Computational and Fine-Structure Aspects of Intersection Types
A personal encounter with intersection types

Jakob Rehof

Technische Universität Dortmund

TLT – Types and Logic in Torino
Colloquium in honor of Mariangiola Dezani-Ciancaglini, Simona Ronchi Della Rocca and Mario Coppo
Torino, Italy, September 22, 2017
Intersection types combine great logical simplicity and beauty with enormous expressive power, capturing deep semantic properties of $\lambda$-terms (normalization, solvability, ...)

A NEW TYPE ASSIGNMENT FOR $\lambda$-TERMS

M. Coppo and M. Dezani-Ciancaglini

Abstract

In the present paper we propose a new type assignment for $\lambda$-terms whose motivation is to introduce a system with simple inferential rules to study termination (i.e. the property of having a normal form) of $\lambda$-terms. The main results that will be proved in this paper are:

a) all $\lambda$-terms in normal form possess a type,

b) all $\lambda$-terms which possess a type reduce to normal form.

FUNCTIONAL CHARACTERISTICS OF SOLVABLE TERMS

by M. COPPO, M. DEZANI-CIANCAGLINI and R. VENNERI in Torino (Italy)
Motivations

The classical decision problems (typability and inhabitation) are undecidable for intersection types. Still, many interesting and useful problems can be solved computationally.

- Fine structure: Explore borderline between decidability and undecidability.
- Computational aspects: Algorithms & complexity of components and restrictions of the system.
- Applications: Intersection types as specifications (in typability, type checking, program analysis, refinement, synthesis, ...)


My students in Dortmund (former and present) including:
Jan Bessai (Dortmund), Boris Düdder (formerly Dortmund, now Copenhagen), Andrej Dudenhofner (Dortmund), Moritz Martens (formerly Dortmund, now in industry)

Collaborators and colleagues, including:
Mariangiola Dezani, Simona Ronchi Della Rocca, Mario Coppo and the Torino \(\lambda\)-calculus group, Tzu-Chun Chen (Darmstadt), George Heineman (WPI Boston), Ugo de’Liguoro (Torino), Paweł Urzyczyn, Aleksy Schubert and the Warsaw group, and Roger Hindley (Swansey)
A FILTER LAMBDA MODEL AND THE COMPLETENESS
OF TYPE ASSIGNMENT

HENK BARENDREGT, MARIO COPPO AND MARIANGIOLA DEZANI-CIANCAGLINI

In [6, p. 317] Curry described a formal system assigning types to terms of the
type-free $\lambda$-calculus. In [11] Scott gave a natural semantics for this type assign-
ment and asked whether a completeness result holds.

Inspired by [4] and [5] we extend the syntax and semantics of the Curry types
in such a way that filters in the resulting type structure form a domain in the sense
of Scott [12]. We will show that it is possible to turn the domain of types into a
$\lambda$-model, among other reasons because all $\lambda$-terms possess a type. This model
gives the completeness result for the extended system. By a conservativity result
the completeness for Curry's system follows.

[BCDC83]
Intersection Types

Definition

The set $\mathbb{T}$ of intersection types, ranged over by $\sigma, \tau, \rho$, is given by

$$ \mathbb{T} \ni \sigma, \tau, \rho ::= a \mid \alpha \mid \omega \mid \sigma \to \tau \mid \sigma \cap \tau $$

where $a, b, c, \ldots$ range over type constants drawn from the set $\mathbb{C}$, $\omega$ is a special (universal type) constant, and $\alpha, \beta, \gamma$ range over type variables drawn from the set $\mathbb{V}$.

As a matter of notational convention, function types associate to the right, and $\cap$ binds stronger than $\to$. A type $\tau \cap \sigma$ is said to have $\tau$ and $\sigma$ as components. Intersection $\cap$ is tacitly ACI.
**λ-Calculus with Intersection Types**

**Definition ([CDCV80],[BCDC83], . . .)**

(Var) \[ \Gamma, x : \tau \vdash x : \tau \]  

(→I) \[ \Gamma, x : \sigma \vdash M : \tau \]  
\[ \Gamma \vdash \lambda x. M : \sigma \rightarrow \tau \]  

(→E) \[ \Gamma \vdash M : \sigma \rightarrow \tau \]  
\[ \Gamma \vdash N : \sigma \]  
\[ \Gamma \vdash M \ N : \tau \]  

(∩I) \[ \Gamma \vdash M : \tau_1 \]  
\[ \Gamma \vdash M : \tau_2 \]  
\[ \Gamma \vdash M : \tau_1 \cap \tau_2 \]  

(∩E) \[ \Gamma \vdash M : \tau_1 \cap \tau_2 \]  
\[ \Gamma \vdash M : \tau_i \]  

(≤) \[ \Gamma \vdash M : \tau \]  
\[ \tau \leq \sigma \]  
\[ \Gamma \vdash M : \sigma \]  

The system is centrally placed in the theory of typed λ-calculus, see Barendregt, Dekkers, Statman, *Lambda Calculus with Types* [BDS13].
Subtyping (BCD)

Definition

Subtyping $\leq$ is the least preorder (reflexive and transitive relation) over $\mathbb{T}$ (cf. [BCDC83]) such that

- $\sigma \leq \omega, \quad \omega \leq \omega \rightarrow \omega$
- $\sigma \cap \tau \leq \sigma, \quad \sigma \cap \tau \leq \tau$
- $(\sigma \rightarrow \tau_1) \cap (\sigma \rightarrow \tau_2) \leq \sigma \rightarrow \tau_1 \cap \tau_2$
- $\sigma \leq \tau_1 \land \sigma \leq \tau_2 \Rightarrow \sigma \leq \tau_1 \cap \tau_2$
- $\sigma_2 \leq \sigma_1 \land \tau_1 \leq \tau_2 \Rightarrow \sigma_1 \rightarrow \tau_1 \leq \sigma_2 \rightarrow \tau_2$

Write $\sigma = \tau$ for $\sigma \leq \tau \land \tau \leq \sigma$. Then $\cap$ is ACI, and

- $(\sigma \rightarrow \tau_1) \cap (\sigma \rightarrow \tau_2) = \sigma \rightarrow (\tau_1 \cap \tau_2)$
- $(\sigma_1 \rightarrow \tau_1) \cap (\sigma_2 \rightarrow \tau_2) \leq (\sigma_1 \cap \sigma_2) \rightarrow (\tau_1 \cap \tau_2)$
**Problem (Subtyping)**

*Given* \( \sigma, \tau \in \mathbb{T} \), *does* \( \sigma \leq \tau \) *hold?*

**Lemma (Beta-Soundness [BCDC83])**

*Given* \( \sigma = \bigcap_{i \in I} (\sigma_i \rightarrow \tau_i) \cap \bigcap_{j \in J} a_j \cap \bigcap_{k \in K} \alpha_k \), we have

(i) If \( \sigma \leq a \) for some \( a \in \mathbb{C} \), then \( a \equiv a_j \) for some \( j \in J \).

(ii) If \( \sigma \leq \alpha \) for some \( \alpha \in \mathbb{V} \), then \( \alpha \equiv \alpha_k \) for some \( k \in K \).

(iii) If \( \sigma \leq \sigma' \rightarrow \tau' \neq \omega \) for some \( \sigma', \tau' \in \mathbb{T} \), then \( l' = \{ i \in I \mid \sigma' \leq \sigma_i \} \neq \emptyset \) and \( \bigcap_{i \in l'} \tau_i \leq \tau' \).

**Theorem ([DMR17])**

Subtyping is decidable in quadratic time.
Problem (Subtyping)

Given $\sigma, \tau \in \mathbb{T}$, does $\sigma \leq \tau$ hold?

Lemma (Beta-Soundness [BCDC83])

Given $\sigma = \bigcap_{i \in I} (\sigma_i \rightarrow \tau_i) \cap \bigcap_{j \in J} a_j \cap \bigcap_{k \in K} \alpha_k$, we have

(i) If $\sigma \leq a$ for some $a \in \mathbb{C}$, then $a \equiv a_j$ for some $j \in J$.

(ii) If $\sigma \leq \alpha$ for some $\alpha \in \mathbb{V}$, then $\alpha \equiv \alpha_k$ for some $k \in K$.

(iii) If $\sigma \leq \sigma' \rightarrow \tau' \neq \omega$ for some $\sigma', \tau' \in \mathbb{T}$, then $l' = \{i \in l \mid \sigma' \leq \sigma_i\} \neq \emptyset$ and $\bigcap_{i \in l'} \tau_i \leq \tau'$.

Theorem ([DMR17])

Subtyping is decidable in quadratic time.
Problem (Matching)

Given a set of constraints \( C = \{ \sigma_1 \dot{\leq} \tau_1, \ldots, \sigma_n \dot{\leq} \tau_n \} \), where for each \( i \in \{1, \ldots, n\} \) we have \( \text{Var}(\sigma_i) = \emptyset \) or \( \text{Var}(\tau_i) = \emptyset \), is there a substitution \( S: \mathcal{V} \rightarrow \mathcal{T} \) such that \( S(\sigma_i) \leq S(\tau_i) \) for \( 1 \leq i \leq n \)?

We say that a substitution \( S \) satisfies \( \{ \sigma_1 \dot{\leq} \tau_1, \ldots, \sigma_n \dot{\leq} \tau_n \} \) if \( S(\sigma_i) \leq S(\tau_i) \) for \( 1 \leq i \leq n \).

Theorem ([DMR13])

Matching is NP-complete.

Matching remains NP-hard even when restricted to a single type variable and a single type constant in the input [DMR17].
Subtyping (BCD)

Problem (Matching)

Given a set of constraints \( C = \{\sigma_1 \leq \tau_1, \ldots, \sigma_n \leq \tau_n\} \), where for each \( i \in \{1, \ldots, n\} \) we have \( \text{Var}(\sigma_i) = \emptyset \) or \( \text{Var}(\tau_i) = \emptyset \), is there a substitution \( S: \mathbb{V} \rightarrow \mathbb{T} \) such that \( S(\sigma_i) \leq S(\tau_i) \) for \( 1 \leq i \leq n \)?

We say that a substitution \( S \) satisfies \( \{\sigma_1 \leq \tau_1, \ldots, \sigma_n \leq \tau_n\} \) if \( S(\sigma_i) \leq S(\tau_i) \) for \( 1 \leq i \leq n \).

Theorem ([DMR13])

Matching is NP-complete.

Matching remains NP-hard even when restricted to a single type variable and a single type constant in the input [DMR17].
Subtyping (BCD)

Problem (Satisfiability)

Given a set of constraints $C = \{\sigma_1 \vdash \tau_1, \ldots, \sigma_n \vdash \tau_n\}$, is there a substitution $S : \forall \rightarrow T$ such that $S(\sigma_i) \leq S(\tau_i)$ for $1 \leq i \leq n$?

Problem (Algebraic unification)

Given a set of constraints $C = \{\sigma_1 \doteq \tau_1, \ldots, \sigma_n \doteq \tau_n\}$, is there a substitution $S : \forall \rightarrow T$ such that $S(\sigma_i) = S(\tau_i)$ for $1 \leq i \leq n$?

Theorem ([DMR16, DMR17])

The algebraic unification problem is $\text{EXPTIME}$-hard.

Open problem

Is algebraic unification decidable?
Subtyping (BCD)

Problem (Satisfiability)

Given a set of constraints $C = \{\sigma_1 \leq \tau_1, \ldots, \sigma_n \leq \tau_n\}$, is there a substitution $S : \mathbb{V} \rightarrow \mathbb{T}$ such that $S(\sigma_i) \leq S(\tau_i)$ for $1 \leq i \leq n$?

Problem (Algebraic unification)

Given a set of constraints $C = \{\sigma_1 = \tau_1, \ldots, \sigma_n = \tau_n\}$, is there a substitution $S : \mathbb{V} \rightarrow \mathbb{T}$ such that $S(\sigma_i) = S(\tau_i)$ for $1 \leq i \leq n$?

Theorem ([DMR16, DMR17])

The algebraic unification problem is EXPTIME-hard.

Open problem

Is algebraic unification decidable?
Subtyping (BCD)

Problem (Satisfiability)

Given a set of constraints $C = \{ \sigma_1 \leq \tau_1, \ldots, \sigma_n \leq \tau_n \}$, is there a substitution $S : \forall \rightarrow \mathbb{T}$ such that $S(\sigma_i) \leq S(\tau_i)$ for $1 \leq i \leq n$?

Problem (Algebraic unification)

Given a set of constraints $C = \{ \sigma_1 \equiv \tau_1, \ldots, \sigma_n \equiv \tau_n \}$, is there a substitution $S : \forall \rightarrow \mathbb{T}$ such that $S(\sigma_i) = S(\tau_i)$ for $1 \leq i \leq n$?

Theorem ([DMR16, DMR17])

The algebraic unification problem is EXPTIME-hard.

Open problem

Is algebraic unification decidable?
Subtyping (BCD)

An axiomatization of the equational theory of intersection type subtyping (without $\omega$) is derived in [Sta15]. We add two additional axioms (U) and (RE) to incorporate the universal type $\omega$.

**Definition (ACIUD_{ReAb})**

The equational theory $\text{ACIUD}_{\text{ReAb}}$ is given by

(A) $\sigma \cap (\tau \cap \rho) = (\sigma \cap \tau) \cap \rho$

(C) $\sigma \cap \tau = \tau \cap \sigma$

(I) $\sigma \cap \sigma = \sigma$

(U) $\sigma \cap \omega = \sigma$

(D) $(\sigma \rightarrow \tau) \cap (\sigma \rightarrow \tau') = \sigma \rightarrow \tau \cap \tau'$

(RE) $\omega = \omega \rightarrow \omega$

(AB) $\sigma \rightarrow \tau = (\sigma \rightarrow \tau) \cap (\sigma \cap \sigma' \rightarrow \tau)$
Subtyping (BCD)

Writing $\cap$ as $+$ and $\rightarrow$ as $\ast$

**Definition (ACIUD\_ReAB)**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>$\sigma + (\tau + \rho) = (\sigma + \tau) + \rho$</td>
</tr>
<tr>
<td>(C)</td>
<td>$\sigma + \tau = \tau + \sigma$</td>
</tr>
<tr>
<td>(I)</td>
<td>$\sigma + \sigma = \sigma$</td>
</tr>
<tr>
<td>(U)</td>
<td>$\sigma + \omega = \sigma$</td>
</tr>
<tr>
<td>(D_I)</td>
<td>$(\sigma \ast \tau) + (\sigma \ast \tau') = \sigma \ast (\tau + \tau')$</td>
</tr>
<tr>
<td>(RE)</td>
<td>$\omega = \omega \ast \omega$</td>
</tr>
<tr>
<td>(AB)</td>
<td>$\sigma \ast \tau = (\sigma \ast \tau) + ((\sigma + \sigma') \ast \tau)$</td>
</tr>
</tbody>
</table>

Close to Exptime-complete ACID-theory studied in [ANR04, ANR03] ... Yet, due to (AB), probably far from it.
Principality and Unification

1. INTRODUCTION

Basic functionality theory (as in other illative systems) was introduced by Curry (Curry and Feys, 1958) for overcoming the weakness (revealed, for example, by some well-known “paradoxes”) of pure combinators or λ-calculus systems as a support for studying properties of functions and logical reductions.

Let’s recall that, according to (Curry et al., 1958), a formula in a functionality theory is a statement \( \tau \rightarrow x \), where \( x \) is a term and \( \tau \) its functional character or type. So \( \tau \rightarrow x \) means that \( \tau \) is a type for \( x \). Functional characters are built from a set of basic elements (which are left uninterpreted) and a composition operator \( \cdot \). If \( \tau, \tau' \) are types, \( \tau \cdot \tau' \) is the type of a term which defines a function from terms of type \( \tau \) to terms of type \( \tau' \).

As pointed out by Curry (1969) and Hindley (1969), an important concept in functionality theory is the one of principal type scheme of an object. It turns out, in fact, that each typed term has an infinite set of functional characters rather than a single one. The existence of the principal type scheme of a stratified term \( x \) (in the sense that all types of \( x \) can be obtained from one as instances) shows an internal coherence between all functional characters of \( x \).

Curry’s system, however, remains too weak to be fully satisfactory. In fact types are not preserved by convertibility (unless we positulate it explicitly) and it is not possible to assign types to a large class of terms. In

[CDVC80, RDR88, CG95]
On the Power of Subtyping

- Restriction without \((\cap I)\) studied by Kurata & Takahashi, TLCA 95 [KT95].
- Subtyping (distributivity) captures a certain amount of \((\cap I)\):

\[
\{x : (a \to c) \cap (b \to d), y : a \cap b\} \vdash (xy) : c \cap d
\]

Theorem ([RU12])

The inhabitation problem for the system of [KT95] is EXPSPACE-complete with subtyping and PSPACE-complete without subtyping.\(^a\)

\(^a\)But including \((\cap E)\).
Problem (Inhabitation $\Gamma \vdash ? : \tau$)

Given $\Gamma$ and $\tau$, does there exist a term $M$ such that $\Gamma \vdash M : \tau$?
Inhabitation and Synthesis

Problem (Inhabitation $\Gamma \vdash ? : \tau$)

*Given $\Gamma$ and $\tau$, does there exist a term $M$ such that $\Gamma \vdash M : \tau$?*
Inhabitation and Synthesis

Component-oriented Synthesis

Synthesis *relative to library* (repository) of components

Combinatory Logic Synthesis (CLS)

Libraries need *classification systems* to enable *retrieval and composition*

Bottom-up specification

Hoare logic
Über die **Bausteine** der mathematischen Logik.

Von

M. Schönfinkel in Moskau 1).

§ 1.

Es entspricht dem Wesen der axiomatic method, wie sie heute vor allem durch die Arbeiten Hilberts zur Anerkennung gelangt ist, daß man nicht allein hinsichtlich der Zahl und des Gehalts der **Axiome** nach möglichster Beschränkung strebt, sondern auch die Anzahl der als un- definiert zugrunde zu legenden **Begriffe** so klein wie möglich zu machen sucht, indem man nach Begriffen fahndet, die vorzugsweise geeignet sind, um aus ihnen alle anderen Begriffe des fraglichen Wissenszweiges auf- zubauen. Begreiflicherweise wird man sich im Sinne dieser Aufgabe be- züglich des Verlangens nach Einfachheit der an den Anfang zu stellenden Begriffe entsprechend bescheiden müssen.

Bekanntlich lassen sich die grundlegenden **Aussagenverknüpfungen** der mathematischen Logik, die ich hier in der von Hilbert in seinen Vor- lesungen verwendeten Bezeichnungsweise wiedergebe:

\[ \bar{a}, \quad a \lor b, \quad a \land b, \quad a \rightarrow b, \quad a \sim b \]


Mathematische Annalen, 92.
- CL over *arbitrary bases*:
  
  General theory of component collections (repositories)

- The implementation theory” \( \{S, K, I\} \) is one, very special case
Can we use *inhabitation in combinatory logic with intersection types* as a foundation for component-oriented, type-based synthesis?

- Typed combinators $X : \tau$ as named interfaces
- Automated composition synthesis via inhabitation
- Intersection types as *semantic types* (cf. also Haack, Wells, Yakobowski et al. [HHSW02, WY05]) for specification
- Beyond purely functional composition via meta-programming – compose a meta-program which, when executed, computes (say) a Java program
\[ \Gamma \vdash_{\text{CL}} e : \tau \]

- Combinatory basis
- Component repository
  - Inhabitant
  - Synthesized meta-program
- Intersection type
- Specification (synthesis goal)
Relativized Inhabitation

- We consider the *relativized inhabitation* problem:
  - **Given a set of typed combinators** $\Gamma$ and $\tau$, does there exist combinatory expression $e$ such that $\Gamma \vdash e : \tau$?

- Inhabitation for fixed base $\{S, K, I\}$ is $\text{PSPACE}$-complete in simple types (Statman’s Theorem [Sta79])

- Relativized inhabitation is much harder
  - Undecidable in simple types: Linial-Post theorems, 1948ff. [LP49]\(^1\)

- The CLS view: Already in simple types, relativized inhabitation defines a Turing-complete logic programming language for component composition
  - Reduction from 2-counter automata [Reh13]
  - Similar idea used to prove undecidability for synthesis in ML relative to library of functions [BSWC16]

---

\(^1\) See also A. Dudenhefner, JR: *Lower End of the Linial-Post Spectrum*, TYPES 2017
We consider the relativized inhabitation problem:

- Given a set of typed combinators $\Gamma$ and $\tau$, does there exist combinatory expression $e$ such that $\Gamma \vdash e : \tau$?

Inhabitation for fixed base $\{S, K, I\}$ is $PSPACE$-complete in simple types (Statman’s Theorem [Sta79])

Relativized inhabitation is much harder

- Undecidable in simple types: Linial-Post theorems, 1948ff. [LP49]¹

The CLS view: Already in simple types, relativized inhabitation defines a Turing-complete logic programming language for component composition

- Reduction from 2-counter automata [Reh13]
- Similar idea used to prove undecidability for synthesis in ML relative to library of functions [BSWC16]

¹ See also A. Dudenhefner, JR: Lower End of the Linial-Post Spectrum, TYPES 2017
Relativized Inhabitation

- We consider the *relativized inhabitation* problem:
  - **Given a set of typed combinators** $\Gamma$ and $\tau$, does there exist combinatory expression $e$ such that $\Gamma \vdash e : \tau$?

- Inhabitation for fixed base $\{S, K, I\}$ is $\text{PSPACE}$-complete in simple types (Statman’s Theorem [Sta79])

- Relativized inhabitation is much harder
  - *Undecidable in simple types*: Linial-Post theorems, 1948ff. [LP49]\(^1\)
  - *The CLS view*: Already in simple types, relativized inhabitation defines a Turing-complete logic programming language for component composition
    - Reduction from 2-counter automata [Reh13]
    - Similar idea used to prove undecidability for synthesis in ML relative to library of functions [BSWC16]

\(^1\) See also A. Dudenhefner, JR: *Lower End of the Linial-Post Spectrum*, TYPES 2017
We consider the *relativized inhabitation* problem:

Given a set of typed combinators $\Gamma$ and $\tau$, does there exist combinatory expression $e$ such that $\Gamma \vdash e : \tau$?

Inhabitation for fixed base $\{S, K, I\}$ is $\text{PSPACE}$-complete in simple types (Statman’s Theorem [Sta79])

Relativized inhabitation is much harder

Undecidable in simple types: Linial-Post theorems, 1948ff. [LP49]\(^1\)

The CLS view: Already in simple types, relativized inhabitation defines a Turing-complete logic programming language for component composition

- Reduction from 2-counter automata [Reh13]
- Similar idea used to prove undecidability for synthesis in ML relative to library of functions [BSWC16]

---

\(^1\) See also A. Dudenhefner, JR: *Lower End of the Linial-Post Spectrum*, TYPES 2017
Combinatory Logic with Intersection Types $\mathcal{CL}(\rightarrow, \cap)$

Definition

\[
\begin{align*}
\Gamma, X : \tau & \vdash X : S(\tau) \quad \text{(var)} \\
\Gamma, e : \tau & \rightarrow \sigma, \Gamma, e' : \tau & \vdash (e \ e') : \sigma \quad \text{(→E)} \\
\Gamma, e : \tau & \vdash \Gamma, e : \sigma \quad \text{(∩I)} \\
\Gamma, e : \tau & \vdash \tau \leq \sigma, \Gamma, e : \tau & \vdash e : \sigma \quad \text{(≤)}
\end{align*}
\]

- The SKI-calculus has been studied with intersection types (Dezani and Hindley [DCH92])

Note

But, in CLS, the combinatory theory $\Gamma$ represents an arbitrary repository (basis not fixed)
**Definition (Levels)**

\[
\ell(a) = 0, \text{ for } a \in A;
\]

\[
\ell(\tau \to \sigma) = 1 + \max\{\ell(\tau), \ell(\sigma)\};
\]

\[
\ell(\bigcap_{i=1}^{n} \tau_i) = \max\{\ell(\tau_i) \mid i = 1, \ldots, n\}.
\]

\[
\ell(S) = \max\{\ell(S(\alpha)) \mid S(\alpha) \neq \alpha\}
\]

**Definition (\(\mathbf{bcl}_k(\to, \cap)\), \(k \geq 0\))**

\[
\frac{[\ell(S) \leq k]}{\Gamma, X : \tau \vdash_k X : S(\tau)} \quad \frac{\Gamma \vdash_k e : \tau \to \sigma \quad \Gamma \vdash_k e' : \tau}{\Gamma \vdash_k (e \; e') : \sigma} \quad (\to \text{E})
\]

\[
\frac{\Gamma \vdash_k e : \tau \quad \Gamma \vdash_k e : \sigma}{\Gamma \vdash_k e : \tau \cap \sigma} \quad (\cap \text{I})
\]

\[
\frac{\Gamma \vdash_k e : \tau \quad \Gamma \vdash_k e : \sigma \quad \tau \leq \sigma}{\Gamma \vdash_k e : \sigma} \quad (\leq)
\]

- \(\mathbf{bcl}_k\): Bounded Combinatory Logic, CSL 2012 [DMRU12]
- FCL: Finite Combinatory Logic with Intersection Types, TLCA 2011 [RU11], taking \(S = id\).
Bounded Combinatory Logic $\text{bcl}_k(\to, \cap)$

**Definition (Levels)**

\[
\begin{align*}
\ell(a) &= 0, \text{ for } a \in A; \\
\ell(\tau \to \sigma) &= 1 + \max\{\ell(\tau), \ell(\sigma)\}; \\
\ell(\bigcap_{i=1}^n \tau_i) &= \max\{\ell(\tau_i) \mid i = 1, \ldots, n\}. \\
\ell(S) &= \max\{\ell(S(\alpha)) \mid S(\alpha) \neq \alpha\}
\end{align*}
\]

**Definition ($\text{bcl}_k(\to, \cap), k \geq 0$)**

\[
\begin{align*}
\frac{[\ell(S) \leq k]}{\Gamma, \, X : \tau \vdash_k X : S(\tau)} \quad (\text{var}) & \quad \frac{\Gamma \vdash_k e : \tau \to \sigma \quad \Gamma \vdash_k e' : \tau}{\Gamma \vdash_k (e \, e') : \sigma} & \quad (\to E) \\
\frac{\Gamma \vdash_k e : \tau \quad \Gamma \vdash_k e : \sigma}{\Gamma \vdash_k e : \tau \cap \sigma} \quad (\cap I) & \quad \frac{\Gamma \vdash_k e : \tau \quad \tau \leq \sigma}{\Gamma \vdash_k e : \sigma} \quad (\leq)
\end{align*}
\]

- $\text{BCL}_k$: *Bounded Combinatory Logic*, CSL 2012 [DMRU12]
- FCL: *Finite Combinatory Logic with Intersection Types*, TLCA 2011 [RU11], taking $S = \text{id}$. 

---

25 / 55
Complexity for Finite and Bounded CL

Theorem (TLCA 2011 [RU11])

For finite combinatory logic $\text{FCL}$:

1. Relativized inhabitation in $\text{FCL}(\to)$ is in $\text{PTime}$
2. Relativized inhabitation in $\text{FCL}(\to, \cap)$ is $\text{EXPTIME}$-complete

Theorem (CSL 2012 [DMRU12])

For bounded combinatory logic $\text{BCL}_k$:

1. Relativized inhabitation in $\text{BCL}_k(\to)$ is $\text{EXPTIME}$-complete for all $k$
2. Relativized inhabitation in $\text{BCL}_k(\to, \cap)$ is $(k + 2)$-$\text{EXPTIME}$-complete
Upper Bound ATM for $\text{bcl}_k(\to, \cap)$: $\text{ASPACE}(\exp_{k+1}(n))$

**Input**: $\Gamma, \tau, k$

\[
\Gamma = \{f : (0 \to 1) \cap (1 \to 0),
\quad x : (\alpha \to \beta) \to (\beta \to \gamma) \to (\alpha \to \gamma)\}
\]

\[
\tau = (0 \to 0) \cap (1 \to 1)
\]

**loop**:

1. $\text{CHOOSE } (x : \sigma) \in \Gamma$;
2. $\sigma' := \bigcap \{S(\sigma) \mid S \in S_{x}(\Gamma, \tau, k)\}$;
3. $\text{CHOOSE } m \in \{0, \ldots, ||\sigma'||\}$;
4. $\text{CHOOSE } P \subseteq \mathbb{P}_m(\sigma')$;
5. **IF** $(\bigcap_{\pi \in P} \text{tgt}_m(\pi) \leq \tau)$ **THEN**
   - **IF** $(m = 0)$ **THEN** ACCEPT;
   - **ELSE**
     - $\text{FORALL } (i = 1 \ldots m)$
     - $\tau := \bigcap_{\pi \in P} \text{arg}_i(\pi)$;
     - $\tau := (0 \to 1) \cap (1 \to 0)$
     - $\tau := (1 \to 0) \cap (0 \to 1)$
5. **GOTO** loop;
6. **ELSE** REJECT;

\[
(x \ f) : (0 \to 0) \cap (1 \to 1)
\]
Upper Bound ATM for $\text{bcl}_k(\rightarrow, \cap): \text{ASPACE}(\exp_{k+1}(n))$

**Input**: $\Gamma, \tau, k$

$\Gamma = \{f : (0 \rightarrow 1) \cap (1 \rightarrow 0),
\quad x : (\alpha \rightarrow \beta) \rightarrow (\beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \gamma)\}$

$\tau = (0 \rightarrow 0) \cap (1 \rightarrow 1)$

**loop**:

1. **choose** $(x : \sigma) \in \Gamma$;
2. $\sigma' := \bigcap \{S(\sigma) | S \in S^\Gamma_{X, \tau, k}\}$;
3. **choose** $m \in \{0, \ldots, ||\sigma'||\}$;
4. **choose** $P \subseteq \mathcal{P}_m(\sigma')$;
5. **if** $(\bigcap_{\pi \in P} \text{tgt}_m(\pi) \leq \tau)$ **then**
   6. **if** $(m = 0)$ **then** accept;
   7. **else**
5. **forall** $(i = 1 \ldots m)$
   9. $\tau := \bigcap_{\pi \in P} \text{arg}_i(\pi)$;
   10. **goto** loop;
11. **else** reject;

$(x f) f : (0 \rightarrow 0) \cap (1 \rightarrow 1)$
Upper Bound ATM for $\text{bcl}_k(\rightarrow, \cap): \text{ASPACE}(\exp_{k+1}(n))$

**Input**: $\Gamma, \tau, k$

$\Gamma = \{f : (0 \rightarrow 1) \cap (1 \rightarrow 0),\ x : (\alpha \rightarrow \beta) \rightarrow (\beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \gamma)\}$

$\tau = (0 \rightarrow 0) \cap (1 \rightarrow 1)$

**loop**:

1. \textbf{CHOOSE} $(x : \sigma) \in \Gamma$;
2. $\sigma' := \bigcap \{S(\sigma) \mid S \in S^R(\Gamma, \tau, k)\}$;
3. \textbf{CHOOSE} $m \in \{0, \ldots, ||\sigma'||\}$;
4. \textbf{CHOOSE} $P \subseteq \mathbb{P}_m(\sigma')$;

5. \textbf{IF} $(\bigcap_{\pi \in P} \text{tgt}_m(\pi) \leq \tau)$ \textbf{THEN} $(0 \rightarrow 0)(1 \rightarrow 0)(0 \rightarrow 0)\cap (1 \rightarrow 1) \leq \tau$

6. \textbf{IF} $(m = 0)$ \textbf{THEN} ACCEPT;

7. ELSE

8. \textbf{FORALL} $(i = 1 \ldots m)$
9. $\tau := \bigcap_{\pi \in P} \text{arg}_i(\pi)$;
10. $\tau := (0 \rightarrow 1)(1 \rightarrow 0)$

11. \textbf{ELSE} REJECT;

12. \textbf{GOTO} \textit{loop};

13. $(x f) f : (0 \rightarrow 0) \cap (1 \rightarrow 1)$
Upper Bound ATM for $\text{bcl}_k(\rightarrow, \cap): \text{ASPACE}(\exp_{k+1}(n))$

Input: $\Gamma, \tau, k$

$\Gamma = \{f : (0 \rightarrow 1) \cap (1 \rightarrow 0),$ 
$\quad x : (\alpha \rightarrow \beta) \rightarrow (\beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \gamma)\}$

$\tau = (0 \rightarrow 0) \cap (1 \rightarrow 1)$

loop:

1. choose $(x : \sigma) \in \Gamma$;

2. $\sigma' := \bigcap\{S(\sigma) \mid S \in S_X^{(\Gamma, \tau, k)}\}$;

3. choose $m \in \{0, \ldots, ||\sigma'||\}$;

4. choose $P \subseteq \mathbb{P}_m(\sigma')$;

5. IF $(\bigcap_{\pi \in P} \text{tgt}_m(\pi) \leq \tau)$ THEN

6. IF ($m = 0$) THEN ACCEPT;

7. ELSE

8. forall ($i = 1 \ldots m$)

9. $\tau := \bigcap_{\pi \in P} \text{arg}_i(\pi)$;

10. GOTO loop;

11. ELSE REJECT;

$(x f) f : (0 \rightarrow 0) \cap (1 \rightarrow 1)$
Upper Bound ATM for $bcl_k(\rightarrow, \cap): \text{ASPACE}(\exp_{k+1}(n))$

*Input:* $\Gamma, \tau, k$

$\Gamma = \{f : (0 \rightarrow 1) \cap (1 \rightarrow 0),$

$$x : (\alpha \rightarrow \beta) \rightarrow (\beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \gamma)\}$$

$\tau = (0 \rightarrow 0) \cap (1 \rightarrow 1)$

*loop:*

1. **choose** $(x : \sigma) \in \Gamma;$

2. $\sigma' := \bigcap \{S(\sigma) \mid S \in S^f(\Gamma, \tau, k)\};$

3. **choose** $m \in \{0, \ldots, ||\sigma'||\};$

4. **choose** $P \subseteq \mathbb{P}_m(\sigma');$

5. **if** $(\bigcap_{\pi \in P} \text{tgt}_m(\pi) \leq \tau)$ **then** $(0 \rightarrow 0)(1 \rightarrow 0)(0 \rightarrow 0) \cap$

6. **if** $(m = 0)$ **then** accept;

7. **else**

8. **forall** $(i = 1 \ldots m)$

9. $\tau := \bigcap_{\pi \in P} \text{arg}_i(\pi);$

10. **goto** *loop*;

11. **else** reject;

$(x f) f : (0 \rightarrow 0) \cap (1 \rightarrow 1)$
Upper Bound ATM for \( bcl_k(\rightarrow, \cap) : ASPACE(\exp_{k+1}(n)) \)

Input : \( \Gamma, \tau, k \)

\[ \Gamma = \{ f : (0 \to 1) \cap (1 \to 0), \]
\[ x : (\alpha \to \beta) \to (\beta \to \gamma) \to (\alpha \to \gamma) \} \]
\[ \tau = (0 \to 0) \cap (1 \to 1) \]

\( \text{loop :} \)

1. \text{choose} \( (x : \sigma) \in \Gamma; \)
2. \( \sigma' := \bigcap \{ S(\sigma) \mid S \in S_{\alpha, \beta, \gamma}^{(\Gamma, \tau, k)} \}; \)
3. \text{choose} \( m \in \{0, \ldots, ||\sigma'||\}; \)
4. \text{choose} \( P \subseteq P_m(\sigma'); \)
5. \text{if} \( \bigcap_{\pi \in P} tgt_m(\pi) \leq \tau \) \text{then} \( (0 \to 0) \cap (1 \to 1) \leq \tau \)
   \[ \text{if} \ (m = 0) \ \text{then accept;} \]
6. \( \text{else} \)
7. \( \forall i = 1 \ldots m \)
8. \( \tau := \bigcap_{\pi \in P} arg_i(\pi); \)
9. \( \tau := (0 \to 1) \cap (1 \to 0) \)
10. \( \tau := (1 \to 0) \cap (0 \to 1) \)
11. \( \text{goto } \text{loop;} \)
12. \( \text{else reject;} \)

\( (x f) f : (0 \to 0) \cap (1 \to 1) \)

**Algorithm 4.5: ATM with lookahead-test**

\[
\begin{align*}
\text{Input} & : \Gamma, \tau \quad \text{all types in } \Gamma \text{ and } \tau = \bigcap_{i \in I} \tau_i \text{ organized} \\
\text{loop} : & \\
1 & \text{CHOOSE } (x : \sigma) \in \Gamma; \\
2 & \text{write } \sigma \equiv \bigcap_{j \in J} \sigma_j \\
3 & \text{FOR EACH } i \in I, j \in J, m \leq \|\sigma\| \text{ DO} \\
4 & \quad \text{candidates}(i, j, m) := \text{Match}(t_{g_t}(\sigma_j) \leq \tau_i) \\
5 & \quad M := \{m \leq \|\sigma\| \mid \forall i \in I \exists j \in J : \text{candidates}(i, j, m) = \text{true}\} \\
6 & \text{CHOOSE } m \in M; \\
7 & \text{FOR EACH } i \in I \text{ DO} \\
8 & \quad \text{CHOOSE } \sigma' \in J \text{ with candidates}(i, \sigma', m) = \text{true} \\
9 & \quad \text{CHOOSE } S_i \text{ a substitution} \\
10 & \quad \text{CHOOSE } \pi_i \in P_m(S_i(\sigma_{j_i})) \text{ with } t_{g_t}(\sigma_i) \leq \tau_i \text{ and} \\
11 & \quad \forall 1 \leq l \leq m \forall \pi' \in \text{arg}_{l}(\pi_i) \exists (y : \rho') \in \Gamma \exists \text{ a path } \rho' \\
12 & \quad \text{in } \rho \exists k : \text{Match}(t_{g_t}(\rho') \leq \pi') = \text{true} \\
13 & \quad \text{IF } (m = 0) \text{ THEN ACCEPT;} \\
14 & \quad \text{ELSE FORALL}(l = 1 \ldots m) \\
15 & \quad \quad \tau := \bigcap_{i \in I} \text{arg}_{l}(\pi_i); \\
16 & \quad \quad \text{GOTO loop;} \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>$n$</th>
<th>$m$</th>
<th>#I</th>
<th>RT/ms</th>
<th>#I</th>
<th>RT/ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>73</td>
<td>112.2</td>
<td>9</td>
<td>109.8</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>973</td>
<td>209.6</td>
<td>49</td>
<td>111.4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>11665</td>
<td>2245.2</td>
<td>257</td>
<td>124.4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>43905</td>
<td>12504</td>
<td>55</td>
<td>124</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>$1.4 \times 10^{10}$</td>
<td>-</td>
<td>2188</td>
<td>354</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>$4.8 \times 10^{12}$</td>
<td>-</td>
<td>78733</td>
<td>50959.6</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>$1.3 \times 10^{14}$</td>
<td>-</td>
<td>33</td>
<td>142.4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>$6.6 \times 10^{18}$</td>
<td>-</td>
<td>3889</td>
<td>1314.6</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>$3.3 \times 10^{23}$</td>
<td>-</td>
<td>640001</td>
<td>$7.5 \times 10^6$</td>
</tr>
</tbody>
</table>

Table 4.1: Experimental Data for $\Gamma^n$
Refinement (after [FP91])

Definition ([SMGB12])

Let $T_o$ be simple types over an atom $o$. Fix $X \subseteq A$ and define uniform types $U_X(\tau)$ for $\tau \in T_o$:

\[
U_X(o) = X \\
U_X(\tau \rightarrow \sigma) = (U_X(\tau) \Rightarrow U_X(\sigma))^\cap
\]

With such types we can represent any finite function $f : A \rightarrow B$ at the type level by $\bigcap_{a \in A} (a \rightarrow f(a))$.

We can express finite abstract interpretations, e.g.,

\[
\text{succ} : (\text{Nat} \rightarrow \text{Nat}) \cap (\text{zero} \rightarrow \text{pos}) \cap (\text{pos} \rightarrow \text{pos}) \cap (\text{even} \rightarrow \text{odd}) \cap (\text{odd} \rightarrow \text{even})
\]

Inhabitation ($\lambda$-calculus) is undecidable. Proof: Note that [SMGB12] uses only uniform types for $\lambda$-definability. □
**CL(→, ∩) over Uniform (Refinement) Types**

**Definition**

Let $\mathbb{T}_o$ be simple types over an atom $o$. Fix $X \subseteq A$ and define *uniform types* $U_X(\tau)$ for $\tau \in \mathbb{T}_o$:

\[
U_X(o) = X
\]
\[
U_X(\tau \rightarrow \sigma) = (U_X(\tau) \Rightarrow U_X(\sigma))^\cap
\]

**Corollary**

Relativized inhabitation with uniform types is nonelementary recursive.

**Proof.**

Upper bound: every problem $\Gamma \vdash ? : \sigma$ is decidable within $\text{bcl}_k(\rightarrow, \cap)$ with $k = \max\{\ell(\tau) \mid \tau \in \text{rn}(\Gamma)\}$.

Lower bound: notice that all constructions in l.b. for $\text{bcl}_k(\rightarrow, \cap)$ can be carried out with uniform types. □
Corollary: Henkin’s theory $\Omega$ in $\mathcal{BCL}_k(\to, \cap)$

Satisfiability of formulae

$$\Phi ::= 0 \in x^1 \mid 1 \in x^1 \mid x^k \in y^{k+1} \mid \neg \Phi \mid \forall x^k. \Phi \mid \Phi \land \Phi'$$

where $x^k$ ranges over $D_k$ with $D_0 = \{0, 1\}$, $D_{k+1} = \mathcal{P}(D_k)$.


Representation in $\mathcal{BCL}_k(\to, \cap)$ (for sufficiently large $k$):

- A set variable $x^k$ is represented by a type variable $x^k$.
- Membership predicate $\text{Mem}_k$

$$\text{Num}_k(x^k) \to \text{Num}_{k+1}(y^{k+1}) \to \text{In}_k(x^k, y^{k+1}) \to \text{Mem}_k(x^k, y^{k+1})$$

where $\text{In}_k(x^k, x^k \to 1)$ and $\text{NotIn}(x^k, x^k \to 0)$ are axioms.

- Use alternation to code quantifiers as usual (Urzyczyn 1997).
Scala-integrated framework and experiments by Bessai (Dortmund), Düdder (Copenhagen), Dudenhofner (Dortmund) in collaboration with Chen (formerly Torino), De’Liguoro (Torino), Heineman (Boston), Martens (formerly Dortmund), Urzyczyn (Warsaw) [Reh13] [DGM+12] [DMR13] [BDD+14] [DMR14] [BDD+15] [DRH15] [HHDR15] [BDHR16] [HBDR16a] [BDD+16a]
CLS Framework

\[ \Gamma = \{ \text{customerForm} : (\text{String} \rightarrow \text{java.net.URL} \rightarrow \text{OptionSelection} \rightarrow \text{Form}) \land \\
\text{dropDownSelector} : (\text{java.net.URL} \rightarrow \text{OptionSelection}) \land \\
\text{radioButtonSelector} : (\text{java.net.URL} \rightarrow \text{OptionSelection}) \land \\
\text{companyTitle} : \text{String} \land \\
\text{databaseLocation} : \text{java.net.URL} \land \\
\text{logoLocation} : \text{java.net.URL} \land \\
\text{alternateLogoLocation} : \text{java.net.URL} \} \]

\[ \Gamma \vdash \text{CompilationUnit} \land \text{OrderMenu}(\omega) \]

Inhabitants:

\{ \text{customerForm} (\text{companyTitle}, \text{logoLocation}), \\
\text{radioButtonSelector} (\text{databaseLocation}), \\
\text{customerForm} (\text{companyTitle}, \text{alternateLogoLocation}, \\
\text{radioButtonSelector} (\text{databaseLocation})), \\
\text{customerForm} (\text{companyTitle}, \text{logoLocation}, \\
\text{dropDownSelector} (\text{databaseLocation})), \\
\text{customerForm} (\text{companyTitle}, \text{alternateLogoLocation}, \\
\text{dropDownSelector} (\text{databaseLocation})) \}

Interpreted as calls to combinators and resulting code is made available via branches in source code versioning system (git).
CLS Framework – Experiments

ArchiType [Düd14], Combinatory Process Synthesis [BDD+ 16b], LaunchPad (Feature-Oriented Synthesis) [HBDR16b].

1. setSensors': □(subproc ∩ twoLightSensors ∩ stopsOnTouch ∩ set)
   a subprocess to set the light and touch sensors.
2. executeJob': □(subproc ∩
   twoLightSensors ∩ stopsOnTouch ∩ car ∩ followsLine) ∩ jobProc
   a subprocess that makes the robot execute the job until aborted.
3. stop': □(subproc ∩ car ∩ stop)
   a subprocess that makes a car robot stop.

createRobotProgram(setSensors', executeJob', stop'): robotProgram
∩ □(proc ∩ twoLightSensors ∩ stopsOnTouch ∩ car ∩ followsLine)

Inhabitation: Given the repository $D$ of typed PCs and typed CCs, asking the inhabitation question $D ⊨ □(proc ∩ car ∩ followsLine ∩ twoLightSensors ∩ stopsOnTouch) ∩ robotProgram$ automatically synthesizes the following applicative term:

createRobotProgram(setSensors(box setTwoLightsSensors, taskToSubProc(box setTouchSensor)),
executeJob(taskToSubProc(box readTouchSensor),
box abortCondition(box readTwoLightsSensor, box readTwoLightsSensor,
conditionalMove(box turnLeftCondition(box turnLeftCondition(box stopCar)),
    box turnRightCondition(box turnRightCondition(box turnLeftCar, box moveForwardCar)),
box stopCar)).}
**CLS – Research Challenges**

**INTERSECTION TYPE SPECIFICATION**
\[ \text{fstproc} \cap \text{car} \cap \text{followsLine} \cap \text{twoLightSensors} \cap \text{stopsOnTouch} \cap \text{robotProgram} \]

**Component Repository in SCALA extension**

**Inhabitation algorithm for CL**

**Combinatory Meta-Program**

**Output Program**

**Execution of Meta-Program**
CLS – Research Challenges

- Larger-scale experiments
- Model theory of semantic types
- Generate-Test and Learning
  - The idea of using intersection types as foundation for type-based synthesis also taken up for $\lambda$-calculus inhabitation: Frankle, Osera, Walker, and Zdancewic, *Example-directed synthesis: a type-theoretic interpretation*, POPL 2016 [FOWZ16]
- Integration with theorem proving
Inhabitation in \(\lambda\)-Calculus with Intersection Types

Inhabitation in \(\lambda\)-calculus with intersection types is undecidable


Rank 2-inhabitation is decidable and \(\text{ExpSPACE}\)-complete, and rank \(k\)-inhabitation is undecidable for all ranks \(k > 2\)


- Proof techniques via *bus machines*, an alternating, expanding instruction device, also used to show \(\text{ExpSPACE}\)-completeness of inhabitation with explicit intersection [RU12]. Direct TM-reduction: TYPES 2016, Rank 3 *Inhabitation of Intersection Types Revisited* [BDDR16] and extended version at arXiv.

Related to the \(\lambda\)-definability problem

- Undecidability of \(\lambda\)-definability: Loader 1993 [Loa01]


- Salvati, Manzonetto, Gehrke, Barendregt, Urzyczyn and Loader are logically related, ICALP 2012 [SMGB12]
Inhabitation in $\lambda$-Calculus with Intersection Types

Inhabitation in $\lambda$-calculus with intersection types is undecidable


Rank 2-inhabitation is decidable and ExpSpace-complete, and rank $k$-inhabitation is undecidable for all ranks $k > 2$


- Proof techniques via *bus machines*, an alternating, expanding instruction device, also used to show ExpSpace-completeness of inhabitation with explicit intersection [RU12]. Direct TM-reduction: TYPES 2016, *Rank 3 Inhabitation of Intersection Types Revisited* [BDDR16] and extended version at arXiv.

Related to the $\lambda$-definability problem

- Undecidability of $\lambda$-definability: Loader 1993 [Loa01]


- Salvati, Manzonetto, Gehrke, Barendregt, Urzyczyn and Loader are logically related, ICALP 2012 [SMGB12]
Inhabitation in $\lambda$-Calculus with Intersection Types

Inhabitation in $\lambda$-calculus with intersection types is undecidable


Rank 2-inhabitation is decidable and $\text{ExpSPACE}$-complete, and rank $k$-inhabitation is undecidable for all ranks $k > 2$

- Proof techniques via *bus machines*, an alternating, expanding instruction device, also used to show $\text{ExpSPACE}$-completeness of inhabitation with explicit intersection [RU12]. Direct TM-reduction: TYPES 2016, Rank 3 *Inhabitation of Intersection Types Revisited* [BDDR16] and extended version at arXiv.

Related to the $\lambda$-definability problem

- Undecidability of $\lambda$-definability: Loader 1993 [Loa01]
- Salvati, Manzonetto, Gehrke, Barendregt, *Urzyczyn and Loader are logically related*, ICALP 2012 [SMGB12]
Dimensional $\lambda$-Calculus

- Intersection Type Calculi of Bounded Dimension, POPL 2017 [DR17a].
- Typability in Bounded Dimension, LICS 2017 [DR17b].
Strict Intersection Type System

Definition (Strict Intersection Types)

\[ A, B ::= a \mid \sigma \rightarrow A \]
\[ \sigma, \tau ::= [A_1, \ldots, A_n] \quad n \geq 1 \]

Definition (Strict Type Assignment [vB11](Def. 5.1))

\[ \frac{1 \leq i \leq n}{\Gamma, x : [A_1, \ldots, A_n] \vdash_s x : [A_i]} \] (Var)
\[ \frac{\Gamma \vdash_s M : [A_i] \text{ for } i = 1 \ldots n}{\Gamma \vdash_s M : [A_1, \ldots, A_n]} \] (∩l)
\[ \frac{\Gamma \vdash_s M : [\sigma \rightarrow A] \quad \Gamma \vdash_s N : \sigma}{\Gamma \vdash_s M N : [A]} \] (→E)
\[ \frac{\Gamma, x : \sigma \vdash_s M : [A]}{\Gamma \vdash_s \lambda x. M : [\sigma \rightarrow A]} \] (→I)
### Definition \((\Gamma \vdash P : \sigma)\)

\[
\frac{1 \leq i \leq n}{\Gamma, x : [A_1, \ldots, A_n] \vdash x\langle [A_i] \rangle : [A_i]} \quad \text{(Var)}
\]

\[
\frac{\Gamma, x : \sigma \vdash P : [A]}{\Gamma \vdash (\lambda x. P)\langle [\sigma \rightarrow A] \rangle : [\sigma \rightarrow A]} \quad \text{\((\rightarrow I)\)}
\]

\[
\frac{\Gamma \vdash P : [\sigma \rightarrow A] \quad \Gamma \vdash Q : \sigma}{\Gamma \vdash (P \; Q)\langle [A] \rangle : [A]} \quad \text{\((\rightarrow E)\)}
\]

\[
\frac{\Gamma \vdash P_i : [A_i] \text{ for } i = 1 \ldots n}{\Gamma \vdash \bigsqcup_{i=1}^{n} P_i : [A_1, \ldots, A_n]} \quad \text{\((\cap I)\)}
\]

### Intuition

The operation \(\bigsqcup_{i=1}^{n} P_i\) allows us to measure usage of \((\cap I)\) as a logical resource under norm \(\|\|\).
### Definition $(\Gamma \vdash P : \sigma)$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Expression</th>
</tr>
</thead>
</table>
| $(\text{Var})$ | $1 \leq i \leq n$  
\[ \frac{\Gamma, x : [A_1, \ldots, A_n] \vdash x \langle [A_i] \rangle : [A_i]}{} \] |
| $(\rightarrow I)$ | $\Gamma, x : \sigma \vdash P : [A]$  
\[ \frac{\Gamma \vdash (\lambda x. P) \langle [\sigma \rightarrow A] \rangle : [\sigma \rightarrow A]}{} \] |
| $(\rightarrow E)$ | $\Gamma \vdash P : [\sigma \rightarrow A]$  
\[ \frac{\Gamma \vdash Q : \sigma}{\Gamma \vdash (P \ Q) \langle [A] \rangle : [A]} \] |
| $(\cap I)$ | $\Gamma \vdash P_i : [A_i]$ for $i = 1 \ldots n$  
\[ \frac{\Gamma \vdash \bigsqcup_{i=1}^{n} P_i : [A_1, \ldots, A_n]}{} \] |

### Intuition

The operation $\bigsqcup_{i=1}^{n} P_i$ allows us to measure usage of $(\cap I)$ as a logical resource under norm $\| \bullet \|$.
### Norm

#### Definition (P △ Q, defined for \([P] \equiv [Q]\))

\[
x\langle S \rangle \sqcup x\langle S' \rangle \equiv x\langle S \cup S' \rangle \\
(\lambda x. P)\langle S \rangle \sqcup (\lambda x. Q)\langle S' \rangle \equiv (\lambda x. P \sqcup Q)\langle S \cup S' \rangle \\
(PQ)\langle S \rangle \sqcup (P'Q')\langle S' \rangle \equiv ((P \sqcup P')(Q \sqcup Q'))\langle S \cup S' \rangle
\]

#### Definition (Norm \(\|\cdot\|\))

\[
\|x\langle S \rangle\| = |S| \\
\|(\lambda x. P)\langle S \rangle\| = \max\{|P|, |S|\} \\
\|(PQ)\langle S \rangle\| = \max\{|P|, |Q|, |S|\}
\]

- **Non-negativity**: \(\|P\| > 0\)
- **Subadditivity**: \(\|P \sqcup Q\| \leq \|P\| + \|Q\|\) for \([P] \equiv [Q]\)
**Definition (P ⊔ Q, defined for [P] ≡ [Q])**

\[ x\langle S \rangle ⊔ x\langle S' \rangle ≡ x\langle S \cup S' \rangle \]

\[ (\lambda x. P)\langle S \rangle ⊔ (\lambda x. Q)\langle S' \rangle ≡ (\lambda x. P ⊔ Q)\langle S \cup S' \rangle \]

\[ (PQ)\langle S \rangle ⊔ (P'Q')\langle S' \rangle ≡ ((P ⊔ P')(Q ⊔ Q'))\langle S \cup S' \rangle \]

**Definition (Norm ||•||)**

\[ ||x\langle S \rangle|| = |S| \]

\[ ||(\lambda x. P)\langle S \rangle|| = \max\{||P||, |S|\} \]

\[ ||(PQ)\langle S \rangle|| = \max\{||P||, ||Q||, |S|\} \]

Non-negativity :  ||P|| > 0
Subadditivity :  ||P ⊔ Q|| ≤ ||P|| + ||Q|| for [P] ≡ [Q]
### Intersection Type Calculus of Bounded Dimension

#### Definition

Write $\Gamma \vdash M \leftrightarrow P : \sigma$ iff $\Gamma \vdash P : \sigma$ with $M \equiv [P]$.

- Clearly, $\Gamma \vdash s M : \sigma$ iff $\exists P. \Gamma \vdash M \leftrightarrow P : \sigma$

#### Definition ($\lambda_n^{[\cap]}$)

$\Gamma \vdash_n M : \sigma$ iff $\exists P. \Gamma \vdash M \leftrightarrow P : \sigma$ with $\|P\| \leq n$

#### Lemma

$\Gamma \vdash_s M : \sigma$ iff $\Gamma \vdash_n M : \sigma$ for some $n > 0$

#### Definition (Dimension)

The *set theoretic dimension* of a term $M$ at $\Gamma$ and $\sigma$ is

$$\dim_{\Gamma}^{\sigma} = \min\{n \mid \Gamma \vdash_n M : \sigma\}$$
Terms can be elaborated in non-increasing norm under $\beta$-reduction:

**Theorem (Subject Reduction for $\lambda^{[n]}_n$)**

*If $\Gamma \vdash M \rightarrow P : \tau$ and $M \rightarrow_{\beta} M'$, then there exists $R$ with $\|R\| \leq \|P\|$ such that $\Gamma \vdash M' \rightarrow R : \tau$.*

**Consequences:**

- Each dimensional fragment $\lambda^{[n]}_n$ is a meaningful type system.
- Inhabitation in bounded dimension for $\lambda^{[n]}_n$ can be limited to search for normal forms.
Problem (Inhabitation for $\lambda^{[n]}$)

Given environment $\Gamma$, type $\sigma$ and number $n > 0$:
is there a term $M$ such that $\Gamma \vdash_n M : \sigma$?

Theorem

The inhabitation problem for $\lambda^{[n]}$ is undecidable.

Proof.

By subject reduction and normalization it suffices to search for normal forms in norm $n$. Let $N$ be the size of input. By the subformula property [BCDC83] (Lemma 4.5), inhabitation in bounded norm $N$ is equivalent to inhabitation.

For $n = 1$ set-theoretic inhabitation is PSPACE-complete ([RU12] Cor. 22).
Inhabitation in Bounded Set Theoretic Dimension

Problem (Inhabitation for $\lambda^{[n]}$)

Given environment $\Gamma$, type $\sigma$ and number $n > 0$:

is there a term $M$ such that $\Gamma \vdash_n M : \sigma$?

Theorem

The inhabitation problem for $\lambda^{[n]}$ is undecidable.

Proof.

By subject reduction and normalization it suffices to search for normal forms in norm $n$. Let $N$ be the size of input. By the subformula property [BCDC83] (Lemma 4.5), inhabitation in bounded norm $N$ is equivalent to inhabitation.

For $n = 1$ set-theoretic inhabitation is PSPACE-complete ([RU12] Cor. 22).
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, and we prove decidability through a sound and complete algorithm. We then consider the inhabitation problem for an extended system typing the λ-calculus with pairs, and we prove the decidability in this case too. The extended system is interesting in its own, since it allows to characterize solvable terms in the λ-calculus with pairs.
### Definition

Treat types \([A_1, \ldots, A_n]\) and sets \(S\) as *multisets* \(s\) and let \(\sqcup\) denote multiset union.

#### Definition \((\Delta \vdash P : s)\)

\[
\begin{align*}
1 \leq i \leq n & \quad (\text{Var}) \\
\Delta, x : [A_1, \ldots, A_n] \vdash x\langle[A_i]\rangle : [A_i] & \\
\Delta, x : s \vdash P : [A] & \quad (\rightarrow I) \\
\Delta \vdash (\lambda x.P)\langle[s \rightarrow A]\rangle : [s \rightarrow A] & \\
\Delta \vdash P : [s \rightarrow A] \quad \Delta \vdash Q : s & \quad (\rightarrow E) \\
\Delta \vdash (P Q)\langle[A]\rangle : [A] & \\
\Delta \vdash P_i : [A_i] \text{ for } i = 1 \ldots n & \quad (\star) \\
\Delta \vdash \sqcup_{i=1}^n P_i : [A_1, \ldots, A_n] & \quad (\cap I)
\end{align*}
\]

\((\star)\) For each \(x\langle s\rangle\) in \(\sqcup_{i=1}^n P_i\): if \(x\) free in \(M\), then \(s \subseteq \Delta(x)\).
Inhabitation in Bounded Multiset Dimension

Definition
\[ \Gamma \vdash_n M : \sigma \iff \exists \Delta, P, s. (\Delta \vdash M \iff P : s \text{ with } \Gamma = \Delta^° \text{ and } \sigma = s^° \text{ and } \|P\| \leq n) \]
where \((\_)^°\) collapses multisets to underlying sets.

Problem \((\Gamma \vdash_n ? : \sigma)\)

Given \(\Gamma, \sigma\) and \(n > 0\):

is there a term \(M\) such that \(\Gamma \vdash_n M : \sigma\)?

Theorem

Inhabitation in bounded multiset dimension is \(\text{EXPSPACE}\)-complete. For each dimensional bound \(d > 0\), inhabitation is in \(\text{ATIME}(N^{2d})\) where \(N\) denotes the size of the input \(\Gamma\) and \(\sigma\).

Corollary

For each fixed \(n\) inhabitation in multiset dimension \(n\) is \(\text{PSPACE}\)-complete.
Dimensional Analysis of Rank 2 Typings

Proposition
Suppose we can derive $\Delta \vdash N \iff P : [A_1, \ldots, A_n]$ in rank 2, where $N$ is a normal form. Then $|P| = n$.

Consequence
Inhabitation in bounded multiset dimension is EXPSPACE-complete. Substantial generalization of inhabitation in rank 2 fragment [Urz09], generalizing across all ranks within EXPSPACE.

Compare to linear, non-idempotent system of Bucciareli, Kesner, Ronchi Della Rocca [BKRDR14]:

- Inhabitation is decidable [BKRDR14] and NP-complete [DR17a]
- Typability is undecidable.
\[ \Gamma = \{ x : (e \cap f) \rightarrow g, y : (((a \rightarrow a) \cap (b \rightarrow b)) \rightarrow e) \cap (((b \rightarrow b) \cap (c \rightarrow c)) \rightarrow f) \} \]
\[ \Gamma \models \_n : g \]

\[ X \equiv x(y(\lambda z.z)) \]
\[ Y \equiv y(\lambda z.z) \]
\[ Z \equiv \lambda z.z \]
\[ W \equiv z \]

\[ \Gamma \models Z : a \rightarrow a \]
\[ \Gamma \models Z : b \rightarrow b \]
\[ \Gamma \models Z : b \rightarrow b \]
\[ \Gamma \models Z : c \rightarrow c \]

\[ \Gamma, z : a \vdash W : a \]
\[ \Gamma, z : b \vdash W : b \]
\[ \Gamma, z : b \vdash W : b \]
\[ \Gamma, z : c \vdash W : c \]

\[ \| \bullet \| \]
\[ \| \bullet \| \leq n \]

See also talk at TYPES 2016, Rank 3 Inhabitation of Intersection Types Revisited [BDDR16] and extended version at arXiv.
Bounded Width Theorem

Definition (Width)

\[
\|a\| = 1, \\
\|\sigma \rightarrow A\| = \max\{\|\sigma\|, \|A\|\}, \\
\|A_1 \cap \cdots \cap A_m\| = \max\{m, \|A_1\|, \ldots, \|A_m\|\}
\]

Lift to environments, elaborations, and derivations by taking maximal width over all types appearing.

Theorem (Bounded Width Property, LICS 2017 [DR17b])

Let a derivation \(D \triangleright \Gamma \vdash M \leftarrow P : \sigma\) be given with \(\|P\| \leq d\). Then there exists a derivation \(D' \triangleright \Gamma' \vdash M \leftarrow P' : \sigma'\) such that \(\|D'\| \leq d \cdot |M|\) and \(\|P'\| = \|P\|\).

Proof.

By filtration with \(\mathcal{T}_P\) and using \(\|\mathcal{T}_P(D)\| \leq |T_P|\) together with

\[
|T_P| \leq \|P\| \cdot |M| \leq d \cdot |M|
\]
Typability in Bounded Dimension

Problem (Typability in bounded set-theoretic dimension)

- Given a λ-term $M$ and a dimension $d$, does there exist $\Gamma$ and $\sigma$ such that $\Gamma \vdash_d M : \sigma$?

(Recall: $\Gamma \vdash_d M : \sigma$ iff $\exists P. \Gamma \vdash P : \sigma$, $\lceil P \rceil \equiv M$, $\| P \| \leq d$)

Theorem (LICS 2017 [DR17b])

The typability problem in bounded set-theoretic dimension is PSPACE-complete.\(^a\)

The typability problem in bounded multiset dimension is in NP.

\(^a\)Upper bound constructed by nondeterministic reduction to standard unification.

Problem is nonelementary in rank:

- Kfoury, Mairson, Turbak, Wells, Relating Typability and Expressiveness in Finite-Rank Intersection Type Systems ICFP 1999 [KMTW99]
Dimensional Calculus – Research Challenges

- Implementation and applications of algorithms (synthesis and type inference)
- Models of dimensional calculus and relation to linear systems
- Is there a corresponding Church-style variant?
- Complexity: What is the complexity of $\beta$-equality under dimensional bound?
- Abstract vector space structure of elaborations
- Theory of principality in bounded dimension
- ...
Conclusion

Intersection types combine great logical simplicity and beauty with enormous expressive power ... ... and the work of the Torino group continues to inspire new and interesting problems and to enable new and unforeseen applications.
Thanks

... for all the inspiration and hospitality!
ET IN ARCADIA EGO


Jan Bessai, Andrej Dudenhefner, Boris Düdder, Moritz Martens, and Jakob Rehof. Combinatory Logic Synthesizer.


Jan Bessai, Andrej Dudenhofner, Boris Düdder, Moritz Martens, and Jakob Rehof.
Combinatory Process Synthesis.

Jan Bessai, Boris Düdder, Andrej Dudenhofner, and Jakob Rehof.
Rank 3 Inhabitation of Intersection Types Revisited.

Jan Bessai, Boris Düdder, George T. Heineman, and Jakob Rehof.
Combinatory Synthesis of Classes using Feature Grammars.


Marcin Benke, Aleksy Schubert, and Daria Walukiewicz-Chrzaszcz.
Synthesis of Functional Programs with Help of First-Order Intuitionistic Logic.

Mario Coppo, Mariangiola Dezani-Ciancaglini, and Betti Venneri.
*Principal Type Schemes and Lambda-Calculus Semantics*, pages 480–490.
Ed.: R. Hindley and J. Seldin.

Mario Coppo and Paola Giannini.
Principal types and unification for a simple intersection type system.


M. Dezani-Ciancaglini and R. Hindley.
Intersection Types for Combinatory Logic.

Boris Düdder, Oliver Garbe, Moritz Martens, Jakob Rehof, and Paweł Urzyczyn.
Using Inhabitation in Bounded Combinatory Logic with Intersection Types for Composition Synthesis.

Boris Düdder, Moritz Martens, and Jakob Rehof.
Intersection Type Matching with Subtyping.
Boris Düdder, Moritz Martens, and Jakob Rehof.  
Staged Composition Synthesis. 

Andrej Dudenhefner, Moritz Martens, and Jakob Rehof.  
The Intersection Type Unification Problem. 

A. Dudenhefner, M. Martens, and J. Rehof.  
The Algebraic Intersection Type Unification Problem.
Boris Düdder, Moritz Martens, Jakob Rehof, and Paweł Urzyczyn.
Bounded Combinatory Logic.

Andrej Dudenhefner and Jakob Rehof.
Intersection Type Calculi of Bounded Dimension.

Andrej Dudenhefner and Jakob Rehof.
Typability in Bounded Dimension.


T. Freeman and F. Pfenning.
Refinement Types for ML.

George T. Heineman, Jan Bessai, Boris Düdder, and Jakob Rehof.
A long and winding road towards modular synthesis.


Assaf J. Kfoury, Harry G. Mairson, Franklyn A. Turbak, and J. B. Wells.
Relating typability and expressiveness in finite-rank intersection type systems (extended abstract).

T. Kurata and M. Takahashi.
Decidable properties of intersection type systems.

Dariusz Kusmierek.
The Inhabitation Problem for Rank Two Intersection Types.


Simona Ronchi Della Rocca.
Principal Type Scheme and Unification for Intersection Type Discipline.


Jakob Rehof.
Towards Combinatory Logic Synthesis.

Jakob Rehof and Paweł Urzyczyn.
Finite Combinatory Logic with Intersection Types.

Jakob Rehof and Paweł Urzyczyn.
The Complexity of Inhabitation with Explicit Intersection.
In Robert L. Constable and Alexandra Silva, editors, *Logic and Program Semantics - Essays Dedicated to Dexter Kozen on*

S. Salvati.
Recognizability in the Simply Typed Lambda-Calculus.

Urzyczyn and Loader are logically related.

Richard Statman.
Intuitionistic Propositional Logic is Polynomial-space Complete.

R. Statman.
A finite model property for intersection types.

P. Urzyczyn.
The Emptiness Problem for Intersection Types.

P. Urzyczyn.
Inhabitation of Low-Rank Intersection Types.

Steffen van Bakel.
Strict Intersection Types for the Lambda Calculus.
Joe B. Wells and Boris Yakobowski.
Graph-Based Proof Counting and Enumeration with Applications for Program Fragment Synthesis.