Tutorium 09: `let`-Polymorphismus

David Kaufmann

09. Januar 2023

Tutorium Programmierparadigmen am KIT

Rückblick

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-Polymorphismus

Let-Polymorphismus: Motivation

$$\lambda f\,.\,f\ f$$

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

Let-Polymorphismus: Motivation

$$\lambda f\,.\,f\ f$$

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

Typschemata und Instanziierung

- 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:

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

Typschemata und Instanziierung

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

$$\forall\alpha.\alpha \to \alpha \succeq \mathsf{int} \to \mathsf{int}$$
$$\forall\alpha.\alpha \to \alpha \succeq \tau \to \tau$$
$$\forall\alpha.\alpha \to \alpha \not\succeq \tau \to \sigma$$
$$\forall\alpha.\alpha \to \alpha \not\succeq \tau \to \tau \to \tau$$
$$\forall\alpha.\alpha \to \alpha \succeq (\tau \to \tau) \to (\tau \to \tau)$$

Zusammenfassung

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

Def. aus VL: Für Typen $\tau_1, \ldots, \tau_n$ ist der Typ $\tau[\alpha_1 \mapsto \tau_1, \ldots, \alpha_n \mapsto \tau_n]$ eine **Instanziierung** vom Typschema $\forall\alpha_1. \ldots \forall\alpha_n.\tau$.
Schreibweise: $\forall\alpha_1. \ldots \forall\alpha_n.\tau \succeq \tau[\alpha_1 \mapsto \tau_1, \ldots, \alpha_n \mapsto \tau_n]$

Das Typschema $ta(\tau, \Gamma) = \forall\alpha_1. \ldots \forall\alpha_n.\tau$ heißt **Typabstraktion** von τ relativ zu Γ, wobei $\alpha_i \in \mathrm{FV}(\tau) \setminus \mathrm{FV}(\Gamma)$

Let-Polymorphismus

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

$$\frac{\Gamma(x) = \phi \qquad \phi \succeq_{\text{frische } \alpha_i} \tau}{\Gamma \vdash x : \alpha_j}\,\textsc{Var}$$

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 einen LET-Term wird ein Typschema eingeführt:

$$\frac{\Gamma \vdash t_1 : \alpha_i \qquad \Gamma' \vdash t_2 : \alpha_j}{\Gamma \vdash \texttt{let}\ x = t_1\ \texttt{in}\ t_2 : \alpha_k}\,\textsc{Let}$$

$$\begin{aligned}
\sigma_{let} &= mgu(C_{let}) \\
\Gamma' &= \sigma_{let}(\Gamma), x : ta(\sigma_{let}(\alpha_i), \sigma_{let}(\Gamma)) \\
C'_{let} &= \{\alpha_n = \sigma_{let}(\alpha_n) \mid \sigma_{let}(\alpha_n) \text{ ist definiert}\}
\end{aligned}$$

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

Übung

$$\mathit{let}\ f = \lambda x.2\ \mathit{in}\ f\,(f\ \mathit{true}) : \text{int}$$

Gegeben:

$\lambda x.2 : \alpha \to \text{int}$

$\forall\alpha.\,\alpha \to \text{int} \succeq \text{bool} \to \text{int}$ oder $\forall\alpha.\,\alpha \to \text{int} \succeq \text{int} \to \text{int}$

Let-Typregel

$$\textsc{Let}\ \frac{\Gamma \vdash t_1 : \tau_1 \qquad \Gamma, x : ta(\tau_1, \Gamma) \vdash t_2 : \tau_2}{\Gamma \vdash \texttt{let}\ x = t_1\ \texttt{in}\ t_2 : \tau_2}$$

Angepasste Regeln:

$$\textsc{Var}\ \frac{\Gamma(x) = \phi \qquad \phi \succeq \tau}{\Gamma \vdash x : \tau}$$

Herleitung

Typinferenz ohne `let`

- 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 $C'_{\text{let}} \cup C_{\text{body}} \cup$ mit $C'_{\text{let}} := \{\alpha_n = \sigma_{\text{let}}(\alpha_n) | \sigma_{\text{let}} \text{ definiert für } \alpha_n\}$

Beispiel: Let-Polymorphismus

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

Beispiel: Let-Polymorphismus

$$\dfrac{\dfrac{\dfrac{\dots}{\vdash \lambda x.\, x : \alpha_2}\textsc{Abs} \qquad \dfrac{\dfrac{\Gamma'(f) = \forall\alpha_5.\alpha_5 \to \alpha_5 \quad \succeq \alpha_8 \to \alpha_8}{\Gamma' \vdash f : \alpha_6}\textsc{Var} \qquad \dfrac{\Gamma'(f) = \forall\alpha_5.\alpha_5 \to \alpha_5 \quad \succeq \alpha_9 \to \alpha_9}{\Gamma' \vdash f : \alpha_7}\textsc{Var}}{\Gamma' \vdash f\ f : \alpha_3}\textsc{App}}{\vdash \texttt{let}\ \ f = \lambda x.\, x\ \ \texttt{in}\ \ f\ f : \alpha_1}\textsc{Let}$$

$$\begin{aligned}
C_{let} &= \{\alpha_2 = \alpha_4 \to \alpha_5, \alpha_4 \to \alpha_5\} \\
\sigma_{let} &= [\alpha_2 \Rightarrow \alpha_5 \to \alpha_5, \alpha_4 \Rightarrow \alpha_5] \\
\Gamma' &= x : \forall\alpha_5.\alpha_5 \to \alpha_5 \\
C'_{let} &= \{\alpha_2 = \alpha_5 \to \alpha_5, \alpha_4 = \alpha_5\} \\
C_{body} &= \{\alpha_6 = \alpha_7 \to \alpha_3, \alpha_6 = \alpha_8 \to \alpha_8, \alpha_7 = \alpha_9 \to \alpha_9\} \\
C &= C'_{let} \cup C_{body} \cup \{\alpha_3 = \alpha_1\}
\end{aligned}$$

Klausuraufgaben