

Übungen zur Vorlesung

„Grundlagen der Programm- und Systementwicklung“

Aufgabe 1 Fehleralgebren

Betrachten Sie folgende algebraische Spezifikation für die Rechenstruktur natürlicher Zahlen:

```

SPEC NAT =
{
  based_on BOOL
  sort Nat
  0
  succ, pred
  .+, *.
  iszero
  Nat generated_by 0, succ
}

```

|                 |         |           |             |           |           |
|-----------------|---------|-----------|-------------|-----------|-----------|
| iszero(0)       | = true  | 0+y       | = y         | 0*y       | = 0       |
| iszero(succ(x)) | = false | succ(x)+y | = succ(x+y) | succ(x)*y | = y+x*y   |
| pred(succ(x))   | = x     | x+y       | = y+x       | x*y       | = y*x     |
|                 |         | (x+y)+z   | = x+(y+z)   | (x*y)*z   | = x*(y*z) |

- (a) Handelt es sich dabei um eine partielle oder um eine totale Algebra?
- (b) Erweitern Sie diese Spezifikation durch Funktionsanreicherung um Operationen für die Division und die Moduloberechnung. Diskutieren Sie dabei Ansätze zur Behebung eventueller Fehler. Geben Sie eine entsprechende Fehleralgebra an.
- (c) Welche Techniken stehen in realen Programmiersprachen zur Fehlerbehandlung zur Verfügung?

Aufgabe 2 Fehlermodellierung

Das Ziel dieser Aufgabe ist die Modellierung eines Geldwechselautomaten. Dazu muß der eingegebene Geldbetrag als Summe von Münzen vorgegebenen Werts ausgedrückt werden. Wir nehmen an, dass zu jedem Münzwert unendlich viele Münzen zur Verfügung stehen. Angenommen, der Automat enthalte Münzen der Werte 10, 5 und 2, dann ist bei Eingabe des Betrags 42 die Sequenz <10, 10, 10, 5, 5, 2> eine mögliche Ausgabe. Mit den hier vorgegebenen Münzwerten lässt sich beispielsweise der Betrag 3 nicht wechseln.

- (a) Diskutieren Sie mögliche Fehlersituationen für den Wechselautomaten.
- (b) Geben Sie für die relevanten Datenmengen algebraische Spezifikationen an.
- (c) Formulieren Sie eine Operation **change**, die den Wechselvorgang modelliert. Wie behandeln Sie die erkannten Fehlersituationen?

Aufgabe 3 Funktionale Programmierung

In der Vorlesung haben Sie Funktionsdefinition, Funktionsanwendung, bedingte Auswahl, und Rekursion als die wesentlichen Sprachelemente der Funktionalen Programmierung kennengelernt.

- (a) Welche Sprachmittel, die Ihnen aus anderen Programmierstilen bekannt sind, fehlen in der funktionalen Programmierung, bzw. besitzen dort untergeordnete Bedeutung?
- (b) Schreiben Sie ein funktionales Programm zur Reversion von Sequenzen.
- (c) Schreiben Sie ein funktionales Programm **swap** zur Spiegelung knotenmarkierter Bäume. Beweisen Sie dann, dass für einen solchen Baum **t** gilt:

$$\text{swap}(\text{swap}(t)) = t$$

Aufgabe 4 Currying

Funktionen höherer Stufe haben Parameter bzw. Rückgabewerte, die selbst wieder Funktionen sind. Besondere höhere Funktionen sind durch folgende Vereinbarung gegeben (sei  $i \in \mathbb{N}, 1 \leq i \leq n$ ):

$$\text{curry}_i : (\alpha_1 \times \dots \times \alpha_n \rightarrow \alpha_{n+1}) \rightarrow (\alpha_1 \times \dots \times \alpha_i \rightarrow (\alpha_{i+1} \times \dots \times \alpha_n \rightarrow \alpha_{n+1}))$$

Durch Anwendung der entsprechenden *curry*-Funktion („currying“) lassen sich zu einer vorgegebenen Funktion  $f : \alpha_1 \times \dots \times \alpha_n \rightarrow \alpha_{n+1}$  neue Funktionen erzeugen, die jeweils einer Teilauswertung der ursprünglichen Funktion  $f$  entsprechen. Statt *curry*<sub>i</sub> schreiben wir vereinfachend auch nur *curry*.

Als Beispiel betrachten wir die Funktionen *map*, *s*, + und \*, definiert durch

```

map = (f : α → β, y : List α) :
  if (y = <>) then <> else if y = (x ◦ xs) then f(x) ◦ (map(f, xs)) fi fi

s = (x : Nat) Nat :
  succ(x)
+ = (x : Nat, y : Nat) :
  if (x = 0) then y else if x = s(z) then s(z + y) fi fi
* = (x : Nat, y : Nat) :
  if (x = 0) then 0 else if x = s(z) then y + (z * y) fi fi

```

(a) Berechnen Sie

$$\text{map}(s, < 0, s(0), s(s(0)) >)$$

und

$$\text{map}((x : \text{Nat}) \text{Nat} : x * x, < 0, s(0), s(s(0)) >)$$

- (b) Welchen Typ haben jeweils die Funktionen  $\text{map}$ ,  $s$ ,  $+$  und  $*$ ?  
(c) Welchen Typ haben jeweils die Funktionen  $\text{curry}(\text{map})$ ,  $\text{curry}(s)$ ,  $\text{curry}(+)$  und  $\text{curry}(*)$ ?  
(d) Berechnen Sie  $\text{curry}(+, 0)$ ,  $\text{curry}(+, 1)$ ,  $\text{curry}(*, 0)$ ,  $\text{curry}(*, 1)$ ,  $\text{curry}(*, 2)$