Quantitative Inhabitation

Victor Arrial    Delia Kesner

Joint work with Giulio Guerrieri

60 ans d’Antonio Bucciarelli, Paris, 20 Juin 2023
### Some Fruitful Collaborations on Intersection Types

<table>
<thead>
<tr>
<th>Year Range</th>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016 and 2017</td>
<td><strong>Non-Idempotent Intersection Types for Lambda-Calculus.</strong></td>
<td>Bucciarelli-Kesner-Ventura.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="https://upload.wikimedia.org/wikipedia/commons/thumb/b/b0/Flag_of_Brazil.svg/20px-Flag_of_Brazil.svg" alt="Brazil" /> <img src="https://upload.wikimedia.org/wikipedia/commons/thumb/1/11/Flag_of_France.svg/20px-Flag_of_France.svg" alt="France" /></td>
</tr>
<tr>
<td>2014 and 2018</td>
<td><strong>Inhabitation for Non-Idempotent Intersection Types.</strong></td>
<td>Bucciarelli-Kesner-RonchiDellaRocca</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="https://upload.wikimedia.org/wikipedia/commons/thumb/b/b0/Flag_of_Brazil.svg/20px-Flag_of_Brazil.svg" alt="Brazil" /> <img src="https://upload.wikimedia.org/wikipedia/commons/thumb/1/11/Flag_of_France.svg/20px-Flag_of_France.svg" alt="France" /> <img src="https://upload.wikimedia.org/wikipedia/commons/thumb/c/c9/Flag_of_Italy.svg/20px-Flag_of_Italy.svg" alt="Italy" /></td>
</tr>
<tr>
<td>2020</td>
<td><strong>The Bang Calculus Revisited.</strong></td>
<td>Bucciarelli-Kesner-Rios-Viso</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="https://upload.wikimedia.org/wikipedia/commons/thumb/b/b0/Flag_of_Brazil.svg/20px-Flag_of_Brazil.svg" alt="Brazil" /> <img src="https://upload.wikimedia.org/wikipedia/commons/thumb/1/11/Flag_of_France.svg/20px-Flag_of_France.svg" alt="France" /> <img src="https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Flag_of_Argentina.svg/20px-Flag_of_Argentina.svg" alt="Argentina" /></td>
</tr>
</tbody>
</table>
That Inspired an Amazing Paper ...
That Inspired an Amazing Paper ...

Trailer
That Inspired an Amazing Paper ...

Trailer
What is Inhabitation?
What is Inhabitation?

Typing Problem:

\[ t \]
What is Inhabitation?

Typing Problem:

\[ \Gamma \vdash t : \sigma \]
What is Inhabitation?

Typing Problem:

\[ \Gamma \vdash t : \sigma \]

Computational: [Milner78]

Typers
What is Inhabitation?

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

Inhabitation Problem (IP):

Computational: [Milner78]

Typers
What is Inhabitation?

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

Inhabitation Problem (IP):
\[ \Gamma \vdash \sigma \]

Computational: [Milner78]
Typers
What is Inhabitation?

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

Inhabitation Problem (IP):
\[ \Gamma \vdash t : \sigma \]

Computational: [Milner78]
Typers
What is Inhabitation?

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

Inhabitation Problem (IP):
\[ \Gamma \vdash t : \sigma \]

Computational: [Milner78]
Typers

Program Synthesis

Logical: [HodasMiller94]
Proof Search and Logic Programming
What is Inhabitation?

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

Inhabitation Problem (IP):
\[ \Gamma \vdash t : \sigma \]

Computational: [HughesOrchard20]
Program Synthesis

Logical: [HodasMiller94]
Proof Search and Logic Programming
What is Inhabitation?

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

Inhabitation Problem (IP):
\[ \Gamma \vdash t : \sigma \]

Computational: [HughesOrchard20]
Program Synthesis

Logical: [HodasMiller94]
Proof Search and Logic Programming
Quantitative Inhabitation for Different Lambda Calculi in a Unifying Framework
Quantitative Inhabitation for Different Lambda Calculi in a Unifying Framework
Unifying Frameworks
Unifying Frameworks

Different Models of Computation:

Call-by-Name

Call-by-Value

\[
\begin{array}{c}
\text{t, } u :: = x | \lambda x.t | tu | !t \\
\text{Values |
\text{der (t) |
\text{Computations |
\text{t [x: ] = u ] |
\end{array}
\]
Quantitative Inhabitation for Different Lambda Calculi in a Unifying Framework
Quantitative Inhabitation for **Different Lambda Calculi in a Unifying Framework**
Unifying Frameworks

Different Models of Computation:

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

Unifying Frameworks:

- Call-by-Push-Value [Levy99]
Unifying Frameworks

Different Models of Computation:

Call-by-Name

Call-by-Value

Unifying Frameworks:

- Call-by-Push-Value [Levy99]
- Bang Calculus [EhrhardGuerrieri16]

\[
\begin{array}{c}
t, u ::= x | \lambda x.t | tu | !t \\
\end{array}
\]
Different Models of Computation:

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

Unifying Frameworks:

- Call-by-Push-Value [Levy99]

- Bang Calculus [EhrhardGuerrieri16]:

\[
\begin{align*}
t, u & ::= x \mid \lambda x.t \mid tu \\
\end{align*}
\]
Different Models of Computation:

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

Unifying Frameworks:

- Call-by-Push-Value [Levy99]
- Bang Calculus [EhrhardGuerrieri16]:

\[
\begin{align*}
t, u & ::= x \mid \lambda x.t \mid tu \\
& \mid ! t \\
\end{align*}
\]

Values
Different Models of Computation:

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

Unifying Frameworks:

- Call-by-Push-Value [Levy99]

- Bang Calculus [EhrhardGuerrieri16]:

\[
t, u ::= x \mid \lambda x.t \mid tu \\
| \,!t \quad \text{Values} \\
| \text{der}(t) \quad \text{Computations}
\]
Different Models of Computation:

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

Unifying Frameworks:

- Call-by-Push-Value [Levy99]

- Distant Bang Calculus [EhrhardGuerrieri16] [BucciarelliKesnerRiosViso20,23]:

\[
t, u ::= x \mid \lambda x.t \mid tu \\
| !t \quad \text{Values} \\
| \text{der}(t) \quad \text{Computations} \\
| t[x:=u] \quad \text{Let}
\]
Different Models of Computation: Call-by-Need Revisited

Antonio Bucciarelli, Delia Kesner,1, 2, Alejandro Ríos,3, and Andrés Vigo3,4

1 Université de Paris, CNRS, IRIF, Paris, France
2 Institut Universitaire de France, Paris, France
3 Universidad de Buenos Aires, Buenos Aires, Argentina
4 Universidad Nacional de Quilmes, Buenos Aires, Argentina

Abstract. Call-by-Name (CBN) and Call-by-Value (CBV) are two fundamental paradigms in functional programming. In the context of the Bang Calculus, a linearized version of the original Bang Calculus, the paradigm was recently equipped by means of Linear Logic, subsuming both CBN and CBV versions in the original system. The paradigm connecting CBN and CBV versions in the original system. A second contribution of this work is a new language presenting a revisited version of the Bang Calculus.
The bang calculus revisited

Antonio Bucciarelli a, Delia Kesner a,b,*, Alejandro Ríos c, Andrés Viso d,*

a Université Paris Cité, CNRS, IRIF, France
b Institut Universitaire de France, France
c Universidad de Buenos Aires, Argentina
d Inria, France

ARTICLE INFO

Article history:
Received 27 September 2022
Received in revised form 26 April 2023
Accepted 4 May 2023
Available online 10 May 2023

Keywords:
Call-by-push-value
Bang calculus
Intersection types

ABSTRACT

Call-by-Push-Value (CBPV) is a programming paradigm subsuming both Call-by-Name (CBN) and Call-by-Value (CBV) semantics. The essence of this paradigm is captured by the Bang Calculus, a term language connecting CBPV and Linear Logic.

This paper presents a revisited version of the Bang Calculus, called \( \lambda_! \), enjoying some important properties missing in the original formulation. Indeed, the new calculus integrates permutative conversions to unblock value redexes while preserving confluence.

A second contribution is related to non-idempotent types. We provide a quantitative type system for our \( \lambda_! \)-calculus, giving upper bounds to the length of the reduction to normal form plus its size. We also explore the properties of this type system with respect to CBN/CBV translations. Last but not least, the quantitative system is refined to provide a reduction length and the normal form size which transforms the previous upper bound into two independent lower bounds.
Quantitative Inhabitation

Distant Bang: A Subsuming Paradigm

\[ N \rightarrow V \]

Static Properties:
[BucciarelliKesnerRiosViso20,23, Arrial23]

Normal form \( N \) \iff \( V \)

Dynamic Properties:
[BucciarelliKesnerRiosViso20,23, Arrial23]

\[ N \rightarrow u \] \iff \[ V \rightarrow u \]

Can we do the same thing with inhabitation?
Distant Bang: A Subsuming Paradigm

\[ t^N : \text{NAME} \rightarrow \text{BANG} \]
Distant Bang: A Subsuming Paradigm

\[ t^N \rightarrow \text{NAME} \rightarrow \text{BANG} \]

Static Properties: [BucciarelliKesnerRiosViso20,23, Arrial23]

\[ \text{NAME} \rightarrow t \text{ normal form} \]
Quantitative Inhabitation

Distant Bang: A Subsuming Paradigm

\[ t^N : \text{NAME} \rightarrow \text{BANG} \]

Static Properties: [BucciarelliKesnerRiosViso20,23, Arrial23]

\[ \text{NAME} \text{\ normal form} \iff t^N \text{\ normal form} \]

Can we do the same thing with inhabitation?
Distant Bang: A Subsuming Paradigm

$$t^N : \text{NAME} \rightarrow \text{BANG}$$

Static Properties: [BucciarelliKesnerRiosViso20,23, Arrial23]

$$\text{NAME} \quad t \text{ normal form} \iff t^N \text{ normal form} \quad \text{BANG}$$

Dynamic Properties: [BucciarelliKesnerRiosViso20,23, Arrial23]

$$\text{NAME} \quad t \rightarrow u$$
Distant Bang: A Subsuming Paradigm

**Static Properties:** [BucciarelliKesnerRiosViso20,23, Arrial23]

\[ t \to u \Leftrightarrow t^N \to u^N \]

**Dynamic Properties:** [BucciarelliKesnerRiosViso20,23, Arrial23]

\[ t \to u \Leftrightarrow t^N \to u^N \]
Quantitative Inhabitation

Distant Bang: A Subsuming Paradigm

\[
\begin{align*}
    t^N &: \text{NAME} &\rightarrow &\text{BANG} \\
    t^V &: \text{VALUE} &\rightarrow &\text{BANG}
\end{align*}
\]

Static Properties: [BucciarelliKesnerRiosViso20,23, Arrial23]

\[
\begin{align*}
    \text{NAME} \quad t \text{ normal form} &\iff t^N \text{ normal form} \quad \text{BANG}
\end{align*}
\]

Dynamic Properties: [BucciarelliKesnerRiosViso20,23, Arrial23]

\[
\begin{align*}
    \text{NAME} \quad t \rightarrow u &\iff t^N \rightarrow u^N \quad \text{BANG}
\end{align*}
\]
Quantitative Inhabitation

Distant Bang: A Subsuming Paradigm

$N$: $t^N : \textbf{NAME} \rightarrow \textbf{BANG}$

$V$: $t^V : \textbf{VALUE} \rightarrow \textbf{BANG}$

**Static Properties:** [BucciarelliKesnerRiosViso20,23, Arrial23]

- **NAME**: $t$ normal form $\iff t^N$ normal form
- **VALUE**: $t$ normal form $\iff t^V$ normal form

**Dynamic Properties:** [BucciarelliKesnerRiosViso20,23, Arrial23]

- **NAME**: $t \rightarrow u \iff t^N \rightarrow u^N$
- **VALUE**: $t \rightarrow u \iff t^V \rightarrow u^V$

Can we do the same thing with inhabitation?
Distant Bang: A Subsuming Paradigm

\[
\begin{align*}
\text{NAME: } t^N : & \rightarrow \text{BANG} \\
\text{VALUE: } t^V : & \rightarrow \text{BANG}
\end{align*}
\]

Static Properties: [BucciarelliKesnerRiosViso20,23, Arrial23]

\[
\begin{align*}
\text{NAME} & \quad t \text{ normal form } \iff t^N \text{ normal form} \\
\text{VALUE} & \quad t \text{ normal form } \iff t^V \text{ normal form}
\end{align*}
\]

Dynamic Properties: [BucciarelliKesnerRiosViso20,23, Arrial23]

\[
\begin{align*}
\text{NAME} & \quad t \rightarrow u \iff t^N \rightarrow u^N \\
\text{VALUE} & \quad t \rightarrow u \iff t^V \rightarrow u^V
\end{align*}
\]

Can we do the same thing with inhabitation?
Quantitative Inhabitation for Different Lambda Calculi in a Unifying Framework
Quantitative Inhabitation for Different Lambda Calculi in a Unifying Framework
Simple Types Versus Intersection Types

\[ A, B ::= \sigma \mid A \Rightarrow B \]

Untyped terms
Terminating terms
Typable terms

Idempotent
\[ A \cap A = A \]

Idempotent
\[ A \cap A, A \cap A \]

\[ \text{Commutativity:} \quad A \cap B = B \cap A \]

\[ \text{Associativity:} \quad A \cap (B \cap C) = (A \cap B) \cap C \]

\[ \text{Idempotency:} \quad A \cap A = A \]

\[ \text{[CoppoDezani78,80, Gardner94, Kfoury00]} \]

\[ \text{[deCarvalho07]} \]
Simple Types Versus Intersection Types

\[ A, B ::= \sigma \mid A \Rightarrow B \]

Untyped terms

Terminating terms

Typable terms

Qualitative properties

Quantitative properties

Idempotent

Idempotent

Non-Idempotent

\[ \text{Associativity:} \quad A \cap (B \cap C) = (A \cap B) \cap C \]

\[ \text{Commutativity:} \quad A \cap B = B \cap A \]

\[ \text{Idempotency:} \quad A \cap A = A \]

\[ \text{[CoppoDezani78,80]} \quad \text{[Gardner94]} \quad \text{[Kfoury00]} \]
Simple Types Versus Intersection Types

\[ A, B ::= \sigma \mid A \Rightarrow B \]

- Untyped terms
- Terminating terms
- Typable terms
Simple Types Versus *Intersection Types*

\[ A, B ::= \sigma \mid A \Rightarrow B \mid A \cap B \]

Untyped terms

Terminating terms

Typable terms

**Qualitative properties**

**Quantitative properties**

\[ \text{Idempotent} \]

\[ \text{Non-Idempotent} \]

[CoppoDezani78,80, Gardner94, Kfoury00]

[deCarvalho07]
Simple Types Versus **Intersection Types**

\[ A, B ::= \sigma \mid A \Rightarrow B \mid A \cap B \]

- **Untyped terms**
- **Terminating terms**
- **Typable terms**

### Qualitative properties
- Idempotent
- Non-Idempotent

### Quantitative properties
- Associativity
- Commutativity

[CoppoDezani78,80], [Gardner94], [Kfoury00]
Simple Types Versus Intersection Types

\[ A, B ::= \sigma \mid A \Rightarrow B \mid A \cap B \]

Untyped terms

Terminating terms

Typable terms

\[ A \cap (B \cap C) = (A \cap B) \cap C \]

\[ A \cap B = B \cap A \]

\[ A \cap A = A \]

[deCarvalho07]

[deCarvalho07, CoppoDezani78,80, Gardner94, Kfoury00]
Simple Types Versus Intersection Types

\[ A, B ::= \sigma \mid A \Rightarrow B \mid A \cap B \]

- Untyped terms
- Terminating terms
- Typable terms

**Associativity**

\[ A \cap (B \cap C) = (A \cap B) \cap C \]

**Commutativity**

\[ A \cap B = B \cap A \]

**Idempotency**

\[ A \cap A = A \]

- Idempotent
- Non-Idempotent

**Qualitative properties**

\[ \text{deCarvalho07} \]

**Quantitative properties**

\[ \text{CoppoDezani78,80, Gardner94, Kfoury00} \]
Simple Types Versus Intersection Types

\[ A, B ::= \sigma \mid A \Rightarrow B \mid A \cap B \]

**Associativity:**

\[ A \cap (B \cap C) = (A \cap B) \cap C \]
Simple Types Versus Intersection Types

\[ A, B ::= \sigma \mid \sigma \rightarrow B \mid A \cap B \]

- **Associativity:**
  \[ A \cap (B \cap C) = (A \cap B) \cap C \]

- **Commutativity:**
  \[ A \cap B = B \cap A \]

Untyped terms

Terminating terms

Typable terms

Qualitative properties

Quantitative properties

\[ \text{Idempotent} \]

\[ \text{Non-Idempotent} \]

\[ \text{[CoppoDezani78,80]} \]

\[ \text{[Gardner94]} \]

\[ \text{[Kfoury00]} \]
Simple Types Versus Intersection Types

\[ A, B ::= \sigma \mid A \Rightarrow B \mid A \cap B \]

- **Untyped terms**
- **Terminating terms**
- **Typable terms**

- **Associativity:**
  \[ A \cap (B \cap C) = (A \cap B) \cap C \]

- **Commutativity:**
  \[ A \cap B = B \cap A \]

- **Idempotency?**

[CoppoDezani78,80] [Gardner94] [Kfoury00] [deCarvalho07]

Qualitative properties
Quantitative properties
Simple Types Versus Intersection Types

\[ A, B ::= \sigma \mid A \Rightarrow B \mid A \cap B \]

- **Associativity:**
  \[ A \cap (B \cap C) = (A \cap B) \cap C \]

- **Commutativity:**
  \[ A \cap B = B \cap A \]

- **Idempotency?**

  
  
  - **Idempotent**
  - **Non-Idempotent**

  \[ A \cap A = A \]
Simple Types Versus Intersection Types

\[ A, B ::= \sigma \mid A \Rightarrow B \mid A \cap B \]

- **Associativity:**
  \[ A \cap (B \cap C) = (A \cap B) \cap C \]

- **Commutativity:**
  \[ A \cap B = B \cap A \]

- **Idempotency?**

  **Idempotent**
  [CoppoDezani78,80]

  \[ A \cap A = A \]

**Qualitative properties**

- ✓
- ✗
Simple Types Versus Intersection Types

\[
A, B ::= \sigma \mid A \Rightarrow B \mid A \cap B
\]

- **Untyped terms**
- **Terminating terms**
  - **Typable terms**

- **Associativity**:
  \[
  A \cap (B \cap C) = (A \cap B) \cap C
  \]

- **Commutativity**:
  \[
  A \cap B = B \cap A
  \]

- **Idempotency?**
  - **Idempotent** [CoppoDezani78,80]
    \[
    A \cap A = A
    \]
  - **Non-Idempotent** [Gardner94], [Kfoury00]
    \[
    A \cap A \neq A
    \]

**Qualitative properties**

- ![✓](true)
- ![✗](false)
Simple Types Versus Intersection Types

\[ A, B ::= \sigma \mid A \Rightarrow B \mid A \cap B \]

- **Untyped terms**
- **Terminating terms**
- **Typable terms**

<table>
<thead>
<tr>
<th>Idempotent</th>
<th>Non-Idempotent</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CoppoDezani78,80]</td>
<td>[Gardner94], [Kfoury00]</td>
</tr>
<tr>
<td>( A \cap A = A )</td>
<td>( A \cap A \neq A )</td>
</tr>
</tbody>
</table>

- **Associativity:**
  \[ A \cap (B \cap C) = (A \cap B) \cap C \]

- **Commutativity:**
  \[ A \cap B = B \cap A \]

- **Idempotency?**

**Qualitative properties**

**Quantitative properties**

[CoppoDezani78,80], [Gardner94], [Kfoury00]

[deCarvalho07]
Non-idempotent intersection types for the Lambda-Calculus

ANTONIO BUCCIARELLI* and DELIA KESNER**, Institut de Recherche en Informatique Fondamentale, CNRS and Université Paris Diderot, Paris, France.

DANIEL VENTURA†, Universidade Federal de Goiás, Instituto de Informática, Goiânia, Brazil.

Abstract

We study non-idempotent intersection types in the framework of the \(\lambda\)-calculus. Different topics are addressed, including normalization, weak normalization, weak head normalization, strong normalization, reducibility technique, traditionally used when working with idempotent intersection types.
<table>
<thead>
<tr>
<th>Typability</th>
<th>Inhabitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Types</td>
<td>Decidable</td>
</tr>
<tr>
<td></td>
<td>Idempotent Types</td>
</tr>
<tr>
<td>Non-Idempotent Types</td>
<td>Indecidable</td>
</tr>
<tr>
<td></td>
<td>(CBN) Decidable</td>
</tr>
<tr>
<td></td>
<td>(CBV) ?</td>
</tr>
</tbody>
</table>
Typability and Inhabitation in Intersection Types

<table>
<thead>
<tr>
<th>Typing</th>
<th>Inhabitation</th>
</tr>
</thead>
</table>
## Typability and Inhabitation in Intersection Types

<table>
<thead>
<tr>
<th>Typing</th>
<th>Inhabitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vdash t : ?$</td>
<td>$\Gamma \vdash ? : \sigma$</td>
</tr>
</tbody>
</table>

- **Simple Types**
- **Idempotent Types**
- **Non-Idempotent Types**

- **Decidable**
- **Indecidable** (CBN)
- **Indecidable** (CBV)

[Urzyczyn99] [BKR'18]
<table>
<thead>
<tr>
<th></th>
<th>Typing ( \alpha \vdash t : \tau )</th>
<th>Inhabitation ( \Gamma \vdash ? : \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Types</td>
<td>Decidable</td>
<td></td>
</tr>
<tr>
<td>Idempotent Types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-Idempotent Types</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Typability and Inhabitation in Intersection Types

<table>
<thead>
<tr>
<th></th>
<th>Typing $\vdash t : ?$</th>
<th>Inhabitation $\Gamma \vdash ? : \sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Types</td>
<td>Decidable</td>
<td></td>
</tr>
<tr>
<td>Idempotent Types</td>
<td></td>
<td>Indecidable</td>
</tr>
<tr>
<td>Non-Idempotent Types</td>
<td></td>
<td>Indecidable</td>
</tr>
<tr>
<td>Simple Types</td>
<td>Typing ( ? \vdash t : ? )</td>
<td>Inhabitation ( \Gamma \vdash ? : \sigma )</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td></td>
<td>Decidable</td>
<td>Decidable</td>
</tr>
<tr>
<td><strong>Idempotent Types</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Indecidable</td>
<td></td>
</tr>
<tr>
<td><strong>Non-Idempotent Types</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Indecidable</td>
<td></td>
</tr>
</tbody>
</table>
## Typability and Inhabitation in Intersection Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Typing $\ ? \vdash t : \ ?$</th>
<th>Inhabitation $\Gamma \vdash \ ? : \sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Types</td>
<td>Decidable</td>
<td>Decidable</td>
</tr>
<tr>
<td>Idempotent Types</td>
<td>Indecidable</td>
<td>Indecidable [Urzyczyn99]</td>
</tr>
<tr>
<td>Non-Idempotent Types</td>
<td>Indecidable</td>
<td></td>
</tr>
</tbody>
</table>
## Typability and Inhabitation in Intersection Types

<table>
<thead>
<tr>
<th></th>
<th>Typing $\vdash t : ?$</th>
<th>Inhabitation $\Gamma \vdash ? : \sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simple Types</strong></td>
<td>Decidable</td>
<td>Decidable</td>
</tr>
<tr>
<td><strong>Idempotent Types</strong></td>
<td>Indecidable</td>
<td>Indecidable [Urzyczyn99]</td>
</tr>
<tr>
<td><strong>Non-Idempotent Types</strong></td>
<td>Indecidable</td>
<td>(CBN) Decidable [BKR’18]</td>
</tr>
</tbody>
</table>
Typability and Inhabitation in Intersection Types

The Inhabitation Problem for Non-idempotent Intersection Types

Antonio Bucciarelli¹, Delia Kesner¹, and Simona Ronchi Della Rocca²

¹ Univ Paris Diderot, Sorbonne Paris Cit, PPS, UMR 7126, CNRS, Paris, France
² Dipartimento di Informatica, Università di Torino, Italy

Abstract. The inhabitation problem for intersection types is known to be undecidable. We study the problem in the case of non-idempotent intersection types, and we prove decidability through a sound and complete calculus for a subclass with pairs, and we prove the decidability is interesting in its own, since it...
ABSTRACT. The inhabitation problem for intersection types in \( \lambda \)-calculus is known to be undecidable. We study the problem in the case of non-idempotent intersection \( \lambda \)-terms. We prove the decidability of the inhabitation problem by providing sound and complete inhabitation for Non-idempotent Intersection Types.

ANTONIO BUCCIARELLI \(^a\), DELIA KESNER \(^b\), AND SIMONA RONCHI DELLA ROCCA \(^c\)

\(^a\) IRIF, CNRS and Univ Paris-Diderot, France
e-mail address: \{buccia,kesner\}@irif.fr

\(^b\) Dipartimento di Informatica, Università di Torino, Italy
e-mail address: ronchi@di.unito.it

In honour of Furio Honsell, in occasion of his 60th birthday
ABSTRACT. We extend the classical notion of solvability to a λ-calculus equipped with pattern matching. We prove that solvability can be characterized by means of typability and inhabitation in an intersection type system $P$ based on non-idempotent types. We show first that the system $P$ characterizes the set of terms having canonical form, i.e. that a term is typable if and only if it reduces to a canonical form. But the set of solvable terms is properly contained in the set of canonical forms. Thus, typability alone is not sufficient to characterize solvability, in contrast to the case for the λ-calculus. We then prove that typability, together with inhabitation, provides a full characterization of solvability, in the
## Typability and Inhabitation in Intersection Types

<table>
<thead>
<tr>
<th></th>
<th>Typing ( ? \vdash t : ? )</th>
<th>Inhabitation ( \Gamma \vdash ? : \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simple Types</strong></td>
<td>Decidable</td>
<td>Decidable</td>
</tr>
<tr>
<td><strong>Idempotent Types</strong></td>
<td>Indecidable</td>
<td>Indecidable ([\textbf{Urzyczyn99}])</td>
</tr>
<tr>
<td><strong>Non-Idempotent Types</strong></td>
<td>Indecidable</td>
<td>(CBN) Decidable ([\textbf{BKR’18}])</td>
</tr>
</tbody>
</table>
## Typability and Inhabitation in Intersection Types

<table>
<thead>
<tr>
<th></th>
<th>Typing $\ ? \vdash t : \ ?$</th>
<th>Inhabitation $\Gamma \vdash \ ? : \sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Types</td>
<td>Decidable</td>
<td>Decidable</td>
</tr>
<tr>
<td>Idempotent Types</td>
<td>Indecidable</td>
<td>Indecidable $[\text{Urzyczyn99}]$</td>
</tr>
<tr>
<td>Non-Idempotent Types</td>
<td>Indecidable</td>
<td>(CBN) Decidable $[\text{BKR’18}]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(CBV) ?</td>
</tr>
</tbody>
</table>
## Typability and Inhabitation in Intersection Types

<table>
<thead>
<tr>
<th></th>
<th>Typing $\vdash t : ?$</th>
<th>Inhabitation $\Gamma \vdash ? : \sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simple Types</strong></td>
<td>Decidable</td>
<td>Decidable</td>
</tr>
<tr>
<td><strong>Idempotent Types</strong></td>
<td>Indecidable</td>
<td>Indecidable [Urzyczyn99]</td>
</tr>
<tr>
<td><strong>Non-Idempotent Types</strong></td>
<td>Indecidable</td>
<td>(CBN) Decidable [BKR’18]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(CBV) ?</td>
</tr>
</tbody>
</table>
## Typability and Inhabitation in Intersection Types

<table>
<thead>
<tr>
<th></th>
<th>Typing $\ ? \vdash t : \ ?$</th>
<th>Inhabitation $\Gamma \vdash ? : \sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simple Types</strong></td>
<td>Decidable</td>
<td>Decidable</td>
</tr>
<tr>
<td><strong>Idempotent Types</strong></td>
<td>Indecidable</td>
<td>Indecidable [Urzyczyn99]</td>
</tr>
<tr>
<td><strong>Non-Idempotent Types</strong></td>
<td>Indecidable</td>
<td>(CBN) Decidable [BKR’18] (CBV) Decidable</td>
</tr>
</tbody>
</table>
Intersection Types and Distant Bang Calculus

Three Typing Systems:

[BucciarelliKesnerRiosViso20,23]:

\[ \Gamma \vdash N_t : \sigma \iff \Gamma \vdash B_t : \sigma \]

\[ \Gamma \vdash V_t : \sigma \iff \Gamma \vdash B_t : \sigma \]
Three Typing Systems: [BucciarelliKesnerRiosViso20,23]

\[\text{Name} : \mathcal{N} \quad \text{Value} : \mathcal{V} \quad \text{Bang} : \mathcal{B}\]
Intersection Types and Distant Bang Calculus

Three Typing Systems: \[\text{[BucciarelliKesnerRiosViso20,23]}\]

\[
\begin{array}{c}
\text{NAME} : N \\
\text{VALUE} : V \\
\text{BANG} : B
\end{array}
\]

Static Properties: \[\text{[BucciarelliKesnerRiosViso20,23]}\]

\[
\Gamma \vdash_N t : \sigma
\]
Intersection Types and Distant Bang Calculus

Three Typing Systems: [BucciarelliKesnerRiosViso20,23]

\[
\begin{align*}
\text{NAME} & : \mathcal{N} \\
\text{VALUE} & : \mathcal{V} \\
\text{BANG} & : \mathcal{B}
\end{align*}
\]

Static Properties: [BucciarelliKesnerRiosViso20,23]

\[
\Gamma \vdash_\mathcal{N} t : \sigma \iff \Gamma \vdash_\mathcal{B} t^N : \sigma
\]

\[
\begin{align*}
\text{NAME} \\
\Gamma \vdash_\mathcal{N} t : \sigma
\end{align*}
\]

\[
\begin{align*}
\text{BANG} \\
\Gamma \vdash_\mathcal{B} t^N : \sigma
\end{align*}
\]
Intersection Types and Distant Bang Calculus

Three Typing Systems: [BucciarelliKesnerRiosViso20,23]

- **NAME**: $N$
- **VALUE**: $V$
- **BANG**: $B$

Static Properties: [BucciarelliKesnerRiosViso20,23]

- $\Gamma \vdash_N t : \sigma \iff \Gamma \vdash_B t^N : \sigma$
- $\Gamma \vdash_V t : \sigma \iff \Gamma \vdash_B t^V : \sigma$
Quantitative Inhabitation for Different Lambda Calculi in a Unifying Framework
Coming Back to Inhabitation

First Goal

- **Decidability** of the (more general) BANG Inhabitation Problem (IP).
Coming Back to Inhabitation

First Goal + More Ambitious Second Goal

- **Decidability** of the (more general) BANG Inhabitation Problem (IP).
- **Decidability** of the **NAME** and **VALUE** IP from **decidability** of the BANG IP.

Using generic properties so that other encodable models of computation can use these results.
Coming Back to Inhabitation

First Goal + More Ambitious Second Goal

- **Decidability** of the (more general) **BANG** Inhabitation Problem (IP).
- **Decidability** of the **NAME** and **VALUE** IP from **decidability** of the **BANG** IP.

More Ambitious Third Goal
Coming Back to Inhabitation

First Goal + More Ambitious Second Goal

- **Decidability** of the (more general) \textbf{BANG} Inhabitation Problem (IP).
- **Decidability** of the \textbf{NAME} and \textbf{VALUE} IP from **decidability** of the \textbf{BANG} IP.

More Ambitious Third Goal

- Decidability by **finding all inhabitants** in the \textbf{BANG} IP.
Coming Back to Inhabitation

First Goal + More Ambitious Second Goal

- **Decidability** of the (more general) **BANG** Inhabitation Problem (IP).
- **Decidability** of the **NAME** and **VALUE** IP from **decidability** of the **BANG** IP.

More Ambitious Third Goal

- Decidability by **finding all inhabitants** in the **BANG** IP.
- Decidability of the **NAME** and **VALUE** IP by **finding all inhabitants** from those of the **BANG** IP.
**Coming Back to Inhabitation**

**First Goal + More Ambitious Second Goal**

- **Decidability** of the (more general) **BANG** Inhabitation Problem (IP).
- **Decidability** of the **NAME** and **VALUE** IP from **decidability** of the **BANG** IP.

**More Ambitious Third Goal**

- Decidability by **finding all inhabitants** in the **BANG** IP.
- Decidability of the **NAME** and **VALUE** IP by **finding all inhabitants** from those of the **BANG** IP.
- Using generic properties so that other encodable models of computation can use these results.
Instead of just one solution:

\[ \Gamma \vdash t : \sigma \]

We want to compute all solutions:

\[ \text{Sol}(\Gamma, \sigma) = \{ t | \Gamma \vdash t : \sigma \} \]

The set \( \text{Sol}(\Gamma, \sigma) \) is either empty or infinite.

We compute a finite generator:

\( \text{Basis}(\Gamma, \sigma) \)

Which is correct and complete:

\[ \text{span}(\text{Basis}(\Gamma, \sigma)) = \text{Sol}(\Gamma, \sigma) \]

Theorem

For any typing \( (\Gamma, \sigma) \), \( \text{Basis}(\Gamma, \sigma) \) exists, is finite, correct, and complete.
Instead of just one solution:
\[ \Gamma \vdash t : \sigma \]

We want to compute all solutions:
\[ \text{Sol}(\Gamma, \sigma) := \{ t \mid \Gamma \vdash t : \sigma \} \]
Instead of just one solution:
\[ \Gamma \vdash t : \sigma \]
We want to compute all solutions:
\[ \text{Sol}(\Gamma, \sigma) := \{ t \mid \Gamma \vdash t : \sigma \} \]
Solving the Inhabitation Problem - Methodology

Instead of just one solution:
\[ \Gamma \vdash t : \sigma \]

We want to compute all solutions:
\[ \text{Sol}(\Gamma, \sigma) := \{ t \mid \Gamma \vdash t : \sigma \} \]

Problem

✗ The set \( \text{Sol}(\Gamma, \sigma) \) is either empty or infinite
Instead of just one solution:
\[ \Gamma \vdash t : \sigma \]
We want to compute all solutions:
\[
\text{Sol}(\Gamma, \sigma) := \{ t \mid \Gamma \vdash t : \sigma \}
\]

Problem

The set \( \text{Sol}(\Gamma, \sigma) \) is either empty or infinite

We compute a finite generator:
\[
\text{Basis}(\Gamma, \sigma)
\]
Which is correct and complete:
\[
\text{span}(\text{Basis}(\Gamma, \sigma)) = \text{Sol}(\Gamma, \sigma)
\]
Instead of just one solution:
$$\Gamma \vdash t : \sigma$$
We want to compute all solutions:
$$\text{Sol}(\Gamma, \sigma) := \{ t \mid \Gamma \vdash t : \sigma \}$$

Problem

The set $\text{Sol}(\Gamma, \sigma)$ is either empty or infinite

We compute a finite generator:
$$\text{Basis}(\Gamma, \sigma)$$
Which is correct and complete:
$$\text{span}(\text{Basis}(\Gamma, \sigma)) = \text{Sol}(\Gamma, \sigma)$$
Instead of *just one* solution:

\[ \Gamma \vdash t : \sigma \]

We want to compute *all* solutions:

\[ \text{Sol}(\Gamma, \sigma) := \{ t \mid \Gamma \vdash t : \sigma \} \]

**Problem**

❌ The set \( \text{Sol}(\Gamma, \sigma) \) is either empty or infinite

We compute a *finite* generator:

\[ \text{Basis}(\Gamma, \sigma) \]

Which is *correct* and *complete*:

\[ \text{span}(\text{Basis}(\Gamma, \sigma)) = \text{Sol}(\Gamma, \sigma) \]
Solving the Inhabitation Problem - Methodology

Instead of just one solution:
\[ \Gamma \vdash t : \sigma \]
We want to compute all solutions:
\[ \text{Sol}(\Gamma, \sigma) := \{ t \mid \Gamma \vdash t : \sigma \} \]

Problem

- The set \( \text{Sol}(\Gamma, \sigma) \) is either empty or infinite

We compute a finite generator:
\[ \text{Basis}(\Gamma, \sigma) \]
Which is correct and complete:
\[ \text{span}(\text{Basis}(\Gamma, \sigma)) = \text{Sol}(\Gamma, \sigma) \]

Theorem

- For any typing \((\Gamma, \sigma)\), \( \text{Basis}_{\mathcal{B}}(\Gamma, \sigma) \) exists, is finite, correct and complete.
Solving the Inhabitation Problem - Methodology

Instead of just one solution:
\[ \Gamma \vdash t : \sigma \]
We want to compute all solutions:
\[ \text{Sol}(\Gamma, \sigma) := \{ t \mid \Gamma \vdash t : \sigma \} \]

Problem

- The set \( \text{Sol}(\Gamma, \sigma) \) is either empty or infinite

We compute a finite generator:

\[ \text{Basis}(\Gamma, \sigma) \]

Which is correct and complete:

\[ \text{span}(\text{Basis}(\Gamma, \sigma)) = \text{Sol}(\Gamma, \sigma) \]

Theorem

- For any typing \((\Gamma, \sigma)\), \( \text{Basis}_B(\Gamma, \sigma) \) exists, is finite, correct and complete.
Following the Typing and a Grammar

Quantitative Inhabitation

Following the Typing and a Grammar

Computing the basis:

Recreate typing trees, but only on elements of the Basis.

Follows two sets of rules:

Typing rules
Grammar rules

Sol \((\Gamma, \sigma)\)

Basis B
Computing the basis:
Recreate typing trees, but only on elements of the Basis.
Following the Typing and a Grammar

**Computing the basis:**
Recreate typing trees, but only on elements of the Basis.

follows two sets of rules:
Computing the basis:
Recreate typing trees, but only on elements of the Basis.

Follows two sets of rules:
- Typing rules

\[ \text{Sol}(\Gamma, \sigma) \]
Following the Typing and a Grammar

**Computing the basis:**
Recreate typing trees, but only on elements of the Basis.

Follows two sets of rules:
- Typing rules
- Grammar rules
Following the Typing and a Grammar

**Computing the basis:**
Recreate **typing trees**, but only on **elements of the Basis**.

Follows two sets of rules:
- Typing rules
- Grammar rules
The Full Algorithm
The Full Algorithm

\[
g \rightarrow \text{Var} \quad \sigma \vdash x \text{ H}^{\kappa}[\sigma](\varnothing; \sigma) \quad \sigma \vdash y \text{ S}(\tau; \varnothing) \quad a \text{ H}^{\kappa}[\tau](\Gamma; \sigma) \quad \text{der}(a) \vdash a \text{ N}(\Gamma; \sigma)
\]

\[
g \rightarrow g' \quad a \text{ H}^{\kappa}[\tau](\Gamma; \sigma) \quad a \text{ N}(\Gamma; \sigma) \quad g \text{ N}(\Gamma; \sigma)
\]

\[
g \rightarrow g' \quad a \text{ H}^{\kappa}[\tau](\Gamma; \sigma) \quad a \text{ N}(\Gamma; \sigma) \quad g \text{ N}(\Gamma; \sigma)
\]

\[
g \rightarrow \text{App}(g_a, g_b) \quad \Gamma = \Gamma_a + \Gamma_b \quad \mathcal{M} \Rightarrow \sigma \vdash S(\tau, \Diamond \Rightarrow \sigma) \quad a \vdash g_a \text{ H}^{\kappa}[\tau](\Gamma_a; \mathcal{M} \Rightarrow \sigma) \quad b \vdash g_b \text{ N}(\Gamma_b; \mathcal{M})
\]

\[
ab \vdash g \text{ H}^{\kappa}[\tau](\Gamma; \sigma)
\]

\[
g \rightarrow \text{Sub}(g_a, g_b) \quad \Gamma = \Gamma_a + \Gamma_b \quad n \in [1, sz(\tau)] \quad a \vdash g_a \text{ H}^{\kappa}[\tau](\Gamma_a[y = \{\rho_i\}_{i\in[1,n]}], \mathcal{M}; \sigma) \quad b \vdash g_b \text{ H}^{\kappa}[\tau](\Gamma_b; \mathcal{M})
\]

\[
g \rightarrow \text{Sub}(g_a, g_b) \quad \Gamma = \Gamma_a + \Gamma_b + \{x\} \quad n \in [1, sz(\tau)] \quad a \vdash g_a \text{ N}(\Gamma_a, y : \mathcal{M}; \sigma) \quad b \vdash g_b \text{ H}^{\kappa}[\tau](\Gamma_b; \mathcal{M})
\]

\[
\text{ES-H}
\]

\[
\text{ES-N}
\]

\[
\text{ES-CN}
\]
The Full Algorithm
The Full Algorithm

\[ g \looparrowright \text{App}(g_a, g_b) \]
\[ \Gamma = \Gamma_a + \Gamma_b \]
\[ \mathcal{M} \Rightarrow \sigma \quad a \vdash_{g_a} H^{x_1}[\tau] (\Gamma_a; \mathcal{M} \Rightarrow \sigma) \quad b \vdash_{g_b} N(\Gamma_b; \mathcal{M}) \]
\[ a b \vdash_g H^{x_1}[\tau] (\Gamma; \sigma) \]

\[ g \vdash \text{Sub}(g_a, g_b) \]
\[ \Gamma = \Gamma_a + \Gamma_b \]
\[ n \in [1, \text{sz}(\tau)], \quad \mathcal{M} \vdash \text{S}(\tau, \rho_1, \ldots, \rho_n) \]
\[ a \vdash_{g_a} H^{x_1}[\tau] (\Gamma_a; \mathcal{M} \Rightarrow \sigma) \quad b \vdash_{g_b} H^{x_1}[\tau] (\Gamma_b; \mathcal{M}) \]

\[ g \vdash \text{Sub}(g_a, g_b) \]
\[ \Gamma = \Gamma_a + \Gamma_b + \epsilon \]
\[ n \in [0, \text{sz}(\tau)], \quad \mathcal{M} \vdash \text{S}(\tau, \rho_1, \ldots, \rho_n) \]
\[ a \vdash_{g_a} N(\Gamma_a; \mathcal{M} \Rightarrow \sigma) \quad b \vdash_{g_b} H^{x_1}[\tau] (\Gamma_b; \mathcal{M}) \]
The Full Algorithm

\[ g \mapsto \text{App}(g_a, g_b) \]
\[ \Gamma = \Gamma_a + \Gamma_b \]
\[ \mathcal{M} \Rightarrow \sigma \models S(\tau, \diamond \Rightarrow \sigma) \quad a \models_{g_a} H^{x:[\tau]}_{\mathcal{M}}(\Gamma_a; \mathcal{M} \Rightarrow \sigma) \quad b \models_{g_b} N(\Gamma_b; \mathcal{M}) \]

\[ ab \models_{g} H^{x:[\tau]}(\Gamma; \sigma) \]

\[ g \mapsto \text{Sub}(g_a, g_b) \]
\[ \Gamma = \Gamma_a + \Gamma_b + x : [\tau] \]
\[ n \in [0, \text{sz}(\tau)), \mathcal{M} \models S(\tau_n, [\Phi_1, \ldots, \Phi_n]) \quad a \models_{g_a} H^{x:[\tau]}(\Gamma_a, y : [\Phi_1]; \mathcal{M}; \sigma) \quad b \models_{g_b} H^{x:[\tau]}(\Gamma_b; [\Phi_1]; \mathcal{M}) \]

\[ a[y/b] \models_{g} H^{x:[\tau]}(\Gamma; \sigma) \]

\[ g \mapsto \text{Sub}(g_a, g_b) \]
\[ \Gamma = \Gamma_a + \Gamma_b + x : [\tau] \]
\[ n \in [0, \text{sz}(\tau)), \mathcal{M} \models S(\tau_n, [\Phi_1, \ldots, \Phi_n]) \quad a \models_{g_a} N(\Gamma_a, y : \mathcal{M}; \sigma) \quad b \models_{g_b} H^{x:[\tau]}(\Gamma_b; \mathcal{M}) \]

\[ a[y/b] \models_{g} N(\Gamma; \sigma) \]
The Full Algorithm
The Full Algorithm and its Implementation

An Implementation of the Quantitative Inhabitation Algorithm for Different Lambda Calculi in a Unifying Framework

github/ArrialVictor/InhabitationLambdaBang
Properties of the Inhabitation Algorithm

Non-deterministic algorithm

Theorem
The inhabitation algorithm terminates. The algorithm is sound and complete (i.e. it exactly computes $B(\Gamma, \sigma)$).

More Ambitious Third Goal
Decidability by finding all inhabitants in the IP.
Decidability of the and IP by finding all inhabitants from those of the IP.

Using generic properties so that other encodable models of computation can use these results.
Properties of the Inhabitation Algorithm

Non-deterministic algorithm
Properties of the Inhabitation Algorithm

Non-deterministic algorithm

Theorem

The inhabitation algorithm terminates.
Properties of the Inhabitation Algorithm

Non-deterministic algorithm

Theorem

- The inhabitation algorithm terminates.
- The algorithm is **sound and complete** (i.e. it exactly computes $\text{Basis}_B(\Gamma, \sigma)$).
Properties of the Inhabitation Algorithm

Non-deterministic algorithm

Theorem

- The inhabitation algorithm terminates.
- The algorithm is sound and complete (i.e. it exactly computes $\text{Basis}_B(\Gamma, \sigma)$).

More Ambitious Third Goal

- Decidability by finding all inhabitants in the BANG IP.
Properties of the Inhabitation Algorithm

Non-deterministic algorithm

Theorem

- The inhabitation algorithm terminates.
- The algorithm is sound and complete (i.e. it exactly computes $\text{Basis}_B(\Gamma, \sigma)$).

More Ambitious Third Goal

- Decidability by finding all inhabitants in the BANG IP.
- Decidability of the NAME and VALUE IP by finding all inhabitants from those of the BANG IP.
- Using generic properties so that other encodable models of computation can use these results.
Solving Inhabitation - Standard Methodology

For any typing \((\Gamma,\sigma)\), Basis \(N(\Gamma,\sigma)\) exists, is finite, correct and complete.

Built an algorithm computing \(N(\Gamma,\sigma)\):

\[\text{[BucciarelliKesnerRios14]}\]
Theorem ([BucciarelliKesnerRios14])

For any typing \((\Gamma, \sigma)\), \(\text{Basis}_N(\Gamma, \sigma)\) exists, is finite, correct and complete.
Theorem ([BucciarelliKesnerRios14])

For any typing \((\Gamma, \sigma)\), \text{Basis}_N(\Gamma, \sigma) exists, is finite, correct and complete.

Built an algorithm computing \text{Basis}_N(\Gamma, \sigma): [BucciarelliKesnerRios14]

\[
\frac{a \vdash T(\Gamma + x : A, \tau)}{\lambda x. a \vdash T(\Gamma, A \rightarrow \tau)} \quad \text{(Abs)}
\]

\[
\frac{(a_i \vdash T(\Gamma_i, \sigma_i))_{i \in I}}{\bigvee_{i \in I} a_i} \quad \text{(Union)}
\]

\[
\Gamma = \Gamma_1 + \Gamma_2 \quad a \vdash H^{x:A_1 \rightarrow \ldots A_n \rightarrow B \rightarrow \tau}(\Gamma_1, B \rightarrow \tau) \quad b \vdash T(\Gamma_2, B) \quad n \geq 0 \quad \text{(Head}_{>0})
\]

\[
\frac{ab \vdash H^{x:A_1 \rightarrow \ldots A_n \rightarrow B \rightarrow \tau}(\Gamma, \tau)}{x \vdash H^{x}[\tau](\emptyset, \tau)} \quad \text{(Head}_0)
\]

\[
\frac{a \vdash H^{x:A_1 \rightarrow \ldots A_n \rightarrow \tau}(\Gamma, \tau)}{a \vdash T(\Gamma + x : [A_1 \rightarrow \ldots A_n \rightarrow \tau], \tau)} \quad \text{(Head)}
\]
Quantitative Inhabitation: Solving Inhabitation through Basis preservation by the embedding:

\[ t \in \text{Basis} \mathcal{N}(\Gamma, \sigma) \iff t \mathcal{N} \in \text{Basis} \mathcal{B}(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

**Theorem**

\[ t \in \text{Basis}_N(\Gamma,\sigma) \]
The Basis is preserved by the embedding:

**Theorem**

\[ t \in \text{Basis}_N(\Gamma, \sigma) \iff t^N \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

Theorem

\[ t \in \text{Basis}_N(\Gamma, \sigma) \iff t^N \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

**Theorem**

\[
t \in \text{Basis}_N(\Gamma, \sigma) \iff t^N \in \text{Basis}_B(\Gamma, \sigma)
\]
The Basis is preserved by the embedding:

Theorem

\[ t \in \text{Basis}_N(\Gamma, \sigma) \iff t^N \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

Theorem

\[ t \in \text{Basis}_N(\Gamma, \sigma) \iff t^N \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

**Theorem**

\[ t \in \text{Basis}_N(\Gamma, \sigma) \iff t^N \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

**Theorem**

\[ t \in \text{Basis}_N(\Gamma, \sigma) \iff t^N \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

Theorem

\[ t \in \text{Basis}_N(\Gamma, \sigma) \iff t^N \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

Theorem

\[ t \in \text{Basis}_N(\Gamma, \sigma) \iff t^N \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

Theorem

\[ t \in \text{Basis}_N(\Gamma, \sigma) \iff t^N \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

**Theorem**

\[ t \in \text{Basis}_N(\Gamma, \sigma) \iff t^N \in \text{Basis}_B(\Gamma, \sigma) \]
Theorem

For any typing \((\Gamma, \sigma)\), basis \(V(\Gamma, \sigma)\) exists, is finite, correct and complete.

Built an algorithm computing \(V(\Gamma, \sigma)\):
Quantitative Inhabitation

Solving Inhabitation - Usual Methodology

Theorem

For any typing \((\Gamma, \sigma)\), \(\text{Basis}_{\mathcal{V}}(\Gamma, \sigma)\) exists, is finite, correct and complete.
Theorem

For any typing \((\Gamma, \sigma)\), \(\text{Basis}_\mathcal{V}(\Gamma, \sigma)\) exists, is finite, correct and complete.

Built an algorithm computing \(\text{Basis}_\mathcal{V}(\Gamma, \sigma)\):
Theorem

For any typing \((\Gamma, \sigma)\), \(\text{Basis}_V(\Gamma, \sigma)\) exists, is finite, correct and complete.

Built an algorithm computing \(\text{Basis}_V(\Gamma, \sigma)\):

\[
\begin{align*}
\Gamma = \Gamma_1 + \Gamma_2, & \quad \text{fix } x \notin \text{dom}(\Gamma) \cup \{x\} \\
\sigma \vdash \tau, & \quad n \in [0, \text{sz}(\rho)], \quad M \vdash \sigma(\rho, \Omega_1, \ldots, \Omega_n) \\
\end{align*}
\]
Solving Inhabitation : through Inhabitation
The Basis is preserved by the embedding:

Theorem

\[ t \in \text{Basis}_V(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

**Theorem**

\[ t \in \text{Basis}_V(\Gamma, \sigma) \iff t^V \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

**Theorem**

\[ t \in \text{Basis}_V(\Gamma, \sigma) \iff t^V \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

**Theorem**

\[ t \in \text{Basis}_V(\Gamma, \sigma) \iff t^V \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

**Theorem**

\[ t \in \text{Basis}_V(\Gamma, \sigma) \iff t^V \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

Theorem

\[ t \in \text{Basis}_V(\Gamma, \sigma) \iff t^v \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

\[ t \in \text{Basis}_V(\Gamma, \sigma) \iff t^V \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

**Theorem**

\[
t \in \text{Basis}_V(\Gamma, \sigma) \iff t^V \in \text{Basis}_B(\Gamma, \sigma)
\]
The Basis is preserved by the embedding:

**Theorem**

\[ t \in \text{Basis}_V(\Gamma, \sigma) \iff t^V \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

Theorem

\[ t \in \text{Basis}_V(\Gamma, \sigma) \iff t^V \in \text{Basis}_B(\Gamma, \sigma) \]
The Basis is preserved by the embedding:

**Theorem**

\[
\text{VALUE} \quad t \in \text{Basis}_V(\Gamma, \sigma) \iff \overset{\text{V}}{t} \in \text{Basis}_B(\Gamma, \sigma) \quad \text{BANG}
\]
The Basis is preserved by the embedding:

Theorem

\[ t \in \text{Basis}_V(\Gamma, \sigma) \iff t^V \in \text{Basis}_B(\Gamma, \sigma) \]
Properties of the Indirect **NAME** and **VALUE** Algorithm

The inhabitation algorithm terminates. The algorithm is sound and complete (i.e. it exactly computes $\text{Basis} (\Gamma, \sigma)$).

More Ambitious Third Goal

Decidability by finding all inhabitants in the IP.

Decidability of the and IP by finding all inhabitants from those of the IP.

Using generic properties so that other encodable models of computation can use these results.
Properties of the Indirect NAME and VALUE Algorithm

Theorem

- The inhabitation algorithm terminates.
- The algorithm is sound and complete (i.e. it exactly computes $\text{Basis}_{G}(\Gamma, \sigma)$).
Theorem

- The inhabitation algorithm terminates.
- The algorithm is sound and complete (i.e. it exactly computes $\text{Basis}_G(\Gamma, \sigma)$).

More Ambitious Third Goal

- Decidability by finding all inhabitants in the BANG IP.
- Decidability of the NAME and VALUE IP by finding all inhabitants from those of the BANG IP.
- Using generic properties so that other encodable models of computation can use these results.
Properties of the Indirect **NAME** and **VALUE** Algorithm

**Theorem**

- The inhabitation algorithm terminates.
- The algorithm is sound and complete (i.e. it exactly computes Basis \((\Gamma,\sigma)\)).

**More Ambitious Third Goal**

- Decidability by **finding all inhabitants** in the **BANG** IP.
- Decidability of the **NAME** and **VALUE** IP by **finding all inhabitants** from those of the **BANG** IP.
- Using generic properties so that other encodable models of computation can use these results.
Properties of the Indirect NAME and VALUE Algorithm

Theorem

- The inhabitation algorithm terminates.
- The algorithm is sound and complete (i.e. it exactly computes Basis \((\Gamma, \sigma)\)).

More Ambitious Third Goal

- Decidability by finding all inhabitants in the BANG IP.
- Decidability of the NAME and VALUE IP by finding all inhabitants from those of the BANG IP.
- Using generic properties so that other encodable models of computation can use these results.
Properties of the Indirect NAME and VALUE Algorithm

Theorem

- The inhabitation algorithm terminates.
- The algorithm is sound and complete (i.e. it exactly computes Basis \((\Gamma, \sigma)\)).

More Ambitious Third Goal

- Decidability by finding all inhabitants in the BANG IP.
- Decidability of the NAME and VALUE IP by finding all inhabitants from those of the BANG IP.
  - Using generic properties so that other encodable models of computation can use these results.
Properties of the Indirect **NAME** and **VALUE** Algorithm

**Theorem**
- The inhabitation algorithm terminates.
- The algorithm is sound and complete (i.e. it exactly computes Basis \((\Gamma, \sigma)\)).

**More Ambitious Third Goal**
- Decidability by **finding all inhabitants** in the **BANG** IP.
- Decidability of the **NAME** and **VALUE** IP by **finding all inhabitants** from those of the **BANG** IP.
- Using generic properties so that other encodable models of computation can use these results.
Properties of the Indirect NAME and VALUE Algorithm

Theorem

- The inhabitation algorithm terminates.
- The algorithm is sound and complete (i.e. it exactly computes Basis \((\Gamma, \sigma)\)).

More Ambitious Third Goal

- Decidability by finding all inhabitants in the BANG IP.
- Decidability of the NAME and VALUE IP by finding all inhabitants from those of the BANG IP.
- Using generic properties so that other encodable models of computation can use these results.
Conclusion
Summary:

- Solving the generalized inhabitation problem
- A several-for-one deal: BANG NAME VALUE OTHERS
- An implementation: (github/ArrialVictor/InhabitationLambdaBang)
Summary:
- Solving the generalized inhabitation problem
- A several-for-one deal: BANG NAME VALUE OTHERS
- An implementation: (github/ArrialVictor/InhabitationLambdaBang)

Further questions and ongoing work:
- Solvability (for Different Calculi in a Unified Framework)
Summary:
- Solving the generalized inhabitation problem
- A several-for-one deal: BANG NAME VALUE OTHERS
- An implementation: (github/ArrialVictor/InhabitationLambdaBang)

Further questions and ongoing work:
- Solvability (for Different Calculi in a Unified Framework)

Thanks for your attention!
Thank you!
Happy Birthday!

Thank you!