Prof. Martin Hofmann
Dr. Ulrich Schöpp

Ludwig-Maximilians-Universität München
Institut für Informatik
8. Juli 2011

Formale Sprachen und Komplexität

Blatt 10

Aufgabe 10-1. (3 Punkte) Schreiben Sie ein LOOP-Programm, welches die Subtraktionsfunktion $s\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ mit

$$s(x,y) = \begin{cases} x - y & \text{falls } x > y \\ 0 & \text{sonst} \end{cases}$$

berechnet. Benutzen sie keine abkürzenden Notationen.

Aufgabe 10-2. Sei c eine beliebige natürliche Zahl. Schreiben Sie ein LOOP-Programm, welches die Funktion $f\colon \mathbb{N} \to \mathbb{N}$ mit $f(x) = x \mod c$ berechnet. Versuchen Sie nur eine einzige `loop`-Schleife zu verwenden.

Aufgabe 10-3. Schreiben Sie ein LOOP-Programm für die Fibonacci-Zahlen. Das heißt, Ihr Programm soll die Funktion $f\colon \mathbb{N} \to \mathbb{N}$ berechnen, die durch $f(0) = 1$, $f(1) = 1$ und $f(x+2) = f(x+1) + f(x)$ für $x \geq 0$ definiert ist.

Aufgabe 10-4. (3 Punkte) In LOOP-Programmen gibt es keine `if`-Tests, man kann diese jedoch definieren. Schreibe `sign` x_j als Abkürzung für das Programm `loop` x_j `do` $x_j := 1$ `end`. Hat x_j den Wert 0, so hat x_j auch nach Ausführung von `sign` x_j diesen Wert. Hat x_j einen Wert größer als 0, so hat x_j nach Ausführung von `sign` x_j den Wert 1. Damit kann man

$$\texttt{if } x_i > 0 \texttt{ then } P \texttt{ end}$$

als Abkürzung für das Programm

$$x_j := x_i;\ \texttt{sign } x_j;\ \texttt{loop } x_j \texttt{ do } P \texttt{ end}$$

definieren. Dabei wird j größer als i und größer als jede in P vorkommende Variablennummer gewählt.

Definieren Sie analog eine Abkürzung `if` $x_i = k$ `then` P_1 `else` P_2 `end` für eine beliebige gegebene natürliche Zahl k sowie beliebige LOOP-Programme P_1 und P_2. Wenn der Wert der Variable x_i gleich k ist, so soll P_1 ausgeführt werden, sonst P_2.

Aufgabe 10-5. In dieser Aufgabe sollen einige Funktionen zur Manipulation von Listen von Nullen und Einsen (Beispiel: $[0,1,0,1,1,0]$) geschrieben werden. Solche Listen sollen folgendermaßen in natürliche Zahlen kodiert sein: Eine Null wird durch die Bitfolge 10 kodiert und eine Eins durch die Bitfolge 11. Die Codes der einzelnen Elemente der Liste werden in *umgekehrter Reihenfolge* konkateniert und als Binärdarstellung einer Zahl aufgefasst. Diese Zahl ist dann der Code der Liste. Zum Beispiel wird die Liste $[0,1,0,1,1,0]$ kodiert durch die Zahl mit Binärdarstellung 101111101110, also 3054. Die leere Liste $[\,]$ wird so durch die Zahl 0 kodiert. (Die Listen werden nicht direkt als Bitfolgen aufgefasst, da z.B. die Listen $[1]$, $[0,1]$ und $[1,0]$ verschiedene Codes haben müssen.)

Schreiben Sie WHILE-Programme für die Listenfunktionen `cons` (Anhängen eines Zeichens an den Anfang der Liste), `head` (Kopf der Liste), `tail` (Rest der Liste) sowie `len` (Länge der Liste). Beispiele:

$$\texttt{cons}(0,[0,1,1]) = [0,0,1,1] \quad \texttt{head}([0,1,1]) = 0 \quad \texttt{tail}([0,1,1]) = [1,1] \quad \texttt{len}([0,1,1]) = 3$$
$$\texttt{cons}(1,[0,1,1]) = [1,0,1,1] \qquad \texttt{head}([\,]) = 0 \qquad \texttt{tail}([\,]) = [\,] \qquad \texttt{len}([\,]) = 0$$

In diesen Beispielen werden der besseren Lesbarkeit wegen Listen anstelle ihrer Codes benutzt. Die beiden Gleichung für `cons` stehen eigentlich für $\texttt{cons}(0,62) = 250$ und $\texttt{cons}(1,62) = 251$.

Abgabe: Sie können ihre Lösungen bis Freitag, den 15.07., um 18:00 Uhr im Abgabekasten in der Theresienstraße oder über UniWorX abgeben. In UniWorX werden Dateien im `txt`-Format (reiner Text) oder im `pdf`-Format akzeptiert.