Reasoning Operationally About Probabilistic Higher-Order Programs

Ugo Dal Lago

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Inria informatiques mathématiques

European Research Council
Established by the European Commission

CIRM, Marseille, May 14, 2024

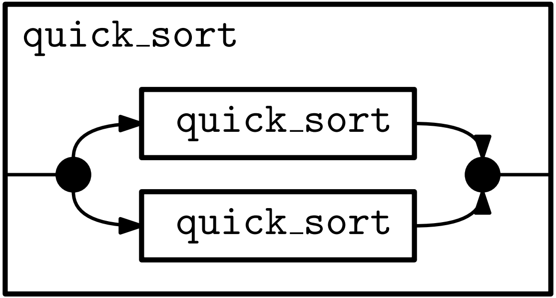
Part I

Probabilistic Higher-Order Programs

QuickSort

```
let app x y z = x @ (z::y);;

let partition = function
    | pivot :: rest -> (List.partition (( > ) pivot) (rest)),pivot;;

let rec quick_sort = function
    | []
    | [_] as list -> list
    | list ->
        let (l1, l2), el = partition list in
          app (quick_sort l1) (quick_sort l2) el;;
```

The Structure of QuickSort



QuickSort, HO

```
let app = function
    | (x,y),z -> x @ (z::y);;

let partition = function
    | pivot :: rest -> (List.partition (( > ) pivot) (rest)),pivot;;

let rec dac divide conquer = function
    | []
    | [_] as list -> list
    | list ->
        let (l1, l2),el = divide list in
          conquer ((dac divide conquer l1, dac divide conquer l2),el);;

let quick_sort = dac partition app;;
```

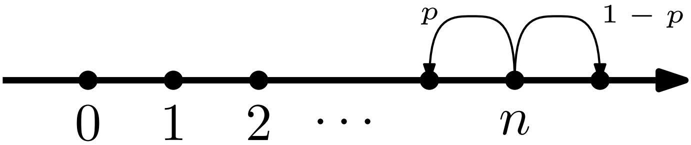
Randomized QuickSort (1)

```
let app = function
    | (x,y),z -> x @ (z::y);;

let rec extract = function
   | [],_ -> ([],0)
   | hd::tl,n ->
      if n==0 then
        (tl,hd)
      else
        let (l,el) = extract(tl,n-1) in
           (hd::l,el);;

let partition list =
  let (rest,pivot) = extract (list,(Random.int (List.length list))) in
    (List.partition (( > ) pivot) (rest)),pivot;;
```

Randomized QuickSort (2)

```
let rec dac divide conquer = function
      | []
      | [_] as list -> list
      | list ->
          let (l1, l2),el = divide list in
            conquer ((dac divide conquer l1, dac divide conquer l2),el);;

let rand_quick_sort = dac partition app;;
```

Random Walk



Two Kinds of Random Walks

```
let rec iter f g n = if n==0 then g else let m=pred(n) in f m (iter f g m);;

let mult m n = succ(m)*n;;

let fact = iter mult 1;;

let rec param_iter f g step n =
    if n==0 then g else let m=step(n) in f m (param_iter f g step m);;

let succ_2 m n = n+1;;

let updown_fair x = x+(2*Random.int(2)-1);;

let fair_random_walk = param_iter succ_2 0 updown_fair;;

let updown_biased x = if Random.int(3)==0 then x+1 else x-1;;

let biased_random_walk = param_iter succ_2 0 updown_biased;;
```

The Coupon Collector

- A supermarket distributes a large quantity of coupons, each labelled with $i \in \{1, \ldots, n\}$.

The Coupon Collector

- A supermarket distributes a large quantity of coupons, each labelled with $i \in \{1, \dots, n\}$.
- Every day, you collect one coupon at the supermarket. Any label i has probability $\frac{1}{n}$ to occur.

The Coupon Collector

- A supermarket distributes a large quantity of coupons, each labelled with $i \in \{1, \ldots, n\}$.
- Every day, you collect one coupon at the supermarket. Any label i has probability $\frac{1}{n}$ to occur.
- You win a prize when you collect a set of n coupons, each with a distinct label.

The Coupon Collector

- A supermarket distributes a large quantity of coupons, each labelled with $i \in \{1, \ldots, n\}$.
- Every day, you collect one coupon at the supermarket. Any label i has probability $\frac{1}{n}$ to occur.
- You win a prize when you collect a set of n coupons, each with a distinct label.
- **Example**: if $n = 5$, you could get the following coupons:

$$3, 1, 5, 2, 3, 1, 5, 2, 3, 1, 5, 2, \ldots$$

The Coupon Collector

- A supermarket distributes a large quantity of coupons, each labelled with $i \in \{1, \ldots, n\}$.
- Every day, you collect one coupon at the supermarket. Any label i has probability $\frac{1}{n}$ to occur.
- You win a prize when you collect a set of n coupons, each with a distinct label.
- **Example**: if $n = 5$, you could get the following coupons:

$$3, 1, 5, 2, 3, 1, 5, 2, 3, 1, 5, 2, \ldots$$

- Are you guaranteed to win the prize with probability 1? After how many days, on the average?

The Coupon Collector

```
let rec base_param_iter f g step base e =
  if base(e) then g else let d=step(e) in f d (base_param_iter f g step base d);;

let second_zero = function
  | (_,0) -> true
  | _ -> false;;

let succ_2 m n = n+1;;

let step_2 = function
  | (n,m) -> if Random.int(n)<=m then (n,m-1) else (n,m);;

let coupon_collector x = base_param_iter succ_2 0 step_2 second_zero (x,x);;
```

Part II

Probabilistic Termination

What Algorithms Compute

- **Deterministic Computation**
  - For every input x, there is *at most* one output y any algorithm $\mathcal{A}$ produces when fed with x.
  - As a consequence:

$$\mathcal{A} \qquad \rightsquigarrow \qquad [\![\mathcal{A}]\!] : \mathbb{N} \rightharpoonup \mathbb{N}.$$

What Algorithms Compute

- **Deterministic Computation**
  - For every input x, there is *at most* one output y any algorithm $\mathcal{A}$ produces when fed with x.
  - As a consequence:

$$\mathcal{A} \qquad \rightsquigarrow \qquad [\![\mathcal{A}]\!] : \mathbb{N} \rightharpoonup \mathbb{N}.$$

- **Randomized Computation**
  - For every input x, any algorithm $\mathcal{A}$ outputs y with a probability $0 \leq p \leq 1$.
  - As a consequence:

$$\mathcal{A} \qquad \rightsquigarrow \qquad [\![\mathcal{A}]\!] : \mathbb{N} \rightarrow \mathscr{D}(\mathbb{N}).$$

  - The distribution $[\![\mathcal{A}]\!](n)$ sums to anything between 0 and 1, thus accounting for the probability of divergence.

Deterministic vs. Probabilistic Transition Systems

M

Deterministic vs. Probabilistic Transition Systems

$$M \longrightarrow N$$

Deterministic vs. Probabilistic Transition Systems

$$M \longrightarrow N \longrightarrow L$$

Deterministic vs. Probabilistic Transition Systems

$$M \longrightarrow N \longrightarrow L \longrightarrow \cdots$$

Deterministic vs. Probabilistic Transition Systems

$$M \longrightarrow N \longrightarrow L \longrightarrow \cdots$$

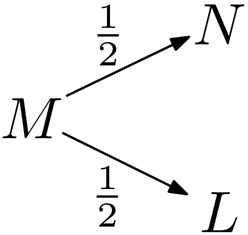
$$(\mathcal{A}, \longrightarrow\!\!\!\blacktriangleright)$$

$$\longrightarrow\!\!\!\blacktriangleright : \mathcal{A} \rightharpoonup \mathcal{A}$$

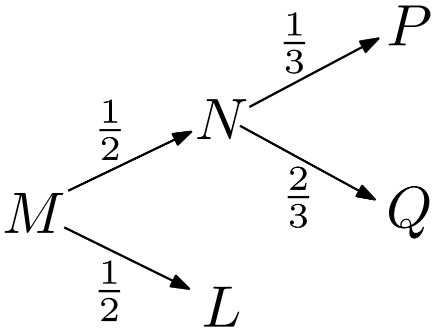
Deterministic vs. Probabilistic Transition Systems

M

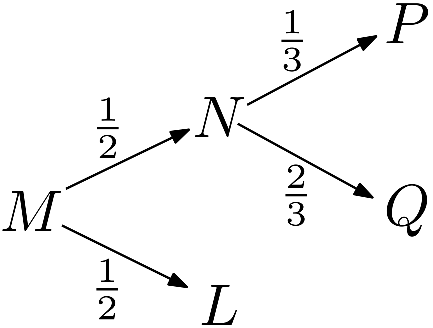
Deterministic vs. Probabilistic Transition Systems



Deterministic vs. Probabilistic Transition Systems



Deterministic vs. Probabilistic Transition Systems



$$(\mathcal{A}, \longrightarrow)$$

$$\longrightarrow : \mathcal{A} \rightharpoonup \mathscr{D}(\mathcal{A})$$

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

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

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**:

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

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

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

$$E ::= [\cdot] \mid EM \mid VE$$

$$\frac{}{E[(\lambda x.M)V] \rightarrow \{E[M[x/V]]^1\}} \qquad \frac{}{E[M \oplus N] \rightarrow \{E[M]^{\frac{1}{2}}, E[N]^{\frac{1}{2}}\}}$$

$$\frac{}{M \Rightarrow \emptyset} \qquad \frac{}{V \Rightarrow \{V^1\}} \qquad \frac{M \rightarrow \mathcal{D} \qquad \{P \Rightarrow \mathcal{E}_P\}_{P \in \mathsf{S}\mathcal{D}}}{M \Rightarrow \sum_{P \in \mathsf{S}\mathcal{D}} \mathcal{D}(P)\mathcal{E}_P}$$

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

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

$$C ::= [\cdot] \;\big|\; \lambda x.C \;\big|\; CM \;\big|\; MC \;\big|\; C \oplus M \;\big|\; M \oplus C$$

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

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

Syntax and Operational Semantics of $\Lambda_\oplus$

All this can be easily generalized to:

- Typed calculi.
- CBN
- Recursion.
- ...

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

Notions of Probabilistic Termination

- Given $M \in \mathcal{A}$, let RF_M be the number of transitions leading M to an irreducible N (or ∞ if such an N does not exist).

Notions of Probabilistic Termination

- Given $M \in \mathcal{A}$, let RF_M be the number of transitions leading M to an irreducible N (or ∞ if such an N does not exist).
  - In *deterministic* transition systems, $\mathsf{RF}_M \in \mathbb{N}^{\infty}$.
  - In *probabilistic* transition systems RF_M is a random variable.

Notions of Probabilistic Termination

- Given $M \in \mathcal{A}$, let RF_M be the number of transitions leading M to an irreducible N (or ∞ if such an N does not exist).
  - In *deterministic* transition systems, $\mathsf{RF}_M \in \mathbb{N}^\infty$.
  - In *probabilistic* transition systems RF_M is a random variable.
- **Almost-Sure Termination** (AST).

$$\Pr(\mathsf{RF}_M < \infty) = 1$$

Notions of Probabilistic Termination

- Given $M \in \mathcal{A}$, let RF_M be the number of transitions leading M to an irreducible N (or ∞ if such an N does not exist).
  - In *deterministic* transition systems, $\mathsf{RF}_M \in \mathbb{N}^\infty$.
  - In *probabilistic* transition systems RF_M is a random variable.
- **Almost-Sure Termination** (AST).

$$\Pr(\mathsf{RF}_M < \infty) = 1$$

- **Positive Almost-Sure Termination** (PAST).

$$\mathbb{E}(\mathsf{RF}_M) < \infty$$

Notions of Probabilistic Termination

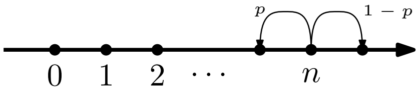
- Given $M \in \mathcal{A}$, let RF_M be the number of transitions leading M to an irreducible N (or ∞ if such an N does not exist).
  - In *deterministic* transition systems, $\mathsf{RF}_M \in \mathbb{N}^\infty$.
  - In *probabilistic* transition systems RF_M is a random variable.
- **Almost-Sure Termination** (AST).

$$\Pr(\mathsf{RF}_M < \infty) = 1$$

- **Positive Almost-Sure Termination** (PAST).

$$\mathbb{E}(\mathsf{RF}_M) < \infty$$

- PAST implies AST, but not *viceversa*. A counterexample is the *fair* (i.e. $p = \frac{1}{2}$) random walk:

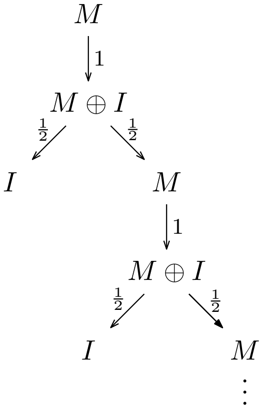


An Example

$$M = (\lambda x.xx \oplus I)(\lambda x.xx \oplus I)$$

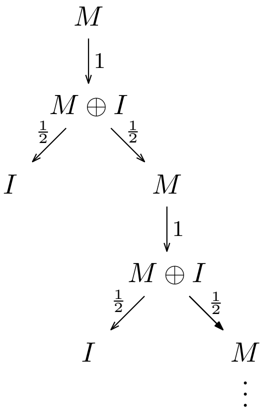
An Example

$$M = (\lambda x.xx \oplus I)(\lambda x.xx \oplus I)$$



An Example

$$M = (\lambda x.xx \oplus I)(\lambda x.xx \oplus I)$$



$$\Pr(\mathsf{Time}(M) < \infty) = \sum_{i=0}^{\infty} \frac{1}{2^{i+1}} = 1$$

$$\mathbb{E}(\mathsf{Time}(M)) = \sum_{i=0}^{\infty} \Pr(\mathsf{Time}(M) > i)$$

$$= 2 \cdot \sum_{i=0}^{\infty} \frac{1}{2^i} = 4$$

The Landscape: *Recursion* Theory

- ***Deterministic* Computation**

	Instance	Universal
Termination	Σ_1^0-complete	Π_2^0-complete

The Landscape: *Recursion* Theory

- ***Deterministic* Computation**

	Instance	Universal
Termination	Σ^0_1-complete	Π^0_2-complete

- ***Probabilistic* Computation** [KK2013]

	Instance	Universal
AST	Π^0_2-complete	Π^0_2-complete
PAST	Σ^0_2-complete	Π^0_3-complete

Part III

Probabilistic Termination and Types

The Landscape: Termination and Types

Simple Types

$$\tau \ ::= \ \cdots \mid \tau \to \tau$$

The Landscape: Termination and Types

- Corresponds to Intuitionistic Logic or HA;
- Sound for termination, in presence of primitive recursion;
- Very limited expressive power.

Simple Types

$$\tau \ ::= \ \cdots \mid \tau \to \tau$$

The Landscape: Termination and Types

Sized Types

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

Simple Types

$$\tau \ ::= \ \cdots \mid \tau \to \tau$$

The Landscape: Termination and Types

Sized Types

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

Simple Types

$$\cdots \mid \tau \to \tau$$

- Reasonably expressive, intensionally;
- Type inference remains decidable.

The Landscape: Termination and Types

Sized Types

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

Intersection Types

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

Simple Types

$$\tau \;::=\; \cdots \mid \tau \to \tau$$

The Landscape: Termination and Types

Sized Types

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

Intersection Types

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

Simple Types

$$\tau ::= \cdots$$

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

The Landscape: Termination and Types

Sized Types

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

Intersection Types

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

Simple Types

$$\tau \;::=\; \cdots \mid \tau \to \tau$$

Linear Dependent Types

$$\tau \;::=\; \cdots \mid \iota[I] \mid !_{i<I}\tau$$

The Landscape: Termination and Types

Sized Types

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

Intersection Types

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

- Relatively complete for termination and complexity analysis;
- Type inference is hard, but can be mechanized.

S

$$\tau ::= \cdots \mid \tau \rightarrow \tau$$

Linear Dependent Types

$$\tau ::= \cdots \mid \iota[I] \mid !_{i<I}\tau$$

Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - The absence of full recursion can be cumbersome.

Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - The absence of full recursion can be cumbersome

- 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**.
  - The absence of full recursion can be cumbersome.
- 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**.
  - The absence of full recursion can be cumbersome.
- What if we endow it with **full recursion** as a `fix` binder?

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

Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - The absence of full recursion can be cumbersome.
- What if we endow it with **full recursion** as a `fix` binder?
- Termination is **not guaranteed** anymore, for very good reasons.

Deterministic Sized Types

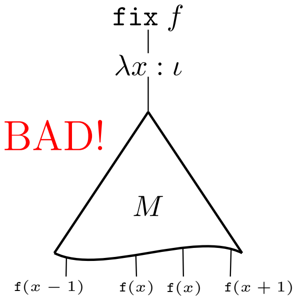
- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - The absence of full recursion can be cumbersome.
- What if we endow it with **full recursion** as a `fix` binder?
- Termination is **not guaranteed** anymore, for very good reasons.
- Is **everything** lost?

Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - The absence of full recursion can be cumbersome.
- What if we endow it with **full recursion** as a `fix` binder?
- Termination is **not guaranteed** anymore, for very good reasons.
- Is **everything** lost?
- **NO!**

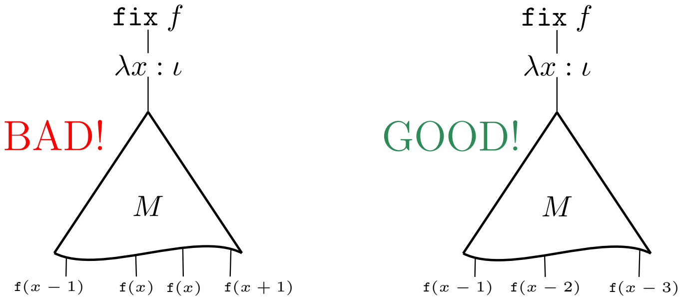
Deterministic Sized Types

- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - The absence of full recursion can be cumbersome.
- What if we endow it with **full recursion** as a `fix` binder?
- Termination is **not guaranteed** anymore, 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**.
  - The absence of full recursion can be cumbersome.
- What if we endow it with **full recursion** as a `fix` binder?
- Termination is **not guaranteed** anymore, 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**.

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

- **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.**

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

- **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**.

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

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

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

- **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**:

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

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

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

Probabilistic Termination

- **Examples**:

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

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

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

Unbiased Random Walk, AST

Probabilistic Termination

- **Examples**:

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

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

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

Unbiased Random Walk, AST

Biased Random Walk, PAST

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

- **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**:

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

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

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

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**:

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

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.

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**.

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

λ-variables: 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**.

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

λ-variables: every higher-order variable occurs **at most once**.

`fix`-variables, which are attributed a **finite distribution** of types.

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**.

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

λ-variables: every higher-order variable occurs **at most once**.

`fix`-variables, which are attributed a **finite distribution** of types.

A monadic type, namely a **finite distribution** of types.

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**.

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

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**.

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

- $\mathit{OCBMC}(\sigma)$ interprets σ as a one-counter Markov Chain.
- This is sufficient for typing:
  - *Unbiased* random walks:

$$\sigma = \left\{ \frac{1}{2} : \iota[a+2] \to \tau, \frac{1}{2} : \iota[a] \to \tau \right\}$$

  - *Biased* random walks, with a similar σ.

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**.

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

- **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**.

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

- **Typing Probabilistic Choice**

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

- **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 Fix**
- **Typing Pro**

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

- 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*?
- Many normalizing terms *cannot* be typed, e.g.:

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

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?
- Many normalizing terms *cannot* be typed, e.g.:

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

- **Types**

$$\tau ::= A \to B \qquad\qquad A ::= [\tau_1, \dots, \tau_n]$$

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?
- Many normalizing terms *cannot* be typed, e.g.:

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

- **Types**

$$\tau ::= A \to B \qquad\qquad A ::= [\tau_1, \ldots, \tau_n]$$

- **Typing Rules**

$$\frac{}{x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \qquad \frac{(\Gamma_i \vdash V : \tau_i)_i}{\biguplus \Gamma_i \vdash V : [\tau_i]_i}$$

$$\frac{\Gamma \vdash V : [A \to B] \quad \Delta \vdash W : A}{\Gamma \uplus \Delta \vdash VW : B} \qquad \frac{\Gamma \vdash M : A \quad \Delta, x : A \vdash N : B}{\Gamma \uplus \Delta \vdash \texttt{let } x = M \texttt{ in } N : B}$$

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?
- Many normalizing terms *cannot* be typed, e.g.:

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

- **Types**

$$\tau ::= A \to B \qquad A ::= [\tau_1, \ldots, \tau_n]$$

- **Typing Rules**

$$\frac{}{x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \qquad \frac{(\Gamma_i \vdash V : \tau_i)_i}{\biguplus \Gamma_i \vdash V : [\tau_i]_i}$$

$$\frac{\Gamma \vdash V : [A \to B] \quad \Delta \vdash W : A}{\Gamma \uplus \Delta \vdash VW : B} \qquad \frac{\Gamma \vdash M : A \quad \Delta, x : A \vdash N : B}{\Gamma \uplus \Delta \vdash \texttt{let}\ x = M\ \texttt{in}\ N : B}$$

- **Soundness for Termination**
  - By reducibility.

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?
- Many normalizing terms *cannot* be typed, e.g.:

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

- **Types**

$$\tau ::= A \to B \qquad\qquad A ::= [\tau_1, \ldots, \tau_n]$$

- **Typing Rules**

$$\frac{}{x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \qquad \frac{(\Gamma_i \vdash V : \tau_i)_i}{\biguplus \Gamma_i \vdash V : [\tau_i]_i}$$

$$\frac{\Gamma \vdash V : [A \to B] \quad \Delta \vdash W : A}{\Gamma \uplus \Delta \vdash VW : B} \qquad \frac{\Gamma \vdash M : A \quad \Delta, x : A \vdash N : B}{\Gamma \uplus \Delta \vdash \texttt{let}\ x = M\ \texttt{in}\ N : B}$$

- **Soundness for Termination**
  - By reducibility.
- **Completeness for Termination**
  - By *subject expansion*, the dual of subject reduction.

Going Probabilistic: What Do We Need?

- **Terms can have a Type *Probabilistically***, for example in

$$\Omega \oplus (\lambda x.x)$$

Going Probabilistic: What Do We Need?

- **Terms can have a Type *Probabilistically***, for example in

$$\Omega \oplus (\lambda x.x)$$

  - This can be captured by switching from types to *multidistributions* of types, namely expressions in the form $\langle p_1 \tau_1, \ldots, p_n \tau_n \rangle$, where $\sum p_i \leq 1$.

Going Probabilistic: What Do We Need?

- **Terms can have a Type *Probabilistically***, for example in

$$\Omega \oplus (\lambda x.xx)$$

$$\mathcal{A} \neq \frac{1}{2}\mathcal{A} + \frac{1}{2}\mathcal{A}$$

  - This can be captured by switching from types to *multidistributions* of types, namely expressions in the form $\langle p_1\tau_1, \ldots, p_n\tau_n \rangle$, where $\sum p_i \leq 1$.

Going Probabilistic: What Do We Need?

- **Terms can have a Type *Probabilistically***, for example in

$$\Omega \oplus (\lambda x.x)$$

  - This can be captured by switching from types to *multidistributions* of types, namely expressions in the form $\langle p_1\tau_1, \ldots, p_n\tau_n\rangle$, where $\sum p_i \leq 1$.

- **Functions can use their Arguments *Probabilistically***, for example in

$$\lambda x.(xx) \oplus I$$

Going Probabilistic: What Do We Need?

- **Terms can have a Type *Probabilistically***, for example in

$$\Omega \oplus (\lambda x.x)$$

  - This can be captured by switching from types to *multidistributions* of types, namely expressions in the form $\langle p_1 \tau_1, \ldots, p_n \tau_n \rangle$, where $\sum p_i \leq 1$.

- **Functions can use their Arguments *Probabilistically***, for example in

$$\lambda x.(xx) \oplus I$$

  - This can be captured by switching from multisets to *scaled multisets* of types, namely expressions in the form $[q_1.\tau_1, \ldots, q_n.\tau_n]$. Here $\sum q_i \in \mathbb{Q}^+$.

Going Probabilistic: What Do We Need?

- **Terms can have a Type *Probabilistically***, for example in

$$\Omega \oplus (\lambda x.x)$$

  - This can be captured by switching from types to *multidistributions* of types, namely expressions in the form $\langle p_1\tau_1, \ldots, p_n\tau_n\rangle$, where $\sum p_i \leq 1$.

- **Functions can use their Arguments *Probabilistically***, for example in

$$\lambda x.(xx) \oplus I$$

  - This can be captured by switching from multisets to *scaled multisets* of types, namely expressions in the form $[q_1.\tau_1, \ldots, q_n.\tau_n]$. Here $\sum q_i \in \mathbb{Q}^+$.

- **Summing up...**

$$\tau ::= A \rightarrow \mathcal{A} \qquad A ::= [q_i.\tau_i]_{1\leq i\leq n} \qquad \mathcal{A} ::= \langle p_i A_i\rangle_{1\leq i\leq n}$$

Going Probabilistic: What Do We Need?

- **Terms can have a Type *Probabilistically***, for example in

$$\Omega \oplus (\lambda x.x)$$

  - This can be captured by switching from types to *multidistributions* of types, namely expressions in the form $\langle p_1 \tau_1, \ldots, p_n \tau_n \rangle$, where $\sum p_i \leq 1$.

- **Functions can use their Arguments *Probabilistically***, for example in

$$\lambda x.(xx) \oplus I$$

  - This can be captured by switching from multisets to *scaled multisets* of types, namely expressions in the form $[q_1.\tau_1, \ldots, q_n.\tau_n]$. Here $\sum q_i \in \mathbb{Q}^+$.

Arrow Types

- **Summing up...**

$$\tau ::= A \to \mathcal{A} \qquad A ::= [q_i.\tau_i]_{1 \leq i \leq n} \qquad \mathcal{A} ::= \langle p_i A_i \rangle_{1 \leq i \leq n}$$

Going Probabilistic: What Do We Need?

- **Terms can have a Type *Probabilistically***, for example in

$$\Omega \oplus (\lambda x.x)$$

  - This can be captured by switching from types to *multidistributions* of types, namely expressions in the form $\langle p_1\tau_1, \ldots, p_n\tau_n\rangle$, where $\sum p_i \leq 1$.

- **Functions can use their Arguments *Probabilistically***, for example in

$$\lambda x.(xx) \oplus I$$

  - This can be captured by switching from multisets to *scaled multisets* of types, namely expressions in t … Here $\sum q_i \in \mathbb{Q}^+$.

Intersection Types

- **Summing up…**

$$\tau ::= A \to \mathcal{A} \qquad A ::= [q_i.\tau_i]_{1\leq i\leq n} \qquad \mathcal{A} ::= \langle p_i A_i\rangle_{1\leq i\leq n}$$

Going Probabilistic: What Do We Need?

- **Terms can have a Type *Probabilistically***, for example in

$$\Omega \oplus (\lambda x.x)$$

  - This can be captured by switching from types to *multidistributions* of types, namely expressions in the form $\langle p_1\tau_1, \ldots, p_n\tau_n\rangle$, where $\sum p_i \leq 1$.

- **Functions can use their Arguments *Probabilistically***, for example in

$$\lambda x.(xx) \oplus I$$

  - This can be captured by switching from multisets to *scaled multisets* of types, namely expressions in the form $[q_1.\tau_1, \ldots, q_n.\tau_n]$. Here $\sum$ … $\in \mathbb{Q}^+$

Type Distributions

- **Summing up. . .**

$$\tau ::= A \rightarrow \mathcal{A} \qquad A ::= [q_i.\tau_i]_{1\leq i\leq n} \qquad \mathcal{A} ::= \langle p_i A_i\rangle_{1\leq i\leq n}$$

Typing Rules [DLFR2021]

$$\frac{}{x : A \vdash x : A} \qquad \frac{}{\vdash M : \mathbf{0}} \qquad \frac{\Gamma, x : A \vdash M : \mathcal{B}}{\Gamma \vdash \lambda x.M : A \to \mathcal{B}}$$

$$\frac{\Gamma \vdash V : [A \to \mathcal{B}] \quad \Delta \vdash W : A}{\Gamma \uplus \Delta \vdash VW : \mathcal{B}} \qquad \frac{\Gamma \vdash M : \mathcal{A} \quad \Delta \vdash N : \mathcal{B}}{\frac{1}{2}\Gamma \uplus \frac{1}{2}\Delta \vdash M \oplus N : \frac{1}{2}\mathcal{A} + \frac{1}{2}\mathcal{B}}$$

$$\frac{\Gamma \vdash M : \langle p_k A_k \rangle_k \quad (\Delta_k, x : A_k \vdash N : \mathcal{B}_k)_k}{\Gamma \uplus \biguplus_k \Delta_k \vdash \mathtt{let}\ x = M\ \mathtt{in}\ N : \sum_k p_k \mathcal{B}_k}$$

$$\frac{\Gamma \vdash V : \mathcal{A}}{\Gamma \vdash V : \langle \mathcal{A} \rangle} \qquad \frac{\{\Gamma_i \vdash V : \tau_i\}_i}{\biguplus \Gamma_i \vdash V : [q_i.\tau_i]_i}$$

Typing Rules [DLFR2021]

$$\frac{}{x : A \overset{0}{\vdash} x : A} \qquad \frac{}{\overset{0}{\vdash} M : \mathbf{0}} \qquad \frac{\Gamma, x : A \overset{w}{\vdash} M : \mathcal{B}}{\Gamma \overset{w+1}{\vdash} \lambda x.M : A \to \mathcal{B}}$$

$$\frac{\Gamma \overset{w}{\vdash} V : [A \to \mathcal{B}] \quad \Delta \overset{v}{\vdash} W : A}{\Gamma \uplus \Delta \overset{w+v}{\vdash} VW : \mathcal{B}} \qquad \frac{\Gamma \overset{w}{\vdash} M : \mathcal{A} \quad \Delta \overset{v}{\vdash} N : \mathcal{B}}{\frac{1}{2}\Gamma \uplus \frac{1}{2}\Delta \overset{\frac{1}{2}w+\frac{1}{2}v+1}{\vdash} M \oplus N : \frac{1}{2}\mathcal{A} + \frac{1}{2}\mathcal{B}}$$

$$\frac{\Gamma \overset{w}{\vdash} M : \langle p_k A_k \rangle_k \quad (\Delta_k, x : A_k \overset{v_k}{\vdash} N : \mathcal{B}_k)_k}{\Gamma \uplus \biguplus_k \Delta_k \overset{w+\sum p_k v_k+1}{\vdash} \ldots}$$

Typing judgements are attributed a *weight*.

$$\frac{\Gamma \overset{w}{\vdash} V : \mathcal{A}}{\Gamma \overset{w}{\vdash} V : \langle \mathcal{A} \rangle} \qquad \frac{\{\Gamma_i \overset{w_i}{\vdash} V : \tau_i\}_i}{\biguplus \Gamma_i \overset{\sum q_i w_i}{\vdash} V : [q_i.\tau_i]_i}$$

Typing Rules [DLFR2021]

$$\frac{}{x : A \overset{0}{\vdash} x : A}$$

$$\frac{}{\overset{0}{\vdash} M \ \ldots}$$

$$\frac{\Gamma, x : A \overset{w}{\vdash} M : \mathcal{B}}{\ldots \overset{w+1}{\vdash} \ldots}$$

$$\frac{\Gamma \overset{w}{\vdash} V : [A \to \mathcal{B}] \quad \Delta \overset{v}{\vdash} W : A}{\Gamma \uplus \Delta \overset{w+v}{\vdash} VW : \mathcal{B}}$$

$$\ldots \quad \tfrac{1}{2}\mathcal{B}$$

- The body N needs to be typed several times.
- The resulting type is obtained as a mixture of the various types of N.

$$\frac{\Gamma \overset{w}{\vdash} M : \langle p_k A_k \rangle_k \quad (\Delta_k, x : A_k \overset{v_k}{\vdash} N : \mathcal{B}_k)_k}{\Gamma \uplus \biguplus_k \Delta_k \overset{w + \sum p_k v_k + 1}{\vdash} \texttt{let } x = M \texttt{ in } N : \sum_k p_k \mathcal{B}_k}$$

$$\frac{\Gamma \overset{w}{\vdash} V : \mathcal{A}}{\Gamma \overset{w}{\vdash} V : \langle \mathcal{A} \rangle}$$

$$\frac{\{\Gamma_i \overset{w_i}{\vdash} V : \tau_i\}_i}{\biguplus \Gamma_i \overset{\sum q_i w_i}{\vdash} V : [q_i . \tau_i]_i}$$

An AST Example

$$M = NN, \text{ where } N = \lambda x.xx \oplus I$$

An AST Example

$$M = NN, \text{ where } N = \lambda x.xx \oplus I$$

$$\vdash N : \left([] \to \left\langle \frac{1}{2}[] \right\rangle\right) = \tau_1 \qquad \vdash N : [] \qquad \vdash^{2} M : \left\langle \frac{1}{2}[] \right\rangle$$

An AST Example

$$M = NN, \text{ where } N = \lambda x.xx \oplus I$$

$\vdash N : \left([] \to \left\langle \frac{1}{2}[] \right\rangle\right) = \tau_1$

$\vdash N : []$

$\vdash^2 M : \left\langle \frac{1}{2}[] \right\rangle$

$\vdash N : \left(\left[\frac{1}{2}.\tau_1\right] \to \left\langle \frac{1}{2}[], \frac{1}{4}[] \right\rangle\right) = \tau_2$

$\vdash N : \left[\frac{1}{2}.\tau_1\right]$

$\vdash^3 M : \left\langle \frac{1}{2}[], \frac{1}{4}[] \right\rangle$

An AST Example

$$M = NN, \text{ where } N = \lambda x.xx \oplus I$$

$$\vdash N : \left([] \to \left\langle \frac{1}{2}[] \right\rangle\right) = \tau_1 \qquad \vdash N : [] \qquad \vdash^{2} M : \left\langle \frac{1}{2}[] \right\rangle$$

$$\vdash N : \left(\left[\frac{1}{2}.\tau_1\right] \to \left\langle \frac{1}{2}[], \frac{1}{4}[] \right\rangle\right) = \tau_2 \qquad \vdash N : \left[\frac{1}{2}.\tau_1\right] \qquad \vdash^{3} M : \left\langle \frac{1}{2}[], \frac{1}{4}[] \right\rangle$$

$$\vdots$$

$$\vdash N : \left(\left[\frac{1}{2^i}.\tau_i\right]_{i<n} \to \left\langle \frac{1}{2^i}[] \right\rangle_{i\leq n}\right) = \tau_n \qquad \vdash N : \left[\frac{1}{2^i}.\tau_i\right]_{i<n} \qquad \vdash^{4-\frac{1}{2^{n-2}}} M : \left\langle \frac{1}{2^i}[] \right\rangle_{i\leq n}$$

A not-AST Example

$$\Omega = \Delta\Delta, \ \text{where } \Delta = \lambda x.xx$$

$$\vdash \Delta : \langle [] \to \langle\rangle \rangle = \rho_1 \qquad \vdash \Delta : [] \qquad \vdash^{1} \Omega : \langle\rangle$$

$$\vdash \Delta : \langle [1.\rho_1] \to \langle\rangle \rangle = \rho_2 \qquad \vdash \Delta : [1.\rho_1] \qquad \vdash^{2} \Omega : \langle\rangle$$

$$\vdots$$

$$\vdash \Delta : \big\langle [1.\rho_i]_{i<n} \to \langle\rangle \big\rangle = \rho_n \qquad \vdash \Delta : [1.\rho_i]_{i<n} \qquad \vdash^{n} \Omega : \langle\rangle$$

AST and PAST, Precisely Characterized

$$\Pr[M\downarrow] = \sup_{\vdash M:\mathcal{A}\in\mathcal{T}} ||\mathcal{A}||$$

$$\mathbb{E}_{\mathsf{Time}(M)} = \sup_{\overset{w}{\vdash} M:\mathbf{0}} w$$

AST and PAST, Precisely Characterized

$$\Pr[M\downarrow] = \sup_{\vdash M:\mathcal{A}\in\mathfrak{T}} ||\mathcal{A}||$$

$$\mathbb{E}_{\mathsf{Time}(M)} = \sup_{\vdash^{w} M:\mathbf{0}} w$$

$\mathfrak{T} = \{\mathcal{A} \mid \mathcal{A} = \langle p_i[]\rangle_i\}$

AST and PAST, Precisely Characterized

$$\Pr[M\downarrow] = \sup_{\vdash M:\mathcal{A}\in\mathcal{T}} ||\mathcal{A}||$$

$$||\langle p_i A_i\rangle_i|| = \sum_i p_i$$

$$\mathbb{E}_{\mathsf{Time}(M)} = \sup_{\overset{w}{\vdash} M:\mathbf{0}} w$$

AST and PAST, Precisely Characterized

$$\Pr[M\downarrow] = \sup ||\mathcal{A}||$$

Why?

- Subject **Reduction**.
  - If $\vdash M : \mathcal{A}$ and M rewrites to N_i with probability p_i, then $\vdash N_i : \mathcal{B}_i$ such that $\mathcal{A} = \sum_i p_i \mathcal{B}_i$.
  - This implies that $\Pr[M\downarrow] \geq ||\mathcal{A}||$.
- Subject **Expansion**, which implies that $\Pr[M\downarrow] \leq ||\mathcal{A}||$.

AST and PAST, Precisely Characterized

$$\Pr[M\downarrow] = \sup_{\vdash M:\mathcal{A}\in\mathcal{T}} ||\mathcal{A}||$$

$$\mathbb{E}_{\mathsf{Time}(M)} = \sup_{\overset{w}{\vdash} M:\mathbf{0}} w$$

$\mathbf{0} = \langle\rangle$

AST and PAST, Precisely Characterized

$$\Pr[M\downarrow] = \sup_{\vdash M:\mathcal{A}\in\mathfrak{T}} ||\mathcal{A}||$$

$$\mathbb{E}_{\mathsf{Time}(M)} = \sup_{\overset{w}{\vdash} M:\mathbf{0}} w$$

Why?

As above, but **weighted** versions of reduction and expansion theorems are needed.

Other Approaches

- **Linear Dependent Types** [ADLG2019]
  - Intersection Types are complete, but only for *computations* .
  - In deterministic linear dependent types [DLG2011], one is **relatively complete** for first-order functions.
  - How about probabilism?
    - Monadic types can be made indexed:

$$\mu ::= \{\sigma[i] : p[i]\}_{i \in I}$$

    - Subtyping is coupling-based.
    - Nontrivial examples like `RandomizedQuicksort` and `CouponCollector` can be captured.
    - Unfortunately, relative completeness is hard to achieve.

Other Approaches

- **Linear Dependent Types** [ADLG2019]
  - Intersection Types are complete, but only for *computations* .
  - In deterministic linear dependent types [DLG2011], one is **relatively complete** for first-order functions.
  - How about probabilism?
    - Monadic types can be made indexed:

$$\mu ::= \{\sigma[i] : p[i]\}_{i \in I}$$

    - Subtyping is coupling-based.
    - Nontrivial examples like `RandomizedQuicksort` and `CouponCollector` can be captured.
    - Unfortunately, relative completeness is hard to achieve.
- **Probabilistic Termination in Rewriting** [ADLY2019, Faggian2022, FaggianGuerrieri2022];

Other Approaches

- **Linear Dependent Types** [ADLG2019]
  - Intersection Types are complete, but only for *computations* .
  - In deterministic linear dependent types [DLG2011], one is **relatively complete** for first-order functions.
  - How about probabilism?
    - Monadic types can be made indexed:

$$\mu ::= \{\sigma[i] : p[i]\}_{i \in I}$$

    - Subtyping is coupling-based.
    - Nontrivial examples like `RandomizedQuicksort` and `CouponCollector` can be captured.
    - Unfortunately, relative completeness is hard to achieve.
- **Probabilistic Termination in Rewriting** [ADLY2019, Faggian2022, FaggianGuerrieri2022];
- **Higher-Order Model Checking** [KDLG2020];

Other Approaches

- **Linear Dependent Types** [ADLG2019]
  - Intersection Types are complete, but only for *computations* .
  - In deterministic linear dependent types [DLG2011], one is **relatively complete** for first-order functions.
  - How about probabilism?
    - Monadic types can be made indexed:

$$\mu ::= \{\sigma[i] : p[i]\}_{i \in I}$$

    - Subtyping is coupling-based.
    - Nontrivial examples like `RandomizedQuicksort` and `CouponCollector` can be captured.
    - Unfortunately, relative completeness is hard to achieve.
- **Probabilistic Termination in Rewriting** [ADLY2019, Faggian2022, FaggianGuerrieri2022];
- **Higher-Order Model Checking** [KDLG2020];
- **CPS Expectation Transformers** [ADLB2021] ;

Other Approaches

- **Linear Dependent Types** [ADLG2019]
  - Intersection Types are complete, but only for *computations* .
  - In deterministic linear dependent types [DLG2011], one is **relatively complete** for first-order functions.
  - How about probabilism?
    - Monadic types can be made indexed:

$$\mu ::= \{\sigma[i] : p[i]\}_{i \in I}$$

    - Subtyping is coupling-based.
    - Nontrivial examples like `RandomizedQuicksort` and `CouponCollector` can be captured.
    - Unfortunately, relative completeness is hard to achieve.
- **Probabilistic Termination in Rewriting** [ADLY2019, Faggian2022, FaggianGuerrieri2022];
- **Higher-Order Model Checking** [KDLG2020];
- **CPS Expectation Transformers** [ADLB2021] ;
- **Expectations from Probabilistic Coherent Spaces** [Ehrhard2022];

Other Approaches

- **Linear Dependent Types** [ADLG2019]
  - Intersection Types are complete, but only for *computations* .
  - In deterministic linear dependent types [DLG2011], one is **relatively complete** for first-order functions.
  - How about probabilism?
    - Monadic types can be made indexed:

$$\mu ::= \{\sigma[i] : p[i]\}_{i \in I}$$

    - Subtyping is coupling-based.
    - Nontrivial examples like `RandomizedQuicksort` and `CouponCollector` can be captured.
    - Unfortunately, relative completeness is hard to achieve.
- **Probabilistic Termination in Rewriting** [ADLY2019, Faggian2022, FaggianGuerrieri2022];
- **Higher-Order Model Checking** [KDLG2020];
- **CPS Expectation Transformers** [ADLB2021] ;
- **Expectations from Probabilistic Coherent Spaces** [Ehrhard2022];
- **Curry-Howard Correspondence with Counting Propositional Logic** [ADLP2022].

Other Approaches

- **Linear Dependent Types** [ADLG2019]
  - Intersection Types are complete, but only for *computations* .
  - In deterministic linear dependent types [DLG2011], one is **relatively complete** for first-order functions.
  - How about probabilism?
    - Monadic types can be made indexed:

$$\mu ::= \{\sigma[i] : p[i]\}_{i \in I}$$

    - Subtyping is coupling-based.
    - Nontrivial examples like `RandomizedQuicksort` and `CouponCollector` can be captured.
    - Unfortunately, relative completeness is hard to achieve.
- **Probabilistic Termination in Rewriting** [ADLY2019, Faggian2022, FaggianGuerrieri2022];
- **Higher-Order Model Checking** [KDLG2020];
- **CPS Expectation Transformers** [ADLB2021] ;
- **Expectations from Probabilistic Coherent Spaces** [Ehrhard2022];
- **Curry-Howard Correspondence with Counting Propositional Logic** [ADLP2022].
- ....

Part IV

Relational Reasoning

Examples

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

Examples

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.

$I \oplus \Omega$ vs. I

$I \oplus \Omega$ vs. Ω

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.

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

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

A Labelled Markov Chain for $\Lambda_\oplus$

Terms

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

Terms

Values

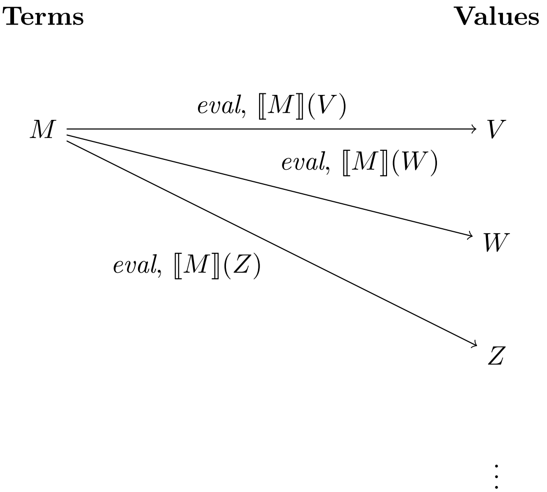
A Labelled Markov Chain for $\Lambda_\oplus$

Terms

Values

M

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



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

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

A Labelled Markov Chain for $\Lambda_\oplus$

Terms **Values**

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

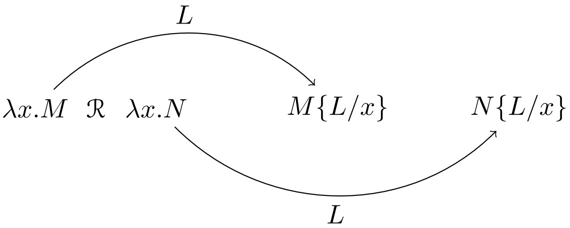
Probabilistic Applicative Bisimulation

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

Probabilistic Applicative Bisimulation

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

Probabilistic Applicative Bisimulation



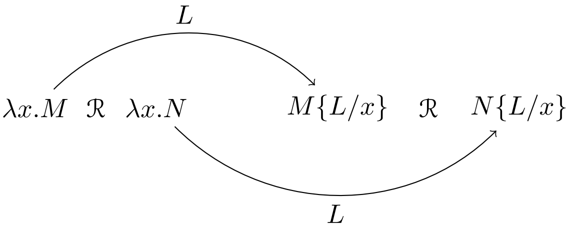
Probabilistic Applicative Bisimulation

$$\lambda x.M \xrightarrow{L} M\{L/x\}$$

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

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

Probabilistic Applicative Bisimulation



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

Probabilistic Applicative Bisimulation

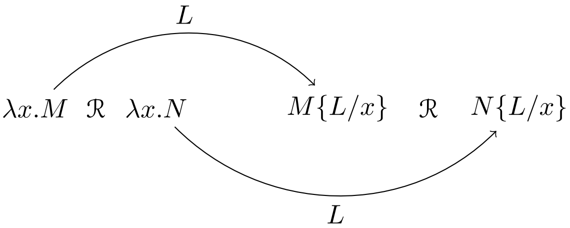
$$\lambda x.M \;\; \mathcal{R} \;\; \lambda x.N \qquad\qquad M\{L/x\} \;\; \mathcal{R} \;\; N\{L/x\}$$

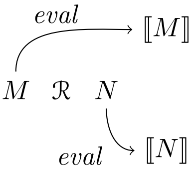
$$\lambda x.M \xrightarrow{L} M\{L/x\} \qquad \lambda x.N \xrightarrow{L} N\{L/x\}$$

$$M \xrightarrow{eval} [\![M]\!]$$

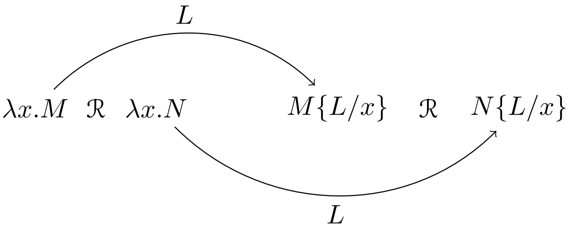
$$M \;\; \mathcal{R} \;\; N$$

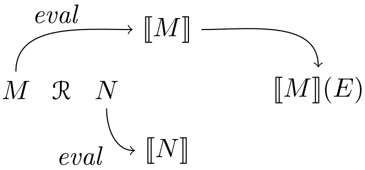
Probabilistic Applicative Bisimulation



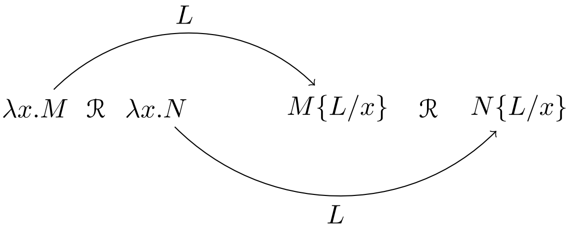


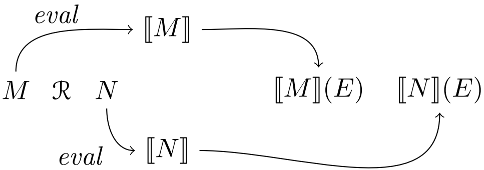
Probabilistic Applicative Bisimulation



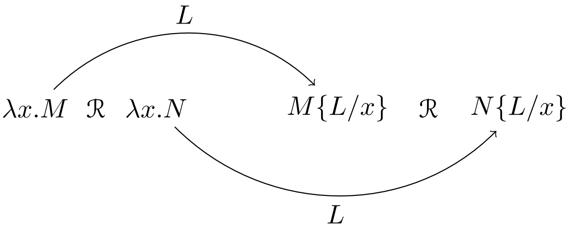


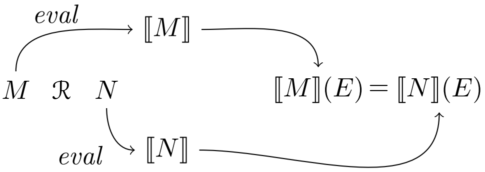
Probabilistic Applicative Bisimulation





Probabilistic Applicative Bisimulation





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:

$$M \sim N \implies C[M] \sim C[N] \implies \sum [\![C[M]]\!] = \sum [\![C[N]]\!]$$
$$\implies M \equiv N.$$

  - 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 $LM \sim LN$.

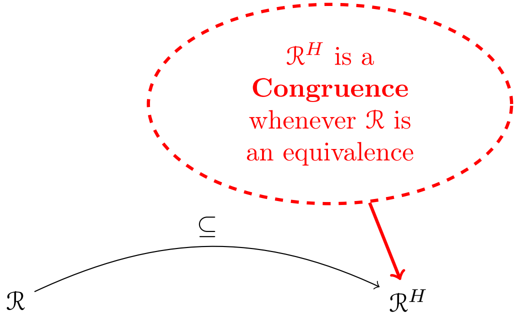
Howe's Technique

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

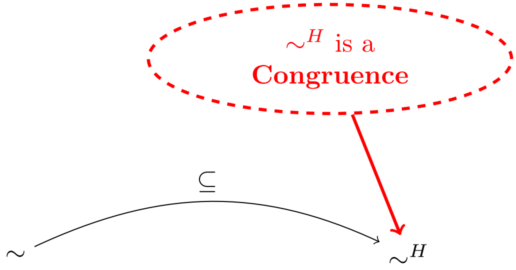
Howe's Technique

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

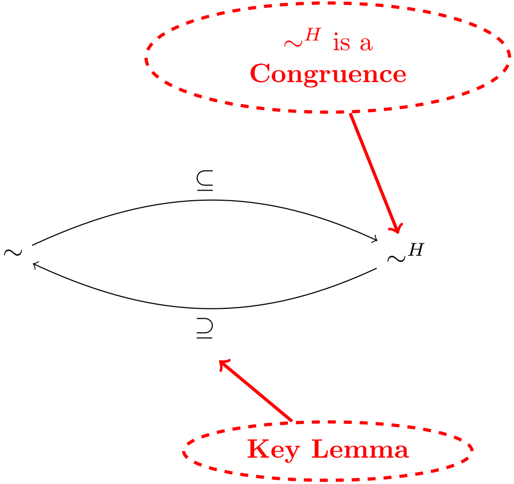
Howe's Technique



Howe's Technique



Howe's Technique



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	$T ::= \omega \mid a \cdot T$
$P\Lambda_\oplus$	$T ::= \omega \mid a \cdot T \mid \langle T, T\rangle$
$N\Lambda_\oplus$	$T ::= \omega \mid a \cdot T \mid \bigwedge_{i\in I} T_i \mid \ldots$

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:

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

- 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 $\Gamma \vdash M$.

[CDL2015]

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

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

- None of the abstract notions of distance δ gives us that $\delta(I, I \oplus \Omega) = 1$.

Context Distance: the General Case [CDL2016]

- None of the abstract notions of distance δ gives us that $\delta(I, I \oplus \Omega) = 1$.
- The underlying LMC **does not** reflect copying.

Context Distance: the General Case [CDL2016]

- None of the abstract notions of distance δ gives us that $\delta(I, I \oplus \Omega) = 1$.
- The underlying LMC **does not** reflect copying.
- **A Tuple LMC**.
  - **Preterms**: $M ::= x \mid \lambda x.M \mid \lambda !x.M \mid MM \mid M \oplus M \mid !M$
  - **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]

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

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

- The underlying LMC **does not** refle... copying.
- **A Tuple LMC.**
  - **Preterms**: $M ::= x \mid \lambda x.M \mid \lambda !x.M \mid MM \mid M \oplus M \mid !M$
  - **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]

- None of the abstract notions of distance δ gives us that $\delta(I, I \oplus \Omega) = 1$.
- The underlying LMC **does not** reflect 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]

- None of the abstract notions of distance δ gives us that $\delta(I, I \oplus \Omega) = 1$.
- The underlying LMC **does not** reflect copying.
- **A Tuple LMC**.
  - **Preterms**: $M ::= x \mid \lambda x.M \mid \lambda !x.M \mid MM \mid M \oplus M \mid !M$
  - **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]

- None of the abstract notions of distance δ gives us that $\delta(I, I \oplus \Omega) = 1$.
- The underlying LMC **does not** reflect copying.
- **A Tuple LMC.**
  - **Preterms**: $M ::= x \mid \lambda x.M \mid \lambda !x.M \mid MM \mid M \oplus M \mid !M$
  - **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**: $\delta^t(!(I \oplus \Omega), !\Omega) = \frac{1}{2}$ $\quad\quad$ $\delta^t(!(I \oplus \Omega), !I) = 1$.
- **Trivialisation** does not hold in general, but becomes true in *strongly normalising* fragments or in presence of *parellel disjuction*.

Other Approaches

- **Logical Relations** [BizjakBirkedal2015];

Other Approaches

- **Logical Relations** [BizjakBirkedal2015];
- **Quantitative Algebras** [MardarePanangadenPlotkin2016];

Other Approaches

- **Logical Relations** [BizjakBirkedal2015];
- **Quantitative Algebras** [MardarePanangadenPlotkin2016];
- **Probabilistic Böhm Trees** [Leventis2018];

Other Approaches

- **Logical Relations** [BizjakBirkedal2015];
- **Quantitative Algebras** [MardarePanangadenPlotkin2016];
- **Probabilistic Böhm Trees** [Leventis2018];
- **Probabilistic Taylor Expansion** [DLLeventis2019];

Other Approaches

- **Logical Relations** [BizjakBirkedal2015];
- **Quantitative Algebras** [MardarePanangadenPlotkin2016];
- **Probabilistic Böhm Trees** [Leventis2018];
- **Probabilistic Taylor Expansion** [DLLeventis2019];
- **(Monadic) Differential Logical Relations** [DLGavazzo2022];

Other Approaches

- **Logical Relations** [BizjakBirkedal2015];
- **Quantitative Algebras** [MardarePanangadenPlotkin2016];
- **Probabilistic Böhm Trees** [Leventis2018];
- **Probabilistic Taylor Expansion** [DLLeventis2019];
- **(Monadic) Differential Logical Relations** [DLGavazzo2022];
- **Denotational Distance from Probabilistic Coherent Spaces** [Ehrhard2022];

Other Approaches

- **Logical Relations** [BizjakBirkedal2015];
- **Quantitative Algebras** [MardarePanangadenPlotkin2016];
- **Probabilistic Böhm Trees** [Leventis2018];
- **Probabilistic Taylor Expansion** [DLLeventis2019];
- **(Monadic) Differential Logical Relations** [DLGavazzo2022];
- **Denotational Distance from Probabilistic Coherent Spaces** [Ehrhard2022];
- **Observational Equivalence and Computational Indistinguishability** [DLGiusti2022].

Other Approaches

- **Logical Relations** [BizjakBirkedal2015];
- **Quantitative Algebras** [MardarePanangadenPlotkin2016];
- **Probabilistic Böhm Trees** [Leventis2018];
- **Probabilistic Taylor Expansion** [DLLeventis2019];
- **(Monadic) Differential Logical Relations** [DLGavazzo2022];
- **Denotational Distance from Probabilistic Coherent Spaces** [Ehrhard2022];
- **Observational Equivalence and Computational Indistinguishability** [DLGiusti2022].
- ...

Wrapping Up

- Some of the techniques for termination, complexity, and relational analysis of deterministic higher-order programs **can be adapted** to the probabilistic setting.
- This is however **not trivial**, since termination and program equivalence have a different, more subtle, nature

Wrapping Up

- Some of the techniques for termination, complexity, and relational analysis of deterministic higher-order programs **can be adapted** to the probabilistic setting.
- This is however **not trivial**, since termination and program equivalence have a different, more subtle, nature

Thank you! Questions?