

Übungen zu „Übersetzerbau“ Blatt 7

Aufgabe 29. Zeige das folgende Gesetz („banana split“):

$$([h]) \Delta ([k]) = ([h \cdot F \text{ outl} \Delta k \cdot F \text{ outr}])$$

Welche bekannte Optimierung formalisiert das Gesetz? Spezialisiere das Gesetz für natürliche Zahlen: **data** *Nat* = *Zero* | *Succ Nat*.

Aufgabe 30. Definiere einen imperativen Übersetzer für erweiterte arithmetische Ausdrücke (siehe Aufgabe 21). Verwende als Zielsprache Stackmaschinen-Code:

```
data Code = Pop
           | Push Int
           | Swap
           | App Op
           | Copy Int
           | Code :^: Code
```

Die Instruktion *Pop* entfernt ein Element vom Stack; *Copy n* kopiert das Element an Position *n* (von oben gezählt) nach oben.

Welche Übersetzungsaktionen werden benötigt? Welche Informationen müssen im Zustand verwaltet werden?

Aufgabe 31.

1. Erweitere Ausdrücke um Boolesche und Vergleichsoperationen. Erweitere den Übersetzer aus Aufgabe 30 entsprechend.
2. Füge weiterhin eine einfache Typprüfung hinzu. Wir unterscheiden zwei Typen: *Bool* und *Int*. Die Vergleichsoperationen haben den Typ $Int \times Int \rightarrow Bool$, die Booleschen Operationen $Bool \times Bool \rightarrow Bool$.
3. Wie muss die Typprüfung geändert werden, wenn wir die Vergleichsoperatoren überladen, so dass sie den Typ $\alpha \times \alpha \rightarrow Bool$ für $\alpha \in \{Bool, Int\}$ besitzen?

Aufgabe 32. Die folgenden Baumsprachen repräsentieren Ableitungsbäume der LL(1)-Grammatik für arithmetische Ausdrücke (zu jeder Produktion korrespondiert genau ein Knotentyp).

```
data Expr = Start Term Expr'
data Expr' = End | Add Term Expr'
data Term = Start Factor Term'
data Term' = End | Mul Factor Term'
data Factor = Paren Expr | Num Int
```

Definiere eine Attributkopplung, die einen Ableitungsbaum in den korrespondierenden arithmetischen Ausdruck übersetzt.