# Komponenten- und Service-orientierte Softwarekonstruktion

## Lecture 1: Organisatorisches und Einführung

Jakob Rehof
LS XIV – Software Engineering


TU Dortmund
Sommersemester 2015

SS 2015

# Inhaltsübersicht

# Inhaltsübersicht

# Organisatoren

Vorlesung Prof. Dr. Jakob Rehof

jakob.rehof@cs.tu-dortmund.de

Übung Dr. Boris Düdder

boris.duedder@cs.tu-dortmund.de

# Modulhandbuch

- Modul INF-MSc-312: Komponenten- und Service-Orientierte Softwarekonstruktion
- Zeit: Dienstags, 10-12 Uhr
- Ort: OH12, Raum 2.013
- Webseite: `http://www-seal.cs.tu-dortmund.de/seal/pages/teaching/KSOS/KSOS_15_de.shtml`

# Prüfungsordnungen

## Master (MPO2007/MPO2013)

- Credits: 3 Vorlesung + 3 Übung = 6 CPs
- Modulprüfung: mündliche Prüfung (mind. 30 Minuten)
- Studienleistung: 40% der Punkte und regelmäßige, aktive Teilnahme an der Übung

## Diplom (DPO2001)

- 2 Vorlesung + 2 Übung = 4 SWS = 6 CPs
- Schwerpunktgebiete:
  - (1) Software-Konstruktion
  - (3) Verteilte Systeme

# Zeitplan

| | | |
|---|---|---|
| 1 | 14.04.2015 | Einführung (Komponenten+CLS) |
| 2 | 21.04.2015 | Typentheorie ($\lambda \rightarrow$, CL) |
| 3 | 28.04.2015 | Typentheorie ($\lambda\cap$, CL$\cap$) |
| 4 | 05.05.2015 | Inhabitation ($\lambda \rightarrow$, ATM) |
| 5 | 12.05.2015 | Inhabitation (BCL, theoretischer Algo.) |
| 6 | 19.05.2015 | BCL-Algorithmus (engineered) |
| 7 | 26.05.2015 | Beispiele (Tracking mit Algo.) |
| 8 | 02.06.2015 | Beispiele (2-Counter, BEAT) |
| 9 | 09.06.2015 | $\lambda^{\Box}$ Kalkül (Davies & Pfenning) |
| 10 | 16.06.2015 | Staged Composition Synthesis (SCS) |
| 11 | 23.06.2015 | SCS Beispiel (Tracking, BEAT) + Prozesssynthese |
| 12 | 07.07.2015 | ArchiType |
| 13 | 14.07.2015 | Launchpad |

Achtung: Übung und Vorlesung am 30.06.2015 entfallen.

# Literatur für Vorlesung und Übung

- **(CL)S**-Framework: *Combinatory Logic Synthesizer*. With J. Bessai, A. Dudenhefner, M. Martens and J. Rehof. ISOLA 2014.
- *Staged Composition Synthesis*. With M. Martens and J. Rehof. ESOP 2014.
- *Intersection Type Matching with Subtyping*. With M. Martens and Jakob Rehof. TLCA 2013.
- *Towards Combinatory Logic Synthesis*. J. Rehof. BEAT 2013.
- *Bounded Combinatory Logic*. With M. Martens, J. Rehof and P. Urzyczyn. CSL 2012.
- *Using Inhabitation in Bounded Combinatory Logic with Intersection Types for Composition Synthesis*. With O. Garbe, M. Martens, J. Rehof and P. Urzyczyn. EPTCS 2012.
- *Finite Combinatory Logic with Intersection Types*. P. Urzyczyn and J. Rehof. TLCA 2011.

# Inhaltsübersicht

# Übung

- Zeit: Montag, 12:15-14:00 Uhr
- Raum: OH12, Raum 2.013
- Ausgabe der Übungszettel: Dienstags nach der Vorlesung
- Abgabe: Montag 12 Uhr nach der Ausgabe (normalerweise)
- Online: `http://www-seal.cs.tu-dortmund.de/seal/pages/teaching/KSOS/KSOS_15_de.shtml`
- Sammelabgabe von bis zu **drei** Personen möglich und **erwünscht**

# Zeitplan der Übungen

| | Datum | Ausgabe | Inhalt |
|---|---|---|---|
| 1 | 28.04.2015 | 21.04.2015 | Typentheorie ($\lambda \rightarrow$, CL) |
| 2 | 05.05.2015 | 28.04.2015 | Typentheorie ($\lambda\cap$, CL$\cap$) |
| 3 | 12.05.2015 | 05.05.2015 | Inhabitation ($\lambda \rightarrow$, ATM) |
| 4 | 19.05.2015 | 12.05.2015 | Inhabitation (BCL, theoretischer Algo.) |
| 5 | 26.05.2015 | 19.05.2015 | BCL-Algorithmus (engineered) |
| 6 | 02.06.2015 | 26.05.2015 | Beispiele (Tracking mit Algo.) |
| 7 | 09.06.2015 | 02.06.2015 | Beispiele (2-Counter, BEAT) |
| 8 | 16.06.2015 | 09.06.2015 | $\lambda^{\Box}$ Kalkül (Davies & Pfenning) |
| 9 | 23.06.2015 | 16.06.2015 | Staged Composition Synthesis (SCS) |
| 10 | 07.07.2015 | 23.06.2015 | SCS Beispiel (Tracking, BEAT) |
| 11 | 14.07.2015 | 07.07.2015 | ArchiType |

Achtung: Übung und Vorlesung am 30.06.2015 entfallen.

# Inhaltsübersicht

# Inhaltsübersicht

# Component

- Has no generally accepted formally precise meaning in software
- We will introduce one
- In general, a component is some kind of encapsulated unit of functionality
- We think of a component here as a unit of *composition* (a building block)

# But that could mean many things ...

- Compilation unit (link files, precompiled header files, whole program unit,...)
- Whole applications or systems components (like MS Word, think of Unix pipes, ...)
- A service
- A unit in a UML-model, a UML interface
- In functional programming, a function
- In object-oriented programming, an interface (?)
- A single instruction (in a processor architecture)
- ...
- Here: something called a *combinator*. This Vorlesung is here to tell you what that means.

# Inhaltsübersicht

# Combinatory Logic Synthesis

- A *component-oriented* approach to synthesis
- Rather than doing synthesis "from scratch", we consider synthesis by *composition* from a given set of prefabricated building blocks (components)

# Combinatory Logic Synthesis

- A *component-oriented* approach to synthesis
- Rather than doing synthesis "from scratch", we consider synthesis by *composition* from a given set of prefabricated building blocks (components)
- We will develop a logical and algorithmic framework, in which the notions of components and composition are given precise, mathematical foundations

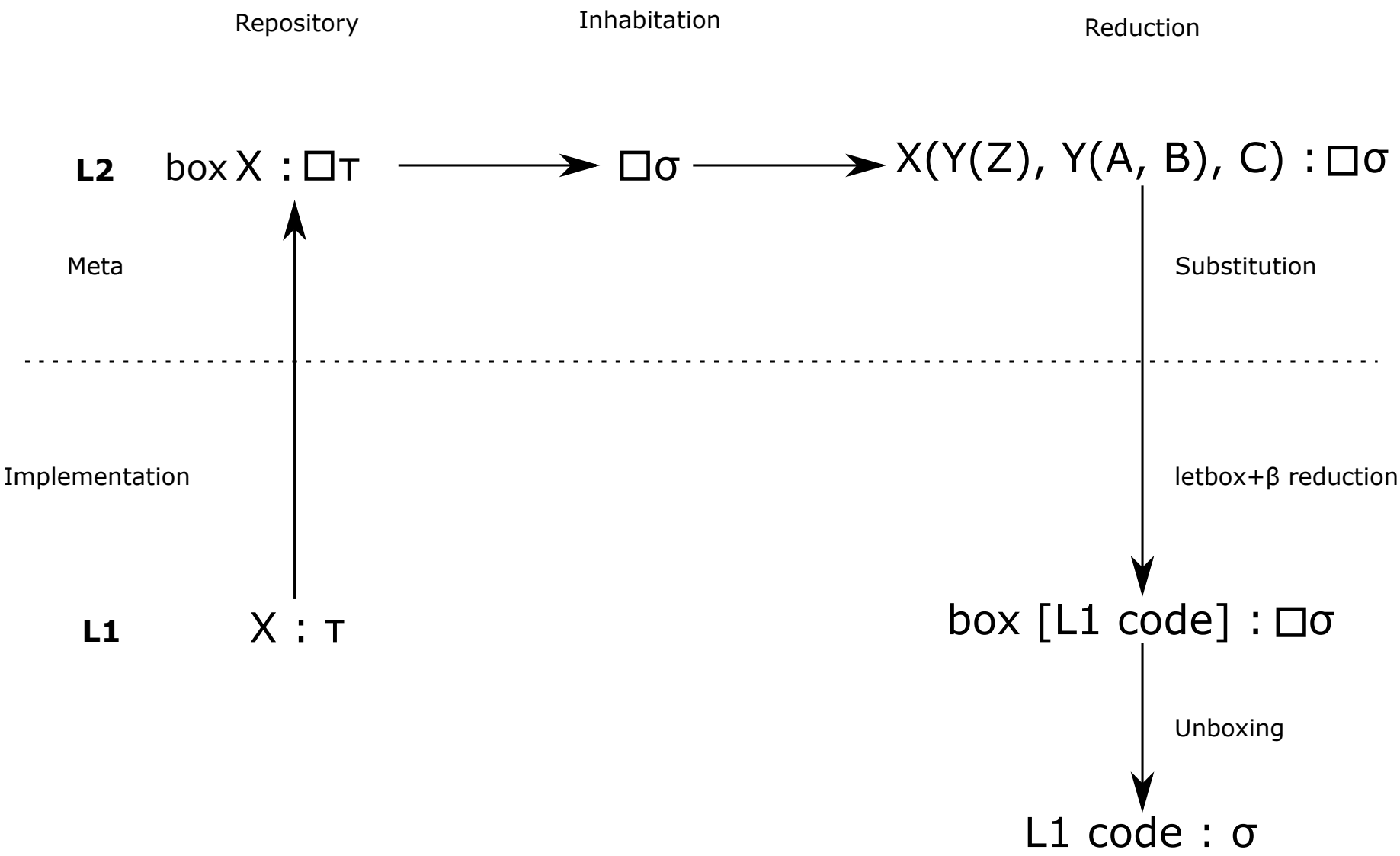
# Combinatory Logic Synthesis

- A *component-oriented* approach to synthesis
- Rather than doing synthesis "from scratch", we consider synthesis by *composition* from a given set of prefabricated building blocks (components)
- We will develop a logical and algorithmic framework, in which the notions of components and composition are given precise, mathematical foundations
- Components will be represented by *combinators* and the formal theory of composition is based on *combinatory logic*

# Combinatory Logic Synthesis

- A *component-oriented* approach to synthesis
- Rather than doing synthesis "from scratch", we consider synthesis by *composition* from a given set of prefabricated building blocks (components)
- We will develop a logical and algorithmic framework, in which the notions of components and composition are given precise, mathematical foundations
- Components will be represented by *combinators* and the formal theory of composition is based on *combinatory logic*
- Components will be exposed to synthesis via enriched *type interfaces*

# Staged Composition Synthesis (SCS): Overview



# Many applications (ongoing research)

- Product lines
- Rapid prototyping
- Architectural synthesis
- Process synthesis
- OO-synthesis
- ...

# Team SEAL

**S**oftware **E**ngineering with **A**lgorithms and **L**ogic

- Jan Bessai (Doktorand)
- Andrej Dudenhefner (Doktorand)
- Boris Düdder (Postdoc)
- Jakob Rehof (Prof.)

# Collaboration

- Theory of inhabitation, with P. Urzyczyn, U Warsaw
- Product lines, with G. Heinemann, WPI Boston
- OO-synthesis, with U. De'Liguoro, U Turin