

# Komponenten- und Service-orientierte Softwarekonstruktion

## Lecture 3: Types and combinators

Jakob Rehof  
LS XIV – Software Engineering



TU Dortmund  
Sommersemester 2015

SS 2015



# Simple types

## Definition 1 (Simple types)

Let  $\mathbb{V}$  denote a denumerable set of *type variables*, ranged over by metavariables  $\alpha, \beta, \gamma, \dots$ , and let  $\mathbb{B}$  range over a set  $\mathbb{B}$  of *type constants*. The set  $\mathbb{T}$  of *simple types*, ranged over by  $\tau, \sigma, \rho, \dots$ , is defined inductively by:

- $\alpha \in \mathbb{V} \Rightarrow \alpha \in \mathbb{T}$
- $b \in \mathbb{B} \Rightarrow b \in \mathbb{T}$
- $\tau \in \mathbb{T}, \sigma \in \mathbb{T} \Rightarrow \tau \rightarrow \sigma \in \mathbb{T}$

We write  $\tau \rightarrow \sigma \rightarrow \rho$  for  $\tau \rightarrow (\sigma \rightarrow \rho)$ .

A *type environment* is a finite set  $\Gamma$  of *type assumptions* of the form

$\Gamma = \{(x_1 : \tau_1), \dots, (x_n : \tau_n)\}$  with  $x_i \not\equiv x_j$  for  $i \neq j$ . We let

$\text{Dm}(\Gamma) = \{x \in \mathbf{V} \mid \exists \tau \in \mathbb{T}. (x : \tau) \in \Gamma\}$ . We write  $\Gamma, x : \tau$  for  $\Gamma \cup \{(x : \tau)\}$ .



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

$$\frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \rightarrow \sigma} (\rightarrow I)$$

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



$$\frac{}{\Gamma, x : \tau \vdash^* x : \tau} (\text{var})$$

$$\frac{\Gamma, x : \tau \vdash^* M : \sigma}{\Gamma \vdash^* \lambda x : \tau. M : \tau \rightarrow \sigma} (\rightarrow\text{I})$$

$$\frac{\Gamma \vdash^* M : \tau \rightarrow \sigma \quad \Gamma \vdash^* N : \tau}{\Gamma \vdash^* MN : \sigma} (\rightarrow\text{E})$$

## Exercise 1

Show by induction on  $M$ : if  $\Gamma \vdash^* M : \sigma$  and  $\Gamma \vdash^* M : \tau$ , then  $\sigma \equiv \tau$ .



# Basis lemma

Let  $\Gamma \downarrow_V = \{(x : \tau) \in \Gamma \mid x \in V\}$ .

## Lemma 2

- If  $\Gamma \subseteq \Gamma'$  then  $\Gamma \vdash M : \tau$  implies  $\Gamma' \vdash M : \tau$ ,
- If  $\Gamma \vdash M : \tau$ , then  $FV(M) \subseteq Dm(\Gamma)$ ,
- If  $\Gamma \vdash M : \tau$ , then  $\Gamma \downarrow_{FV(M)} \vdash M : \tau$ .

## Exercise 2

*Prove Lemma 2.*



# Inversion

## Lemma 3 (Generation lemma)

Suppose that  $\Gamma \vdash M : \tau$ .

- 1 If  $M \equiv x$ , then  $\Gamma = \Gamma', x : \tau$ ,
- 2 If  $M \equiv \lambda x.N$ , then  $\tau \equiv \rho \rightarrow \sigma$  and  $\Gamma, x : \rho \vdash N : \sigma$ ,
- 3 If  $M \equiv PQ$ , then  $\Gamma \vdash P : \sigma \rightarrow \tau$  and  $\Gamma \vdash Q : \sigma$  for some  $\sigma$ .

Proof.

Immediate by inspection of the last rule used in the derivation of  $\Gamma \vdash M : \tau$ . □

## Exercise 3

Call a term  $M$  *typable*, if  $\Gamma \vdash M : \tau$  for some  $\Gamma$  and  $\tau$ .

Show that every subterm of a typable term is typable.



# Substitutivity

A *type substitution* is a map  $S : \mathbb{V} \rightarrow \mathbb{T}$ , and it is lifted homomorphically to a map  $S : \mathbb{T} \rightarrow \mathbb{T}$ . Let  $S(\Gamma) = \{(x : S(\tau)) \mid (x : \tau) \in \Gamma\}$ .

## Lemma 4 (Substitution)

- 1 If  $\Gamma, x : \tau \vdash M : \sigma$  and  $\Gamma \vdash N : \tau$ , then  $\Gamma \vdash M[x := N] : \sigma$ .
- 2 If  $\Gamma \vdash M : \tau$ , then  $S(\Gamma) \vdash M : S(\tau)$ , for any type substitution  $S$ .

## Exercise 4

*Prove Lemma 4 by induction on derivations.*



# Subject reduction

## Theorem 5 (Subject Reduction)

*If  $\Gamma \vdash M : \tau$  and  $M \rightarrow_{\beta} N$ , then  $\Gamma \vdash N : \tau$ .*

*Proof.*

(Sketch) First prove the statement for the case when  $M \rightarrow_{\beta} N$  by reduction of a redex  $R$  in  $M$ , by induction on  $C$  where  $M \equiv C[R]$ , using previous lemmas. Then prove the statement by induction in the length of the reduction  $M \rightarrow_{\beta} N$ . □

## Exercise 5

*Complete the proof sketch of Theorem 5.*





# Strong normalization

## Definition 6

A  $\lambda$ -term  $M$  is called *strongly normalizing*, if every  $\beta$ -reduction sequence starting from  $M$  is finite, and *weakly normalizing*, if there exists a finite reduction sequence starting from  $M$ .

## Theorem 7

Every typable term in  $\lambda^{\rightarrow}$  is strongly normalizing.

## Proof.

See [?].



# Combinators

Let  $\mathcal{C}$  be a set of *combinator symbols*, ranged over by  $X, Y, Z$ . The set  $\Xi_{\mathcal{C}}$  of *combinatory expressions*, ranged over by  $F, G, H$  are defined inductively by:

- $X \in \mathcal{C} \Rightarrow X \in \Xi_{\mathcal{C}}$ ,
- $x \in \mathcal{V} \Rightarrow x \in \Xi_{\mathcal{C}}$ ,
- $F, G \in \Xi_{\mathcal{C}} \Rightarrow (FG) \in \Xi_{\mathcal{C}}$

Consider the set  $\text{SKI} = \{\mathbf{S}, \mathbf{K}, \mathbf{I}\}$  of combinator symbols (referred to as a *combinatory base*) and define the notion of *weak reduction*  $\triangleright_w$  on  $\Xi_{\text{SKI}}$  by setting, for all  $X, Y, Z \in \Xi_{\text{SKI}}$ :

$$\begin{array}{ll} \mathbf{I}F & \triangleright_w F \\ \mathbf{K}FG & \triangleright_w F \\ \mathbf{S}FGH & \triangleright_w (FH)(GH) \end{array}$$

Let  $\rightarrow_w$  and  $\twoheadrightarrow_w$  be the reduction relations on  $\Xi_{\text{SKI}}$  induced by  $\triangleright_w$ , by closing  $\triangleright_w$  under contexts  $C ::= [] \mid CX \mid XC, (X \in \text{SKI})$ .



# Combinatory bases

By choosing different sets  $\mathfrak{B} \subseteq \mathcal{C}$  of combinators (such as  $\mathfrak{B} = SKI = \{\mathbf{S}, \mathbf{K}, \mathbf{I}\}$ ) we can study different combinatory calculi, since in each case we can consider a  $\mathfrak{B}$ -calculus generated from the combinators in  $\mathfrak{B}$ . In such cases, we refer to the set  $\mathfrak{B}$  as a *combinatory base*.

## Exercise 6

*Show that the combinator  $\mathbf{I}$  can be coded in terms of  $\mathbf{S}$  and  $\mathbf{K}$ . In other words, the base  $SKI = \{\mathbf{S}, \mathbf{K}, \mathbf{I}\}$  is redundant. Or, in yet other words, the base  $SK = \{\mathbf{S}, \mathbf{K}\}$  is complete with respect to  $SKI$ -calculus. For this reason, one also talks about  $SK$ -calculus.*

 $\Xi_{\text{SKI}} \mapsto \Lambda$ 

Define the map  $(-)_{\Lambda} : \Xi_{\text{SKI}} \rightarrow \Lambda$  by induction on expressions in  $\Xi_{\text{SKI}}$ :

$$\begin{aligned}(x)_{\Lambda} &\equiv x, \text{ for } x \in \mathcal{V} \\ (\mathbf{I})_{\Lambda} &\equiv \lambda x.x \\ (\mathbf{K})_{\Lambda} &\equiv \lambda xy.x \\ (\mathbf{S})_{\Lambda} &\equiv \lambda xyz.(xz)(yz) \\ (FG)_{\Lambda} &\equiv (F)_{\Lambda}(G)_{\Lambda}\end{aligned}$$

## Proposition 1

*If  $F \rightarrow_w G$ , then  $(F)_{\Lambda} \rightarrow_{\beta} (G)_{\Lambda}$ .*

## Exercise 7

*Prove Proposition 1 by induction on the length of  $F \rightarrow_w G$ .*

$\Lambda \mapsto \Xi_{\text{SKI}}$ 

Define, for each  $x \in \mathcal{V}$ , the “bracket abstraction” map  $[x] : \Xi_{\text{SKI}} \rightarrow \Xi_{\text{SKI}}$  by induction on expressions in  $\Xi_{\text{SKI}}$ :

$$\begin{aligned} [x]x &\equiv \mathbf{I} \\ [x]F &\equiv \mathbf{K}F, \text{ if } x \notin \text{FV}(F) \\ [x](FG) &\equiv \mathbf{S}([x]F)([x]G), \text{ otherwise} \end{aligned}$$

## Proposition 2 (Combinatory completeness)

- 1  $([x]F)G \rightarrow_w F[x := G]$
- 2  $([x]F)_\Lambda \rightarrow_\beta \lambda x.(F)_\Lambda$
- 3  $\forall x \in \mathcal{V}. \forall F \in \Xi_{\text{SKI}}. \exists H \in \Xi_{\text{SKI}}. \forall G \in \Xi_{\text{SKI}}. HG \rightarrow_w F[x := G]$

## Exercise 8

*Prove Proposition 2.*

 $\Lambda \mapsto \Xi_{SKI}$ 

Define the map  $(-)_{\Xi} : \Lambda \rightarrow \Xi_{SKI}$  by induction on  $\lambda$ -terms:

$$\begin{aligned}(x)_{\Xi} &\equiv x, \text{ for } x \in \mathcal{V} \\ (MN)_{\Xi} &\equiv (M)_{\Xi}(N)_{\Xi} \\ (\lambda x.M)_{\Xi} &\equiv [x](M)_{\Xi}\end{aligned}$$

## Proposition 3

For all  $M \in \Lambda$ , one has  $((M)_{\Xi})_{\Lambda} \rightarrow_{\beta} M$ .

## Exercise 9

Prove Proposition 3.



# Combinatory logic SKI

$$\frac{}{\Gamma, x : \tau \vdash_{\text{SKI}} x : \tau} (\text{var})$$

$$\frac{}{\Gamma \vdash_{\text{SKI}} \mathbf{I} : \tau \rightarrow \tau} (\mathbf{I})$$

$$\frac{}{\Gamma \vdash_{\text{SKI}} \mathbf{K} : \tau \rightarrow \sigma \rightarrow \tau} (\mathbf{K})$$

$$\frac{}{\Gamma \vdash_{\text{SKI}} \mathbf{S} : (\tau \rightarrow \sigma \rightarrow \rho) \rightarrow (\tau \rightarrow \sigma) \rightarrow \tau \rightarrow \rho} (\mathbf{S})$$

$$\frac{\Gamma \vdash_{\text{SKI}} F : \tau \rightarrow \sigma \quad \Gamma \vdash_{\text{SKI}} G : \tau}{\Gamma \vdash_{\text{SKI}} (FG) : \sigma} (\rightarrow\text{E})$$

Notice that variables  $x$  have fixed, *monomorphic types*, whereas combinators  $\mathbf{S}$ ,  $\mathbf{K}$ ,  $\mathbf{I}$  have infinitely many types (their types are *schematic* or *polymorphic*). We shall return to this important point in Lecture 5.



# Combinatory logic SKI

## Lemma 8 (Deduction theorem for SKI)

If  $\Gamma, x : \sigma \vdash_{\text{SKI}} F : \tau$ , then  $\Gamma \vdash_{\text{SKI}} [x]F : \sigma \rightarrow \tau$ .

## Proposition 4

- 1 If  $\Gamma \vdash_{\text{SKI}} F : \tau$ , then  $\Gamma \vdash_{\Lambda} (F)_{\Lambda} : \tau$ .
- 2 If  $\Gamma \vdash_{\Lambda} M : \tau$ , then  $\Gamma \vdash_{\text{SKI}} (M)_{\Xi} : \tau$ .

## Exercise 10

*Prove Proposition 4. Hint: The first statement is proven by induction on the derivation of  $\Gamma \vdash_{\text{SKI}} F : \tau$ . The second statement is proven by induction on the derivation of  $\Gamma \vdash_{\Lambda} M : \tau$  using Lemma 8.*





# Decision problems

- *Type checking* ( $\Gamma \vdash M : \tau?$ ). Given  $\Gamma$ ,  $M$  and  $\tau$ , does  $\Gamma \vdash M : \tau$  hold?
- *Typability* ( $? \vdash M : ?$ ). Given  $M$ , are there  $\Gamma$  and  $\tau$  such that  $\Gamma \vdash M : \tau$ ?
- *Inhabitation* ( $\Gamma \vdash ? : \tau$ ). Given  $\Gamma$  and  $\tau$ , does there exist a term (“inhabitant”)  $M$  such that  $\Gamma \vdash M : \tau$ ?

Type checking and typability for  $\lambda^{\rightarrow}$  are linear time solvable.

Inhabitation for  $\lambda^{\rightarrow}$  is PSPACE-complete. See Lecture 4.

**Inhabitation can be seen as a program synthesis problem:**

- Construct *program*  $M$  satisfying *specification*  $\tau$ .



Barendregt, H. P. (1992).

Lambda Calculi with Types.

In *Handbook of Logic in Computer Science*, Eds.: S. Abramsky and D. M. Gabbay and T. S. E. Maibaum, pages 117–309. Oxford University Press.