AN INTRODUCTION TO QUANTITATIVE TYPES

Delia KESNER
kesner@irif.fr

Université Paris Cité

Differential λ-Calculus and Differential Linear Logic, 20 Years Later
Luminy, 13 May 2024

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

Outline

From Simple Types to Quantitative Types

Simple Types - Main Ideas

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

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

Simple Types - Main Ideas

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

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

- **Logical** (Curry-Howard) interpretation.

Simple Types - Main Ideas

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

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

- **Logical** (Curry-Howard) interpretation.
- **Monomorphic** information.

Simple Types - Main Ideas

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

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

- **Logical** (Curry-Howard) interpretation.
- **Monomorphic** information.
- Lack expressivity power but **typability is decidable**.

Simple Types - Main Ideas

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

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

- **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**:

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

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**:

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

- **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

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

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**:

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

- **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.

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**:

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

- **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**

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**:

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

- **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

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**:

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

- **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

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**:

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

- **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

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

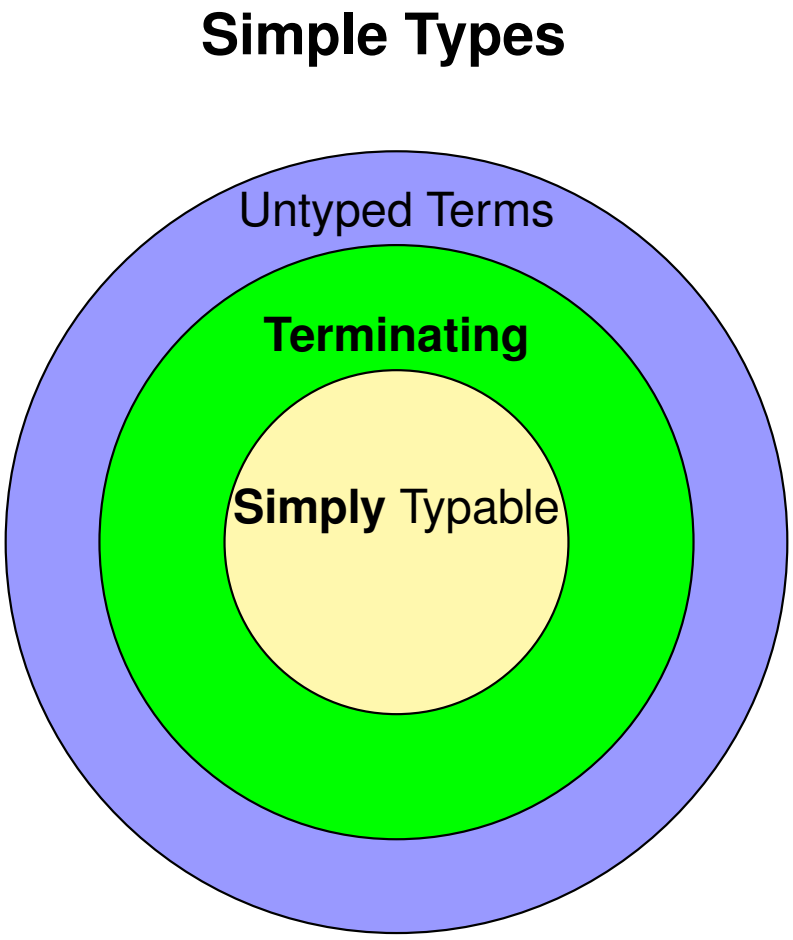
- But **more expressive**:

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

is perfectly admissible.

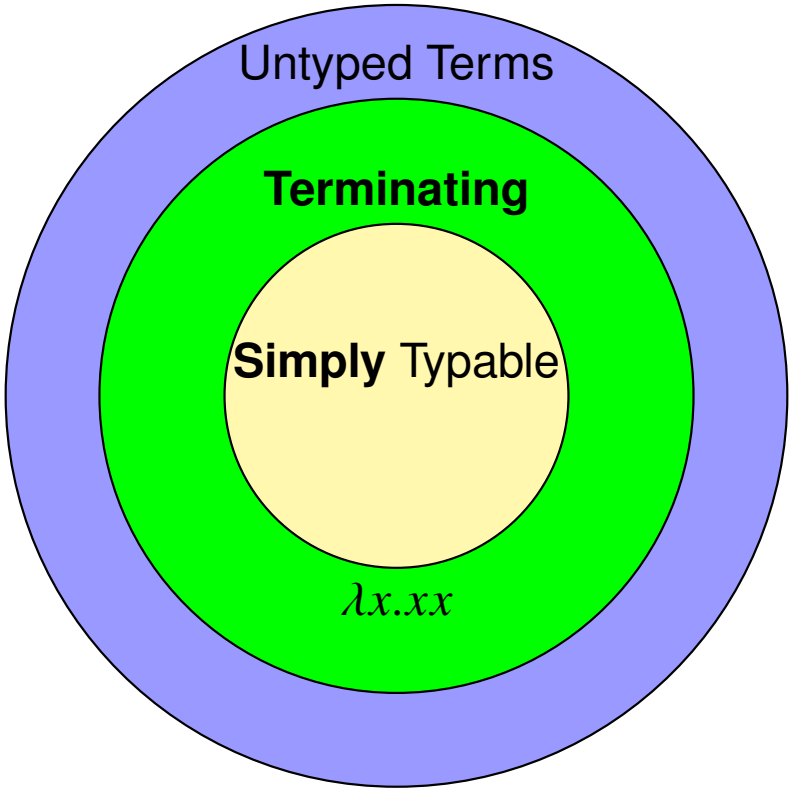
- Typability becomes **undecidable**
- Are **flexible** and can be adapted to different frameworks
- Provide **models** for different languages
- Very **powerful** tool to reason about properties of higher-order languages

Simple vs Intersection Types



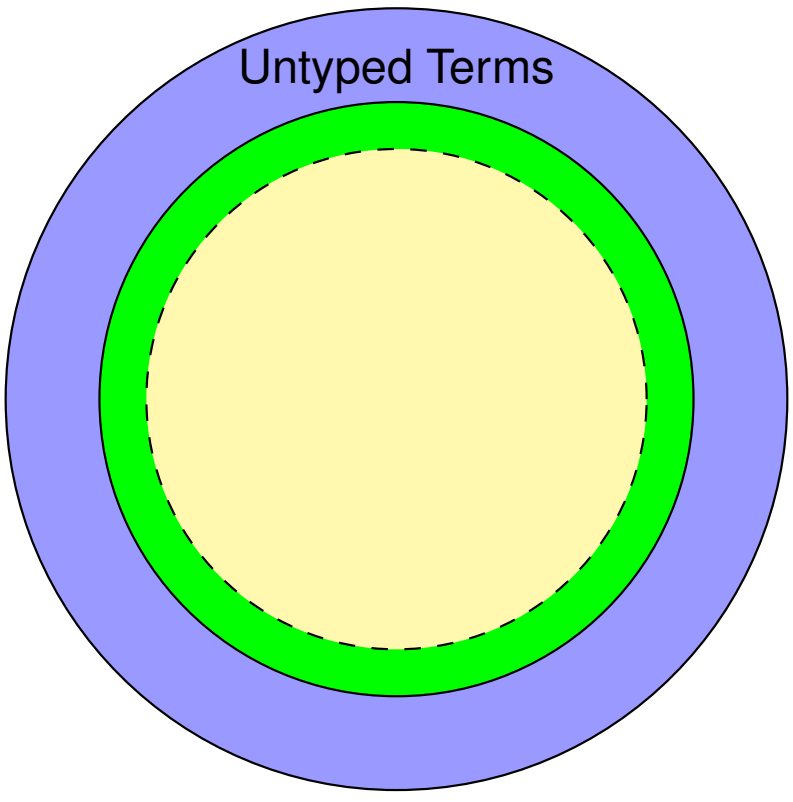
Simple vs Intersection Types

Simple Types

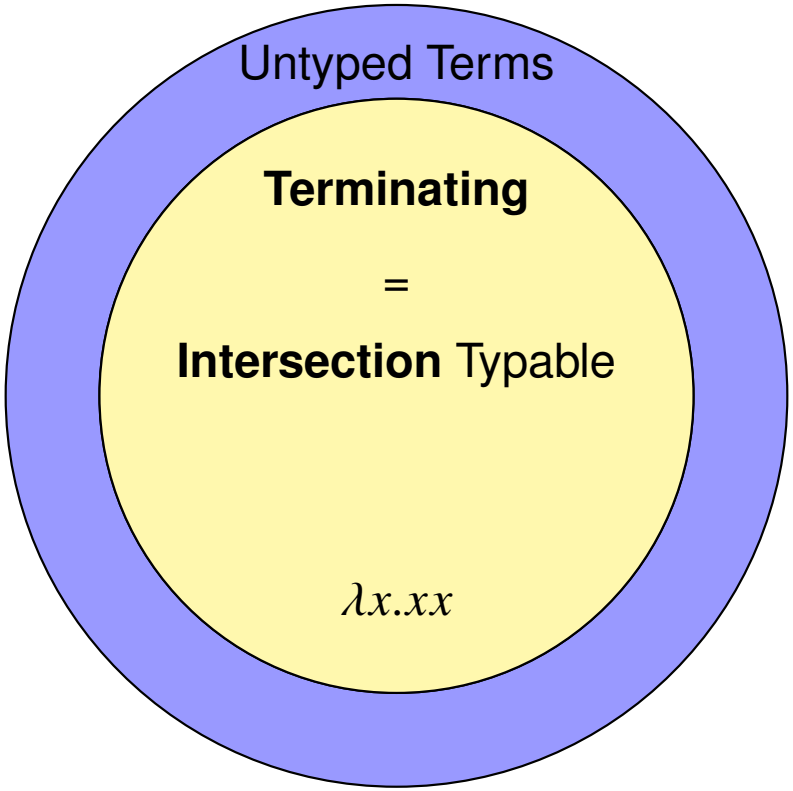


The term $\lambda x.xx$ is not simply typable

Simple vs Intersection Types



Intersection Types



The term $\lambda x.xx$ is intersection typable

Which kind of intersection constructor?

Which kind of intersection constructor?

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

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

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**

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

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**

$A \cap A \sim A$ $\qquad$ $A \cap A \nsim A$

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**

$A \cap A \sim A$ $\qquad$ $A \cap A \nsim A$

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

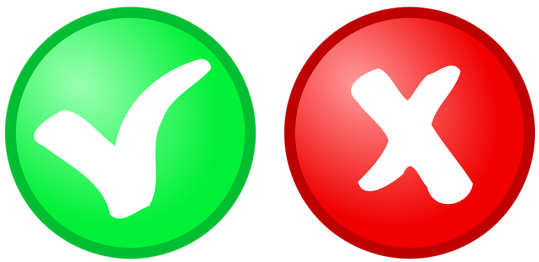
Idempotent vs Non-Idempotent Intersection Types

Idempotent	**Non-idempotent**
Coppo & Dezani in the eighties	**Gardner** and **Kfoury** in the nineties (**Girard**'s Linear Logic flavour)

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** $[A, A, C]$

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** 

Outline

An Emblematic Example: Gardner's system revisited

Grammar:

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

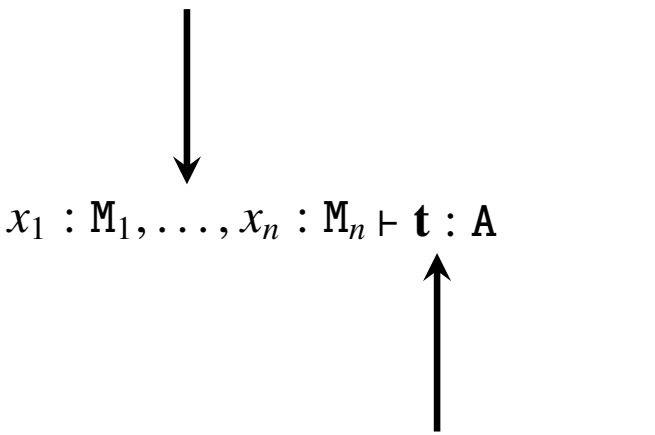
An Emblematic Example: Gardner's system revisited

Grammar:

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

Judgements:

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



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

An Emblematic Example: Gardner's system revisited

The Typing Rules:

An Emblematic Example: Gardner's system revisited

The Typing Rules:

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

Relevant axiom (no weakening).

An Emblematic Example: Gardner's system revisited

The Typing Rules:

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

- **Notation** $\Gamma(x)$ represents the type of x in the environment Γ.
- **Operation** $\Gamma \setminus x$ yields the environment Γ deprived from x.
- **Erasing** abstractions are typed with $[\,] \to \mathtt{A}$ (thus $(\lambda x.y)\Omega$ is typable).

An Emblematic Example: Gardner's system revisited

The Typing Rules:

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

- **Multiplicative** rule.
- **Notation** $[\mathtt{A}_i]_{i=1..n}$ represents a non-idempotent intersection $\mathtt{A}_1 \cap \ldots \cap \mathtt{A}_n$.
- **Operation** $x : \mathtt{A} \sqcup x : \mathtt{B}$ yields multiset **union** for multi-types $x : \mathtt{A} \sqcup \mathtt{B}$.
- **Special case**: any term t can be "artificially typed" with the **empty multi-type** $[\,]$: then t is said to be **untyped.**

An Emblematic Example: Gardner's system revisited

The Typing Rules:

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

- **Multiplicative** rule.
- **Typed** terms may contain **untyped** subterms (case $\mathtt{M} = \emptyset$).

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

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

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

- **Syntax Directed** system.

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

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

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

- The single **counter** computes the number of typing rules different from (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

$$\Pi \rhd_{\mathcal{S}} \Gamma \vdash^{(C_1,\ldots,C_n)} t : A$$

- Π is a **(tree) derivation**,
- $\mathcal{S}$ is a **type system**,
- Γ is a set of **type declarations**,
- $(C_1, \ldots, C_n)$ are **counters**.
- t is a **program/term**,
- A is a **type**.

(Standard) Notation for Type Derivability

$$\Pi \rhd_{\mathcal{S}} \Gamma \vdash^{(C_1,\ldots,C_n)} t : A$$

- Π is a **(tree) derivation**, (sometimes omitted)
- $\mathcal{S}$ is a **type system**, (sometimes omitted)
- Γ is a set of **type declarations**,
- $(C_1, \ldots, C_n)$ are **counters**. (sometimes omitted)
- t is a **program/term**,
- 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,

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

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

Another Example: Church Numerals

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

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

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

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

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

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

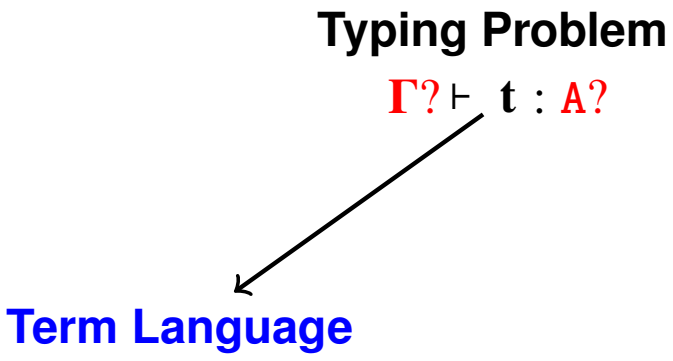
Outline

Duality between Typing and Inhabitation

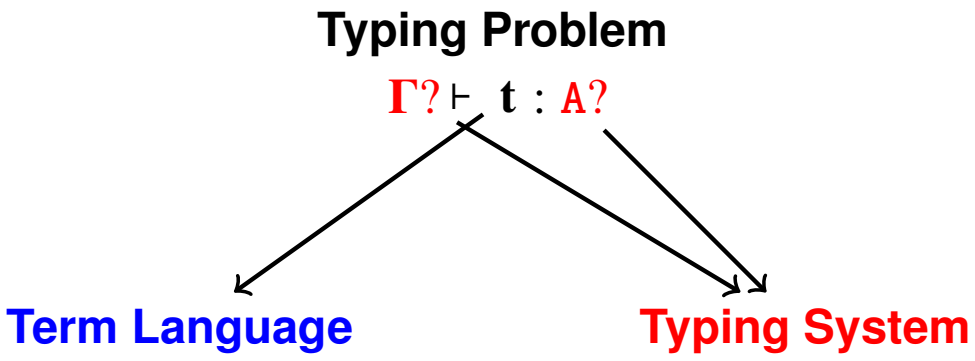
Typing Problem

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

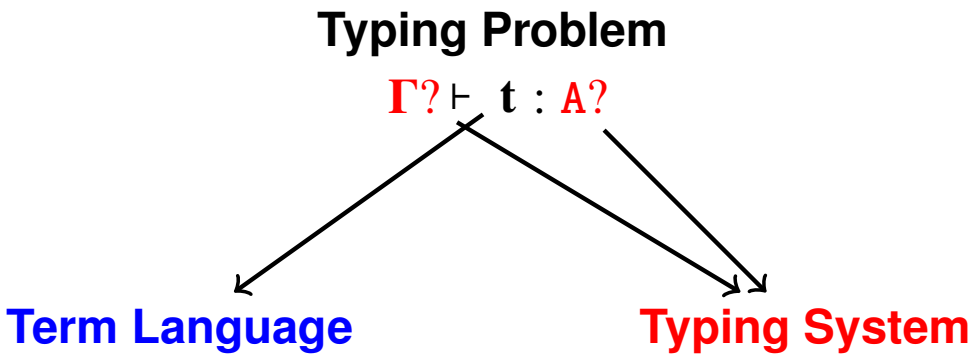
Duality between Typing and Inhabitation



Duality between Typing and Inhabitation



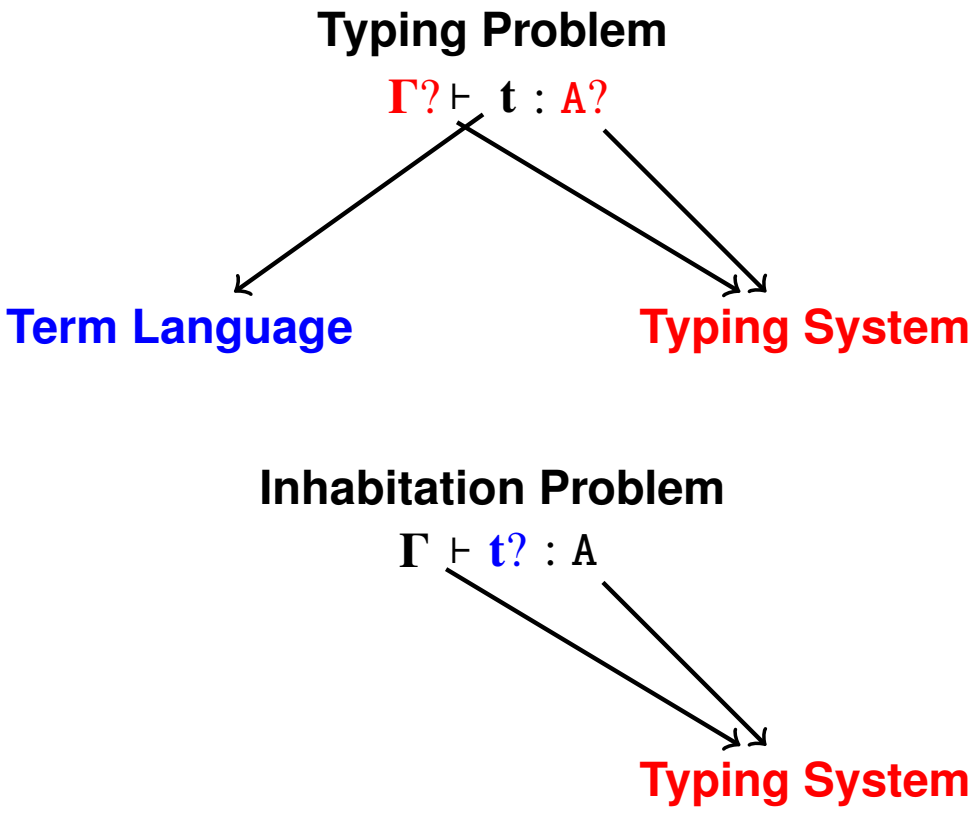
Duality between Typing and Inhabitation



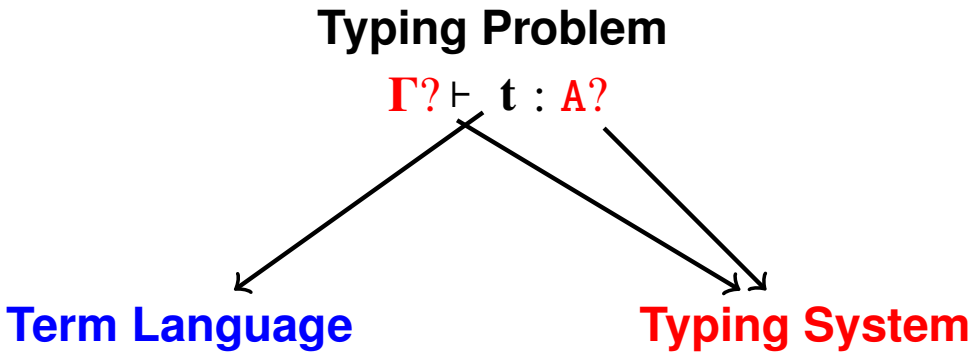
Inhabitation Problem

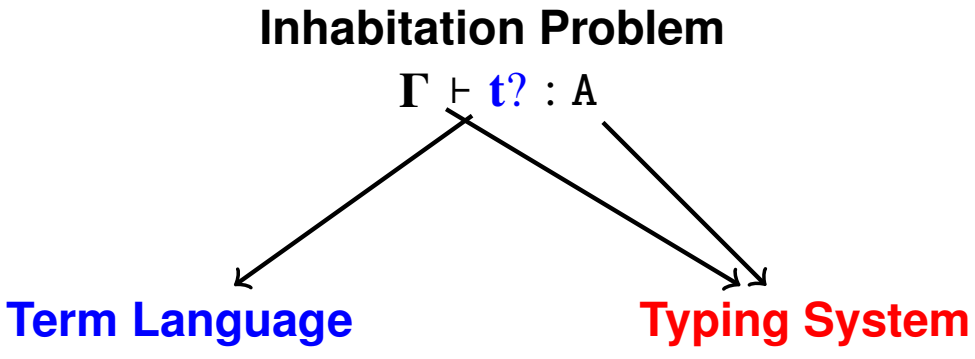
$\Gamma \vdash t? : A$

Duality between Typing and Inhabitation



Duality between Typing and Inhabitation



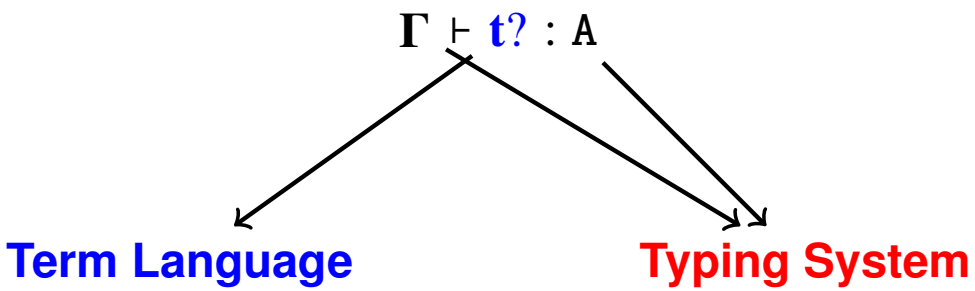


Equivalent Problems

Inhabitation

Proof Search

Program Synthesis



Typing and Inhabitation Problems for Lambda-Calculus

Call-by-Name Lambda-Calculus	**Typing** $? \vdash \mathbf{t} : \ ?$	**Inhabitation** $\Gamma \vdash ? : A$

Typing and Inhabitation Problems for Lambda-Calculus

Call-by-Name Lambda-Calculus	**Typing** $? \vdash \mathbf{t} : ?$	**Inhabitation** $\Gamma \vdash ? : A$
Simple Types	**Decidable**	**Decidable**

Typing and Inhabitation Problems for Lambda-Calculus

Call-by-Name Lambda-Calculus	**Typing** $? \vdash \mathbf{t} : ?$	**Inhabitation** $\Gamma \vdash ? : A$
Simple Types	**Decidable**	**Decidable**
Unrestricted Idempotent Types	**Undecidable**	**Undecidable** **Urzyczyn** (Infinite Resources)

Typing and Inhabitation Problems for Lambda-Calculus

Call-by-Name Lambda-Calculus	**Typing** $? \vdash \mathbf{t} : ?$	**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)

Typing and Inhabitation Problems for Lambda-Calculus

Call-by-Name Lambda-Calculus	**Typing** $? \vdash \mathbf{t} : ?$	**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)

Typing and Inhabitation Problems for Lambda-Calculus

Call-by-Name Lambda-Calculus	**Typing** $? \vdash \mathbf{t} : ?$	**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)

on Infinite Resources

$\Rightarrow$

on Finite Resources

Properties of the Inhabitation Algorithm

Non-deterministic algorithm

Theorem

Properties of the Inhabitation Algorithm

Non-deterministic algorithm

Theorem

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

Properties of the Inhabitation Algorithm

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* $\Gamma \vdash \mathbf{T} : \sigma$.

Properties of the Inhabitation Algorithm

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* $\mathbf{T}$.

Properties of the Inhabitation Algorithm

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.

Outline

Intersection Types and Quantitative Analysis

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

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

Intersection Types and Quantitative Analysis

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

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

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

If $\mathbf{t} \to_{\texttt{operational}} \mathbf{u}$, then $[\![\mathbf{t}]\!] = [\![\mathbf{u}]\!]$

Intersection Types and Quantitative Analysis

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

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

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

If $\mathbf{t} \to_{\texttt{operational}} \mathbf{u}$, then $[\![\mathbf{t}]\!] = [\![\mathbf{u}]\!]$

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

Intersection Types and Quantitative Analysis

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

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

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

If $\mathbf{t} \to_{\texttt{operational}} \mathbf{u}$, then $[\![\mathbf{t}]\!] = [\![\mathbf{u}]\!]$

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

called Subject Reduction and Expansion

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

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

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

If $\mathbf{t} \to_{\texttt{operational}} \mathbf{u}$, then $[\![\mathbf{t}]\!] = [\![\mathbf{u}]\!]$

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

called Subject Reduction and Expansion

Qualitative	**Quantitative**
analysis	**analysis**
(Idempotent types)	**(Non-idempotent types)**
$C \# C'$	$C > C'$

t is a **typable** term $\iff$ $\mathbf{t} \to \ldots \to$ **result**

Characterizing Evaluation by Means of **Qualitative**/Quantitative Types

$\mathbf{t}$ is a **typable** term $\iff$ $\mathbf{t} \to \ldots \to$ **result**

More precisely:

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

Characterizing Evaluation by Means of **Qualitative**/Quantitative Types

$\mathbf{t}$ is a **typable** term $\iff$ $\mathbf{t} \to \ldots \to$ **result**

More precisely:

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

Correspondence:

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

Characterizing Evaluation by Means of **Qualitative**/Quantitative Types

$\mathbf{t}$ is a **typable** term $\iff$ $\mathbf{t} \to \ldots \to$ **result**

More precisely:

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

Correspondence:

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

(Idempotent)

Characterizing Evaluation by Means of Qualitative/**Quantitative** Types

$$\triangleright_S \Gamma \vdash^{(C)} t : A \quad \Longleftrightarrow \quad t \underbrace{\rightarrow \ldots \rightarrow}_{\textbf{length}} \underbrace{\textbf{result}}_{\textbf{size}}$$

Characterizing Evaluation by Means of Qualitative/**Quantitative** Types

$$\triangleright_{\mathcal{S}} \Gamma \vdash^{(C)} t : A \iff t \underbrace{\rightarrow \ldots \rightarrow}_{\textbf{length}} \underbrace{\textbf{result}}_{\textbf{size}}$$

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

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

Characterizing Evaluation by Means of Qualitative/**Quantitative** Types

$$\triangleright_{\mathcal{S}} \Gamma \vdash^{(C)} t : A \iff t \underbrace{\to \ldots \to}_{\textbf{length}} \underbrace{\textbf{result}}_{\textbf{size}}$$

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

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

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

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

Characterizing Evaluation by Means of Qualitative/**Quantitative** Types

$$\triangleright_{\mathcal{S}} \Gamma \vdash^{(C)} t : A \iff t \underbrace{\rightarrow \ldots \rightarrow}_{\textbf{length}} \underbrace{\textbf{result}}_{\textbf{size}}$$

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

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

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

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

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

Characterizing Evaluation by Means of Qualitative/**Quantitative** Types

$$\triangleright_{\mathcal{S}} \Gamma \vdash^{(L,S)} t : A \iff t \underbrace{\rightarrow \ldots \rightarrow}_{\textbf{length}} \underbrace{\textbf{result}}_{\textbf{size}}$$

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

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

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

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

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

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

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

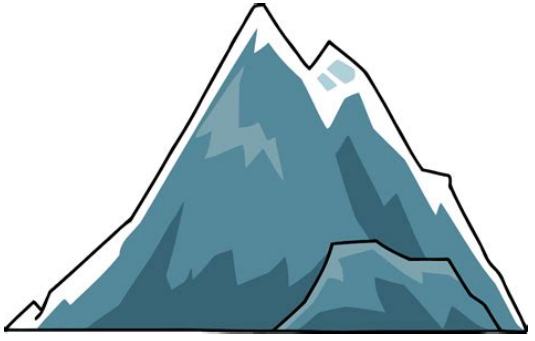
Graphically

Idempotent Types

Graphically

Idempotent Types

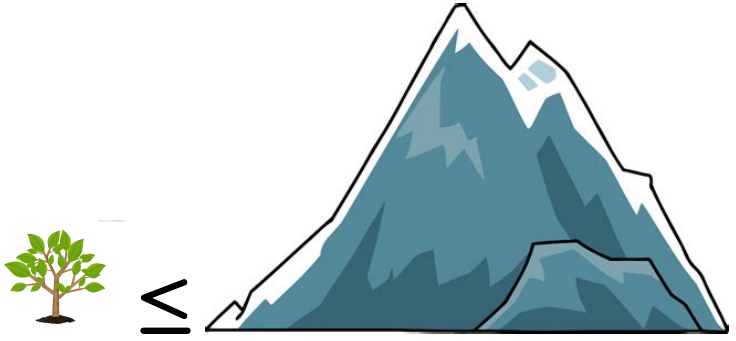
$\leq$



Non-Idempotent Types with Upper Bounds

Graphically

Idempotent Types

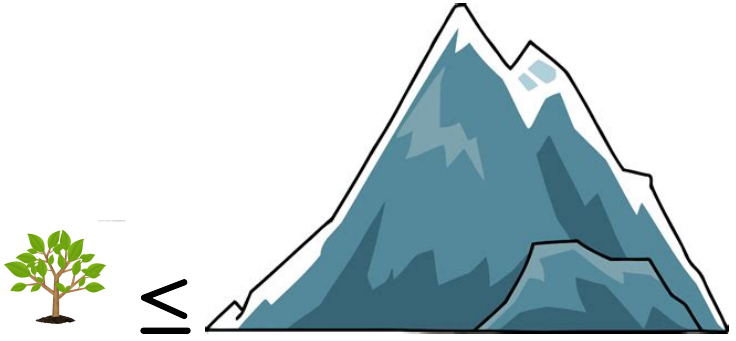


Non-Idempotent Types with Upper Bounds

Non-Idempotent Types with Exact Measures

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

Typability Characterizes Quantitative Properties of Languages

This scheme applies to

- **Different** normalization notions:
  - ▷ Head normalization
  - ▷ Linear head normalization
  - ▷ Leftmost normalization
  - ▷ Strong normalization

Typability Characterizes Quantitative Properties of Languages

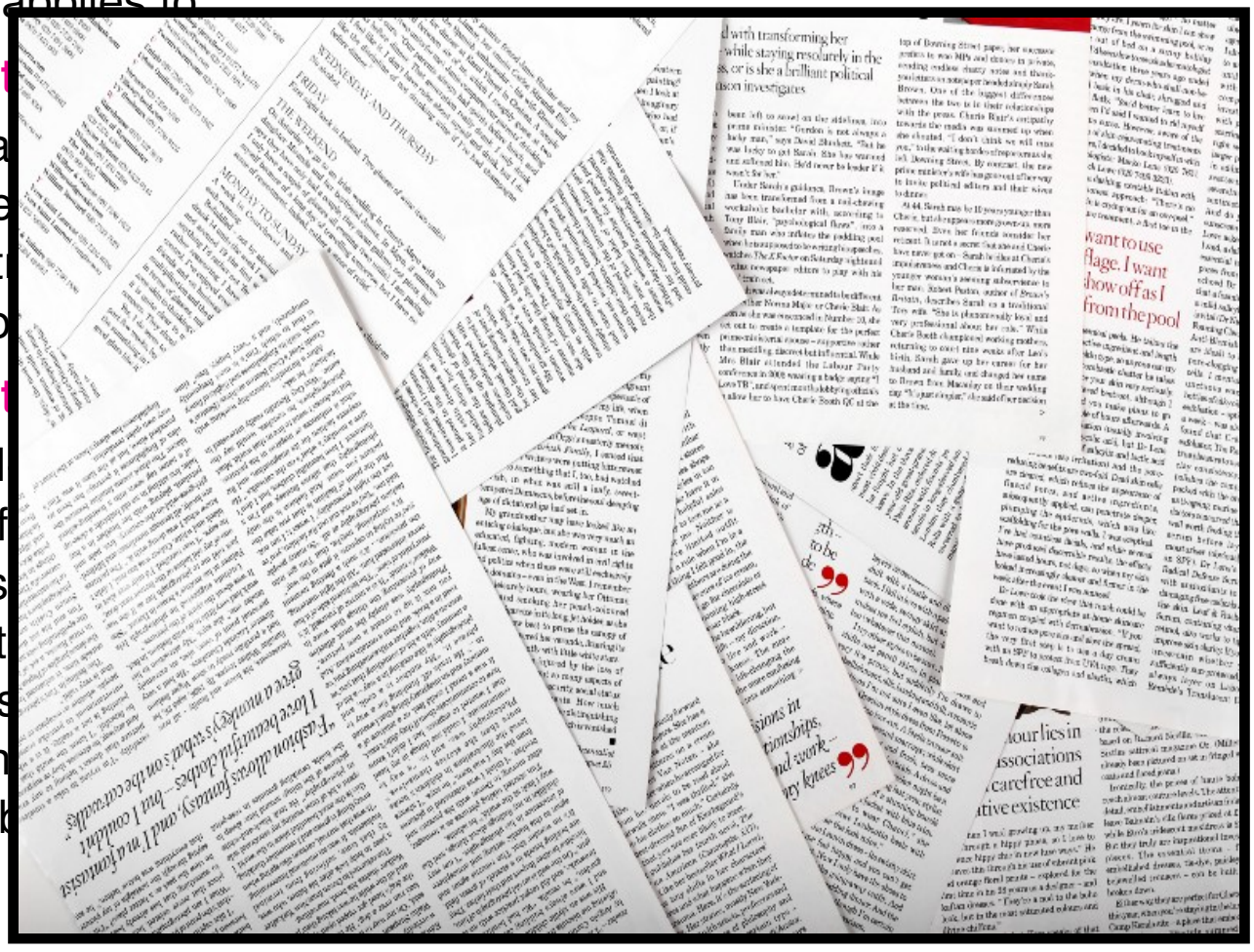
This scheme applies to

- **Different** normalization notions:
  - ▷ Head normalization
  - ▷ Linear head normalization
  - ▷ Leftmost normalization
  - ▷ Strong normalization
- **Different** models of computation:
  - ▷ Call-by-Name, Call-by-Value, Call-by-Need
  - ▷ Unifying models (*e.g.* Call-by-Push-Value, Bang Calculus)
  - ▷ Resource and explicit substitution calculi, proof-nets
  - ▷ Pattern matching features
  - ▷ Classical Calculi
  - ▷ Non-deterministic languages
  - ▷ Probabilistic languages, Bayesian inference

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 $\lambda x_1 \dots \lambda x_n . y t_1 \dots t_m$ $(n, m \geq 0)$

Head Evaluation : No reduction inside arguments of applications.

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

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

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.

The Emblematic Example: Head Normalization

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

Head Evaluation : No reduction inside arguments of applications.

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

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

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.

The Emblematic Example: Head Normalization

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

Head Evaluation : No reduction inside arguments of applications.

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

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

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.

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.

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

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

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**.

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**)	$\mathtt{A}$	$::=$	$\mathtt{tt} \mid \mathtt{M} \to \mathtt{A}$
(**Multi-types**)	$\mathtt{M}$	$::=$	$[\mathtt{A}_i]_{i \in I}$

Characterizing Head Normalization with Split Measures

- **Grammar**:

(**Tight Constants**)	$\mathtt{tt}$	$::=$	$\mathtt{n} \mid \mathtt{a}$
(**Types**)	$\mathtt{A}$	$::=$	$\mathtt{tt} \mid \mathtt{M} \to \mathtt{A}$
(**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

Characterizing Head Normalization with Split Measures

- **Grammar**:

(**Tight Constants**)	$\mathtt{tt}$	$::=$	$\mathtt{n} \mid \mathtt{a}$
(**Types**)	$\mathtt{A}$	$::=$	$\mathtt{tt} \mid \mathtt{M} \to \mathtt{A}$
(**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

Characterizing Head Normalization with Split Measures

- **Grammar**:

(**Tight Constants**)	$\mathtt{tt}$	$::=$	$\mathtt{n} \mid \mathtt{a}$
(**Types**)	$\mathtt{A}$	$::=$	$\mathtt{tt} \mid \mathtt{M} \to \mathtt{A}$
(**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.

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

$$\frac{\Gamma; x : \mathtt{M} \vdash t : \mathtt{A}}{\Gamma \vdash \lambda x.t : \mathtt{M} \to \mathtt{A}}\ (\mathtt{fun_c})$$

$$\frac{\Gamma \vdash t : \mathtt{M} \to \mathtt{A} \qquad \Gamma' \vdash u : \mathtt{M}}{\Gamma \sqcup \Gamma' \vdash tu : \mathtt{A}}\ (\mathtt{app_c})$$

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

Accattolli&Graham-Lengrand&K.

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

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

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

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

Accattolli&Graham-Lengrand&K.

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

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

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

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

Accattolli&Graham-Lengrand&K.

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

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

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

Rule	Role	Incrementation
$\mathtt{fun_c}$	**c**onsuming function	only first counter
$\mathtt{fun_p}$	**p**ersistent function	only second counter
$\mathtt{app_c}$	**c**onsuming application	no counter
$\mathtt{app_p}$	**p**ersistent application	only second counter

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

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

$$\vdash^{(2,1)} (\lambda x.x\mathtt{I})\mathtt{I} : \mathsf{a}$$

- 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

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

$$\Longleftrightarrow$$

t head normalizes in $\mathbf{L}$ steps
to a HNF of size $\mathbf{S}$.

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 $t \to_{hd} t'$, then t' is typable.

Qualitative Versus Quantitative Subject Reduction

Qualitative

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

Quantitative (**Upper Bounds**)

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

Qualitative Versus Quantitative Subject Reduction

Qualitative

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

Quantitative (**Upper Bounds**)

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

Qualitative Versus Quantitative Subject Reduction

Qualitative

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

Quantitative (**Upper Bounds**)

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

Quantitative (**Exact Measures**)

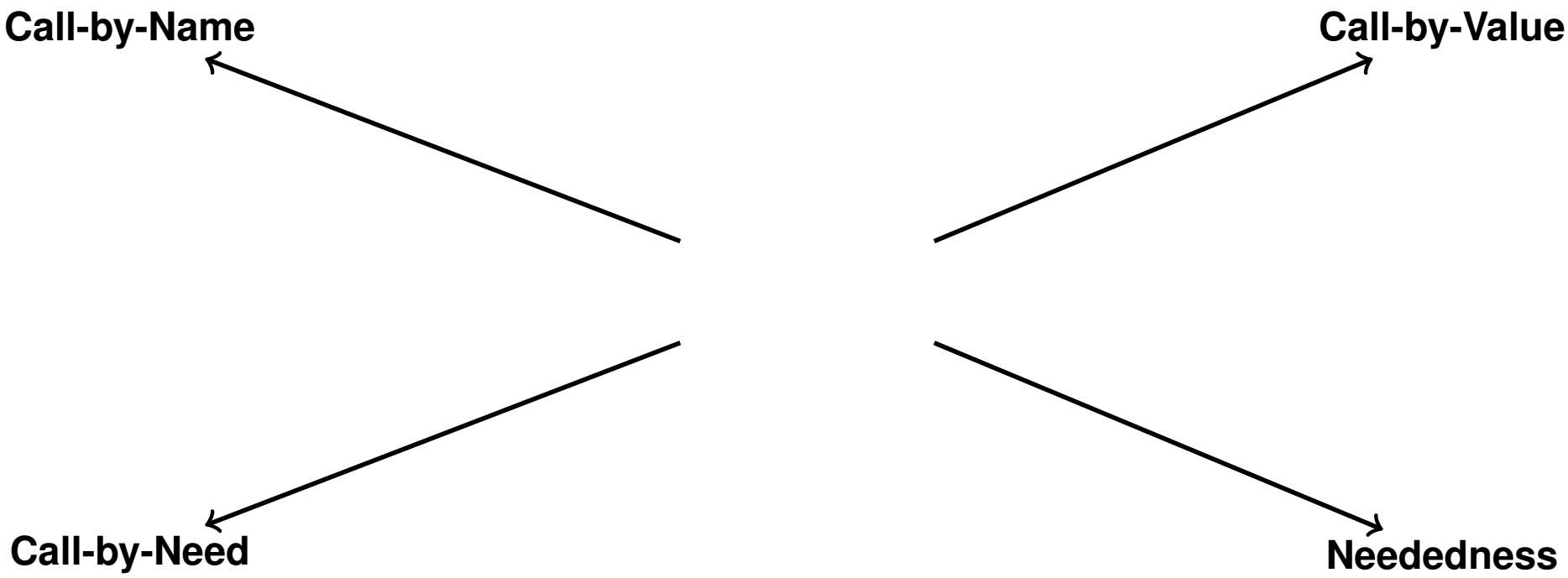
If $\triangleright^{(\mathbf{C})}_{\text{tight}} t$ and $t \to_{hd} t'$, then $\exists\ \triangleright^{(\mathbf{C}')}_{\text{tight}} t'$ s.t. $\mathbf{C} = \mathbf{C}' + 1$.

Quantitative (**Split Measures**)

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

Outline

Observational Equivalence



Observational Equivalence

Call-by-Name **Call-by-Value**

$\mathbf{t} \cong_{\mathcal{R}_1} \mathbf{u}$ iff $\mathbf{t} \cong_{\mathcal{R}_2} \mathbf{u}$?

Call-by-Need **Neededness**

Call-by-Need Different from Call-by-Name

Call-by-Need Different from Call-by-Name

Call-by-need is different from **call-by-name**:

$$\texttt{Twice}\ (4+3) \to_{\texttt{cbname}} (4+3)+(4+3) \to_{\texttt{cbname}} 7+(4+3) \to_{\texttt{cbname}} 7+7 \to_{\texttt{cbname}} 14$$
$$\texttt{Twice}\ (4+3) \to_{\texttt{cbneed}} \texttt{Twice}\ 7 \to_{\texttt{cbneed}} 7+7 \to_{\texttt{cbneed}} 14$$

where $\texttt{Twice} = \lambda x.x + x$.

Call-by-Need Different from Call-by-Value

Call-by-Need Different from Call-by-Value

Call-by-need is different from **call-by-value**:

$$(\lambda x.8)(4+3) \rightarrow_{\texttt{cbvalue}} (\lambda x.8)7 \rightarrow_{\texttt{cbvalue}} 8$$
$$(\lambda x.8)(4+3) \rightarrow_{\texttt{cbneed}} 8$$

In particular

$$(\lambda x.8)\Omega \not\rightarrow_{\texttt{cbvalue}}$$
$$(\lambda x.8)\Omega \rightarrow_{\texttt{cbneed}} 8$$

Call-by-Need Different from Neededness

Call-by-Need Different from Neededness

(Syntactical) call-by-need is different from **(semantical) neededness**

$$(\lambda x.x)(4+3) \rightarrow_{\text{cbneed}} (\lambda x.x)7 \rightarrow_{\text{cbneed}} 7$$
$$(\lambda x.x)(4+3) \rightarrow_{\text{neededness}} 4+3 \rightarrow_{\text{neededness}} 7$$

Observational Equivalence by Means of Type Theory

Same typing system to capture **different** models of computation

Observational Equivalence by Means of Type Theory

Same typing system to capture **different** models of computation

t is typable in type system $\mathcal{A}$ **if and only if** **t** terminates in call-by-need.

t is typable in type system $\mathcal{A}$ **if and only if** **t** terminates in call-by-name.

t is typable in type system $\mathcal{A}$ **if and only if** **t** terminates w.r.t. neededness.

Observational Equivalence by Means of Type Theory

Same typing system to capture **different** models of computation

t is typable in type system $\mathcal{A}$ **if and only if** **t** terminates in call-by-need.

t is typable in type system $\mathcal{A}$ **if and only if** **t** terminates in call-by-name.

t is typable in type system $\mathcal{A}$ **if and only if** **t** terminates w.r.t. neededness.

Theorem (**K.'16, K.&Viso&Ríos'18**)

$\mathbf{t} \cong_{\textit{call-by-name}} \mathbf{u}$ **if and only if** $\mathbf{t} \cong_{\textit{call-by-need}} \mathbf{u}$ **if and only if** $\mathbf{t} \cong_{\textit{neededness}} \mathbf{u}$.

Some references

- **First** quantitative systems for lambda-calculus: Gardner'94, Kfoury'00, MøllerNeergaard-Mairson'04.
- **Pioneer** for quantitative properties: Carvalho 07, Carvalho'18.
- **Survey** on quantitative types and lambda-calculus: Bucciarelli-K.-Ventura'17.
- Quantitative types and **exact measures**: Bernadet-GrahamLengrand'11 and 13, Accattoli-GrahamLengrand-K.'18.
- Quantitative types for **call-by-value**: Ehrhard'12, Carraro-Guerrieri'14, Accattoli-Guerrieri'18, Guerrieri'19, Manzonetto-Pagani-RonchiDellaRocca'19, Kerinec-Manzonetto-RonchiDellaRocca'21, K.-Viso'2022, Accattoli-Guerrieri'22.
- Quantitative types for **call-by-need**: K.'16, K.-Rios-Viso'18, Balabonski-Bonelli-Barenbaum-K.'17, Accattoli-Guerrieri-Leberle'19, Accattoli-Leberle'22.
- Quantitative types for **call-by-push-value**: Guerrieri-Manzonetto'19, Bucciarelli-K.-Ríos-Viso'20.
- Quantitative types for **global memory**: Alves-K.-Ramos'23.

Some references

- Quantitative types for **proof-theory**: deCarvalho-Pagani-TortoradeFalco'11, Pimentel-RonchiDellaRocca-Roversi'12, TortoradeFalco-DeCarvalho'13, K.-Ventura'15 and 17, deCarvalho-TortoradeFalco'16, Ehrhard'20, Guerrieri-Heijltjes-Paulus'21, K.-Peyrot-Ventura'21, EspíritoSanto-K.-Peyrot'22.
- Quantitative types and **relational models**: Bucciarelli-Ehrhard'01, Bucciarelli-Ehrhard-Manzonetto'07, Paolini-Piccolo-RonchiDellaRocca'17.
- Quantitative types and **complexity classes**: DeBenedetti-RonchiDellaRocca'16.
- Quantitative types and **category-theory**: Ehrhard'12, Mazza-Pellissier-Vial'18, Guerrieri-Olimpieri'21. Kerinec-Manzonetto-Olimpieri'23.
- Quantitative types and **concurrency**: DalLago-de Visme-Mazza-Yoshimizu'19.
- Quantitative types for **classical term calculi**: K.-Vial'17 and 20.
- Quantitative types for **infinite calculi**: Vial'17.
- Quantitative types for **pattern-matching calculi**: Bucciarelli-K.-RonchiDellaRocca'15 and 21, Alves-K.-Ventura'19, Alves-K.-Ramos'22.
- Quantitative types for **probabilistic programming**: DalLago-Faggian-RonchiDellaRocca.
- Quantitative types for **space bounds**: Accattoli-DalLago-Vanoni'21.
- **Inhabitation** for quantitative types: Bucciarelli-RonchiDellaRocca-K.'14, Bucciarelli-RonchiDellaRocca-K.'21, Arrial-Guerrieri-K.'23.

Some Related Topics

- Taylor Expansion
- Böhm Trees
- Proof-Nets
- Resource Calculi and Explicit Substitutions
- Observational Equivalence
- Game Semantics
- Relational Models
- Higher-Order Model Checking

Outline

Power of quantitative types

- (Quantitative) **characterization** of different notions of termination (head, head-linear, head-needed, weak, strong, value, infinitary etc).

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**).

Concluding Remarks

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).

Concluding Remarks

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.

Concluding Remarks

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**.

Concluding Remarks

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.

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.

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**

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.

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.

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**.