;
  - Sampling from **continuous** distributions;
  - **Conditioning**.

Bayesian Programming: Semantics

- Giving semantics to programming languages like Anglican or Hakaru is nontrivial:
  - Real numbers;
  - Sampling from **continuous** distributions;
  - **Conditioning**.
- **Key ingredients**:
  - In $M \Downarrow \mathscr{D}$, we need $\mathscr{D}$ to be a *measure*, because the set of term is not countable anymore.

Bayesian Programming: Semantics

- Giving semantics to programming languages like Anglican or Hakaru is nontrivial:
  - Real numbers;
  - Sampling from **continuous** distributions;
  - **Conditioning**.
- **Key ingredients**:
  - In $M \Downarrow \mathscr{D}$, we need $\mathscr{D}$ to be a *measure*, because the set of term is not countable anymore.
    - Terms must thus be equipped with the structure of a measurable space.

Bayesian Programming: Semantics

- Giving semantics to programming languages like Anglican or Hakaru is nontrivial:
  - Real numbers;
  - Sampling from **continuous** distributions;
  - **Conditioning**.
- **Key ingredients**:
  - In $M \Downarrow \mathscr{D}$, we need $\mathscr{D}$ to be a *measure*, because the set of term is not countable anymore.
    - Terms must thus be equipped with the structure of a measurable space.
  - From

$$\frac{M \Downarrow \mathscr{K} \qquad \{P[N/x] \Downarrow \mathscr{E}_P\}_{\lambda x.P \in \mathsf{S}\mathscr{K}}}{MN \Downarrow \sum_{\lambda x.P \in \mathsf{S}\mathscr{K}} \mathscr{K}(\lambda x.P) \cdot \mathscr{E}_P}$$

  we go to

$$\frac{M \Downarrow \mathscr{K} \qquad \{P[N/x] \Downarrow \mathscr{E}_P\}_{\lambda x.P \in \mathsf{S}\mathscr{K}}}{MN \Downarrow \int \mathscr{E}_P \cdot d\mathscr{K}(\lambda x.P)}$$

Bayesian Programming: Semantics

- Giving semantics to programming languages like Anglican or Hakaru is nontrivial:
  - Real numbers;
  - Sampling from **continuous** distributions;
  - **Conditioning**.
- **Key ingredients**:
  - In $M \Downarrow$ ... use the set of term is ...
    - Te... ucture of a me...
  - From

$$\frac{\qquad \in \mathsf{S}\mathscr{K}}{\qquad P}$$

we go to

$$\frac{M \Downarrow \mathscr{K} \qquad \{P[N/x] \Downarrow \mathscr{E}_P\}_{\lambda x.P \in \mathsf{S}\mathscr{K}}}{MN \Downarrow \int \mathscr{E}_P \cdot d\mathscr{K}(\lambda x.P)}$$

- This Lebesgue integral does not necessarily exist.
- We must ensure that $\Downarrow$ gives rise to a *stochastic kernel*.
- In presence of conditioning, we need even more.

Part III

Termination

The Landscape: *Type* Theory

Simple Types

$$\tau ::= \iota \quad \Big| \quad \tau \to \tau$$

The Landscape: *Type* Theory

Simple Types

$$\tau ::= \iota \quad | \quad \tau \rightarrow \tau$$

- Sound for termination, in absence of recursion.
- Poor expressive power.
- Intuitionistic Logic.

The Landscape: *Type* Theory

Simple Types

$$\tau ::= \iota \;\big|\; \tau \to \tau$$

Polymorphic Types

$$\tau ::= \cdots \;\big|\; \alpha \;\big|\; \forall \alpha . \tau$$

The Landscape: *Type* Theory

- Second-order Logic.
- Very expressive, extensionally.
- Still poor, intensionally.

ypes

$\tau \to \tau$

Polymorphic Types

$$\tau \ ::= \cdots \ \big| \ \alpha \ \big| \ \forall \alpha . \tau$$

The Landscape: *Type* Theory

Simple Types

$$\tau ::= \iota \mid \tau \to \tau$$

Polymorphic Types

$$\tau ::= \cdots \mid \alpha \mid \forall \alpha . \tau$$

Intersection Types

$$\tau ::= \cdots \mid \tau \wedge \tau$$

The Landscape: *Type* Theory

- Motivated by Semantics.
- *Complete* for termination.
- Type inference is undecidable.

Polymorphic Types

$$\tau \ ::= \cdots \ \big| \ \alpha \ \big| \ \forall\alpha.\tau$$

Intersection Types

$$\tau \ ::= \cdots \ \big| \ \tau \wedge \tau$$

The Landscape: *Type* Theory

Simple Types

$\tau ::= \iota \mid \tau \to \tau$

Polymorphic Types

$\tau ::= \cdots \mid \alpha \mid \forall \alpha . \tau$

Intersection Types

$\tau ::= \cdots \mid \tau \wedge \tau$

Sized Types

$\tau ::= \cdots \mid \iota[\xi]$

The Landscape: *Type* Theory

- Reasonably expressive, intensionally.
- Type inference remains decidable

Polymorphic Types

$\tau \ ::= \cdots \ \big| \ \alpha \ \big| \ \forall \alpha . \tau$

Intersection Types

$\tau \ ::= \cdots \ \big| \ \tau \wedge \tau$

Sized Types

$\tau \ ::= \cdots \ \big| \ \iota[\xi]$

The Landscape: *Recursion* Theory

Determinism

$M\overline{s} \to^* N_s$

The Landscape: *Recursion* Theory

Determinism	**Probabilism**
$M\overline{s} \rightarrow^* N_s$	$[\![M\overline{s}]\!] = \mathcal{D}_s$

The Landscape: *Recursion* Theory

$\sum \mathcal{D}_s$ can be smaller than 1.

Determinism	**Probabilism**
$M\overline{s} \rightarrow^* N_s$	$[\![M\overline{s}]\!] = \mathcal{D}_s$

The Landscape: *Recursion* Theory

	Determinism	**Probabilism**
	$M\overline{s} \to^* N_s$	$[\![M\overline{s}]\!] = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	

Undecidable;
Σ^0_1-complete.

	sm	**Probabilism**
	$M\overline{s} \to^* N_s$	$[\![M\overline{s}]\!] = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	

The Landscape: *Recursion* Theory

	Determinism	**Probabilism**
	$M\overline{s} \rightarrow^* N_s$	$[\![M\overline{s}]\!] = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	$\sum \mathcal{D}_s = 1$

The Landscape: *Recursion* Theory

Almost-Sure Termination
Π^0_2-complete.

	De	
	$M\overline{s} \to^* N_s$	$[\![M\overline{s}]\!] = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	$\sum \mathcal{D}_s = 1$

The Landscape: *Recursion* Theory

	Determinism	**Probabilism**
	$M\overline{s} \to^* N_s$	$[\![M\overline{s}]\!] = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	$\sum \mathcal{D}_s = 1$
Uniform Termination	$\forall s.\exists N_s \in NF$	

Π^0_2-complete.

	Determinism	**Probabilism**
	$M\overline{s} \to^* N_s$	$[\![M\overline{s}]\!] = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	$\sum \mathcal{D}_s = 1$
Uniform Termination	$\forall s.\exists N_s \in NF$	

The Landscape: *Recursion* Theory

	Determinism	**Probabilism**
	$M\overline{s} \to^* N_s$	$[\![M\overline{s}]\!] = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	$\sum \mathcal{D}_s = 1$
Uniform Termination	$\forall s.\exists N_s \in NF$	$\forall s. \sum \mathcal{D}_s = 1$

The Landscape: *Recursion* Theory

	Determinism	**Probabilism**
	$M\overline{s} \to^* N_s$	$[\![M\overline{s}]\!] = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	$\sum \mathcal{D}_s = 1$
Uniform Termination	$\forall s.\exists N_s \in NF$	$\forall s. \sum \mathcal{D}_s = 1$

Π^0_2-complete.

Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - But useless as a programming language.

Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - But useless as a programming language.

- For every type τ, define a set of reducible terms Red_τ.
- Prove that all reducible terms are normalizing...
- ...and that all typable terms are reducible.

Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - But useless as a programming language.
- What if we endow it with **full recursion** as a `fix` binder?

Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - But useless as a programming language.
- What if we endow it with **full recursion** as a `fix` binder?

$$(\mathtt{fix}\, x.M)V \to M\{\mathtt{fix}\, x.M/x\}V$$

Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - But useless as a programming language.
- What if we endow it with **full recursion** as a `fix` binder?
- All the termination properties are **lost**, for very good reasons.

Deterministic Sized Types

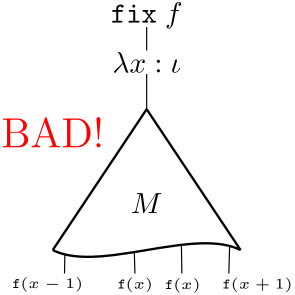
- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - But useless as a programming language.
- What if we endow it with **full recursion** as a `fix` binder?
- All the termination properties are **lost**, for very good reasons.
- Is **everything** lost?

Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - But useless as a programming language.
- What if we endow it with **full recursion** as a `fix` binder?
- All the termination properties are **lost**, for very good reasons.
- Is **everything** lost?
- **NO!**

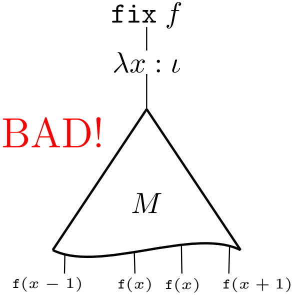
Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - But useless as a programming language.
- What if we endow it with **full recursion** as a `fix` binder?
- All the termination properties are **lost**, for very good reasons.
- Is **everything** lost?
- **NO!**



Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - But useless as a programming language.
- What if we endow it with **full recursion** as a `fix` binder?
- All the termination properties are **lost**, for very good reasons.
- Is **everything** lost?
- **NO!**



Deterministic Sized Types, Technically

- **Types**.

$$\xi ::= a \;\Big|\; \omega \;\Big|\; \xi + 1; \qquad\qquad \tau ::= \iota[\xi] \;\Big|\; \tau \to \tau.$$

Deterministic Sized Types, Technically

- **Types**.

$$\xi ::= a \;\Big|\; \omega \;\Big|\; \xi + 1; \qquad\qquad \tau ::= \iota[\xi] \;\Big|\; \tau \to \tau.$$

Index Terms

Deterministic Sized Types, Technically

- **Types.**

$$\xi ::= a \;\Big|\; \omega \;\Big|\; \xi + 1; \qquad\qquad \tau ::= \iota[\xi] \;\Big|\; \tau \to \tau.$$

- **Typing Fixpoints.**

$$\frac{\Gamma, x : \iota[a] \to \tau \vdash M : \iota[a+1] \to \tau}{\Gamma \vdash \mathtt{fix}\, x.M : \iota[\xi] \to \tau}$$

Deterministic Sized Types, Technically

- **Types**.

$$\xi ::= a \;\Big|\; \omega \;\Big|\; \xi + 1; \qquad\qquad \tau ::= \iota[\xi] \;\Big|\; \tau \to \tau.$$

- **Typing Fixpoints**.

$$\frac{\Gamma, x : \iota[a] \to \tau \vdash M : \iota[a+1] \to \tau}{\Gamma \vdash \mathtt{fix}\, x.M : \iota[\xi] \to \tau}$$

- **Quite Powerful**.
  - Can type many forms of structural recursion.

Deterministic Sized Types, Technically

- **Types**.

$$\xi ::= a \;\big|\; \omega \;\big|\; \xi + 1; \qquad\qquad \tau ::= \iota[\xi] \;\big|\; \tau \to \tau.$$

- **Typing Fixpoints**.

$$\frac{\Gamma, x : \iota[a] \to \tau \vdash M : \iota[a+1] \to \tau}{\Gamma \vdash \mathtt{fix}\, x.M : \iota[\xi] \to \tau}$$

- **Quite Powerful**.
  - Can type many forms of structural recursion.
- **Termination**.
  - Proved by **Reducibility**.
  - ...but of an indexed form.

Deterministic Sized Types, Technically

- **Types**.

$$\xi ::= a \;\big|\; \omega \;\big|\; \xi + 1; \qquad\qquad \tau ::= \iota[\xi] \;\big|\; \tau \to \tau.$$

- **Typing Fixpoints**.

- Reducibility sets are of the form Red_{τ}^{θ}.
- θ is an environment for index variables.
- Proof of reducibility for $\texttt{fix}\, x.M$ is rather delicate.

- **Q**
  - Can type many forms of structural recursion.
- **Termination**.
  - Proved by **Reducibility**.
  - …but of an indexed form.

Deterministic Sized Types, Technically

- **Types**.

$$\xi ::= a \;\big|\; \omega \;\big|\; \xi + 1; \qquad\qquad \tau ::= \iota[\xi] \;\big|\; \tau \to \tau.$$

- **Typing Fixpoints**.

$$\frac{\Gamma, x : \iota[a] \to \tau \vdash M : \iota[a+1] \to \tau}{\Gamma \vdash \mathtt{fix}\, x.M : \iota[\xi] \to \tau}$$

- **Quite Powerful**.
  - Can type many forms of structural recursion.
- **Termination**.
  - Proved by **Reducibility**.
  - . . . but of an indexed form.
- **Type Inference**.
  - It is indeed *decidable*.
  - But *nontrivial*.

Probabilistic Termination

- **Examples**:

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ x > 0\ \texttt{then if}\ FairCoin\ \texttt{then}\ f(x-1)\ \texttt{else}\ f(x+1);$

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ x > 0\ \texttt{then if}\ BiasedCoin\ \texttt{then}\ f(x-1)\ \texttt{else}\ f(x+1);$

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ BiasedCoin\ \texttt{then}\ f(x+1)\ \texttt{else}\ x.$

Probabilistic Termination

- **Examples**:

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ x > 0\ \texttt{then if}\ \mathit{FairCoin}\ \texttt{then}\ f(x-1)\ \texttt{else}\ f(x+1);$

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ x > 0\ \texttt{then if}\ \mathit{BiasedCoin}\ \texttt{then}\ f(x-1)\ \texttt{else}\ f(x+1);$

  $\texttt{then}\ f(x+1)\ \texttt{else}\ x.$

Unbiased Random Walk

Probabilistic Termination

- **Examples**:

  fix $f.\lambda x.$if $x > 0$ then if $FairCoin$ then $f(x-1)$ else $f(x+1)$;

  fix $f.\lambda x.$if $x > 0$ then if $BiasedCoin$ then $f(x-1)$ else $f(x+1)$;

  then $f(x+1)$ else x.

Unbiased Random Walk

Biased Randomn Walk

Probabilistic Termination

- **Examples**:

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ x > 0\ \texttt{then}\ \texttt{if}\ \mathit{FairCoin}\ \texttt{then}\ f(x-1)\ \texttt{else}\ f(x+1);$

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ x > 0\ \texttt{then}\ \texttt{if}\ \mathit{BiasedCoin}\ \texttt{then}\ f(x-1)\ \texttt{else}\ f(x+1);$

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ \mathit{BiasedCoin}\ \texttt{then}\ f(x+1)\ \texttt{else}\ x.$

- **Non-Examples**:

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ \mathit{FairCoin}\ \texttt{then}\ f(x-1)\ \texttt{else}\ (f(x+1); f(x+1));$

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ \mathit{BiasedCoin}\ \texttt{then}\ f(x+1)\ \texttt{else}\ f(x-1);$

Probabilistic Termination

- **Examples**:

  $\texttt{fix } f.\lambda x.\texttt{if } x > 0 \texttt{ then if } \mathit{FairCoin} \texttt{ then } f(x-1) \texttt{ else } f(x+1);$

  $\texttt{fix } f.\lambda x.\texttt{if } x > 0 \texttt{ then if } \mathit{BiasedCoin} \texttt{ then } f(x-1) \texttt{ else } f(x+1);$

  $\texttt{fix } f.\lambda x.\texttt{if } \mathit{BiasedCoin} \texttt{ then } f(x+1) \texttt{ else } x.$

- **Non-Examples**:

  $\texttt{fix } f.\lambda x.\texttt{if } \mathit{FairCoin} \texttt{ then } f(x-1) \texttt{ else } (f(x+1); f(x+1));$

  $\texttt{fix } f.\lambda x.\texttt{if } \mathit{BiasedCoin} \texttt{ then } f(x+1) \texttt{ else } f(x-1);$

Unbiased Random Walk, with **two** upward calls.

Probabilistic Termination

- **Examples**:

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ x > 0\ \texttt{then if}\ \mathit{FairCoin}\ \texttt{then}\ f(x-1)\ \texttt{else}\ f(x+1);$

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ x > 0\ \texttt{then if}\ \mathit{BiasedCoin}\ \texttt{then}\ f(x-1)\ \texttt{else}\ f(x+1);$

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ \mathit{BiasedCoin}\ \texttt{then}\ f(x+1)\ \texttt{else}\ x.$

- **Non-Examples**:

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ \mathit{FairCoin}\ \texttt{then}\ f(x-1)\ \texttt{else}\ (f(x+1); f(x+1));$

  $\texttt{fix}\ f.\lambda x.\texttt{if}\ \mathit{BiasedCoin}\ \texttt{then}\ f(x+1)\ \texttt{else}\ f(x-1);$

Unbiased Random Walk, with **two** upward calls.

Biased Random Walk, the "wrong" way.

Probabilistic Termination

- **Examples**:

$$\texttt{fix } f.\lambda x.\texttt{if } x > 0 \texttt{ then if } \mathit{FairCoin} \texttt{ then } f(x-1) \texttt{ else } f(x+1);$$
$$\texttt{fix } f.\lambda x.\texttt{if } x > 0 \texttt{ then if } \mathit{BiasedCoin} \texttt{ then } f(x-1) \texttt{ else } f(x+1);$$
$$\texttt{fix } f.\lambda x.\texttt{if } \mathit{BiasedCoin} \texttt{ then } f(x+1) \texttt{ else } x.$$

- **Non-Examples**:

$$\texttt{fix } f.\lambda x.\texttt{if } \mathit{FairCoin} \texttt{ then } f(x-1) \texttt{ else } (f(x+1); f(x+1));$$
$$\texttt{fix } f.\lambda x.\texttt{if } \mathit{BiasedCoin} \texttt{ then } f(x+1) \texttt{ else } f(x-1);$$

- Probabilistic termination **is** thus:
  - Sensitive to *the actual distribution* from which we sample.
  - Sensitive to *how many recursive calls* we perform.

One-Counter Blind Markov Chains

- They are automata of the form (Q, δ) where
  - Q is a finite set of *states*.
  - $\delta : Q \to \mathsf{Dist}(Q \times \{-1, 0, 1\})$.
- They are a very special form of One-Counter Markov Decision Processeses [BBEK2011].
  - The model is fully probabilistic, there is no nondeterminism.
  - The counter value is ignored.

One-Counter Blind Markov Chains

- They are automata of the form (Q, δ) where
  - Q is a finite set of *states*.
  - $\delta : Q \to \mathsf{Dist}(Q \times \{-1, 0, 1\})$.
- They are a very special form of One-Counter Markov Decision Processeses [BBEK2011].
  - The model is fully probabilistic, there is no nondeterminism.
  - The counter value is ignored.
- The probability of reaching a configuration where the counter is 0 can be approximated arbitrarily well *in polynomial time*.

Probabilistic Sized Types [DLGrellois2017]

- **Basic Idea**: craft a sized-type system in such a way as to mimick the recursive structure by a OCBMC.

Probabilistic Sized Types [DLGrellois2017]

- **Basic Idea**: craft a sized-type system in such a way as to mimick the recursive structure by a OCBMC.
- **Judgments**.

$$\Gamma \mid \Delta \vdash M : \mu$$

Probabilistic Sized Types [DLGrellois2017]

- **Basic Idea**: craft a sized-type system in such a way as to mimick the recursive structure by a OCBMC.
- **Judgments**.

$$\Gamma \mid \Delta \vdash M : \mu$$

Every higher-order variable occurs **at most once**.

Probabilistic Sized Types [DLGrellois2017]

- **Basic Idea**: craft a sized-type system in such a way as to mimick the recursive structure by a OCBMC.
- **Judgments**.

$$\Gamma \mid \Delta \vdash M : \mu$$

- **Typing Fixpoints**.

$$\frac{\Gamma \mid x : \sigma \vdash V : \iota[a+1] \to \tau \quad OCBMC(\sigma) \text{ terminates.}}{\Gamma \mid x : \sigma \vdash V : \iota[\xi] \to \tau}$$

Probabilistic Sized Types [DLGrellois2017]

- **Basic Idea**: craft a sized-type system in such a way as to mimick the recursive structure by a OCBMC.
- **Judgments**.

$$\Gamma \mid \Delta \vdash M : \mu$$

- **Typing Fixpoints**.

$$\frac{\Gamma \mid x : \sigma \vdash V : \iota[a+1] \to \tau \quad OCBMC(\sigma) \text{ terminates.}}{\Gamma \mid x : \sigma \vdash V : \iota[\xi] \to \tau}$$

This is sufficient for typing:

- Unbiased random walks;
- Biased random walks.

Probabilistic Sized Types [DLGrellois2017]

- **Basic Idea**: craft a sized-type system in such a way as to mimick the recursive structure by a OCBMC.
- **Judgments**.

$$\Gamma \mid \Delta \vdash M : \mu$$

- **Typing Fixpoints**.

$$\frac{\Gamma \mid x : \sigma \vdash V : \iota[a+1] \to \tau \quad OCBMC(\sigma) \text{ terminates.}}{\Gamma \mid x : \sigma \vdash V : \iota[\xi] \to \tau}$$

- **Typing Probabilistic Choice**

$$\frac{\Gamma \mid \Delta \vdash M : \tau \quad \Gamma \mid \Omega \vdash N : \rho}{\Gamma \mid \frac{1}{2}\Delta + \frac{1}{2}\Omega \vdash M \oplus N : \frac{1}{2}\tau + \frac{1}{2}\rho}$$

Probabilistic Sized Types [DLGrellois2017]

- **Basic Idea**: craft a sized-type system in such a way as to mimick the recursive structure by a OCBMC.
- **Judgments**.

$$\Gamma \mid \Delta \vdash M : \mu$$

- **Typing Fixpoints**.

$$\frac{\Gamma \mid x : \sigma \vdash V : \iota[a+1] \to \tau \quad OCBMC(\sigma) \text{ terminates.}}{\Gamma \mid x : \sigma \vdash V : \iota[\xi] \to \tau}$$

- **Typing Probabilistic Choice**

$$\frac{\Gamma \mid \Delta \vdash M : \tau \quad \Gamma \mid \Omega \vdash N : \rho}{\Gamma \mid \frac{1}{2}\Delta + \frac{1}{2}\Omega \vdash M \oplus N : \frac{1}{2}\tau + \frac{1}{2}\rho}$$

- **Termination**.
  - By a quantitative nontrivial refinement of reducibility.

Probabilistic Sized Types [DLGrellois2017]

- **Basic Idea**: craft a sized-type system in such a way as to mimick the recursive structure by a OCBMC.
- **Judgments**.

$$\Gamma \mid \Delta \vdash M : \mu$$

- **Typing Fixpoints**.

- Reducibility sets are now on the form $Red_{\tau}^{\theta,p}$
- p stands for the *probability* of being reducible.
- Reducibility sets are continuous:

$$Red_{\tau}^{\theta,p} = \bigcup_{q<p} Red_{\tau}^{\theta,q}$$

- **Termination**.
  - By a quantitative nontrivial refinement of reducibility.

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?
- Very simple examples of normalizing terms which *canoot* be typed:

$$\Delta = \lambda x.xx \qquad\qquad \Delta(\lambda x.x).$$

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?
- Very simple examples of normalizing terms which *canoot* be typed:

$$\Delta = \lambda x.xx \qquad\qquad \Delta(\lambda x.x).$$

- **Types**

$$\tau ::= \star \;\Big|\; A \to B \qquad\qquad A ::= \{\tau_1, \ldots, \tau_n\}$$

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?
- Very simple examples of normalizing terms which *canoot* be typed:

$$\Delta = \lambda x.xx \qquad\qquad \Delta(\lambda x.x).$$

- **Types**

$$\tau ::= \star \;\Big|\; A \to B \qquad\qquad A ::= \{\tau_1, \ldots, \tau_n\}$$

- **Typing Rules: Examples**

$$\frac{\{\Gamma \vdash M : \tau_i\}_{1 \le i \le n}}{\Gamma \vdash M : \{\tau_1, \ldots, \tau_n\}} \qquad\qquad \frac{\Gamma \vdash M : \{A \to B\} \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?
- Very simple examples of normalizing terms which *canoot* be typed:

$$\Delta = \lambda x.xx \qquad\qquad \Delta(\lambda x.x).$$

- **Types**

$$\tau ::= \star \;\Big|\; A \to B \qquad\qquad A ::= \{\tau_1, \dots, \tau_n\}$$

- **Typing Rules: Examples**

$$\frac{\{\Gamma \vdash M : \tau_i\}_{1 \le i \le n}}{\Gamma \vdash M : \{\tau_1, \dots, \tau_n\}} \qquad\qquad \frac{\Gamma \vdash M : \{A \to B\} \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

- **Termination**
  - Again by reducibility.

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?
- Very simple examples of normalizing terms which *canoot* be typed:

$$\Delta = \lambda x.xx \qquad\qquad \Delta(\lambda x.x).$$

- **Types**

$$\tau ::= \star \;\Big|\; A \to B \qquad\qquad A ::= \{\tau_1, \ldots, \tau_n\}$$

- **Typing Rules: Examples**

$$\frac{\{\Gamma \vdash M : \tau_i\}_{1 \le i \le n}}{\Gamma \vdash M : \{\tau_1, \ldots, \tau_n\}} \qquad\qquad \frac{\Gamma \vdash M : \{A \to B\} \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

- **Termination**
  - Again by reducibility.
- **Completeness**
  - By *subject expansion*, the dual of subject reduction.

Oracle Intersection Types [BreuvartDL2017]

- Probabilistic choice can be seen as a form of read operation:

$$M \oplus N = \texttt{if } \mathit{BitInput} \texttt{ then } M \texttt{ else } N$$

Oracle Intersection Types [BreuvartDL2017]

- Probabilistic choice can be seen as a form of read operation:

$$M \oplus N = \texttt{if}\ \mathit{BitInput}\ \texttt{then}\ M\ \texttt{else}\ N$$

- **Types**

$$\tau ::= \star \;\Big|\; A \to s \cdot B \qquad A ::= \{\tau_1, \ldots, \tau_n\} \qquad s \in \{0,1\}^*$$

Oracle Intersection Types [BreuvartDL2017]

- Probabilistic choice can be seen as a form of read operation:

$$M \oplus N = \texttt{if } \mathit{BitInput} \texttt{ then } M \texttt{ else } N$$

- **Types**

$$\tau ::= \star \;\Big|\; A \to s \cdot B \qquad A ::= \{\tau_1, \ldots, \tau_n\} \qquad s \in \{0,1\}^*$$

- **Typing Rules: Examples**

$$\frac{\Gamma \vdash M : s \cdot A}{\Gamma \vdash M \oplus N : 0s \cdot A} \qquad \frac{\Gamma \vdash M : r \cdot \{A \to s \cdot B\} \quad \Gamma \vdash N : q \cdot A}{\Gamma \vdash MN : (rqs) \cdot B}$$

Oracle Intersection Types [BreuvartDL2017]

- Probabilistic choice can be seen as a form of read operation:

$$M \oplus N = \texttt{if}\ \mathit{BitInput}\ \texttt{then}\ M\ \texttt{else}\ N$$

- **Types**

$$\tau ::= \star \;\Big|\; A \to s \cdot B \qquad A ::= \{\tau_1, \ldots, \tau_n\} \qquad s \in \{0,1\}^*$$

- **Typing Rules: Examples**

$$\frac{\Gamma \vdash M : s \cdot A}{\Gamma \vdash M \oplus N : 0s \cdot A} \qquad \frac{\Gamma \vdash M : r \cdot \{A \to s \cdot B\} \quad \Gamma \vdash N : q \cdot A}{\Gamma \vdash MN : (rqs) \cdot B}$$

- **Termination and Completeness**
  - Formulated in a rather *unusual* way.
  - Proved as usual, but relative to a single probabilistic branch

Oracle Intersection Types [BreuvartDL2017]

- Probabilistic choice can be seen as a form of read operation:

$$M \oplus N = \texttt{if } \mathit{BitInput} \texttt{ then } M \texttt{ else } N$$

- **Types**

$$\tau ::= \star \;\Big|\; A \to s \cdot B \qquad A ::= \{\tau_1, \ldots, \tau_n\} \qquad s \in \{0,1\}^*$$

$$\mathbb{P}(M\downarrow) = \sum_{\vdash M : s \cdot \star} 2^{|s|}$$

- **Ty**

$$\frac{\Gamma \vdash M : s \cdot A}{\Gamma \vdash M \oplus N : 0s \cdot A} \qquad \frac{\Gamma \vdash M : r \cdot \{A \to s \cdot B\} \quad \Gamma \vdash N : q \cdot A}{\Gamma \vdash MN : (rqs) \cdot B}$$

- **Termination and Completeness**
  - Formulated in a rather *unusual* way.
  - Proved as usual, but relative to a single probabilistic branch

Oracle Intersection Types [BreuvartDL2017]

- Probabilistic choice can be seen as a form of read operation:

$$M \oplus N = \texttt{if}\ \mathit{BitInput}\ \texttt{then}\ M\ \texttt{else}\ N$$

- **Types**

$$\tau ::= \star \;\Big|\; A \to s \cdot B \qquad A ::= \{\tau_1, \ldots, \tau_n\} \qquad s \in \{0,1\}^*$$

$$\mathbb{P}(M \downarrow) = \sum_{\vdash M : s \cdot \star} 2^{|s|}$$

- **Ty**

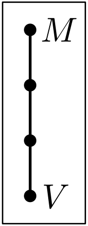
This is **unavoidable**, due to recursion theory.

$\Gamma \vdash N : q \cdot A$

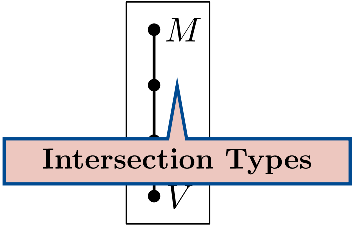
$\Gamma \vdash M \oplus N : 0s \cdot A$ $\qquad$ $\Gamma \vdash MN : (rqs) \cdot B$

- **Termination and Completeness**
  - Formulated in a rather *unusual* way.
  - Proved as usual, but relative to a single probabilistic branch

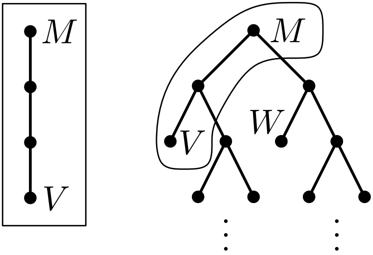
Intersection Types and Computations



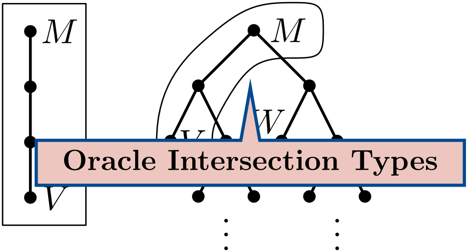
Intersection Types and Computations



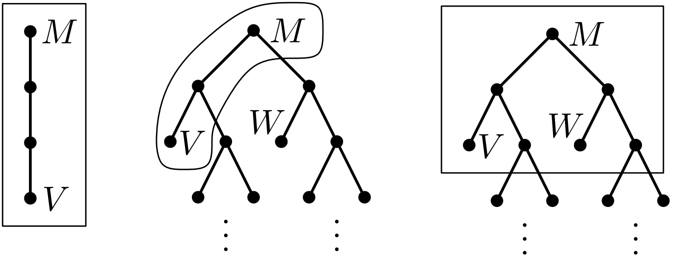
Intersection Types and Computations



Intersection Types and Computations



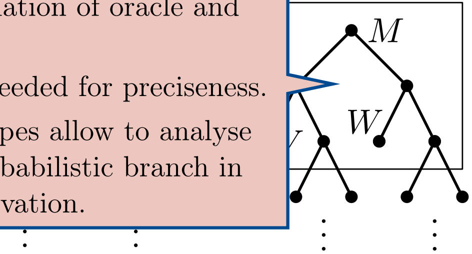
Intersection Types and Computations



Intersection Types and Computations

Monadic Intersection Types [BDL2017]

- They are a combination of oracle and sized types.
- Intersections are needed for preciseness.
- Distributions of types allow to analyse more than one probabilistic branch in the same type derivation.



These Slides, and More...

These Slides, and More...

Questions?