Technische Universität München
Fakultät für Informatik
Prof. Dr. M. Broy

WS 2001
Übungsblatt 8
04. Dezember 2001

Übungen zu Einführung in die Informatik I

Aufgabe 31 Das Prinzip der Induktion

In dieser Aufgabe lernen Sie die Induktion kennen, ein mathematisches Hilfsmittel, das im weiteren Verlauf der Übungen noch häufig verwendet werden wird.

Es sei für $x \in \mathbb{N}$ die boolesche Funktion `gerade` (x), die ermittelt, ob x durch 2 teilbar ist oder nicht, wie folgt spezifiziert:

$$\texttt{gerade}\ (x) = (\mathrm{mod}(x,2) \stackrel{?}{=} 0)$$

(Hinweis: Die Funktion $\mathrm{mod}(x,2)$ berechnet den Rest bei ganzzahliger Division von x durch 2.) Darüber hinaus sei die Funktion `gerade`$'$ (x) durch

$$\texttt{gerade}'\ (x) = \begin{cases} \texttt{True}, & \text{falls } x = 0 \\ \neg\ (\texttt{gerade}'\ (\texttt{x} - 1)), & \text{falls } x \in \mathbb{N} \setminus \{0\} \end{cases}$$

gegeben.

Aufgabenstellung: Zeigen Sie mit Hilfe der Induktion, dass die beiden Funktionen `gerade` (x) und `gerade`$'$ (x) äquivalent sind.

Aufgabe 32 Induktion über den Termaufbau: Negationsnormalform

Ein Boolescher Term ist in Negationsnormalform, wenn in ihm Negationszeichen nur unmittelbar vor Atomen oder vor Identifikatoren vorkommen. Zeigen Sie mit Induktion über den Termaufbau, daß sich jeder Boolesche Term auf einen semantisch äquivalenten Term in Negationsnormalform reduzieren läßt. Es genügt beim Beweis, sich auf Terme zu beschränken, in denen als Operatoren nur $\neg$, $\vee$ und $\wedge$ vorkommen. Die anderen faßt man als Schreibabkürzungen auf, ebenso die Konstanten `true` und `false`. Die Normalform ist natürlich nicht eindeutig, z.B. sind die folgenden semantisch äquivalenten Terme in Negationsnormalform.

$$(a \vee b) \Rightarrow c,\ (\neg a \wedge \neg b) \vee c,\ (\neg a \vee c) \wedge (\neg b \vee c)$$

Aufgabe 33 Funktionsabstraktion, call-by-value, call-by-name

Gegeben sei folgende Funktionsabstraktion:

(**nat** t, **nat** x, **nat** y, **nat** z) **nat**: E

mit E = **if** t $\stackrel{?}{=}$ 0 **then** x*x*x
else if t $\stackrel{?}{=}$ 1 **then** x*x*y
else x*y*z
fi
fi

Werten Sie die Funktionsapplikation

$$((\textbf{nat}\ t, \textbf{nat}\ x, \textbf{nat}\ y, \textbf{nat}\ z)\ \textbf{nat}\text{: E})(1*1, 12-8+100, 12-8, 12-8+100)$$

gemäß der Strategie call-by-value und gemäß der Strategie call-by-name aus und geben Sie jeweils die Anzahl der Auswertungen für t, x, y und z an.

Aufgabe 34 (P) Guarded Equations und bedingte Ausdrücke

a) Gegeben sei die Funktionsdefinition der Funktion ggT in Vorlesungsnotation:

```
fct ggT = (nat a, nat b) nat:
    if a=?b then a
    elif a > b then ggT(a-b, b)
               else ggT (a, b-a)
    fi
```

Übertragen Sie diese Funktionsdefinition in Gofernotation:

(i) mit Hilfe von Guarded Equations

(ii) mit Hilfe des bedingten Ausdrucks `if-then-else`.

Hinweis: Verwenden sie in Gofer statt der Sorte **nat** den Datentyp `Int`.

b) In der Zentralübung wurde der Datentyp `data Rat = Quot (Int, Int)` der rationalen Zahlen und die Funktionen `zaehler`, `nenner` und `kuerze` eingeführt. Implementieren Sie unter Verwendung von Pattern matching, bedingter Ausdrücke und/oder Garded Equations folgende Funktionen:

(i) `isdef :: Rat -> Bool` prüft, ob die rationale Zahl definiert ist, d.h. ob der Nenner ungleich null ist.

(ii) `isInteger :: Rat -> Bool` prüft, ob die rationale Zahl eine ganze Zahl ist.

(iii) `hauptnenner :: Rat -> Rat -> Int` ermittelt den Hauptnenner zweier rationaler Zahlen. (Hinweis: Sei können zur Implementierung der Funktion `hauptnenner` die Funktion `lcm` benutzen.)

(iv) `addiere :: Rat -> Rat -> Rat` addiert zwei rationale Zahlen

(v) Testen Sie ihre Implementierung mit:

```
r = Quot(3,6)
s = Quot(1,2)
t = Quot(2,0)

test1 = isdef t
test2 = isInteger r
test3 = hauptnenner r s
test4 = kuerze (addiere r s)
test5 = addiere r t
```

Hinweis: Prüfen Sie bei der Implementierung der Funktionen `hauptnenner` und `addiere` mit der Funktion `isdef`, ob deren Argumente definiert sind.

Aufgabe 35 (H) BNF und induktive Charakterisierung formaler Sprachen

Über dem Zeichenvorrat C = {a, b, c} sei die formale Sprache

$$L = \{x \in C^* \,|\, x = a^n b^m c^k, \text{ mit } n, m, k \in \mathbb{N} \text{ und } n = m + k\}$$

definiert.

a) Geben Sie BNF-Regeln an, die die formale Sprache L beschreiben.

b) Die in Teilaufgabe a) anzugebenden BNF-Regeln besitzen die Struktur $<e_j>$ ::= R_j, wobei der Index j zur Numerierung der von Ihnen eingeführten Regeln dient.

Die Nonterminale $<e_j>$ repräsentieren formale Sprachen $X_j \in C^*$. Charakterisieren Sie diese Sprachen X_j induktiv durch Angabe der formalen Sprachen X_j^i mit $i \in \{0, 1, 2, 3\}$, wie dies auch schon in Teilaufgabe 28 b) durchgeführt wurde.