Übungen zur Vorlesung

Funktionale Programmierung

Sommersemester 2025

Übungsblatt Nr. 11

Abgabetermin: -

04.07.2025

Anmerkungen zu diesem Blatt: Neben Haskell kann man das ungetypte λ-Kalkül als eine zweite Programmiersprache für FuPro bezeichnen. Wenn eine Aufgabenstellung also von Ihnen fordert, einen λ-Term anzugeben/zu definieren, dann wird von Ihnen auch genau dies erwartet. Das heißt, dass z.B. Haskell Code, oder λ-Terme die Synonyme für andere Terme benutzen, keine korrekten Lösungen dieser Aufgaben sind. Sollte eine Aufgabe die Verwendung von Synonymen für Terme erlauben, so wird das explizit in der Aufgabenstellung erwähnt.

Aufgabe 1 (Nat & Bool)

Gegeben seien die folgenden Haskell-Datentypen für Wahrheitswerte und natürliche Zahlen.

```haskell
data Bool = True | False

data Nat = Z | S Nat
```

Außerdem seien folgende Church Kodierungen der Konstruktoren von `Bool` und `Nat` im ungetypten λ-Kalkül gegeben.

$$true := \lambda t.\lambda f.\ t$$
$$false := \lambda t.\lambda f.\ f$$

$$zero := \lambda z.\lambda s.\ z$$
$$succ := \lambda n.\lambda z.\lambda s.\ s\ (n\ z\ s)$$

Beachten Sie, dass diese Church Kodierung von `Nat` nicht der Definition von den Folien entspricht! Beide Church Kodierungen sind allerdings isomorph. Um auf diesem Übungsblatt eindeutig in unseren Bezeichnungen zu sein, werden wir für die obige Kodierung nicht von Church Numeralen, sondern von Church kodierten `Nat` sprechen.

1. Geben Sie einen λ-Term *add* im ungetypten λ-Kalkül an, der zwei Church kodierte `Nat` addiert.
2. Geben Sie einen λ-Term *even* im ungetypten λ-Kalkül an, der prüft ob eine Church kodierte `Nat` gerade ist. Appliziert auf eine Church kodierte `Nat` soll *even* also zu dem entsprechenden Church kodierten `Bool` reduzieren.

Aufgabe 2 (Binärzahlen)
Gegeben sei der folgende Haskell-Datentyp für Binärzahlen.

```
data Bin = LSB | Zero Bin | One Bin
```

Der Konstruktor `LSB` dient als „Markierung" des niedrigsten Stellenwerts.
So entspricht z.B.
`LSB` der Zahl 0,
`Zero LSB` der Zahl 0,
`One LSB` der Zahl 1,
`One (Zero LSB)` der Zahl 2,
`One (Zero (Zero LSB))` der Zahl 4 und so weiter.

1. Geben Sie eine Church Kodierung für den Haskell-Datentyp `Bin` im ungetypten λ-Kalkül an, indem Sie λ-Terme *lsb*, *zero* und *one* angeben, die den Church Kodierungen der drei Konstruktoren `LSB`, `Zero` und `One` entsprechen.

2. Geben Sie einen λ-Term *shift* im ungetypten λ-Kalkül an, der eine Church kodierte Binärzahl verdoppelt. Appliziert auf eine Church kodierte Binärzahl soll *shift* also zu einer Church kodierten Binärzahl reduzieren, die an der Stelle mit dem niedrigsten Stellenwert ein 0-Bit hinzugefügt hat.

Aufgabe 3 (Binärbäume)

Auf Übungsblatt 5 hatten Sie den folgenden Haskell-Datentypen für Binärbäume kennengelernt:

```
data BinBaum a = BinLeer | BinKnoten a (BinBaum a) (BinBaum a)
```

1. Church kodieren Sie die Haskell-Datentypen im ungetypten λ-Kalkül, indem Sie λ-Terme für die Konstruktoren des Datentypen angeben.

2. Geben Sie einen Beispiel λ-Term für einen Church kodierten Binärbaum über Church kodierten `Nat` an, der mindestens Höhe 2 hat. (Anm.: BinLeer hat Höhe 0.)

3. Geben Sie einen λ-Term für die Haskell-Funktion knotenSumme von Blatt 5 an, die einen Church kodierten Binärbaum über Church kodierten `Nat` auf die Summe seiner Knoten abbildet. Sie dürfen zur Definition das Synonym *add* für den in Aufgabe 1 definierten λ-Term für die Addition von zwei Church kodierten `Nat` benutzen.

4. β-Reduzieren Sie die Applikation von knotenSumme auf ihren Beispielterm solange, bis die Normalform erreicht ist. Entspricht das Ergebnis dem erwarteten Church kodierten `Nat`?