A Categorical Approach to Resource Approximation

Federico Olimpieri

University of Leeds

Outline

Outline

Approximation of Programs

$$\mathcal{L} \ni P, Q \qquad P \rightsquigarrow Q$$

- Potentially *infinite* computational behaviour.

$$P \rightsquigarrow P_1 \rightsquigarrow \cdots \rightsquigarrow P_n \rightsquigarrow \ldots$$

- An auxilliary language with *finitary beahviour*:

$$\mathcal{M} \ni p, q \qquad p \rightsquigarrow q \qquad P = \bigvee_{p \lhd P} p$$

- *Simulating evaluation*:

$$\begin{array}{ccc} P & \rightsquigarrow & Q \\ \triangledown & & \triangledown \\ p & \rightsquigarrow & q \end{array}$$

Approximation of Programs

$$\mathcal{L} \ni P, Q \qquad P \rightsquigarrow Q$$

- Potentially *infinite* computational behaviour.

$$P \rightsquigarrow P_1 \rightsquigarrow \cdots \rightsquigarrow P_n \rightsquigarrow \ldots$$

- An auxilliary language with *finitary beahviour*:

$$\mathcal{M} \ni p, q \qquad p \rightsquigarrow q \qquad P = \bigvee_{p \lhd P} p$$

- *Simulating evaluation*:

$$\begin{array}{ccc} P & \rightsquigarrow & Q \\ \triangledown & & \triangledown \\ p & \rightsquigarrow & q \end{array}$$

Approximation of Programs

$$\mathcal{L} \ni P, Q \qquad P \rightsquigarrow Q$$

- Potentially *infinite* computational behaviour.

$$P \rightsquigarrow P_1 \rightsquigarrow \cdots \rightsquigarrow P_n \rightsquigarrow \ldots$$

- An auxilliary language with *finitary beahviour*:

$$\mathcal{M} \ni p, q \qquad p \rightsquigarrow q \qquad P = \bigvee_{p \lhd P} p$$

- *Simulating evaluation*:

$$\begin{array}{ccc} P & \rightsquigarrow & Q \\ \triangledown & & \triangledown \\ p & \rightsquigarrow & q \end{array}$$

Approximation of Programs

$$\mathcal{L} \ni P, Q \qquad P \rightsquigarrow Q$$

- Potentially *infinite* computational behaviour.

$$P \rightsquigarrow P_1 \rightsquigarrow \cdots \rightsquigarrow P_n \rightsquigarrow \dots$$

- An auxilliary language with *finitary beahviour*:

$$\mathcal{M} \ni p, q \qquad p \rightsquigarrow q \qquad P = \bigvee_{p \lhd P} p$$

- *Simulating evaluation*:

$$\begin{array}{ccc} P & \rightsquigarrow & Q \\ \triangledown & & \triangledown \\ p & \rightsquigarrow & q \end{array}$$

Approximation of Programs

$$\mathcal{L} \ni P, Q \qquad P \rightsquigarrow Q$$

- Potentially *infinite* computational behaviour.

$$P \rightsquigarrow P_1 \rightsquigarrow \cdots \rightsquigarrow P_n \rightsquigarrow \ldots$$

- An auxilliary language with *finitary beahviour*:

$$\mathcal{M} \ni p, q \qquad p \rightsquigarrow q \qquad P = \bigvee_{p \lhd P} p$$

- *Simulating evaluation*:

$$\begin{array}{ccc} P & \rightsquigarrow & Q \\ \triangledown & & \triangledown \\ p & \rightsquigarrow & q \end{array}$$

The λ-Calculus

$$\Lambda \ni M, N ::= x \mid \lambda x.M \mid MN$$

$$(\lambda x.M)N \to M\{N/x\}$$

- $\lambda x.M$ stands for the function $x \mapsto M$.
- The λ-calculus is *Turing complete*.

Example

$$I = \lambda x.x \qquad \Delta = \lambda x.xx \qquad \Omega = \Delta\Delta$$

$$IM \to M \qquad \Delta M \to MM \qquad \Omega \to \Omega$$

The λ-Calculus

$$\Lambda \ni M, N ::= x \mid \lambda x.M \mid MN$$

$$(\lambda x.M)N \to M\{N/x\}$$

- $\lambda x.M$ stands for the function $x \mapsto M$.
- The λ-calculus is *Turing complete*.

Example

$$I = \lambda x.x \qquad \Delta = \lambda x.xx \qquad \Omega = \Delta\Delta$$

$$IM \to M \qquad \Delta M \to MM \qquad \Omega \to \Omega$$

The λ-Calculus

$$\Lambda \ni M, N ::= x \mid \lambda x.M \mid MN$$

$$(\lambda x.M)N \to M\{N/x\}$$

- $\lambda x.M$ stands for the function $x \mapsto M$.
- The λ-calculus is *Turing complete*.

Example

$$I = \lambda x.x \qquad \Delta = \lambda x.xx \qquad \Omega = \Delta\Delta$$

$$IM \to M \qquad \Delta M \to MM \qquad \Omega \to \Omega$$

The λ-Calculus

$$\Lambda \ni M, N ::= x \mid \lambda x.M \mid MN$$

$$(\lambda x.M)N \to M\{N/x\}$$

- $\lambda x.M$ stands for the function $x \mapsto M$.
- The λ-calculus is *Turing complete*.

Example

$$I = \lambda x.x \qquad \Delta = \lambda x.xx \qquad \Omega = \Delta\Delta$$

$$IM \to M \qquad \Delta M \to MM \qquad \Omega \to \Omega$$

Normalization

The computation ends when a *normal form* is reached.

Example

$$((\lambda x.\lambda y.x + y)2)3 \to (\lambda y.2 + y)3 \to 2 + 3 \to 5$$

$$(\lambda x.xx)y \to yy$$

A term is *linear* if it uses its input exaclty once during computation.

Normalization

The computation ends when a *normal form* is reached.

Example

$$((\lambda x.\lambda y.x + y)2)3 \to (\lambda y.2 + y)3 \to 2 + 3 \to 5$$

$$(\lambda x.xx)y \to yy$$

A term is *linear* if it uses its input exaclty once during computation.

Normalization

The computation ends when a *normal form* is reached.

Example

$$((\lambda x.\lambda y.x + y)2)3 \to (\lambda y.2 + y)3 \to 2 + 3 \to 5$$

$$(\lambda x.xx)y \to yy$$

A term is *linear* if it uses its input exaclty once during computation.

Curry and Howard

Types:

$$A ::= \star \mid A \Rightarrow B$$

Typing Derivations:

$$\frac{}{x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \Rightarrow B}$$

$$\frac{\Gamma \vdash M : A \Rightarrow B \qquad \Gamma N : A}{\Gamma \vdash MN : B}$$

Theorem

If M is simply typable then it is s.n.

Curry and Howard

Types:

$$A ::= \star \mid A \Rightarrow B$$

Typing Derivations:

$$\frac{}{x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \Rightarrow B}$$

$$\frac{\Gamma \vdash M : A \Rightarrow B \qquad \Gamma N : A}{\Gamma \vdash MN : B}$$

Theorem

If M is simply typable then it is s.n.

Curry and Howard

Types:

$$A ::= \star \mid A \Rightarrow B$$

Typing Derivations:

$$\frac{}{x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \Rightarrow B}$$

$$\frac{\Gamma \vdash M : A \Rightarrow B \qquad \Gamma N : A}{\Gamma \vdash MN : B}$$

Theorem

If M is simply typable then it is s.n.

Intersection Types

- Multiple typing becomes relevant (Coppo-Dezani 1978):

$$A, B ::= a \mid A \Rightarrow B \mid A \cap B \mid \Omega$$

- $A \cap B$ can be associative, commutative, idempotent.
- When $A \cap A \neq A$ the system becomes *resource sensitive.*
- Very useful: characterizing *normalization properties*, *execution time* . . .

Intersection Types Semantics

$$[\![M]\!] = \{(\Gamma, A) \mid \Gamma \vdash M : A\}$$

$$M \to N \quad \Rightarrow \quad [\![M]\!] = [\![N]\!]$$

Intersection Types

- Multiple typing becomes relevant (Coppo-Dezani 1978):

$$A, B ::= a \mid A \Rightarrow B \mid A \cap B \mid \Omega$$

- $A \cap B$ can be associative, commutative, idempotent.
- When $A \cap A \neq A$ the system becomes *resource sensitive.*
- Very useful: characterizing *normalization properties*, *execution time* ...

Intersection Types Semantics

$$[\![M]\!] = \{(\Gamma, A) \mid \Gamma \vdash M : A\}$$

$$M \to N \quad \Rightarrow \quad [\![M]\!] = [\![N]\!]$$

Intersection Types

- Multiple typing becomes relevant (Coppo-Dezani 1978):

$$A, B ::= a \mid A \Rightarrow B \mid A \cap B \mid \Omega$$

- $A \cap B$ can be associative, commutative, idempotent.
- When $A \cap A \neq A$ the system becomes *resource sensitive.*
- Very useful: characterizing *normalization properties*, *execution time* ...

Intersection Types Semantics

$$[\![M]\!] = \{(\Gamma, A) \mid \Gamma \vdash M : A\}$$

$$M \to N \quad \Rightarrow \quad [\![M]\!] = [\![N]\!]$$

Intersection Types

- Multiple typing becomes relevant (Coppo-Dezani 1978):

$$A, B ::= a \mid A \Rightarrow B \mid A \cap B \mid \Omega$$

- $A \cap B$ can be associative, commutative, idempotent.
- When $A \cap A \neq A$ the system becomes *resource sensitive.*
- Very useful: characterizing *normalization properties*, *execution time* ...

Intersection Types Semantics

$$[\![M]\!] = \{(\Gamma, A) \mid \Gamma \vdash M : A\}$$

$$M \to N \quad \Rightarrow \quad [\![M]\!] = [\![N]\!]$$

Intersection Types

- Multiple typing becomes relevant (Coppo-Dezani 1978):

$$A, B ::= a \mid A \Rightarrow B \mid A \cap B \mid \Omega$$

- $A \cap B$ can be associative, commutative, idempotent.
- When $A \cap A \neq A$ the system becomes *resource sensitive.*
- Very useful: characterizing *normalization properties*, *execution time* ...

Intersection Types Semantics

$$[\![M]\!] = \{(\Gamma, A) \mid \Gamma \vdash M : A\}$$

$$M \to N \quad \Rightarrow \quad [\![M]\!] = [\![N]\!]$$

A Logical Decomposition

Linear Logic (Girard 1980s):

$$A \Rightarrow B = {!A} \multimap B$$

Linear arrow $A \multimap B$ and *exponential modality* !, the linear *conjunction* is the *tensor product* $A \otimes B$.

The exponential as a '*limit*' construction:

$$!A = \lim_{n\to\infty} \overbrace{A \otimes \cdots \otimes A}^{n \text{ times}}$$

A Logical Decomposition

Linear Logic (Girard 1980s):

$$A \Rightarrow B = !A \multimap B$$

Linear arrow $A \multimap B$ and *exponential modality* !, the linear *conjunction* is the *tensor product* $A \otimes B$.

The exponential as a '*limit*' construction:

$$!A = \lim_{n \to \infty} \overbrace{A \otimes \cdots \otimes A}^{n \text{ times}}$$

Outline

A Resource Calculus

Syntax (Ehrhard, Regnier 2008; Mazza et al. 2017)

$$\Lambda_r \ni s, t ::= x \mid \lambda\langle x_1, \ldots, x_k\rangle.s \mid s\langle t_1, \ldots, t_k\rangle \mid 0$$

- $\lambda\vec{x}.0 = 0\vec{t} = s(\vec{t}_1 :: \langle 0\rangle :: \vec{t}_2) = 0.$
- Each variable is assumed to appear at most *once*.
- *Linear Substitution*:

$$s[\vec{t}/\vec{x}] = \begin{cases} s\{\vec{t}/\vec{x}\} & \text{if } \mathrm{len}(\vec{x}) = \mathrm{len}(\vec{t}) \\ 0 & \text{otherwise.} \end{cases}$$

- Resource reduction:

$$(\lambda\vec{x}.s)\vec{t} \to s[\vec{t}/\vec{x}]$$

A Resource Calculus

Syntax (Ehrhard, Regnier 2008; Mazza et al. 2017)

$$\Lambda_r \ni s, t ::= x \mid \lambda\langle x_1, \ldots, x_k\rangle.s \mid s\langle t_1, \ldots, t_k\rangle \mid 0$$

- $\lambda\vec{x}.0 = 0\vec{t} = s(\vec{t}_1 :: \langle 0\rangle :: \vec{t}_2) = 0.$
- Each variable is assumed to appear at most *once*.
- *Linear Substitution*:

$$s[\vec{t}/\vec{x}] = \begin{cases} s\{\vec{t}/\vec{x}\} & \text{if } \mathrm{len}(\vec{x}) = \mathrm{len}(\vec{t}) \\ 0 & \text{otherwise.} \end{cases}$$

- Resource reduction:

$$(\lambda\vec{x}.s)\vec{t} \to s[\vec{t}/\vec{x}]$$

A Resource Calculus

Syntax (Ehrhard, Regnier 2008; Mazza et al. 2017)

$$\Lambda_r \ni s, t ::= x \mid \lambda\langle x_1, \ldots, x_k\rangle.s \mid s\langle t_1, \ldots, t_k\rangle \mid 0$$

- $\lambda\vec{x}.0 = 0\vec{t} = s(\vec{t}_1 :: \langle 0\rangle :: \vec{t}_2) = 0.$
- Each variable is assumed to appear at most *once*.
- *Linear Substitution*:

$$s[\vec{t}/\vec{x}] = \begin{cases} s\{\vec{t}/\vec{x}\} & \text{if } \mathrm{len}(\vec{x}) = \mathrm{len}(\vec{t}) \\ 0 & \text{otherwise.} \end{cases}$$

- Resource reduction:

$$(\lambda\vec{x}.s)\vec{t} \to s[\vec{t}/\vec{x}]$$

A Resource Calculus

Syntax (Ehrhard, Regnier 2008; Mazza et al. 2017)

$$\Lambda_r \ni s, t ::= x \mid \lambda\langle x_1, \ldots, x_k\rangle.s \mid s\langle t_1, \ldots, t_k\rangle \mid 0$$

- $\lambda\vec{x}.0 = 0\vec{t} = s(\vec{t}_1 :: \langle 0\rangle :: \vec{t}_2) = 0.$
- Each variable is assumed to appear at most *once*.
- *Linear Substitution*:

$$s[\vec{t}/\vec{x}] = \begin{cases} s\{\vec{t}/\vec{x}\} & \text{if } \mathrm{len}(\vec{x}) = \mathrm{len}(\vec{t}) \\ 0 & \text{otherwise.} \end{cases}$$

- Resource reduction:

$$(\lambda\vec{x}.s)\vec{t} \to s[\vec{t}/\vec{x}]$$

A Resource Calculus

Syntax (Ehrhard, Regnier 2008; Mazza et al. 2017)

$$\Lambda_r \ni s, t ::= x \mid \lambda\langle x_1, \ldots, x_k\rangle.s \mid s\langle t_1, \ldots, t_k\rangle \mid 0$$

- $\lambda\vec{x}.0 = 0\vec{t} = s(\vec{t}_1 :: \langle 0\rangle :: \vec{t}_2) = 0$.
- Each variable is assumed to appear at most *once*.
- *Linear Substitution*:

$$s[\vec{t}/\vec{x}] = \begin{cases} s\{\vec{t}/\vec{x}\} & \text{if } \mathrm{len}(\vec{x}) = \mathrm{len}(\vec{t}) \\ 0 & \text{otherwise.} \end{cases}$$

- Resource reduction:

$$(\lambda\vec{x}.s)\vec{t} \to s[\vec{t}/\vec{x}]$$

A Finatary Dynamics

$$(\lambda\langle x\rangle.x)\langle t\rangle \to t \qquad (\lambda\langle x,y\rangle.x\langle y\rangle)\langle t,q\rangle \to t\langle q\rangle$$

Failure of computation:

$$(\lambda\langle x,y\rangle.x)\langle t,q\rangle \to 0 \qquad (\lambda\langle x,y\rangle.x\langle y\rangle)\lambda\langle x,y\rangle.x\langle y\rangle \to 0$$

Theorem (Ehrhard, Regnier 2008)

The resource reduction is strongly normalizing.

Proof.

The result is a corollary of resource calculus linearity. □

A Finatary Dynamics

$$(\lambda\langle x\rangle.x)\langle t\rangle \to t \qquad (\lambda\langle x,y\rangle.x\langle y\rangle)\langle t,q\rangle \to t\langle q\rangle$$

Failure of computation:

$$(\lambda\langle x,y\rangle.x)\langle t,q\rangle \to 0 \qquad (\lambda\langle x,y\rangle.x\langle y\rangle)\lambda\langle x,y\rangle.x\langle y\rangle \to 0$$

Theorem (Ehrhard, Regnier 2008)

The resource reduction is strongly normalizing.

Proof.

The result is a corollary of resource calculus linearity. □

A Finatary Dynamics

$$(\lambda\langle x\rangle.x)\langle t\rangle \to t \qquad (\lambda\langle x,y\rangle.x\langle y\rangle)\langle t,q\rangle \to t\langle q\rangle$$

Failure of computation:

$$(\lambda\langle x,y\rangle.x)\langle t,q\rangle \to 0 \qquad (\lambda\langle x,y\rangle.x\langle y\rangle)\lambda\langle x,y\rangle.x\langle y\rangle \to 0$$

Theorem (Ehrhard, Regnier 2008)

The resource reduction is strongly normalizing.

Proof.

The result is a corollary of resource calculus linearity. □

Types for Resource Terms : Variable

Intersection Types:

$$D_A \ni a ::= o \in A \mid a_1 \otimes \cdots \otimes a_k \multimap a$$

Types approximation:

$$o \lhd \star \qquad \frac{a_1 \lhd A \ldots a_k \lhd A \qquad b \lhd B}{a_1 \otimes \cdots \otimes a_k \multimap b \lhd A \Rightarrow B}$$

$$\frac{}{x : a \vdash x : a}$$

Corolla Representation

Types for Resource Terms : Variable

Intersection Types:

$$D_A \ni a ::= o \in A \mid a_1 \otimes \cdots \otimes a_k \multimap a$$

Types approximation:

$$o \lhd \star \qquad \frac{a_1 \lhd A \ldots a_k \lhd A \qquad b \lhd B}{a_1 \otimes \cdots \otimes a_k \multimap b \lhd A \Rightarrow B}$$

$$\frac{}{x : a \vdash x : a}$$

Corolla Representation

Types for Resource Terms : Variable

Intersection Types:

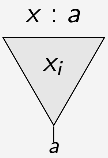
$$D_A \ni a ::= o \in A \mid a_1 \otimes \cdots \otimes a_k \multimap a$$

Types approximation:

$$o \triangleleft \star \qquad \frac{a_1 \triangleleft A \ldots a_k \triangleleft A \qquad b \triangleleft B}{a_1 \otimes \cdots \otimes a_k \multimap b \triangleleft A \Rightarrow B}$$

$$\frac{}{x \triangleleft y : a \triangleleft A \vdash x \triangleleft y : a \triangleleft A}$$

Corolla Representation



Types for Resource Terms: Abstraction

Multilinear λ-abstraction:

$$\frac{\gamma, x_1 : a_1, \ldots, x : a_k \vdash s : a}{\gamma \vdash \lambda\langle x_1, \ldots, x_k \rangle.s : \langle a_1, \ldots, a_k \rangle \multimap a}$$

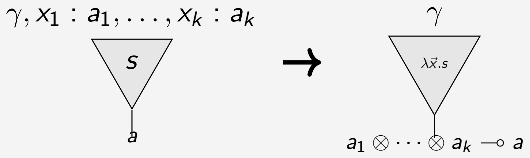
Corolla Representation

Types for Resource Terms: Abstraction

Multilinear λ-abstraction:

$$\frac{\gamma, x_1 : a_1, \dots, x : a_k \vdash s : a}{\gamma \vdash \lambda\langle x_1, \dots, x_k\rangle.s : \langle a_1, \dots, a_k\rangle \multimap a}$$

Corolla Representation



Types for Resource Terms: Application

Multilinear application:

$$\frac{\gamma_0 \vdash s : \langle a_1, \ldots, a_k \rangle \multimap b \qquad (\gamma_1 \vdash t_i : a_i)_{i=1}^{k}}{\gamma_1, \ldots, \gamma_k \vdash s\langle t_1, \ldots, t_k \rangle : b}$$

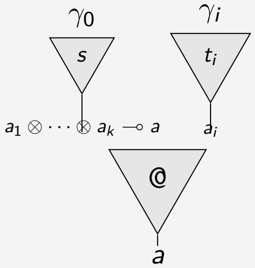
Corolla Representation

Types for Resource Terms: Application

Multilinear application:

$$\frac{\gamma_0 \vdash s \triangleleft M : \langle a_1, \ldots, a_k \rangle \multimap b \triangleleft A \Rightarrow B \qquad (\gamma_1 \vdash t_i \triangleleft N : a_i \triangleleft A)_{i=1}^{k}}{\gamma_1, \ldots, \gamma_k \vdash s\langle t_1, \ldots, t_k \rangle \triangleleft MN : b \triangleleft B}$$

Corolla Representation



Approximation of Ordinary λ-Terms

Approximation judgments (Mazza et al., 2017):

$$\frac{}{y \lhd x \vdash y \lhd x} \qquad \frac{\Gamma, y_1 \lhd x, \ldots, y_k \lhd x \vdash s \lhd M}{\Gamma \vdash \lambda\langle x_1, \ldots, x_k\rangle.s \lhd \lambda x.M}$$

$$\frac{\Gamma_0 \vdash s \lhd M \qquad \Gamma_1 \vdash t_1 \lhd N \ldots \Gamma_k \vdash t_k \lhd M_k}{\Gamma_1, \ldots, \Gamma_k \vdash (s\langle t_1, \ldots, t_k\rangle) \lhd (MN)}$$

Example

$$\delta = (\lambda\langle x_1, x_2\rangle.x_1\langle x_2\rangle) \lhd \Delta = (\lambda x.xx)$$

$$\delta\langle m_1, m_2\rangle \to m_1\langle m_2\rangle \qquad \Delta M \to MM$$

Theorem (cf. De Carvalho, 2008)

$$\mathrm{nf}(s) \neq 0 \iff s \textit{ is typable in } E$$

Approximation of Ordinary λ-Terms

Approximation judgments (Mazza et al., 2017):

$$\frac{}{y \lhd x \vdash y \lhd x} \qquad \frac{\Gamma, y_1 \lhd x, \ldots, y_k \lhd x \vdash s \lhd M}{\Gamma \vdash \lambda\langle x_1, \ldots, x_k\rangle.s \lhd \lambda x.M}$$

$$\frac{\Gamma_0 \vdash s \lhd M \qquad \Gamma_1 \vdash t_1 \lhd N \ldots \Gamma_k \vdash t_k \lhd M_k}{\Gamma_1, \ldots, \Gamma_k \vdash (s\langle t_1, \ldots, t_k\rangle) \lhd (MN)}$$

Example

$$\delta = (\lambda\langle x_1, x_2\rangle.x_1\langle x_2\rangle) \lhd \Delta = (\lambda x.xx)$$

$$\delta\langle m_1, m_2\rangle \to m_1\langle m_2\rangle \qquad \Delta M \to MM$$

Theorem (cf. De Carvalho, 2008)

$$\mathrm{nf}(s) \neq 0 \iff s \textit{ is typable in } E$$

Expansions of Ordinary Terms

- *Untyped* Expansion:

$$\mathsf{T}(M)(n) = \{s \in \Lambda_r \mid \Gamma \vdash s \lhd M \text{ and } \mathrm{len}(\Gamma) = n\}$$

- *Typed* Expansion:

$$\mathsf{T}(M)(\vec{x} : \gamma, a) = \{s \in \Lambda_r \mid \vec{x} : \gamma \vdash s : a \text{ and } \vec{x} \lhd \mathrm{fv}(M) \vdash s \lhd M\}$$

Example

$$T(\lambda x.xx)(0) = \{\lambda\langle x_0, \ldots, x_k\rangle.x_0\langle x_1, \ldots, x_k\rangle \mid k \in \mathbb{N}\}$$

$$\lambda\langle x_0, x_1\rangle.x_0\langle x_1\rangle \approx \lambda x.xx$$

Expansions of Ordinary Terms

- *Untyped* Expansion:

$$\mathsf{T}(M)(n) = \{s \in \Lambda_r \mid \Gamma \vdash s \lhd M \text{ and } \mathsf{len}(\Gamma) = n\}$$

- *Typed* Expansion:

$$\mathsf{T}(M)(\vec{x} : \gamma, a) = \{s \in \Lambda_r \mid \vec{x} : \gamma \vdash s : a \text{ and } \vec{x} \lhd \mathsf{fv}(M) \vdash s \lhd M\}$$

Example

$$T(\lambda x.xx)(0) = \{\lambda\langle x_0, \ldots, x_k\rangle.x_0\langle x_1, \ldots, x_k\rangle \mid k \in \mathbb{N}\}$$

$$\lambda\langle x_0, x_1\rangle.x_0\langle x_1\rangle \approx \lambda x.xx$$

Expansions of Ordinary Terms

- *Untyped* Expansion:

$$\mathsf{T}(M)(n) = \{s \in \Lambda_r \mid \Gamma \vdash s \lhd M \text{ and } \mathrm{len}(\Gamma) = n\}$$

- *Typed* Expansion:

$$\mathsf{T}(M)(\vec{x} : \gamma, a) = \{s \in \Lambda_r \mid \vec{x} : \gamma \vdash s : a \text{ and } \vec{x} \lhd \mathrm{fv}(M) \vdash s \lhd M\}$$

Example

$$T(\lambda x.xx)(0) = \{\lambda\langle x_0, \ldots, x_k\rangle.x_0\langle x_1, \ldots, x_k\rangle \mid k \in \mathbb{N}\}$$

$$\lambda\langle x_0, x_1\rangle.x_0\langle x_1\rangle \approx \lambda x.xx$$

Outline

An Interconnected World

Slogan

The way that objects interact with each other is more important than the objects themselves.

A *category*:

$A \quad B$

C

An Interconnected World

Slogan

The way that objects interact with each other is more important than the objects themselves.

A *category*:

$$A \qquad B$$

$$C$$

An Interconnected World

Slogan

The way that objects interact with each other is more important than the objects themselves.

A *category*:

$A \quad B$

C

An Interconnected World

Slogan

The way that objects interact with each other is more important than the objects themselves.

A *category*:

A B

C

An Interconnected World

Slogan

The way that objects interact with each other is more important than the objects themselves.

A *category*:

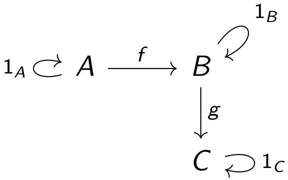
$$
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
 & & \big\downarrow{\scriptstyle g} \\
 & & C
\end{array}
$$

An Interconnected World

Slogan

The way that objects interact with each other is more important than the objects themselves.

A *category*:

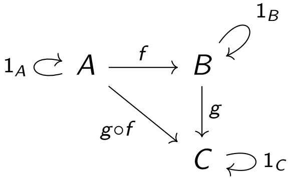


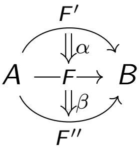
An Interconnected World

Slogan

The way that objects interact with each other is more important than the objects themselves.

A 2-*category*:



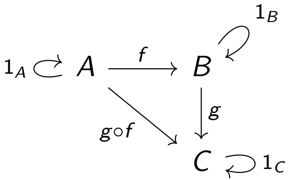


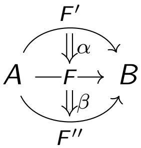
An Interconnected World

Slogan

The way that objects interact with each other is more important than the objects themselves.

A 2-*category*:





$$f \circ 1 = 1 \circ f = f \qquad (h \circ g) \circ f = h \circ (g \circ f)$$

Logic
Formulae
Proofs
Cut-elimination

$$\dfrac{\dfrac{\begin{array}{c}\pi_1\\ \vdots \\ \Gamma, A \vdash B\end{array}}{\Gamma \vdash A \Rightarrow B} \quad \begin{array}{c}\pi_2\\ \vdots \\ \Gamma \vdash A\end{array}}{\Gamma \vdash B} \quad \rightarrow \quad \begin{array}{c}\pi_1\{\pi_2/(A \vdash)\}\\ \vdots \\ \Gamma \vdash B\end{array}$$

Logic
Formulae
Proofs
Cut-elimination

$$\dfrac{\dfrac{\begin{array}{c}\pi_1\\ \vdots\\ \Gamma, A \vdash B\end{array}}{\Gamma \vdash A \Rightarrow B} \qquad \begin{array}{c}\pi_2\\ \vdots\\ \Gamma \vdash A\end{array}}{\Gamma \vdash B} \quad \to \quad \begin{array}{c}\pi_1\{\pi_2/(A\vdash)\}\\ \vdots\\ \Gamma \vdash B\end{array}$$

PL	Logic
Types	Formulae
Programs	Proofs
Evaluation	Cut-elimination

$$\dfrac{\dfrac{\begin{array}{c}\pi_1\\ \vdots\\ \Gamma, x : A \vdash M : B\end{array}}{\Gamma \vdash \lambda x.M : A \Rightarrow B} \qquad \begin{array}{c}\pi_2\\ \vdots\\ \Gamma \vdash N : A\end{array}}{\Gamma \vdash (\lambda x.M)N : B} \quad \to \quad \begin{array}{c}\pi_1\{\pi_2/(A \vdash)\}\\ \vdots\\ \Gamma \vdash M\{N/x\} : B\end{array}$$

PL	Logic	Categories
Types	Formulae	Objects
Programs	Proofs	Morphisms
Evaluation	Cut-elimination	Equality

$$\dfrac{\dfrac{\begin{matrix}\pi_1 \\ \vdots \\ \Gamma, x : A \vdash M : B\end{matrix}}{\Gamma \vdash \lambda x.M : A \Rightarrow B} \qquad \begin{matrix}\pi_2 \\ \vdots \\ \Gamma \vdash N : A\end{matrix}}{\Gamma \vdash (\lambda x.M)N : B} \quad \to \quad \begin{matrix}\pi_1\{\pi_2/(A \vdash)\} \\ \vdots \\ \Gamma \vdash M\{N/x\} : B\end{matrix}$$

Denotational Semantics (Strachey-Scott, 1970s)

$$M \to N \quad \Rightarrow \quad [\![M]\!] = [\![N]\!]$$

Categorical Semantics

- *Product* $A \,\&\, B$ and *Arrow* B^A such that

$$C(\Gamma \,\&\, A, B) \cong C(\Gamma, B^A)$$

$$M \mapsto \lambda x.M$$

- *Interpretation*:

$$[\![A \Rightarrow B]\!] = [\![B]\!]^{[\![A]\!]} \qquad [\![A_1, \dots, A_n]\!] = [\![A_1]\!] \,\&\, \cdots \,\&\, [\![A_n]\!]$$

$$\Gamma \vdash M : A \qquad [\![M]\!] : [\![\Gamma]\!] \to [\![A]\!]$$

- *2-dimensional*:

$$[\![M \to N]\!] = \beta : [\![M]\!] \to [\![N]\!]$$

Categorical Semantics

- *Product* $A \,\&\, B$ and *Arrow* B^A such that

$$C(\Gamma \,\&\, A, B) \cong C(\Gamma, B^A)$$

$$M \mapsto \lambda x.M$$

- *Interpretation*:

$$[\![A \Rightarrow B]\!] = [\![B]\!]^{[\![A]\!]} \qquad [\![A_1, \ldots, A_n]\!] = [\![A_1]\!] \,\&\, \cdots \,\&\, [\![A_n]\!]$$

$$\Gamma \vdash M : A \qquad [\![M]\!] : [\![\Gamma]\!] \to [\![A]\!]$$

- *2-dimensional*:

$$[\![M \to N]\!] = \beta : [\![M]\!] \to [\![N]\!]$$

Categorical Semantics

- *Product* $A \,\&\, B$ and *Arrow* B^A such that

$$C(\Gamma \,\&\, A, B) \cong C(\Gamma, B^A)$$

$$M \mapsto \lambda x.M$$

- *Interpretation*:

$$[\![A \Rightarrow B]\!] = [\![B]\!]^{[\![A]\!]} \qquad [\![A_1, \ldots, A_n]\!] = [\![A_1]\!] \,\&\, \cdots \,\&\, [\![A_n]\!]$$

$$\Gamma \vdash M : A \qquad [\![M]\!] : [\![\Gamma]\!] \to [\![A]\!]$$

- *2-dimensional*:

$$[\![M \to N]\!] = \beta : [\![M]\!] \to [\![N]\!]$$

A Simple Model of λ-calculus

- The category of sets and relations (Rel).

$$\mathrm{ob}(\mathrm{Rel}) = \mathrm{ob}(\mathrm{Set}) \qquad \mathrm{Rel}(X, Y) = \wp(X \times Y)$$

- Arrow type:

$$A \Rightarrow B = !A \multimap B = \mathsf{Multisets}(A) \times B$$

- Syntactic presentation *via* *intersection types*.

$$a_1 \cap \cdots \cap a_k \multimap b := \langle [a_1, \ldots, a_k], b \rangle$$

Daniel de Carvalho. "Semantique de la logique lineaire et temps de calcul". In: PhD thesis, Aix-Marseille Université, 2007

A Simple Model of λ-calculus

- The category of sets and relations (Rel).

$$\mathrm{ob}(\mathrm{Rel}) = \mathrm{ob}(\mathrm{Set}) \qquad \mathrm{Rel}(X, Y) = \wp(X \times Y)$$

- Arrow type:

$$A \Rightarrow B = !A \multimap B = \mathsf{Multisets}(A) \times B$$

- Syntactic presentation *via* *intersection types*.

$$a_1 \cap \cdots \cap a_k \multimap b := \langle [a_1, \ldots, a_k], b \rangle$$

Daniel de Carvalho. "Semantique de la logique lineaire et temps de calcul". In: PhD thesis, Aix-Marseille Université, 2007

A Simple Model of λ-calculus

- The category of sets and relations (Rel).

$$\mathrm{ob}(\mathrm{Rel}) = \mathrm{ob}(\mathrm{Set}) \qquad \mathrm{Rel}(X, Y) = \wp(X \times Y)$$

- Arrow type:

$$A \Rightarrow B = !A \multimap B = \mathsf{Multisets}(A) \times B$$

- Syntactic presentation *via* *intersection types*.

$$a_1 \cap \cdots \cap a_k \multimap b := \langle [a_1, \ldots, a_k], b \rangle$$

Daniel de Carvalho. "Semantique de la logique lineaire et temps de calcul". In: PhD thesis, Aix-Marseille Université, 2007

Proof-Irrelevance of Relations

$$[\![M]\!](\delta, a) = \begin{cases} 1 & \text{if } \delta \vdash M : a \\ 0 & \text{otherwise.} \end{cases}$$

The structure

$$(\!|M|\!)(\Delta, a) = \left\{ \begin{array}{c} \pi \\ \vdots \\ \delta \vdash M : a \end{array} \right\}$$

is neither a relation nor a *denotational semantics*.
Reasoning on type derivations happens *outside* the model.

Proof-Irrelevance of Relations

$$[\![M]\!](\delta, a) = \begin{cases} 1 & \text{if } \delta \vdash M : a \\ 0 & \text{otherwise.} \end{cases}$$

The structure

$$(\!|M|\!)(\Delta, a) = \left\{ \begin{array}{c} \pi \\ \vdots \\ \delta \vdash M : a \end{array} \right\}$$

is neither a relation nor a *denotational semantics*.

Reasoning on type derivations happens *outside* the model.

Proof-Irrelevance of Relations

$$[\![M]\!](\delta, a) = \begin{cases} 1 & \text{if } \delta \vdash M : a \\ 0 & \text{otherwise.} \end{cases}$$

The structure

$$(\!|M|\!)(\Delta, a) = \left\{ \begin{array}{c} \pi \\ \vdots \\ \delta \vdash M : a \end{array} \right\}$$

is neither a relation nor a *denotational semantics*.
Reasoning on type derivations happens *outside* the model.

What We Want

Set-Theoretic	Category-Theoretic
sets	categories
functions	functors
equations	(natural) isomorphisms

Relations $\Rightarrow$ Distributors

$$[\![M]\!](\delta, a) = \begin{cases} 1 & \text{if } \delta \vdash M : a \\ 0 & \text{otherwise.} \end{cases} \quad \rightsquigarrow \quad [\![M]\!](\Delta, a) \cong \mathsf{T}(M)(\delta, a)$$

What We Want

Set-Theoretic	Category-Theoretic
sets	categories
functions	functors
equations	(natural) isomorphisms

Relations $\Rightarrow$ **Distributors**

$$\llbracket M \rrbracket(\delta, a) = \begin{cases} 1 & \text{if } \delta \vdash M : a \\ 0 & \text{otherwise.} \end{cases} \quad \rightsquigarrow \quad \llbracket M \rrbracket(\Delta, a) \cong \mathsf{T}(M)(\delta, a)$$

Distributors

Relations

$$f \subseteq A \times B \qquad f : A \times B \to \{0,1\}$$

We could generalize a little bit...

Distributors, aka Profunctors

$$F : A^{op} \times B \to \mathrm{Set}$$

and achieve *proof-relevant* relations!

$F(a,b) \in \mathrm{Set}$ **the set of "witnesses" for** aFb

Distributors

Relations

$$f \subseteq A \times B \qquad f : A \times B \to \{0,1\}$$

We could generalize a little bit...

Distributors, aka Profunctors

$$F : A^{op} \times B \to \mathrm{Set}$$

and achieve *proof-relevant* relations!

$F(a,b) \in \mathrm{Set}$ **the set of "witnesses" for** aFb

Distributors

Relations

$$f \subseteq A \times B \qquad f : A \times B \to \{0, 1\}$$

We could generalize a little bit...

Distributors, aka Profunctors

$$F : A^{op} \times B \to \mathrm{Set}$$

and achieve *proof-relevant* relations!

$F(a, b) \in \mathrm{Set}$ **the set of "witnesses" for** aFb

Between Syntax and Semantics

We can interpret λ-terms via distributors.

$$[\![M]\!] : [\![\Gamma]\!] \to [\![A]\!] \qquad [\![M]\!] : ![\![\Gamma]\!]^{op} \times [\![A]\!] \to \mathrm{Set}$$

$$!A \quad \approx \quad \textbf{category of finite lists over } A$$

Marcelo Fiore et al. "The cartesian closed bicategory of generalised species of structures". In: *Journal of the London Mathematical Society* (2008)

This interpretation is given by *Expansion*:

Theorem (O., 2021)

$$[\![M]\!](\gamma, a) = \{s \in \Lambda_r \mid \gamma \vdash s : a\}$$

Theorem (O. 2021, Dynamic Denotational Semantics)

$$M \to N \qquad \mathrm{T}(M) \cong \mathrm{T}(N)$$

Between Syntax and Semantics

We can interpret λ-terms via distributors.

$$[\![M]\!] : [\![\Gamma]\!] \to [\![A]\!] \qquad [\![M]\!] : ![\![\Gamma]\!]^{op} \times [\![A]\!] \to \mathrm{Set}$$

$$!A \quad \approx \quad \textbf{category of finite lists over } A$$

Marcelo Fiore et al. "The cartesian closed bicategory of generalised species of structures". In: *Journal of the London Mathematical Society* (2008)

This interpretation is given by *Expansion*:

Theorem (O., 2021)

$$[\![M]\!](\gamma, a) = \{s \in \Lambda_r \mid \gamma \vdash s : a\}$$

Theorem (O. 2021, Dynamic Denotational Semantics)

$$M \to N \qquad \mathrm{T}(M) \cong \mathrm{T}(N)$$

Between Syntax and Semantics

We can interpret λ-terms via distributors.

$$[\![M]\!] : [\![\Gamma]\!] \to [\![A]\!] \qquad [\![M]\!] : ![\![\Gamma]\!]^{op} \times [\![A]\!] \to \mathrm{Set}$$

$$!A \quad \approx \quad \textbf{category of finite lists over } A$$

Marcelo Fiore et al. "The cartesian closed bicategory of generalised species of structures". In: *Journal of the London Mathematical Society* (2008)

This interpretation is given by *Expansion*:

Theorem (O., 2021)

$$[\![M]\!](\gamma, a) = \{s \in \Lambda_r \mid \gamma \vdash s : a\}$$

Theorem (O. 2021, Dynamic Denotational Semantics)

$$M \to N \qquad \mathrm{T}(M) \cong \mathrm{T}(N)$$

What We Get

Categorification is about making things *explicit*.

- The possibility to exploit general categorical techniques to study the syntactic theory of approximation.
- Refinement and improvement of resource approximation and relational semantics.
- General semantic theory of resource calculi.
- A *dynamic* denotational semantics:

$$M \to^* N \quad \Rightarrow \quad \beta_{M,N} : [\![M]\!] \cong [\![N]\!]$$

$$\begin{array}{ccc} M & \to^* & N \\ \triangledown & & \triangledown \\ s & \to^* & t \end{array}$$

What We Get

Categorification is about making things *explicit*.

- The possibility to exploit general categorical techniques to study the syntactic theory of approximation.
- Refinement and improvement of resource approximation and relational semantics.
- General semantic theory of resource calculi.
- A *dynamic* denotational semantics:

$$M \to^* N \quad \Rightarrow \quad \beta_{M,N} : [\![M]\!] \cong [\![N]\!]$$

$$\begin{array}{ccc} M & \to^* & N \\ \triangledown & & \triangledown \\ s & \to^* & t \end{array}$$

What We Get

Categorification is about making things *explicit*.

- The possibility to exploit general categorical techniques to study the syntactic theory of approximation.
- Refinement and improvement of resource approximation and relational semantics.
- General semantic theory of resource calculi.
- A *dynamic* denotational semantics:

$$M \to^* N \quad \Rightarrow \quad \beta_{M,N} : [\![M]\!] \cong [\![N]\!]$$

$$\begin{array}{ccc} M & \to^* & N \\ \triangledown & & \triangledown \\ s & \to^* & t \end{array}$$

What We Get

Categorification is about making things *explicit*.

- The possibility to exploit general categorical techniques to study the syntactic theory of approximation.
- Refinement and improvement of resource approximation and relational semantics.
- General semantic theory of resource calculi.
- A *dynamic* denotational semantics:

$$M \to^* N \quad \Rightarrow \quad \beta_{M,N} : [\![M]\!] \cong [\![N]\!]$$

$$\begin{array}{ccc} M & \to^* & N \\ \triangledown & & \triangledown \\ s & \to^* & t \end{array}$$

What We Get

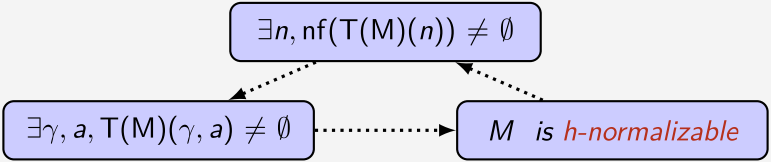
Categorification is about making things *explicit*.

- The possibility to exploit general categorical techniques to study the syntactic theory of approximation.
- Refinement and improvement of resource approximation and relational semantics.
- General semantic theory of resource calculi.
- A *dynamic* denotational semantics:

$$M \to^* N \quad \Rightarrow \quad \beta_{M,N} : [\![M]\!] \cong [\![N]\!]$$

$$\begin{array}{ccc} M & \to^* & N \\ \triangledown & & \triangledown \\ s & \to^* & t \end{array}$$

Theorem (Qualitative Head-Normalization)



The *execution time* of λ-terms:

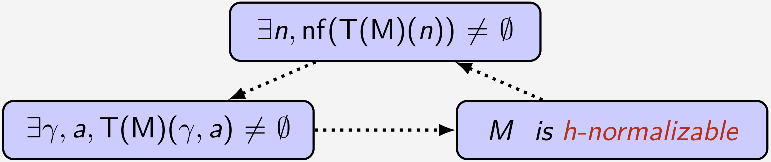
Theorem (cf. de Carvalho 2008)

There exists $s \in \mathrm{nf}(T(M))$ *s.t.*

$$\mathrm{len}(\mathrm{execution}(M)) = \mathrm{size}(s).$$

Studying λ-Terms *via* Their Expansions

Theorem (Qualitative Head-Normalization)



The *execution time* of λ-terms:

Theorem (cf. de Carvalho 2008)

There exists $s \in \mathrm{nf}(T(M))$ *s.t.*

$$\mathrm{len}(\mathrm{execution}(M)) = \mathrm{size}(s).$$

Outline

Towards a Categorical Theory of Approximation

A syntax-independent theory of resource approximation by the means of category theory.

Damiano Mazza. "An Axiomatic Notion of Approximation for Programming Languages and Machines". 2021

$$\mathrm{App}_{\mathcal{L},\mathcal{M}} : \mathrm{A} \to \mathcal{M} \times \mathcal{L}$$

$$s \lhd M \mapsto \langle s, M \rangle$$

Main goals:

- Achieve a categorical definition of *approximation system*.
- study necessary and sufficient conditions to get 'once and for all' normalization theorems and quantitative characterization.
- New kinds of approximations.

Towards a Categorical Theory of Approximation

A syntax-independent theory of resource approximation by the means of category theory.

Damiano Mazza. "An Axiomatic Notion of Approximation for Programming Languages and Machines". 2021

$$\mathsf{App}_{\mathcal{L},\mathcal{M}} : \mathsf{A} \to \mathcal{M} \times \mathcal{L}$$

$$s \lhd M \mapsto \langle s, M \rangle$$

Main goals:

- Achieve a categorical definition of *approximation system*.
- study necessary and sufficient conditions to get 'once and for all' normalization theorems and quantitative characterization.
- New kinds of approximations.

Towards a Categorical Theory of Approximation

A syntax-independent theory of resource approximation by the means of category theory.

Damiano Mazza. "An Axiomatic Notion of Approximation for Programming Languages and Machines". 2021

$$\mathsf{App}_{\mathcal{L},\mathcal{M}} : \mathsf{A} \to \mathcal{M} \times \mathcal{L}$$

$$s \lhd M \mapsto \langle s, M \rangle$$

Main goals:

- Achieve a categorical definition of *approximation system*.
- study necessary and sufficient conditions to get 'once and for all' normalization theorems and quantitative characterization.
- New kinds of approximations.

Towards a Categorical Theory of Approximation

A syntax-independent theory of resource approximation by the means of category theory.

Damiano Mazza. “An Axiomatic Notion of Approximation for Programming Languages and Machines”. 2021

$$\mathsf{App}_{\mathcal{L},\mathcal{M}} : \mathsf{A} \to \mathcal{M} \times \mathcal{L}$$

$$s \lhd M \mapsto \langle s, M \rangle$$

Main goals:

- Achieve a categorical definition of *approximation system*.
- study necessary and sufficient conditions to get 'once and for all' normalization theorems and quantitative characterization.
- New kinds of approximations.

Towards a Categorical Theory of Approximation

A syntax-independent theory of resource approximation by the means of category theory.

Damiano Mazza. "An Axiomatic Notion of Approximation for Programming Languages and Machines". 2021

$$\mathsf{App}_{\mathcal{L},\mathcal{M}} : \mathsf{A} \to \mathcal{M} \times \mathcal{L}$$

$$s \lhd M \mapsto \langle s, M \rangle$$

Main goals:

- Achieve a categorical definition of *approximation system*.
- study necessary and sufficient conditions to get 'once and for all' normalization theorems and quantitative characterization.
- New kinds of approximations.

Approximation and Games

Game semantics is a powerful setting for the study of programming languages.

main goals:

- Proper formal understanding of the relationship between approximation semantics and game semantics.
- Concurrent games and distributors Pierre Clairambault and Hugo Paquet. "The Quantitative Collapse of Concurrent Games with Symmetry". In: *CoRR* abs/2107.03155 (2021)
- Template games: Paul-André Melliès. "Template games and differential linear logic". In: *LICS*. 2019

Approximation and Games

Game semantics is a powerful setting for the study of programming languages.
main goals:

- Proper formal understanding of the relationship between approximation semantics and game semantics.
- Concurrent games and distributors Pierre Clairambault and Hugo Paquet. "The Quantitative Collapse of Concurrent Games with Symmetry". In: *CoRR* abs/2107.03155 (2021)
- Template games: Paul-André Melliès. "Template games and differential linear logic". In: *LICS*. 2019

Approximation and Games

Game semantics is a powerful setting for the study of programming languages.
main goals:

- Proper formal understanding of the relationship between approximation semantics and game semantics.
- Concurrent games and distributors Pierre Clairambault and Hugo Paquet. "The Quantitative Collapse of Concurrent Games with Symmetry". In: *CoRR* abs/2107.03155 (2021)
- Template games: Paul-André Melliès. "Template games and differential linear logic". In: *LICS*. 2019

Approximation and Games

Game semantics is a powerful setting for the study of programming languages.
main goals:

- Proper formal understanding of the relationship between approximation semantics and game semantics.
- Concurrent games and distributors Pierre Clairambault and Hugo Paquet. "The Quantitative Collapse of Concurrent Games with Symmetry". In: *CoRR* abs/2107.03155 (2021)
- Template games: Paul-André Melliès. "Template games and differential linear logic". In: *LICS*. 2019

Complexity Results

Resource approximation has been exploited to obtain quantitative results over the λ-calculus.

Daniel de Carvalho. "Execution time of λ-terms via denotational semantics and intersection types". In: *Math. Struct. Comput. Sci.* (2018)

Main goals:

- Categorical quantitative study of reduction.
- Mazza has extended these notion to the framework of *Turing Machine*. **Results**: a new proof of Cook-Levin Theorem.
  Damiano Mazza. "Church Meets Cook and Levin". In: *LICS*. 2016
- Characterization of complexity classes.

Complexity Results

Resource approximation has been exploited to obtain quantitative results over the λ-calculus.

Daniel de Carvalho. "Execution time of λ-terms via denotational semantics and intersection types". In: *Math. Struct. Comput. Sci.* (2018)

Main goals:

- Categorical quantitative study of reduction.
- Mazza has extended these notion to the framework of *Turing Machine*. **Results**: a new proof of Cook-Levin Theorem.
  Damiano Mazza. "Church Meets Cook and Levin". In: *LICS*. 2016
- Characterization of complexity classes.

Complexity Results

Resource approximation has been exploited to obtain quantitative results over the λ-calculus.

Daniel de Carvalho. "Execution time of λ-terms via denotational semantics and intersection types". In: *Math. Struct. Comput. Sci.* (2018)

Main goals:

- Categorical quantitative study of reduction.
- Mazza has extended these notion to the framework of *Turing Machine*. **Results**: a new proof of Cook-Levin Theorem.
  Damiano Mazza. "Church Meets Cook and Levin". In: *LICS*. 2016
- Characterization of complexity classes.

Complexity Results

Resource approximation has been exploited to obtain quantitative results over the λ-calculus.

Daniel de Carvalho. "Execution time of λ-terms via denotational semantics and intersection types". In: *Math. Struct. Comput. Sci.* (2018)

Main goals:

- Categorical quantitative study of reduction.
- Mazza has extended these notion to the framework of *Turing Machine*. **Results**: a new proof of Cook-Levin Theorem.

  Damiano Mazza. "Church Meets Cook and Levin". In: *LICS*. 2016
- Characterization of complexity classes.

Computation with Effects

Development of approximation theory for extended notions of computation.

- A categorical approximation theory for probabilistic and non-deterministic computation (relationship with *probabilistic coherence spaces*).
- Possible application for process calculi (cf. Mazza et al., 2019)
- Consider the cases of Moggi's computational λ-calculus and modal λ-calculi (cf. Pfenning's approach).

Computation with Effects

Development of approximation theory for extended notions of computation.

- A categorical approximation theory for probabilistic and non-deterministic computation (relationship with *probabilistic coherence spaces*).
- Possible application for process calculi (cf. Mazza et al., 2019)
- Consider the cases of Moggi's computational λ-calculus and modal λ-calculi (cf. Pfenning's approach).

Computation with Effects

Development of approximation theory for extended notions of computation.

- A categorical approximation theory for probabilistic and non-deterministic computation (relationship with *probabilistic coherence spaces*).
- Possible application for process calculi (cf. Mazza et al., 2019)
- Consider the cases of Moggi's computational λ-calculus and modal λ-calculi (cf. Pfenning's approach).

Computation with Effects

Development of approximation theory for extended notions of computation.

- A categorical approximation theory for probabilistic and non-deterministic computation (relationship with *probabilistic coherence spaces*).
- Possible application for process calculi (cf. Mazza et al., 2019)
- Consider the cases of Moggi's computational λ-calculus and modal λ-calculi (cf. Pfenning's approach).

Thank You!