LMSE

Logische Methoden des Software Engineerings

Prof. Dr. Jakob Rehof

Lehrstuhl XIV, Software Engineering

Diese Vorlesung

- Typentheorie: Einfache Typen
  (Simple typed lambda calculus)

Lesen und Übungen

- Lesen: LCHI, Kap. 3, S. 41-51
- Übungen:
  - LCHI 3.6.1.
  - Übung Folie 8
  - Übung Folie 11
  - Übung Folie 15

Simply typed lambda calculus

- Type systems are deductive systems assigning types to lambda terms
- The typable terms are in many (most) cases strict subsets of the set of untyped terms
- Type systems come in different strengths: a hierarchy with simple types at the bottom
- Two kinds of type systems:
  - *a la Curry* : *implicit* type system (type inference)
  - *a la Church* : *explicit* type system (type checking)

Type expressions

3.1.1. DEFINITION.

(i) Let U denote a denumerably infinite alphabet whose members will be called *type variables*. The set Π of *simple types* is the set of strings defined by the grammar:

$$\Pi ::= U \mid (\Pi \to \Pi)$$

We use $\alpha, \beta, \ldots$ to denote arbitrary type variables, and $\sigma, \tau, \ldots$ to denote arbitrary types. We omit outermost parentheses, and omit other parentheses with the convention that $\to$ associates to the right.

Type expressions

(ii) The set C of *contexts* is the set of all sets of pairs of the form

$$\{x_1 : \tau_1, \ldots, x_n : \tau_n\}$$

with $\tau_1, \ldots, \tau_n \in \Pi$, $x_1, \ldots, x_n \in V$ (variables of Λ) and $x_i \neq x_j$ for $i \neq j$.

(iii) The *domain* of a context $\Gamma = \{x_1 : \tau_1, \ldots, x_n : \tau_n\}$ is defined by:

$$\mathrm{dom}(\Gamma) = \{x_1, \ldots, x_n\}$$

We write $x : \tau$ for $\{x : \tau\}$ and Γ, Γ' for $\Gamma \cup \Gamma'$ if $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma') = \{\}$.

(iv) The *range* of a context $\Gamma = \{x_1 : \tau_1, \ldots, x_n : \tau_n\}$ is defined by:

$$|\Gamma| = \{\tau \in \Pi \mid (x : \tau) \in \Gamma, \text{ for some } x\}.$$

Type system (a la Curry)

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}$$ [Var]

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}$$ [Abs]

$$\frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M\ N : \tau}$$ [App]

Example

3.1.2. EXAMPLE. Let σ, τ, ρ be arbitrary types. Then:

(i) $\vdash \lambda x.x : \sigma \to \sigma$;

(ii) $\vdash \lambda x.\lambda y.x : \sigma \to \tau \to \sigma$;

(iii) $\vdash \lambda x.\lambda y.\lambda z.x\ z\ (y\ z) : (\sigma \to \tau \to \rho) \to (\sigma \to \tau) \to \sigma \to \rho$.

Exercise: Construct type derivations for each of the above judgements

Free variables lemma

3.1.5. LEMMA (Free variables lemma). *Assume that* $\Gamma \vdash M : \sigma$. *Then:*

(i) $\Gamma \subseteq \Gamma'$ *implies* $\Gamma' \vdash M : \sigma$;

(ii) $\mathrm{FV}(M) \subseteq \mathrm{dom}(\Gamma)$;

(iii) $\Gamma' \vdash M : \sigma$ *where* $\mathrm{dom}(\Gamma') = \mathrm{FV}(M)$ *and* $\Gamma' \subseteq \Gamma$.

PROOF. (i) by induction on the derivation of $\Gamma \vdash M : \sigma$.

(ii)-(iii) by induction on the derivation of $\Gamma \vdash M : \sigma$.

Generation lemma

3.1.6. LEMMA (Generation lemma).

(i) $\Gamma \vdash x : \sigma$ *implies* $x : \sigma \in \Gamma$;

(ii) $\Gamma \vdash M\ N : \sigma$ *implies that there is a* τ *such that* $\Gamma \vdash M : \tau \to \sigma$ *and* $\Gamma \vdash N : \tau$.

(iii) $\Gamma \vdash \lambda x.M : \sigma$ *implies that there are* τ *and* ρ *such that* $\Gamma, x : \tau \vdash M : \rho$ *and* $\sigma = \tau \to \rho$.

PROOF. By induction on the length of the derivation. □

Subterm property

A type system $\vdash$ has the subterm property if the following is true:

PROPERTY. *Whenever* $\Gamma \vdash M : \sigma$ *for some* Γ *and* σ*, then it is true for every subterm* N *of* M *that there exists* Γ' *and* σ' *such that* $\Gamma' \vdash N : \sigma'$.

In words: Every subterm of a typable term is typable.

EXERCISE. *Show that the subterm property is true for* $\lambda^{\rightarrow}$.

Type substitution

3.1.7. DEFINITION. The *substitution of type τ for type variable α in type σ*, written $\sigma[\alpha := \tau]$, is defined by:

$$
\begin{array}{lcll}
\alpha[\alpha := \tau] & = & \tau & \\
\beta[\alpha := \tau] & = & \beta & \text{if } \alpha \neq \beta \\
(\sigma_1 \to \sigma_2)[\alpha := \tau] & = & \sigma_1[\alpha := \tau] \to \sigma_2[\alpha := \tau] &
\end{array}
$$

The notation $\Gamma[\alpha := \tau]$ stands for the context $\{(x : \sigma[\alpha := \tau]) \mid (x : \sigma) \in \Gamma\}$.

Substitution lemma

3.1.8. PROPOSITION (Substitution lemma).

(i) *If* $\Gamma \vdash M : \sigma$, *then* $\Gamma[\alpha := \tau] \vdash M : \sigma[\alpha := \tau]$.

(ii) *If* $\Gamma, x : \tau \vdash M : \sigma$ *and* $\Gamma \vdash N : \tau$ *then* $\Gamma \vdash M[x := N] : \sigma$.

PROOF. By induction on the derivation of $\Gamma \vdash M : \sigma$ and generation of $\Gamma, x : \tau \vdash M : \sigma$, respectively. □

- The substitution property is a limited form of polymorphism

Subject reduction

3.1.9. Proposition (Subject reduction). *If* $\Gamma \vdash M : \sigma$ *and* $M \to_\beta N$, *then* $\Gamma \vdash N : \sigma$.

Proof. By induction on the derivation of $M \to_\beta N$ using the substitution lemma and the generation lemma. □

3.1.11. Corollary. *If* $\Gamma \vdash M : \sigma$ *and* $M \twoheadrightarrow_\beta N$, *then* $\Gamma \vdash N : \sigma$.

Subject expansion

Subject expansion for a type system $\vdash$ is the following property:

$$(\Gamma \vdash N : \sigma \wedge M \twoheadrightarrow_\beta N) \;\Rightarrow\; (\Gamma \vdash N : \sigma)$$

EXERCISE. *Show that subject expansion fails for* $\lambda^{\to}$.

Simple types a la Church

3.2.1. DEFINITION.

(i) The set Λ_Π of pseudo-terms is defined by the following grammar:

$$\Lambda_\Pi ::= V \mid (\lambda x{:}\,\Pi\, \Lambda_\Pi) \mid (\Lambda_\Pi\ \Lambda_\Pi)$$

where V is the set of (λ-term) variables and Π is the set of simple types.[1] We adopt the same terminology, notation, and conventions for pseudo-terms as for λ-terms, see 1.3–1.10, mutatis mutandis.

(ii) The *typability* relation $\Vdash$ on $C \times \Lambda_\Pi \times \Pi$ is defined by:[2]

$$\frac{}{\Gamma, x:\tau \vdash^* x:\tau} \qquad \frac{\Gamma, x:\sigma \vdash^* M:\tau}{\Gamma \vdash^* \lambda x{:}\sigma.M : \sigma \to \tau} \qquad \frac{\Gamma \vdash^* M:\sigma \to \tau \quad \Gamma \vdash^* N:\sigma}{\Gamma \vdash^* M\ N:\tau}$$

where we require that $x \notin \mathrm{dom}(\Gamma)$ in the first and second rule.

(iii) The simply typed λ-calculus à la Church ($\lambda{\to}$ *à la Church,* for short) is the triple $(\Lambda_\Pi, \Pi, \Vdash)$.

(iv) If $\Gamma \vdash^* M:\sigma$ then we say that M *has type* σ *in* Γ. We say that $M \in \Lambda_\Pi$ is *typable* if there are Γ and σ such that $\Gamma \vdash^* M:\sigma$.

Example

3.2.2. EXAMPLE. Let σ, τ, ρ be arbitrary simple types. Then:

(i) $\vdash^* \lambda x{:}\sigma.x : \sigma \to \sigma$;

(ii) $\vdash^* \lambda x{:}\sigma.\lambda y{:}\tau.x : \sigma \to \tau \to \sigma$;

(iii) $\vdash^* \lambda x{:}\sigma{\to}\tau{\to}\rho.\lambda y{:}\sigma{\to}\tau.\lambda z{:}\sigma.(x\ z)\ y\ z : (\sigma{\to}\tau{\to}\rho){\to}(\sigma{\to}\tau){\to}\sigma{\to}\rho.$

Special properties of Church-system

3.2.12. Proposition (Uniqueness of types).

(i) *If* $\Gamma \vdash^* M : \sigma$ *and* $\Gamma \vdash^* M : \tau$ *then* $\sigma = \tau$.

(ii) *If* $\Gamma \vdash^* M : \sigma$ *and* $\Gamma \vdash^* N : \tau$ *and* $M =_\beta N$, *then* $\sigma = \tau$.