Computational and Fine-Structure Aspects of Intersection Types

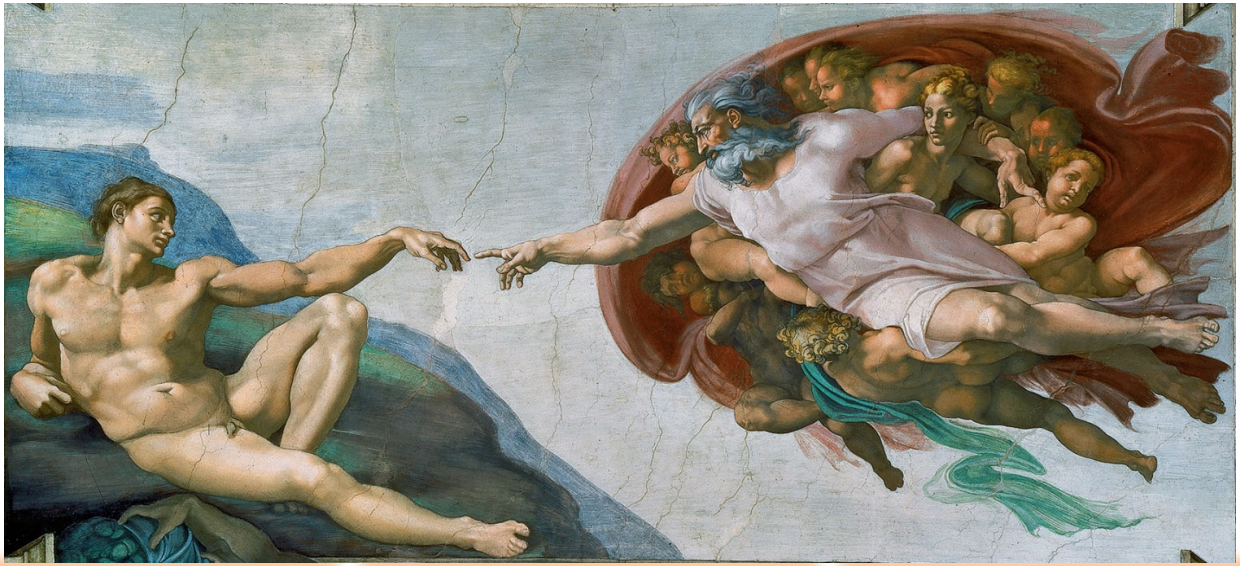
A personal encounter with intersection types

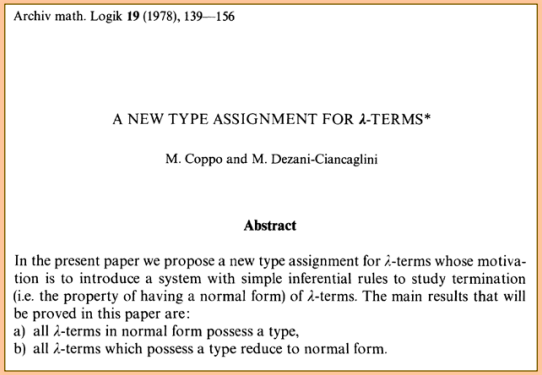
Jakob Rehof

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

From the Beginning ...





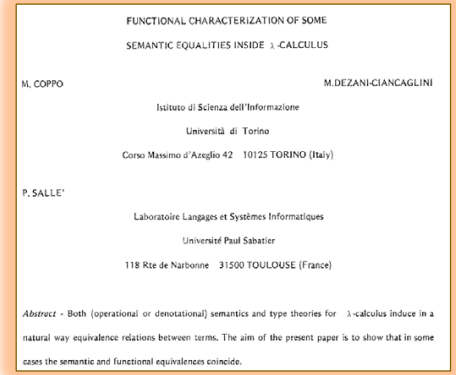
Archiv math. Logik **19** (1978), 139—156

A NEW TYPE ASSIGNMENT FOR λ-TERMS*

M. Coppo and M. Dezani-Ciancaglini

Abstract

In the present paper we propose a new type assignment for λ-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 λ-terms. The main results that will be proved in this paper are:
a) all λ-terms in normal form possess a type,
b) all λ-terms which possess a type reduce to normal form.



FUNCTIONAL CHARACTERIZATION OF SOME

SEMANTIC EQUALITIES INSIDE λ-CALCULUS

M. COPPO M.DEZANI-CIANCAGLINI

Istituto di Scienza dell'Informazione

Università di Torino

Corso Massimo d'Azeglio 42 10125 TORINO (Italy)

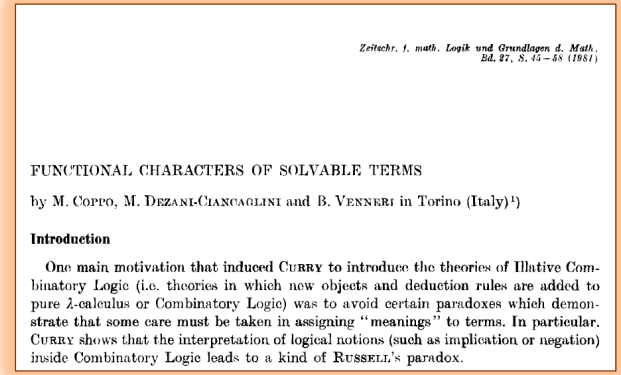
P. SALLE'

Laboratoire Langages et Systèmes Informatiques

Université Paul Sabatier

118 Rte de Narbonne 31500 TOULOUSE (France)

Abstract - Both (operational or denotational) semantics and type theories for λ-calculus induce in a natural way equivalence relations between terms. The aim of the present paper is to show that in some cases the semantic and functional equivalences coincide.



Zeitschr. f. math. Logik und Grundlagen d. Math.
Bd. 27, S. 45–58 (1981)

FUNCTIONAL CHARACTERS OF SOLVABLE TERMS

by M. Coppo, M. Dezani-Ciancaglini and B. Venneri in Torino (Italy)[1])

Introduction

One main motivation that induced Curry to introduce the theories of Illative Combinatory Logic (i.e. theories in which new objects and deduction rules are added to pure λ-calculus or Combinatory Logic) was to avoid certain paradoxes which demonstrate that some care must be taken in assigning "meanings" to terms. In particular, Curry shows that the interpretation of logical notions (such as implication or negation) inside Combinatory Logic leads to a kind of Russell's paradox.

Intersection types combine great logical simplicity and beauty with enormous expressive power, capturing deep semantic properties of λ-terms (normalization, solvability, ...)

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, ...)

Acknowledgements

- My students in Dortmund (former and present) including: Jan Bessai (Dortmund), Boris Düdder (formerly Dortmund, now Copenhagen), Andrej Dudenhefner (Dortmund), Moritz Martens (formerly Dortmund, now in industry)
- Collaborators and colleagues, including: Mariangiola Dezani, Simona Ronchi Della Rocca, Mario Coppo and the Torino λ-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)

“BCD”

THE JOURNAL OF SYMBOLIC LOGIC
Volume 48, Number 4, Dec. 1983

A FILTER LAMBDA MODEL AND THE COMPLETENESS OF TYPE ASSIGNMENT[1]

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 λ-calculus. In [11] Scott gave a natural semantics for this type assignment 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 λ-model, among other reasons because all λ-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 σ, τ, ρ, 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 $\mathbb{C}$, ω is a special (universal type) constant, and α, β, γ 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 τ and σ as *components*. Intersection $\cap$ is tacitly ACI.

λ-Calculus with Intersection Types

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

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}\ (\text{Var}) \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}\ (\to\text{I})$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau}\ (\to\text{E})$$

$$\frac{\Gamma \vdash M : \tau_1 \qquad \Gamma \vdash M : \tau_2}{\Gamma \vdash M : \tau_1 \cap \tau_2}\ (\cap\text{I})$$

$$\frac{\Gamma \vdash M : \tau_1 \cap \tau_2}{\Gamma \vdash M : \tau_i}\ (\cap\text{E}) \qquad \frac{\Gamma \vdash M : \tau \qquad \tau \leq \sigma}{\Gamma \vdash M : \sigma}\ (\leq)$$

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

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

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

- $(\sigma \to \tau_1) \cap (\sigma \to \tau_2) = \sigma \to (\tau_1 \cap \tau_2)$
- $(\sigma_1 \to \tau_1) \cap (\sigma_2 \to \tau_2) \leq (\sigma_1 \cap \sigma_2) \to (\tau_1 \cap \tau_2)$

Subtyping (BCD)

Problem (Subtyping)

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

Lemma (Beta-Soundness [BCDC83])

Given $\sigma = \bigcap_{i \in I}(\sigma_i \to \tau_i) \cap \bigcap_{j \in J} a_j \cap \bigcap_{k \in K} \alpha_k$, 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 $\sigma \leq \sigma' \to \tau' \neq \omega$ for some $\sigma', \tau' \in \mathbb{T}$, then $I' = \{i \in I \mid \sigma' \leq \sigma_i\} \neq \emptyset$ and $\bigcap_{i \in I'} \tau_i \leq \tau'$.*

Theorem ([DMR17])

Subtyping is decidable in quadratic time.

Subtyping (BCD)

Problem (Subtyping)

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

Lemma (Beta-Soundness [BCDC83])

Given $\sigma = \bigcap_{i \in I}(\sigma_i \to \tau_i) \cap \bigcap_{j \in J} a_j \cap \bigcap_{k \in K} \alpha_k$, 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 $\sigma \leq \sigma' \to \tau' \neq \omega$ for some $\sigma', \tau' \in \mathbb{T}$, then $I' = \{i \in I \mid \sigma' \leq \sigma_i\} \neq \emptyset$ and $\bigcap_{i \in I'} \tau_i \leq \tau'$.*

Theorem ([DMR17])

Subtyping is decidable in quadratic time.

Subtyping (BCD)

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* $\mathrm{Var}(\sigma_i) = \emptyset$ *or* $\mathrm{Var}(\tau_i) = \emptyset$*, is there a substitution* $S \colon \mathbb{V} \to \mathbb{T}$ *such that* $S(\sigma_i) \leq S(\tau_i)$ *for* $1 \leq i \leq n$*?*

We say that a substitution S satisfies $\{\sigma_1 \dot{\leq} \tau_1, \ldots, \sigma_n \dot{\leq} \tau_n\}$ 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 $C = \{\sigma_1 \dot{\leq} \tau_1, \ldots, \sigma_n \dot{\leq} \tau_n\}$*, where for each* $i \in \{1, \ldots, n\}$ *we have* $\mathrm{Var}(\sigma_i) = \emptyset$ *or* $\mathrm{Var}(\tau_i) = \emptyset$*, is there a substitution* $S \colon \mathbb{V} \to \mathbb{T}$ *such that* $S(\sigma_i) \leq S(\tau_i)$ *for* $1 \leq i \leq n$*?*

We say that a substitution S satisfies $\{\sigma_1 \dot{\leq} \tau_1, \ldots, \sigma_n \dot{\leq} \tau_n\}$ 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 (Satisfiability)

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

Problem (Algebraic unification)

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

Problem (Algebraic unification)

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

Problem (Algebraic unification)

Given a set of constraints $C = \{\sigma_1 \doteq \tau_1, \ldots, \sigma_n \doteq \tau_n\}$*, is there a substitution* $S \colon \mathbb{V} \to \mathbb{T}$ *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 ω) is derived in [Sta15]. We add two additional axioms **(U)** and **(RE)** to incorporate the universal type ω.

Definition (ACIUD$_l$ReAb)

The equational theory ACIUD$_l$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$_l$) $(\sigma \to \tau) \cap (\sigma \to \tau') = \sigma \to \tau \cap \tau'$

(RE) $\omega = \omega \to \omega$

(AB) $\sigma \to \tau = (\sigma \to \tau) \cap (\sigma \cap \sigma' \to \tau)$

Subtyping (BCD)

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

Definition (ACIUD$_l$ReAb)

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

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

(I) $\sigma + \sigma = \sigma$

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

(D$_l$) $(\sigma * \tau) + (\sigma * \tau') = \sigma * (\tau + \tau')$

(RE) $\omega = \omega * \omega$

(AB) $\sigma * \tau = (\sigma * \tau) + ((\sigma + \sigma') * \tau)$

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

Principality and Unification

PRINCIPAL TYPE SCHEMES AND λ-CALCULUS SEMANTICS

M. Coppo, M. Dezani-Ciancaglini and B. Venneri
Istituto di Scienza dell'Informazione
Università di Torino
Corso M. D'Azeglio 42 - 10125 TORINO

Dedicated to H.B. Curry on the occasion of his 80th Birthday

1. INTRODUCTION

Basic functionality theory (as 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 deductions.

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

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 this one as instances) shows an internal coherence between all functional characters of X.

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

Theoretical Computer Science 59 (1988) 181–209
North-Holland

PRINCIPAL TYPE SCHEME AND UNIFICATION FOR INTERSECTION TYPE DISCIPLINE

Simona RONCHI DELLA ROCCA
Dipartimento di Informatica, Universita' di Torino, 10149 Torino, Italy

Abstract. The intersection type discipline for the λ-calculus (ITD) is an extension of the classical functionality theory of Curry. In the ITD a term satisfying a given property has a principal type scheme in an extended meaning, i.e., there is a type scheme deducible for it from which all and only the type schemes deducible for it are reachable, by means of suitable operations. The problem of finding the principal type scheme for a term, if it exists, is semidecidable. In the paper a procedure is shown, building the principal type scheme of a term through the construction of the most general unifier for intersection type schemes.

INFORMATION AND COMPUTATION 122, 70–96 (1995)

Principal Types and Unification for Simple Intersection Type Systems

MARIO COPPO AND PAOLA GIANNINI
Università di Torino, Dipartimento di Informatica, Corso Svizzera 185, 10149 Turin, Italy
E-mail: {coppo, giannini}@di.unito.it

In this paper we introduce a decidable restriction of the intersection type discipline. The restriction is based on a uniform principle rather than on a notion of rank. For this system we give a type checking algorithm that we prove to be sound and complete. An extended language of types with labelled intersections and intersection schemes is needed to describe the principal type of terms which turns out to be unique modulo a set of equations between labels. The unification algorithm for labelled types, that extends first order unification, could be used as a basis to define inference algorithms for other (more powerful) systems with intersection.

terms have a proper type, even when they are used in a consistent way. Take for instance the operators $twice = \lambda f.\lambda x.f(fx)$ and $K = \lambda x.\lambda y.x$. The types given to *twice* in functionality theory are such that the term (*twice* K) has no type, whereas (*twice* K) reduces to $\lambda x.\lambda u.\lambda v.x$ which is well typed. In ML, however, we can assign a type to the term $let\ f = K\ in\ \lambda x.f(fx)$. Here the body of *twice* must be typed, in the *let* construct, using explicitly the fact that f is K. That means that any other use of *twice* requires a different type checking. This is contrary to the functional style of

[CDCV80, 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).

Inhabitation

Pieter Brueghel the Elder - The Dutch Proverbs - Google Art Project.jpg
1559

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

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

Inhabitation and Synthesis

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

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

$$\Gamma \vdash M : \tau$$

- Type environment
- Library

- Inhabitant
- Synthesized program

- Type
- Specification (synthesis goal)

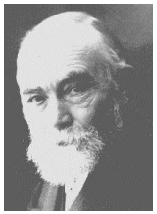
Inhabitation and Synthesis

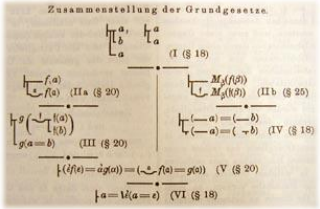
Component-oriented Synthesis

Synthesis *relative* to *library* (repository) of components

Combinatory Logic Synthesis (CLS)

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

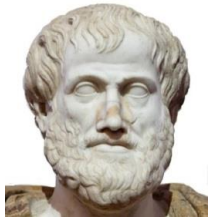




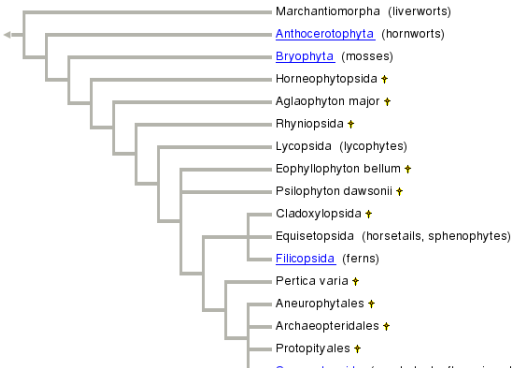
$$
\begin{array}{l}
(\forall\, x; \forall\, y; \forall\, z; (p(x,y) \wedge p(y,z) \rightarrow p(x,z)) \wedge \\
\forall\, x; \forall\, y; (p(x,y) \rightarrow \neg p(y,x)) \wedge \\
\exists\, x; (p(a,x) \wedge p(x,b)) \\
) \rightarrow \neg p(b,a)
\end{array}
$$

Bottom-up specification
Hoare logic

CLS



Classification
Taxonomy
...
Types



Combinatory Logic Synthesis (CLS)

A type-theoretic approach to component-oriented synthesis

Über die Bausteine der mathematischen Logik.

Von

M. Schönfinkel in Moskau[1]).

§ 1.

Es entspricht dem Wesen der axiomatischen Methode, 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 undefiniert 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 aufzubauen. Begreiflicherweise wird man sich im Sinne dieser Aufgabe bezüglich des Verlangens nach Einfachheit der an den Anfang zu stellenden Begriffe entsprechend bescheiden müssen.

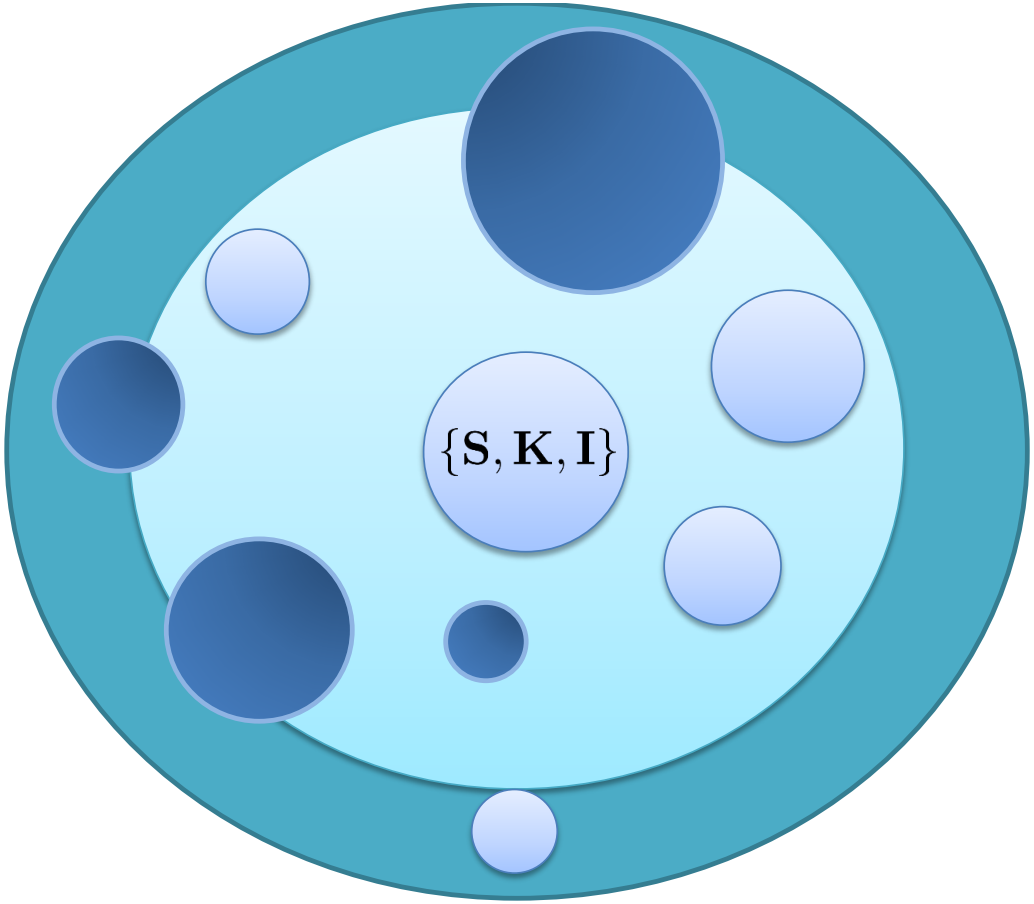
Bekanntlich lassen sich die grundlegenden *Aussagenverknüpfungen* der mathematischen Logik, die ich hier in der von Hilbert in seinen Vorlesungen verwendeten Bezeichnungsweise wiedergebe:

$$\bar{a}, \quad a \vee b, \quad a \,\&\, b, \quad a \rightarrow b, \quad a \sim b$$

[1]) Die folgenden Gedanken wurden vom Verfasser am 7. Dez. 1920 vor der Mathematischen Gesellschaft in Göttingen vorgetragen. Ihre formale und stilistische Durcharbeitung für diese Veröffentlichung wurde von H. Behmann in Göttingen übernommen.

Mathematische Annalen. 92. 20

CLS World View



- CL over *arbitrary bases*:

  General theory of component collections (repositories)

- The implementation theory" $\{\mathbf{S}, \mathbf{K}, \mathbf{I}\}$ is one, very special case

Combinatory Logic Synthesis (CLS)

A type-theoretic approach to component-oriented synthesis

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

CLS World View

$$\Gamma \vdash_{\mathrm{CL}} e : \tau$$

- 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 Γ and τ**, *does there exist combinatory expression e such that* $\Gamma \vdash e : \tau$*?*
- Inhabitation for fixed base $\{\mathbf{S}, \mathbf{K}, \mathbf{I}\}$ 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 Γ and τ**, *does there exist combinatory expression e such that* $\Gamma \vdash e : \tau$*?*
- Inhabitation for fixed base $\{\mathbf{S}, \mathbf{K}, \mathbf{I}\}$ 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 Γ and τ**, *does there exist combinatory expression e such that* $\Gamma \vdash e : \tau$*?*
- Inhabitation for fixed base $\{\mathbf{S}, \mathbf{K}, \mathbf{I}\}$ 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** Γ **and** τ*, does there exist combinatory expression e such that* $\Gamma \vdash e : \tau$*?*
- Inhabitation for fixed base $\{\mathbf{S}, \mathbf{K}, \mathbf{I}\}$ 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

Combinatory Logic with Intersection Types $\text{CL}(\rightarrow, \cap)$

Definition

$$\frac{}{\Gamma, X : \tau \vdash X : S(\tau)}(\text{var}) \qquad \frac{\Gamma \vdash e : \tau \rightarrow \sigma \quad \Gamma \vdash e' : \tau}{\Gamma \vdash (e\ e') : \sigma}(\rightarrow\text{E})$$

$$\frac{\Gamma \vdash e : \tau \quad \Gamma \vdash e : \sigma}{\Gamma \vdash e : \tau \cap \sigma}(\cap\text{I}) \qquad \frac{\Gamma \vdash e : \tau \quad \tau \leq \sigma}{\Gamma \vdash e : \sigma}(\leq)$$

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

Note

But, in CLS, the combinatory theory Γ represents an arbitrary repository (basis *not fixed*)

Bounded Combinatory Logic $\text{BCL}_k(\rightarrow, \cap)$

Definition (Levels)

$$
\begin{array}{lcl}
\ell(a) & = & 0, \text{ for } a \in \mathbb{A}; \\
\ell(\tau \rightarrow \sigma) & = & 1 + \max\{\ell(\tau), \ell(\sigma)\}; \\
\ell(\bigcap_{i=1}^{n} \tau_i) & = & \max\{\ell(\tau_i) \mid i = 1, \ldots, n\}. \\
\\
\ell(S) & = & \max\{\ell(S(\alpha)) \mid S(\alpha) \neq \alpha\}
\end{array}
$$

Definition ($\text{BCL}_k(\rightarrow, \cap)$, $k \geq 0$)

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

$$\frac{\Gamma \vdash_k e : \tau \quad \Gamma \vdash_k e : \sigma}{\Gamma \vdash_k e : \tau \cap \sigma}(\cap\text{I}) \qquad \frac{\Gamma \vdash_k e : \tau \quad \tau \leq \sigma}{\Gamma \vdash_k e : \sigma}(\leq)$$

- BCL_k: *Bounded Combinatory Logic*, CSL 2012 [DMRU12]
- FCL: *Finite Combinatory Logic with Intersection Types*, TLCA 2011 [RU11], taking $S = id$.

Bounded Combinatory Logic $\text{BCL}_k(\rightarrow, \cap)$

Definition (Levels)

$$
\begin{array}{lcl}
\ell(a) & = & 0, \text{ for } a \in \mathbb{A}; \\
\ell(\tau \rightarrow \sigma) & = & 1 + \max\{\ell(\tau), \ell(\sigma)\}; \\
\ell(\bigcap_{i=1}^{n} \tau_i) & = & \max\{\ell(\tau_i) \mid i = 1, \ldots, n\}. \\
\\
\ell(S) & = & \max\{\ell(S(\alpha)) \mid S(\alpha) \neq \alpha\}
\end{array}
$$

Definition ($\text{BCL}_k(\rightarrow, \cap)$, $k \geq 0$)

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

$$
\frac{\Gamma \vdash_k e : \tau \quad \Gamma \vdash_k e : \sigma}{\Gamma \vdash_k e : \tau \cap \sigma}(\cap\text{I}) \qquad \frac{\Gamma \vdash_k e : \tau \quad \tau \leq \sigma}{\Gamma \vdash_k e : \sigma}(\leq)
$$

- BCL_k: *Bounded Combinatory Logic*, CSL 2012 [DMRU12]
- FCL: *Finite Combinatory Logic with Intersection Types*, TLCA 2011 [RU11], taking $S = id$.

Complexity for Finite and Bounded CL

Theorem (TLCA 2011 [RU11])

For finite combinatory logic FCL*:*

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

Theorem (CSL 2012 [DMRU12])

For bounded combinatory logic BCL_k*:*

1. *Relativized inhabitation in* $\text{BCL}_k(\rightarrow)$ *is* EXPTIME*-complete for all* k
2. *Relativized inhabitation in* $\text{BCL}_k(\rightarrow, \cap)$ *is* $(k+2)$*-*EXPTIME*-complete*

Upper Bound ATM for $\textsc{bcl}_k(\rightarrow, \cap)$: $\textsc{Aspace}(\exp_{k+1}(n))$

Input : Γ, τ, 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 :

CHOOSE $(x : \sigma) \in \Gamma$;

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

$\sigma' = (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \cap \cdots \cap$
$(1 \rightarrow 1) \rightarrow (1 \rightarrow 1) \rightarrow (1 \rightarrow 1)$

CHOOSE $m \in \{0, \ldots, \|\sigma'\|\}$;

CHOOSE $P \subseteq \mathbb{P}_m(\sigma')$;

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

IF $(\bigcap_{\pi \in P} tgt_m(\pi) \leq \tau)$ THEN

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

IF $(m = 0)$ THEN ACCEPT;

ELSE

FORALL$(i = 1 \ldots m)$

$\tau := \bigcap_{\pi \in P} arg_i(\pi)$;

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

GOTO *loop*;

ELSE REJECT;

$(x\ f)\ f : (0 \rightarrow 0) \cap (1 \rightarrow 1)$

Upper Bound ATM for $\textsc{bcl}_k(\rightarrow, \cap)$: $\textsc{Aspace}(\exp_{k+1}(n))$

Input : Γ, τ, 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 $\sigma' := \bigcap\{S(\sigma) \mid S \in \mathcal{S}_x^{(\Gamma,\tau,k)}\}$;

$\sigma' = (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \cap \cdots \cap$
$\quad (1 \rightarrow 1) \rightarrow (1 \rightarrow 1) \rightarrow (1 \rightarrow 1)$

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

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

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

5 IF $(\bigcap_{\pi \in P} tgt_m(\pi) \leq \tau)$ THEN

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

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

7 ELSE

8 FORALL$(i = 1 \ldots m)$

9 $\tau := \bigcap_{\pi \in P} arg_i(\pi)$;

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

10 GOTO *loop*;

11 ELSE REJECT;

$(x\ f)\ f : (0 \rightarrow 0) \cap (1 \rightarrow 1)$

Upper Bound ATM for $\textsc{bcl}_k(\rightarrow, \cap)$: $\textsc{Aspace}(\exp_{k+1}(n))$

	Input : Γ, τ, 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 $(x : \sigma) \in \Gamma$;	$\sigma' = (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \cap \cdots \cap$
2	$\sigma' := \bigcap\{S(\sigma) \mid S \in \mathcal{S}_x^{(\Gamma,\tau,k)}\}$;	$(1 \rightarrow 1) \rightarrow (1 \rightarrow 1) \rightarrow (1 \rightarrow 1)$
3	CHOOSE $m \in \{0, \ldots, \lVert\sigma'\rVert\}$;	$(0 \rightarrow 1) \rightarrow (1 \rightarrow 0) \rightarrow (0 \rightarrow 0) \cap$
4	CHOOSE $P \subseteq \mathbb{P}_m(\sigma')$;	$(1 \rightarrow 0) \rightarrow (0 \rightarrow 1) \rightarrow (1 \rightarrow 1)$
5	IF $(\bigcap_{\pi \in P} tgt_m(\pi) \leq \tau)$ THEN	$(0 \rightarrow 0) \cap (1 \rightarrow 1) \leq \tau$
6	IF $(m = 0)$ THEN ACCEPT;	
7	ELSE	
8	FORALL$(i = 1 \ldots m)$	
9	$\tau := \bigcap_{\pi \in P} arg_i(\pi)$;	$\tau := (0 \rightarrow 1) \cap (1 \rightarrow 0)$ $\tau := (1 \rightarrow 0) \cap (0 \rightarrow 1)$
10	GOTO *loop*;	
11	ELSE REJECT;	
		$(x\ f)\ f : (0 \rightarrow 0) \cap (1 \rightarrow 1)$

Upper Bound ATM for $\textsc{bcl}_k(\rightarrow, \cap)$: $\textsc{Aspace}(\exp_{k+1}(n))$

	Input : Γ, τ, 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 $(x : \sigma) \in \Gamma$;	$\sigma' = (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \cap \cdots \cap$
2	$\sigma' := \bigcap\{S(\sigma) \mid S \in \mathcal{S}_x^{(\Gamma,\tau,k)}\}$;	$(1 \rightarrow 1) \rightarrow (1 \rightarrow 1) \rightarrow (1 \rightarrow 1)$
3	CHOOSE $m \in \{0, \ldots, \lVert\sigma'\rVert\}$;	$(0 \rightarrow 1) \rightarrow (1 \rightarrow 0) \rightarrow (0 \rightarrow 0) \cap$
4	CHOOSE $P \subseteq \mathbb{P}_m(\sigma')$;	$(1 \rightarrow 0) \rightarrow (0 \rightarrow 1) \rightarrow (1 \rightarrow 1)$
5	IF $(\bigcap_{\pi \in P} tgt_m(\pi) \leq \tau)$ THEN	$(0 \rightarrow 0) \cap (1 \rightarrow 1) \leq \tau$
6	IF $(m = 0)$ THEN ACCEPT;	
7	ELSE	
8	FORALL$(i = 1 \ldots m)$	
9	$\tau := \bigcap_{\pi \in P} arg_i(\pi)$;	$\tau := (0 \rightarrow 1) \cap (1 \rightarrow 0)$ $\tau := (1 \rightarrow 0) \cap (0 \rightarrow 1)$
10	GOTO *loop*;	
11	ELSE REJECT;	
		$(x\ f)\ f : (0 \rightarrow 0) \cap (1 \rightarrow 1)$

Upper Bound ATM for $\textsc{bcl}_k(\rightarrow, \cap)$: $\textsc{Aspace}(\exp_{k+1}(n))$

	Input : Γ, τ, 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 $(x : \sigma) \in \Gamma$;	$\sigma' = (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \cap \cdots \cap$
2	$\sigma' := \bigcap\{S(\sigma) \mid S \in \mathcal{S}_x^{(\Gamma,\tau,k)}\}$;	$(1 \rightarrow 1) \rightarrow (1 \rightarrow 1) \rightarrow (1 \rightarrow 1)$
3	CHOOSE $m \in \{0, \ldots, \lVert\sigma'\rVert\}$;	$(0 \rightarrow 1) \rightarrow (1 \rightarrow 0) \rightarrow (0 \rightarrow 0) \cap$
4	CHOOSE $P \subseteq \mathbb{P}_m(\sigma')$;	$(1 \rightarrow 0) \rightarrow (0 \rightarrow 1) \rightarrow (1 \rightarrow 1)$
5	IF $(\bigcap_{\pi \in P} tgt_m(\pi) \leq \tau)$ THEN	$(0 \rightarrow 0) \cap (1 \rightarrow 1) \leq \tau$
6	IF $(m = 0)$ THEN ACCEPT;	
7	ELSE	
8	FORALL$(i = 1 \ldots m)$	
9	$\tau := \bigcap_{\pi \in P} arg_i(\pi)$;	$\tau := (0 \rightarrow 1) \cap (1 \rightarrow 0)$ $\tau := (1 \rightarrow 0) \cap (0 \rightarrow 1)$
10	GOTO *loop*;	
11	ELSE REJECT;	
		$(x\ f)\ f : (0 \rightarrow 0) \cap (1 \rightarrow 1)$

Upper Bound ATM for $\textsc{bcl}_k(\rightarrow, \cap)$: $\textsc{Aspace}(\exp_{k+1}(n))$

	Input : Γ, τ, 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 $(x : \sigma) \in \Gamma$;	$\sigma' = (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \cap \cdots \cap$
2	$\sigma' := \bigcap\{S(\sigma) \mid S \in \mathcal{S}_x^{(\Gamma,\tau,k)}\}$;	$(1 \rightarrow 1) \rightarrow (1 \rightarrow 1) \rightarrow (1 \rightarrow 1)$
3	CHOOSE $m \in \{0, \ldots, \lVert\sigma'\rVert\}$;	$(0 \rightarrow 1) \rightarrow (1 \rightarrow 0) \rightarrow (0 \rightarrow 0) \cap$
4	CHOOSE $P \subseteq \mathbb{P}_m(\sigma')$;	$(1 \rightarrow 0) \rightarrow (0 \rightarrow 1) \rightarrow (1 \rightarrow 1)$
5	IF $(\bigcap_{\pi \in P} tgt_m(\pi) \leq \tau)$ THEN	$(0 \rightarrow 0) \cap (1 \rightarrow 1) \leq \tau$
6	IF $(m = 0)$ THEN ACCEPT;	
7	ELSE	
8	FORALL$(i = 1 \ldots m)$	
9	$\tau := \bigcap_{\pi \in P} arg_i(\pi)$;	$\tau := (0 \rightarrow 1) \cap (1 \rightarrow 0)$ $\tau := (1 \rightarrow 0) \cap (0 \rightarrow 1)$
10	GOTO *loop*;	
11	ELSE REJECT;	
		$(x\ f)\ f : (0 \rightarrow 0) \cap (1 \rightarrow 1)$

Ongoing: optimization & algorithm engineering

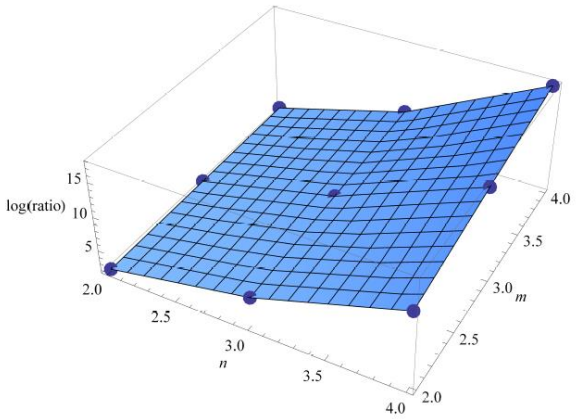
From B. Düdder: *Automatic Synthesis of Component & Connector-Software Architectures with Bounded Combinatory Logic*, Diss. Dortmund, Aug. 2014, [Düd14].

Algorithm 4.5: ATM with lookahead-test

Input : Γ, τ— all types in Γ and $\tau = \bigcap_{i \in I} \tau_i$ organized
loop :
CHOOSE $(x : \sigma) \in \Gamma$;
write $\sigma \equiv \bigcap_{j \in J} \sigma_j$
FOR EACH $i \in I, j \in J, m \leq \|\sigma\|$ DO
  $\texttt{candidates}(i, j, m) := \texttt{Match}(tgt_m(\sigma_j) \leq \tau_i)$
$M := \{m \leq \|\sigma\| \mid \forall_{i \in I} \exists_{j \in J} : \texttt{candidates}(i, j, m) = \textbf{true}\}$
CHOOSE $m \in M$;
FOR EACH $i \in I$ DO
  CHOOSE $j_i \in J$ with $\texttt{candidates}(i, j_i, m) = \textbf{true}$
  CHOOSE S_i a substitution
  CHOOSE $\pi_i \in \mathbb{P}_m(\overline{S_i(\sigma_{j_i})})$ with $tgt_m(\pi_i) \leq \tau_i$ and
  $\forall 1 \leq l \leq m\ \forall \pi' \in \overline{arg_l(\pi_i)}\ \exists (y : \rho) \in \Gamma\ \exists$ a path ρ'
  in $\rho\ \exists k\ :\ \texttt{Match}(tgt_k(\rho') \leq \pi') = \textbf{true}$
IF $(m = 0)$ THEN ACCEPT;
ELSE FORALL$(l = 1 \ldots m)$
  $\tau := \bigcap_{i \in I} arg_l(\pi_i)$;
  GOTO *loop*;

		Unoptimized ATM		Lookahead-ATM	
n	m	#I	$\overline{\text{RT}}$/ms	#I	$\overline{\text{RT}}$/ms
2	2	73	112.2	9	109.8
2	3	973	209.6	49	111.4
2	4	11 665	2245.2	257	124.4
3	2	43 905	12 504	55	124
3	3	$1.4 \times 10^{10*}$	–	2188	354
3	4	$4.8 \times 10^{12*}$	–	78 733	50 959.6
4	2	$1.3 \times 10^{14*}$	–	33	142.4
4	3	$6.6 \times 10^{18*}$	–	3889	1314.6
4	4	$3.3 \times 10^{23*}$	–	640 001	7.5×10^6

Table 4.1: Experimental Data for Γ^m_n



Refinement (after [FP91])

Definition ([SMGB12])

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

$$\begin{array}{lcl} \mathcal{U}_X(o) & = & X^{\cap} \\ \mathcal{U}_X(\tau \to \sigma) & = & (\mathcal{U}_X(\tau) \Rightarrow \mathcal{U}_X(\sigma))^{\cap} \end{array}$$

- With such types we can represent any finite function $f : \mathbf{A} \to \mathbf{B}$ at the type level by $\bigcap_{a \in \mathbf{A}}(a \to f(a))$
- We can express finite abstract interpretations, e.g.,

$$\begin{array}{lcl} \texttt{succ} & : & (\texttt{Nat} \to \texttt{Nat}) \cap (\texttt{zero} \to \texttt{pos}) \cap (\texttt{pos} \to \texttt{pos}) \cap \\ & & (\texttt{even} \to \texttt{odd}) \cap (\texttt{odd} \to \texttt{even}) \end{array}$$

- *Inhabitation* (λ-calculus) is undecidable. *Proof*: Note that [SMGB12] uses only uniform types for λ-definability. □

CL($\rightarrow, \cap$) over Uniform (Refinement) Types

Definition

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

$$\begin{array}{lcl} \mathcal{U}_X(o) & = & X^{\cap} \\ \mathcal{U}_X(\tau \rightarrow \sigma) & = & (\mathcal{U}_X(\tau) \Rightarrow \mathcal{U}_X(\sigma))^{\cap} \end{array}$$

Corollary

Relativized inhabitation with uniform types is nonelementary recursive.

Proof.

Upper bound: every problem $\Gamma \vdash ? : \sigma$ is decidable within $\textsc{bcl}_k(\rightarrow, \cap)$ with $k = \max\{\ell(\tau) \mid \tau \in rn(\Gamma)\}$.
Lower bound: notice that all constructions in l.b. for $\textsc{bcl}_k(\rightarrow, \cap)$ can be carried out with uniform types. $\square$

Corollary: Henkin's theory Ω in $\textsc{bcl}_k(\rightarrow, \cap)$

Satisfiability of formulae

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

where x^k ranges over $\mathbf{D}_k$ with $\mathbf{D}_0 = \{0, 1\}$, $\mathbf{D}_{k+1} = \mathcal{P}(\mathbf{D}_k)$.

L. Henkin: A theory of propositional types, Fundamenta Mathematicae 52 (1963) 323–344.

Representation in $\textsc{bcl}_k(\rightarrow, \cap)$ (for sufficiently large k):

- A set variable x^k is represented by a type variable x^k.
- Membership predicate Mem_k

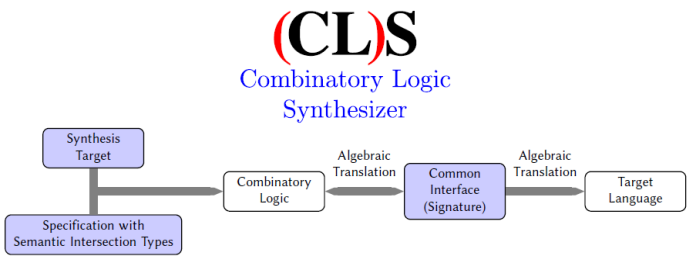
  $$Num_k(x^k) \rightarrow Num_{k+1}(y^{k+1}) \rightarrow In_k(x^k, y^{k+1}) \rightarrow Mem_k(x^k, y^{k+1})$$

  where $In_k(x^k, x^k \rightarrow 1)$ and $NotIn(x^k, x^k \rightarrow 0)$ are axioms.
- Use alternation to code quantifiers as usual (Urzyczyn 1997).

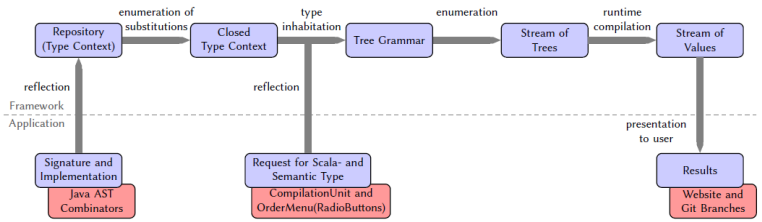
CLS Framework

Scala-integrated framework and experiments by Bessai (Dortmund), Düdder (Copenhagen), Dudenhefner (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]

Abstract Overview



Scala Implementation



Scala Implementation - Request Example

```
class Repository {
  val alpha = Variable("alpha")
  val kinding: Kinding =
    Kinding(alpha).addOption('DropDown).addOption('RadioButtons)
  @combinator object customerForm {
    def apply(title: String,
        logoLocation: URL,
        optionSelector: OptionSelection): CompilationUnit = {
      val form = Java(readFile("CustomerForm.java")).compilationUnit()
      addOptionSelection(form, optionSelector)
      addLogo(form, logoLocation)
      form
    }
    val semanticType: Type =
      'Title =>:
      'Location('Logo) =>:
      'ChoiceDialog(alpha) =>:
      'OrderMenu(alpha)
  }
  /* more combinators */
}
/* Request: */
val repository = new Repository
val Gamma = ReflectedRepository(repository, kinding = repository.kinding)
val results: Results =
  Results.add(Gamma.inhabit[CompilationUnit]('OrderMenu(Omega)))
```

CLS Framework

$$
\begin{aligned}
\Gamma = \{ \quad \textit{customerForm} &: (\texttt{String} \to \texttt{java.net.URL} \to \texttt{OptionSelection} \to \texttt{Form}) \\
\textit{dropDownSelector} &: (\texttt{java.net.URL} \to \texttt{OptionSelection}) \\
\textit{radioButtonSelector} &: (\texttt{java.net.URL} \to \texttt{OptionSelection}) \\
\textit{companyTitle} &: \texttt{String} \\
\textit{databaseLocation} &: \texttt{java.net.URL} \\
\textit{logoLocation} &: \texttt{java.net.URL} \\
\textit{alternateLogoLocation} &: \texttt{java.net.URL} \quad \}
\end{aligned}
$$

$$
\begin{aligned}
\Gamma = \{ \quad \textit{customerForm} &: (\texttt{String} \to \texttt{java.net.URL} \to \texttt{OptionSelection} \to \texttt{CompilationUnit}) \cap \\
&\quad (\textit{Title} \to \textit{Location}(\textit{Logo}) \to \textit{ChoiceDialog}(\alpha) \to \textit{OrderMenu}(\alpha)) \\
\textit{dropDownSelector} &: (\texttt{java.net.URL} \to \texttt{OptionSelection}) \cap \\
&\quad (\textit{Location}(\textit{Database}) \to \textit{ChoiceDialog}(\textit{DropDown})) \\
\textit{radioButtonSelector} &: (\texttt{java.net.URL} \to \texttt{OptionSelection}) \cap \\
&\quad (\textit{Location}(\textit{Database}) \to \textit{ChoiceDialog}(\textit{RadioButtons})) \\
\textit{companyTitle} &: \texttt{String} \cap \textit{Title} \\
\textit{databaseLocation} &: \texttt{java.net.URL} \cap \textit{Location}(\textit{Database}) \\
\textit{logoLocation} &: \texttt{java.net.URL} \cap \textit{Location}(\textit{Logo}) \\
\textit{alternateLogoLocation} &: \texttt{java.net.URL} \cap \textit{Location}(\textit{Logo}) \quad \}
\end{aligned}
$$

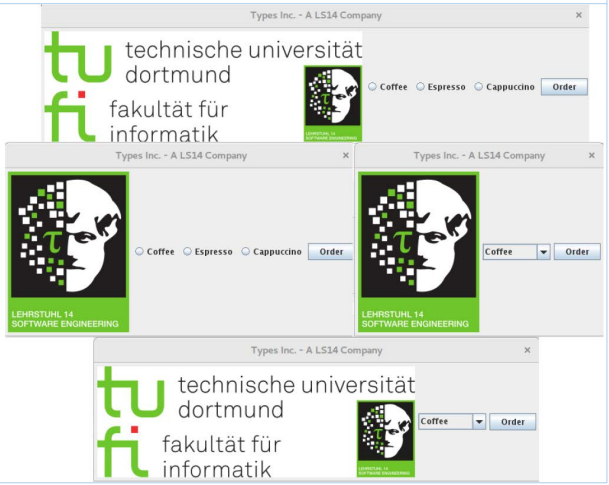
$$\Gamma \vdash ? : \texttt{CompilationUnit} \cap \textit{OrderMenu}(\omega)$$

$$\Gamma \vdash ? : \texttt{CompilationUnit} \cap \textit{OrderMenu}(\omega)$$

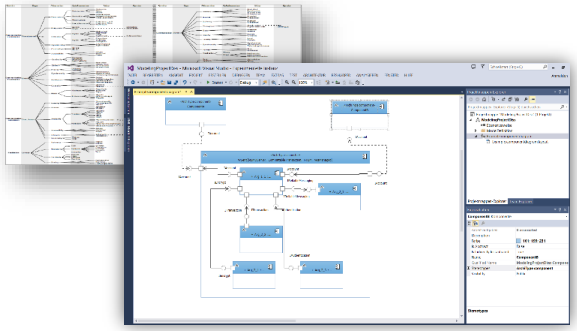
Inhbaitants:

$$
\begin{aligned}
\{ &\textit{customerForm}(\textit{companyTitle}, \textit{logoLocation}, \\
&\quad \textit{radioButtonSelector}(\textit{databaseLocation})), \\
&\textit{customerForm}(\textit{companyTitle}, \textit{alternateLogoLocation}, \\
&\quad \textit{radioButtonSelector}(\textit{databaseLocation})), \\
&\textit{customerForm}(\textit{companyTitle}, \textit{logoLocation}, \\
&\quad \textit{dropDownSelector}(\textit{databaseLocation})), \\
&\textit{customerForm}(\textit{companyTitle}, \textit{alternateLogoLocation}, \\
&\quad \textit{dropDownSelector}(\textit{databaseLocation}))\}
\end{aligned}
$$

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



CLS Framework – Experiments



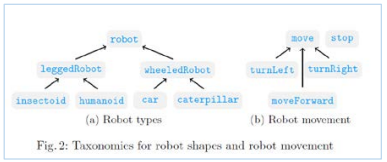
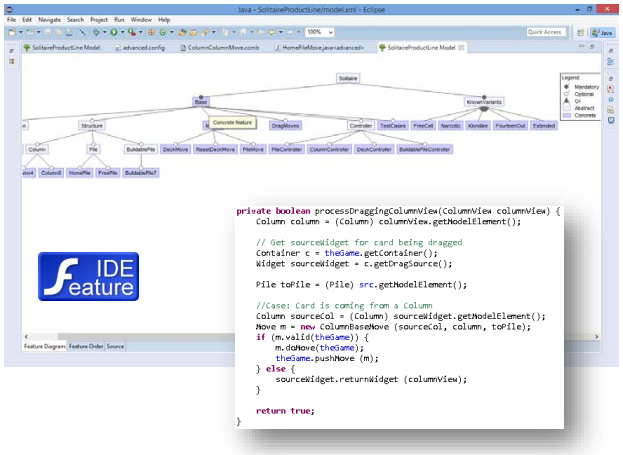


Fig. 2: Taxonomies for robot shapes and robot movement



1. $\texttt{setSensors}' : \Box(\texttt{subproc} \cap \texttt{twoLightSensors} \cap \texttt{stopsOnTouch} \cap \texttt{set})$
   a subprocess to set the light and touch sensors.
2. $\texttt{executeJob}' : \Box(\texttt{subproc} \cap \texttt{twoLightSensors} \cap \texttt{stopsOnTouch} \cap \texttt{car} \cap \texttt{followsLine}) \cap \texttt{jobProc}$
   a subprocess that makes the robot execute the job until aborted.
3. $\texttt{stop}' : \Box(\texttt{subproc} \cap \texttt{car} \cap \texttt{stop})$ a subprocess that makes a car robot stop.

$$\mathit{createRobotProgram}(\texttt{setSensors}', \texttt{executeJob}', \texttt{stop}') : \texttt{robotProgram} \cap \Box(\texttt{proc} \cap \texttt{twoLightSensors} \cap \texttt{stopsOnTouch} \cap \texttt{car} \cap \texttt{followsLine})$$

Inhabitation Given the repository $\mathcal{D}$ of typed PCs and typed CCs, asking the inhabitation question $\mathcal{D} \vdash ? : \Box(\texttt{proc} \cap \texttt{car} \cap \texttt{followsLine} \cap \texttt{twoLightSensors} \cap \texttt{stopsOnTouch}) \cap \texttt{robotProgram}$ automatically synthesizes the following applicative term:

$$\begin{aligned}
&\mathit{createRobotProgram}(\\
&\quad \mathit{setSensors}(\text{box } \texttt{set}_{\texttt{TwoLightsSensors}}, \mathit{taskToSubProc}(\text{box } \texttt{set}_{\texttt{TouchSensor}})),\\
&\quad \mathit{executeJob}(\mathit{taskToSubProc}(\text{box } \texttt{read}_{\texttt{TouchSensor}}),\\
&\qquad \text{box } \texttt{abortCondition}_{\texttt{TouchSensor}}, \text{box } \texttt{read}_{\texttt{TwoLightSensors}},\\
&\qquad \mathit{conditionalMove}(\text{box } \texttt{turnLeftCondition}_{\texttt{LineFollowerTwoLightSensors}},\\
&\qquad\quad \text{box } \texttt{turnRightCondition}_{\texttt{LineFollowerTwoLightSensors}}),\\
&\qquad\quad \text{box } \texttt{turnLeft}_{\texttt{Car}}, \text{box } \texttt{turnRight}_{\texttt{Car}}, \text{box } \texttt{moveForward}_{\texttt{Car}})),\\
&\quad \text{box } \texttt{stop}_{\texttt{Car}})
\end{aligned}$$

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

CLS – Research Challenges

INTERSECTION TYPE *SPECIFICATION*
fstproc ∩ car ∩ followsLine ∩ twoLightSensors ∩ stopsOnTouch ∩ robotProgram

Inhabitation algorithm for CL

(CL)S
Combinatory Logic Synthesizer

Component Repository in SCALA extension

Combinatory Meta-Program

	Minimal parentheses	Full parentheses	Tree diagram
1.	ISK	((IS)K)	
2.	I(SK)	(I(SK))	
3.	SKISKI	((((SK)I)(S)K)I)	
4.	S(KI)(SK)I	((S((KI)(SK))I)	
5.	S(K(I(S(KI))))	(S(K(I(S(KI)))))	

In the form with minimal parentheses, not every subsequence of symbols is well formed (i.e., has balanced implicit parentheses). In 1, notice that SK is not a balanced subterm, and does not correspond to a subtree in the tree diagram. If 2 arose in a derivation, we could not apply Rule 2 to the form KIS, because that is not a well-formed portion, nor could we apply Rule 1 to the form IS. Similarly in 4, (KI)(SK) and ((SK)I) are not well formed.

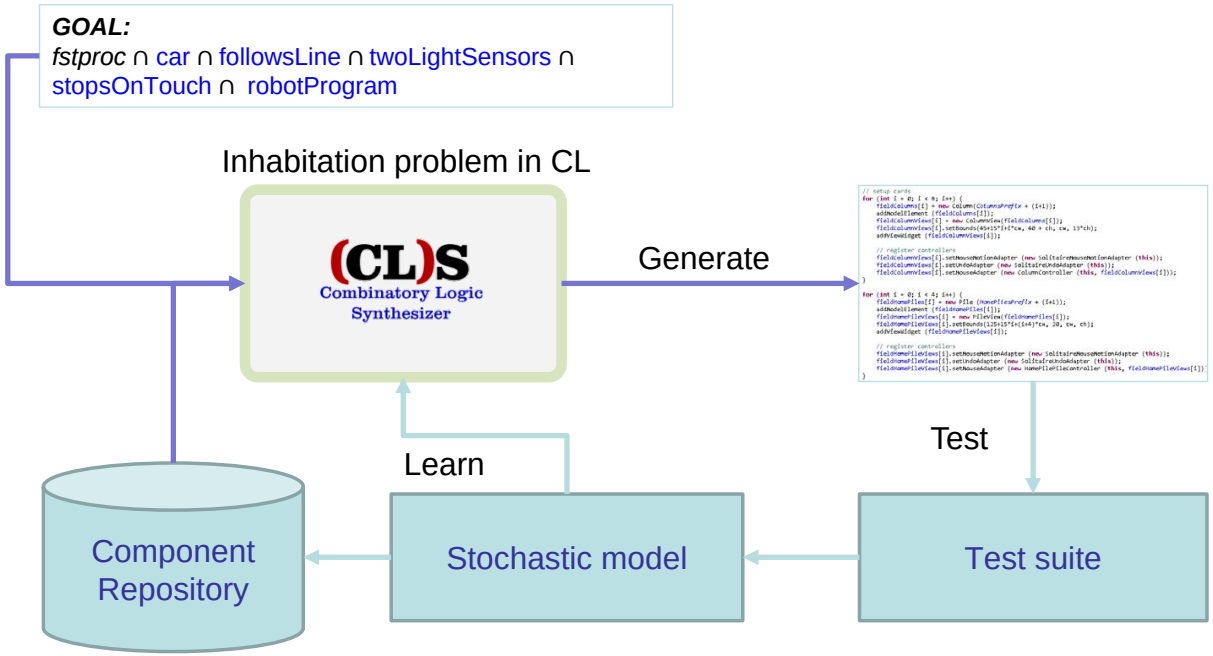
Output Program

```
// setup cards
for (int i = 0; i < 8; i++) {
    fieldColumns[i] = new Column(ColumnsPrefix + (i+1));
    addNodeElement (fieldColumns[i]);
    fieldColumnViews[i] = new ColumnView(fieldColumns[i]);
    fieldColumnViews[i].setBounds(45+15*i+(*cw, 40 + ch, cw, 13*ch);
    addViewWidget (fieldColumnViews[i]);

    // register controllers
    fieldColumnViews[i].setMouseMotionAdapter (new SolitaireMouseMotionAdapter (this));
    fieldColumnViews[i].setWindowAdapter (new SolitaireWindowAdapter (this));
    fieldColumnViews[i].setMouseAdapter (new ColumnController (this, fieldColumnViews[i]));
}

for (int i = 0; i < 4; i++) {
    fieldHomeFiles[i] = new File (HomeFilesPrefix + (i+1));
    addNodeElement (fieldHomeFiles[i]);
    fieldHomeFileViews[i] = new FileView(fieldHomeFiles[i]);
    fieldHomeFileViews[i].setBounds(125+15*i+(i+4)*cw, 20, cw, ch);
    addViewWidget (fieldHomeFileViews[i]);

    // register controllers
    fieldHomeFileViews[i].setMouseMotionAdapter (new SolitaireMouseMotionAdapter (this));
    fieldHomeFileViews[i].setWindowAdapter (new SolitaireWindowAdapter (this));
    fieldHomeFileViews[i].setMouseAdapter (new HomeFileFileController (this, fieldHomeFileViews[i]))
}
```

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 λ-calculus inhabitation: Frankle, Osera, Walker, and Zdancewic, *Example-directed synthesis: a type-theoretic interpretation*, POPL 2016 [FOWZ16]
  - Combinators already used in ML: Liang, Jordan and Klein, *Learning Programs: A Hierarchical Bayesian Approach* ML 2010 [LJK10]
- Integration with theorem proving



Inhabitation in λ-Calculus with Intersection Types

Inhabitation in λ-calculus with intersection types is undecidable

- P. Urzyczyn, *The Emptiness Problem for Intersection Types*, JSL 1999 [Urz99] via reduction from halting problems for queue automata using rank 4 intersections.

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

- P. Urzyczyn, *Inhabitation of Low-Rank Intersection Types*, TLCA 2009 [Urz09] (EXPTIME-hardness [Kus07])
- Proof techniques via *bus machines*, an alternating, expanding instruction device, also used to show EXPXPACE-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 λ-definability problem

- Undecidability of λ-definability: Loader 1993 [Loa01]
- S. Salvati, *Recognizability in the Simply Typed Lambda-Calculus*, WoLLIC 2009 [Sal09]
- Salvati, Manzonetto, Gehrke, Barendregt, *Urzyczyn and Loader are logically related*, ICALP 2012 [SMGB12]

Inhabitation in λ-Calculus with Intersection Types

Inhabitation in λ-calculus with intersection types is undecidable

- P. Urzyczyn, *The Emptiness Problem for Intersection Types*, JSL 1999 [Urz99] via reduction from halting problems for queue automata using rank 4 intersections.

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

- P. Urzyczyn, *Inhabitation of Low-Rank Intersection Types*, TLCA 2009 [Urz09] (EXPTIME-hardness [Kus07])
- Proof techniques via *bus machines*, an alternating, expanding instruction device, also used to show EXPXPACE-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 λ-definability problem

- Undecidability of λ-definability: Loader 1993 [Loa01]
- S. Salvati, *Recognizability in the Simply Typed Lambda-Calculus*, WoLLIC 2009 [Sal09]
- Salvati, Manzonetto, Gehrke, Barendregt, *Urzyczyn and Loader are logically related*, ICALP 2012 [SMGB12]

Inhabitation in λ-Calculus with Intersection Types

Inhabitation in λ-calculus with intersection types is undecidable

- P. Urzyczyn, *The Emptiness Problem for Intersection Types*, JSL 1999 [Urz99] via reduction from halting problems for queue automata using rank 4 intersections.

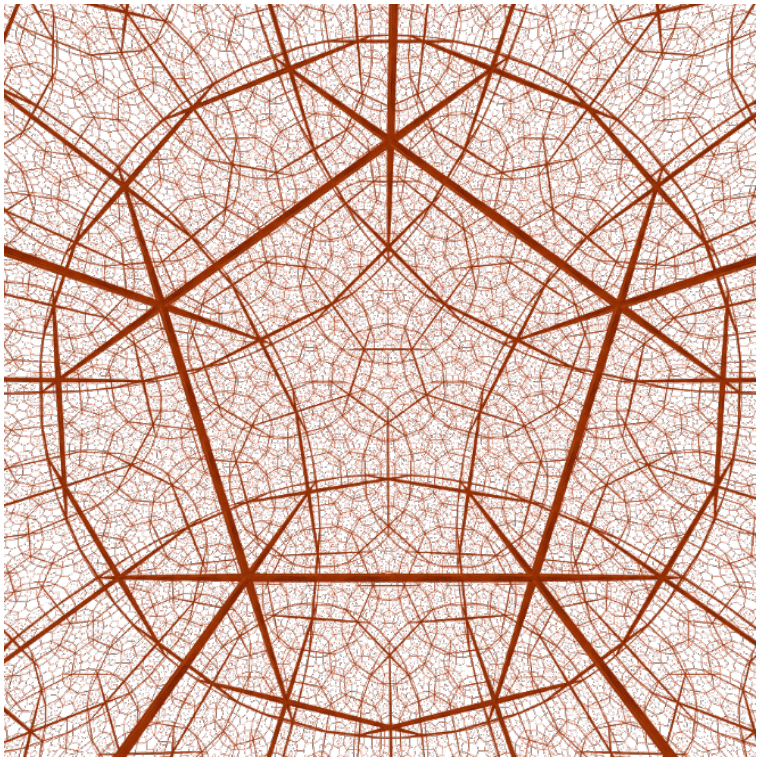
Rank 2-inhabitation is decidable and EXPSPACE-complete, and rank k-inhabitation is undecidable for all ranks $k > 2$

- P. Urzyczyn, *Inhabitation of Low-Rank Intersection Types*, TLCA 2009 [Urz09] (EXPTIME-hardness [Kus07])
- Proof techniques via *bus machines*, an alternating, expanding instruction device, also used to show EXPXPACE-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 λ-definability problem

- Undecidability of λ-definability: Loader 1993 [Loa01]
- S. Salvati, *Recognizability in the Simply Typed Lambda-Calculus*, WoLLIC 2009 [Sal09]
- Salvati, Manzonetto, Gehrke, Barendregt, *Urzyczyn and Loader are logically related*, ICALP 2012 [SMGB12]

Dimensional λ-Calculus



View of Order-4 dodecahedral honeycomb generated by software: http://geometrygames.org/CurvedSpaces Curved Spaces v1.9 Topology and Geometry Software, Jeff Weeks

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

Strict Intersection Type System

Definition (Strict Intersection Types)

$$A, B ::= a \mid \sigma \to A$$
$$\sigma, \tau ::= [A_1, \ldots, A_n] \qquad n \geq 1$$

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

$$\frac{1 \leq i \leq n}{\Gamma, x : [A_1, \ldots, A_n] \vdash_s x : [A_i]}\ (\text{Var}) \qquad \frac{\Gamma \vdash_s M : [A_i] \text{ for } i = 1 \ldots n}{\Gamma \vdash_s M : [A_i, \ldots, A_n]}\ (\cap\text{I})$$

$$\frac{\Gamma \vdash_s M : [\sigma \to A] \qquad \Gamma \vdash_s N : \sigma}{\Gamma \vdash_s M\,N : [A]}\ (\to\text{E}) \qquad \frac{\Gamma, x : \sigma \vdash_s M : [A]}{\Gamma \vdash_s \lambda x.M : [\sigma \to A]}\ (\to\text{I})$$

Set-Theoretic Elaboration System

Definition ($\Gamma \vdash \mathbf{P} : \sigma$)

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

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

$$\frac{\Gamma \vdash \mathbf{P} : [\sigma \to A] \qquad \Gamma \vdash \mathbf{Q} : \sigma}{\Gamma \vdash (\mathbf{P}\ \mathbf{Q})\langle [A] \rangle : [A]} \text{ ($\to$E)}$$

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

Intuition

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

Set-Theoretic Elaboration System

Definition ($\Gamma \vdash \mathbf{P} : \sigma$)

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

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

$$\frac{\Gamma \vdash \mathbf{P} : [\sigma \to A] \qquad \Gamma \vdash \mathbf{Q} : \sigma}{\Gamma \vdash (\mathbf{P}\ \mathbf{Q})\langle [A] \rangle : [A]} \text{ ($\to$E)}$$

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

Intuition

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

Norm

Definition ($\mathbf{P} \sqcup \mathbf{Q}$, defined for $\lceil \mathbf{P} \rceil \equiv \lceil \mathbf{Q} \rceil$)

$$
\begin{aligned}
x\langle S\rangle \sqcup x\langle S'\rangle &\equiv x\langle S \cup S'\rangle \\
(\lambda x.\mathbf{P})\langle S\rangle \sqcup (\lambda x.\mathbf{Q})\langle S'\rangle &\equiv (\lambda x.\mathbf{P} \sqcup \mathbf{Q})\langle S \cup S'\rangle \\
(\mathbf{PQ})\langle S\rangle \sqcup (\mathbf{P'Q'})\langle S'\rangle &\equiv ((\mathbf{P} \sqcup \mathbf{P'})(\mathbf{Q} \sqcup \mathbf{Q'}))\langle S \cup S'\rangle
\end{aligned}
$$

Definition (Norm $\|\bullet\|$)

$$
\begin{aligned}
\|x\langle S\rangle\| &= |S| \\
\|(\lambda x.\mathbf{P})\langle S\rangle\| &= \max\{\|\mathbf{P}\|, |S|\} \\
\|(\mathbf{PQ})\langle S\rangle\| &= \max\{\|\mathbf{P}\|, \|\mathbf{Q}\|, |S|\}
\end{aligned}
$$

Non-negativity	:	$\|\mathbf{P}\| > 0$
Subadditivity	:	$\|\mathbf{P} \sqcup \mathbf{Q}\| \leq \|\mathbf{P}\| + \|\mathbf{Q}\|$ for $\lceil \mathbf{P} \rceil \equiv \lceil \mathbf{Q} \rceil$

Norm

Definition ($\mathbf{P} \sqcup \mathbf{Q}$, defined for $\lceil \mathbf{P} \rceil \equiv \lceil \mathbf{Q} \rceil$)

$$
\begin{aligned}
x\langle S\rangle \sqcup x\langle S'\rangle &\equiv x\langle S \cup S'\rangle \\
(\lambda x.\mathbf{P})\langle S\rangle \sqcup (\lambda x.\mathbf{Q})\langle S'\rangle &\equiv (\lambda x.\mathbf{P} \sqcup \mathbf{Q})\langle S \cup S'\rangle \\
(\mathbf{PQ})\langle S\rangle \sqcup (\mathbf{P'Q'})\langle S'\rangle &\equiv ((\mathbf{P} \sqcup \mathbf{P'})(\mathbf{Q} \sqcup \mathbf{Q'}))\langle S \cup S'\rangle
\end{aligned}
$$

Definition (Norm $\|\bullet\|$)

$$
\begin{aligned}
\|x\langle S\rangle\| &= |S| \\
\|(\lambda x.\mathbf{P})\langle S\rangle\| &= \max\{\|\mathbf{P}\|, |S|\} \\
\|(\mathbf{PQ})\langle S\rangle\| &= \max\{\|\mathbf{P}\|, \|\mathbf{Q}\|, |S|\}
\end{aligned}
$$

Non-negativity	:	$\|\mathbf{P}\| > 0$
Subadditivity	:	$\|\mathbf{P} \sqcup \mathbf{Q}\| \leq \|\mathbf{P}\| + \|\mathbf{Q}\|$ for $\lceil \mathbf{P} \rceil \equiv \lceil \mathbf{Q} \rceil$

Intersection Type Calculus of Bounded Dimension

Definition

Write $\Gamma \vdash M \longmapsto \mathbf{P} : \sigma$ iff $\Gamma \vdash \mathbf{P} : \sigma$ with $M \equiv \lceil \mathbf{P} \rceil$.

- Clearly, $\Gamma \vdash_s M : \sigma$ iff $\exists \mathbf{P}.\ \Gamma \vdash M \longmapsto \mathbf{P} : \sigma$

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

$\Gamma \vdash_n M : \sigma \qquad \text{iff} \qquad \exists \mathbf{P}.\ \Gamma \vdash M \longmapsto \mathbf{P} : \sigma$ with $\|\mathbf{P}\| \leq n$

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 Γ and σ is

$$\dim_\Gamma^\sigma = \min\{n \mid \Gamma \vdash_n M : \sigma\}$$

Subject Reduction in Bounded Dimension

Terms can be elaborated in non-increasing norm under β-reduction:

Theorem (Subject Reduction for $\lambda_n^{[\cap]}$)

If $\Gamma \vdash M \longmapsto \mathbf{P} : \tau$ and $M \to_\beta M'$,
then there exists $\mathbf{R}$ with $\|\mathbf{R}\| \leq \|\mathbf{P}\|$ such that

$$\Gamma \vdash M' \longmapsto \mathbf{R} : \tau$$

Consequences:

- Each dimensional fragment $\lambda_n^{[\cap]}$ is a meaningful type system.
- Inhabitation in bounded dimension for $\lambda_n^{[\cap]}$ can be limited to search for normal forms.

Inhabitation in Bounded Set Theoretic Dimension

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

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

Theorem

The inhabitation problem for $\lambda^{[\cap]}$ 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 $\lambda^{[\cap]}$)

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

Theorem

The inhabitation problem for $\lambda^{[\cap]}$ 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).

Non-idempotence

The inhabitation problem for non-idempotent intersection types

Antonio Bucciarelli[1], Delia Kesner[1], and Simona Ronchi Della Rocca[2]

[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.

[BKRDR14]

Multiset Elaboration System

Definition

Treat types $[A_1, \ldots, A_n]$ and sets S as *multisets* $\mathfrak{s}$ and let $\biguplus$ denote multiset union.

Definition ($\Delta \Vdash \mathbb{P} : \mathfrak{s}$)

$$\frac{1 \leq i \leq n}{\Delta, x : [A_1, \ldots, A_n] \Vdash x\langle [A_i] \rangle : [A_i]} \text{ (Var)}$$

$$\frac{\Delta, x : \mathfrak{s} \Vdash \mathbb{P} : [A]}{\Delta \Vdash (\lambda x.\mathbb{P})\langle [\mathfrak{s} \to A] \rangle : [\mathfrak{s} \to A]} \text{ ($\to$I)}$$

$$\frac{\Delta \Vdash \mathbb{P} : [\mathfrak{s} \to A] \qquad \Delta \Vdash \mathbb{Q} : \mathfrak{s}}{\Delta \Vdash (\mathbb{P}\ \mathbb{Q})\langle [A] \rangle : [A]} \text{ ($\to$E)}$$

$$\frac{\Delta \Vdash \mathbb{P}_i : [A_i] \text{ for } i = 1 \ldots n \qquad (\star)}{\Delta \Vdash \biguplus_{i=1}^{n} \mathbb{P}_i : [A_1, \ldots, A_n]} \text{ ($\cap$I)}$$

($\star$) For each $x\langle \mathfrak{s} \rangle$ in $\biguplus_{i=1}^{n} \mathbb{P}_i$: if x free in M, then $\mathfrak{s} \mathrel{\underline{\in}} \Delta(x)$.

Inhabitation in Bounded Multiset Dimension

Definition

$\Gamma \Vdash_n M : \sigma$ iff
$\exists \Delta, \mathbb{P}, \mathfrak{s}.\ (\Delta \Vdash M \Longmapsto \mathbb{P} : \mathfrak{s}$ with $\Gamma = \Delta^\circ$ and $\sigma = \mathfrak{s}^\circ$ and $\|\mathbb{P}\| \leq n)$
where $(\_)^\circ$ collapses multisets to underlying sets.

Problem ($\Gamma \Vdash_n ? : \sigma$)

Given Γ, σ and $n > 0$:
is there a term M such that $\Gamma \Vdash_n M : \sigma$?

Theorem

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

Corollary

For each fixed n inhabitation in multiset dimension n is PSPACE-complete.

Dimensional Analysis of Rank 2 Typings

Proposition

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

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.

Multiset Norm, Treewidth, and Bus Machines [Urz09]

$$\Gamma = \{x : (e \cap f) \to g, y : (((a \to a) \cap (b \to b)) \to e) \cap (((b \to b) \cap (c \to c)) \to f)\}$$

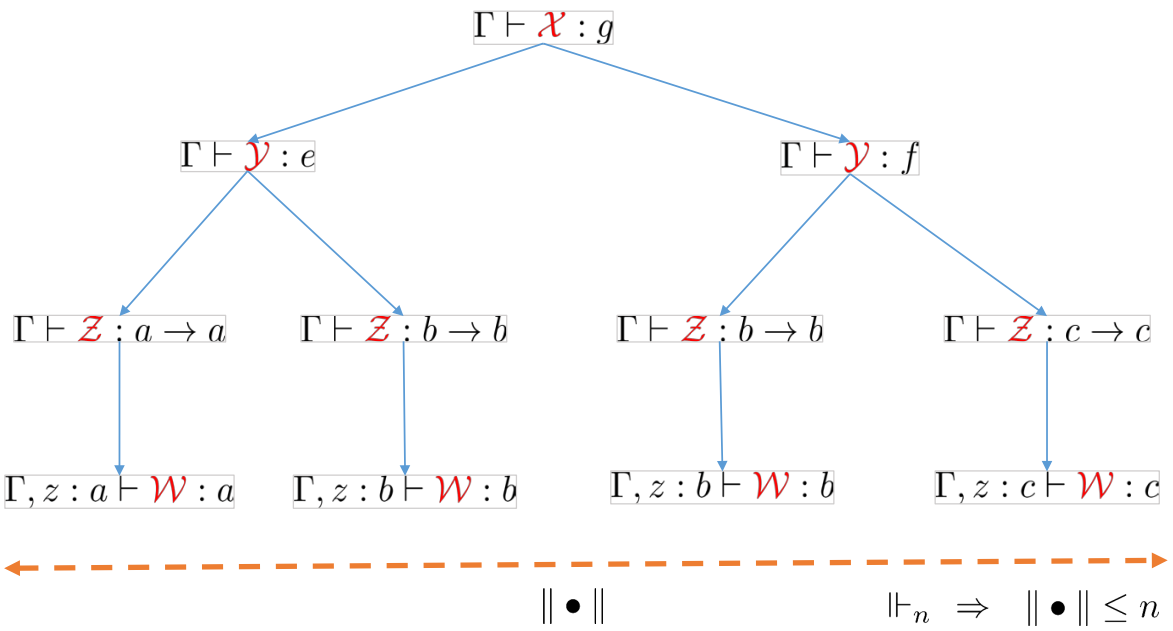
$$\Gamma \Vdash_n ? : g$$

$\mathcal{X} \equiv x(y(\lambda z.z))$

$\mathcal{Y} \equiv y(\lambda z.z)$

$\mathcal{Z} \equiv \lambda z.z$

$\mathcal{W} \equiv z$



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

Bounded Width Theorem

Definition (Width)

$$\begin{array}{lcl} [\![a]\!] & = & 1 \\ [\![\sigma \to A]\!] & = & \max\{[\![\sigma]\!], [\![A]\!]\} \\ [\![A_1 \cap \cdots \cap A_m]\!] & = & \max\{m, [\![A_1]\!], \ldots, [\![A_m]\!]\} \end{array}$$

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

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

Let a derivation $\mathcal{D} \triangleright \Gamma \vdash M \longmapsto \mathbf{P} : \sigma$ be given with $\|\mathbf{P}\| \leq d$. Then there exists a derivation $\mathcal{D}' \triangleright \Gamma' \vdash M \longmapsto \mathbf{P}' : \sigma'$ such that $[\![\mathcal{D}']\!] \leq d \cdot |M|$ and $\|\mathbf{P}'\| = \|\mathbf{P}\|$.

Proof.

By filtration with $\mathscr{F}_{T_{\mathbf{P}}}$ and using $[\![\mathscr{F}_{T_{\mathbf{P}}}(\mathcal{D})]\!] \leq |T_{\mathbf{P}}|$ together with

$$|T_{\mathbf{P}}| \leq \|\mathbf{P}\| \cdot |M| \leq d \cdot |M|$$

□

Typability in Bounded Dimension

Problem (Typability in bounded set-theoretic dimension)

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

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

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]

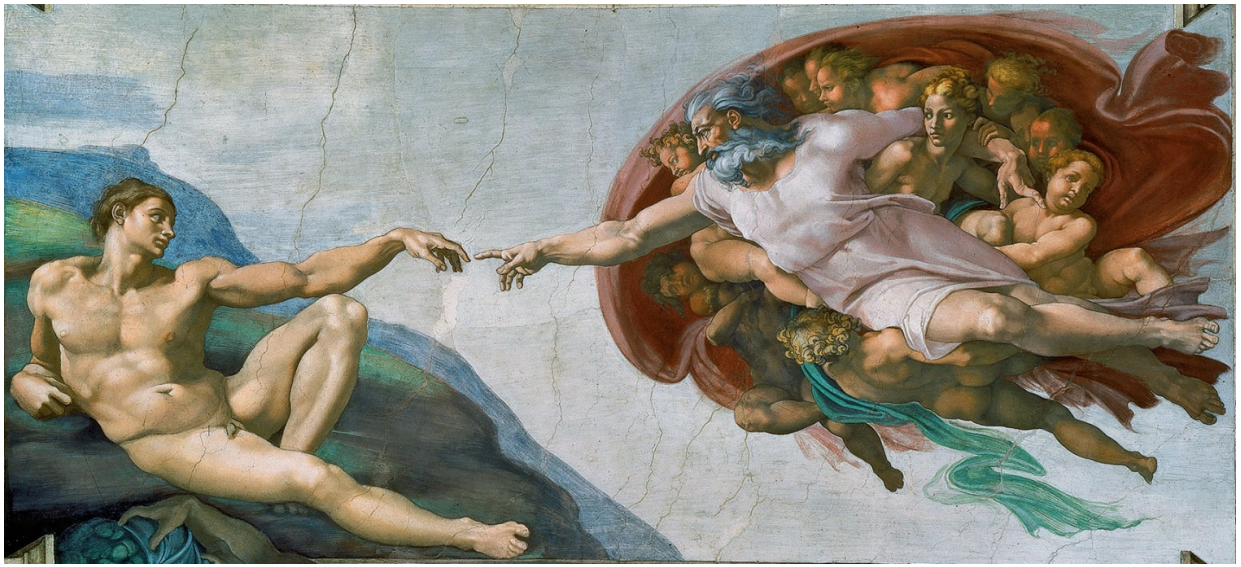
Dimensional Calculus – Research Challenges

- 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 β-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 ...

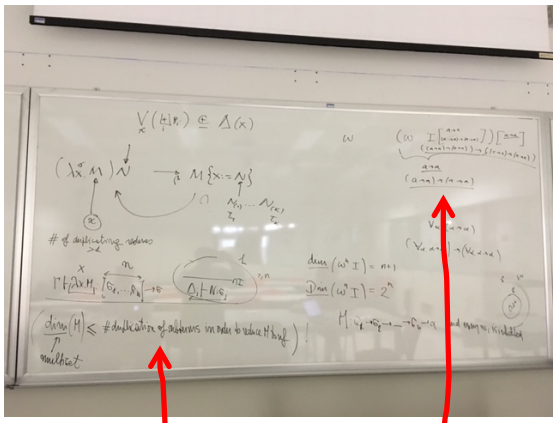
... and the work of the Torino group continues to inspire new and interesting problems and to enable new and unforeseen applications.



Thanks

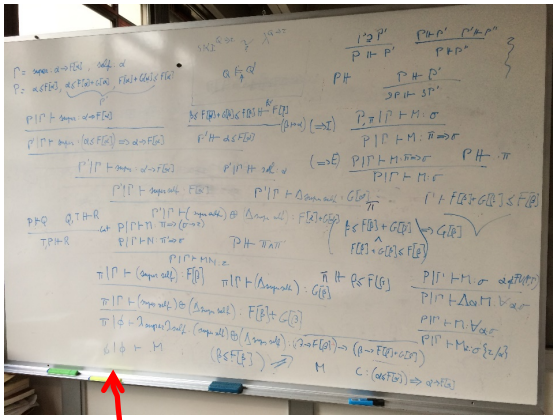
... for all the inspiration and hospitality!

Tzu-Chun



Simona

Andrej



Ugo

Thanks

ET IN ARCADIA EGO

S. Anantharaman, P. Narendran, and M. Rusinowitch.
Acid-unification is NEXPTIME-decidable.
In *Mathematical Foundations of Computer Science 2003*, pages 169–178. Springer, 2003.

S. Anantharaman, P. Narendran, and M. Rusinowitch.
Unification Modulo ACUI Plus Distributivity Axioms.
Journal of Automated Reasoning, 33(1):1–28, 2004.

H. P. Barendregt, M. Coppo, and M. Dezani-Ciancaglini.
A Filter Lambda Model and the Completeness of Type Assignment.
Journal of Symbolic Logic, 48(4):931–940, 1983.

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

In Tiziana Margaria and Bernhard Steffen, editors, *ISoLA'14*, volume 8802, pages 26–40, 2014.

Jan Bessai, Andrej Dudenhefner, Boris Duedder, Ugo De'Liguoro, Tzu-Chun Chen, and Jakob Rehof.
Mixin Composition synthesis Based on Intersection Types.
In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 38. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

Jan Bessai, Andrej Dudenhefner, Boris Düdder, Moritz Martens, and Jakob Rehof.
Combinatory process synthesis.
In *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques - 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part I*, pages 266–281, 2016.

Jan Bessai, Andrej Dudenhefner, Boris Düdder, Moritz Martens, and Jakob Rehof.
Combinatory Process Synthesis.
In *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques - 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part I*, pages 266–281, 2016.

Jan Bessai, Boris Düdder, Andrej Dudenhefner, and Jakob Rehof.
Rank 3 Inhabitation of Intersection Types Revisited .
In *Proceedings TYPES 2016, 22nd International Conference on Types for Proofs and Programs, Novi Sad, May 2016*, 2016.

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

In C. Braga and P.C. Olveczky, editors, *Proceedings of the 12th International Conference on Formal Aspects of Component Software (FACS'15)*, volume 9539, pages 1–18. Springer, 2016.

H. P. Barendregt, W. Dekkers, and R. Statman.
Lambda Calculus with Types.
Perspectives in Logic, Cambridge University Press, 2013.

Antonio Bucciareli, Delia Kesner, and Simona Ronchi Della Rocca.
The inhabitation problem for non-idempotent intersection types.
In *TCS 2014, Proceedings of 8th IFIP TC 1/WG 2.2 International Conference on Theoretical Computer Science, Rome, Italy, September 1-3, 2014*. Springer, 2014.

Marcin Benke, Aleksy Schubert, and Daria Walukiewicz-Chrzaszcz.

Synthesis of Functional Programs with Help of First-Order Intuitionistic Logic.
In Delia Kesner and Brigitte Pientka, editors, *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016)*, volume 52 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Mario Coppo, Mariangiola Dezani-Ciancaglini, and Betti Venneri.
Principal Type Schemes and Lambda-Calculus Semantics, pages 480–490.
Accademic Press, London, 1980.
Ed.: R. Hindley and J. Seldin.

Mario Coppo and Paola Giannini.

Principal types and unification for a simple intersection type system.
Information and Computation, 122(1):70–96, 1995.

M. Dezani-Ciancaglini and R. Hindley.
Intersection Types for Combinatory Logic.
Theoretical Computer Science, 100(2):303–324, 1992.

Boris Düdder, Oliver Garbe, Moritz Martens, Jakob Rehof, and Paweł Urzyczyn.
Using Inhabitation in Bounded Combinatory Logic with Intersection Types for Composition Synthesis.
In Stéphane Graham-Lengrand and Luca Paolini, editors, *Proceedings Sixth Workshop on Intersection Types and Related Systems, ITRS 2012, Dubrovnik, Croatia, 29th June 2012.*, volume 121 of *EPTCS*, pages 18–34, 2012.

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

In *Proceedings of TLCA'13*, volume 4(6) of *LNCS*. http://dx.doi.org/10.4230/DagRep.4.6.29, 2013.

Boris Düdder, Moritz Martens, and Jakob Rehof.
Staged Composition Synthesis.
In *ESOP 2014, Proceedings of European Symposium on Programming, Grenoble, France 2014*, volume 8410 of *LNCS*, pages 67–86. Springer, 2014.

Andrej Dudenhefner, Moritz Martens, and Jakob Rehof.
The Intersection Type Unification Problem.
In Delia Kesner and Brigitte Pientka, editors, *1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016, June 22-26, 2016, Porto, Portugal*, volume 52 of *LIPIcs*, pages 19:1–19:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

A. Dudenhefner, M. Martens, and J. Rehof.
The Algebraic Intersection Type Unification Problem.

Logical Methods in Computer Science, 13(3), 2017.

Boris Düdder, Moritz Martens, Jakob Rehof, and Paweł Urzyczyn.
Bounded Combinatory Logic.
In *CSL 2012, Proceedings of Computer Science Logic*, volume 16 of *LIPIcs*, pages 243–258. Schloss Dagstuhl, 2012.

Andrej Dudenhefner and Jakob Rehof.
Intersection Type Calculi of Bounded Dimension.
In *POPL 2017, Proceedings of the 44th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Paris, France, January*, 2017.

Andrej Dudenhefner and Jakob Rehof.
Typability in Bounded Dimension.
In *LICS 2017, Proceedings of the 32nd ACM/IEEE Symposium on Logic in Computer Science, Reykjavik, Iceland, June*, 2017.

Boris Düdder, Jakob Rehof, and George T. Heineman.
Synthesizing Type-Safe Compositions in Feature Oriented Software Designs Using Staged Composition.
In *Proceedings of the 19th International Conference on Software Product Line, SPLC 2015, Nashville, TN, USA, July 20-24, 2015*, pages 398–401, 2015.

Boris Düdder.
Automatic Synthesis of Component & Connector-Software Architectures with Bounded Combinatory Logic.
PhD thesis, Technische Universität Dortmund, Fakultät für Informatik, Dortmund, 2014, 2014.

Jonathan Frankle, Peter-Michael Osera, David Walker, and Steve Zdancewic.
Example-Directed Synthesis: A Type-Theoretic Interpretation.
In *POPL'16*, pages 802–815. ACM, 2016.

T. Freeman and F. Pfenning.

Refinement Types for ML.
In *Proceedings of PLDI'91*, pages 268–277. ACM, 1991.

George T. Heineman, Jan Bessai, Boris Düdder, and Jakob Rehof.
A long and winding road towards modular synthesis.
In *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques - 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part I*, pages 303–317, 2016.

George T. Heineman, Jan Bessai, Boris Düdder, and Jakob Rehof.
A long and winding road towards modular synthesis.
In *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques - 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part I*, pages 303–317, 2016.

George T. Heineman, Armend Hoxha, Boris Düdder, and Jakob Rehof.
Towards Migrating Object-Oriented Frameworks to Enable Synthesis of Product Line Members.
In *Proceedings of the 19th International Conference on Software Product Line, SPLC 2015, Nashville, TN, USA, July 20-24, 2015*, pages 56–60, 2015.

Christian Haack, Brian Howard, Allen Stoughton, and Joe B. Wells.
Fully Automatic Adaptation of Software Components Based on Semantic Specifications.
In *AMAST*, volume 2422 of *LNCS*, pages 83–98. Springer, 2002.

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).
In Didier Rémi and Peter Lee, editors, *Proceedings of the fourth ACM SIGPLAN International Conference on Functional Programming (ICFP '99), Paris, France, September 27-29, 1999.*, pages 90–101. ACM, 1999.

T. Kurata and M. Takahashi.
Decidable properties of intersection type systems.
In *TLCA 1995, Proceedings of Typed Lambda Calculus and Applications*, volume 902 of *LNCS*, pages 297–311. Springer, 1995.

Dariusz Kusmierek.
The Inhabitation Problem for Rank Two Intersection Types.
In *TLCA 2007, Proceedings of Typed Lambda Calculus and Applications*, volume 4583 of *LNCS*, pages 240–254. Springer, 2007.

Percy Liang, Michael A. Jordan, and Dan Klein.
Learning Programs: A Hierarchical Bayesian Approach.
In *Proceedings of the 26th International Conference on Machine Learning Haifa, Israel*, 2010.

Ralph Loader.
The undecidability of lambda definability.
Logic, Meaning and Computation: Essays in Memory of Alonzo Church, pages 331–342, 2001.

Samuel Linial and Emil L. Post.
Recursive Unsolvability of the Deducibility, Tarski's Completeness and Independence of Axioms Problems of Propositional Calculus.
Bulletin of the American Mathematical Society, 55:50, 1949.

Simona Ronchi Della Rocca.

Principal Type Scheme and Unification for Intersection Type Discipline.
Theor. Comput. Sci., 59:181–209, 1988.

Jakob Rehof.
Towards Combinatory Logic Synthesis.
In *BEAT 2013, 1st International Workshop on Behavioural Types*. ACM, 2013.

Jakob Rehof and Paweł Urzyczyn.
Finite Combinatory Logic with Intersection Types.
In *TLCA 2011, Proceedings of Typed Lambda Calculus and Applications*, volume 6690 of *LNCS*, pages 169–183. Springer, 2011.

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*

the Occasion of His 60th Birthday, volume 7230 of *LNCS*, pages 256–270. Springer, 2012.

S. Salvati.
Recognizability in the Simply Typed Lambda-Calculus.
In H. Ono, M. Kanazawa, and R. J. G. B. de Queiroz, editors, *WoLLIC 2009, Proceedings of Workshop on Logic, Language, Information and Computation*, volume 5514 of *LNCS*, pages 48–60. Springer, 2009.

S. Salvati, G. Manzonetto, M. Gehrke, and H. P. Barendregt.
Urzyczyn and Loader are logically related.
In *ICALP 2012, Proceedings of Automata, Languages, and Programming - 39th International Colloquium*, volume 7392 of *LNCS*, pages 364–376. Springer, 2012.

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

Theoretical Computer Science, 9:67–72, 1979.

R. Statman.
A finite model property for intersection types.
In *Proceedings Seventh Workshop on Intersection Types and Related Systems, ITRS 2014, Vienna, Austria, 18 July 2014.*, pages 1–9, 2015.

P. Urzyczyn.
The Emptiness Problem for Intersection Types.
Journal of Symbolic Logic, 64(3):1195–1215, 1999.

P. Urzyczyn.
Inhabitation of Low-Rank Intersection Types.
In *TLCA 2009, Proceedings of Typed Lambda Calculus and Applications*, volume 5608 of *LNCS*, pages 356–370. Springer, 2009.

Steffen van Bakel.
Strict Intersection Types for the Lambda Calculus.

ACM Computing Surveys, 43(3), 2011.

Joe B. Wells and Boris Yakobowski.
Graph-Based Proof Counting and Enumeration with Applications for Program Fragment Synthesis.
In *LOPSTR 2004*, volume 3573 of *LNCS*, pages 262–277. Springer, 2005.