QUANTITATIVE TYPES FOR HIGHER-ORDER LANGUAGES

Delia KESNER
IRIF, Université Paris Cité, CNRS

HCERES, 28 November 2023

Motivations

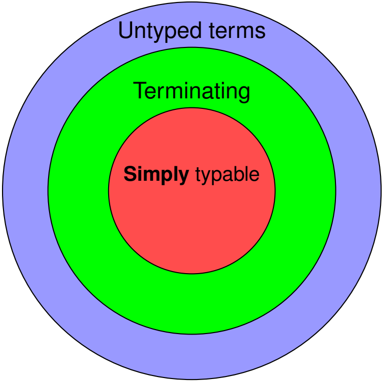
- **Quantitative** techniques emerging in different areas of computer science.
  - Time, space, probability, cost.
  - Verification, model-checking, theorem proving.
  - Automata, logics, algorithm analysis.
  - Performance measurement, network analysis, data mining.
- **Types** are a key tool in programming languages.
- What is a quantitative type **system**?
- **Principles**, **Properties**, and **Applications**.

Outline

1. Some Principles of Quantitative Type Systems
2. Quantitative Types and Inhabitation
3. Quantitative Types for Measuring
4. Quantitative Types and Observational Equivalence
5. Conclusion

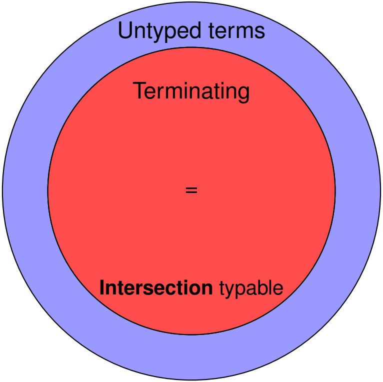
Simple Types

$4 : \texttt{int}, f : \texttt{int} \Rightarrow \texttt{int}$



Intersections Types

$4 : \texttt{int} \cap \texttt{float}, g : \texttt{int} \cap \texttt{bool} \Rightarrow \texttt{int}$



Which Kind of Intersection Constructor?

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

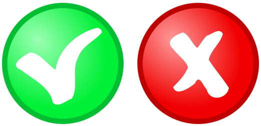
Commutativity $A \cap B \quad \sim \quad B \cap A$

Idempotent versus **Non-idempotent**

$A \cap A = A$ $\qquad$ $A \cap A \neq A$

Infinite 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: $\mathtt{A} \cap \mathtt{A} \cap \mathtt{C}$ is $\{\mathtt{A}, \mathtt{C}\}$	**Multi-Sets** : $\mathtt{A} \cap \mathtt{A} \cap \mathtt{C}$ is $[\mathtt{A}, \mathtt{A}, \mathtt{C}]$
Qualitative properties: **Yes** or **No** 	**Quantitative** properties: **Bounds** and **Exact Measures** **De Carvalho** 

Let $t := \lambda x.x\,(x\;x)$

Idempotent/Qualitative Typing with Sets

$$\boxed{\vdash t : \{\{\mathtt{A}\} \longrightarrow \mathtt{A}, \mathtt{A}\} \longrightarrow \mathtt{A}}$$

Non-Idempotent/Quantitative Typing with Multi-Sets

$$\boxed{\vdash t : [[\mathtt{A}] \longrightarrow \mathtt{A}, [\mathtt{A}] \longrightarrow \mathtt{A}, \mathtt{A}] \longrightarrow \mathtt{A}}$$

(Standard) Notation for Typing

$$\boxed{\Pi \triangleright_{\mathcal{X}} \Gamma \vdash t : \mathtt{A}}$$

- $\mathcal{X}$ is a **type system**,
- Π is a **(tree) derivation**,
- t is a **program/term**,
- $\mathtt{A}$ is a **type**,
- Γ is a set of **type declarations**.
- $\mathcal{X}$ and Π may be omitted to simplify the notation.

Outline

1 Some Principles of Quantitative Type Systems

2 Quantitative Types and Inhabitation

3 Quantitative Types for Measuring

4 Quantitative Types and Observational Equivalence

5 Conclusion

Duality between Typing and Inhabitation

Typing Problem

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

Term Language **Typing System**

Inhabitation Problem

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

Term Language **Typing System**

Equivalent Problems

Inhabitation
Proof Search
Program Synthesis

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

Term Language **Typing System**

Typing and Inhabitation Problems for Lambda-Calculus

Call-by-Name Lambda-Calculus	**Typing** $? \vdash t : ?$	**Inhabitation** $\Gamma \vdash ? : A$
Simple Types	**Decidable**	**Decidable**
Idempotent Types	**Undecidable**	**Undecidable** (Infinite Resources)
Restricted Idempotent Types	**Undecidable**	**Decidable** (Finite Search on Infinite Resources)
Unrestricted Non-Idempotent Types	**Undecidable**	**Decidable** (Finite Resources)

on Infinite Resources

$\Rightarrow$

on Finite Resources

Bucciarelli&K.&Ronchi Della Rocca'{14,18,21}, Arrial&Guerrieri&K.'21

Outline

1. Some Principles of Quantitative Type Systems
2. Quantitative Types and Inhabitation
3. Quantitative Types for Measuring
4. Quantitative Types and Observational Equivalence
5. Conclusion

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

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

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

$$\text{If } t =_{\mathtt{operational}} u, \quad \text{then} \quad [\![t]\!] = [\![u]\!]$$
$$\textit{i.e.} \quad \Pi \triangleright \Gamma \vdash t : \mathtt{A} \Leftrightarrow \Pi' \triangleright \Gamma \vdash u : \mathtt{A}$$

Qualitative	**Quantitative**
$\mathtt{size}(\Pi) \# \mathtt{size}(\Pi')$	$\mathtt{size}(\Pi) > \mathtt{size}(\Pi')$
Idempotent types	**Non-idempotent types**

Let $\mathbf{t}$ be a **typable** term, then $\mathbf{t} \underbrace{\rightarrow \ \ldots \rightarrow}_{\textbf{length ?}} \underbrace{\textbf{result?}}_{\textbf{size ?}}$

(Standard) Notation for Typing

$$\boxed{\Pi \triangleright \Gamma \vdash \mathbf{t} : \mathtt{A}}$$

Quantitative Notation for Typing

$$\boxed{\Pi \triangleright \Gamma \vdash^{(\mathsf{c}_1,\ldots,\mathsf{c}_n)} \mathbf{t} : \mathtt{A}}$$

The **counters** $(\mathsf{c}_1, \ldots, \mathsf{c}_n)$ measure different behaviors of the program $\mathbf{t}$

Let $\Pi \triangleright \Gamma \vdash^{(L,S)} \mathbf{t} : \mathtt{A}$, then $\mathbf{t} \underbrace{\rightarrow \ \dots \ \rightarrow}_{\textbf{length ?}} \ \underbrace{\textbf{result?}}_{\textbf{size ?}}$

IDEMPOTENT TYPES

(RES) $\mathbf{t}$ evaluates to some **result**

NON-IDEMPOTENT TYPES with UPPER BOUNDS

(RES) and $\mathbf{length} + \mathbf{size} \leq \mathtt{size}(\Pi)$

NON-IDEMPOTENT TYPES with SPLIT/EXACT MEASURES

(RES) and $\mathbf{length} = \mathbf{L}$ and $\mathbf{size} = \mathbf{S}$

This scheme a

- **Different**
  - Head
  - Line
  - Leftm
  - Stron
- **Different**
  - Call-
  - Unify
  - Reso
  - Patte
  - Cont
  - Glob
  - Proo
  - Non-
  - Prob

Outline

1. Some Principles of Quantitative Type Systems
2. Quantitative Types and Inhabitation
3. Quantitative Types for Measuring
4. Quantitative Types and Observational Equivalence
5. Conclusion

Observational Equivalence

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

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

Call-by-Need | **Neededness**

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

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

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

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

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

$$(\lambda x.8)\Omega \cancel{\rightarrow_{\texttt{cbvalue}}}$$
$$(\lambda x.8)\Omega \rightarrow_{\texttt{cbneed}} 8$$

Call-by-Need Different from Neededness

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

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

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

- t is typable in type system $\mathscr{A}$ **if and only if** t normalizes in call-by-need.
- t is typable in type system $\mathscr{A}$ **if and only if** t normalizes in call-by-name.
- t is typable in type system $\mathscr{A}$ **if and only if** t normalizes w.r.t. neededness.

Theorem (**K.'16, K.&Viso&Ríos'18**)

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

Outline

1. Some Principles of Quantitative Type Systems
2. Quantitative Types and Inhabitation
3. Quantitative Types for Measuring
4. Quantitative Types and Observational Equivalence
5. Conclusion

Power of Quantitative Types

- Provide **quantitative** information (**upper bounds** and **split/exact measures**).
- **Relational** models.
- **Characterization** of different notions of normalization (head, head-linear, head-needed, weak, strong, value, infinitary etc).
- Inhabitation **decidable**.
- Simple **observational equivalence** proofs by means of types.
- **Characterize** complexity classes.
- **Completeness** of reduction strategies.

Ongoing Work

- New (time) cost models for functional programming (usefulness, pattern matching).
- Effectful computations (global memory, exceptions).
- Unifying frameworks (call-by-push-value, bang calculus).
- Quantitative view of traditional properties (solvability, genericity).

Thanks