LAMBDA CALCULI THROUGH THE LENS OF LINEAR LOGIC

Lesson 3: **Quantitative Types**

Delia KESNER
Email : kesner@irif.fr
URL : www.irif.fr/~kesner

Université Paris Cité and CNRS

Motivations

- **Quantitative** information in computer science:
  - ▷ Time, space, probability, cost, ...
- Emerging in **different areas**:
  - ▷ Automata, graphs, logics, algorithms, ...
  - ▷ Verification, model-checking, programming, theorem proving, ...
  - ▷ Performance measurement, network analysis, data mining, ...
- **Theory of programming languages:**
  - ▷ Quantitative information about programs can be captured by **type systems/relational models**
- **Quantitative Type Systems**:
  - ▷ **Principles**, **Properties**, and **Applications**.

Outline

Agenda

From Simple Types to Quantitative Types

Simple Types - Main Ideas

- **Grammar**: $\mathtt{A}, \mathtt{B} ::= \iota \mid \mathtt{A} \rightarrow \mathtt{B}$
- **Typing rules**:

$$\frac{}{\Gamma, x : \mathtt{A} \vdash x : \mathtt{A}} \qquad \frac{\Gamma, x : \mathtt{A} \vdash t : \mathtt{B}}{\Gamma \vdash \lambda x.t : \mathtt{A} \rightarrow \mathtt{B}} \qquad \frac{\Gamma \vdash t : \mathtt{A} \rightarrow \mathtt{B} \qquad \Gamma \vdash u : \mathtt{A}}{\Gamma \vdash tu : \mathtt{B}}$$

- **Logical** (Curry-Howard) interpretation.
- **Monomorphic** information.
- Lack expressivity power but **typability is decidable**.
- Admits powerful polymorphic **extensions** that are still decidable.

Intersection Types - Main Ideas

- **Grammar** : $\mathtt{A}, \mathtt{B} ::= \iota \mid \mathtt{A} \to \mathtt{B} \mid \mathtt{A} \cap \mathtt{B}$
- **Key typing rule**:

$$\frac{t : \mathtt{A} \qquad t : \mathtt{B}}{t : \mathtt{A} \cap \mathtt{B}}$$

- **Finite polymorphism**:

$$(\mathtt{A} \to \mathtt{A}) \cap ((\mathtt{A} \to \mathtt{B}) \to (\mathtt{A} \to \mathtt{B}))$$

represent two different instances of the polymorphic type

$$\forall X.X \to X.$$

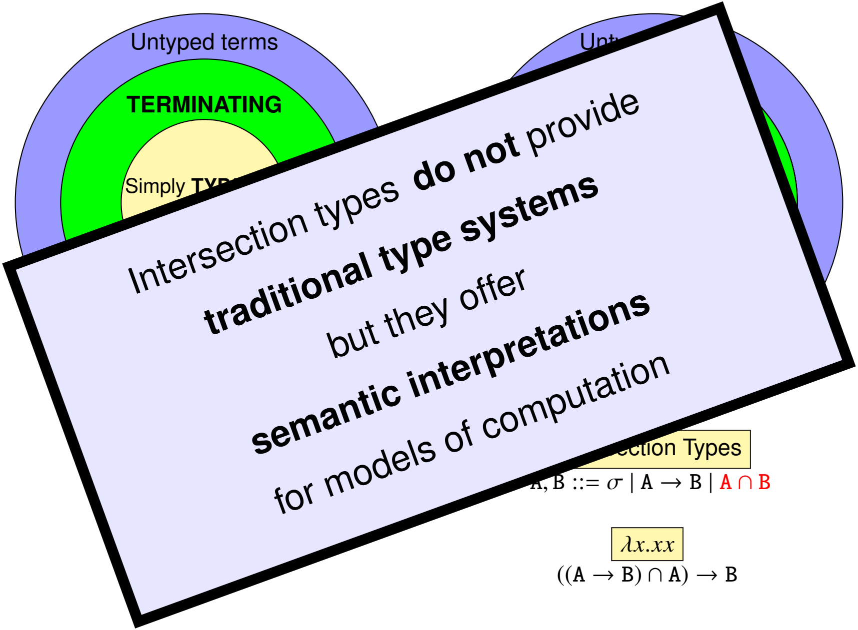
- But **more expressive**:

$$\mathtt{A} \cap (\mathtt{A} \to \mathtt{B}) \cap ((\mathtt{A} \to \mathtt{B}) \to \mathtt{B})$$

is perfectly admissible.

- Typability becomes **undecidable**
- Are **flexible** and can be adapted to different frameworks
- Provide **models** for different languages
- Very **powerful** tool to reason about properties of higher-order languages

Simple versus Intersection Types



Which kind of intersection constructor?

Associativity $(A \cap B) \cap C \;\sim\; A \cap (B \cap C)$

Commutativity $A \cap B \;\sim\; B \cap A$

Idempotent versus **Non-idempotent**

$A \cap A \sim A$ $\qquad$ $A \cap A \not\sim A$

Unbounded Resources $\qquad$ **Finite Resources**

Idempotent vs Non-Idempotent Intersection Types

Idempotent	**Non-idempotent**
Coppo & Dezani in the eighties	**Gardner** and **Kfoury** in the nineties (**Girard**'s Linear Logic flavour)
Sets: $A \cap A \cap C$ is $\{A, C\}$	**Multi-sets:** $A \cap A \cap C$ **is** $[A, A, C]$
Qualitative properties: Yes or No  	**Quantitative** properties: bound and measure **De Carvalho** 

Agenda

An Emblematic Example: Gardner's system revisited

Grammar:

$$\begin{array}{lrcl} \textbf{(Types)} & \mathtt{A} & ::= & \iota \mid \mathtt{M} \to \mathtt{A} \\ \textbf{(Multi-Types)} & \mathtt{M} & ::= & [\mathtt{A}_i]_{i \in I} \end{array}$$

(Quantitative) Linear Logic Intuition:

- $[\mathtt{A}_1, \ldots, \mathtt{A}_n]$ can be understood as $\mathtt{A}_1 \otimes \ldots \otimes \mathtt{A}_n$.
- $\mathtt{M} \to \mathtt{A}$ can be understood as $\mathtt{M}^{\perp} \parr \mathtt{A}$

Judgements:

Multi-Type $\mathtt{M}_i$ for each variable x_i

$$x_1 : \mathtt{M}_1, \ldots, x_n : \mathtt{M}_n \vdash \mathbf{t} : \mathtt{A}$$

Type $\mathtt{A}$ for the term $\mathbf{t}$

The Typing Rules:

Type System $\mathcal{H}$ ($\mathcal{H}$ for $\mathcal{H}$ead)

$$\frac{}{x : [\mathtt{A}] \vdash x : \mathtt{A}}\ (\mathtt{ax})$$

$$\frac{\Gamma \vdash t : \mathtt{A}}{\Gamma \setminus x \vdash \lambda x.t : \Gamma(x) \to \mathtt{A}}\ (\mathtt{fun})$$

$$\frac{(\Gamma_i \vdash t : \mathtt{A}_i)_{i \in I}}{\sqcup_{i \in I} \Gamma_i \vdash t : [\mathtt{A}_i]_{i \in I}}\ (\mathtt{many})$$

$$\frac{\Gamma \vdash t : \mathtt{M} \to \mathtt{B} \qquad \Delta \vdash u : \mathtt{M}}{\Gamma \sqcup \Delta \vdash tu : \mathtt{B}}\ (\mathtt{app})$$

$$\frac{}{x : [\mathtt{A}] \vdash x : \mathtt{A}}\ (\mathtt{ax}) \qquad \frac{\Gamma \vdash t : \mathtt{A}}{\Gamma \setminus x \vdash \lambda x.t : \Gamma(x) \to \mathtt{A}}\ (\to_{\mathtt{i}})$$

$$\frac{(\Gamma_i \vdash t : \mathtt{A}_i)_{i \in I}}{\sqcup_{i \in I} \Gamma_i \vdash t : [\mathtt{A}_i]_{i \in I}}\ (\mathtt{many}) \qquad \frac{\Gamma \vdash t : \mathtt{M} \to \mathtt{B} \qquad \Delta \vdash u : \mathtt{M}}{\Gamma \sqcup \Delta \vdash tu : \mathtt{B}}\ (\to_{\mathtt{e}})$$

Type System $\mathcal{H}$ with a Single Counter

$$\frac{}{x : [\mathtt{A}] \vdash^{(1)} x : \mathtt{A}}\ (\mathtt{ax}) \qquad \frac{\Gamma \vdash^{(C)} t : \mathtt{A}}{\Gamma \setminus x \vdash^{(C+1)} \lambda x.t : \Gamma(x) \rightarrow \mathtt{A}}\ (\rightarrow_{\mathtt{i}})$$

$$\frac{(\Gamma_i \vdash^{(C_i)} t : \mathtt{A}_i)_{i\in I}}{\sqcup_{i\in I} \Gamma_i \vdash^{(+_{i\in I} C_i)} t : [\mathtt{A}_i]_{i\in I}}\ (\mathtt{many}) \qquad \frac{\Gamma \vdash^{(C_1)} t : \mathtt{M} \rightarrow \mathtt{B} \qquad \Delta \vdash^{(C_2)} u : \mathtt{M}}{\Gamma \sqcup \Delta \vdash^{(C_1+C_2+1)} tu : \mathtt{B}}\ (\rightarrow_{\mathtt{e}})$$

- The single **counter** computes the number of typing rules different from ($\mathtt{many}$).
- Other kind of **counters** will be used later.
- **Counters** can be also added to idempotent systems, but they result useless (as we will see).

(Standard) Notation for Type Derivability

$$\Pi \triangleright_{\mathcal{S}} \Gamma \vdash^{(\mathtt{C}_1,\ldots,\mathtt{C}_n)} \mathbf{t} : \mathtt{A}$$

- Π is a **(tree) derivation**, (sometimes omitted)
- $\mathcal{S}$ is a **type system**, (sometimes omitted)
- Γ is a set of **type declarations**,
- $(\mathtt{C}_1, \ldots, \mathtt{C}_n)$ are **counters**. (sometimes omitted)
- $\mathbf{t}$ is a **program/term**,
- $\mathtt{A}$ is a **type**.

A First Example

Let $\Omega := (\lambda x.xx)(\lambda x.xx)$. Let $\mathtt{A} := [\,] \to [\iota_0] \to \iota_1$. Then,

$$\dfrac{\dfrac{\dfrac{}{x : [\mathtt{A}] \vdash x : \mathtt{A}}\ (\mathtt{ax}) \qquad \dfrac{}{\vdash \Omega : [\,]}\ (\mathtt{many})}{x : [\mathtt{A}] \vdash x\Omega : [\iota_0] \to \iota_1}\ (\mathtt{app}) \qquad \dfrac{\dfrac{}{x : [\iota_0] \vdash x : \iota_0}\ (\mathtt{ax})}{x : [\iota_0] \vdash x : [\iota_0]}\ (\mathtt{many})}{\dfrac{x : [\mathtt{A}, \iota_0] \vdash x\ \Omega\ x : \iota_1}{\vdash \lambda x.x\ \Omega\ x : [\mathtt{A}, \iota_0] \to \iota_1}\ (\mathtt{fun})}\ (\mathtt{app})$$

- The subterm Ω is **untyped**
- The bound variable x is typed with an intersection type $[\mathtt{A}, \iota_0]$.

Another Example: Church Numerals

Let $\underline{\mathbf{3}} := \lambda f.\lambda x.f(f(f x))$ and let $\mathtt{B} := [\mathtt{A}] \to \mathtt{A}$.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f : \mathtt{B} \vdash f : [\mathtt{A}] \to \mathtt{A}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f : \mathtt{B} \vdash f : [\mathtt{A}] \to \mathtt{A}} \qquad \dfrac{\dfrac{\dfrac{}{f : \mathtt{B} \vdash f : [\mathtt{A}] \to \mathtt{A}} \qquad \dfrac{\dfrac{}{x : [\mathtt{A}] \vdash x : \mathtt{A}}}{x : [\mathtt{A}] \vdash x : [\mathtt{A}]}}{f : [\mathtt{B}], x : [\mathtt{A}] \vdash f x : \mathtt{A}}}{f : [\mathtt{B}], x : [\mathtt{A}] \vdash f x : [\mathtt{A}]}}{f : [\mathtt{B}, \mathtt{B}], x : [\mathtt{A}] \vdash f(f x) : \mathtt{A}}}{f : [\mathtt{B}, \mathtt{B}], x : [\mathtt{A}] \vdash f(f x) : [\mathtt{A}]}}{}}{f : [\mathtt{B}, \mathtt{B}, \mathtt{B}], x : [\mathtt{A}] \vdash f(f(f x)) : \mathtt{A}}}{f : [\mathtt{B}, \mathtt{B}, \mathtt{B}] \vdash \lambda x.f(f(f x)) : [\mathtt{A}] \to \mathtt{A}}}{\vdash \underline{\mathbf{3}} : [\mathtt{B}, \mathtt{B}, \mathtt{B}] \to [\mathtt{A}] \to \mathtt{A}}}{}
$$

Let $\underline{\mathbf{3}} := \lambda f.\lambda x.f(f(f x)))$ and let $\mathtt{B} := [\mathtt{A}] \to \mathtt{A}$

Non-Idempotent/Quantitative Typing with Multi-Sets

$$\vdash \underline{\mathbf{3}} : [\mathtt{B}, \mathtt{B}, \mathtt{B}] \to [\mathtt{A}] \to \mathtt{A}$$

Idempotent/Qualitative Typing with Sets

$$\vdash \underline{\mathbf{3}} : \{\mathtt{B}\} \to \{\mathtt{A}\} \to \mathtt{A}$$

Agenda

Duality between Typing and Inhabitation

Typing Problem

$\Gamma? \vdash \mathbf{t} : \mathtt{A}?$

Term Language | **Typing System**

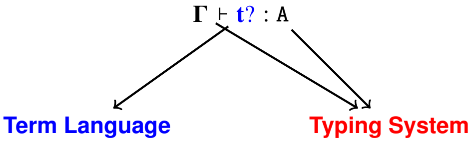
Inhabitation Problem

$\Gamma \vdash \mathbf{t}? : \mathtt{A}$

Term Language | **Typing System**

Equivalent Problems

Inhabitation
Proof Search
Program Synthesis



Typing and Inhabitation Problems for Lambda-Calculus

Call-by-Name Lambda-Calculus	**Typing** $? \vdash t : ?$	**Inhabitation** $\Gamma \vdash ? : A$
Simple Types	**Decidable**	**Decidable**
Unrestricted Idempotent Types	**Undecidable**	**Undecidable** **Urzyczyn** (Infinite Resources)
Restricted Idempotent Types	**Undecidable**	**Decidable** **Rehof, Dudenhefner, etc** (Finite Search on Infinite Resources)
Unrestricted Non-Idempotent Types	**Undecidable**	**Decidable** **Bucciarelli&K.&RonchiDellaRocca** (Finite Resources)

$\Longrightarrow$

on Infinite Resources $\Longrightarrow$ **on Finite Resources**

Non-deterministic algorithm

Theorem

The algorithm **terminates***:*
Every call on (Γ, σ) *generates a finite set of recursive calls.*

The algorithm is **sound***:*
If a call on (Γ, σ) *computes* $\mathbf{T}$*, then* $\Gamma \vdash \mathbf{T} : \sigma$*.*

The algorithm is **complete***:*
If $\Gamma \vdash \mathbf{T} : \sigma$*, then there exists an answer of the algorithm which generates* $\mathbf{T}$*.*

- First inhabitation results for **CBN**: **Bucciarelli&K.&Ronchi Della Rocca'**{**14,18,21**}
- Similar results for **CBV** and **dBang**: **Arrial&Guerrieri&K.'23**
- Implementation by **Arrial** in Ocaml.

Agenda

Intersection Types and Quantitative Analysis

- Intersection type systems provide a mathematical **meaning** of programs:

$$[\![\mathbf{t}]\!] := \{(\Gamma, \mathtt{A}) \mid \triangleright\Gamma \vdash \mathbf{t} : \mathtt{A}\}$$

- This gives **relational models** where **equivalent** programs have the **same** meaning :

If $\mathbf{t} \rightarrow_{\mathtt{operational}} \mathbf{u}$, then $[\![\mathbf{t}]\!] = [\![\mathbf{u}]\!]$

i.e. $\triangleright\Gamma \vdash^{(\mathbf{C})} \mathbf{t} : \mathtt{A} \Leftrightarrow \triangleright\Gamma \vdash^{(\mathbf{C}')} \mathbf{u} : \mathtt{A}$

called Subject Reduction and Expansion

Qualitative analysis (Idempotent types)

$\mathbf{C} \# \mathbf{C}'$

Quantitative analysis (Non-idempotent types)

$\mathbf{C} > \mathbf{C}'$

t is a **typable** term $\Longleftrightarrow$ t $\rightarrow \ldots \rightarrow$ **result**

More precisely:

$\triangleright_{\mathcal{S}} \Gamma \vdash t : A$ $\Longleftrightarrow$ t $\mathcal{N}$-normalizes to a **result**

Correspondence:

Type System $\mathcal{S}$ $\Longleftrightarrow$ $\mathcal{N}$-Normalization

(Idempotent)

✔ ✘

$$\triangleright_{\mathcal{S}}\ \Gamma \vdash^{(\mathbf{CL},\mathbf{S})} \mathbf{t} : \mathtt{A} \quad \Longleftrightarrow \quad \underbrace{\mathbf{t} \rightarrow \ldots \rightarrow}_{\text{length}} \underbrace{\textbf{result}}_{\text{size}}$$

$\mathcal{S}$ = **Non-Idempotent Types** with **UPPER BOUNDS** (e.g. Gardner's System $\mathcal{H}$)

> $\mathbf{t}$ $\mathcal{N}$-normalizes to a result and $\mathbf{length} + \mathbf{size} \leq \mathbf{C}$

$\mathcal{S}$ = **Non-Idempotent Types** with **EXACT MEASURES**

> $\mathbf{t}$ $\mathcal{N}$-normalizes to a result and $\mathbf{length} + \mathbf{size} = \mathbf{C}$

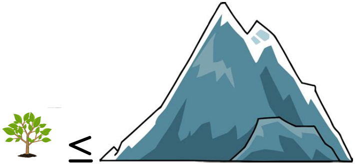
A possible **exponential** gap between **length** and **size**

$\mathcal{S}$ = **Non-Idempotent Types** with **SPLIT MEASURES**

> $\mathbf{t}$ $\mathcal{N}$-normalizes to a result and $\mathbf{length} = \mathbf{L}$ and $\mathbf{size} = \mathbf{S}$

Graphically

Idempotent Types



Non-Idempotent Types with Upper Bounds

Non-Idempotent Types with Exact Measures

Non-Idempotent Types with Split Measures

Typability Characterizes Quantitative Properties of Languages

This scheme applies to

- **Different**
  - ▷ Hea
  - ▷ Line
  - ▷ Left
  - ▷ Stro
- **Different**
  - ▷ Call
  - ▷ Unif
  - ▷ Res
  - ▷ Patt
  - ▷ Clas
  - ▷ Non
  - ▷ Prob

The Emblematic Example: Head Normalization

Head Normal Forms (HNF) : terms of the form $\lambda x_1 \ldots \lambda x_n . y t_1 \ldots t_m$ $(n, m \geq 0)$

Head Evaluation : No reduction inside arguments of applications.

$$\frac{}{(\lambda x.t)u \rightarrow_{hd} t\{x\backslash u\}} \qquad \frac{t \rightarrow_{hd} u}{\lambda x.t \rightarrow_{hd} \lambda x.u} \qquad \frac{t \rightarrow_{hd} u \quad t \neq \lambda}{tv \rightarrow_{hd} uv}$$

Head Normalization : if there exists a HNF u such that $t \rightarrow_{hd} \ldots \rightarrow_{hd} u$.

Example:
Let $\mathtt{I} := \lambda y.y$ and $\Omega := (\lambda y.yy)(\lambda y.yy)$. Then $\lambda x.\mathtt{I}x\Omega$ is head normalizing, while Ω is not.

Theorem (UPPER BOUNDS)

In Type System $\mathcal{H}$

$\triangleright_{\mathcal{H}} \Gamma \vdash^{(\mathbf{C})} \mathbf{t} : \mathtt{A}$

$\Longleftrightarrow$

t **head normalizes** in **length** steps

to a HNF of size **size** and

length + **size** $\leq$ **C**.

Characterizing Head Normalization with Split Measures

- **Grammar**:

(**Tight Constants**)	$\mathtt{tt}$	$::=$	$\mathtt{n} \mid \mathtt{a}$
(**Types**)	$\mathtt{A}$	$::=$	$\mathtt{tt} \mid \mathtt{M} \rightarrow \mathtt{A}$
(**Multi-types**)	$\mathtt{M}$	$::=$	$[\mathtt{A}_i]_{i \in I}$

- **Judgements with two counters**: $x_1 : \mathtt{M}_1, \ldots, x_n : \mathtt{M}_n \vdash^{(\mathbf{L},\mathbf{S})} t : \mathtt{A}$
  - **Multi-type** $\mathtt{M}_i$ for each variable x_i
  - **Type** $\mathtt{A}$ for the term t
  - $\mathbf{L}$ captures **length** of β-steps to head normal form
  - $\mathbf{S}$ captures **size** of the future head normal form
- **Typing Rules**:
  - They describe the increment and decrement of the two counters $(\mathbf{L}, \mathbf{S})$
  - They guess the different possible uses of the constructors along a sequence
- **Tight Derivations**: $\triangleright\Gamma \vdash^{(\mathbf{L},\mathbf{S})} t : \mathtt{A}$ is tight iff all the types in $\Gamma, \mathtt{A}$ are **tight constants**. Tight derivations (noted $\triangleright_{\text{tight}}$) represent **minimal** derivations.

Type System $\mathcal{SH}$ ($\mathcal{S}$ for $\mathcal{S}$plit)

Accattolli&Graham-Lengrand&K.

$$\frac{}{x : [\mathtt{A}] \vdash^{(0,0)} x : \mathtt{A}}\ (\mathtt{ax}) \qquad \frac{(\Gamma_i \vdash^{(L_i,S_i)} t : \mathtt{A}_i)_{i\in I}}{\sqcup_{i\in I} \Gamma_i \vdash^{(+_{i\in I} L_i, +_{i\in I} S_i)} t : [\mathtt{A}_i]}\ (\mathtt{many})$$

$$\frac{\Gamma; x : \mathtt{M} \vdash^{(L,S)} t : \mathtt{A}}{\Gamma \vdash^{(1+L,S)} \lambda x.t : \mathtt{M} \to \mathtt{A}}\ (\mathtt{fun_c}) \qquad \frac{\Gamma; x : \mathtt{M} \vdash^{(L,S)} t : \mathtt{tt} \quad \mathbf{IsItight}(\mathtt{M})}{\Gamma \vdash^{(L,S+1)} \lambda x.t : \mathtt{a}}\ (\mathtt{fun_p})$$

$$\frac{\Gamma \vdash^{(L,S)} t : \mathtt{M} \to \mathtt{A} \quad \Gamma' \vdash^{(L',S')} u : \mathtt{M}}{\Gamma \sqcup \Gamma' \vdash^{(L+L',S+S')} tu : \mathtt{A}}\ (\mathtt{app_c}) \qquad \frac{\Gamma \vdash^{(L,S)} t : \mathtt{n} \quad \Gamma' \vdash^{(L',S')} u : \mathtt{tt}}{\Gamma \sqcup \Gamma' \vdash^{(L+L',S+S'+1)} tu : \mathtt{n}}\ (\mathtt{app_p})$$

Rule	Role	Incrementation
$\mathtt{fun_c}$	consuming function	only first counter
$\mathtt{fun_p}$	persistent function	only second counter
$\mathtt{app_c}$	consuming application	no counter
$\mathtt{app_p}$	persistent application	only second counter

Example

A tight derivation for the term $(\lambda x.x\mathtt{I})\mathtt{I}$

$$
\dfrac{
\dfrac{
\dfrac{
x : [[\mathtt{a}] \to \mathtt{a}] \vdash^{(\mathbf{0},\mathbf{0})} x : [\mathtt{a}] \to \mathtt{a}
\qquad
\dfrac{\dfrac{z : \mathtt{n} \vdash^{(\mathbf{0},\mathbf{0})} z : \mathtt{n}}{\vdash^{(\mathbf{0},\mathbf{1})} \mathtt{I} : \mathtt{a}}}{\vdash^{(\mathbf{0},\mathbf{1})} \mathtt{I} : [\mathtt{a}]}
}{
x : [[\mathtt{a}] \to \mathtt{a}] \vdash^{(\mathbf{0},\mathbf{1})} x\mathtt{I} : \mathtt{a}
}
}{
\vdash^{(\mathbf{1},\mathbf{1})} \lambda x.x\mathtt{I} : [[\mathtt{a}] \to \mathtt{a}] \to \mathtt{a}
}
\qquad
\dfrac{
\dfrac{z : [\mathtt{a}] \vdash^{(\mathbf{0},\mathbf{0})} z : \mathtt{a}}{\vdash^{(\mathbf{1},\mathbf{0})} \mathtt{I} : [\mathtt{a}] \to \mathtt{a}}
}{
\vdash^{(\mathbf{1},\mathbf{0})} \mathtt{I} : [[\mathtt{a}] \to \mathtt{a}]
}
}{
\vdash^{(\mathbf{2},\mathbf{1})} (\lambda x.x\mathtt{I})\mathtt{I} : \mathtt{a}
}
$$

- The evaluation of $t = (\lambda x.x\mathtt{I})\mathtt{I}$ to head normal form has length **2**:

$$(\lambda x.x\mathtt{I})\mathtt{I} \to_\beta \mathtt{I}\mathtt{I} \to_\beta \mathtt{I}$$

- The head normal form $\mathtt{I}$ of t has size **1** (variables do not count).

Theorem (SPLIT MEASURES)

In Type System $\mathcal{SH}$ with two counters

$$\triangleright_{\mathcal{SH}} \Gamma \vdash^{(\mathbf{L},\mathbf{S})} \mathbf{t} : \mathtt{A}$$

$$\Longleftrightarrow$$

$\mathbf{t}$ **head normalizes** in $\mathbf{L}$ steps

to a HNF of size $\mathbf{S}$.

Qualitative Versus Quantitative Subject Reduction

Qualitative

If t is typable and $t \rightarrow_{hd} t'$, then t' is typable.

Quantitative (**Upper Bounds**)

If $\triangleright^{(\mathbf{C})}_{\mathcal{H}} t$ and $t \rightarrow_{hd} t'$, then $\exists \triangleright^{(\mathbf{C}')}_{\mathcal{H}} t'$ s.t. $\mathbf{C} > \mathbf{C}'$.

Quantitative (**Exact Measures**)

If $\triangleright^{(\mathbf{C})}_{\text{tight}} t$ and $t \rightarrow_{hd} t'$, then $\exists \triangleright^{(\mathbf{C}')}_{\text{tight}} t'$ s.t. $\mathbf{C} = \mathbf{C}' + 1$.

Quantitative (**Split Measures**)

If $\triangleright^{(\mathbf{L},\mathbf{S})}_{\text{tight}} t$ and $t \rightarrow_{hd} t'$, then $\exists \triangleright^{(\mathbf{L}-\mathbf{1},\mathbf{S})}_{\text{tight}} t'$.

Agenda

Power of quantitative types

- (Quantitative) **characterization** of different notions of termination (head, head-linear, head-needed, weak, strong, value, infinitary etc).
- Get around the size explosion problem (**upper bounds** versus **split/exact measures**).
- Quantitative **view** of traditional **properties** (solvability, genericity).
- **Relational** models.
- Turn Inhabitation problem **decidable**.
- Simple **observational equivalence** proofs by means of types.
- **Characterize** complexity classes.
- **Completeness** of reduction strategies.

Further Work

- Challenging cases:
  - ▷ **Effectful models of computation** (algebraic, continuations, . . .)
  - ▷ **Useful evaluation** (and other interesting time cost models)
  - ▷ **Strong evaluation** (for proof assistants)
  - ▷ **Deep Inference**
  - ▷ **General rewriting**
- (More) **quantitative view** of traditional properties.
- Compare **efficiency** of different implementations/strategies of programming languages by means of quantitative type theory.
- Identify **decidable** fragments applicable to **programming languages**.


THANK
YOU
FOR
YOUR
ATTENTION


Time for
Questions