On Randomization in (Higher-Order) Programming

Part IV

Ugo Dal Lago

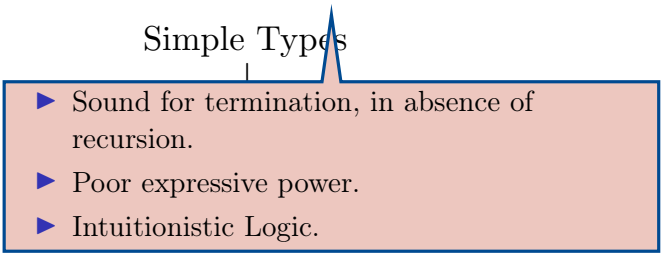
Escuela de Ciencias Informáticas, Buenos Aires, July 2023

The Landscape: *Type* Theory

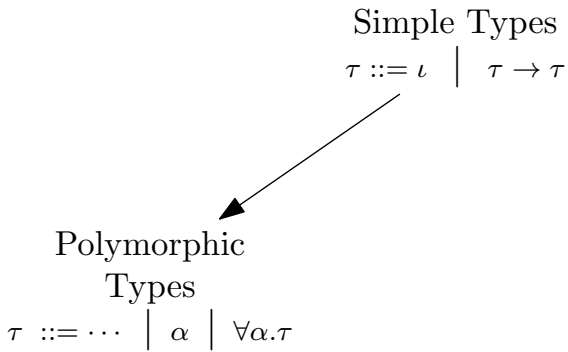
Simple Types

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

The Landscape: *Type* Theory



The Landscape: *Type* Theory



The Landscape: *Type* Theory

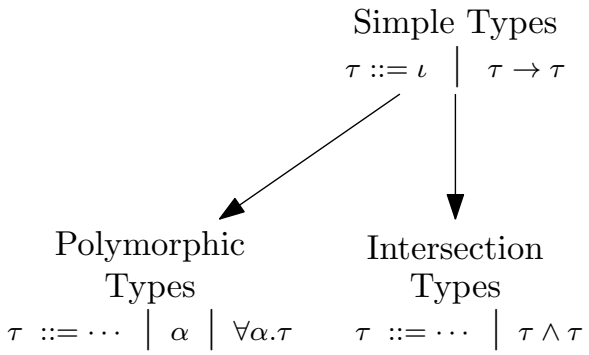
- Second-order Intuistionistic Logic.
- Very expressive, extensionally.
- Still poor, intensionally.

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

Polymorphic Types

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

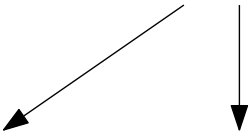
The Landscape: *Type* Theory



The Landscape: *Type* Theory

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

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



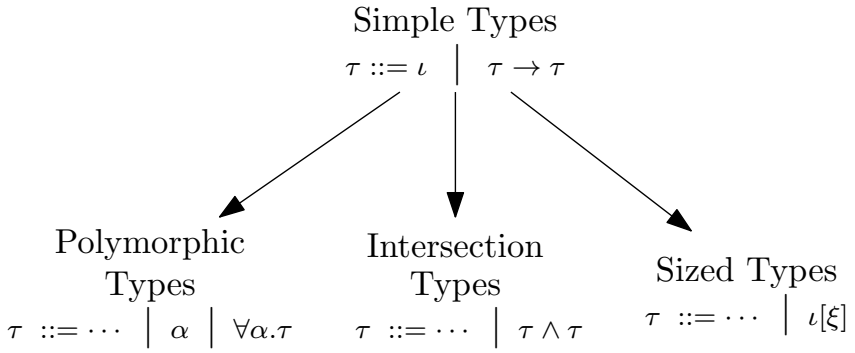
Polymorphic Types

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

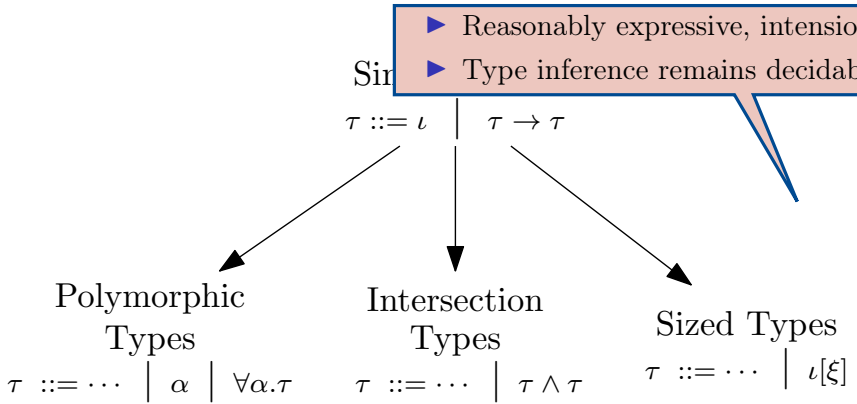
Intersection Types

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

The Landscape: *Type* Theory



The Landscape: *Type* Theory



Determinism

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

The Landscape: *Recursion* Theory

Determinism	**Probabilism**
$M\overline{s} \to^* N_s$	$\langle M\overline{s} \rangle = \mathcal{D}_s$

The Landscape: *Recursion* Theory

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

Determinism	**Probabilism**
$M\overline{s} \to^* N_s$	$\langle M\overline{s} \rangle = \mathcal{D}_s$

The Landscape: *Recursion* Theory

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

The Landscape: *Recursion* Theory

Undecidable;
Σ_1^0-complete.

Probabilism

$\langle M\bar{s} \rangle = \mathcal{D}_s$

Termination

$\exists N_s \in NF$

The Landscape: *Recursion* Theory

	Determinism	**Probabilism**
	$M\overline{s} \to^* N_s$	$\langle M\overline{s} \rangle = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	$\sum \mathcal{D}_s = 1$

The Landscape: *Recursion* Theory

Almost-Sure Termination
Π^0_2-complete.

	Determinism	**Probabilism**
	$M\overline{s} \to^* N_s$	$\langle M\overline{s} \rangle = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	$\sum \mathcal{D}_s = 1$

The Landscape: *Recursion* Theory

	Determinism	**Probabilism**
	$M\overline{s} \to^* N_s$	$\langle M\overline{s} \rangle = \mathcal{D}_s$
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**
	Π^0_2-complete.	
	$M\bar{s} \to N_s$	$\langle M\bar{s} \rangle = \mathcal{D}_s$
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**
	$M\overline{s} \to^* N_s$	$\langle M\overline{s} \rangle = \mathcal{D}_s$
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**
	$M\overline{s} \to^* N_s$	$\langle M\overline{s} \rangle = \mathcal{D}_s$
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.

The Landscape: *Recursion* Theory

	Determinism	**Probabilism**
	$M\overline{s} \to^* N_s$	$\langle M\overline{s} \rangle = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	$\sum \mathcal{D}_s = 1$
Uniform Termination	$\forall s. \exists N_s \in NF$	$\forall s. \sum \mathcal{D}_s = 1$
Positive Termination		$ExLen(M\overline{s}) < +\infty$

The Landscape: *Recursion* Theory

	Determinism	**Probabilism**
	$M\overline{s} \to^* N_s$	$\langle M\overline{s} \rangle = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	Σ_2^0-complete.
Uniform Termination	$\forall s. \exists N_s \in NF$	$\forall s. \sum \mathcal{D}_s = 1$
Positive Termination		$ExLen(M\overline{s}) < +\infty$

The Landscape: *Recursion* Theory

	Determinism	**Probabilism**
	$M\overline{s} \to^* N_s$	$\langle M\overline{s} \rangle = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	$\sum \mathcal{D}_s = 1$
Uniform Termination	$\forall s. \exists N_s \in NF$	$\forall s. \sum \mathcal{D}_s = 1$
Positive Termination		$ExLen(M\overline{s}) < +\infty$
Uniform Positive Termination		$\forall s. ExLen(M\overline{s}) < +\infty$

The Landscape: *Recursion* Theory

	Determinism	**Probabilism**
	$M\overline{s} \to^* N_s$	$\langle M\overline{s} \rangle = \mathcal{D}_s$
Termination	$\exists N_s \in NF$	$\sum \mathcal{D}_s = 1$
Uniform Termination	$\forall s. \exists N_s \in NF$	$\forall s. \sum \mathcal{D}_s = 1$
Positive Termination		$ExLen(M\overline{s}) < +\infty$
Uniform Positive Termination		$\forall s. ExLen(M\overline{s}) < +\infty$

Π^0_3-complete.

Section 1

Sized Types

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 $\mathbb{RCT}_\tau$.
- 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?
- All the termination properties are **lost**, for very good reasons.

Deterministic Sized Types

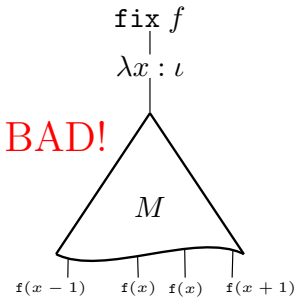
- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - But useless as a programming language.
- What if we endow it with **full recursion** as a `fix` binder?
- All the termination properties are **lost**, for very good reasons.
- Is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

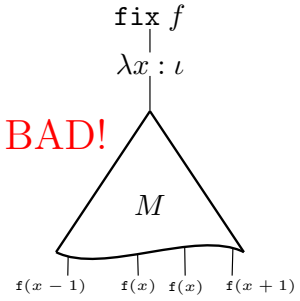
Deterministic Sized Types

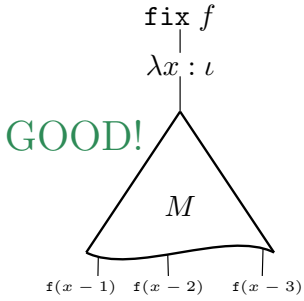
- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - But useless as a programming language.
- What if we endow it with **full recursion** as a `fix` binder?
- All the termination properties are **lost**, for very good reasons.
- Is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

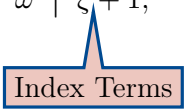
- **Types**.

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

Deterministic Sized Types, Technically

- **Types**.

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



Deterministic Sized Types, Technically

- **Types**.

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

- **Typing Fixpoints**.

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

Deterministic Sized Types, Technically

- **Types**.

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

- **Typing Fixpoints**.

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

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

Deterministic Sized Types, Technically

- **Types**.

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

- **Typing Fixpoints**.

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

- **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 $\mathbb{RCT}^{\theta}_{\tau}$.
- θ is an environment for index variables.
- Proof of reducibility for $\texttt{fix}\, x.V$ is rather delicate.

- 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**.

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

- **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*.

Almost-Sure Termination

- **Examples**:

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

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

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

Almost-Sure Termination

- **Examples**:

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

Unbiased Random Walk

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

Almost-Sure Termination

- **Examples**:

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

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

Unbiased Random Wall

Biased Randomn Walk

Almost-Sure Termination

- **Examples**:

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

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

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

- **Non-Examples**:

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

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

Almost-Sure Termination

- **Examples**:

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

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

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

- **Non-Examples**:

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

  $\texttt{fix}\ f.\lambda x.$ Unbiased Random Walk, with **two** upward calls. $\texttt{else}\ \star;$

Almost-Sure Termination

- **Examples**:

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

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

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

- **Non-Examples**:

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

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

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

Biased Random Walk, the "wrong" way.

Almost-Sure Termination

- **Examples**:

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

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

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

- **Non-Examples**:

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

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

- 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*.
  - $\delta : Q \to \mathbf{D}(Q \times \{-1, 0, 1\})$.
- They are a very special form of One-Counter Markov Decision Processeses [BBEK2011].
  - Everything is purely deterministic.
  - 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*.
  - $\delta : Q \to \mathbf{D}(Q \times \{-1, 0, 1\})$.
- They are a very special form of One-Counter Markov Decision Processeses [BBEK2011].
  - Everything is purely deterministic.
  - 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 siz
  recursive structure by a OCBMC.
- **Judgments**.

A **distribution type**, i.e., a finite distribution of types.

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

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**.

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

Probabilistic Sized Types [DLGrellois2017]

- **Basic Idea**: craft a sized-ty
  recursive structure by a OCB
- **Judgments**.

Form σ, one can build a OCBMC:

- σ is a distribution type.
- It keeps track of the probability of each recursive call.

- **Typing Fixpoints**.

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

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**.

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

- **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**.

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

- **Typing Probabilistic Choice**

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

- **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**.

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

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

$$\mathbb{RCT}_\tau^{\theta,p} = \bigcap_{q<p} \mathbb{RCT}_\tau^{\theta,q}$$

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

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

Section 2

Intersection Types

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 *cannot* 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 *cannot* be typed:

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

- **Types**

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

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?
- Very simple examples of normalizing terms which *cannot* be typed:

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

- **Types**

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

- **Typing Rules: Examples**

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

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?
- Very simple examples of normalizing terms which *cannot* be typed:

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

- **Types**

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

- **Typing Rules: Examples**

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

- **Termination**
  - Again by reducibility.

Deterministic Intersection Types

- **Question**: what are simple types *missing* as a way to precisely capture *termination*?
- Very simple examples of normalizing terms which *cannot* be typed:

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

- **Types**

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

- **Typing Rules: Examples**

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

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

Oracle Intersection Types [BreuvartDL2018]

- 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 [BreuvartDL2018]

- 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 [BreuvartDL2018]

- 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**

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

- **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 [BreuvartDL2018]

- 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**

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

- **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 [BreuvartDL2018]

- 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**

$$\tau ::= \star \mid A \to s \cdot B \mid \dots \quad \dots \subseteq \{0,1\}^*$$

$$\sum \langle M \rangle = \sum_{\vdash M : s \cdot \star} 2^{-|s|}$$

- **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 [BreuvartDL2018]

- 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**

$$\tau ::= \star \mid A \to s \cdot B \mid \bigwedge_{i \in I} \{A_i \to s_i \cdot B_i\} \quad s \in \{0,1\}^*$$

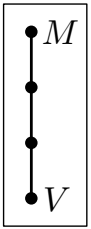
- **Typing Rules: Examples**

$$\frac{\Gamma \vdash M : r \cdot \{A \to s \cdot B\} \quad \Gamma \vdash N : q \cdot A}{\Gamma \vdash MN : rq \cdot B}$$

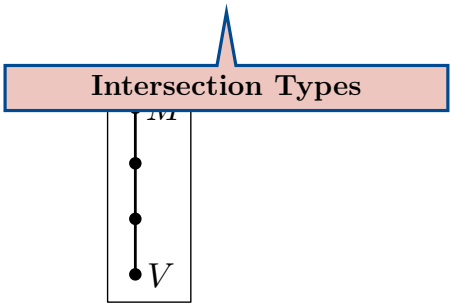
$$\sum \langle M \rangle = \sum_{\vdash M : s \cdot \star} 2^{-|s|}$$

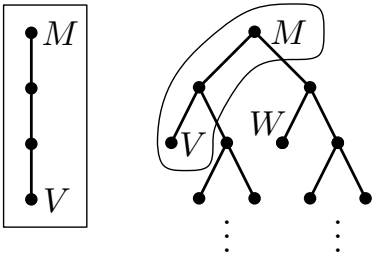
This is **unavoidable**, due to recursion theory.

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

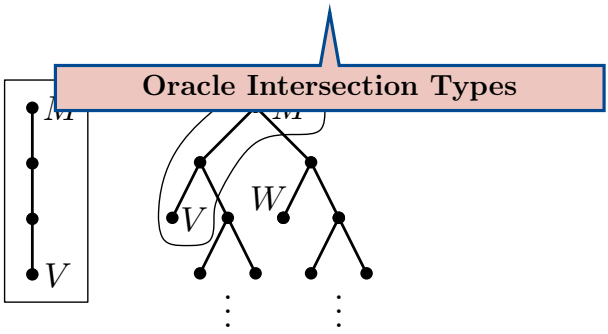

M
V

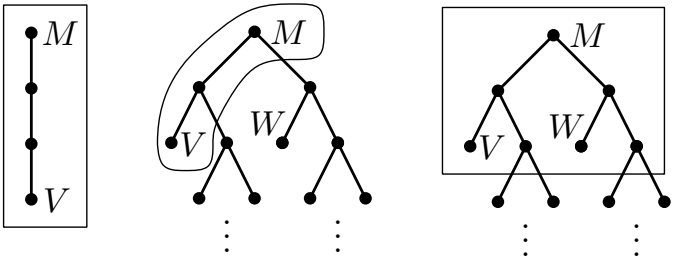
Intersection Types and Computations




M
V
M
W
V

Intersection Types and Computations

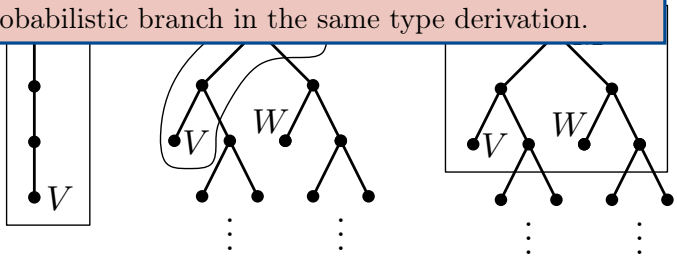



M
V
M
V
W
M
V
W

Intersecti

Monadic Intersection Types [BDL2018,DLFR2021]

- 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.



Thank You!

Questions?