Acknowledgements

- My students in Dortmund (former and present) including Jan Bessai, Boris Düdder (now Copenhagen), Andrej Dudenhofner, Moritz Martens (now in industry)
- Collaborators, including Paweł Urzyczyn (Warsaw), George Heineman (WPI Boston), Ugo de’Liguoro (Torino)
- Colleagues, including Mariangiola Dezani, Simona Ronchi Della Rocca, Mario Coppo and the Torino $\lambda$-calculus group, Roger Hindley (Swansey), Aleksy Schubert and Warsaw group
- TYPES (Warsaw 2010, Bergen 2011, Novi Sad 2016)
Intersection Types and Decision Problems

- Intersection types [CDC80; CDCV81; BCDC83] allow terms to be assigned multiple types \( \Gamma \vdash M : \tau_1 \cap \ldots \cap \tau_n \)

- Due to their enormous expressive power, the classical decision problems, typability and inhabitation, are undecidable

- Compare with simple types:
  - Typability is in linear time (unification)
  - Inhabitation (provability in minimal intuitionistic logic) is \( \text{PSPACE}\)-complete [Sta79]
Intersection Types

Definition ([CDCV80],[BCDC83], …, [BDS13])

Types $\mathbb{T} : \sigma, \tau ::= a \mid \sigma \rightarrow \tau \mid \sigma \cap \tau$, Atoms $a \in \mathbb{A}$

\[
\frac{}{\Gamma, x : \tau \vdash x : \tau} \quad (\text{Var}) \quad \frac{\Gamma, x : \sigma \vdash M : A}{\Gamma \vdash \lambda x. M : \sigma \rightarrow A} \quad (\rightarrow \text{I})
\]

\[
\frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \quad (\rightarrow E)
\]

\[
\frac{\Gamma \vdash M : \tau_1 \quad \Gamma \vdash M : \tau_2}{\Gamma \vdash M : \tau_1 \cap \tau_2} \quad (\cap \text{I}) \quad \frac{\Gamma \vdash M : \tau_1 \cap \tau_2}{\Gamma \vdash M : \tau_i} \quad (\cap \text{E})
\]
Bounding principles

- We are concerned with understanding the borderline between decidability and undecidability for these decision problems.
- Motivations can be practical, purely theoretical, or (preferably) both.
Bounding principles

- We are concerned with understanding the borderline between decidability and undecidability for these decision problems.
- Motivations can be practical, purely theoretical, or (preferably) both.
- We are especially interested in “generic” bounding principles such that:
  - The rules (logic) of the system remain(s) intact.
  - Interesting levels of expressiveness are retained (depending on bound), up to complete recovery, e.g., \( S = \bigcup_k S_k \).
  - Decidability, complexity, algorithms “pivot” around the bound.
  - New (fine) structure of the decision problems is exposed.
Some examples ... a very non-exhaustive list ...

- **TIME**(\(k\)):
  - In many cases not very “revealing”
- **SPACE**(\(k\))
  - regular languages
- **TIME**(\(f(N)\)), **SPACE**(\(f(N)\))
  - complexity theory
- Quantifier alternation
  - Logical hierarchies
- Bounded depth Frege systems
  - Proof complexity
- Transfinite induction up to \(\alpha\)
  - Gentzen \(\epsilon_0\), ordinal analysis

...
In $\lambda$-Calculus Type Systems and This Talk

Pervasive principles

- Functional order and rank of a logical operator (after Leivant [Lei83])

For intersection types:

- **Refinement**
  - After Freeman and Pfenning, PLDI 1991 [FP91]

- **Semantic types**
  - Bounded combinatory logic, TLCA 2011, CSL 2012 ff. [RU11; D"ud+12]

- **Non-idempotence**
  - Bucciareli, Kesner, Ronchi Della Rocca, TCS 2014 [BKRDR14]

- **Norm and dimension**
  - D&R, POPL 2017, LICS 2017 [DR17a; DR17b]
Rank-bound wrt. a logical operator (after Leivant [Lei83]):

- **Rank** = maximal order (nesting depth to the left of →) at which the operator can appear.
- Example: system F (operator ∀)
- Example: intersection types (operator ∩)

**Definition (Intersection type rank)**

\[
\begin{align*}
\text{rank}(\tau) &= 0 \text{ if } \tau \text{ is a simple type} \\
\text{rank}(\sigma \rightarrow \tau) &= \max(\text{rank}(\sigma) + 1, \text{rank}(\tau)) \\
\text{rank}(\sigma \cap \tau) &= \max(1, \text{rank}(\sigma), \text{rank}(\tau))
\end{align*}
\]
Typability and Inhabitation

Given a (standard) type system $\Gamma \vdash M : \tau$ we consider decision problems

- **Typability.** Given $M$, does there exist $\Gamma$ and $\tau$ such that $\Gamma \vdash M : \tau$?
- **Inhabitation.** Given $\Gamma$ and $\tau$, does there exist $M$ such that $\Gamma \vdash M : \tau$?
Typability and Inhabitation

Given a (standard) type system $\Gamma \vdash M : \tau$ we consider decision problems

- **Typability.** Given $M$, does there exist $\Gamma$ and $\tau$ such that $\Gamma \vdash M : \tau$?
- **Inhabitation.** Given $\Gamma$ and $\tau$, does there exist $M$ such that $\Gamma \vdash M : \tau$?

Assuming the standard property (for $\Gamma = \{x_i : \tau_i\}_{i=1\ldots n}$)

$$\Gamma \vdash M : \tau \iff \vdash \lambda x_1 \ldots \lambda x_n. M : \tau_1 \to \cdots \to \tau_n \to \tau$$

we can consider rank $k$-bounded versions of the problems by asking for $\vdash M : \sigma$, where $\sigma$ is rank $\leq k$. 

# Typability and Inhabitation (in rank $k$)

<table>
<thead>
<tr>
<th></th>
<th>Typability</th>
<th>Inhabitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{F}$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td></td>
<td>J. Wells, LICS 1994</td>
<td>M.H. Löb, JSL 1976</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cf. also P. Urzyczyn, TLCA 1997</td>
</tr>
<tr>
<td>$\mathbb{F}_k$</td>
<td>$k = 2$: decidable $k &gt; 2$: $\infty$</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>A. Kfoury &amp; J. Wells, LFP 1994</td>
<td></td>
</tr>
<tr>
<td>$\lambda \sqcup$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td></td>
<td>Coppo, Dezani, Venneri, ZmLGM 1981</td>
<td>P. Urzyczyn, JSL 1999</td>
</tr>
<tr>
<td></td>
<td>$(k - 1)$-EXPTIME complete</td>
<td>$k = 2$: EXPSPACE complete</td>
</tr>
<tr>
<td></td>
<td>A. Kfoury &amp; J. Wells, POPL 1999</td>
<td>$k &gt; 2$: $\infty$</td>
</tr>
<tr>
<td></td>
<td>Kfoury, Mairson, Turbak, Wells, ICFP 1999</td>
<td>P. Urzyczyn, TLCA 2009</td>
</tr>
</tbody>
</table>
Inhabitation is undecidable

- See also P. Urzyczyn, *Inhabitation in typed lambda-calculi (A syntactic approach)*, TLCA 1997 [Urz97]

Typability is undecidable

- J. Wells, *Typability and Type-Checking in the Second-Order lambda-Calculus are Equivalent and Undecidable*, LICS 1994 [Wel94]

Typability is decidable for rank 2 and undecidable for all ranks $k > 2$

Intersection Types: Typability

- **Typability is undecidable**
  - Follows from characterization of normalization and solvability properties
    Coppo, Dezani, Venneri [CDCV81], Pottinger [Pot80], see also Ghilezan [Ghi96] and Barendregt, Dekkers, Statman [BDS13]

- **Rank-bounded fragments**
  - Decidable in all bounded-rank fragments: A. Kfoury and J. Wells, *Principality and Decidable Type Inference for Finite-Rank Intersection Types*, POPL 1999 [KW99]
  - Complexity exponential in rank: Kfoury, Mairson, Turbak and Wells, *Relating Typability and Expressiveness in Finite-Rank Intersection Type Systems*, ICFP 1999 [Kfo+99]
Intersection Types: Inhabitation

- Inhabitation is undecidable
  - Related to the $\lambda$-definability problem
    - Salvati, Manzonetto, Gehrke and Barendregt, *Urzyczyn and Loader are logically related*, ICALP 2012 [Sal+12]
    - Undecidability of $\lambda$-definability: Loader 1993 [Loa01]
Intersection Types: Rank-Bounded Inhabitation

- Rank 2-inhabitation is decidable
  - Equivalent to rank 2-polymorphism: H. Yokouchi, *Embedding a Second Order Type System into an Intersection Type System*, I&C 1995 [Yok95]
  - $\text{EXPTIME}$-hard: D. Kuśmierek, *The Inhabitation Problem for Rank Two Intersection Types*, TLCA 2007 [Kus07]
- Rank 2-inhabitation is $\text{EXPSPACE}$-complete and rank $k$-inhabitation is undecidable for all ranks $k > 2$
  - P. Urzyczyn, *Inhabitation of Low-Rank Intersection Types*, TLCA 2009 [Urz09]
  - Proof techniques via *bus machines*, an alternating, expanding instruction device, also used to show $\text{EXPSPACE}$-completeness of inhabitation with explicit intersection [RU12]
Combinatory Logic Synthesis (CLS)
A type-theoretic approach to component-oriented synthesis

Component-oriented Synthesis
Synthesis relative to library (repository) of components

Combinatory Logic Synthesis (CLS)
Libraries need classification systems to enable retrieval and composition

(∀ x; ∀ y; ∃ z; (p(x, y) ∧ p(y, z) → p(x, z)) ∧
∀ x; ∀ y; (p(x, y) → ¬p(y, x)) ∧
∃ x; (p(a, x) ∧ p(x, b))
) → ¬p(b, a)

Bottom-up specification
Hoare logic

Classification
Taxonomy
Types

Marichactomorpha (liveworts)
Anthocerophyta (hornworts)
Bryophyta (mosses)
Hornworts
Aglrophnoton major
Physneophyta
Lycopeplia (lycophytes)
Euphylytphytae (billyum)
Psilophyton dwarai
Cladophytoidea
Equireteplia (hornworts, sphenophytes)
Fikus (foms)
Porista varia
Anurophyta
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 undeinfitiert 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:

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


Mathematische Annalen. 92.
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. [Haa+02; WY05]) for specification
- Beyond purely functional composition via meta-programming – compose a meta-program which, when executed, computes (say) a Java program
Combinatory Logic Synthesis (CLS)
This idea was developed in a series of papers, including:

- **Finite Combinatory Logic with Intersection Types**, TLCA 2011 [RU11]
  ▶ First mention of CL with intersection types as semantic specifications for synthesis via inhabitation

- **Bounded Combinatory Logic**, CSL 2012 [Düd+12]
  ▶ Complexity hierarchy for relativized inhabitation with CL

- **Using Inhabitation in Bounded Combinatory Logic with Intersection Types for Composition Synthesis**, ITRS 2012 [Düd+12]
  ▶ First short experiments

- **Towards Combinatory Logic Synthesis**, BEAT 2013 [Reh13]
  ▶ Outline of research program

- **Design and Synthesis from Components** (*Dagstuhl Seminar 14232*), 2014 (organized by Rehof and Vardi) [RV14]
  ▶ The idea of synthesis from components

- **Staged Composition Synthesis**, ESOP 2014 [DMR14]
  ▶ Moving to the meta-level via modal types

- **Mixin Composition Synthesis Based on Intersection Types**, TLCA 2015 [Bes+15]
  ▶ First study of object-oriented features

The idea of using intersection types as foundation for type-based synthesis also taken up for λ-calculus inhabitation, see Frankle, Osera, Walker, Zdancewic, POPL 2016 [Fra+16]
Combinatory Logic Synthesis (CLS)

It is currently being developed into a meta-programming framework based on a combinatory extension to Scala led by Jan Bessai (TU Dortmund), Boris Düdder (U Copenhagen), George Heineman (WPI Boston)

- Combinatory Logic Synthesizer, ISOLA 2014 [Bes+14]
- Synthesizing Type-safe Compositions in Feature Oriented Software Designs using Staged Composition, ModSyn-PL 2015 [DRH15]
- Towards Migrating Object-Oriented Frameworks to Enable Synthesis of Product Line Members, SPLC 2015 [Hei+15]
- Combinatory Synthesis of Classes using Feature Grammars, FACS 2015 [Bes+16b]
- A Long and Winding Road Towards Modular Synthesis, ISOLA 2016 [Hei+16]
- Combinatory Process Synthesis, ISOLA 2016 [Bes+16a]
- Component-Oriented Synthesis via Algebras and Combinatory Logic, in preparation
Example Repository

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

From: Scala integrated framework for CLS by Bessai, Düdder, Dudenhofner, Heinemann
Example Repository

\[ \Gamma = \{ \text{customerForm} : (\text{String} \rightarrow \text{java.net.URL} \rightarrow \text{OptionSelection} \rightarrow \text{Form}) \cap \\
(\text{Title} \rightarrow \text{Location(Logo)} \rightarrow \text{ChoiceDialog(\(\alpha\))} \rightarrow \text{OrderMenu(\(\alpha\))}) \\
\text{dropDownSelector} : (\text{java.net.URL} \rightarrow \text{OptionSelection}) \cap \\
(\text{Location(Database)} \rightarrow \text{ChoiceDialog(DropDown)}) \\
\text{radioButtonSelector} : (\text{java.net.URL} \rightarrow \text{OptionSelection}) \cap \\
(\text{Location(Database)} \rightarrow \text{ChoiceDialog(RadioButtons)}) \\
\text{companyTitle} : \text{String} \cap \text{Title} \\
\text{databaseLocation} : \text{java.net.URL} \cap \text{Location(Database)} \\
\text{logoLocation} : \text{java.net.URL} \cap \text{Location(Logo)} \\
\text{alternateLogoLocation} : \text{java.net.URL} \cap \text{Location(Logo)} \} \]
Combinator Interface in Scala

```
trait Repository {
  type Form
  type OptionSelection

  lazy val alpha = Variable("alpha")
  lazy val kinding = Kinding(alpha).addOption('DropDown).addOption('RadioButtons)

trait CustomerForm {
  def apply(title: String, logolocation: URL, optionSelector: OptionSelection): Form
  val semanticType: Type = 'Title => 'Location('Logo) => 'ChoiceDialog(alpha) => 'OrderMenu(alpha)
}
val customerForm: CustomerForm
```
Combinator Implementation in Scala

class SwingRepository extends Repository {
  type Form = CompilationUnit
  type OptionSelection = CompilationUnit => CompilationUnit

  @combinator object customerForm extends CustomerForm {

    override def apply(title: String, logoLocation: URL, optionSelector: OptionSelection): CompilationUnit = {
      val file = scala.io.Source.fromInputStream(getClass.getResourceAsStream("CustomerForm.java")).mkString
      val form = Java(file).compilationUnit()
      val cls = form.getClassByName("CustomerForm").get
      val initMethod = cls.getMethodsByName("initComponents").get(0)

      optionSelector(form)

      form.addImport("java.net.URL")
      initMethod
        .getBody.get()
        .addStatement(0, Java("\n        try {
        |   this.add(new JLabel(new ImageIcon(new URL("$logoLocation")));\n        | } catch (Exception e) {
        |
        |}"""".stripMargin).statement()

      initMethod
        .getBody.get()
        .addStatement(0, Java("\n        this.setTitle("$title");"""").statement())

      form
    }
  }
}
Variations
Web Interface to Inhabitation Service

Requests:

\[ \Gamma \vdash ? : \text{com.github.javaparser.astCompilationUnit \& OrderMenu(omega)} \]

Solutions:

**Variation 0:**

<table>
<thead>
<tr>
<th>Raw</th>
<th>Git</th>
</tr>
</thead>
<tbody>
<tr>
<td>List[Tree(customerForm,List[Tree(companyTitle,List())], Tree(locationLocation,List())), Tree(dropDownSelector,List[Tree(databaseLocation,List())])]</td>
<td></td>
</tr>
</tbody>
</table>

**Variation 1:**

<table>
<thead>
<tr>
<th>Raw</th>
<th>Git</th>
</tr>
</thead>
</table>
| # clone into new git:  
git clone -b variation_1 http://localhost:9000/guidemo/guidemo.git  
* checkout branch in existing git:  
git fetch origin  
git checkout -b variation_1 origin/variation_1 | |

**Variation 2:**

Compute

**Variation 3:**

Compute
Combinatory Logic With Intersection Types $\mathcal{CL}(\to, \cap)$

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

- Types are taken modulo associativity, commutativity and idempotence of $\cap$, and $\leq$ is least preorder containing (cf. [BCDC83])

\[
\begin{align*}
\sigma & \leq \omega, \quad \omega \leq \omega \to \omega, \quad \sigma \cap \tau \leq \sigma, \quad \sigma \cap \tau \leq \tau, \quad \sigma \leq \sigma \cap \sigma; \\
(\sigma \to \tau) & \cap (\sigma \to \rho) \leq \sigma \to \tau \cap \rho; \\
\text{If } \sigma \leq \sigma' \text{ and } \tau \leq \tau' \text{ then } \sigma \cap \tau & \leq \sigma' \cap \tau' \text{ and } \sigma' \to \tau \leq \sigma \to \tau'.
\end{align*}
\]

- The $\text{SKI}$-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)
CLS World View

- CL over arbitrary bases is a general theory of component collections (repositories)
- Each repository is a combinatory theory
- The special “implementation theory” $\{S, K, I\}$ is one, very special case
Relativized Inhabitation

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

\[^1\]See also talk on Wednesday by A. Dudenhefner, \textit{Lower End of the Linial-Post Spectrum}
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 $PSPACE$-complete in simple types (Statman’s theorem)

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

---

\(^1\) See also talk on Wednesday by A. Dudenhefner, *Lower End of the Linial-Post Spectrum*
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 $PSPACE$-complete in simple types (Statman’s theorem)

- Relativized inhabitation is much harder
  - **Undecidable in simple types**: Linial-Post theorems, 1948ff. [LP49]

- 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 talk on Wednesday by A. Dudenhefner, *Lower End of the Linial-Post Spectrum*
Linial-Post Spectrum

\[ \infty \]

\[ \text{Ptime} \quad \text{co-NP} \quad \text{Pspace} \quad \text{Exptime} \quad 2\text{Exptime} \]

\[ \text{CPL} \quad \text{IPL (S4)} \quad \text{T} \quad \text{R} \]\n
Intermediate logics
Subintuitionistic logics

\[ \text{R} \rightarrow \text{T} \rightarrow \text{R} \rightarrow \text{IPL (S4)} \rightarrow \text{CPL} \]

Subintuitionistic logics  Intermediate logics
Bounded Combinatory Logic $\mathsf{BCL}_k(\to, \cap)$

- **Levels**

  \[
  \ell(a) = 0, \text{ for } a \in \mathbb{A}; \\
  \ell(\tau \to \sigma) = 1 + \max\{\ell(\tau), \ell(\sigma)\}; \\
  \ell(\bigcap_{i=1}^n \tau_i) = \max\{\ell(\tau_i) \mid i = 1, \ldots, n\}. \\
  \ell(S) = \max\{\ell(S(\alpha)) \mid S(\alpha) \neq \alpha\}
  \]
Bounded Combinatory Logic $\mathbb{BCL}_k(\to, \cap)$

- Levels

$$\ell(a) = 0, \text{ for } a \in \mathbb{A};$$
$$\ell(\tau \to \sigma) = 1 + \max\{\ell(\tau), \ell(\sigma)\};$$
$$\ell(\bigcap_{i=1}^{n} \tau_i) = \max\{\ell(\tau_i) \mid i = 1, \ldots, n\}.$$

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

- $\mathbb{BCL}_k(\to, \cap), \ k \geq 0$ (and finite CL, $\mathbb{FCL}$, with $S = \text{id}$).

$$\frac{[\ell(S) \leq k]}{\Gamma, X : \tau \vdash_k X : S(\tau)} \quad (\text{var})$$

$$\frac{\Gamma \vdash_k e : \tau \to \sigma \quad \Gamma \vdash_k e' : \tau}{\Gamma \vdash_k (e \ e') : \sigma} \quad (\to E)$$

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

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

- *Bounded Combinatory Logic*, CSL 2012 [Düd+12]
Complexity for Finite and Bounded CL

Theorem (TLCA 2011 [RU11])

For finite combinatory logic $fcl$:

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

Theorem (CSL 2012 [Düd+12])

For bounded combinatory logic $bcl_k$:

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

Input: $\Gamma, \tau, k$

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

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

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

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

**loop:**

1. \text{CHOOSE} \((x : \sigma) \in \Gamma; \)

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

\[ \sigma' = (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \land \cdots \land \]
\[ (1 \rightarrow 1) \rightarrow (1 \rightarrow 1) \rightarrow (1 \rightarrow 1) \]

\[ (1 \rightarrow 1) \rightarrow (1 \rightarrow 1) \rightarrow (1 \rightarrow 1) \]
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. $\sigma' := \bigcap\{S(\sigma) \mid S \in S_x^{(\Gamma, \tau, k)}\};$

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

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

5. if $(\bigcap_{\pi \in P} \tau_{m}^{\pi} \leq \tau)$ then $(0 \rightarrow 0) \cap (1 \rightarrow 1)$

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

7. else for all $(i = 1 \ldots m)$

8. $\tau := \bigcap_{\pi \in P} \tau_{m}^{\pi}$

9. $(0 \rightarrow 1) \rightarrow (1 \rightarrow 0) \rightarrow (0 \rightarrow 0) \cap$

10. $(1 \rightarrow 1) \rightarrow (1 \rightarrow 1) \rightarrow (1 \rightarrow 1)$

11. **goto** loop;

12. else reject.
Upper Bound ATM for $bcl_k(\rightarrow, \cap)$: \textsc{Aspace}(exp\(_{k+1}(n))$)

\textit{Input}: $\Gamma, \tau, k$

\begin{align*}
\Gamma &= \{f : (0 \rightarrow 1) \cap (1 \rightarrow 0), \\
x : (\alpha \rightarrow \beta) \rightarrow (\beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \gamma)\}\n\end{align*}

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

\textit{loop}:

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

2. $\sigma' := \cap\{S(\sigma') | S \in S_{x}^{(\Gamma,\tau,k)}\}$;

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

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

5. \textsc{if} $(\cap_{\pi \in P} tgt_m(\pi) \leq \tau)$ \texttt{then}

6. \textsc{if} $(m = 0)$ \texttt{then accept}.

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

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

$(0 \rightarrow 0) \cap (1 \rightarrow 1) \leq \tau$
Upper Bound ATM for $\mathsf{bcl}_k(\to, \cap): \mathsf{ASPACE}(\exp_{k+1}(n))$

**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)\}\n
\[\tau = (0 \to 0) \cap (1 \to 1)\]

**loop:**
1. \(\text{CHOOSE } (x : \sigma) \in \Gamma;\)
2. \(\sigma' := \bigcap\{S(\sigma) \mid S \in S_x^{(\Gamma,\tau,k)}\};\)
3. \(\text{CHOOSE } m \in \{0, \ldots, \lVert \sigma' \rVert\};\)
4. \(\text{CHOOSE } P \subseteq \mathcal{P}_m(\sigma');\)
5. \(\text{IF } (\bigcap_{\pi \in P} \text{tgt}_m(\pi) \leq \tau) \text{ THEN }\)
6. \(\text{IF } (m = 0) \text{ THEN ACCEPT; }\)
7. \(\text{ELSE }\)
8. \(\text{FORALL}(i = 1 \ldots m)\)
9. \(\tau := \bigcap_{\pi \in P} \text{arg}_i(\pi);\)
10. \(\text{GO TO loop; }\)
11. \(\text{ELSE REJECT; }\)
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 $(\bigcap_{\pi \in P} tgt_m(\pi) \leq \tau)$ then
   - if ($m = 0$) then accept;
   - else
     - forall ($i = 1 \ldots m$)
       - $\tau := \bigcap_{\pi \in P} \text{arg}_i(\pi)$;
     - $\tau := (0 \rightarrow 1) \cap (1 \rightarrow 0)$
     - $\tau := (1 \rightarrow 0) \cap (0 \rightarrow 1)$
   - goto loop;
5. else reject;

$(x \ f) \ f : (0 \rightarrow 0) \cap (1 \rightarrow 1)$
Lower bound - main ideas

- Generic reduction by simulation of $\exp_{k+1}(n)$-space bounded alternating Turing machines.
- Given ATM $\mathcal{M}$, we construct, in polynomial time, an environment $\Gamma$ such that $\mathcal{M}$ is accepting if and only if $\Gamma \vdash_k ? : \text{Tape}$ is solvable.
Lower bound - main ideas

- Generic reduction by simulation of $\exp_{k+1}(n)$-space bounded alternating Turing machines.
- Given ATM $M$, we construct, in polynomial time, an environment $\Gamma$ such that $M$ is accepting if and only if $\Gamma \vdash_k ? : \text{Tape}$ is solvable.

For any fixed level-parameter $K$:

- Intersection type numerals $\langle i \rangle_K$: we can represent numbers $0 \leq i \leq \exp_{K+1}(n) - 1$ as intersection types.
- We can represent ATM configurations $C$ of size $\exp_{K+1}(n)$ as intersection types $[C]$.
- Exploiting $K$-bounded polymorphism, we can represent these types implicitly in polynomial sized types.
- ATM sequences $C_1 C_2 \cdots C_m$ coded by reverse implications $[C_{i+1}] \rightarrow [C_i]$ in $\Gamma$. 
Refinement (after [FP91])

**Definition**

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

$$
\begin{align*}
U_X(o) &= X

U_X(\tau \rightarrow \sigma) &= (U_X(\tau) \Rightarrow U_X(\sigma))^\cap
\end{align*}
$$

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.,

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

Typability (\lambda-calculus) is in linear time over simple types, and type inference is computable [FP91].

Inhabitation (\lambda-calculus) is undecidable.

Proof: Note that [Sal+12] uses only uniform types for \lambda-definability. □
Refinement (after [FP91])

**Definition**

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

- $U_X(o) = X \cap$
- $U_X(\tau \rightarrow \sigma) = (U_X(\tau) \Rightarrow U_X(\sigma)) \cap$

With such types we can represent any finite function $f : A \rightarrow B$ at the type level by $\bigcap_{a \in A} (a \rightarrow f(a))$
Refinement (after [FP91])

Definition

Let $\mathbb{T}_o$ be simple types over an atom $o$. Fix $X \subseteq A$ and define uniform

types $U_X(\tau)$ for $\tau \in \mathbb{T}_o$:

\[
\begin{align*}
U_X(o) &= X \\
U_X(\tau \rightarrow \sigma) &= (U_X(\tau) \Rightarrow U_X(\sigma))^\cap
\end{align*}
\]

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.,

\[
\text{succ} : (\text{Nat} \rightarrow \text{Nat}) \cap (\text{zero} \rightarrow \text{pos}) \cap (\text{pos} \rightarrow \text{pos}) \cap (\text{even} \rightarrow \text{odd}) \cap (\text{odd} \rightarrow \text{even})
\]
Refinement (after [FP91])

**Definition**

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

- $U_X(o) = X$\(\cap\)
- $U_X(\tau \rightarrow \sigma) = (U_X(\tau) \Rightarrow U_X(\sigma))$\(\cap\)

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.,

- $\text{succ} : (\text{Nat} \rightarrow \text{Nat}) \cap (\text{zero} \rightarrow \text{pos}) \cap (\text{pos} \rightarrow \text{pos}) \cap (\text{even} \rightarrow \text{odd}) \cap (\text{odd} \rightarrow \text{even})$

*Typability* ($\lambda$-calculus) is in linear time over simple types, and type inference is computable [FP91]
Refinement (after [FP91])

**Definition**

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

\[
U_X(o) = X^n \\
U_X(\tau \to \sigma) = (U_X(\tau) \Rightarrow U_X(\sigma))^n
\]

- With such types we can represent any finite function $f : A \to B$ at the type level by $\bigcap_{a \in A} (a \to f(a))$
- We can express finite abstract interpretations, e.g.,
  \[
succ : (Nat \to Nat) \cap (zero \to pos) \cap (pos \to pos) \cap (even \to odd) \cap (odd \to even)
  \]
- *Typability* ($\lambda$-calculus) is in linear time over simple types, and type inference is computable [FP91]
- *Inhabitation* ($\lambda$-calculus) is undecidable.
Refinement (after [FP91])

**Definition**

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

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

- 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.,
  \[
  \text{succ} : (\text{Nat} \rightarrow \text{Nat}) \cap (\text{zero} \rightarrow \text{pos}) \cap (\text{pos} \rightarrow \text{pos}) \cap (\text{even} \rightarrow \text{odd}) \cap (\text{odd} \rightarrow \text{even})
  \]
- *Typability* (\( \lambda \)-calculus) is in linear time over simple types, and type inference is computable [FP91]
- *Inhabitation* (\( \lambda \)-calculus) is undecidable. *Proof*: Note that [Sal+12] 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 $X \subseteq \mathbb{A}$ 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 $k = \max\{\ell(\tau) \mid \sigma \in \text{rn}(\Gamma) \cap U_X(\tau)\}$.

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 $\mathbb{BCL}_k(\rightarrow, \cap)$

Satisfiability of formulae

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

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

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

Representation in $\mathbb{BCL}_k(\rightarrow, \cap)$ (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 $\text{In}_k(x^k, x^k \rightarrow 1)$ and $\text{NotIn}(x^k, x^k \rightarrow 0)$ are axioms.

- Use alternation to code quantifiers as usual (Urzyczyn 1997).
Ongoing: optimization & algorithm engineering


<table>
<thead>
<tr>
<th>Algorithm 4.5: ATM with lookahead-test</th>
</tr>
</thead>
</table>

\[
\text{Input: } \Gamma, \tau \rightarrow \text{all types in } \Gamma \text{ and } \tau = \bigcap_{i \in I} \tau_i \text{ organized loop:}
\]

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}(tgt_m(\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 } j_i \in J \text{ with candidates}(i, j_i, 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 } tgt_m(\pi_i) \leq \tau_i \text{ and}
11. \quad \forall 1 \leq l \leq m \forall \pi' \in \text{argl}(\pi_i) \exists (y : \rho) \in \Gamma \exists \text{a path } \rho' \in \rho \exists k : \text{Match}(tgt_k(\rho') \leq \pi') = \text{true}
12. \text{IF } (m = 0) \text{ THEN ACCEPT;}
13. \text{ELSE FORALL } (l = 1 \ldots m)
14. \quad \tau := \bigcap_{i \in I} \text{argl}(\pi_i);
15. \text{GO TO loop;}

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>Unoptimized ATM</th>
<th>Lookahead-ATM</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>73</td>
<td>112.2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>973</td>
<td>209.6</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>11 665</td>
<td>2245.2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>43 905</td>
<td>12 504</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>$1.4 \times 10^{16}$</td>
<td>–</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>$4.8 \times 10^{12}$</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>$1.3 \times 10^{14}$</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>$6.6 \times 10^{18}$</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>$3.3 \times 10^{28}$</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 4.1: Experimental Data for $\Gamma^m$
Inhabitation in Bounded Dimension

Intersection Type Calculi of Bounded Dimension, POPL 2017 [DR17a]
Strict Intersection Types

**Definition (Strict Intersection Types)**

\[
\begin{align*}
T_s A, B & ::= a | \sigma \to A \\
T_s \sigma, \tau & ::= A_1 \cap \cdots \cap A_n \quad n \geq 1
\end{align*}
\]

**Definition (Strict Type Assignment [Bak11](Def. 5.1))**

\[
\begin{align*}
\frac{1 \leq i \leq n}{\Gamma, x : \bigcap_{i=1}^n A_i \vdash_s x : A_i} & \quad (\text{Var}) \\
\frac{\Gamma \vdash_s M : A_i \text{ for } i = 1 \ldots n}{\Gamma \vdash_s M : \bigcap_{i=1}^n A_i} & \quad (\cap I) \\
\frac{\Gamma \vdash_s M : \sigma \to A \quad \Gamma \vdash_s N : \sigma}{\Gamma \vdash_s MN : A} & \quad (\to E) \\
\frac{\Gamma, x : \sigma \vdash_s M : A}{\Gamma \vdash_s \lambda x . M : \sigma \to A} & \quad (\to I)
\end{align*}
\]
Notational Variant

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 \cite{Bak11}(Def. 5.1))

\[
\frac{1 \leq i \leq n}{\Gamma, x : [A_1, \ldots, A_n] \vdash_s x : [A_i]} \quad (\text{Var})
\frac{\Gamma \vdash_s M : [A_i] \text{ for } i = 1 \ldots n}{\Gamma \vdash_s M : [A_1, \ldots, A_n]} \quad (\cap !)
\]
\[
\frac{\Gamma \vdash_s M : [\sigma \rightarrow A] \quad \Gamma \vdash_s N : \sigma}{\Gamma \vdash_s MN : [A]} \quad (\rightarrow \text{E})
\]
\[
\frac{\Gamma, x : \sigma \vdash_s M : [A]}{\Gamma \vdash_s \lambda x. M : [\sigma \rightarrow A]} \quad (\rightarrow \text{I})
\]
**Set Theoretic Elaborations**

### Definition (Elaborations)

\[ P, Q, R ::= x\langle S \rangle \mid (\lambda x. P)\langle S \rangle \mid (P \ Q)\langle S \rangle, \ S = [A_1, \ldots, A_n], \ n \geq 1 \]

### Definition \((P \sqcup Q, \text{defined for } \lceil P \rceil \equiv \lceil Q \rceil)\)

\[
\begin{align*}
x\langle S \rangle \sqcup x\langle S' \rangle & \equiv x\langle S \cup S' \rangle \\
(\lambda x. P)\langle S \rangle \sqcup (\lambda x. Q)\langle S' \rangle & \equiv (\lambda x. P \sqcup Q)\langle S \cup S' \rangle \\
(PQ)\langle S \rangle \sqcup (P'Q')\langle S' \rangle & \equiv ((P \sqcup P')(Q \sqcup Q'))\langle S \cup S' \rangle
\end{align*}
\]

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

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

- **Non-negativity**: \(||P|| > 0\)
- **Subadditivity**: \(||P \sqcup Q|| \leq ||P|| + ||Q||\) for \(\lceil P \rceil \equiv \lceil Q \rceil\)
Set Theoretic Elaborations

Definition (Elaborations)

\[ P, Q, R ::= x(S) \mid (\lambda x.P)(S) \mid (P \land Q)(S), \ S = [A_1, \ldots, A_n], \ n \geq 1 \]

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

\[ x(S) \uplus x(S') \equiv x(S \cup S') \]
\[ (\lambda x.P)(S) \uplus (\lambda x.Q)(S') \equiv (\lambda x.(P \uplus Q))(S \cup S') \]
\[ (PQ)(S) \uplus (P'Q')(S') \equiv ((P \uplus P')(Q \uplus Q'))(S \cup S') \]

Definition (Norm \( \|•\| \))

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

Non-negativity  :  \( \|P\| > 0 \)
Subadditivity    :  \( \|P \uplus Q\| \leq \|P\| + |Q| \) for \([P] \equiv [Q] \)
Set Theoretic Elaboration System

Definition \((\Gamma \vdash P : \sigma)\)

\[
\begin{align*}
1 \leq i \leq n & \quad (\text{Var}) \\
\Gamma, x : [A_1, \ldots, A_n] \vdash x\langle[A_i]\rangle : [A_i] \\
\Gamma, x : \sigma \vdash P : [A] & \quad (\rightarrow I) \\
\Gamma \vdash (\lambda x. P)\langle[\sigma \rightarrow A]\rangle : [\sigma \rightarrow A] \\
\Gamma \vdash P : [\sigma \rightarrow A] & \quad (\rightarrow E) \\
\Gamma \vdash Q : \sigma \\
\Gamma \vdash (P \ Q)\langle[A]\rangle : [A] \\
\Gamma \vdash P_i : [A_i] \text{ for } i = 1 \ldots n & \quad (\cap I) \\
\Gamma \vdash \bigsqcup_{i=1}^{n} P_i : [A_1, \ldots, A_n] \\
\end{align*}
\]
Set Theoretic Elaboration System

Definition ($\Gamma \vdash P : \sigma$)

1. \(1 \leq i \leq n\) (Var)

\[ \Gamma, x : [A_1, \ldots, A_n] \vdash x\langle[A_i]\rangle : [A_i] \]

2. \(\Gamma, x : \sigma \vdash P : [A] \) (→I)

\[ \Gamma \vdash (\lambda x.P)\langle[\sigma \rightarrow A]\rangle : [\sigma \rightarrow A] \]

3. \(\Gamma \vdash P : [\sigma \rightarrow A], \Gamma \vdash Q : \sigma \) (→E)

\[ \Gamma \vdash (P \ Q)\langle[A]\rangle : [A] \]

4. \(\Gamma \vdash P_i : [A_i] \) for \(i = 1 \ldots n\) (∩I)

\[ \Gamma \vdash \bigsqcup_{i=1}^n P_i : [A_1, \ldots, A_n] \]

Intuition

The operation \(\bigsqcup_{i=1}^n P_i\) allows us to measure usage of (∩I) as a logical resource under norm \(||\bullet||\)
Set Theoretic Elaboration System

Definition \((\Gamma \vdash P : \sigma)\)

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

Definition

Write \(\Gamma \vdash M \mapsto P : \sigma\) iff \(\Gamma \vdash P : \sigma\) with \(M \equiv [P]\).

- Clearly, \(\Gamma \vdash_s M : \sigma\) iff \(\exists P. \Gamma \vdash M \mapsto P : \sigma\)
Intersection Type Calculus of Bounded Dimension

**Definition (\(\lambda^{[n]}\))**

\[ \Gamma \vdash_n M : \sigma \quad \text{iff} \quad \exists P. \Gamma \vdash M \rightarrow P : \sigma \text{ with } \|P\| \leq n \]

**Lemma**

\[ \Gamma \vdash_s M : \sigma \quad \text{iff} \quad \Gamma \vdash_n M : \sigma \text{ for some } n > 0 \]

**Definition (Dimension)**

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

\[ \dim^\sigma_{\Gamma} = \min\{n \mid \Gamma \vdash_n M : \sigma\} \]
Subject Reduction in Bounded Dimension

Terms can be elaborated in non-increasing norm under \( \beta \)-reduction:

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

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 \( \lambda^{[n]} \) is a meaningful type system.
- Inhabitation in bounded dimension for \( \lambda^{[n]} \) 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 $\Gamma \vdash_n M : \sigma$?
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.

**Proposition**

For \( n = 1 \) set-theoretic inhabitation is \( \text{PSPACE-complete} \) \([RU12] \) Cor. 22).
Multiset Elaboration System

**Definition**

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

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

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

where \((\bigcap \text{I})\) is for each \( x<s> \) in \( \biguplus_{i=1}^{n} P_i \) if \( x \) free in \( M \), then \( s \in \Delta(x) \).
Examples

Example ($I \equiv \lambda x.x$)

\[
\begin{align*}
\frac{x\langle [a] \rangle}{(\lambda x.x\langle [a] \rangle)\langle [a \to a] \rangle} & \quad (\rightarrow I) \\
\frac{x\langle [b] \rangle}{(\lambda x.x\langle [b] \rangle)\langle [b \to b] \rangle} & \quad (\rightarrow I) \\
\mathcal{P} & := (\lambda x.x\langle [a, b] \rangle)\langle [a \to a, b \to b] \rangle \\
||\mathcal{P}|| & = 2
\end{align*}
\]

Example ($\omega \equiv \lambda x.xx$)

\[
\begin{align*}
\frac{x\langle [a \to a] \rangle \quad x\langle [a] \rangle}{(x\langle [a \to a] \rangle \ x\langle [a] \rangle)\langle [a] \rangle} & \quad (\rightarrow E) \\
\frac{(x\langle [a \to a] \rangle \ x\langle [a] \rangle)\langle [a] \rangle}{(\lambda x.(x\langle [a \to a] \rangle \ x\langle [a] \rangle)\langle [a] \rangle)\langle [[a, a \to a] \to a] \rangle} & \quad (\rightarrow I) \\
\mathcal{P} & := (\lambda x.(x\langle [a \to a] \rangle \ x\langle [a] \rangle)\langle [a] \rangle)\langle [[a, a \to a] \to a] \rangle \\
||\mathcal{P}|| & = 1
\end{align*}
\]

Note that the system is non-idempotent and non-linear.
Examples

Example \((x(y(\lambda z.z)))\)

Let \(A \equiv [a \to a, b \to b] \to e, B \equiv [b \to b, c \to c] \to f,\)
\(\Delta = \{x : [e, f] \to g, y : [A, B]\}.\)

We have \(\Delta \vdash P : [e],\) where \(P\) is

\[x\langle[[e, f] \to g]\rangle(y\langle[A, B]\rangle (\lambda z.z\langle\begin{bmatrix} a \\ b \\ c \end{bmatrix}\rangle\langle\begin{bmatrix} a \to a \\ b \to b \\ c \to c \end{bmatrix}\rangle))\]
Beyond Rank 2 Inhabitation

- The rank hierarchy jumps from \( \text{EXPSPACE} \) in rank 2 to infinity at ranks \( k \geq 3 \), as was shown by urzyczyn [Urz09]
- But we can show [DR17a] that dimensional bound provides a rank-independent stratification, decidable in each dimensional bound, and subsuming the rank 2 fragment in \( \text{EXPSPACE} \)
Inhabitation in Bounded Multiset Dimension

Definition

\[ \Gamma \vdash_n M : \sigma \text{ iff } \exists \Delta, P, s. (\Delta \vdash M \iff P : s \text{ with } \Gamma = \Delta^\circ \text{ and } \sigma = s^\circ \text{ and } \|P\| \leq n) \]

where \((\_)^\circ\) collapses multisets to underlying sets.
Inhabitation in Bounded Multiset Dimension

Definition

\[ \Gamma \vdash_n M : \sigma \text{ iff } \exists \Delta, P, s. (\Delta \vdash M \iff P : s \text{ with } \Gamma = \Delta^\circ \text{ and } \sigma = s^\circ \text{ and } \|P\| \leq n) \]

where \((\_)^\circ\) collapses multisets to underlying sets.

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

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

is there a term \(M\) such that \(\Gamma \vdash_n M : \sigma\)?
Inhabitation in Bounded Multiset Dimension

Definition

\( \Gamma \vdash_n M : \sigma \) iff
\[ \exists \Delta, P, s. (\Delta \vdash M \iff P : s \text{ with } \Gamma = \Delta^\circ \text{ and } \sigma = s^\circ \text{ and } ||P|| \leq n) \]
where \((\_)^\circ\) 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 \textsc{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 \(\sigma\).
Inhabitation in Bounded Multiset Dimension

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

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}\text{-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}\text{-complete}.
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 $\|P\| = n$. 
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
Substantial generalization of inhabitation in rank 2 fragment, generalizing across all ranks within ExpSpace.
Dimensional Analysis of Rank 2 Typings

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

Consequence
Substantial generalization of inhabitation in rank 2 fragment, generalizing across all ranks within EXPSPACE.

Remark
Compare to linear, non-idempotent system of Bucciareli, Kesner, Ronchi Della Rocca [BKRDR14]:
- Inhabitation is decidable [BKRDR14] and NP-complete [DR17a]
- Typability is undecidable.
The World

Cellarius
Harmonia Macrocosmica
Planisphaerium Copernicanum

Cellarius
Harmonia Macrocosmica
Planisphaerium Copernicanum
The World
Multiset Dimension and Parallel Constraints
Inhabitation (after Urzyczyn [Urz97])

\[ \vdash ? : (a \rightarrow c) \rightarrow (b \rightarrow a \rightarrow c) \rightarrow a \rightarrow b \rightarrow c \]
Inhabitation

\[
\vdash \ ? : (a \rightarrow c) \rightarrow (b \rightarrow a \rightarrow c) \rightarrow a \rightarrow b \rightarrow c
\]

\[
\{ f : a \rightarrow c, g : b \rightarrow a \rightarrow c, x : a, y : b \} \vdash X : c
\]
Inhabitation

\[ \vdash ? : (a \rightarrow c) \rightarrow (b \rightarrow a \rightarrow c) \rightarrow a \rightarrow b \rightarrow c \]

\[ \{f : a \rightarrow c, g : b \rightarrow a \rightarrow c, x : a, y : b\} \vdash x : c \]

\[ \exists \]

\[ \exists \]

\[ \{\ldots\} \vdash f y : c \]

\[ \{\ldots\} \vdash g w : c \]
Inhabitation

\[ \vdash \ ? \ : (a \to c) \to (b \to a \to c) \to a \to b \to c \]

\[
\{ f : a \to c, g : b \to a \to c, x : a, y : b \} \vdash X : c
\]

\[
\exists \ 
\{ \ldots \} \vdash f Y : c
\]

\[
\exists \ 
\{ \ldots \} \vdash g Z W : c
\]

\[
\{ \ldots \} \vdash Y : a
\]
Inhabitation

\[ \vdash \ ? \colon (a \to c) \to (b \to a \to c) \to a \to b \to c \]

\[ \{ \ldots \} \vdash \lambda f.\lambda g.\lambda x.\lambda y. fx : \sigma \]

\[ \{ f : a \to c, g : b \to a \to c, x : a, y : b \} \vdash \mathcal{X} : c \]

\[ \{ \ldots \} \vdash fx : c \]

\[ \{ \ldots \} \vdash fy : c \]

\[ \{ \ldots \} \vdash gZW : c \]

\[ \{ \ldots \} \vdash y : a \]

\[ \{ \ldots \} \vdash x : a \]
Inhabitation

\[ \vdash ? : (a \to c) \to (b \to a \to c) \to a \to b \to c \]

\{\ldots\} \vdash \lambda f. \lambda g. \lambda x. \lambda y. f x : \sigma

\{f : a \to c, g : b \to a \to c, x : a, y : b\} \vdash X : c

\{\ldots\} \vdash f x : c
\{\ldots\} \vdash f Y : c

\{\ldots\} \vdash Y : a
\{\ldots\} \vdash x : a

\{\ldots\} \vdash g Z W : c

\{\ldots\} \vdash g Z W : c
\{\ldots\} \vdash Z : b
\{\ldots\} \vdash W : a
Inhabitation

\[ \vdash \ ? : (a \rightarrow c) \rightarrow (b \rightarrow a \rightarrow c) \rightarrow a \rightarrow b \rightarrow c \]

\[ \{ \ldots \} \vdash \lambda f.\lambda g.\lambda x.\lambda y.fx : \sigma \]

\[ \{ \ldots \} \vdash \lambda f.\lambda g.\lambda x.\lambda y.gyx : \sigma \]

\[ \{ \ldots \} \vdash f : a \rightarrow c, g : b \rightarrow a \rightarrow c, x : a, y : b \vdash \exists X : c \]

\[ \{ \ldots \} \vdash fx : c \]

\[ \{ \ldots \} \vdash fy : c \]

\[ \{ \ldots \} \vdash gzw : c \]

\[ \{ \ldots \} \vdash gyx : c \]

\[ \{ \ldots \} \vdash y : a \]

\[ \{ \ldots \} \vdash z : b \]

\[ \{ \ldots \} \vdash w : a \]

\[ \{ \ldots \} \vdash x : a \]
\[ \vdash ? : (a \to c) \to (b \to a \to c) \to a \to b \to c \]

\[ \{\ldots\} \vdash \lambda f.\lambda g.\lambda x.\lambda y.f\ x : \sigma \]

\[ \{\ldots\} \vdash \lambda f.\lambda g.\lambda x.\lambda y.gyx : \sigma \]

\[ \{\ldots\} \vdash f : a \to c, g : b \to a \to c, x, y : a, y : b \vdash X : c \]

\[ \{\ldots\} \vdash f : c \]

\[ \{\ldots\} \vdash fY : c \]

\[ \{\ldots\} \vdash gZW : c \]

\[ \{\ldots\} \vdash gyX : c \]

\[ \{\ldots\} \vdash Y : a \]

\[ \{\ldots\} \vdash x : a \]

\[ \{\ldots\} \vdash Z : b \]

\[ \{\ldots\} \vdash y : b \]

\[ \{\ldots\} \vdash W : a \]

\[ \{\ldots\} \vdash x : a \]

- For simple types this is \textsc{Aptime} = \textsc{Pspace}, and inhabitation is \textsc{Pspace}-complete (Statman 1979)
Intersection Type Inhabitation (after Urzyczyn [Urz09])

\[ \{ f : ((a \to a) \to c) \cap ((b \to b) \to d) \} \vdash ? : c \cap d \]
Intersection Type Inhabitation

\[ \{f : ((a \to a) \to c) \cap ((b \to b) \to d)\} \vdash ? : c \cap d \]
Intersection Type Inhabitation

\[ \{f : ((a \rightarrow a) \rightarrow c) \cap ((b \rightarrow b) \rightarrow d)\} \vdash ? : c \cap d \]
Intersection Type Inhabitation

\[ \{ f : (a \rightarrow a) \rightarrow c \} \cap \{ (b \rightarrow b) \rightarrow d \} \vdash ? : c \cap d \]
Intersection Type Inhabitation

\{f : ((a \rightarrow a) \rightarrow c) \cap ((b \rightarrow b) \rightarrow d)\} \vdash \ ? : c \cap d

\{\ldots\} \vdash f \ \lambda z. z : c
\{\ldots\} \vdash \lambda z. z : a \rightarrow a
\{\ldots, z : a\} \vdash z : a
\{\ldots, z : a\} \vdash Z : a

\{\ldots\} \vdash \lambda z. z : a \rightarrow a
\{\ldots\} \vdash Y : a \rightarrow a
\{\ldots, z : a\} \vdash Y : b \rightarrow b
\{\ldots, z : b\} \vdash Z : b
\{\ldots, z : b\} \vdash z : b

\forall \{\ldots\} \vdash X : a
\forall \{\ldots\} \vdash f \ \lambda z. z : a
Intersection Type Inhabitation

\[ \{ f : ((a \to a) \to c) \cap ((b \to b) \to d) \} \vdash ? : c \cap d \]

- Simultaneous constraints are \textit{coupled} through \( \mathcal{X}, \mathcal{Y}, \mathcal{Z} \)
Multiset Norm and Dimension

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

\( \Gamma \vdash \_ : g \)

\( X \equiv x(y(\lambda z.z)) \)

\( Y \equiv y(\lambda z.z) \)

\( Z \equiv \lambda z.z \)

\( W \equiv z \)

\( \Gamma \vdash Z : a \rightarrow a \)

\( \Gamma, z : a \vdash W : a \)

\( \Gamma, z : a \vdash W : b \)

\( \Gamma, z : b \vdash W : b \)

\( \Gamma, z : c \vdash W : c \)

\( \Gamma \vdash X : g \)

\( \Gamma \vdash Y : e \)

\( \Gamma \vdash Y : f \)

\( \Gamma \vdash Z : b \rightarrow b \)

\( \Gamma \vdash Z : b \rightarrow b \)

\( \Gamma \vdash Z : c \rightarrow c \)

\( \| \cdot \| \)

\( \Gamma_n \Rightarrow \| \cdot \| \leq n \)
Multiset Norm and Dimension

\[ \Gamma = \{ x : (e \cap f) \rightarrow g, y : (((a \rightarrow a) \cap (b \rightarrow b)) \rightarrow e) \cap (((b \rightarrow b) \cap (c \rightarrow c)) \rightarrow f) \} \]

\[ \Gamma \vdash_n ? : g \]

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

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

\[ \mathcal{Z} \equiv \lambda z.z \]

\[ \mathcal{W} \equiv z \]

\[ \Gamma \vdash \mathcal{X} : g \]

\[ \Gamma \vdash \mathcal{Y} : e \]

\[ \Gamma \vdash \mathcal{Y} : f \]

\[ \Gamma \vdash \mathcal{Z} : a \rightarrow a \]

\[ \Gamma \vdash \mathcal{Z} : b \rightarrow b \]

\[ \Gamma \vdash \mathcal{Z} : b \rightarrow b \]

\[ \Gamma \vdash \mathcal{Z} : c \rightarrow c \]

\[ \Gamma, z : a \vdash \mathcal{W} : a \]

\[ \Gamma, z : b \vdash \mathcal{W} : b \]

\[ \Gamma, z : b \vdash \mathcal{W} : b \]

\[ \Gamma, z : c \vdash \mathcal{W} : c \]

\[ \| \cdot \| \]

\[ \vdash_n \Rightarrow \| \cdot \| \leq n \]

See also our talk at TYPES 2016, Rank 3 Inhabitation of Intersection Types Revisited [Bes+16c] and extended version at arXiv.
Typability in Bounded Dimension

Five-set Venn diagram using congruent ellipses in a 5-fold rotationally symmetrical arrangement devised by Branko Grünbaum.
https://commons.wikimedia.org/w/index.php?curid=14250677

*Typability in Bounded Dimension*, LICS 2017 [DR17b]
Elaboration Types

\((\lambda x. x \langle a \rangle) \langle [a, b] \rightarrow a \rangle\)
Elaboration Types

\((\lambda x.x\langle a\rangle)\langle [a, b] \rightarrow a \rangle\)

Decoration type
Elaboration Types

\[(\lambda x. x \langle a \rangle) \langle [a, b] \rightarrow a \rangle\]

Decoration type  Constituent type
Elaboration Types

\((\lambda x. x \langle a \rangle) \langle [a, b] \rightarrow a \rangle\)

Decoration type  Constituent type  Observable type
Subformula Filtration

- The classical *subformula property* ([BCDC83], Lemma 4.5) says that, on *normal forms*, every decoration type is a subformula of some observable type.
- We can show that, on *all typable terms*, one can obtain the *subformula filtration property*:
  - Every constituent subformula is also a decoration type.
- For $X \subseteq \mathbb{T}_s$, we induce a *filtration function* $\mathcal{F}_X : \mathbb{T} \rightarrow \mathbb{T}$ which removes subformula components of intersection types not appearing in $X$. 

Subformula Filtration

- The classical *subformula property* ([BCDC83], Lemma 4.5) says that, on *normal forms*, every decoration type is a subformula of some observable type.

- We can show that, on *all typable terms*, one can obtain the *subformula filtration property*:
  
  - Every constituent subformula is also a decoration type.

- For $X \subseteq \mathbb{T}_s$, we induce a *filtration function* $\mathcal{F}_X : \mathbb{T} \rightarrow \mathbb{T}$ which removes subformula components of intersection types not appearing in $X$.

### Proposition

Let $T_P$ denote the decoration types in $P$ and suppose $T_P \subseteq X$. Then

$$\Gamma \vdash P : \sigma \Rightarrow \mathcal{F}_X(\Gamma) \vdash \mathcal{F}_X(P) : \mathcal{F}_X(\sigma)$$

In particular, taking $T_P = X$ we have

$$\Gamma \vdash P : \sigma \Rightarrow \mathcal{F}_{T_P}(\Gamma) \vdash \mathcal{F}_{T_P}(P) : \mathcal{F}_{T_P}(\sigma)$$
Subformula Filtration

Example

For \( P \equiv (\lambda x. x\langle a\rangle)\langle [a, b] \to a \rangle \) we have

\[
T_P = \{ a, [a, b] \to a \}
\]

and since \( b \notin T_P \) we get

\[
F_{T_P}(P) = (\lambda x. x\langle a\rangle)\langle [a] \to a \rangle
\]
Bounded Width Theorem

**Definition (Width)**

\[
\begin{align*}
\llbracket a \rrbracket &= 1 \\
\llbracket \sigma \to A \rrbracket &= \max\{\llbracket \sigma \rrbracket, \llbracket A \rrbracket\} \\
\llbracket A_1 \cap \cdots \cap A_m \rrbracket &= \max\{m, \llbracket A_1 \rrbracket, \ldots, \llbracket A_m \rrbracket\}
\end{align*}
\]

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

**Theorem (Bounded Width Property)**

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

**Proof.**

By filtration with \( \mathcal{T}_P \) and using \( \llbracket \mathcal{T}_P(\mathcal{D}) \rrbracket \leq \|T_P\| \) 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 $\exists P. \Gamma \vdash P : \sigma$, $[P] \equiv M$, $\|P\| \leq d$)
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 $\exists P. \Gamma \vdash P : \sigma$, $[P] \equiv M$, $\|P\| \leq d$)

Theorem

The typability problem in bounded set-theoretic dimension is $P_{SPACE}$-complete.
Typability in Bounded Dimension

Problem (Typability in bounded set-theoretic dimension)

- Given a $\lambda$-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 $\exists P. \Gamma \vdash P : \sigma$, $[P] \equiv M, \|P\| \leq d$)

Theorem

The typability problem in bounded set-theoretic dimension is $PSPACE$-complete.

Remark

The upper bound is constructed by nondeterministic reduction to standard unification.
We can probably do *much* more (algorithmically) with intersection types than (I, at least) thought possible!

Principles of bounding complementary to that of rank exist and are both useful and logically interesting.
Ongoing & Future Work

- Further work on Scala-integrated, combinatory meta-programming framework and applications to software composition synthesis
- Type inference and theory of principal elaborations in bounded dimension
- Theory of abstract vector space structure of elaborations (semimodule over semiring of type substitution actions)

More speculative:

- Integration of CLS with generate-and-test loops
- Applications in probabilistic program induction and learning
- Applications to theorem proving: automatic composition of proof tactics?


