Musterlösung IV. Übungsblatt “Informatik I” WS 95/96

Christian Schulte

Aufgabe IV.1 (Paarfunktion)

1. Wir müssen uns davon überzeugen, daß paar-1 und paar-2 wohldefiniert sind. Da die Funktion paar injektiv ist, gilt:

   $$i \neq i' \text{ oder } j \neq j' \qquad \Rightarrow \qquad \text{paar}(i,j) \neq \text{paar}(i',j').$$

   Das heißt, zu einem k kann es höchstens ein Paar (i,j) geben, so daß paar$(i,j) = k$ ist. Also sind die beiden Funktionen wohldefiniert.

2. Diese Funktion ist injektiv. Dies liegt an der Eindeutigkeit der Primfaktorzerlegung von natürlichen Zahlen.

3. $$\text{paar-1}(k) := \begin{cases} 1 + \text{paar-1}(k/2) & \text{falls } \exists i,j \in \mathbb{N} : k = 2^{i+1}3^j \\ 0 & \text{falls } k = 1 \\ \texttt{\#f} & \text{sonst} \end{cases}$$

   Die Funktion paar-2 läßt sich analog definieren.

4. ```
   (define (paar i j)
      (* (exp 2 i) (exp 3 j)))
   ```

5. Wir verwenden eine iterative Version, die in einem zusätzlichen Argument den bisher berechneten Wert bekommt. Stellt sich heraus, daß die eingegebene Zahl nicht von der Form 2^i3^j ist, so kann auch direkt der Wert `#f` zurückgeliefert werden. Eine einfache rekursive Version müßte hier jeweils den Rückgabewert des rekursiven Aufrufes darauf überprüfen, ob er gleich `#f` ist.

   ```
   (define (paar-iter-1 k i)
      (cond ((= k 1)
              i)
             ((= (remainder k 2) 0)
              (paar-iter-1 (/ k 2) (+ i 1)))
             ((= (remainder k 3) 0)
              (paar-iter-1 (/ k 3) i))
    (else
              #f)))

   (define (paar-1 k) (paar-iter-1 k 0))
   ```

   Die Prozedur `paar--2` geht analog dazu.

Aufgabe IV.2 (Zahlentripel und Zahlenquadrupel) Die einfachste Lösung ist, die Funktionen aus der vorherigen Aufgabe zu verwenden.
Definiert man die Funktion triple wie folgt, so ist der Rest klar:

$$\text{tripel}(i,j,k) := \text{paar}(i, \text{paar}(j,k)).$$
   ```

Das gleiche kann man auch für die Funktion quadrupel tun:

$$\text{quadrupel}(i, j, k, l) := \text{paar}(\text{paar}(i, j), \text{paar}(k, l)).$$

Aufgabe IV.3 (Semantikbäume) Vielen Dank an Ulrich Callmeier (`uc@brian.lunetix.de`), dessen tolles Tool zum automatischen Erzeugen der Semantikbäume hier Anwendung findet. Die Sematikbäume finden Sie in den beiden Extradateien.

Aufgabe IV.4 (Münzwechseln)

```
(define (div d1 d2)
  (/ (- d1 (remainder d1 d2)) d2))

(define (wert m)
  (cond ((= m 1) 1)
((= m 2) 2)
((= m 3) 5)))

(define (wert-neu a betrag anzahl)
  (cond ((= a -1)
 anzahl)
(else
 (wert-neu (- a 1)
   betrag
   (+ anzahl (div (- betrag (* a (wert 3))) (wert 2)) 1)))))

(define (wechsle betrag)
  (wert-neu (div betrag (wert 3)) betrag 0))
```