

Tutorium 09: let-Polymorphismus

David Kaufmann

09. Januar 2023

Tutorium Programmierparadigmen am KIT

Rückblick

Bisherige Themen:

- Haskell: Funktionale Programmierung
- Lambda-Kalkül: Beta-Reduktion, Church-Encodings
- Typisierung: Lambda-Typen, Let, Typinferenz
- Prolog: Logische Programmierung

Ab jetzt:

- Parallelprogrammierung: MPI, Java
- Design by Contract: OpenJML
- Compiler: Parser + Java ByteCode

ÜB 7/8

Let-Polymorphism

Let-Polymorphism

Let-Polymorphismus: Motivation

$$\lambda f. f f$$

- Diese Funktion verwendet f auf zwei Arten:
 - $\alpha \rightarrow \alpha$: Rechte Seite.
 - $(\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)$: Linke Seite, nimmt f als Argument und gibt es zurück.

Let-Polymorphismus: Motivation

$$\lambda f. f f$$

- Diese Funktion verwendet f auf zwei Arten:
 - $\alpha \rightarrow \alpha$: Rechte Seite.
 - $(\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)$: Linke Seite, nimmt f als Argument und gibt es zurück.
- Problem: $\alpha \rightarrow \alpha$ und $(\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)$ sind nicht unifizierbar!
 - „occurs check“: α darf sich nicht selbst einsetzen.
- Idee: Bei jeder Verwendung eines polymorphen Typen erzeugen wir *neue Typvariablen*, um diese Beschränkung zu umgehen.

- Idee: Bei jeder Verwendung eines polymorphen Typen erzeugen wir *neue Typvariablen*, um diese Beschränkung zu umgehen.
- Ein *Typschema* ist ein Typ, in dem manche Typvariablen allquantifiziert sind:

$$\phi = \forall \alpha_1. \dots \forall \alpha_n. \tau$$
$$\alpha_i \in FV(\tau)$$

Typschemata und Instanziierung

- Ein Typschema spannt eine Menge von Typen auf, mit denen es *instanziiert* werden kann:

$$\forall \alpha. \alpha \rightarrow \alpha \succeq \text{int} \rightarrow \text{int}$$

$$\forall \alpha. \alpha \rightarrow \alpha \succeq \tau \rightarrow \tau$$

$$\forall \alpha. \alpha \rightarrow \alpha \not\succeq \tau \rightarrow \sigma$$

$$\forall \alpha. \alpha \rightarrow \alpha \not\succeq \tau \rightarrow \tau \rightarrow \tau$$

$$\forall \alpha. \alpha \rightarrow \alpha \succeq (\tau \rightarrow \tau) \rightarrow (\tau \rightarrow \tau)$$

Def. aus VL: Für $n \in \mathbb{N}_0$ heist $\forall \alpha_1 \dots \forall \alpha_n. \tau$ **Typscheema**. Es bindet freie Typvariablen $\alpha_1, \dots, \alpha_n$ in τ .

Def. aus VL: Für Typen τ_1, \dots, τ_n ist der Typ $\tau[\alpha_1 \mapsto \tau_1, \dots, \alpha_n \mapsto \tau_n]$ eine **Instanziierung** vom Typscheema $\forall \alpha_1 \dots \forall \alpha_n. \tau$.

Schreibweise: $\forall \alpha_1 \dots \forall \alpha_n. \tau \succeq \tau[\alpha_1 \mapsto \tau_1, \dots, \alpha_n \mapsto \tau_n]$

Das Typscheema $ta(\tau, \Gamma) = \forall \alpha_1 \dots \forall \alpha_n. \tau$ heißt **Typabstraktion** von τ relativ zu Γ , wobei $\alpha_i \in FV(\tau) \setminus FV(\Gamma)$

Um Typschemata bei der Inferenz zu verwenden, müssen wir zunächst die Regel für Variablen anpassen:

$$\frac{\Gamma(x) = \phi \quad \phi \succeq_{\text{frische } \alpha_i} \tau}{\Gamma \vdash x : \alpha_j} \text{VAR}$$

Constraint: $\{\alpha_j = \tau\}$

- $\succeq_{\text{frische } \alpha_i}$ instanziiert ein Typschema mit α_i , die noch nicht im Baum vorkommen.
- Jetzt brauchen wir noch eine Möglichkeit, Typschemata zu erzeugen.

Let-Polymorphismus

Mit einem LET-Term wird ein Typschema eingeführt:

$$\frac{\Gamma \vdash t_1 : \alpha_i \quad \Gamma' \vdash t_2 : \alpha_j}{\Gamma \vdash \text{let } x = t_1 \text{ in } t_2 : \alpha_k} \text{LET}$$

$$\sigma_{\text{let}} = \text{mgu}(C_{\text{let}})$$

$$\Gamma' = \sigma_{\text{let}}(\Gamma), x : \text{ta}(\sigma_{\text{let}}(\alpha_i), \sigma_{\text{let}}(\Gamma))$$

$$C'_{\text{let}} = \{\alpha_n = \sigma_{\text{let}}(\alpha_n) \mid \sigma_{\text{let}}(\alpha_n) \text{ ist definiert}\}$$

$$\text{Constraints: } C'_{\text{let}} \cup C_{\text{body}} \cup \{a_j = a_k\}$$

$let\ f = \lambda x.2\ in\ f(f\ true) : int$

Gegeben:

$\lambda x.2 : \alpha \rightarrow int$

$\forall \alpha. \alpha \rightarrow int \succeq bool \rightarrow int$ oder $\forall \alpha. \alpha \rightarrow int \succeq int \rightarrow int$

Let-Typregel

$$\text{LET} \frac{\Gamma \vdash t_1 : \tau_1 \quad \Gamma, x : ta(\tau_1, \Gamma) \vdash t_2 : \tau_2}{\Gamma \vdash \text{let } x = t_1 \text{ in } t_2 : \tau_2}$$

Angepasste Regeln:

$$\text{VAR} \frac{\Gamma(x) = \phi \quad \phi \succeq \tau}{\Gamma \vdash x : \tau}$$

Herleitung

- Herleitungsbaum aufstellen (mit den Regeln und frischen Variablen für Typen in den Voraussetzungen)
- Sammle dabei Constraints, die erfüllt sein müssen damit der Herleitungsbaum gültig ist
- Bestimme mgu des Gleichungssystems (Robinson Algorithmus)

Typinferenz mit let-Regel

- Betrachte zunächst nur den linken Teilbaum, um τ_1 zu berechnen
 - Constraints: C_{let}
 - $\sigma_{\text{let}} = \text{mgu}(C_{\text{let}})$
- Berechne $\Gamma' := \sigma_{\text{let}}(\Gamma), x : ta(\sigma_{\text{let}}(\tau_1), \sigma_{\text{let}}(\Gamma))$
- Benutze Γ' im rechten Teilbaum, sammle Constrains in C_{body}
- Ergebnisconstraints sind $C'_{\text{let}} \cup C_{\text{body}}$ mit $C'_{\text{let}} := \{\alpha_n = \sigma_{\text{let}}(\alpha_n) \mid \sigma_{\text{let}} \text{ definiert für } \alpha_n\}$

Beispiel: Let-Polymorphismus

$$\vdash \text{let } f = \lambda x. x \text{ in } f f : \alpha_1$$

Beispiel: Let-Polymorphismus

$$\begin{array}{c}
 \frac{\dots}{\vdash \lambda x. x : \alpha_2} \text{ABS} \quad \frac{\frac{\Gamma'(f) = \forall \alpha_5. \alpha_5 \rightarrow \alpha_5 \quad \succeq \alpha_8 \rightarrow \alpha_8}{\Gamma' \vdash f : \alpha_6} \text{VAR} \quad \frac{\Gamma'(f) = \forall \alpha_5. \alpha_5 \rightarrow \alpha_5 \quad \succeq \alpha_9 \rightarrow \alpha_9}{\Gamma' \vdash f : \alpha_7} \text{VAR}}{\Gamma' \vdash f f : \alpha_3} \text{APP} \\
 \hline
 \vdash \text{let } f = \lambda x. x \text{ in } f f : \alpha_1 \text{LET}
 \end{array}$$

$$C_{\text{let}} = \{\alpha_2 = \alpha_4 \rightarrow \alpha_5, \alpha_4 \rightarrow \alpha_5\}$$

$$\sigma_{\text{let}} = [\alpha_2 \dot{\rightarrow} \alpha_5 \rightarrow \alpha_5, \alpha_4 \dot{\rightarrow} \alpha_5]$$

$$\Gamma' = x : \forall \alpha_5. \alpha_5 \rightarrow \alpha_5$$

$$C'_{\text{let}} = \{\alpha_2 = \alpha_5 \rightarrow \alpha_5, \alpha_4 = \alpha_5\}$$

$$C_{\text{body}} = \{\alpha_6 = \alpha_7 \rightarrow \alpha_3, \alpha_6 = \alpha_8 \rightarrow \alpha_8, \alpha_7 = \alpha_9 \rightarrow \alpha_9\}$$

$$C = C'_{\text{let}} \cup C_{\text{body}} \cup \{\alpha_3 = \alpha_1\}$$

Klausuraufgaben
