Non-Laziness in Implicit Computational Complexity and Probabilistic λ-calculus

Final PhD defense

Gianluca Curzi

Università di Torino
Dipartimento di Informatica

June 12 2020

ABSTRACT

Riassunto della Tesi

La tesi esplora i vantaggi della “non-laziness” sia nella Complessità Computazionale Implicita che nella computazione probabilistica.

Nella **prima parte** analizziamo i meccanismi di cancellazione e duplicazione lineare e introduciamo i sistemi di assegnazione di tipi LEM (Linearly Exponential Multiplicative Type Assignment) and LAM (Linearly Additive Multiplicative Type Assignment). Il primo sistema ha regole esponenziali “più deboli” e soddisfa un teorema di cut-elimination cubico, mentre il secondo sistema ha regole additive “più deboli”, chiamate *additivi lineari*, e soddisfa una normalizzazione lineare forte. La normalizzazione lineare in presenza di regole additive viene quindi recuperata senza introdurre strategie di riduzione **lazy**.

Studiamo inoltre una versione probabilistica degli additivi lineari nel sistema $\mathsf{STA}_{\oplus}$, il quale permette una caratterizzazione implicita delle le funzioni probabilistiche polinomiali che non dipende dalla strategia di riduzione. Tale caratterizzazione non richiede quindi l’uso di riduzioni **lazy**.

Nella **seconda parte** mostriamo che la relazione di bisimilarità nel lambda calcolo probabilistico e l’equivalenza contestuale coincidono rispetto alla semantica operazionale non-lazy e call-by-name (*full abstraction*). Questo risultato testimonia il potere discriminante della **non-laziness**, in quanto tale corrispondenza fallisce nel caso lazy.

Résumé de Thèse

Cette thèse analyses les avantages de la “non-paresse” dans la Complexité Computationnelle Implicite et dans le lambda calcul probabiliste.

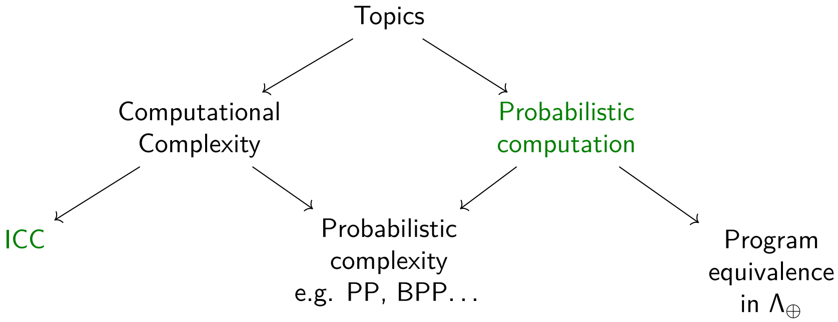
Dans la **première partie** on étude des mécanismes d’effacement et de duplication linéaire, et on introduit les systèmes LEM (Linearly Exponential Multiplicative Type Assignment) et LAM (Linearly Additive Multiplicative Type Assignment). Le premier système a des règles exponentielles “plus faibles” et satisfait l’élimination des coupures en temps cubique. Le second système a des règles additives “plus faibles”, appelées *additifs linéaires*, et satisfait une normalisation linéaire forte. On peut donc retrouver une normalisation linéaire en présence de règles additives sans introduire de stratégies **paresseuses**.

On étude aussi une version probabiliste des additifs linéaires dans le système $\mathsf{STA}_{\oplus}$, qui permet une caractérisation implicite des fonctions probabilistes polynomiales qui ne dépend pas de la stratégie de réduction. Donc, cette caractérisation ne nécessite pas l’utilisation de réductions **paresseuses**.

Dans la **deuxième partie** on montre que la relation de bisimilarité dans le lambda calcul probabiliste correspond à l’équivalence observationnelle par rapport à la sémantique opérationnelle non-paresseuse d’appel par nom (*full abstraction*). Ce résultat témoigne du pouvoir discriminant de la **non-paresse**, étant donné que cette correspondance manquait dans le cadre paresseux.

INTRODUCTION

A genealogical tree. . .



Topics

Computational Complexity

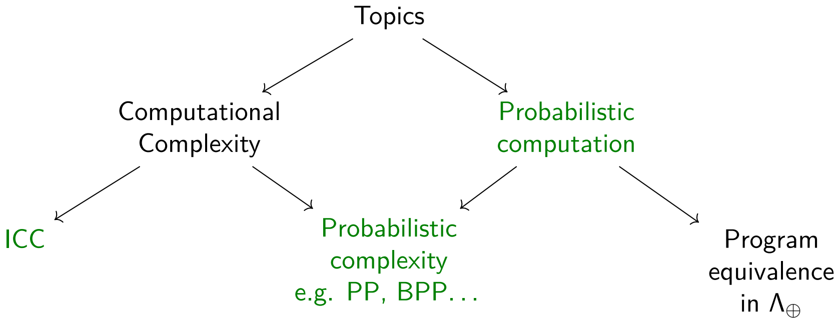
Probabilistic computation

ICC

Probabilistic complexity e.g. PP, BPP. . .

Program equivalence in $\Lambda_{\oplus}$

A genealogical tree...



Topics

Computational Complexity

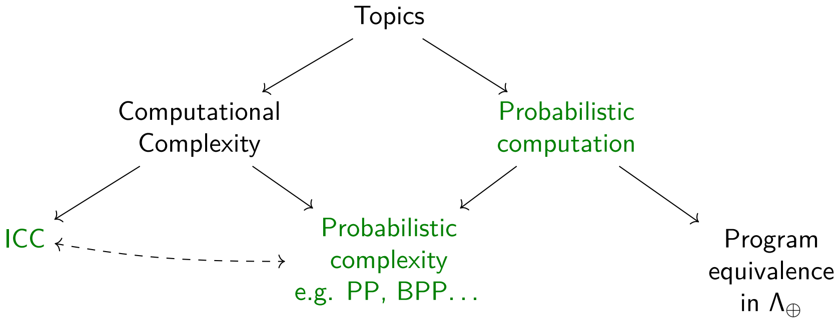
Probabilistic computation

ICC

Probabilistic complexity e.g. PP, BPP...

Program equivalence in $\Lambda_\oplus$

A genealogical tree. . .



Topics

Computational Complexity

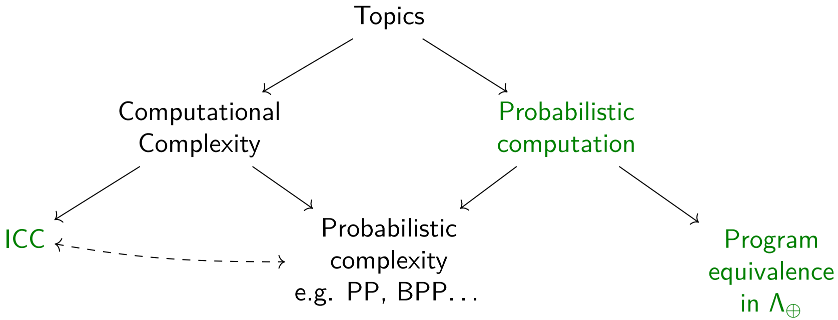
Probabilistic computation

ICC

Probabilistic complexity e.g. PP, BPP. . .

Program equivalence in $\Lambda_{\oplus}$

A genealogical tree…



Topics

Computational Complexity

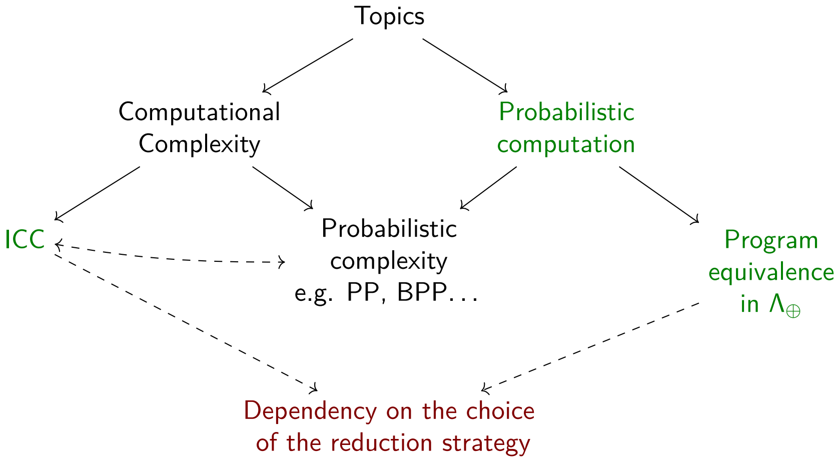
Probabilistic computation

ICC

Probabilistic complexity e.g. PP, BPP…

Program equivalence in $\Lambda_{\oplus}$

A genealogical tree. . .



Topics

Computational Complexity

Probabilistic computation

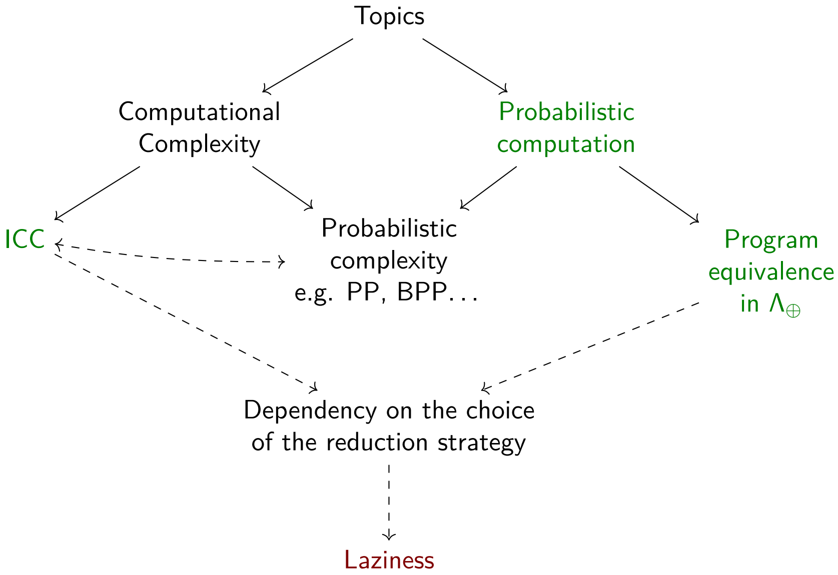
ICC

Probabilistic complexity e.g. PP, BPP. . .

Program equivalence in $\Lambda_\oplus$

Dependency on the choice of the reduction strategy

A genealogical tree. . .



Topics

Computational Complexity

Probabilistic computation

ICC

Probabilistic complexity e.g. PP, BPP. . .

Program equivalence in $\Lambda_{\oplus}$

Dependency on the choice of the reduction strategy

Laziness

Goal of this thesis

Benefits of “non-laziness” in ICC and program equivalence in $\Lambda_\oplus$:

- **(Part I)** System LEM to compactly express linear erasure/duplication in Multiplicative Linear Logic. New tools for ICC: “linear” additive rules?
- **(Part II)**
  - (i) System LAM with *linear additives* has strong linear normalization (no laziness).
  - (ii) System $\mathsf{STA}_\oplus$ with *probabilistic linear additives* captures probabilistic polytime functions in the style of ICC (no laziness).
- **(Part III)** Non-laziness: key step to recover the missing matching between bisimilarity and contextual equivalence in $\Lambda_\oplus^{\mathrm{cbn}}$.

Goal of this thesis

Benefits of "non-laziness" in ICC and program equivalence in $\Lambda_{\oplus}$:

- **(Part I)** System LEM to compactly express linear erasure/duplication in Multiplicative Linear Logic. New tools for ICC: "linear" additive rules?
- **(Part II)**
  - (i) System LAM with *linear additives* has strong linear normalization (no laziness).
  - (ii) System $\mathsf{STA}_{\oplus}$ with *probabilistic linear additives* captures probabilistic polytime functions in the style of ICC (no laziness).
- **(Part III)** Non-laziness: key step to recover the missing matching between bisimilarity and contextual equivalence in $\Lambda_{\oplus}^{\mathrm{cbn}}$.

Goal of this thesis

Benefits of “non-laziness” in ICC and program equivalence in $\Lambda_{\oplus}$:

- **(Part I)** System LEM to compactly express linear erasure/duplication in Multiplicative Linear Logic. New tools for ICC: “linear” additive rules?
- **(Part II)**
  - (i) System LAM with *linear additives* has strong linear normalization (no laziness).
  - (ii) System $\mathsf{STA}_{\oplus}$ with *probabilistic linear additives* captures probabilistic polytime functions in the style of ICC (no laziness).
- **(Part III)** Non-laziness: key step to recover the missing matching between bisimilarity and contextual equivalence in $\Lambda_{\oplus}^{\mathrm{cbn}}$.

PART I

A TYPE ASSIGNMENT OF LINEAR ERASURE AND DUPLICATION

Gianluca Curzi and Luca Roversi. *A type-assignment of linear erasure and duplication*. In *Theoretical Computer Science*, 2020.

Introduction

- IMLL_2 as type assignment for the linear λ-calculus.
- Encoding **boolean circuits** in IMLL_2 [Mairson 03, Mairson&Terui 03].

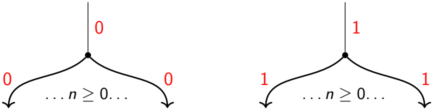
- **Linear erasure/duplication**: a way to "linearly" express *fan-out*.
- **General** linear erasure/duplication:

$$\text{boolean data type} \quad \mapsto \quad \textbf{finite} \text{ data types}$$

- This thesis: system LEM to compactly express linear erasure/duplication of IMLL_2. Cut-elimination? Expressiveness?

Introduction

- IMLL$_2$ as type assignment for the linear λ-calculus.
- Encoding **boolean circuits** in IMLL$_2$ [Mairson 03, Mairson&Terui 03].



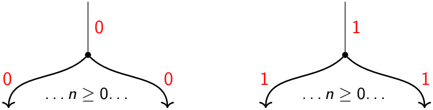
- **Linear erasure/duplication**: a way to "linearly" express *fan-out*.
- **General** linear erasure/duplication:

$$\text{boolean data type} \quad \mapsto \quad \textbf{finite}\ \text{data types}$$

- This thesis: system LEM to compactly express linear erasure/duplication of IMLL$_2$. Cut-elimination? Expressiveness?

Introduction

- IMLL_2 as type assignment for the linear λ-calculus.
- Encoding **boolean circuits** in IMLL_2 [Mairson 03, Mairson&Terui 03].



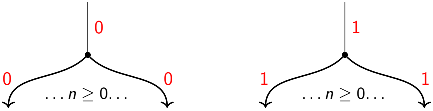
- **Linear erasure/duplication**: a way to "linearly" express *fan-out*.
- **General** linear erasure/duplication:

$$\text{boolean data type} \quad \mapsto \quad \textbf{finite} \text{ data types}$$

- This thesis: system LEM to compactly express linear erasure/duplication of IMLL_2. Cut-elimination? Expressiveness?

Introduction

- IMLL_2 as type assignment for the linear λ-calculus.
- Encoding **boolean circuits** in IMLL_2 [Mairson 03, Mairson&Terui 03].



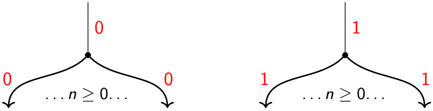
- **Linear erasure/duplication**: a way to "linearly" express *fan-out*.
- **General** linear erasure/duplication:

boolean data type $\mapsto$ **finite** data types

- This thesis: system LEM to compactly express linear erasure/duplication of IMLL_2. Cut-elimination? Expressiveness?

Introduction

- IMLL_2 as type assignment for the linear λ-calculus.
- Encoding **boolean circuits** in IMLL_2 [Mairson 03, Mairson&Terui 03].



- **Linear erasure/duplication**: a way to "linearly" express *fan-out*.
- **General** linear erasure/duplication:

$$\text{boolean data type} \quad \mapsto \quad \textbf{finite} \text{ data types}$$

- This thesis: system LEM to compactly express linear erasure/duplication of IMLL_2. Cut-elimination? Expressiveness?

Linear erasure and duplication in IMLL_2

- "Linear" Booleans:

$$\mathbf{B} \triangleq \forall\alpha.\alpha \multimap \alpha \multimap \alpha \otimes \alpha \qquad \underline{0} \triangleq \lambda x.\lambda y.x \otimes y \qquad \underline{1} \triangleq \lambda x.\lambda y.y \otimes x$$

- Linear erasure by **consumption of data**:

$$\mathtt{E}_{\mathbf{B}} \triangleq \lambda z.\mathtt{let}\ z\mathbf{I}\mathbf{I}\ \mathtt{be}\ x,y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x)$$

- Example: $\mathtt{E}_{\mathbf{B}}\,\underline{0} \triangleq (\lambda z.\mathtt{let}\ z\mathbf{I}\mathbf{I}\ \mathtt{be}\ x,y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x))\,(\lambda x.\lambda y.x \otimes y)$

- Linear duplication by **selection and erasure**:

$$\mathtt{D}_{\mathbf{B}} \triangleq \lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1}))$$

- Example: $\mathtt{D}_{\mathbf{B}}\,\underline{1} \triangleq (\lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1})))(\lambda x.\lambda y.y \otimes x)$

Linear erasure and duplication in IMLL_2

- “Linear” Booleans:

$$\mathbf{B} \triangleq \forall\alpha.\alpha \multimap \alpha \multimap \alpha \otimes \alpha \qquad \underline{0} \triangleq \lambda x.\lambda y.x \otimes y \qquad \underline{1} \triangleq \lambda x.\lambda y.y \otimes x$$

- Linear erasure by **consumption of data**:

$$\mathtt{E}_{\mathbf{B}} \triangleq \lambda z.\mathtt{let}\ z\mathbf{I}\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x)$$

- Example: $\mathtt{E}_{\mathbf{B}}\,\underline{0} \triangleq (\lambda z.\mathtt{let}\ z\mathbf{I}\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x))\,(\lambda x.\lambda y.x \otimes y)$

- Linear duplication by **selection and erasure**:

$$\mathtt{D}_{\mathbf{B}} \triangleq \lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1}))$$

- Example: $\mathtt{D}_{\mathbf{B}}\,\underline{1} \triangleq (\lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1})))(\lambda x.\lambda y.y \otimes x)$

Linear erasure and duplication in IMLL$_2$

- "Linear" Booleans:

$$\mathbf{B} \triangleq \forall\alpha.\alpha \multimap \alpha \multimap \alpha \otimes \alpha \qquad \underline{0} \triangleq \lambda x.\lambda y.x \otimes y \qquad \underline{1} \triangleq \lambda x.\lambda y.y \otimes x$$

- Linear erasure by **consumption of data**:

$$\mathtt{E}_{\mathbf{B}} \triangleq \lambda z.\mathtt{let}\ z\mathbf{I}\mathbf{I}\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x)$$

- Example: $\mathtt{let}\ (\lambda x.\lambda y.x \otimes y)\,\mathbf{I}\mathbf{I}\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x)$

- Linear duplication by **selection and erasure**:

$$\mathtt{D}_{\mathbf{B}} \triangleq \lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1}))$$

- Example: $\mathtt{D}_{\mathbf{B}}\,\underline{1} \triangleq (\lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1})))(\lambda x.\lambda y.y \otimes x)$

Linear erasure and duplication in IMLL_2

- "Linear" Booleans:

$$\mathbf{B} \triangleq \forall\alpha.\alpha \multimap \alpha \multimap \alpha \otimes \alpha \qquad \underline{0} \triangleq \lambda x.\lambda y.{\color{red}x} \otimes {\color{red}y} \qquad \underline{1} \triangleq \lambda x.\lambda y.{\color{red}y} \otimes {\color{red}x}$$

- Linear erasure by **consumption of data**:

$$\mathtt{E}_{\mathbf{B}} \triangleq \lambda z.\mathtt{let}\ z\mathbf{I}\mathbf{I}\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x)$$

- Example: $\mathtt{let}\ {\color{red}\mathbf{I}} \otimes {\color{red}\mathbf{I}}\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x)$

- Linear duplication by **selection and erasure**:

$$\mathtt{D}_{\mathbf{B}} \triangleq \lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1}))$$

- Example: $\mathtt{D}_{\mathbf{B}}\,\underline{1} \triangleq (\lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1})))(\lambda x.\lambda y.y \otimes x)$

Linear erasure and duplication in IMLL$_2$

- "Linear" Booleans:

$$\mathbf{B} \triangleq \forall\alpha.\alpha \multimap \alpha \multimap \alpha \otimes \alpha \qquad \underline{0} \triangleq \lambda x.\lambda y.x \otimes y \qquad \underline{1} \triangleq \lambda x.\lambda y.y \otimes x$$

- Linear erasure by **consumption of data**:

$$\mathtt{E}_{\mathbf{B}} \triangleq \lambda z.\mathtt{let}\ z\mathbf{I}\mathbf{I}\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x)$$

- Example: $\mathbf{I}$

- Linear duplication by **selection and erasure**:

$$\mathtt{D}_{\mathbf{B}} \triangleq \lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1}))$$

- Example: $\mathtt{D}_{\mathbf{B}}\ \underline{1} \triangleq (\lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1})))(\lambda x.\lambda y.y \otimes x)$

Linear erasure and duplication in IMLL$_2$

- “Linear” Booleans:

$$\mathbf{B} \triangleq \forall\alpha.\alpha \multimap \alpha \multimap \alpha \otimes \alpha \qquad \underline{0} \triangleq \lambda x.\lambda y.x \otimes y \qquad \underline{1} \triangleq \lambda x.\lambda y.y \otimes x$$

- Linear erasure by **consumption of data**:

$$\mathtt{E}_{\mathbf{B}} \triangleq \lambda z.\mathtt{let}\ z\mathbf{I}\mathbf{I}\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x)$$

- Example: $\mathbf{I}$

- Linear duplication by **selection and erasure**:

$$\mathtt{D}_{\mathbf{B}} \triangleq \lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1}))$$

- Example: $\mathtt{D}_{\mathbf{B}}\,\underline{1} \triangleq (\lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1})))(\lambda x.\lambda y.y \otimes x)$

Linear erasure and duplication in IMLL_2

- “Linear” Booleans:

$$\mathbf{B} \triangleq \forall\alpha.\alpha \multimap \alpha \multimap \alpha \otimes \alpha \qquad \underline{0} \triangleq \lambda x.\lambda y.x \otimes y \qquad \underline{1} \triangleq \lambda x.\lambda y.y \otimes x$$

- Linear erasure by **consumption of data**:

$$\mathtt{E}_{\mathbf{B}} \triangleq \lambda z.\mathtt{let}\ z\mathbf{I}\mathbf{I}\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x)$$

- Example: $\mathbf{I}$

- Linear duplication by **selection and erasure**:

$$\mathtt{D}_{\mathbf{B}} \triangleq \lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1}))$$

- Example: $\pi_1((\lambda x.\lambda y.y \otimes x)(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1}))$

Linear erasure and duplication in IMLL_2

- "Linear" Booleans:

$$\mathbf{B} \triangleq \forall\alpha.\alpha \multimap \alpha \multimap \alpha \otimes \alpha \qquad \underline{0} \triangleq \lambda x.\lambda y.x \otimes y \qquad \underline{1} \triangleq \lambda x.\lambda y.y \otimes x$$

- Linear erasure by **consumption of data**:

$$\mathtt{E}_{\mathbf{B}} \triangleq \lambda z.\mathtt{let}\ z\mathbf{I}\mathbf{I}\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x)$$

- Example: $\mathbf{I}$

- Linear duplication by **selection and erasure**:

$$\mathtt{D}_{\mathbf{B}} \triangleq \lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1}))$$

- Example: $\pi_1((\underline{1} \otimes \underline{1}) \otimes (\underline{0} \otimes \underline{0}))$

Linear erasure and duplication in IMLL_2

- "Linear" Booleans:

$$\mathbf{B} \triangleq \forall\alpha.\alpha \multimap \alpha \multimap \alpha \otimes \alpha \qquad \underline{0} \triangleq \lambda x.\lambda y.x \otimes y \qquad \underline{1} \triangleq \lambda x.\lambda y.y \otimes x$$

- Linear erasure by **consumption of data**:

$$\mathtt{E}_{\mathbf{B}} \triangleq \lambda z.\mathtt{let}\ z\mathbf{I}\mathbf{I}\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x)$$

- Example: $\mathbf{I}$

- Linear duplication by **selection and erasure**:

$$\mathtt{D}_{\mathbf{B}} \triangleq \lambda z.\pi_1(z(\underline{0} \otimes \underline{0})(\underline{1} \otimes \underline{1}))$$

- Example: $\underline{1} \otimes \underline{1}$

The system LEM (this thesis)

- **Generalization** in IMLL_2: from $\mathbf{B}$ to *closed* Π_1-*types* (no negative $\forall$):

$$\begin{array}{lcl} A \text{ closed } \Pi_1 & \to & \mathtt{E}_A \\ A \text{ closed } \Pi_1 \ + \textbf{inhabited} & \to & \mathtt{D}_A \end{array}$$

- How to express general linear erasure/duplication of IMLL_2?
- LEM = IMLL_2 + "weaker" exponential rules:

$$\frac{x_1 : \downarrow\sigma_1, \ldots, x_n : \downarrow\sigma_n \vdash M : \sigma}{x_1 : \downarrow\sigma_1, \ldots, x_n : \downarrow\sigma_n \vdash M : \downarrow\sigma}\ p \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma, y : \downarrow\sigma \vdash M[y/x] : \tau}\ d$$

$$\frac{\Gamma \vdash M : \tau}{\Gamma, x : \downarrow\sigma \vdash \mathtt{discard}_\sigma\ x\ \mathtt{in}\ M : \tau}\ w \qquad \frac{\Gamma, y : \downarrow\sigma, z : \downarrow\sigma \vdash M : \tau \qquad \vdash V : \sigma}{\Gamma, x : \downarrow\sigma \vdash \mathtt{copy}^V_\sigma\ x\ \mathtt{as}\ y, z\ \mathtt{in}\ M : \tau}\ c$$

σ *closed and lazy* (no negative $\forall$) and V *value* (closed normal linear λ-term).

The system LEM (this thesis)

- **Generalization** in IMLL_2: from $\mathbf{B}$ to *closed* Π_1*-types* (no negative $\forall$):

$$\begin{array}{lcl} A \text{ closed } \Pi_1 & \rightarrow & \mathtt{E}_A \\ A \text{ closed } \Pi_1 \; + \textbf{inhabited} & \rightarrow & \mathtt{D}_A \end{array}$$

- How to express general linear erasure/duplication of IMLL_2?

- LEM $=$ IMLL_2 $+$ “weaker” exponential rules:

$$\frac{x_1 : \downarrow\sigma_1, \ldots, x_n : \downarrow\sigma_n \vdash M : \sigma}{x_1 : \downarrow\sigma_1, \ldots, x_n : \downarrow\sigma_n \vdash M : \downarrow\sigma}\ p \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma, y : \downarrow\sigma \vdash M[y/x] : \tau}\ d$$

$$\frac{\Gamma \vdash M : \tau}{\Gamma, x : \downarrow\sigma \vdash \mathtt{discard}_{\sigma}\ x \ \mathtt{in}\ M : \tau}\ w \qquad \frac{\Gamma, y : \downarrow\sigma, z : \downarrow\sigma \vdash M : \tau \qquad \vdash V : \sigma}{\Gamma, x : \downarrow\sigma \vdash \mathtt{copy}^{V}_{\sigma}\ x \ \mathtt{as}\ y, z \ \mathtt{in}\ M : \tau}\ c$$

σ *closed and lazy* (no negative $\forall$) and V *value* (closed normal linear λ-term).

Cut-elimination for lazy types (this thesis)

- Reduction rules:

$$\begin{aligned} (\lambda x.M)N &\rightarrow M[N/x] \\ \mathtt{discard}_\sigma\ V\ \mathtt{in}\ M &\rightarrow M \\ \mathtt{copy}^{V'}_\sigma\ V\ \mathtt{as}\ x,y\ \mathtt{in}\ M &\rightarrow M[V/x, V/y] \end{aligned}$$

V *value* (closed normal linear λ-term).

- “Weaker” cut-elimination rules:

$$\dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V : \sigma}}{\vdash V : \downarrow\sigma} \quad \dfrac{\Delta, y : \downarrow\sigma, z : \downarrow\sigma \vdash M : \tau \quad \vdash V' : \sigma}{\Delta, x : \downarrow\sigma \vdash \mathtt{copy}^{V'}_\sigma\ x\ \mathtt{as}\ y, z\ \mathtt{in}\ M : \tau}}{\Delta \vdash \mathtt{copy}^{V'}_\sigma\ V\ \mathtt{as}\ y, z\ \mathtt{in}\ M : \tau} \rightsquigarrow \dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V : \sigma}}{\vdash V : \downarrow\sigma} \quad \dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V : \sigma}}{\vdash V : \downarrow\sigma} \quad \Delta, y : \downarrow\sigma, z : \downarrow\sigma \vdash M : \tau}{\Delta, z : \downarrow\sigma \vdash M[V/y] : \tau}}{\Delta \vdash M[V/y, V/z] : \tau}$$

Theorem (Cut-elimination for lazy types)

Any derivation of a lazy type (no negative $\forall$) can be rewritten into a cut-free one by the “weaker” cut-elimination rules in **cubic time.**

Cut-elimination for lazy types (this thesis)

- Reduction rules:

$$\begin{aligned} (\lambda x.M)N &\rightarrow M[N/x] \\ \mathtt{discard}_\sigma\ V \mathtt{\ in\ } M &\rightarrow M \\ \mathtt{copy}^{V'}_\sigma\ V \mathtt{\ as\ } x, y \mathtt{\ in\ } M &\rightarrow M[V/x, V/y] \end{aligned}$$

V *value* (closed normal linear λ-term).

- "Weaker" cut-elimination rules:

$$\dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V : \sigma}}{\vdash V : \downarrow\sigma} \quad \dfrac{\Delta, y : \downarrow\sigma, z : \downarrow\sigma \vdash M : \tau \quad \vdash V' : \sigma}{\Delta, x : \downarrow\sigma \vdash \mathtt{copy}^{V'}_\sigma\ x \mathtt{\ as\ } y, z \mathtt{\ in\ } M : \tau}}{\Delta \vdash \mathtt{copy}^{V'}_\sigma\ V \mathtt{\ as\ } y, z \mathtt{\ in\ } M : \tau} \rightsquigarrow \dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V : \sigma}}{\vdash V : \downarrow\sigma} \quad \dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V : \sigma}}{\vdash V : \downarrow\sigma} \quad \Delta, y : \downarrow\sigma, z : \downarrow\sigma \vdash M : \tau}{\Delta, z : \downarrow\sigma \vdash M[V/y] : \tau}}{\Delta \vdash M[V/y, V/z] : \tau}$$

Theorem (Cut-elimination for lazy types)

Any derivation of a lazy type (no negative $\forall$) can be rewritten into a cut-free one by the "weaker" cut-elimination rules in **cubic time**.

Cut-elimination for lazy types (this thesis)

- Reduction rules:

$$\begin{array}{rcl} (\lambda x.M)N & \to & M[N/x] \\ \texttt{discard}_\sigma\ V\ \texttt{in}\ M & \to & M \\ \texttt{copy}_\sigma^{V'}\ V\ \texttt{as}\ x, y\ \texttt{in}\ M & \to & M[V/x, V/y] \end{array}$$

  V *value* (closed normal linear λ-term).

- "Weaker" cut-elimination rules:

$$\dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V : \sigma}}{\vdash V : \downarrow\sigma} \quad \dfrac{\Delta, y : \downarrow\sigma, z : \downarrow\sigma \vdash M : \tau \quad \vdash V' : \sigma}{\Delta, x : \downarrow\sigma \vdash \texttt{copy}_\sigma^{V'}\ x\ \texttt{as}\ y, z\ \texttt{in}\ M : \tau}}{\Delta \vdash \texttt{copy}_\sigma^{V'}\ V\ \texttt{as}\ y, z\ \texttt{in}\ M : \tau} \quad \rightsquigarrow \quad \dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V : \sigma}}{\vdash V : \downarrow\sigma} \quad \dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V : \sigma}}{\vdash V : \downarrow\sigma} \quad \Delta, y : \downarrow\sigma, z : \downarrow\sigma \vdash M : \tau}{\Delta, z : \downarrow\sigma \vdash M[V/y] : \tau}}{\Delta \vdash M[V/y, V/z] : \tau}$$

Theorem (Cut-elimination for lazy types)

Any derivation of a lazy type (no negative $\forall$) can be rewritten into a cut-free one by the "weaker" cut-elimination rules in **cubic time**.

Exponential compression and applications (this thesis)

- *Translation* $(\_)^{\bullet} : \mathsf{LEM} \to \mathsf{IMLL}_2$:

$$\begin{array}{rcll} \sigma \quad \text{(closed lazy)} & \mapsto & \sigma^{\bullet} & \text{(closed } \Pi_1\text{)} \\ \mathtt{discard}_{\sigma} & \mapsto & \mathtt{E}_{\sigma^{\bullet}} & \\ \mathtt{copy}^{V}_{\sigma} & \mapsto & \mathtt{D}_{\sigma^{\bullet}} & \end{array}$$

- Since $\mathrm{size}(\mathtt{D}_A) \in \mathcal{O}(2^{\mathrm{size}(A)^2})$:

Theorem (Exponential compression)

If $\Gamma \vdash_{\mathsf{LEM}} M : \sigma$ then $\mathrm{size}(M^{\bullet})$ can be **exponential** w.r.t $\mathrm{size}(M)$.

- **Applications**: compact encodings of **boolean circuits** and **natural numbers**.

$$\overline{2} \triangleq \lambda fx.\mathtt{copy}^{!}_{1}\, f \text{ as } f_1, f_2 \text{ in } f_1(f_2\, x)$$

$$\mathtt{succ} \triangleq \lambda nfx.\mathtt{copy}^{!}_{1}\, f \text{ as } f_1, f_2 \text{ in } f_1(nf_2x)$$

$$\mathtt{add} \triangleq \lambda mnfx.\mathtt{copy}^{!}_{1}\, f \text{ as } f_1, f_2 \text{ in } mf_1(nf_2x).$$

Exponential compression and applications (this thesis)

- *Translation* $(\_)^\bullet : \mathsf{LEM} \to \mathsf{IMLL}_2$:

$$\begin{array}{rcll} \sigma \quad \text{(closed lazy)} & \mapsto & \sigma^\bullet & \text{(closed } \Pi_1\text{)} \\ \mathtt{discard}_\sigma & \mapsto & \mathtt{E}_{\sigma^\bullet} & \\ \mathtt{copy}^V_\sigma & \mapsto & \mathtt{D}_{\sigma^\bullet} & \end{array}$$

- Since $\mathrm{size}(\mathtt{D}_A) \in \mathcal{O}(2^{\mathrm{size}(A)^2})$:

Theorem (Exponential compression)

If $\Gamma \vdash_{\mathsf{LEM}} M : \sigma$ then $\mathrm{size}(M^\bullet)$ can be **exponential** w.r.t $\mathrm{size}(M)$.

- **Applications**: compact encodings of **boolean circuits** and **natural numbers**.

$$\overline{2} \triangleq \lambda fx.\mathrm{copy}^!_1\ f \text{ as } f_1, f_2 \text{ in } f_1(f_2\,x)$$

$$\mathrm{succ} \triangleq \lambda nfx.\mathrm{copy}^!_1\ f \text{ as } f_1, f_2 \text{ in } f_1(nf_2x)$$

$$\mathrm{add} \triangleq \lambda mnfx.\mathrm{copy}^!_1\ f \text{ as } f_1, f_2 \text{ in } mf_1(nf_2x).$$

Exponential compression and applications (this thesis)

- *Translation* $(\_)^{\bullet} : \mathsf{LEM} \to \mathsf{IMLL}_2$:

$$\begin{array}{rcl} \sigma \quad \text{(closed lazy)} & \mapsto & \sigma^{\bullet} \quad \text{(closed } \Pi_1\text{)} \\ \mathtt{discard}_{\sigma} & \mapsto & \mathtt{E}_{\sigma^{\bullet}} \\ \mathtt{copy}^{V}_{\sigma} & \mapsto & \mathtt{D}_{\sigma^{\bullet}} \end{array}$$

- Since $\mathrm{size}(\mathtt{D}_A) \in \mathcal{O}(2^{\mathrm{size}(A)^2})$:

Theorem (Exponential compression)

If $\Gamma \vdash_{\mathsf{LEM}} M : \sigma$ then $\mathrm{size}(M^{\bullet})$ can be **exponential** w.r.t $\mathrm{size}(M)$.

- **Applications**: compact encodings of **boolean circuits** and **natural numbers**.

$$\overline{2} \triangleq \lambda fx.\mathtt{copy}^{!}_{\mathbf{1}}\ f \ \mathtt{as}\ f_1, f_2 \ \mathtt{in}\ f_1(f_2\, x)$$

$$\mathtt{succ} \triangleq \lambda nfx.\mathtt{copy}^{!}_{1}\ f \ \mathtt{as}\ f_1, f_2 \ \mathtt{in}\ f_1(nf_2x)$$

$$\mathtt{add} \triangleq \lambda mnfx.\mathtt{copy}^{!}_{1}\ f \ \mathtt{as}\ f_1, f_2 \ \mathtt{in}\ mf_1(nf_2x).$$

PART II

LINEAR ADDITIVES AND PROBABILISTIC POLYNOMIAL TIME

Gianluca Curzi. *Linear Additives*. To be presented in the workshop *TLLA*, 2020.

Introduction

- *Additive rules* of Linear Logic:

$$\langle M, N\rangle : A \,\&\, B \qquad \pi_1 : A \,\&\, B \multimap A \qquad \pi_2 : A \,\&\, B \multimap B$$

- Variants of & to capture NP [Maurel 03, Matsuoka 04, Gaboardi et al. 08].

- Drawback: **exponential** blow up

$$\text{e.g.} \qquad (\lambda x.\langle x, x\rangle)M \quad \rightsquigarrow \quad \langle M, M\rangle$$

- *Lazy reduction* to "freeze" evaluation [Girard 96].

- This thesis: exploit linear erasure/duplication of IMLL_2
  - *linear additive rules* for **strong** linear normalization;
  - **probabilistic** linear additive rules to capture the probabilistic polytime functions.

Introduction

- *Additive rules* of Linear Logic:

$$\langle M, N\rangle : A \,\&\, B \qquad \pi_1 : A \,\&\, B \multimap A \qquad \pi_2 : A \,\&\, B \multimap B$$

- Variants of & to capture NP [Maurel 03, Matsuoka 04, Gaboardi et al. 08].
- Drawback: **exponential** blow up

$$\text{e.g.} \qquad (\lambda x.\langle x, x\rangle)M \quad \rightsquigarrow \quad \langle M, M\rangle$$

- *Lazy reduction* to "freeze" evaluation [Girard 96].
- This thesis: exploit linear erasure/duplication of IMLL_2
  - *linear additive rules* for **strong** linear normalization;
  - **probabilistic** linear additive rules to capture the probabilistic polytime functions.

Introduction

- *Additive rules* of Linear Logic:

$$\langle M, N\rangle : A \,\&\, B \qquad \pi_1 : A \,\&\, B \multimap A \qquad \pi_2 : A \,\&\, B \multimap B$$

- Variants of & to capture NP [Maurel 03, Matsuoka 04, Gaboardi et al. 08].

- Drawback: **exponential** blow up

$$\text{e.g.} \qquad (\lambda x.\langle x, x\rangle)M \quad \rightsquigarrow \quad \langle M, M\rangle$$

- *Lazy reduction* to "freeze" evaluation [Girard 96].

- This thesis: exploit linear erasure/duplication of IMLL_2
  - *linear additive rules* for **strong** linear normalization;
  - **probabilistic** linear additive rules to capture the probabilistic polytime functions.

Introduction

- *Additive rules* of Linear Logic:

$$\langle M, N \rangle : A \,\&\, B \qquad \pi_1 : A \,\&\, B \multimap A \qquad \pi_2 : A \,\&\, B \multimap B$$

- Variants of & to capture NP [Maurel 03, Matsuoka 04, Gaboardi et al. 08].
- Drawback: **exponential** blow up

$$\text{e.g.} \qquad (\lambda x.\langle x, x \rangle)M \quad \rightsquigarrow \quad \langle M, M \rangle$$

- *Lazy reduction* to "freeze" evaluation [Girard 96].
- This thesis: exploit linear erasure/duplication of IMLL_2
  - *linear additive rules* for **strong** linear normalization;
  - **probabilistic** linear additive rules to capture the probabilistic polytime functions.

Introduction

- *Additive rules* of Linear Logic:

$$\langle M, N\rangle : A \mathbin{\&} B \qquad \pi_1 : A \mathbin{\&} B \multimap A \qquad \pi_2 : A \mathbin{\&} B \multimap B$$

- Variants of & to capture NP [Maurel 03, Matsuoka 04, Gaboardi et al. 08].
- Drawback: **exponential** blow up

$$\text{e.g.} \qquad (\lambda x.\langle x, x\rangle)M \quad \rightsquigarrow \quad \langle M, M\rangle$$

- *Lazy reduction* to "freeze" evaluation [Girard 96].
- This thesis: exploit linear erasure/duplication of IMLL_2
  - *linear additive rules* for **strong** linear normalization;
  - **probabilistic** linear additive rules to capture the probabilistic polytime functions.

Linear additives (this thesis)

- Linear additives ("weaker" additives), e.g.

$$\frac{x_1 : A \vdash M_1 : B_1 \qquad x_2 : A \vdash M_2 : B_2 \qquad \vdash V : A}{x : A \vdash \mathtt{copy}^{V}_{A}\, N \;\mathtt{as}\; x_1, x_2 \;\mathtt{in}\; \langle M_1, M_2 \rangle : B_1 \wedge B_2} \wedge\mathrm{R}$$

A *closed and lazy* (no negative $\forall$) and V *value* (closed normal linear λ-term).

- System LAM = IMLL$_2$ + linear additives:

Theorem (Strong linear normalization)

If $\Gamma \vdash_{\mathsf{LAM}} M : A$ then M normalizes in a linear number of steps.

- No need of **laziness** to recover linear normalization with additives!
- **Probabilistic** linear additives to capture probabilistic polytime functions?

Linear additives (this thesis)

- Linear additives ("weaker" additives), e.g.

$$\frac{x_1 : A \vdash M_1 : B_1 \qquad x_2 : A \vdash M_2 : B_2 \qquad \vdash V : A}{x : A \vdash \texttt{copy}_A^V\, N \;\texttt{as}\; x_1, x_2 \;\texttt{in}\; \langle M_1, M_2 \rangle : B_1 \wedge B_2}\ \wedge\mathrm{R}$$

A *closed and lazy* (no negative $\forall$) and V *value* (closed normal linear λ-term).

- System LAM $=$ IMLL$_2$ $+$ linear additives:

Theorem (Strong linear normalization)

If $\Gamma \vdash_{\mathsf{LAM}} M : A$ then M normalizes in a linear number of steps.

- No need of **laziness** to recover linear normalization with additives!

- **Probabilistic** linear additives to capture probabilistic polytime functions?

Linear additives (this thesis)

- Linear additives ("weaker" additives), e.g.

$$\frac{x_1 : A \vdash M_1 : B_1 \qquad x_2 : A \vdash M_2 : B_2 \qquad \vdash V : A}{x : A \vdash \mathtt{copy}_A^V\, N \,\mathtt{as}\, x_1, x_2 \,\mathtt{in}\, \langle M_1, M_2 \rangle : B_1 \wedge B_2} \wedge\mathrm{R}$$

A *closed and lazy* (no negative $\forall$) and V *value* (closed normal linear λ-term).

- System LAM = IMLL_2 + linear additives:

Theorem (Strong linear normalization)

If $\Gamma \vdash_{\mathrm{LAM}} M : A$ then M normalizes in a linear number of steps.

- No need of **laziness** to recover linear normalization with additives!
- **Probabilistic** linear additives to capture probabilistic polytime functions?

The system STA$_\oplus$ (this thesis)

- STA (**polynomial time**) [Gaboardi&Ronchi 09].
- *Linear Lambda Calculus* (**confluence**) [Simpson 05].
- Explicit dereliction (**subject reduction**) [Ronchi&Roversi 97].
- Linear additives + type-dependency (**non-determinism**) [Diaz-Caro 13].

$$\frac{\Gamma, x_1 : \sigma, \overset{n}{\ldots}, x_n : \sigma \vdash M : \tau \qquad (n \geq 0)}{\Gamma, x : !\sigma \vdash M[x/x_1, \ldots, x/x_n] : \tau}\ m$$

$$\frac{x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \sigma}{!x_1 : \sigma_1, \ldots, x_n : !\sigma_n \vdash M : !\sigma}\ sp$$

The system $\text{STA}_{\oplus}$ (this thesis)

- STA (**polynomial time**) [Gaboardi&Ronchi 09].
- *Linear Lambda Calculus* (**confluence**) [Simpson 05].
- Explicit dereliction (**subject reduction**) [Ronchi&Roversi 97].
- Linear additives + type-dependency (**non-determinism**) [Diaz-Caro 13].

$$\frac{\Gamma, x_1 : \sigma, \overset{n}{\ldots}, x_n : \sigma \vdash M : \tau \qquad (n \geq 0)}{\Gamma, x : !\sigma \vdash M[x/x_1, \ldots, x/x_n] : \tau}\ m$$

$$\frac{x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau}{y_1 : !\sigma_1, \ldots, y_n : !\sigma_n \vdash !M[y_1/x_1, \ldots, y_n/x_n] : !\tau}\ sp$$

The system STA$_\oplus$ (this thesis)

- STA (**polynomial time**) [Gaboardi&Ronchi 09].
- *Linear Lambda Calculus* (**confluence**) [Simpson 05].
- Explicit dereliction (**subject reduction**) [Ronchi&Roversi 97].
- Linear additives + type-dependency (**non-determinism**) [Diaz-Caro 13].

$$\frac{\Gamma, x_1 : \sigma, \overset{n}{\ldots}, x_n : \sigma \vdash M : \tau \qquad (n \geq 0)}{\Gamma, x : !\sigma \vdash M[\mathtt{d}(x)/x_1, \ldots, \mathtt{d}(x)/x_n] : \tau}\ m$$

$$\frac{x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau}{y_1 : !\sigma_1, \ldots, y_n : !\sigma_n \vdash !M[\mathtt{d}(y_1)/x_1, \ldots, \mathtt{d}(y_n)/x_n] : !\tau}\ sp$$

The system $\mathrm{STA}_{\oplus}$ (this thesis)

- STA (**polynomial time**) [Gaboardi&Ronchi 09].
- *Linear Lambda Calculus* (**confluence**) [Simpson 05].
- Explicit dereliction (**subject reduction**) [Ronchi&Roversi 97].
- Linear additives + type-dependency (**non-determinism**) [Diaz-Caro 13].

$$\frac{\Gamma, x_1 : \sigma, \overset{n}{\ldots}, x_n : \sigma \vdash M : \tau \qquad (n \geq 0)}{\Gamma, x : !\sigma \vdash M[\mathtt{d}(x)/x_1, \ldots, \mathtt{d}(x)/x_n] : \tau}\ m$$

$$\frac{x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau}{y_1 : !\sigma_1, \ldots, y_n : !\sigma_n \vdash !M[\mathtt{d}(y_1)/x_1, \ldots, \mathtt{d}(y_n)/x_n] : !\tau}\ sp$$

$$\frac{\Gamma \vdash M : A_1 \wedge A_2 \qquad C \in \{A_1, A_2\}}{\Gamma \vdash \mathtt{proj}_{C}^{A_1 \wedge A_2}(M) : C}\ \wedge\mathrm{E}$$

Probabilistic features from type-dependency (this thesis)

- (1) let non-determinism arise naturally from type-dependency:

$$\mathtt{proj}_{C}^{A_1 \wedge A_2}\langle M_1, M_2 \rangle \qquad\qquad C \in \{A_1, A_2\}$$

surface reduction (no evaluation under the scope of !).

- (2) from non-determinism to *probabilistic distributions* [Dal Lago&Toldin 15]:

$$M \Rightarrow \mathscr{D}$$

- Starting from [Gaboardi&Ronchi 09] and [Dal Lago&Toldin 15]:

Theorem (Probabilistic Polytime Characterization)

$\mathrm{STA}_{\oplus}$ captures the probabilistic polytime functions (no **lazy** reduction strategy).

- Similar characterizations for the classes PP ($\epsilon \leq \frac{1}{2}$) and BPP ($\epsilon < \frac{1}{2}$).

Probabilistic features from type-dependency (this thesis)

- (1) let non-determinism arise naturally from type-dependency:

$$M_1 \leftarrow \texttt{proj}_{A}^{A \wedge A} \langle M_1, M_2 \rangle \rightarrow M_2$$

  surface reduction (no evaluation under the scope of !).

- (2) from non-determinism to *probabilistic distributions* [Dal Lago&Toldin 15]:

$$M \Rightarrow \mathscr{D}$$

- Starting from [Gaboardi&Ronchi 09] and [Dal Lago&Toldin 15]:

Theorem (Probabilistic Polytime Characterization)

$\mathrm{STA}_{\oplus}$ captures the probabilistic polytime functions (no **lazy** reduction strategy).

- Similar characterizations for the classes PP ($\epsilon \leq \frac{1}{2}$) and BPP ($\epsilon < \frac{1}{2}$).

Probabilistic features from type-dependency (this thesis)

- (1) let non-determinism arise naturally from type-dependency:

$$M_1 \leftarrow \mathtt{proj}_{A}^{A \wedge A} \langle M_1, M_2 \rangle \rightarrow M_2$$

surface reduction (no evaluation under the scope of !).

- (2) from non-determinism to *probabilistic distributions* [Dal Lago&Toldin 15]:

$$M \Rightarrow \mathscr{D}$$

- Starting from [Gaboardi&Ronchi 09] and [Dal Lago&Toldin 15]:

Theorem (Probabilistic Polytime Characterization)

$\mathrm{STA}_{\oplus}$ captures the probabilistic polytime functions (no **lazy** reduction strategy).

- Similar characterizations for the classes PP ($\epsilon \leq \frac{1}{2}$) and BPP ($\epsilon < \frac{1}{2}$).

Probabilistic features from type-dependency (this thesis)

- (1) let non-determinism arise naturally from type-dependency:

$$M_1 \leftarrow \texttt{proj}_{A}^{A\ \wedge A} \langle M_1, M_2 \rangle \rightarrow M_2$$

surface reduction (no evaluation under the scope of !).

- (2) from non-determinism to *probabilistic distributions* [Dal Lago&Toldin 15]:

$$M \Rightarrow \mathscr{D}$$

- Starting from [Gaboardi&Ronchi 09] and [Dal Lago&Toldin 15]:

Theorem (Probabilistic Polytime Characterization)

$\mathrm{STA}_{\oplus}$ captures the probabilistic polytime functions (no **lazy** reduction strategy).

- Similar characterizations for the classes PP ($\epsilon \leq \frac{1}{2}$) and BPP ($\epsilon < \frac{1}{2}$).

Probabilistic features from type-dependency (this thesis)

- (1) let non-determinism arise naturally from type-dependency:

$$M_1 \leftarrow \texttt{proj}_{A}^{A\ \wedge A}\ \langle M_1, M_2\rangle \rightarrow M_2$$

surface reduction (no evaluation under the scope of !).

- (2) from non-determinism to *probabilistic distributions* [Dal Lago&Toldin 15]:

$$M \Rightarrow \mathscr{D}$$

- Starting from [Gaboardi&Ronchi 09] and [Dal Lago&Toldin 15]:

Theorem (Probabilistic Polytime Characterization)

$\mathrm{STA}_{\oplus}$ captures the probabilistic polytime functions (no **lazy** reduction strategy).

- Similar characterizations for the classes PP ($\epsilon \leq \frac{1}{2}$) and BPP ($\epsilon < \frac{1}{2}$).

PART III

THE BENEFIT OF BEING NON-LAZY IN PROBABILISTIC λ-CALCULUS

Gianluca Curzi and Michele Pagani. *The Benefit of Being Non-lazy in Probabilistic λ-calculus*. In *LICS*, 2020.

Introduction

- Program equivalence: contextual equivalence vs bisimilarity.

- *Applicative bisimilarity* [Abramsky 93].

$$\Lambda^{\mathrm{cbn}} = \mathsf{LTS}$$

- *Probabilistic applicative bisimilarity (PAB)* [Dal Lago et al. 13].

$$\Lambda^{\mathrm{cbn}}_{\oplus} = \mathsf{LMC}$$

Introduction

- Program equivalence: contextual equivalence vs bisimilarity.

- *Applicative bisimilarity* [Abramsky 93].

$$\Lambda^{\mathrm{cbn}} = \mathsf{LTS}$$

- *Probabilistic applicative bisimilarity (PAB)* [Dal Lago et al. 13].

$$\Lambda_{\oplus}^{\mathrm{cbn}} = \mathsf{LMC}$$

Introduction

- Program equivalence: contextual equivalence vs bisimilarity.

- *Applicative bisimilarity* [Abramsky 93].

$$\Lambda^{\mathrm{cbn}} = \mathsf{LTS}$$

- *Probabilistic applicative bisimilarity (PAB)* [Dal Lago et al. 13].

$$\Lambda_{\oplus}^{\mathrm{cbn}} = \mathsf{LMC}$$

Introduction

- Program equivalence: contextual equivalence vs bisimilarity.
- *Applicative bisimilarity* [Abramsky 93].

$$\Lambda^{\mathrm{cbn}} = \mathsf{LTS}$$

- *Probabilistic applicative bisimilarity (PAB)* [Dal Lago et al. 13].

$$\Lambda^{\mathrm{cbn}}_{\oplus} = \mathsf{LMC}$$

- Contextual equivalence ($=_{\text{cxt}}$) vs PAB ($\sim$).

Full Abstraction = Soundness + Completeness.

- Contextual equivalence ($=_{\text{cxt}}$) vs PAB ($\sim$).

value = **function** (e.g. $\lambda x.M$)

	lazy			

Full Abstraction = Soundness + Completeness.

- Contextual equivalence ($=_{\mathrm{cxt}}$) vs PAB ($\sim$).

value = **function** (e.g. $\lambda x.M$)

$\lambda x.(\lambda y.y)x \to_{\beta} \lambda x.x$

	lazy			*non-lazy*

Full Abstraction = Soundness + Completeness.

- Contextual equivalence ($=_{\mathsf{cxt}}$) vs PAB ($\sim$).

$$(\lambda x.xx)(\mathbf{II}) \to_\beta (\mathbf{II})(\mathbf{II})$$

	lazy			*non-lazy*
	cbn			

Full Abstraction = Soundness + Completeness.

- Contextual equivalence ($=_{\mathrm{cxt}}$) vs PAB ($\sim$).

$$(\lambda x.xx)(\mathbf{II}) \rightarrow_\beta (\mathbf{II})(\mathbf{II})$$

	lazy			*non-lazy*
	cbn	cbv		

$$(\lambda x.xx)(\mathbf{II}) \rightarrow_\beta (\lambda x.xx)\mathbf{I} \rightarrow_\beta \mathbf{II}$$

Full Abstraction = Soundness + Completeness.

- Contextual equivalence ($=_{\text{cxt}}$) vs PAB ($\sim$). Previous results:

	lazy			*non-lazy*
	cbn	cbv		
Soundness ($\sim\ \subseteq\ =_{\text{cxt}}$)	✓			
Completeness ($=_{\text{cxt}}\ \subseteq\ \sim$)	✗			

[Dal Lago et al. 14]

Full Abstraction = Soundness + Completeness.

- Contextual equivalence ($=_{\text{cxt}}$) vs PAB ($\sim$). Previous results:

	lazy			*non-lazy*
	cbn	cbv		
Soundness ($\sim \subseteq =_{\text{cxt}}$)	✓	✓		
Completeness ($=_{\text{cxt}} \subseteq \sim$)	*X*	✓		

[Crubillé&Dal Lago 14]

Full Abstraction = Soundness + Completeness.

- Contextual equivalence ($=_{\mathrm{cxt}}$) vs PAB ($\sim$). Previous results:

	lazy			*non-lazy*
	cbn	cbv	cbn+`let`	
Soundness ($\sim\ \subseteq\ =_{\mathrm{cxt}}$)	✓	✓	✓	
Completeness ($=_{\mathrm{cxt}}\ \subseteq\ \sim$)	X	✓	✓	

[Kasterovic&Pagani 19]

Full Abstraction = Soundness + Completeness.

- Contextual equivalence ($=_{\mathsf{cxt}}$) vs PAB ($\sim$). This thesis:

	lazy			*non-lazy*
	cbn	cbv	cbn+`let`	head
Soundness ($\sim \subseteq =_{\mathsf{cxt}}$)	✓	✓	✓	✓
Completeness ($=_{\mathsf{cxt}} \subseteq \sim$)	✗	✓	✓	✓

Full Abstraction = Soundness + Completeness.

- Contextual equivalence ($=_{\text{cxt}}$) vs PAB ($\sim$). This thesis:

via *Context Lemma* (no Howe's method)

	lazy			*non-lazy*
	cbn	cbv	cbn+`let`	head
Soundness ($\sim \subseteq =_{\text{cxt}}$)	✓	✓	✓	✓
Completeness ($=_{\text{cxt}} \subseteq \sim$)	*X*	✓	✓	

Full Abstraction = Soundness + Completeness.

- Contextual equivalence ($=_{\text{cxt}}$) vs PAB ($\sim$). This thesis:

via *Context Lemma* (no Howe's method)

	lazy			*non-lazy*
	cbn	cbv	cbn+`let`	head
Soundness ($\sim\ \subseteq\ =_{\text{cxt}}$)	✓	✓	✓	✓
Completeness ($=_{\text{cxt}}\ \subseteq\ \sim$)	*X*	✓	✓	✓

via *Separation Theorem* [Leventis 18] (no testing equivalence)

Full Abstraction = Soundness + Completeness.

Probabilistic λ-calculus $\Lambda_\oplus$ and operational semantics $[\![\cdot]\!]$

- Probabilistic λ-calculus $\Lambda_\oplus$:

$$\begin{aligned} M &:= x \mid \lambda x.M \mid (MM) \mid M \oplus M \\ H &:= \lambda x_1 \dots x_n.yM_1 \dots M_m \qquad\qquad \text{(head nf)} \end{aligned}$$

- *Big-step approximation* $\Downarrow \subseteq \Lambda_\oplus \times \mathfrak{D}(\mathrm{HEAD})$, e.g.:

$$\frac{M \Downarrow \mathscr{D} \qquad \{H[N/x] \Downarrow \mathscr{E}_{H,N}\}_{\lambda x.H \in \mathrm{supp}(\mathscr{D})}}{MN \Downarrow \sum_{\lambda x.H \in \mathrm{supp}(\mathscr{D})} \mathscr{D}(\lambda x.H) \cdot \mathscr{E}_{H,N} + \sum_{\substack{H \in \mathrm{supp}(\mathscr{D}) \\ H \text{ is neutral}}} \mathscr{D}(H) \cdot HN}\ s5$$

- Big-step semantics: $[\![M]\!] := \sup\{\mathscr{D} \mid M \Downarrow \mathscr{D}\}$

Theorem (Head = Spine)

$\forall M \in \Lambda_\oplus, \forall H \in \mathrm{HEAD}, \forall n \in \mathbb{N}: \mathrm{Prob}^n_{head}[M, H] = \mathrm{Prob}^n_{spine}[M, H]$.

Probabilistic λ-calculus $\Lambda_\oplus$ and operational semantics $[\![\cdot]\!]$

- Probabilistic λ-calculus $\Lambda_\oplus$:

$$M := x \mid \lambda x.M \mid (MM) \mid M \oplus M$$
$$H := \lambda x_1 \dots x_n . y M_1 \dots M_m \qquad \text{(head nf)}$$

- *Big-step approximation* $\Downarrow \subseteq \Lambda_\oplus \times \mathfrak{D}(\mathrm{HEAD})$, e.g.:

$$\frac{M \Downarrow \mathscr{D} \qquad \{H[N/x] \Downarrow \mathscr{E}_{H,N}\}_{\lambda x.H \in \mathrm{supp}(\mathscr{D})}}{MN \Downarrow \sum_{\lambda x.H \in \mathrm{supp}(\mathscr{D})} \mathscr{D}(\lambda x.H) \cdot \mathscr{E}_{H,N} + \sum_{\substack{H \in \mathrm{supp}(\mathscr{D}) \\ H \text{ is neutral}}} \mathscr{D}(H) \cdot HN}\; s5$$

- Big-step semantics: $[\![M]\!] := \sup\{\mathscr{D} \mid M \Downarrow \mathscr{D}\}$

Theorem (Head = Spine)

$\forall M \in \Lambda_\oplus, \forall H \in \mathrm{HEAD}, \forall n \in \mathbb{N}$: $\mathrm{Prob}^n_{head}[M, H] = \mathrm{Prob}^n_{spine}[M, H]$.

Probabilistic λ-calculus $\Lambda_\oplus$ and operational semantics $[\![\cdot]\!]$

- Probabilistic λ-calculus $\Lambda_\oplus$:

$$\begin{aligned} M &:= x \mid \lambda x.M \mid (MM) \mid M \oplus M \\ H &:= \lambda x_1 \ldots x_n.yM_1 \ldots M_m \qquad\qquad \text{(head nf)} \end{aligned}$$

- *Big-step approximation* $\Downarrow \subseteq \Lambda_\oplus \times \mathfrak{D}(\mathrm{HEAD})$, e.g.:

$$\frac{M \Downarrow \mathscr{D} \qquad \{{\color{red}H}[N/x] \Downarrow \mathscr{E}_{H,N}\}_{{\color{red}\lambda x.H} \in \mathrm{supp}(\mathscr{D})}}{MN \Downarrow \displaystyle\sum_{\lambda x.H \in \mathrm{supp}(\mathscr{D})} \mathscr{D}(\lambda x.H) \cdot \mathscr{E}_{H,N} + \sum_{\substack{H \in \mathrm{supp}(\mathscr{D}) \\ H \text{ is neutral}}} \mathscr{D}(H) \cdot HN}\ s5$$

- Big-step semantics: $\qquad [\![M]\!] := \sup\{\mathscr{D} \mid M \Downarrow \mathscr{D}\}$

Theorem (Head = Spine)

$\forall M \in \Lambda_\oplus, \forall H \in \mathrm{HEAD}, \forall n \in \mathbb{N}$: $\mathrm{Prob}^n_{head}[M, H] = \mathrm{Prob}^n_{spine}[M, H]$.

Probabilistic λ-calculus $\Lambda_\oplus$ and operational semantics $[\![\cdot]\!]$

- Probabilistic λ-calculus $\Lambda_\oplus$:

$$M := x \mid \lambda x.M \mid (MM) \mid M \oplus M$$
$$H := \lambda x_1 \ldots x_n.yM_1 \ldots M_m \qquad \text{(head nf)}$$

- *Big-step approximation* $\Downarrow \subseteq \Lambda_\oplus \times \mathfrak{D}(\mathrm{HEAD})$, e.g.:

$$\frac{M \Downarrow \mathscr{D} \qquad \{H[N/x] \Downarrow \mathscr{E}_{H,N}\}_{\lambda x.H \in \mathrm{supp}(\mathscr{D})}}{MN \Downarrow \sum_{\lambda x.H \in \mathrm{supp}(\mathscr{D})} \mathscr{D}(\lambda x.H) \cdot \mathscr{E}_{H,N} + \sum_{\substack{H \in \mathrm{supp}(\mathscr{D}) \\ H \text{ is neutral}}} \mathscr{D}(H) \cdot HN}\ s5$$

- Big-step semantics: $\qquad [\![M]\!] := \sup\{\mathscr{D} \mid M \Downarrow \mathscr{D}\}$

Theorem (Head = Spine)

$\forall M \in \Lambda_\oplus$, $\forall H \in \mathrm{HEAD}$, $\forall n \in \mathbb{N}$: $\mathrm{Prob}^n_{head}[M, H] = \mathrm{Prob}^n_{spine}[M, H]$.

Contextual equivalence ($=_{\mathrm{cxt}}$)

- Contextual equivalence ($=_{\mathrm{cxt}}$):

$$M =_{\mathrm{cxt}} N \quad \text{iff} \quad \forall \mathcal{C} \quad \sum [\![\mathcal{C}[M]]\!] = \sum [\![\mathcal{C}[N]]\!].$$

$\mathcal{C}$ = term with a "hole" $[\cdot]$, e.g. $\lambda x.\lambda y.[\cdot]xy$.

- Example: $\lambda z.z(\mathbf{\Omega} \oplus \mathbf{I}) \neq_{\mathrm{cxt}} \lambda z.(z\mathbf{\Omega} \oplus z\mathbf{I})$. If $\mathcal{C} \triangleq [\cdot]\mathbf{\Delta}$ then:

$$(\lambda z.z(\mathbf{\Omega} \oplus \mathbf{I}))\mathbf{\Delta} \xrightarrow[0.25]{}^{*} \mathbf{I}$$

$$(\lambda z.(z\mathbf{\Omega} \oplus z\mathbf{I}))\mathbf{\Delta} \xrightarrow[0.50]{}^{*} \mathbf{I}$$

where $\mathbf{I} \triangleq \lambda x.x$, $\mathbf{\Delta} \triangleq \lambda x.xx$, and $\mathbf{\Omega} \triangleq \mathbf{\Delta\Delta}$.

Contextual equivalence ($=_{\mathrm{cxt}}$)

- Contextual equivalence ($=_{\mathrm{cxt}}$):

$$M =_{\mathrm{cxt}} N \quad \text{iff} \quad \forall \mathcal{C} \quad \sum [\![\mathcal{C}[M]]\!] = \sum [\![\mathcal{C}[N]]\!].$$

$\mathcal{C}$ = term with a "hole" $[\cdot]$, e.g. $\lambda x.\lambda y.[\cdot]xy$.

- Example: $\lambda z.z(\mathbf{\Omega} \oplus \mathbf{I}) \neq_{\mathrm{cxt}} \lambda z.(z\mathbf{\Omega} \oplus z\mathbf{I})$. If $\mathcal{C} \triangleq [\cdot]\mathbf{\Delta}$ then:

$$(\lambda z.z(\mathbf{\Omega} \oplus \mathbf{I}))\mathbf{\Delta} \xrightarrow[0.25]{}^{*} \mathbf{I}$$

$$(\lambda z.(z\mathbf{\Omega} \oplus z\mathbf{I}))\mathbf{\Delta} \xrightarrow[0.50]{}^{*} \mathbf{I}$$

where $\mathbf{I} \triangleq \lambda x.x$, $\mathbf{\Delta} \triangleq \lambda x.xx$, and $\mathbf{\Omega} \triangleq \mathbf{\Delta\Delta}$.

Probabilistic Applicative Bisimilarity ($\sim$)

- $\Lambda_{\oplus}^{\mathrm{head}}$ as a LMC:

$$\begin{tikzcd}
M \arrow[r, "\tau", "p"'] \arrow[ddr, "\tau"', "p'"] & H \\
 & \cdots \\
 & H'
\end{tikzcd}
\qquad\qquad
\lambda x.H \xrightarrow[1]{M} H[M/x]$$

- *Probabilistic applicative bisimulation*: $\mathcal{R}$ = equivalence relation such that, e.g.

$$\begin{tikzcd}
 & M \arrow[dr, "\tau", "p"'] & \\
\forall H & \mathcal{R} & \{H' \mid H' \,\mathcal{R}\, H\} \\
 & N \arrow[ur, "\tau", "p"'] &
\end{tikzcd}$$

- Example: if $\mathsf{fix} \triangleq (\lambda y.\mathbf{I} \oplus yy)(\lambda y.\mathbf{I} \oplus yy)$ then $\lambda x.x \oplus x \sim \mathsf{fix}$, since:

$$\lambda x.x \oplus x \xrightarrow[1]{\tau} \mathbf{I} \qquad\qquad \mathsf{fix} \xrightarrow[1]{\tau} \mathbf{I}$$

so $\mathcal{R} \triangleq \{(\lambda x.x \oplus x, \mathsf{fix}), (\mathsf{fix}, \lambda x.x \oplus x)\} \cup \{(N, N) \mid N \in \Lambda_{\oplus}^{\emptyset}\}$ is bisimulation.

Probabilistic Applicative Bisimilarity ($\sim$)

- $\Lambda_{\oplus}^{\mathrm{head}}$ as a LMC:

$$\begin{tikzcd}
M \arrow[r, "\tau", "p"'] \arrow[ddr, "\tau"', "p'"] & H \\
& \cdots \\
& H'
\end{tikzcd}
\qquad\qquad
\lambda x.H \xrightarrow[1]{M} H[M/x]$$

- *Probabilistic applicative bisimulation*: $\mathcal{R}$ = equivalence relation such that, e.g.

$$\begin{tikzcd}
& M \arrow[dr, "\tau", "p"'] & \\
\forall H & \mathcal{R} & \{H' \mid H' \, \mathcal{R} \, H\} \\
& N \arrow[ur, "\tau", "p"'] &
\end{tikzcd}$$

- Example: if fix $\triangleq (\lambda y.\mathbf{I} \oplus yy)(\lambda y.\mathbf{I} \oplus yy)$ then $\lambda x.x \oplus x \sim$ fix, since:

$$\lambda x.x \oplus x \xrightarrow[1]{\tau} \mathbf{I} \qquad\qquad \mathrm{fix} \xrightarrow[1]{\tau} \mathbf{I}$$

so $\mathcal{R} \triangleq \{(\lambda x.x \oplus x, \mathrm{fix}), (\mathrm{fix}, \lambda x.x \oplus x)\} \cup \{(N, N) \mid N \in \Lambda_{\oplus}^{\emptyset}\}$ is bisimulation.

Probabilistic Applicative Bisimilarity ($\sim$)

- $\Lambda_{\oplus}^{\text{head}}$ as a LMC:

$$\begin{tikzcd} M \arrow[r, "\tau", "p"'] \arrow[ddr, "\tau"', "p'"] & H \\ & \cdots \\ & H' \end{tikzcd}$$

$$\lambda x.H \xrightarrow[1]{M} H[M/x]$$

- Probabilistic applicative bisimilarity (PAB): $\sim$ = the "largest" bisimulation:

$$\forall H \qquad \begin{tikzcd} M \arrow[dr, "\tau", "p"'] & \\ \sim & \{H' \mid H' \sim H\} \\ N \arrow[ur, "\tau", "p"'] & \end{tikzcd}$$

- Example: if $\text{fix} \triangleq (\lambda y.\mathbf{I} \oplus yy)(\lambda y.\mathbf{I} \oplus yy)$ then $\lambda x.x \oplus x \sim \text{fix}$, since:

$$\lambda x.x \oplus x \xrightarrow[1]{\tau} \mathbf{I} \qquad\qquad \text{fix} \xrightarrow[1]{\tau} \mathbf{I}$$

so $\mathcal{R} \triangleq \{(\lambda x.x \oplus x, \text{fix}), (\text{fix}, \lambda x.x \oplus x)\} \cup \{(N, N) \mid N \in \Lambda_{\oplus}^{\emptyset}\}$ is bisimulation.

Probabilistic Applicative Bisimilarity ($\sim$)

- $\Lambda_{\oplus}^{\mathrm{head}}$ as a LMC:

$$\begin{tikzcd} M \arrow[r, "\tau", "p"'] \arrow[ddr, "\tau", "p'"'] & H \\ & \cdots \\ & H' \end{tikzcd}$$

$$\lambda x.H \xrightarrow[1]{M} H[M/x]$$

- Probabilistic applicative bisimilarity (PAB): $\sim$ = the "largest" bisimulation:

$$\begin{tikzcd} & M \arrow[dr, "\tau", "p"'] & \\ \forall H & \sim & \{H' \mid H' \sim H\} \\ & N \arrow[ur, "\tau", "p"'] & \end{tikzcd}$$

- Example: if fix $\triangleq (\lambda y.\mathbf{I} \oplus yy)(\lambda y.\mathbf{I} \oplus yy)$ then $\lambda x.x \oplus x \sim \mathrm{fix}$, since:

$$\lambda x.x \oplus x \xrightarrow[1]{\tau} \mathbf{I} \qquad\qquad \mathrm{fix} \xrightarrow[1]{\tau} \mathbf{I}$$

so $\mathcal{R} \triangleq \{(\lambda x.x \oplus x, \mathrm{fix}), (\mathrm{fix}, \lambda x.x \oplus x)\} \cup \{(N, N) \mid N \in \Lambda_{\oplus}^{\emptyset}\}$ is bisimulation.

Probabilistic Nakajima trees [Leventis 18]

- *Separation Theorem* [Leventis 18]: $M =_{cxt} N$ implies $\mathcal{PT}(M) = \mathcal{PT}(N)$.

- *Böhm tree* ($\mathcal{BT}$):

$\mathcal{BT}(\lambda x_1 \ldots x_n.yM_1 \ldots M_k) \triangleq$

$\lambda x_1 \ldots x_n.y$

$\mathcal{BT}(M_1)$ $\ldots$ $\mathcal{BT}(M_k)$

- Probabilistic Nakajima tree ($\mathcal{PT}$):

$\mathcal{PT}(M) \triangleq$

$\oplus$

p p'

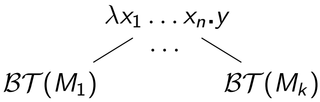
$\mathcal{VT}(H)$ $\ldots$ $\mathcal{VT}(H')$

Probabilistic Nakajima trees [Leventis 18]

- *Separation Theorem* [Leventis 18]: $M =_{cxt} N$ implies $\mathcal{PT}(M) = \mathcal{PT}(N)$.

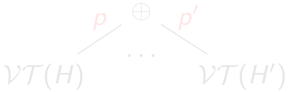
- *Böhm tree* ($\mathcal{BT}$):

$\mathcal{BT}(\lambda x_1 \dots x_n . y M_1 \dots M_k) \triangleq$



- Probabilistic Nakajima tree ($\mathcal{PT}$):

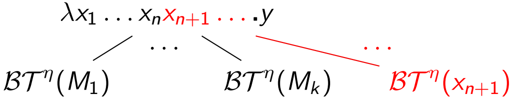
$\mathcal{PT}(M) \triangleq$



Probabilistic Nakajima trees [Leventis 18]

- *Separation Theorem* [Leventis 18]: $M =_{cxt} N$ implies $\mathcal{PT}(M) = \mathcal{PT}(N)$.
- *Nakajima tree* ($\mathcal{BT}^{\eta}$):

$\mathcal{BT}^{\eta}(\lambda x_1 \dots x_n . y M_1 \dots M_k) \triangleq$



$\lambda x_1 \dots x_n x_{n+1} \dots . y$

$\dots$

$\mathcal{BT}^{\eta}(M_1)$

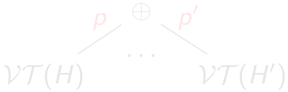
$\mathcal{BT}^{\eta}(M_k)$

$\dots$

$\mathcal{BT}^{\eta}(x_{n+1})$

- Probabilistic Nakajima tree ($\mathcal{PT}$):

$\mathcal{PT}(M) \triangleq$



$\oplus$

p

p'

$\dots$

$\mathcal{VT}(H)$

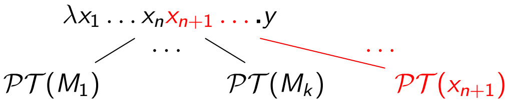
$\mathcal{VT}(H')$

Probabilistic Nakajima trees [Leventis 18]

- *Separation Theorem* [Leventis 18]: $M =_{cxt} N$ implies $\mathcal{PT}(M) = \mathcal{PT}(N)$.

- Value Nakajima tree ($\mathcal{VT}$):

$\mathcal{VT}(\lambda x_1 \dots x_n . y M_1 \dots M_k) \triangleq$



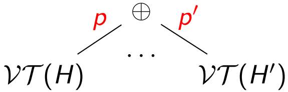
$\lambda x_1 \dots x_n x_{n+1} \dots . y$

$\dots$

$\mathcal{PT}(M_1)$ $\mathcal{PT}(M_k)$ $\dots$ $\mathcal{PT}(x_{n+1})$

- Probabilistic Nakajima tree ($\mathcal{PT}$):

$\mathcal{PT}(M) \triangleq$



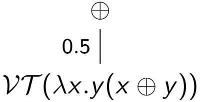
$\oplus$

p p'

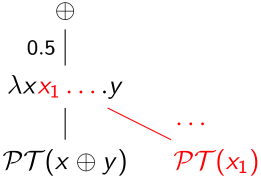
$\dots$

$\mathcal{VT}(H)$ $\mathcal{VT}(H')$

- Example: if $H \triangleq \lambda x.y(x \oplus y)$ then $\mathcal{PT}(H \oplus \mathbf{\Omega})$ is:



- Example: if $H \triangleq \lambda x.y(x \oplus y)$ then $\mathcal{PT}(H \oplus \mathbf{\Omega})$ is:



$\oplus$

0.5

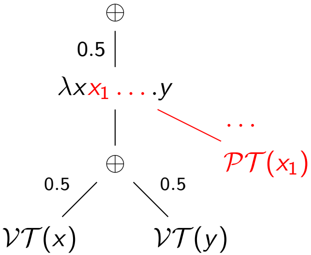
$\lambda x x_1 \ldots .y$

$\ldots$

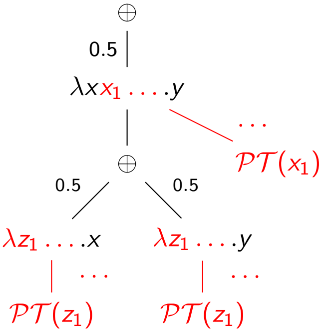
$\mathcal{PT}(x \oplus y)$

$\mathcal{PT}(x_1)$

- Example: if $H \triangleq \lambda x.y(x \oplus y)$ then $\mathcal{PT}(H \oplus \mathbf{\Omega})$ is:



- Example: if $H \triangleq \lambda x.y(x \oplus y)$ then $\mathcal{PT}(H \oplus \mathbf{\Omega})$ is:



Full abstraction (this thesis)

Theorem (Full abstraction)

Let $M, N \in \Lambda_\oplus$. We have $M \sim N$ iff $M =_{\mathrm{cxt}} N$.

- *Soundness* ($\sim \subseteq =_{\mathrm{cxt}}$): from $M \sim N$ we have

$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \mathcal{C}[M] \sim \mathcal{C}[N]$	using [Dal Lago et al. 14]
$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \sum [\![\mathcal{C}[M]]\!] = \sum [\![\mathcal{C}[N]]\!]$	by definition
$\Rightarrow M =_{\mathrm{cxt}} N$	Context Lemma

- *Completeness* ($=_{\mathrm{cxt}} \subseteq \sim$): show that $=_{\mathrm{cxt}}$ is a bisimulation

$\forall H \qquad =_{\mathrm{cxt}}$

Separation Theorem [Leventis 18]: $=_{\mathrm{cxt}}$ implies

Full abstraction (this thesis)

Theorem (Full abstraction)

Let $M, N \in \Lambda_{\oplus}$. We have $M \sim N$ iff $M =_{\text{cxt}} N$.

- *Soundness* ($\sim \subseteq =_{\text{cxt}}$): from $M \sim N$ we have

$$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \mathcal{C}[M] \sim \mathcal{C}[N] \qquad \text{using [Dal Lago et al. 14]}$$

$$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \sum [\![\mathcal{C}[M]]\!] = \sum [\![\mathcal{C}[N]]\!] \qquad \text{by definition}$$

$$\Rightarrow M =_{\text{cxt}} N \qquad \text{Context Lemma}$$

- *Completeness* ($=_{\text{cxt}} \subseteq \sim$): show that $=_{\text{cxt}}$ is a bisimulation

Separation Theorem [Leventis 18]: $=_{\text{cxt}}$ implies

Full abstraction (this thesis)

Theorem (Full abstraction)

Let $M, N \in \Lambda_\oplus$. We have $M \sim N$ iff $M =_{\mathrm{cxt}} N$.

- *Soundness* ($\sim \subseteq =_{\mathrm{cxt}}$): from $M \sim N$ we have

$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \mathcal{C}[M] \sim \mathcal{C}[N]$	using [Dal Lago et al. 14]
$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \sum [\![\mathcal{C}[M]]\!] = \sum [\![\mathcal{C}[N]]\!]$	by definition
$\Rightarrow M =_{\mathrm{cxt}} N$	Context Lemma

- *Completeness* ($=_{\mathrm{cxt}} \subseteq \sim$): show that $=_{\mathrm{cxt}}$ is a bisimulation

$$\begin{tikzcd}
 & M \arrow[rd, "\tau", "p"'] & \\
\forall H & =_{\mathrm{cxt}} & \{H' \mid H' =_{\mathrm{cxt}} H\} \\
 & N \arrow[ru, "\tau", "q"'] &
\end{tikzcd}$$

Separation Theorem [Leventis 18]: $M =_{\mathrm{cxt}} N$ implies $\mathcal{PT}(M) = \mathcal{PT}(N)$

Full abstraction (this thesis)

Theorem (Full abstraction)

Let $M, N \in \Lambda_{\oplus}$. We have $M \sim N$ iff $M =_{\mathrm{cxt}} N$.

- *Soundness* ($\sim \subseteq =_{\mathrm{cxt}}$): from $M \sim N$ we have

$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \mathcal{C}[M] \sim \mathcal{C}[N]$	using [Dal Lago et al. 14]
$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \sum [\![\mathcal{C}[M]]\!] = \sum [\![\mathcal{C}[N]]\!]$	by definition
$\Rightarrow M =_{\mathrm{cxt}} N$	Context Lemma

- *Completeness* ($=_{\mathrm{cxt}} \subseteq \sim$): show that $=_{\mathrm{cxt}}$ is a bisimulation

$$\begin{tikzcd}
 & M \arrow[dr, "\tau", "p"'] & \\
\forall H & =_{\mathrm{cxt}} & \{H' \mid H' =_{\mathrm{cxt}} H\} \\
 & N \arrow[ur, "\tau", "q"'] &
\end{tikzcd}$$

Separation Theorem [Leventis 18]: $M =_{\mathrm{cxt}} N$ implies $\mathcal{PT}(M) = \mathcal{PT}(N)$

Full abstraction (this thesis)

Theorem (Full abstraction)

Let $M, N \in \Lambda_{\oplus}$. We have $M \sim N$ iff $M =_{\mathrm{cxt}} N$.

- *Soundness* ($\sim \subseteq =_{\mathrm{cxt}}$): from $M \sim N$ we have

$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \dots, L_n\,,\ \mathcal{C}[M] \sim \mathcal{C}[N]$	using [Dal Lago et al. 14]
$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \dots, L_n\,,\ \sum [\![\mathcal{C}[M]]\!] = \sum [\![\mathcal{C}[N]]\!]$	by definition
$\Rightarrow M =_{\mathrm{cxt}} N$	Context Lemma

- *Completeness* ($=_{\mathrm{cxt}} \subseteq \sim$): show that $=_{\mathrm{cxt}}$ is a bisimulation

$$\begin{tikzcd}
 & M \arrow[dr, "\tau", "p"'] & \\
\forall H & =_{\mathrm{cxt}} & \{H' \mid H' =_{\mathrm{cxt}} H\} \\
 & N \arrow[ur, "\tau", "q"'] &
\end{tikzcd}$$

Separation Theorem [Leventis 18]: $M =_{\mathrm{cxt}} N$ implies $\mathcal{PT}(M) = \mathcal{PT}(N)$.

Full abstraction (this thesis)

Theorem (Full abstraction)

Let $M, N \in \Lambda_{\oplus}$. We have $M \sim N$ iff $M =_{\mathrm{cxt}} N$.

- *Soundness* ($\sim\, \subseteq\, =_{\mathrm{cxt}}$): from $M \sim N$ we have

$$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \mathcal{C}[M] \sim \mathcal{C}[N] \qquad \text{using [Dal Lago et al. 14]}$$

$$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \sum [\![\mathcal{C}[M]]\!] = \sum [\![\mathcal{C}[N]]\!] \qquad \text{by definition}$$

$$\Rightarrow M =_{\mathrm{cxt}} N \qquad \text{Context Lemma}$$

- *Completeness* ($=_{\mathrm{cxt}}\, \subseteq\, \sim$): show that $=_{\mathrm{cxt}}$ is a bisimulation, $p = q$?

$$\begin{tikzcd}
 & M \arrow[dr, "\tau", "p"'] & \\
\forall H & =_{\mathrm{cxt}} & \{H' \mid H' =_{\mathrm{cxt}} H\} \\
 & N \arrow[ur, "\tau", "q"'] &
\end{tikzcd}$$

Separation Theorem [Leventis 18]: $M =_{\mathrm{cxt}} N$ implies $\mathcal{PT}(M) = \mathcal{PT}(N)$.

Full abstraction (this thesis)

Theorem (Full abstraction)

Let $M, N \in \Lambda_{\oplus}$. We have $M \sim N$ iff $M =_{\mathrm{cxt}} N$.

- *Soundness* ($\sim \subseteq =_{\mathrm{cxt}}$): from $M \sim N$ we have

$$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \mathcal{C}[M] \sim \mathcal{C}[N] \qquad \text{using [Dal Lago et al. 14]}$$

$$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \sum [\![\mathcal{C}[M]]\!] = \sum [\![\mathcal{C}[N]]\!] \qquad \text{by definition}$$

$$\Rightarrow M =_{\mathrm{cxt}} N \qquad \text{Context Lemma}$$

- *Completeness* ($=_{\mathrm{cxt}} \subseteq \sim$): show that $=_{\mathrm{cxt}}$ is a bisimulation, $p = q$?

$$\begin{tikzcd}
\forall H & M \arrow[dr, "\tau", "p"'] & \\
& =_{\mathrm{cxt}} & \{H' \mid \mathcal{VT}(H') = \mathcal{VT}(H)\} \\
& N \arrow[ur, "\tau", "q"'] &
\end{tikzcd}$$

Separation Theorem [Leventis 18]: $M =_{\mathrm{cxt}} N$ implies $\mathcal{PT}(M) = \mathcal{PT}(N)$.

Full abstraction (this thesis)

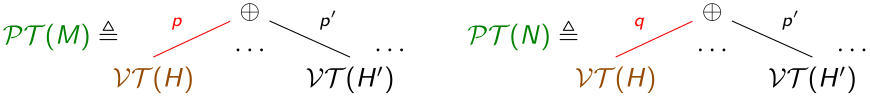
Theorem (Full abstraction)

Let $M, N \in \Lambda_{\oplus}$. We have $M \sim N$ iff $M =_{\mathrm{cxt}} N$.

- *Soundness* ($\sim \subseteq =_{\mathrm{cxt}}$): from $M \sim N$ we have

$$\begin{array}{ll} \Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \mathcal{C}[M] \sim \mathcal{C}[N] & \text{using [Dal Lago et al. 14]} \\ \Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \sum [\![\mathcal{C}[M]]\!] = \sum [\![\mathcal{C}[N]]\!] & \text{by definition} \\ \Rightarrow M =_{\mathrm{cxt}} N & \text{Context Lemma} \end{array}$$

- *Completeness* ($=_{\mathrm{cxt}} \subseteq \sim$): show that $=_{\mathrm{cxt}}$ is a bisimulation, $p = q$?



Separation Theorem [Leventis 18]: $M =_{\mathrm{cxt}} N$ implies $\mathcal{PT}(M) = \mathcal{PT}(N)$.

Full abstraction (this thesis)

Theorem (Full abstraction)

Let $M, N \in \Lambda_{\oplus}$. We have $M \sim N$ iff $M =_{\text{cxt}} N$.

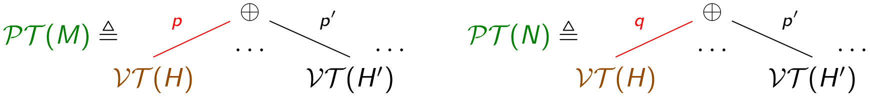
- *Soundness* ($\sim \subseteq =_{\text{cxt}}$): from $M \sim N$ we have

$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \mathcal{C}[M] \sim \mathcal{C}[N]$ using [Dal Lago et al. 14]

$\Rightarrow \forall \mathcal{C} \triangleq [\cdot]L_1, \ldots, L_n\,,\ \sum [\![\mathcal{C}[M]]\!] = \sum [\![\mathcal{C}[N]]\!]$ by definition

$\Rightarrow M =_{\text{cxt}} N$ Context Lemma

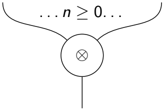
- *Completeness* ($=_{\text{cxt}} \subseteq \sim$): show that $=_{\text{cxt}}$ is a bisimulation, $p = q$?



Separation Theorem [Leventis 18]: $M =_{\text{cxt}} N$ implies $\mathcal{PT}(M) = \mathcal{PT}(N)$.

PERSPECTIVES

- **(Part I)**:
  - Depth-preserving translation of boolean circuits for LEM? *Unbounded fan-in* proof nets [Terui 04, Mogbil&Rahli 07, Aubert 11]:



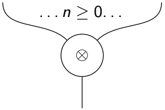
  - Characterization of the number-theoretic functions representable in LEM?

- **(Part II)**:
  - Linear additives based on the Linear Logic additive conjunction &. What about linear additives based on the additive disjunction $\oplus$?

	additives	linear additives
conjunction	&	$\wedge$
disjuncton	$\oplus$	$\vee$

  - Semantic characterizations of PP and BPP? *Obsessional cliques* [Tortora de Falco&Laurent 06]

- **(Part I)**:
  - Depth-preserving translation of boolean circuits for LEM? *Unbounded fan-in* proof nets [Terui 04, Mogbil&Rahli 07, Aubert 11]:



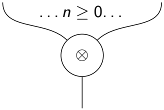
  - Characterization of the number-theoretic functions representable in LEM?

- **(Part II)**:
  - Linear additives based on the Linear Logic additive conjunction &. What about linear additives based on the additive disjunction $\oplus$?

	additives	linear additives
conjunction	&	$\wedge$
disjuncton	$\oplus$	$\vee$

  - Semantic characterizations of PP and BPP? *Obsessional cliques* [Tortora de Falco&Laurent 06]

- **(Part I)**:
  - Depth-preserving translation of boolean circuits for LEM? *Unbounded fan-in* proof nets [Terui 04, Mogbil&Rahli 07, Aubert 11]:

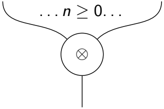


  - Characterization of the number-theoretic functions representable in LEM?
- **(Part II)**:
  - Linear additives based on the Linear Logic additive conjunction &. What about linear additives based on the additive disjunction $\oplus$?

	additives	linear additives
conjunction	&	$\wedge$
disjuncton	$\oplus$	$\vee$

  - Semantic characterizations of PP and BPP? *Obsessional cliques* [Tortora de Falco&Laurent 06]

- **(Part I)**:
  - Depth-preserving translation of boolean circuits for LEM? *Unbounded fan-in* proof nets [Terui 04, Mogbil&Rahli 07, Aubert 11]:



  - Characterization of the number-theoretic functions representable in LEM?

- **(Part II)**:
  - Linear additives based on the Linear Logic additive conjunction &. What about linear additives based on the additive disjunction $\oplus$?

	additives	linear additives
conjunction	&	$\wedge$
disjuncton	$\oplus$	$\vee$

  - Semantic characterizations of PP and BPP? *Obsessional cliques* [Tortora de Falco&Laurent 06]

- **(Part III)**:
  - Failure of full abstraction in the **asymmetric** case (this thesis):

Theorem

Probabilistic applicative similarity ($\precsim$) is sound but not complete (hence not fully abstract) for *contextual preorder* ($\leq_{\mathrm{cxt}}$).

- **Counterexample**: similar to [Crubillé&Dal Lago 14]

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

- In cbv asymmetric full abstraction in $\Lambda_{\oplus}$ + *parallel or* [Crubillé et al. 15].
- **Conjecture**: the same holds for non-lazy cbn.

- **(Part III)**:
  - Failure of full abstraction in the **asymmetric** case (this thesis):

Theorem

Probabilistic applicative similarity ($\precsim$) is sound but not complete (hence not fully abstract) for *contextual preorder* ($\leq_{\mathrm{cxt}}$).

- **Counterexample**: similar to [Crubillé&Dal Lago 14]

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

- In cbv asymmetric full abstraction in $\Lambda_{\oplus}$ + *parallel or* [Crubillé et al. 15].

- **Conjecture**: the same holds for non-lazy cbn.

- **(Part III)**:
  - Failure of full abstraction in the **asymmetric** case (this thesis):

Theorem

Probabilistic applicative similarity ($\precsim$) is sound but not complete (hence not fully abstract) for *contextual preorder* ($\leq_{\mathrm{cxt}}$).

- **Counterexample**: similar to [Crubillé&Dal Lago 14]

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

- In cbv asymmetric full abstraction in $\Lambda_{\oplus}$ + *parallel or* [Crubillé et al. 15].

- **Conjecture**: the same holds for non-lazy cbn.

- **(Part III)**:
  - Failure of full abstraction in the **asymmetric** case (this thesis):

Theorem

Probabilistic applicative similarity ($\precsim$) is sound but not complete (hence not fully abstract) for *contextual preorder* ($\leq_{\mathrm{cxt}}$).

- **Counterexample**: similar to [Crubillé&Dal Lago 14]

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

- In cbv asymmetric full abstraction in $\Lambda_{\oplus}$ + *parallel or* [Crubillé et al. 15].

- **Conjecture**: the same holds for non-lazy cbn.

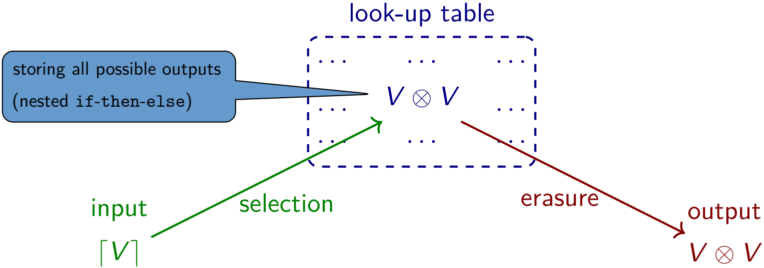
Thank you!
Questions?

APPENDIX

How duplicators work

Let A be a closed Π_1 type. The duplicator of A, written $\mathtt{D}_A$, implements two operations on a closed normal inhabitant V of A:

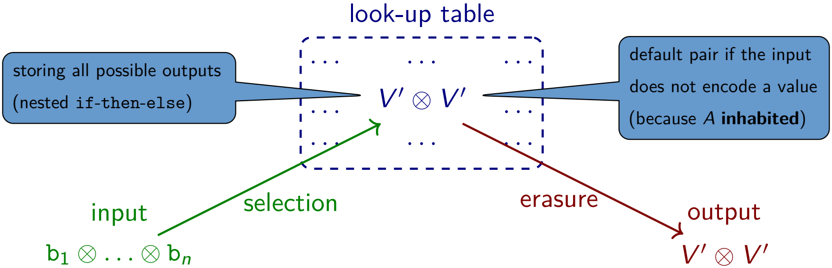
- **encode** V as a Boolean tuple $\lceil V \rceil$;
- **copy and decode** $\lceil V \rceil$ to obtain $V \otimes V$.



How duplicators work

Let A be a closed Π_1 type. The duplicator of A, written $\mathtt{D}_A$, implements two operations on a closed normal inhabitant V of A:

- **encode** V as a Boolean tuple $\lceil V \rceil$;
- **copy and decode** $\lceil V \rceil$ to obtain $V \otimes V$.



look-up table

storing all possible outputs (nested `if-then-else`)

$V' \otimes V'$

default pair if the input does not encode a value (because A **inhabited**)

input

$\mathtt{b}_1 \otimes \ldots \otimes \mathtt{b}_n$

selection

erasure

output

$V' \otimes V'$

Expressiveness of LEM and applications (this thesis)

- **Compact** and **modular** encoding $\lambda(\_)$ of **boolean circuits**:

$$C(\underbrace{x_1, \dots, x_n}_{n \text{ input nodes}}) = (\underbrace{y_1, \dots, y_m}_{m \text{ output nodes}}) \quad \mapsto \quad \lambda(C) : \downarrow\mathbf{B} \otimes \overset{n}{\dots} \otimes \downarrow\mathbf{B} \multimap \mathbf{B} \otimes \overset{m}{\dots} \otimes \mathbf{B}$$

- Encoding of **natural numbers**:

$$\begin{aligned}
\overline{0} &\triangleq \lambda fx.\mathtt{discard}_{\mathbf{1}}\, f \,\mathtt{in}\, x \\
\overline{1} &\triangleq \lambda fx.fx \\
\overline{n+2} &\triangleq \lambda fx.\mathtt{copy}^{!}_{\mathbf{1}}\, f \,\mathtt{as}\, f_1 \dots f_n \,\mathtt{in}\, f_1(\dots(f_n\, x)\dots) \\
\mathtt{succ} &\triangleq \lambda nfx.\mathtt{copy}^{!}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, f_1(nf_2x) \\
\mathtt{add} &\triangleq \lambda mnfx.\mathtt{copy}^{!}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, mf_1(nf_2x).
\end{aligned}$$

$$\downarrow(\forall\alpha.(\alpha \multimap \alpha)) \multimap (\forall\alpha.(\alpha \multimap \alpha)) \quad \text{vs} \quad \forall\alpha.(!(\alpha \multimap \alpha) \multimap (\alpha \multimap \alpha))$$

Numerals cannot be used as **iterators** (no topmost $\forall$).

Probabilistic calculi and confluence

- Example: let $\Lambda + \texttt{rand}$, where $\texttt{tt} \leftarrow \texttt{rand} \rightarrow \texttt{ff}$, and let $(\lambda x.x \otimes x)\,\texttt{rand}$:
  - If we evaluate $\texttt{rand}$ first: $\{\texttt{tt} \otimes \texttt{tt}^{\frac{1}{2}}, \texttt{ff} \otimes \texttt{ff}^{\frac{1}{2}}\}$.
  - If we apply β-reduction first: $\{\texttt{tt} \otimes \texttt{tt}^{\frac{1}{4}}, \texttt{tt} \otimes \texttt{ff}^{\frac{1}{4}}, \texttt{ff} \otimes \texttt{tt}^{\frac{1}{4}}, \texttt{ff} \otimes \texttt{ff}^{\frac{1}{4}}\}$.

- Simpson's *linear λ-calculus* $\Lambda_l^!$:

$$M := \; x \mid !M \mid \lambda x.M \; (*) \mid \lambda !x.M \mid MM \qquad (*) \text{ with } x \text{ linear in } M$$

$$(\lambda x.M)N \rightarrow M[N/x] \qquad (\lambda !x.M)!N \;\; \rightarrow M[N/x]$$

Surface reduction: no evaluation inside the scope of the !-operator.
Idea: in $\Lambda_l^! + \texttt{rand}$, $(\lambda !x.x \otimes x)\,!\texttt{rand}$ forces to apply β-reduction.

Probabilistic operational semantics $[\![\cdot]\!]$

- *Big-step approximation* $\Downarrow \subseteq \Lambda_\oplus \times \mathfrak{D}(\mathrm{HEAD})$:

$$\frac{}{M \Downarrow \perp}\, s1 \quad \frac{}{x \Downarrow x}\, s2 \quad \frac{M \Downarrow \mathscr{D}}{\lambda x.M \Downarrow \lambda x.\mathscr{D}}\, s3 \quad \frac{M \Downarrow \mathscr{D} \qquad N \Downarrow \mathscr{E}}{M \oplus N \Downarrow \frac{1}{2} \cdot \mathscr{D} + \frac{1}{2} \cdot \mathscr{E}}\, s4$$

$$\frac{M \Downarrow \mathscr{D} \qquad \{H[N/x] \Downarrow \mathscr{E}_{H,N}\}_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{D})}}{MN \Downarrow \displaystyle\sum_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{D})} \mathscr{D}(\lambda x.H) \cdot \mathscr{E}_{H,N} + \sum_{\substack{H \,\in\, \mathrm{supp}(\mathscr{D}) \\ H \text{ is neutral}}} \mathscr{D}(H) \cdot HN}\, s5$$

- Big-step semantics: $[\![M]\!] := \sup\{\mathscr{D} \mid M \Downarrow \mathscr{D}\}$

Head reduction vs head spine reduction

- Example:

$$\begin{tikzcd}
 & (\lambda x.x)y \arrow[ddrr, "0.5"] & (\lambda x.\mathbf{I})y \arrow[r, "1"] & \mathbf{I} \\
(\lambda x.x \oplus \mathbf{I})y \arrow[ur, "0.5"] \arrow[urr, "0.5"] \arrow[dr, dashed, "1"] & & & \\
 & y \oplus \mathbf{I} \arrow[uurr, dashed, "0.5"] \arrow[rr, dashed, "0.5"] & & y
\end{tikzcd}$$

Theorem (Equivalence)

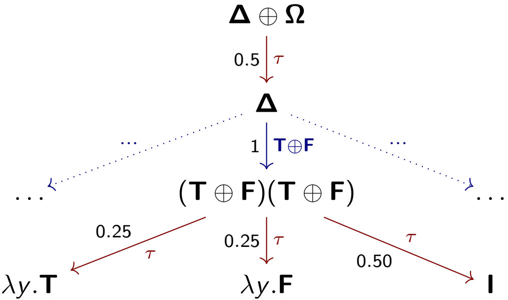
$\forall M \in \Lambda_{\oplus}, \forall H \in \mathrm{HEAD}, \forall n \in \mathbb{N}$: $\mathrm{Prob}^{n}_{head}[M, H] = \mathrm{Prob}^{n}_{spine}[M, H]$.

Labelled Markov Chain (LMC)

- $\Lambda_{\oplus}^{\text{head}}$ as a LMC:

$$\begin{tikzcd} M \arrow[r, "\tau", "p"'] \arrow[rdd, "\tau"', "p'"] & \lambda x.H \\ & \dots \\ & \lambda x.H' \end{tikzcd} \qquad \lambda x.H \xrightarrow[1]{M} H[M/x]$$

- Example: if $\mathbf{T} \triangleq \lambda xy.x$ and $\mathbf{F} \triangleq \lambda xy.y$ then:



Probabilistic Applicative Similarity (PAS)

- *Probabilistic applicative simulation*: $\mathcal{R} =$ preorder relation such that

$$\begin{tikzcd}
M \arrow[r, "\tau", "p"'] \arrow[d, phantom, "\mathcal{R}"] & X \subseteq \mathrm{HEAD} \arrow[d, phantom, "\mathcal{R}"] & & \lambda x.H \arrow[r, "M", "1"'] \arrow[d, phantom, "\mathcal{R}"] & H[M/x] \arrow[d, phantom, "\mathcal{R}"] \\
N \arrow[r, "\tau", "p' \geq p"'] & \mathcal{R}(X) & & \lambda x.H' \arrow[r, "M", "1"'] & H'[M/x]
\end{tikzcd}$$

- Probabilistic applicative similarity (PAS): $\sim =$ the "largest" simulation

$$\begin{tikzcd}
M \arrow[r, "\tau", "p"'] \arrow[d, phantom, "\precsim"] & X \subseteq \mathrm{HEAD} \arrow[d, phantom, "\precsim"] & & \lambda x.H \arrow[r, "M", "1"'] \arrow[d, phantom, "\precsim"] & H[M/x] \arrow[d, phantom, "\precsim"] \\
N \arrow[r, "\tau", "p' \geq p"'] & \precsim(X) & & \lambda x.H' \arrow[r, "M", "1"'] & H'[M/x]
\end{tikzcd}$$