Combinatory Logic Synthesizer

Boris Düdder

Technical University of Dortmund, Germany
Joint work w. J. Bessai, A. Dudenhefner, M. Martens, and J. Rehof

ISoLA 2014
Evolving Critical Systems
Imperial, Corfu, Greece, October 8th 2014

Outline

Composition Synthesis

- Typed function composition (modus ponens)

$$\frac{\Gamma \vdash F : \tau' \to \tau \quad \Gamma \vdash G : \tau'}{\Gamma \vdash (F\ G) : \tau}(\to\mathrm{E})$$

  as logical model of applicative composition of *named component interfaces* $(X : \rho) \in \Gamma$ from a *repository* Γ, satisfying *goal* τ

- *Inhabitation problem* as foundation for *automatic synthesis:* $\exists e.\ \Gamma \vdash e : \tau$? Notation $\Gamma \vdash ? : \tau$
  - Does there exist a program composition e from repository Γ with $\Gamma \vdash e : \tau$? Inhabitation algorithm is used to *construct* (synthesize) e from Γ and τ

- CLS is inherently *component-oriented*

Composition Synthesis

- Typed function composition (modus ponens)

$$\frac{\Gamma \vdash F : \tau' \to \tau \quad \Gamma \vdash G : \tau'}{\Gamma \vdash (F\ G) : \tau}(\to\mathrm{E})$$

  as logical model of applicative composition of *named component interfaces* $(X : \rho) \in \Gamma$ from a *repository* Γ, satisfying *goal* τ
- *Inhabitation problem* as foundation for *automatic synthesis:* $\exists e.\ \Gamma \vdash e : \tau$? Notation $\Gamma \vdash ? : \tau$
  - Does there exist a program composition e from repository Γ with $\Gamma \vdash e : \tau$? Inhabitation algorithm is used to *construct* (synthesize) e from Γ and τ
- CLS is inherently *component-oriented*

Composition Synthesis

- Typed function composition (modus ponens)

$$\frac{\Gamma \vdash F : \tau' \to \tau \quad \Gamma \vdash G : \tau'}{\Gamma \vdash (F\ G) : \tau}(\to\mathrm{E})$$

  as logical model of applicative composition of *named component interfaces* $(X : \rho) \in \Gamma$ from a *repository* Γ, satisfying *goal* τ
- *Inhabitation problem* as foundation for *automatic synthesis:* $\exists e.\ \Gamma \vdash e : \tau$? Notation $\Gamma \vdash ? : \tau$
  - Does there exist a program composition e from repository Γ with $\Gamma \vdash e : \tau$? Inhabitation algorithm is used to *construct* (synthesize) e from Γ and τ
- CLS is inherently *component-oriented*

Composition Synthesis

- Typed function composition (modus ponens)

$$\frac{\Gamma \vdash F : \tau' \to \tau \quad \Gamma \vdash G : \tau'}{\Gamma \vdash (F\ G) : \tau}(\to\mathrm{E})$$

  as logical model of applicative composition of *named component interfaces* $(X : \rho) \in \Gamma$ from a *repository* Γ, satisfying *goal* τ
- *Inhabitation problem* as foundation for *automatic synthesis:* $\exists e.\ \Gamma \vdash e : \tau$? Notation $\Gamma \vdash ? : \tau$
  - Does there exist a program composition e from repository Γ with $\Gamma \vdash e : \tau$? Inhabitation algorithm is used to *construct* (synthesize) e from Γ and τ
- CLS is inherently *component-oriented*

Foundations in Combinatory Logic

$$\begin{array}{llll} \text{Types} & \tau & ::= & \alpha \mid \tau \to \tau' \\ \text{Terms} & e, e' & ::= & X \mid (e\ e') \end{array}$$

Rules

$$\frac{}{\Gamma, (X : \tau) \vdash X : S(\tau)}(\text{var})$$

$$\frac{\Gamma \vdash e : \tau' \to \tau \quad \Gamma \vdash e' : \tau'}{\Gamma \vdash (e\ e') : \tau}(\to \text{E})$$

Under Curry-Howard isomorphism, Hilbert-style presentation of minimal propositional logic (schematism + modus ponens)

Relativized Inhabitation

- We consider the *relativized inhabitation* problem:
  - **Given** Γ **and** τ, *does there exist* e *such that* $\Gamma \vdash e : \tau$*?*
- Relativized inhabitation is much harder
  - *Undecidable*: **Linial-Post theorems, 1948 ff.**
- *The CLS view:* Already in simple types, relativized inhabitation defines a Turing-complete logic programming language for component composition

Relativized Inhabitation

- We consider the *relativized inhabitation* problem:
  - **Given** Γ **and** τ, *does there exist* e *such that* $\Gamma \vdash e : \tau$*?*
- Relativized inhabitation is much harder
  - *Undecidable*: **Linial-Post theorems, 1948 ff.**
- *The CLS view:* Already in simple types, relativized inhabitation defines a Turing-complete logic programming language for component composition

Relativized Inhabitation

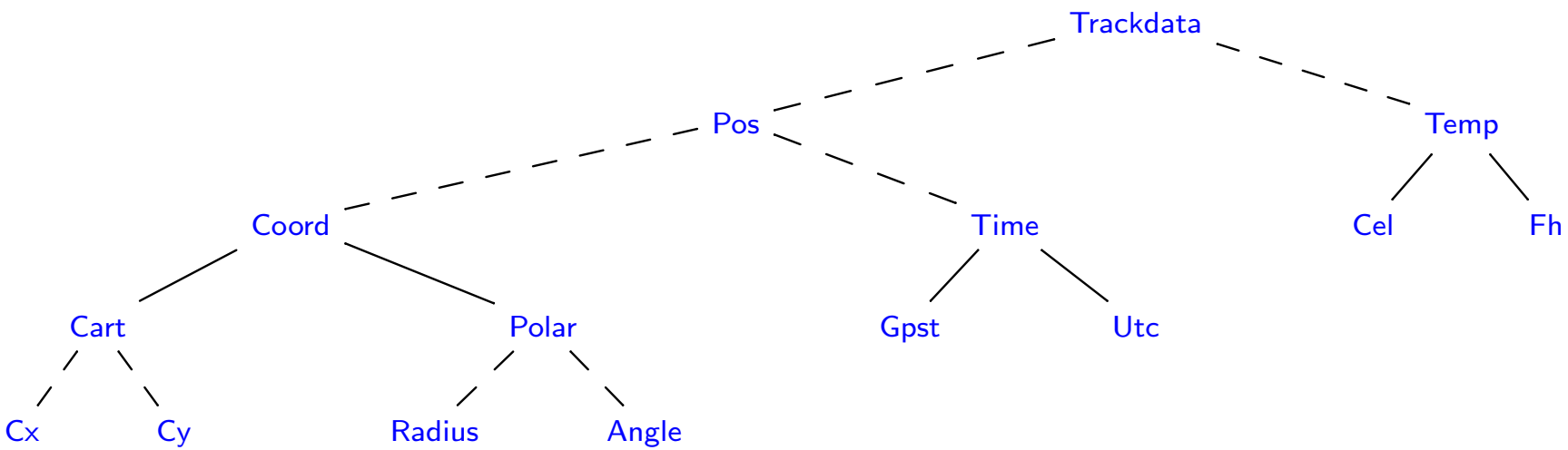
- We consider the *relativized inhabitation* problem:
  - **Given** Γ **and** τ, *does there exist* e *such that* $\Gamma \vdash e : \tau$*?*
- Relativized inhabitation is much harder
  - *Undecidable*: **Linial-Post theorems, 1948 ff.**
- *The CLS view:* Already in simple types, relativized inhabitation defines a Turing-complete logic programming language for component composition

Example Repository

$$
\Gamma = \{
\begin{array}{lcl}
\texttt{O} & : & \texttt{TrObj} \\
\texttt{Tr} & : & \texttt{TrObj} \to \texttt{D}((\texttt{R},\texttt{R}),\texttt{R},\texttt{R}) \\
\texttt{pos} & : & \texttt{D}((\texttt{R},\texttt{R}),\texttt{R},\texttt{R}) \to ((\texttt{R},\texttt{R}),\texttt{R}) \\
\texttt{cdn} & : & ((\texttt{R},\texttt{R}),\texttt{R}) \to (\texttt{R},\texttt{R}) \\
\texttt{fst} & : & (\texttt{R},\texttt{R}) \to \texttt{R} \\
\texttt{snd} & : & (\texttt{R},\texttt{R}) \to \texttt{R} \\
\texttt{tmp} & : & \texttt{D}((\texttt{R},\texttt{R}),\texttt{R},\texttt{R}) \to \texttt{R} \\
\texttt{cc2pl} & : & ((\texttt{R},\texttt{R}),\texttt{R}) \to ((\texttt{R},\texttt{R}),\texttt{R}) \\
\texttt{cl2fh} & : & \texttt{R} \to \texttt{R}
\end{array}
\}
$$

JR: *Towards Combinatory Logic Synthesis*, BEAT'13

Semantic Type Structure



Semantic Repository

$$
\begin{aligned}
\mathscr{C} = \{ & \\
& \texttt{O} : \texttt{TrObj} \\
& \texttt{Tr} : \texttt{TrObj} \to \texttt{D}((\texttt{R},\texttt{R}) \cap \mathit{Cart}, \texttt{R} \cap \mathit{Gpst}, \texttt{R} \cap \mathit{Cel}) \\
& \texttt{pos} : \texttt{D}((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R} \cap \mathfrak{a}', \texttt{R}) \to ((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R} \cap \mathfrak{a}') \cap \mathit{Pos} \\
& \texttt{cdn} : ((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R}) \cap \mathit{Pos} \to (\texttt{R},\texttt{R}) \cap \mathfrak{a} \\
& \texttt{fst} : ((\texttt{R},\texttt{R}) \cap \mathit{Coord} \to \texttt{R}) \cap \\
& \qquad (\mathit{Cart} \to \mathit{Cx}) \cap (\mathit{Polar} \to \mathit{Radius}) \\
& \texttt{snd} : ((\texttt{R},\texttt{R}) \cap \mathit{Coord} \to \texttt{R}) \cap \\
& \qquad (\mathit{Cart} \to \mathit{Cy}) \cap (\mathit{Polar} \to \mathit{Angle}) \\
& \texttt{tmp} : \texttt{D}((\texttt{R},\texttt{R}), \texttt{R}, \texttt{R} \cap \mathfrak{a}) \to \texttt{R} \cap \mathfrak{a} \\
& \texttt{cc2pl} : (\texttt{R},\texttt{R}) \cap \mathit{Cart} \to (\texttt{R},\texttt{R}) \cap \mathit{Polar} \\
& \texttt{cl2fh} : \texttt{R} \cap \mathit{Cel} \to \texttt{R} \cap \mathit{Fh} \\
\}
\end{aligned}
$$

Composition Synthesis via Inhabitation

$$
\mathscr{C} = \{
\begin{array}{lcl}
\texttt{O} & : & \texttt{TrObj} \\
\texttt{Tr} & : & \texttt{TrObj} \to \texttt{D}((\texttt{R},\texttt{R}) \cap \mathit{Cart}, \texttt{R} \cap \mathit{Gpst}, \texttt{R} \cap \mathit{Cel}) \\
\texttt{pos} & : & \texttt{D}((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R} \cap \mathfrak{a}', \texttt{R}) \to ((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R} \cap \mathfrak{a}') \cap \mathit{Pos} \\
\texttt{cdn} & : & ((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R}) \cap \mathit{Pos} \to (\texttt{R},\texttt{R}) \cap \mathfrak{a} \\
\texttt{fst} & : & ((\texttt{R},\texttt{R}) \cap \mathit{Coord} \to \texttt{R}) \cap \\
 & & (\mathit{Cart} \to \mathit{Cx}) \cap (\mathit{Polar} \to \mathit{Radius}) \\
\texttt{snd} & : & ((\texttt{R},\texttt{R}) \cap \mathit{Coord} \to \texttt{R}) \cap \\
 & & (\mathit{Cart} \to \mathit{Cy}) \cap (\mathit{Polar} \to \mathit{Angle}) \\
\texttt{tmp} & : & \texttt{D}((\texttt{R},\texttt{R}), \texttt{R}, \texttt{R} \cap \mathfrak{a}) \to \texttt{R} \cap \mathfrak{a} \\
\texttt{cc2pl} & : & (\texttt{R},\texttt{R}) \cap \mathit{Cart} \to (\texttt{R},\texttt{R}) \cap \mathit{Polar} \\
\texttt{cl2fh} & : & \texttt{R} \cap \mathit{Cel} \to \texttt{R} \cap \mathit{Fh}
\end{array}
\}
$$

$$\mathscr{C} \vdash_{C1} ? : \texttt{R} \cap \mathit{Fh} \rightsquigarrow \mathscr{C} \vdash_{C1} \texttt{cl2fh (tmp (Tr O))} : \texttt{R} \cap \mathit{Fh}$$

$$\mathscr{C} \vdash_{C1} ? : \texttt{R} \cap \mathit{Radius} \rightsquigarrow \mathscr{C} \vdash_{C1} \texttt{fst (cc2pl (cdn (pos (Tr O))))} : \texttt{R} \cap \mathit{Radius}$$

Composition Synthesis via Inhabitation

$$
\mathscr{C} = \{
\begin{array}{lcl}
\texttt{0} & : & \texttt{TrObj} \\
\texttt{Tr} & : & \texttt{TrObj} \to \texttt{D}((\texttt{R},\texttt{R})\cap \mathit{Cart}, \texttt{R}\cap \mathit{Gpst}, \texttt{R}\cap \mathit{Cel}) \\
\texttt{pos} & : & \texttt{D}((\texttt{R},\texttt{R})\cap \mathfrak{a}, \texttt{R}\cap \mathfrak{a}', \texttt{R}) \to ((\texttt{R},\texttt{R})\cap \mathfrak{a}, \texttt{R}\cap \mathfrak{a}')\cap \mathit{Pos} \\
\texttt{cdn} & : & ((\texttt{R},\texttt{R})\cap \mathfrak{a}, \texttt{R})\cap \mathit{Pos} \to (\texttt{R},\texttt{R})\cap \mathfrak{a} \\
\texttt{fst} & : & ((\texttt{R},\texttt{R})\cap \mathit{Coord} \to \texttt{R})\cap \\
 & & (\mathit{Cart} \to \mathit{Cx})\cap(\mathit{Polar} \to \mathit{Radius}) \\
\texttt{snd} & : & ((\texttt{R},\texttt{R})\cap \mathit{Coord} \to \texttt{R})\cap \\
 & & (\mathit{Cart} \to \mathit{Cy})\cap(\mathit{Polar} \to \mathit{Angle}) \\
\texttt{tmp} & : & \texttt{D}((\texttt{R},\texttt{R}), \texttt{R}, \texttt{R}\cap \mathfrak{a}) \to \texttt{R}\cap \mathfrak{a} \\
\texttt{cc2pl} & : & (\texttt{R},\texttt{R})\cap \mathit{Cart} \to (\texttt{R},\texttt{R})\cap \mathit{Polar} \\
\texttt{cl2fh} & : & \texttt{R}\cap \mathit{Cel} \to \texttt{R}\cap \mathit{Fh}
\end{array}
\}
$$

$$\mathscr{C} \vdash_{\mathsf{C1}} ? : \texttt{R}\cap \mathit{Fh} \rightsquigarrow \mathscr{C} \vdash_{\mathsf{C1}} \texttt{cl2fh (tmp (Tr 0))} : \texttt{R}\cap \mathit{Fh}$$

$$\mathscr{C} \vdash_{\mathsf{C1}} ? : \texttt{R}\cap \mathit{Radius} \rightsquigarrow \mathscr{C} \vdash_{\mathsf{C1}} \texttt{fst (cc2pl (cdn (pos (Tr 0))))} : \texttt{R}\cap \mathit{Radius}$$

Composition Synthesis via Inhabitation

$$
\begin{aligned}
\mathscr{C} = \{ \quad & \\
& \texttt{O} && : && \texttt{TrObj} \\
& \texttt{Tr} && : && \texttt{TrObj} \to \texttt{D}((\texttt{R},\texttt{R}) \cap \mathit{Cart}, \texttt{R} \cap \mathit{Gpst}, \texttt{R} \cap \mathit{Cel}) \\
& \texttt{pos} && : && \texttt{D}((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R} \cap \mathfrak{a}', \texttt{R}) \to ((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R} \cap \mathfrak{a}') \cap \mathit{Pos} \\
& \texttt{cdn} && : && ((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R}) \cap \mathit{Pos} \to (\texttt{R},\texttt{R}) \cap \mathfrak{a} \\
& \texttt{fst} && : && ((\texttt{R},\texttt{R}) \cap \mathit{Coord} \to \texttt{R}) \cap \\
& && && (\mathit{Cart} \to \mathit{Cx}) \cap (\mathit{Polar} \to \mathit{Radius}) \\
& \texttt{snd} && : && ((\texttt{R},\texttt{R}) \cap \mathit{Coord} \to \texttt{R}) \cap \\
& && && (\mathit{Cart} \to \mathit{Cy}) \cap (\mathit{Polar} \to \mathit{Angle}) \\
& \texttt{tmp} && : && \texttt{D}((\texttt{R},\texttt{R}), \texttt{R}, \texttt{R} \cap \mathfrak{a}) \to \texttt{R} \cap \mathfrak{a} \\
& \texttt{cc2pl} && : && (\texttt{R},\texttt{R}) \cap \mathit{Cart} \to (\texttt{R},\texttt{R}) \cap \mathit{Polar} \\
& \texttt{cl2fh} && : && \texttt{R} \cap \mathit{Cel} \to \texttt{R} \cap \mathit{Fh} \\
\}
\end{aligned}
$$

$$\mathscr{C} \vdash_{\mathsf{C1}} ? : \texttt{R} \cap \mathit{Fh} \rightsquigarrow \mathscr{C} \vdash_{\mathsf{C1}} \texttt{cl2fh (tmp (Tr O))} : \texttt{R} \cap \mathit{Fh}$$

$$\mathscr{C} \vdash_{\mathsf{C1}} ? : \texttt{R} \cap \mathit{Radius} \rightsquigarrow \mathscr{C} \vdash_{\mathsf{C1}} \texttt{fst (cc2pl (cdn (pos (Tr O))))} : \texttt{R} \cap \mathit{Radius}$$

Composition Synthesis via Inhabitation

$$
\begin{aligned}
\mathscr{C} = \{ \\
&\texttt{0} && : && \texttt{TrObj} \\
&\texttt{Tr} && : && \texttt{TrObj} \to \texttt{D}((\texttt{R},\texttt{R}) \cap \mathit{Cart}, \texttt{R} \cap \mathit{Gpst}, \texttt{R} \cap \mathit{Cel}) \\
&\texttt{pos} && : && \texttt{D}((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R} \cap \mathfrak{a}', \texttt{R}) \to ((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R} \cap \mathfrak{a}') \cap \mathit{Pos} \\
&\texttt{cdn} && : && ((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R}) \cap \mathit{Pos} \to (\texttt{R},\texttt{R}) \cap \mathfrak{a} \\
&\texttt{fst} && : && ((\texttt{R},\texttt{R}) \cap \mathit{Coord} \to \texttt{R}) \cap \\
& && && (\mathit{Cart} \to \mathit{Cx}) \cap (\mathit{Polar} \to \mathit{Radius}) \\
&\texttt{snd} && : && ((\texttt{R},\texttt{R}) \cap \mathit{Coord} \to \texttt{R}) \cap \\
& && && (\mathit{Cart} \to \mathit{Cy}) \cap (\mathit{Polar} \to \mathit{Angle}) \\
&\texttt{tmp} && : && \texttt{D}((\texttt{R},\texttt{R}), \texttt{R}, \texttt{R} \cap \mathfrak{a}) \to \texttt{R} \cap \mathfrak{a} \\
&\texttt{cc2pl} && : && (\texttt{R},\texttt{R}) \cap \mathit{Cart} \to (\texttt{R},\texttt{R}) \cap \mathit{Polar} \\
&\texttt{cl2fh} && : && \texttt{R} \cap \mathit{Cel} \to \texttt{R} \cap \mathit{Fh} \\
\}
\end{aligned}
$$

$$\mathscr{C} \vdash_{\mathsf{C1}} ? : \texttt{R} \cap \mathit{Fh} \rightsquigarrow \mathscr{C} \vdash_{\mathsf{C1}} \texttt{cl2fh (tmp (Tr 0))} : \texttt{R} \cap \mathit{Fh}$$

$$\mathscr{C} \vdash_{\mathsf{C1}} ? : \texttt{R} \cap \mathit{Radius} \rightsquigarrow \mathscr{C} \vdash_{\mathsf{C1}} \texttt{fst (cc2pl (cdn (pos (Tr 0))))} : \texttt{R} \cap \mathit{Radius}$$

Composition Synthesis via Inhabitation

$$
\begin{aligned}
\mathscr{C} = \{ \quad & \\
& \texttt{O} && : && \texttt{TrObj} \\
& \texttt{Tr} && : && \texttt{TrObj} \to \texttt{D}((\texttt{R},\texttt{R}) \cap \mathit{Cart}, \texttt{R} \cap \mathit{Gpst}, \texttt{R} \cap \mathit{Cel}) \\
& \texttt{pos} && : && \texttt{D}((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R} \cap \mathfrak{a}', \texttt{R}) \to ((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R} \cap \mathfrak{a}') \cap \mathit{Pos} \\
& \texttt{cdn} && : && ((\texttt{R},\texttt{R}) \cap \mathfrak{a}, \texttt{R}) \cap \mathit{Pos} \to (\texttt{R},\texttt{R}) \cap \mathfrak{a} \\
& \texttt{fst} && : && ((\texttt{R},\texttt{R}) \cap \mathit{Coord} \to \texttt{R}) \cap \\
& && && (\mathit{Cart} \to \mathit{Cx}) \cap (\mathit{Polar} \to \mathit{Radius}) \\
& \texttt{snd} && : && ((\texttt{R},\texttt{R}) \cap \mathit{Coord} \to \texttt{R}) \cap \\
& && && (\mathit{Cart} \to \mathit{Cy}) \cap (\mathit{Polar} \to \mathit{Angle}) \\
& \texttt{tmp} && : && \texttt{D}((\texttt{R},\texttt{R}), \texttt{R}, \texttt{R} \cap \mathfrak{a}) \to \texttt{R} \cap \mathfrak{a} \\
& \texttt{cc2pl} && : && (\texttt{R},\texttt{R}) \cap \mathit{Cart} \to (\texttt{R},\texttt{R}) \cap \mathit{Polar} \\
& \texttt{cl2fh} && : && \texttt{R} \cap \mathit{Cel} \to \texttt{R} \cap \mathit{Fh} \\
\}
\end{aligned}
$$

$$\mathscr{C} \vdash_{\mathsf{C1}} ? : \texttt{R} \cap \mathit{Fh} \rightsquigarrow \mathscr{C} \vdash_{\mathsf{C1}} \texttt{cl2fh (tmp (Tr O))} : \texttt{R} \cap \mathit{Fh}$$

$$\mathscr{C} \vdash_{\mathsf{C1}} ? : \texttt{R} \cap \mathit{Radius} \rightsquigarrow \mathscr{C} \vdash_{\mathsf{C1}} \texttt{fst (cc2pl (cdn (pos (Tr O))))} : \texttt{R} \cap \mathit{Radius}$$

Complexity for Finite and Bounded CL

Theorem (RU TLCA 2011)

For finite combinatory logic FCL*:*

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

Theorem (DMRU CSL 2012)

For bounded combinatory logic BCL_k*:*

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

Staged Composition Synthesis (DMR ESOP 2014)

Goal: Introduce metalanguage L2 *into composition synthesis*

- L1 may be limited, e.g.:
  - L1 might be very low-level
  - L1 might not have a notion of function application
- L2 should contain λ-calculus
  - Special-purpose composition operators
  - Meta-level computation over L1-code
  - Higher-order abstraction
- Challenges
  - How to expose language distinction in types?
  - How to ensure *implementation type correctness*?
  - How to ensure *staged* composition?

Staged Composition Synthesis (DMR ESOP 2014)

Goal: Introduce metalanguage L2 *into composition synthesis*

- L1 may be limited, e.g.:
  - L1 might be very low-level
  - L1 might not have a notion of function application
- L2 should contain λ-calculus
  - Special-purpose composition operators
  - Meta-level computation over L1-code
  - Higher-order abstraction
- Challenges
  - How to expose language distinction in types?
  - How to ensure *implementation type correctness*?
  - How to ensure *staged* composition?

Staged Composition Synthesis (DMR ESOP 2014)

Goal: Introduce metalanguage L2 *into composition synthesis*

- L1 may be limited, e.g.:
  - L1 might be very low-level
  - L1 might not have a notion of function application
- L2 should contain λ-calculus
  - Special-purpose composition operators
  - Meta-level computation over L1-code
  - Higher-order abstraction
- Challenges
  - How to expose language distinction in types?
  - How to ensure *implementation type correctness*?
  - How to ensure *staged* composition?

Staged Composition Synthesis (DMR ESOP 2014)

Goal: Introduce metalanguage L2 *into composition synthesis*

- L1 may be limited, e.g.:
  - L1 might be very low-level
  - L1 might not have a notion of function application
- L2 should contain λ-calculus
  - Special-purpose composition operators
  - Meta-level computation over L1-code
  - Higher-order abstraction
- Challenges
  - How to expose language distinction in types?
  - How to ensure *implementation type correctness*?
  - How to ensure *staged* composition?

Main Ideas in Staged Composition Synthesis

Use modal types $\Box\phi$ *("code of type* ϕ*") to expose language distinction to composition synthesis.*

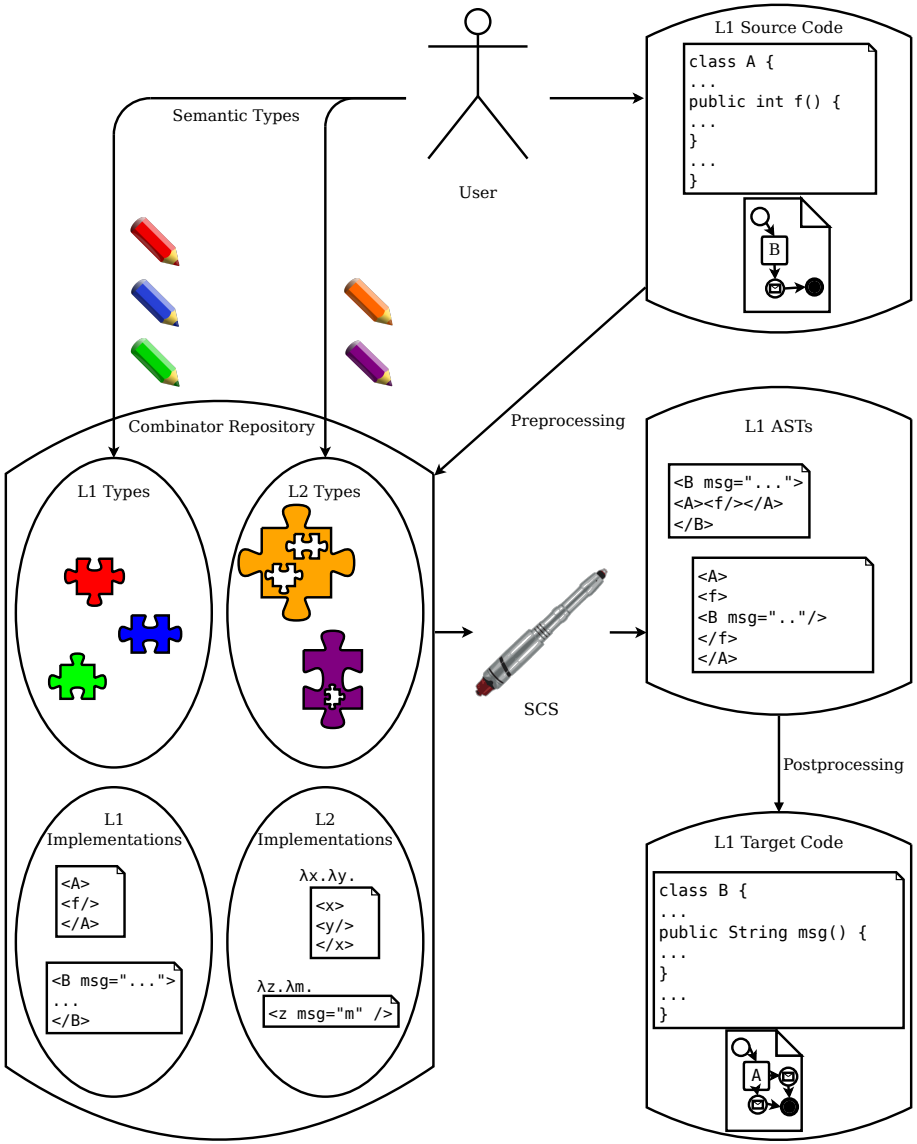
- Introduction of modal intersection types
- Davies and Pfenning's calculus $\lambda_e^{\Box,\to}$ as L2 implementation language
- Challenges of staging solved by theory of $\lambda_e^{\Box,\to}$
- Challenge of implementation type correctness solved by conservative extension theorem
- Extension of inhabitation (semi-)algorithm for modal intersection types
- Extension of **(CL)S**-framework and experiments

Main Ideas in Staged Composition Synthesis

Use modal types $\Box\phi$ *("code of type* ϕ*") to expose language distinction to composition synthesis.*

- Introduction of modal intersection types
- Davies and Pfenning's calculus $\lambda_e^{\Box,\to}$ as L2 implementation language
- Challenges of staging solved by theory of $\lambda_e^{\Box,\to}$
- Challenge of implementation type correctness solved by conservative extension theorem
- Extension of inhabitation (semi-)algorithm for modal intersection types
- Extension of **(CL)S**-framework and experiments

Tool Pipeline



Combinatory Logic Synthesizer Features

- Theorem prover (proofs-as-programs correspondence)
- Combinatory Logic Synthesis for $\mathrm{BCL}_0(\cap, \leq)$
- Version 1.0
  - Proof-of-concept
  - Enumerates inhabitants (even cyclic ones)
  - Variable kinding
  - Atomic subtyping extension for taxonomies
- Version 2.0
  - Algebraic optimizations
  - Co-variant type constructors

Heuristic Optimizations

Strategies

- Algorithm engineering
- Type-theoretic / algebraic optimizations

Experimental Runtime Performance for Γ_n^m in $\mathbb{Z}_n$

(n, m)	Initial (CL)S	Lookahead-(CL)S	Redesigned (CL)S
$(2, 3)$	210 ms	111 ms	93 ms
$(3, 2)$	12504 ms	124 ms	98 ms
$(3, 3)$	–	354 ms	110 ms
$(4, 4)$	–	$7.5 * 10^6$ ms	121 ms
$(7, 7)$	–	–	1063 ms
$(10, 10)$	–	–	54250 ms
$(43, 3)$	–	–	8813 ms

Current & Future Work

- Application to connector synthesis (ArchiType)
- Application to OO-synthesis (mixins, traits, DI)
- Combinatory process synthesis
- Algorithm engineering
- Constrained types $Q(\vec{\alpha}) \Rightarrow \phi$
- Automatic software configuration (OpenNebula Cloud)
- Stratified logics
- Component-oriented synthesis in theorem-proving?
- ...

Application – Spring DI DAO Synthesis

$$\texttt{EmployeeController} : (\Box\texttt{Scope} \cap \alpha) \rightarrow \Box(\texttt{EmployeeDAO} \cap \alpha) \rightarrow \Box(\texttt{EmployeeController} \cap \alpha \cap \texttt{setterTarget})$$

```
EmployeeController :λScope. letbox scope = Scope in
                    λDAO. letbox dao = DAO in
                      box "<bean xsi:type='cls:Constructor'>"
                          "<name>EmployeeController</name>"
                          "<typeName>EmployeeController</typeName>"
                          "<scope>"scope"</scope>"
                          "<argument>"
                          "<typeName>EmployeeDAO</typeName>"
                          "<reference>"dao"</reference>"
                          "</argument>"
                          "</bean>"
```

Application – Spring DI DAO Synthesis

Interface to request Spring Dependency Injection candidates (embedded DSL):

```
public interface InhabitationRequest {
    public Class [] libraryClasses();
    public ConfigurableApplicationContext libraryContext();
    public String classNameExclusionRegexp();
    public Class targetType();
    public String targetScope();
}
```

Instead of using goal type:

$$\square(\texttt{EmployeeController} \cap \texttt{Session} \cap \texttt{setterTarget})$$

Application – Spring DI DAO Synthesis

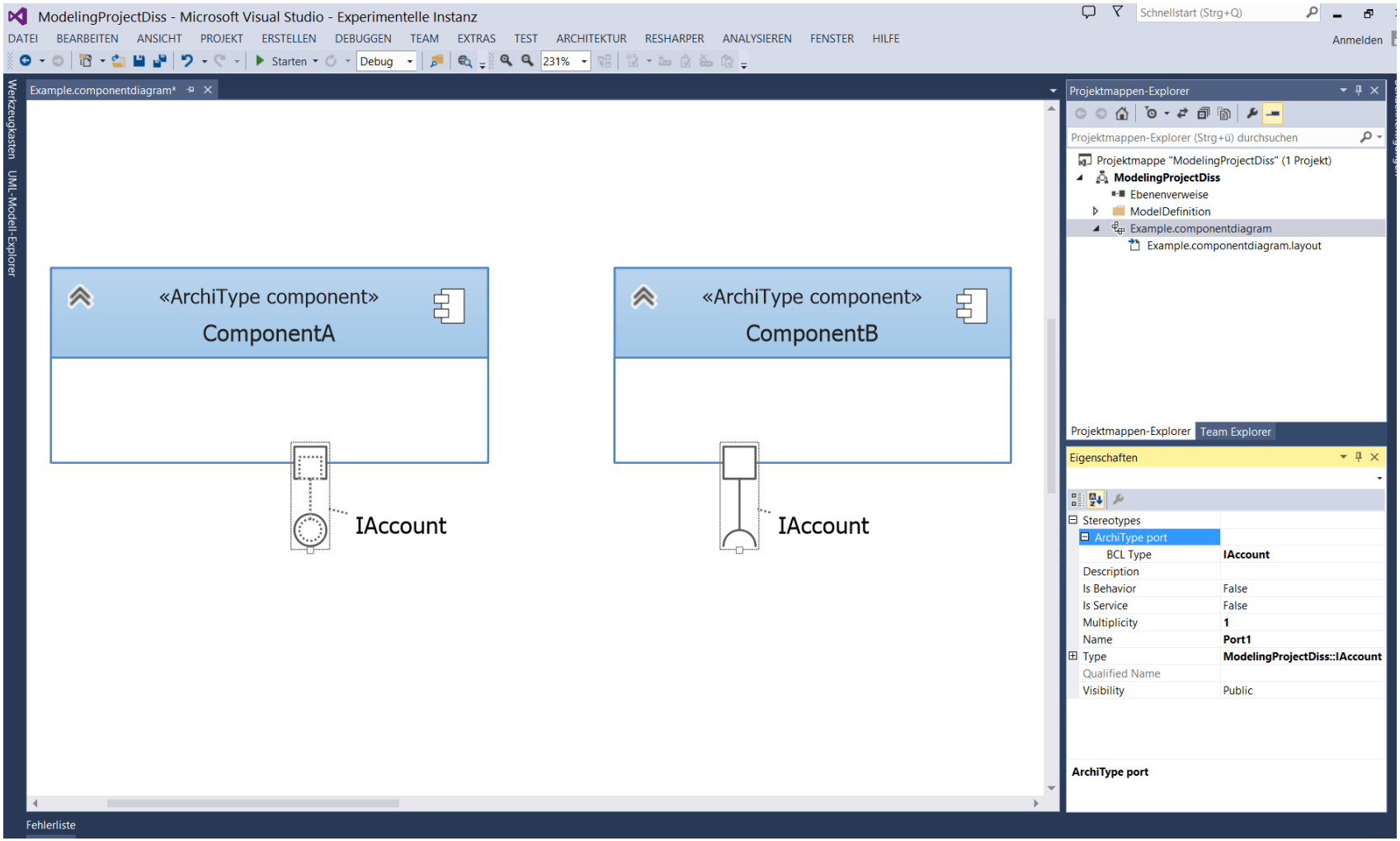
Interface to request Spring Dependency Injection candidates (embedded DSL):

```
public interface InhabitationRequest {
    public Class [] libraryClasses();
    public ConfigurableApplicationContext libraryContext();
    public String classNameExclusionRegexp();
    public Class targetType();
    public String targetScope();
}
```

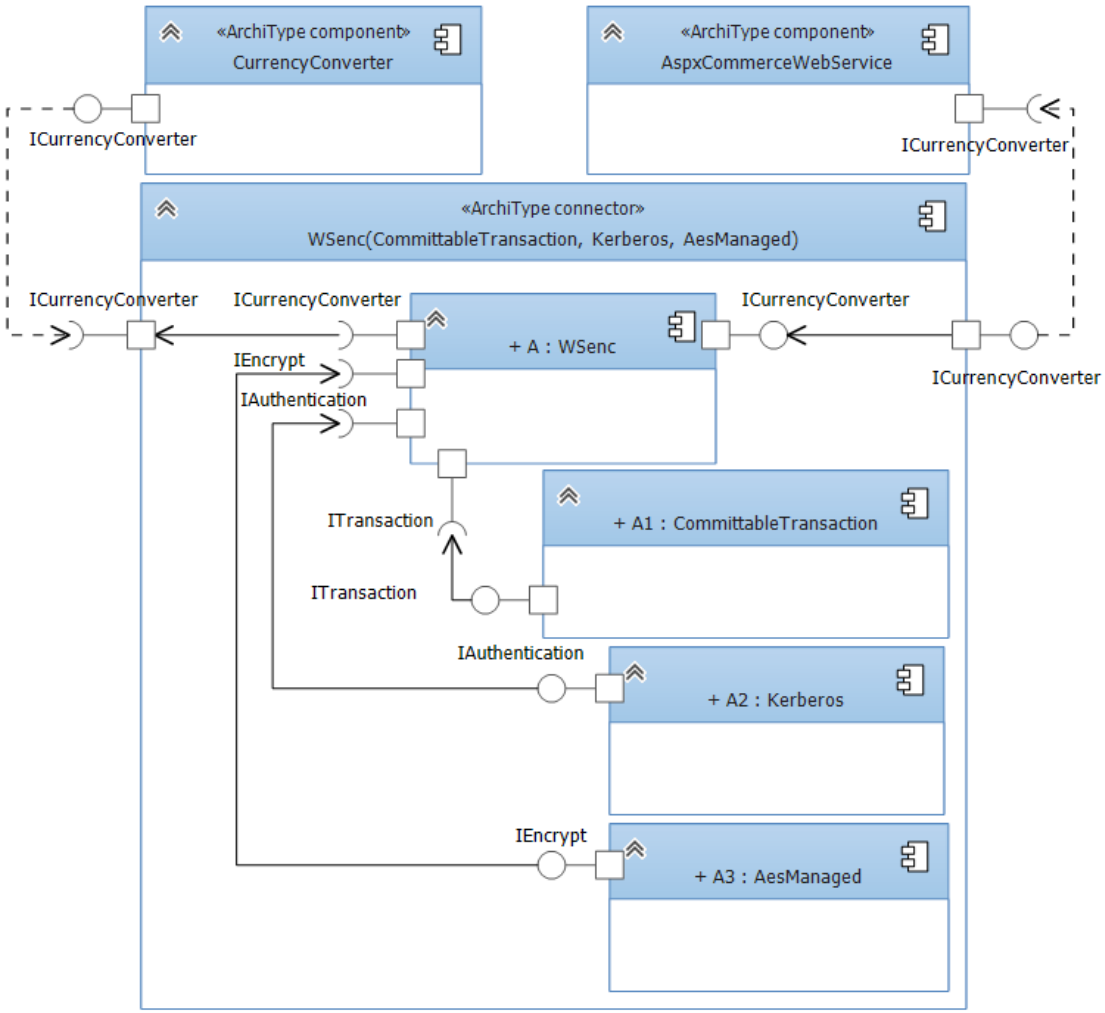
Instead of using goal type:

$$\square(\texttt{EmployeeController} \cap \texttt{Session} \cap \texttt{setterTarget})$$

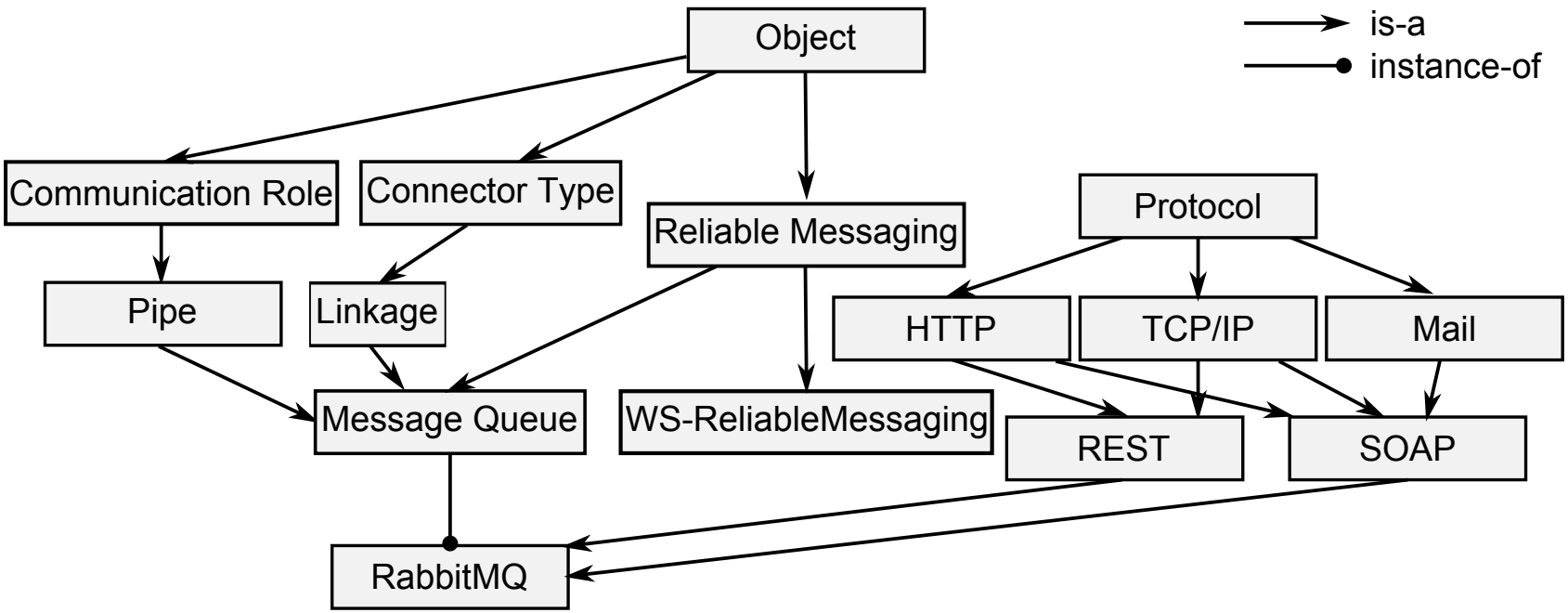
ArchiType


ModelingProjectDiss - Microsoft Visual Studio - Experimentelle Instanz
Example.componentdiagram*
«ArchiType component»
ComponentA
IAccount
«ArchiType component»
ComponentB
IAccount
Projektmappen-Explorer
Projektmappe "ModelingProjectDiss" (1 Projekt)
ModelingProjectDiss
Ebenenverweise
ModelDefinition
Example.componentdiagram
Example.componentdiagram.layout
Eigenschaften
Stereotypes
ArchiType port
BCL Type
IAccount
Description
Is Behavior
False
Is Service
False
Multiplicity
1
Name
Port1
Type
ModelingProjectDiss::IAccount
Qualified Name
Visibility
Public
ArchiType port
Fehlerliste

ArchiType


«ArchiType component»
CurrencyConverter
ICurrencyConverter
«ArchiType component»
AspxCommerceWebService
ICurrencyConverter
«ArchiType connector»
WSenc(CommittableTransaction, Kerberos, AesManaged)
ICurrencyConverter
ICurrencyConverter
ICurrencyConverter
ICurrencyConverter
+ A : WSenc
IEncrypt
IAuthentication
ITransaction
+ A1 : CommittableTransaction
ITransaction
IAuthentication
+ A2 : Kerberos
IEncrypt
+ A3 : AesManaged

ArchiType Taxonomy



ArchiType Experiments

	e-Commerce	ERP	Broker	SC
Subsystems	37	137	11	3
Classes	1293	691	11	3
LLOC	225 763	38 250	839	37
Generated/LLOC	2171	1731	993	257
Time/s	21.98	18.11	7	8.1
Refactoring/LLOC	5	20	No	No

Conclusion

- A new approach to component-oriented synthesis
- Types as logic programs that compute compositions
- Easily applicable to native API's
- Very large set of application scenarios
- Robust foundations in proof theory
- Applied in several experiments and domains

References

- *Staged Composition Synthesis*. With J. Rehof and M. Martens. ESOP 2014.
- *Intersection Type Matching with Subtyping*. With J. Rehof and M. Martens. TLCA 2013.
- *Towards Combinatory Logic Synthesis*. BEAT 2013.
- *Bounded Combinatory Logic*. With J. Rehof, M. Martens and P. Urzyczyn. CSL 2012.
- *Using Inhabitation in Bounded Combinatory Logic with Intersection Types for Composition Synthesis*. With J. Rehof, O.Garbe, M. Martens and P. Urzyczyn. EPTCS 2012.
- *The Complexity of Inhabitation with Explicit Intersection*. J. Rehof and P. Urzyczyn. R.L. Constable and A. Silva (Eds.): Logic and Program Semantics. Essays Dedicated to Dexter Kozen 2012.
- *Finite Combinatory Logic with Intersection Types*. J. Rehof and P. Urzyczyn. TLCA 2011.
- *Automatic Synthesis of Component & Connector-Software Architectures with Bounded Combinatory Logic*. B. Düdder. Dissertation. TU Dortmund 2014.
- **See also**: *Dagstuhl Seminar "Design and Synthesis from Components"*, Schloss Dagstuhl, June 1–6 2014.