Johann Wolfgang Goethe-Universität Frankfurt am Main
Institut für Informatik
Theorie komplexer Systeme
Prof. Dr. Nicole Schweikardt 5. November 2008

Diskrete Modellierung

Wintersemester 2008/2009

Übungsblatt 3

Abgabe: *bis 12. November 2008, 8.^{15} Uhr (vor der Vorlesung oder in Raum RM 11-15/113)*

Aufgabe 1: **($10 + 10 = 20$ Punkte)**

Beweisen Sie folgendes durch vollständige Induktion nach n.

(a) Für alle $n \in \mathbb{N}$ mit $n \geq 1$ gilt: $\sum_{i=1}^{n}(4i-1) \;=\; 2n^2 + n$.

(b) Für alle $x \in \mathbb{R}$ mit $x \geq -1$ und alle $n \in \mathbb{N}$ mit $n \geq 1$ gilt: $1 + n \cdot x \;\leq\; (1+x)^n$.

Aufgabe 2: **($10 + 15 = 25$ Punkte)**

Ein möglicher Algorithmus, um für eine Zahl $n \in \mathbb{N}_{>0}$ den Wert $\mathrm{fib}(n)$ der Fibonacci-Folge zu berechnen, ist:

Algo 1 (bei Eingabe einer Zahl $n \in \mathbb{N}_{>0}$):

(1) Falls $n = 1$ oder $n = 2$, dann gib 1 als Ergebnis zurück.
(2) Falls $n \geq 3$, dann:
(3) Sei x_1 die Ausgabe von *Algo 1* bei Eingabe der Zahl $n-1$.
(4) Sei x_2 die Ausgabe von *Algo 1* bei Eingabe der Zahl $n-2$.
(5) Gib den Wert $(x_1 + x_2)$ als Ergebnis zurück.

Der Algorithmus benötigt bei Eingabe einer Zahl n höchstens $g_1(n)$ Schritte, wobei

$$
\begin{aligned}
g_1(1) &= 1 \quad \text{und} \quad g_1(2) \;=\; 1 \quad \text{und} \\
g_1(n) &= 3 + g_1(n-1) + g_1(n-2) \quad \text{für alle } n \in \mathbb{N} \text{ mit } n \geq 3
\end{aligned}
$$

(der Einfachheit halber zählen die Zeilen 1, 2 und 5 hier jeweils nur als ein Schritt).
Ein anderer Algorithmus, um für eine Zahl $n \in \mathbb{N}_{>0}$ den Wert $\mathrm{fib}(n)$ zu berechnen, ist:

Algo 2 (bei Eingabe einer Zahl $n \in \mathbb{N}_{>0}$):

(1) Falls $n = 1$ oder $n = 2$, dann gib 1 als Ergebnis zurück.
(2) Seien $a_0 := 0$, $a_1 := 1$ und $a_2 := 1$.
(3) Wiederhole für alle i von 3 bis n:
(4) Ersetze a_0 durch a_1 und a_1 durch a_2.
(5) Ersetze a_2 durch $a_0 + a_1$.
(6) Gib den Wert a_2 als Ergebnis zurück.

Dieser Algorithmus benötigt bei Eingabe $n \in \mathbb{N}_{>0}$ höchstens $g_2(n) \;:=\; 2 \cdot (n-2) + 3$ Schritte (wie oben zählen wir die Zeilen 1, 2, 4, 5 und 6 jeweils als nur einen Schritt).

(a) Welcher der beiden Algorithmen läuft im Allgemeinen schneller? D.h. welche der beiden Funktionen g_1 und g_2 liefert kleinere Funktionswerte?

(b) Beweisen Sie, dass Ihre Antwort aus (a) korrekt ist. D.h. falls Sie in (a) geantwortet haben, dass *Algo* i im Allgemeinen schneller als *Algo* j ist, dann finden Sie eine Zahl $n_0 \in \mathbb{N}_{>0}$ und beweisen Sie per Induktion nach n, dass für alle $n \in \mathbb{N}$ mit $n \geq n_0$ gilt: $g_i(n) < g_j(n)$.

Aufgabe 3: (16 + 14 = 30 Punkte)

Im Folgenden wird die Syntax einer sehr einfachen Programmiersprache definiert, der so genannten WHILE-Programme. Die Menge L, die hier definiert wird, ist die Menge aller Zeichenketten über dem Alphabet A, die syntaktisch korrekte WHILE-Programme sind. Hierbei ist $A := \{\mathbf{x}, \mathbf{:=}, \mathbf{+}, \mathbf{-}, \mathbf{\neq}, \mathbf{;}, \mathbf{while}, \mathbf{do}, \mathbf{end}\} \cup \mathbb{N}$, und L ist die folgendermaßen rekursiv definierte Menge:

Basisregeln:	(B1)	Für Zahlen $i, j, c \in \mathbb{N}$ gilt: $\mathbf{x}i \mathbf{:=} \mathbf{x}j \mathbf{+} c \in L$.
	(B2)	Für Zahlen $i, j, c \in \mathbb{N}$ gilt: $\mathbf{x}i \mathbf{:=} \mathbf{x}j \mathbf{-} c \in L$.
Rekursive Regeln:	(R1)	Sind $w_1 \in L$ und $w_2 \in L$, so ist auch $w_1\mathbf{;}\ w_2 \in L$.
	(R2)	Ist $w \in L$ und $i \in \mathbb{N}$, so ist $\mathbf{while}\ \mathbf{x}i \neq 0\ \mathbf{do}\ w\ \mathbf{end} \in L$.

(a) Welche der folgenden Wörter aus A^* gehören zu L und welche nicht? Begründen Sie jeweils Ihre Antwort.

(i) $\mathbf{x}3 \mathbf{:=} \mathbf{x}7 \mathbf{-} 2$

(ii) $\mathbf{x}3 \mathbf{:=} 1\mathbf{;}\ \mathbf{x}2 \mathbf{:=} \mathbf{x}3 \mathbf{+} 5$

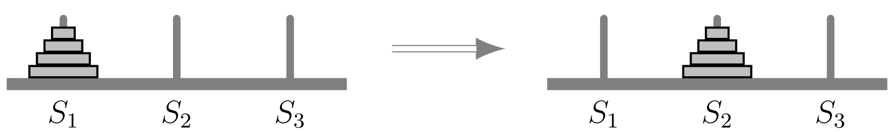
(iii) $\mathbf{while}\ \mathbf{x}1 \neq 0\ \mathbf{do}\ \mathbf{x}0 \mathbf{:=} \mathbf{x}0 \mathbf{+} 1\mathbf{;}\ \mathbf{x}1 \mathbf{:=} \mathbf{x}1 \mathbf{-} 1\ \mathbf{end}$

(iv) $\mathbf{x}1 \mathbf{:=} \mathbf{x}1 \mathbf{+} 42\mathbf{;}\ \mathbf{while}\ \mathbf{x}1 \neq 0\ \mathbf{do}\ \mathbf{x}1 \mathbf{:=} \mathbf{x}1 \mathbf{-} 1$

(b) Für jedes Wort $w \in A^*$ und jedes Symbol $s \in A$ bezeichne $|w|_s$ die Anzahl der Vorkommen des Symbols s in w. Beweisen Sie durch Induktion, dass für alle Wörter $w \in L$ gilt: $|w|_{\mathbf{do}} = |w|_{\mathbf{end}}$.

Aufgabe 4: Türme von Hanoi (10 + 15 = 25 Punkte)

Ein Turm aus $n \in \mathbb{N}_{>0}$ unterschiedlich großen gelochten Scheiben soll von einem Stab (S_1) auf einen zweiten Stab (S_2) unter Zuhilfenahme eines Hilfsstabes (S_3) verschoben werden (das folgende Bild zeigt die Situation für den Fall $n = 4$).



Dabei müssen die folgenden Regeln beachtet werden:

- Pro Zug darf nur eine Scheibe bewegt werden. Es kann also immer nur die oberste Scheibe eines Turmes bewegt werden.
- Es darf nie eine größere Scheibe auf einer kleineren Scheibe liegen.

(a) Beschreiben Sie, wie der Turm im Fall $n = 4$ von S_1 nach S_2 verschoben werden kann.

(b) Beweisen Sie, dass es für alle $n \in \mathbb{N}_{>0}$ möglich ist, die n Scheiben von S_1 nach S_2 zu verschieben.

Hinweis: Beweisen Sie zuerst durch vollständige Induktion nach n, dass die folgende Aussage für alle $n \in \mathbb{N}$ mit $n \geq 1$ gilt:

A(n): Seien $i, j \in \{1, 2, 3\}$ mit $i \neq j$, sei $m \in \mathbb{N}$ mit $m \geq n$, und seien m Scheiben so auf die drei Stäbe verteilt, dass gilt: (1) auf S_i liegen mindestens n Scheiben, und (2) die Scheiben auf den beiden anderen Stäben sind größer als die obersten n Scheiben auf S_i. Dann lassen sich die obersten n Scheiben von S_i so nach S_j verschieben, dass keine der anderen Scheiben bewegt wird.