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 versus Intersection Type Systems

**Simple Types**

4: \( \text{int}, f: \text{int} \Rightarrow \text{int} \)

**Intersections Types**

4: \( \text{int} \cap \text{float}, g: \text{int} \cap \text{bool} \Rightarrow \text{int} \)

Untyped terms

Terminating

Simply typable

Untyped terms

Terminating

Intersection typable
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**

\[A \cap A = A\]

**Non-idempotent**

\[A \cap A \neq A\]
## Idempotent vs Non-Idempotent Intersection Types

<table>
<thead>
<tr>
<th>Idempotent</th>
<th>Non-idempotent</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Coppo &amp; Dezani</strong> in the eighties</td>
<td><strong>Gardner and Kfoury</strong> in the nineties (Girard’s Linear Logic flavour)</td>
</tr>
<tr>
<td><strong>Sets</strong>: $A \cap A \cap C$ is ${A, C}$</td>
<td><strong>Multi-Sets</strong>: $A \cap A \cap C$ is $[A, A, C]$</td>
</tr>
<tr>
<td><strong>Qualitative properties</strong>: Yes or No</td>
<td><strong>Quantitative properties</strong>: Bounds and Exact Measures De Carvalho</td>
</tr>
</tbody>
</table>
An Example

Let \( t := \lambda x. x \ (x \ x) \)

**Idempotent/Qualitative** Typing with Sets

\[ \vdash t : \{\{A\} \to A, A\} \to A \]

**Non-Idempotent/Quantitative** Typing with Multi-Sets

\[ \vdash t : [[A] \to A, [A] \to A, A] \to A \]
(Standard) Notation for Typing

\[ \Pi \triangleright \chi \quad \Gamma \vdash t : A \]

- \( \chi \) is a **type system**,
- \( \Pi \) is a **(tree) derivation**,
- \( t \) is a **program/term**,
- \( A \) is a **type**,
- \( \Gamma \) is a set of **type declarations**.
- \( \chi \) and \( \Pi \) may be omitted to simplify the notation.
Some Principles of Quantitative Type Systems

Duality between Typing and Inhabitation

Typing Problem
\[ \Gamma \vdash t : A \]

Term Language
Typing System

Inhabitation Problem
\[ \Gamma \vdash t : A \]

Term Language
Typing System
Some Principles of Quantitative Type Systems

- Quantitative Types and Inhabitation
- Quantitative Types for Measuring
- Quantitative Types and Observational Equivalence

Conclusion

Equivalent Problems

Inhabitation

Proof Search

Program Synthesis

\[ \Gamma \vdash t? : A \]

Term Language

Typing System
### Typing and Inhabitation Problems for Lambda-Calculus

<table>
<thead>
<tr>
<th>Call-by-Name Lambda-Calculus</th>
<th>Typing</th>
<th>Inhabitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Types</td>
<td>Decidable</td>
<td>Decidable</td>
</tr>
<tr>
<td>Idempotent Types</td>
<td>Undecidable</td>
<td>Undecidable (Infinite Resources)</td>
</tr>
<tr>
<td>Restricted Idempotent Types</td>
<td>Undecidable</td>
<td>Decidable (Finite Search on Infinite Resources)</td>
</tr>
<tr>
<td>Unrestricted Non-Idempotent Types</td>
<td>Undecidable</td>
<td>Decidable (Finite Resources)</td>
</tr>
</tbody>
</table>

Search on Infinite Resources $\Rightarrow$ Search on Finite Resources

Bucciarelli & 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:

$$\llbracket t \rrbracket := \{ (\Gamma, A) \mid \Pi \triangleright \Gamma \vdash t : A \}$$

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

If \( t =_{\text{operational}} u \), then \( \llbracket t \rrbracket = \llbracket u \rrbracket \)

*i.e.* \( \Pi \triangleright \Gamma \vdash t : A \iff \Pi' \triangleright \Gamma \vdash u : A \)

### Qualitative

\( \text{size}(\Pi) \# \text{size}(\Pi') \)

Idempotent types

### Quantitative

\( \text{size}(\Pi) > \text{size}(\Pi') \)

Non-idempotent types
Let $t$ be a **typable** term, then $t \Rightarrow \ldots \Rightarrow \text{result?}$

- **length?**
- **size?**

(Standard) Notation for Typing

$$\Pi \triangleright \Gamma \vdash t : A$$

Quantitative Notation for Typing

$$\Pi \triangleright \Gamma \vdash (C_1, \ldots, C_n) t : A$$

The **counters** $(C_1, \ldots, C_n)$ measure different behaviors of the program $t$
Bounding and Measuring Evaluation by Means of Quantitative Types

Let $\Pi \triangleright \Gamma \vdash_{(L,S)} t : A$, then $t \rightarrow \ldots \rightarrow \text{result?}$

<table>
<thead>
<tr>
<th>Prototype Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDEMPOTENT TYPES</td>
<td>(RES) $t$ evaluates to some result</td>
</tr>
<tr>
<td>NON-IDEMPOTENT TYPES with UPPER BOUNDS</td>
<td>(RES) and $\text{length} + \text{size} \leq \text{size}(\Pi)$</td>
</tr>
<tr>
<td>NON-IDEMPOTENT TYPES with SPLIT/EXACT MEASURES</td>
<td>(RES) and $\text{length} = L$ and $\text{size} = S$</td>
</tr>
</tbody>
</table>
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)
- Resource and explicit substitution calculi
- Pattern Matching features
- Control operators
- Global state
- Proof-nets
- Non-deterministic languages
- Probabilistic languages, Bayesian inference

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
Delia KESNER
IRIF, Université Paris Cité, CNRS

Some Principles of Quantitative Type Systems

Quantitative Types and Inhabitation

Quantitative Types for Measuring

Quantitative Types and Observational Equivalence

Conclusion

Observational Equivalence

$$t \equiv_{R_1} u \text{ iff } t \equiv_{R_2} u$$

Call-by-Name

Call-by-Value

Call-by-Need

Neededness

Call-by-Name

Call-by-Value

Call-by-Need

Neededness

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

\[
\begin{align*}
\text{Twice} (4 + 3) & \rightarrow_{\text{cbname}} (4 + 3) + (4 + 3) \rightarrow_{\text{cbname}} 7 + (4 + 3) \rightarrow_{\text{cbname}} 7 + 7 \rightarrow_{\text{cbname}} 14 \\
\text{Twice} (4 + 3) & \rightarrow_{\text{cbneed}} \text{Twice} 7 \rightarrow_{\text{cbneed}} 7 + 7 \rightarrow_{\text{cbneed}} 14 \\
\end{align*}
\]

where \( \text{Twice} = \lambda x. x + x. \)
Call-by-need is different from call-by-value:

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

In particular

\[(\lambda x.8)\Omega \rightarrow_{cbvalue}\]
\[(\lambda x.8)\Omega \rightarrow_{cbneed} 8\]
(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

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

*Theorem (K.'16, K.&Viso&Ríos’18)*

$t \approx_{\text{call-by-name}} u$ if and only if $t \approx_{\text{call-by-need}} u$ if and only if $t \approx_{\text{neededness}} u$. 
Some Principles of Quantitative Type Systems

Quantitative Types and Inhabitation

Quantitative Types for Measuring

Quantitative Types and Observational Equivalence

Conclusion
Concluding Remarks

**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