On Type Systems For Probabilistic Termination

Ugo Dal Lago
(Based on joint work with
Charles Grellois and Flavien Breuvart)

CRECOGI+ELICA+GDRILL Meeting, October 10th 2018

Deterministc vs. Probabilistic Transition Systems

$$M$$

Deterministc vs. Probabilistic Transition Systems

$$M \longrightarrow N$$

Deterministc vs. Probabilistic Transition Systems

$$M \longrightarrow N \longrightarrow L$$

Deterministc vs. Probabilistic Transition Systems

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

Deterministc vs. Probabilistic Transition Systems

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

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

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

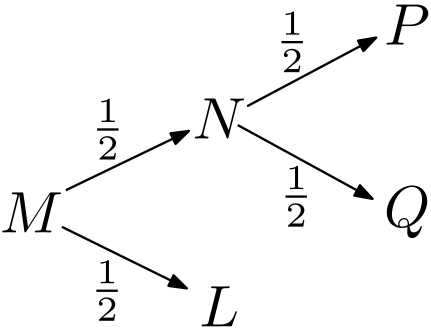
Deterministc vs. Probabilistic Transition Systems

M

Deterministc vs. Probabilistic Transition Systems

$$\begin{tikzcd} & N \\ M \arrow[ur, "\frac{1}{2}"] \arrow[dr, "\frac{1}{2}"'] & \\ & L \end{tikzcd}$$

Deterministc vs. Probabilistic Transition Systems



Deterministc vs. Probabilistic Transition Systems

$$
\begin{tikzcd}
& & P \\
& N \arrow[ur, "\frac{1}{2}"] \arrow[dr, "\frac{1}{2}"'] & \\
M \arrow[ur, "\frac{1}{2}"] \arrow[dr, "\frac{1}{2}"'] & & Q \\
& L &
\end{tikzcd}
$$

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

$$\longrightarrow : \mathcal{A} \rightharpoonup \mathsf{Dist}(\mathcal{A})$$

The Landscape: *Type* Theory

Simple Types

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

The Landscape: *Type* Theory

Simple Types

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

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

The Landscape: *Type* Theory

Simple Types

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

Polymorphic Types

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

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

ypes

$\tau \to \tau$

Polymorphic Types

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

The Landscape: *Type* Theory

Simple Types

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

Polymorphic Types

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

Intersection Types

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

The Landscape: *Type* Theory

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

Polymorphic Types

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

Intersection Types

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

The Landscape: *Type* Theory

Simple Types

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

Polymorphic Types

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

Intersection Types

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

Sized Types

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

The Landscape: *Type* Theory

- Reasonably expressive, intensionally.
- Type inference remains decidable

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

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

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

The Landscape: *Recursion* Theory

Determinism

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

The Landscape: *Recursion* Theory

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

The Landscape: *Recursion* Theory

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

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

The Landscape: *Recursion* Theory

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

Undecidable;
Σ^0_1-complete.

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

The Landscape: *Recursion* Theory

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

Almost-Sure Termination
Π^0_2-complete.

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

The Landscape: *Recursion* Theory

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

Π^0_2-complete.

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

The Landscape: *Recursion* Theory

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

The Landscape: *Recursion* Theory

Π^0_2-complete.

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

Deterministic Sized Types

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

Deterministic Sized Types

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

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

Deterministic Sized Types

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

Deterministic Sized Types

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

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

Deterministic Sized Types

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

Deterministic Sized Types

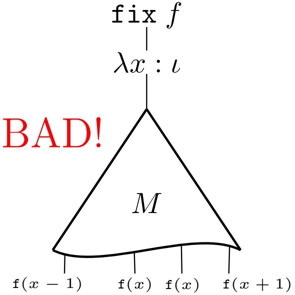
- Pure λ-calculus with simple types is terminating.
  - This can be proved in many ways, including by **reducibility**.
  - But useless as a programming language.
- What if we endow it with **full recursion** as a `fix` binder?
- All the termination properties are **lost**, for very good reasons.
- Is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

Deterministic Sized Types

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

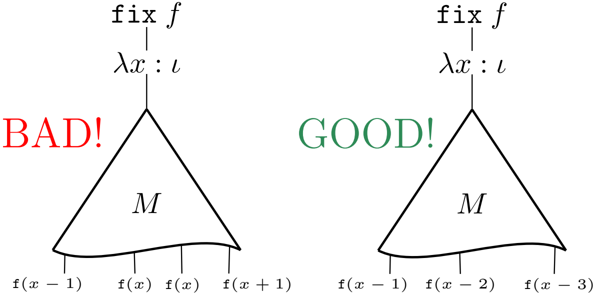
Deterministic Sized Types

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



Deterministic Sized Types

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



Deterministic Sized Types, Technically

- **Types**.

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

Deterministic Sized Types, Technically

- **Types.**

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

Index Terms

Deterministic Sized Types, Technically

- **Types**.

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

- **Typing Fixpoints**.

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

Deterministic Sized Types, Technically

- **Types**.

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

- **Typing Fixpoints**.

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

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

Deterministic Sized Types, Technically

- **Types**.

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

- **Typing Fixpoints**.

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

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

- **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 $\texttt{fix}\, x.M$ is rather delicate.

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

Deterministic Sized Types, Technically

- **Types**.

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

- **Typing Fixpoints**.

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

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

Probabilistic Termination

- **Examples**:

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

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

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

Probabilistic Termination

- **Examples**:

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

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

  Unbiased Random Walk

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

Probabilistic Termination

- **Examples**:

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

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

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

Unbiased Random Walk

Biased Randomn Walk

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**:

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

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

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

- **Non-Examples**:

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

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

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

Biased Random Walk, the “wrong” way.

Probabilistic Termination

- **Examples**:

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

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

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

- **Non-Examples**:

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

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

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

One-Counter Blind Markov Chains

- They are automata of the form (Q, δ) where
  - Q is a finite set of *states*.
  - $\delta : Q \to \mathsf{Dist}(Q \times \{-1, 0, 1\})$.
- They are a very special form of One-Counter Markov Decision Processeses [BBEK2011].
  - 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*.
  - $\delta : Q \to \mathsf{Dist}(Q \times \{-1, 0, 1\})$.
- 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 sized-type system in such a way as to mimick the recursive structure by a OCBMC.
- **Judgments**.

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

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

Probabilistic Sized Types [DLGrellois2017]

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

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

- **Typing Fixpoints**.

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

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**.

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

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**.

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

- **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**.

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

- **Typing Probabilistic Choice**

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

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

Probabilistic Sized Types [DLGrellois2017]

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

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

- **Typing Fixpoints**.

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

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

ates.

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

Part I

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**

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

Deterministic Intersection Types

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

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

- **Types**

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

- **Typing Rules: Examples**

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

Deterministic Intersection Types

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

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

- **Types**

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

- **Typing Rules: Examples**

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

- **Termination**
  - Again by reducibility.

Deterministic Intersection Types

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

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

- **Types**

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

- **Typing Rules: Examples**

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

- **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**

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

- **Typing Rules: Examples**

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

Oracle Intersection Types [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**

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

- **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**

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

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

- **Ty**

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

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

Oracle Intersection Types [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**

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

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

- **Ty**

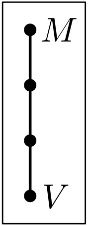
This is **unavoidable**, due to recursion theory.

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

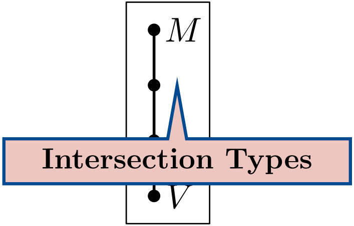
$$\Gamma \vdash M \oplus N : 0s \cdot A \qquad \Gamma \vdash MN : (rqs) \cdot B$$

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

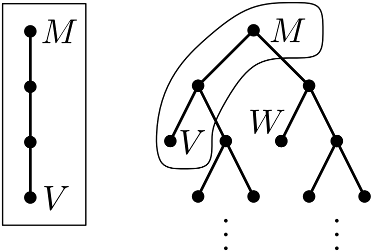
Intersection Types and Computations

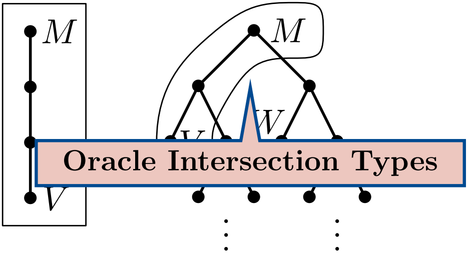


Intersection Types and Computations

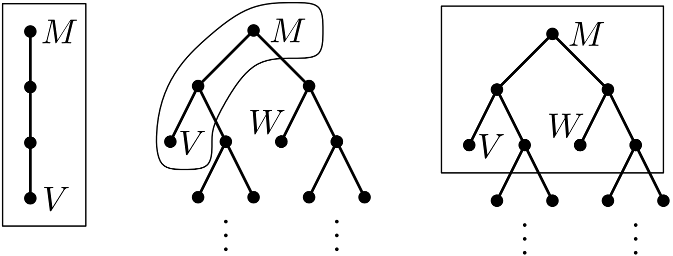


Intersection Types and Computations




M
V
M
V
W
Oracle Intersection Types

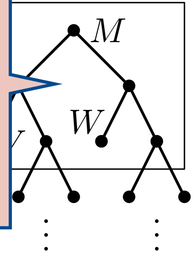
Intersection Types and Computations



Intersection Types and Computations

Monadic Intersection Types [BDL2018]

- 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.



Ongoing and Future Work

- **Non-Idempotent Intersection Types**
  - Monadic and Oracle Intersection Types are idempotent.

Ongoing and Future Work

- **Non-Idempotent Intersection Types**
  - Monadic and Oracle Intersection Types are idempotent.
  - Conjecture:

$$\mathsf{IDEMP} : AST = \mathsf{NONIDEMP} : PAST$$

Ongoing and Future Work

- **Non-Idempotent Intersection Types**
  - Monadic and Oracle Intersection Types are idempotent.
  - Conjecture:

$$\mathsf{IDEMP} : AST = \mathsf{NONIDEMP} : PAST$$

- **Linear Dependent Types**
  - Intersection Types are complete, but only for computations.
  - In linear dependent types [DLG2011], one is (relatively complete) for deterministic first-order functions.

Ongoing and Future Work

- **Non-Idempotent Intersection Types**
  - Monadic and Oracle Intersection Types are idempotent.
  - Conjecture:

$$\mathsf{IDEMP} : AST = \mathsf{NONIDEMP} : PAST$$

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

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

    - Subtyping is coupling-based.

Thank You!

Questions?