

Perlen der Informatik 2

3. Übung

1 Beta-Reduktion

Bestimmen Sie jeweils die Normalform der folgenden Terme, falls sie existiert:

1. $(\lambda x. y) z$
2. $(\lambda x y. y x) y z$
3. $(\lambda x. x x) (\lambda x. x x)$
4. $(\lambda x y. y z x) ((\lambda x. x x) (\lambda x. x x)) (\lambda x y. x)$

2 Typen

2.1 Typisierung

Finden Sie den allgemeinsten Typ der folgenden Terme, und geben Sie die Herleitung dazu an:

1. $\lambda x y. y x x$
2. $\lambda x. x (\lambda x y. y)$

Ordnen Sie auch den folgenden Termen ihren allgemeinsten Typ zu (ohne Herleitung), sofern das möglich ist:

3. $\lambda x y z. (x z) y z$
4. $\lambda x y. x y x$
5. $(\lambda x y. (y z) x) ((\lambda x. x x) (\lambda x. x x)) (\lambda x y. x)$

2.2 Preservation

Zeigen Sie:

1. Wenn $\Gamma, x:\sigma \vdash s:\tau$ und $\Gamma \vdash t:\sigma$ dann $\Gamma \vdash s[t/x] : \tau$.
2. Wenn $s \rightarrow_{\beta} t$ und $\Gamma \vdash s : \tau$ dann $\Gamma \vdash t : \tau$.

Gilt auch die Umkehrung: Wenn $s \rightarrow_{\beta} t$ und $\Gamma \vdash t : \tau$, dann $\Gamma \vdash s : \tau$?

3 Listen

3.1 Konstruktion

Um Listen im Lambda-Kalkül darzustellen benötigt man

1. die leere Liste *nil*
2. einen Konstruktor *cons*, der aus einem ersten Element und einer Restliste eine Liste baut
3. eine Funktion *null*, die testet, ob eine Liste leer ist
4. Funktionen *head* und *tail*, die aus einer nichtleeren Liste das erste Element und die Restliste extrahieren.

Überlegen Sie sich zunächst, welche Gesetze die Operationen erfüllen müssen. Definieren Sie dann diese Operationen als λ -Terme im *untypisierten* Lambda-Kalkül. Falls Sie die Programmiersprache *LISP* kennen, ist es hilfreich, sich zu überlegen, wie dort Listen repräsentiert sind.

Zur Erinnerung:

```
true  $\equiv \lambda x y. x$ 
false  $\equiv \lambda x y. y$ 
cond  $\equiv \lambda x. x$ 
pair  $\equiv \lambda x y z. z x y$ 
fst  $\equiv \lambda z. z (\lambda x y. x)$ 
snd  $\equiv \lambda z. z (\lambda x y. y)$ 
```

3.2 Einschränkungen durch Typen

Funktioniert die Konstruktion noch, wenn man im getypten λ -Kalkül arbeitet?

4 Rekursion im (untypisieren) Lambda-Kalkül

Benutzen sie den Y -Kombinator, um eine Funktion zu definieren, die zwei Listen (vgl. vorheriges Blatt) aneinanderhängt.

Zur Erinnerung: $Y \equiv \lambda f. (\lambda x. f (x x)) (\lambda x. f (x x))$

Hinweis 1: Am 23. Mai entfällt die Übung.

Hinweis 2: Ab 30. Mai werden wir den interaktiven Theorembeweiser *Isabelle* in der Übung einsetzen. Falls Sie ein Notebook besitzen, installieren Sie *Isabelle* darauf und bringen Sie es zur Übung mit. Details dazu finden Sie auf der Website der Veranstaltung.