Friedrich-Alexander-Universität
Technische Fakultät

FAU

Theorie der Programmierung

Übung 10 - Der einfach getypte λ-Kalkül

Leon Vatthauer

12. Juni 2023

Der einfach getypte λ-Kalkül $(\lambda\rightarrow)$

Definitionen

Typen

Sei **V** eine Menge von *Typvariablen* *a*, *b*, etc. und **B** eine Menge von Basistypen, etwa **Bool**, **Int**. Die Grammatik für Typen $\alpha, \beta, \ldots$ ist dann

$$\alpha, \beta ::= \mathbf{a} \mid \mathbf{b} \mid \alpha \rightarrow \beta \qquad (\mathbf{b} \in \mathbf{B}, \mathbf{a} \in \mathbf{V})$$

Der einfach getypte λ-Kalkül ($\lambda\!\rightarrow$)

Definitionen

Typen

Sei **V** eine Menge von *Typvariablen* *a*, *b*, etc. und **B** eine Menge von Basistypen, etwa **Bool**, **Int**. Die Grammatik für Typen $\alpha, \beta, \ldots$ ist dann

$$\alpha, \beta ::= \mathit{a} \mid \mathbf{b} \mid \alpha \to \beta \qquad (\mathbf{b} \in \mathbf{B}, \mathit{a} \in \mathbf{V})$$

Notation

Im Gegensatz zur Applikation von λ-Termen, welche links-assoziativ ist (also *add* 5 3 = (*add* 5) 3), ist der Funktionspfeil rechts-assoziativ, also $\alpha \to \beta \to \gamma = \alpha \to (\beta \to \gamma)$

Der einfach getypte λ-Kalkül ($\lambda\!\!\rightarrow$)

Definitionen

Typen

Sei **V** eine Menge von *Typvariablen* *a*, *b*, etc. und **B** eine Menge von Basistypen, etwa **Bool**, **Int**. Die Grammatik für Typen $\alpha, \beta, \ldots$ ist dann

$$\alpha, \beta ::= \mathit{a} \mid \mathbf{b} \mid \alpha \to \beta \qquad (\mathbf{b} \in \mathbf{B}, \mathit{a} \in \mathbf{V})$$

Notation

Im Gegensatz zur Applikation von λ-Termen, welche links-assoziativ ist (also *add* 5 3 = (*add* 5) 3), ist der Funktionspfeil rechts-assoziativ, also $\alpha \to \beta \to \gamma = \alpha \to (\beta \to \gamma)$

Kontext

Ein *(Typ-)Kontext* ist eine Menge

$$\Gamma = \{x_1 : \alpha_1; \ldots; x_n : \alpha_n\}$$

Der einfach getypte λ-Kalkül ($\lambda\!\rightarrow$)

Definitionen

Typisierung

Wir lesen $\Gamma \vdash t : \alpha$ als „im Kontext Γ hat der Term t den Typ α“ und definieren diese Relation wie folgt:

$$(\text{Ax})\ \frac{}{\Gamma \vdash x : \alpha}\ (x : \alpha \in \Gamma) \qquad (\rightarrow_i)\ \frac{\Gamma[x \mapsto \alpha] \vdash t : \beta}{\Gamma \vdash \lambda x.t : \alpha \rightarrow \beta}$$

$$(\rightarrow_e)\ \frac{\Gamma \vdash t : \alpha \rightarrow \beta \qquad \Gamma \vdash s : \alpha}{\Gamma \vdash t\ s : \beta}$$

Aufgabe 1

Typprüfung einfach getypter Terme

Zeigen Sie, dass die folgenden Aussagen zutreffen, indem Sie jeweils eine korrekte Typherleitung angeben.

1. $x : \texttt{int}, \mathit{add} : \texttt{int} \to \texttt{int} \to \texttt{int} \vdash \lambda y.\mathit{add}\ x\ (\mathit{add}\ x\ y) : \texttt{int} \to \texttt{int}$
2. $\vdash \lambda xy.x : \alpha \to \beta \to \alpha$, für alle Typen α, β

Typisierung

Wir lesen $\Gamma \vdash t : \alpha$ als „im Kontext Γ hat der Term t den Typ α“ und definieren diese Relation wie folgt:

$$(\mathsf{Ax})\ \frac{}{\Gamma \vdash x : \alpha}\ (x : \alpha \in \Gamma) \qquad (\to_i)\ \frac{\Gamma[x \mapsto \alpha] \vdash t : \beta}{\Gamma \vdash \lambda x.t : \alpha \to \beta}$$

$$(\to_e)\ \frac{\Gamma \vdash t : \alpha \to \beta \qquad \Gamma \vdash s : \alpha}{\Gamma \vdash t\ s : \beta}$$

Aufgabe 2

Church-Kodierung von Booleans

Funktionale Sprachen fügen weiterhin dem λ-Kalkül eingebaute Typen (built-in types) hinzu und erlauben auch die Definition von benutzerdefinierten Typen (user-defined types). Diese Typen können jedoch prinzipiell allesamt unter Einsatz der sogenannten Church-Kodierung direkt im λ-Kalkül kodiert werden.

Wir werden uns auf dem kommenden Übungsblatt noch genauer mit derartigen Typen befassen, in diesem Übungsblatt soll es zunächst nur um die λ-Terme selbst gehen. Die beiden Wahrheitswerte und die Fallunterscheidung werden etwa als λ-Terme wie folgt definiert:

```
true = λx y. x
false = λx y. y
ite = λb x y. b x y
```

Aufgabe 2

Church-Kodierung von Booleans

Wir kodieren Booleans als λ-Terme wie folgt:

```
true = λx y. x
false = λx y. y
ite = λb x y. b x y
```

1. Zeigen Sie, dass für alle λ-Terme s und t gilt:

$$\text{ite true } s\ t \to_\beta^* s \qquad \text{ite false } s\ t \to_\beta^* t$$

Aufgabe 2

Church-Kodierung von Booleans

Wir kodieren Booleans als λ-Terme wie folgt:

```
true = λx y. x
false = λx y. y
ite = λb x y. b x y
```

1. Zeigen Sie, dass für alle λ-Terme s und t gilt:

$$\text{ite true } s\ t \to_\beta^* s \qquad \text{ite false } s\ t \to_\beta^* t$$

2. Vervollständigen Sie die folgenden Funktionsdefinitionen, so dass sie boolesche Negation, Konjunktion, exklusives Oder und Implikation berechnen:

```
bot = λb. ...
and = λb₁ b₂. ...
xor = λb₁ b₂. ...
imp = λb₁ b₂. ...
```

Notation: Von nun an schreiben wir “**if** s **then** t **else** u” anstelle von “ite s t u”

Aufgabe 3

Church-Numerale

Eine natürliche Zahl n wird durch einen λ-Term $\lceil n \rceil$ kodiert, der eine gegebene Funktion f wiederholt n-mal auf einen Startwert a anwendet, was wir informell als n „Iterationen“ von f startend bei a ansehen können:

$$\lceil n \rceil := \lambda f\ a.\ \underbrace{f(f(f(\ldots f}_{n}\ a))) \tag{1}$$

Wir kodieren die Konstruktoren von $\mathbb{N}$ wie folgt:

$$\begin{aligned} \mathit{zero} &= \lambda f\ a.a \\ \mathit{succ}\ n &= \lambda f\ a.f\ (n\ f\ a) \end{aligned}$$

Aufgabe 3.1

Church-Numerale

Zeigen Sie, dass für jede natürliche Zahl n gilt: $succ\ \lceil n \rceil \rightarrow_\beta^* \lceil n+1 \rceil$

Church-Numerale

$$\lceil n \rceil := \lambda f\ a.\ \underbrace{f(f(f(\ldots f}_{n}\ a))) \tag{1}$$

$$zero = \lambda f\ a.a \qquad succ = \lambda n\ f\ a.f\ (n\ f\ a)$$

Aufgabe 3.2

Church-Numerale

Definieren Sie die Addition, Multiplikation und Exponentiation natürlicher Zahlen bezüglich dieser Kodierung. Vervollständigen Sie also die folgenden Definitionen:

$$add = \lambda m\ n. \ldots \qquad mult = \lambda m\ n. \ldots \qquad pow = \lambda m\ n. \ldots$$

derart, dass für alle x und y gilt:

$$add\ \lceil x \rceil\ \lceil y \rceil \to_\beta^* \lceil x + y \rceil \qquad mult\ \lceil x \rceil\ \lceil y \rceil \to_\beta^* \lceil x \cdot y \rceil \qquad pow\ \lceil x \rceil\ \lceil y \rceil \to_\beta^* \lceil x^y \rceil$$

Church-Numerale

$$\lceil n \rceil := \lambda f\ a. \underbrace{f(f(f(\ldots f}_{n}\ a))) \tag{1}$$

$$zero = \lambda f\ a.a \qquad succ = \lambda n\ f\ a.f\ (n\ f\ a)$$