Kurs 1655 – Grundlagen der Theoretischen Informatik

Lösungshinweise zur Klausur vom 1. März 2003

Aufgabe 1

(a) Sei $n \in \mathbb{N}$. Wir beweisen die Behauptung durch vollständige Induktion über $i \in \mathbb{N}$.

<u>Induktionsanfang $i = 0$:</u> Offenbar gilt

$$\begin{aligned} ES^{3\cdot 0}(2,(1,n,0,0,\dots)) &= (2,(1,n,0,0,\dots)) \\ &= \left(2,\left(\frac{(2\cdot 0+1)!}{0!\cdot 2^0},n,0,0,\dots\right)\right). \end{aligned}$$

<u>Induktionsschritt $i \to i+1$:</u> Sei $i+1 \le n$. Dann gilt auch $i \le n$, und damit folgt nach Induktionsannahme

$$ES^{3i}(2,(1,n,0,0,\dots)) = \left(2,\left(\frac{(2i+1)!}{i!\cdot 2^i},n,i,0,\dots\right)\right).$$

Daraus erhalten wir

$$\begin{aligned} &ES^{3(i+1)}(2,(1,n,0,0,\dots)) \\ &= ES^3\left(2,\left(\frac{(2i+1)!}{i!\cdot 2^i},n,i,0,\dots\right)\right) \\ &= ES^2\left(3,\left(\frac{(2i+1)!}{i!\cdot 2^i},n,i,0,\dots\right)\right) \quad (\text{da } i+1 \le n, \text{ also } i \ne n) \\ &= ES\left(4,\left(\frac{(2i+1)!}{i!\cdot 2^i},n,i+1,0,\dots\right)\right) \\ &= \left(2,\left(\frac{(2i+1)!}{i!\cdot 2^i}\cdot(2(i+1)+1),n,i+1,0,\dots\right)\right) \\ &= \left(2,\left(\frac{(2i+1)!}{i!\cdot 2^i}\cdot\frac{2(i+1)+1)\cdot(2i+2)}{(i+1)\cdot 2},n,i+1,0,\dots\right)\right) \\ &= \left(2,\left(\frac{(2(i+1)+1)!}{(i+1)!\cdot 2^{i+1}},n,i+1,0,\dots\right)\right), \end{aligned}$$

also die Behauptung für $i+1$.

(b) Sei $n \in \mathbb{N}$. Nach (a) erhalten wir für $i = n$

$$\begin{aligned} &ES^{3n+2}(1,(0,n,0,0,\dots)) \\ &= ES^{3n+1}(2,(1,n,0,\dots)) \\ &= ES\left(2,\left(\frac{(2n+1)!}{n!\cdot 2^n},n,n,0,\dots\right)\right) \quad (\text{nach (a)}) \\ &= \left(5,\left(\frac{(2n+1)!}{n!\cdot 2^n},n,n,0,\dots\right)\right). \end{aligned}$$

Da die letztgenannte Konfiguration eine Endkonfiguration ist, ergibt sich nach Definition der Eingabecodierung und der Ausgabecodierung einer einstelligen Registermaschine $f_M(n) = \frac{(2n+1)!}{n! \cdot 2^n}$.

Aufgabe 2

Die Funktion $g : \mathbf{N}^2 \to \mathbf{N}$, die definiert ist durch

$$g(\langle i,j\rangle, \langle k,l\rangle) := \langle i^2 + j^2 + k, 2ij + l\rangle$$

für alle $i, j, k, l \in \mathbf{N}$, ist berechenbar. Da die Cantorsche Tupelfunktion und ihre Umkehrungen berechenbar sind, zeigt man dies leicht z.B. mit Hilfe einer verallgemeinerten Registermaschine. Nun gilt

$$\begin{aligned}
& f(\nu_{\mathbf{Z}}\langle i,j\rangle, \nu_{\mathbf{Z}}\langle k,l\rangle) \\
&= f(i-j, k-l) \\
&= (i-j)^2 + k - l \quad \checkmark \\
&= i^2 + j^2 + k - 2ij - l \quad \checkmark \\
&= \nu_{\mathbf{Z}}\langle i^2 + j^2 + k, 2ij + l\rangle \\
&= \nu_{\mathbf{Z}} g(\langle i,j\rangle, \langle k,l\rangle)
\end{aligned}$$

für alle $i, j, k, l \in \mathbf{N}$. Also ist f eine $(\nu_{\mathbf{Z}}, \nu_{\mathbf{Z}}, \nu_{\mathbf{Z}})$–berechenbare Funktion.

Aufgabe 3

Wir definieren eine Funktion $h :\subseteq \mathbf{N}^2 \to \mathbf{N}$ durch

$$h(i,n) := 0 \cdot u_\varphi(i,n).$$

Da die zweistellige konstante Nullfunktion berechenbar ist, da u_φ nach dem utm-Theorem berechenbar ist und da die Multiplikation berechenbar ist, ist auch h eine berechenbare Funktion. Nach dem smn-Theorem existiert also eine berechenbare Funktion $r : \mathbf{N} \to \mathbf{N}$, so dass

$$\varphi_{r(i)}(n) = h(i,n)$$

für alle $i, n \in \mathbf{N}$ gilt. Wegen

$$h(i,n) = 0 \cdot u_\varphi(i,n) = \begin{cases} 0 & \text{falls } n \in \mathrm{Def}(\varphi_i), \\ \uparrow & \text{sonst} \end{cases}$$

für alle $i, n \in \mathbf{N}$ ist damit die Behauptung gezeigt.

Aufgabe 4

(a) Wäre die Menge $B \setminus A$ endlich, so wäre sie auch rekursiv, da nach Satz 6.1.3(2) jede endliche Menge rekursiv ist. Da nach Satz 6.3.1(1) die Vereinigung rekursiver Mengen wieder rekursiv ist, wäre dann auch die Menge $A \cup (B \setminus A) = B$ rekursiv. Das ist nach Voraussetzung aber nicht der Fall. Also kann $B \setminus A$ nicht endlich sein.

(b) Nach Satz 6.3.1(1) ist das Komplement rekursiver Mengen wieder rekursiv. Mit B ist also auch $\mathbf{N} \setminus B$ rekursiv. Wäre die Menge $B \setminus A$ endlich, so wäre sie auch rekursiv. Wieder nach Satz 6.3.1(1) wäre auch die Vereinigung $(\mathbf{N} \setminus B) \cup (B \setminus A) = \mathbf{N} \setminus A$ rekursiv, und mit dieser Menge auch ihr Komplement $\mathbf{N} \setminus (\mathbf{N} \setminus A) = A$. Das ist nach Voraussetzung aber nicht der Fall. Also ist $B \setminus A$ auch in diesem Fall unendlich.

(c) Die Menge $K := \{i \in \mathbf{N} \mid i \in \mathrm{Def}(\varphi_i)\}$ ist bekanntlich rekursiv-aufzählbar, aber nicht rekursiv. Wir definieren $A := \{2n \mid n \in K\}$ und $B := \{2n \mid n \in K\} \cup \{2n+1 \mid n \in K\}$. Diese Mengen sind beide rekursiv-aufzählbar, aber nicht rekursiv. A ist eine Teilmenge von B und $B \setminus A$ ist unendlich.
Der Vollständigkeit halber beweisen wir all diese Aussagen. Die Funktionen $f, g : \mathbf{N} \to \mathbf{N}$ mit $f(n) := 2n$ und $g(n) := 2n+1$ für alle $n \in \mathbf{N}$ sind total und berechenbar. Nach Aufgabe 6.3.2(1) im Kurstext ist die Menge $f(K) = A$ rekursiv-aufzählbar, ebenso die Menge $g(K)$. Mit den beiden letztgenannten Mengen ist nach Satz 6.3.1(2) auch ihre Vereinigung $f(K) \cup g(K) = B$ rekursiv-aufzählbar. Wegen $K = f^{-1}[A]$ und da die Menge K nicht rekursiv ist, ist auch A nicht rekursiv (vgl. Satz 6.4.2(1)). Wegen $K = f^{-1}[B]$ ist aus dem gleichen Grund auch die Menge B nicht rekursiv. Da K nicht rekursiv und damit unendlich ist und g injektiv ist, ist die Menge $g(K) = B \setminus A$ unendlich.

Aufgabe 5

(a) Wegen

$$h(n) \leq f(n)$$

für alle $n \in \mathbf{N}$ ist $O(h) \subseteq O(f)$. Wegen

$$2 \cdot h(n) = 2f(n) - 2g(n) = f(n) + (f(n) - 2 \cdot g(n)) \geq f(n)$$

für alle $n \in \mathbf{N}$ ist $f \in O(h)$, also auch $O(f) \subseteq O(h)$. Zusammen erhalten wir $O(h) = O(f)$.

(b) Wir wissen nach Voraussetzung, dass es eine Turingmaschine M_f gibt, die bei Eingabe 0^n die Ausgabe $0^{f(n)}$ liefert und in Zeit $O(f)$ arbeitet, und dass es eine Turingmaschine M_g gibt, die bei Eingabe 0^n die

Ausgabe $0^{g(n)}$ liefert und in Zeit $O(g)$ arbeitet. Wir definieren eine Turingmaschine M_h mit dem Ein-/Ausgabealphabet $\{0\}$ wie folgt. Bei der Eingabe 0^n verfahre sie folgendermaßen:

(1) Sie simuliert die Turingmaschine M_f mit Eingabe 0^n, wobei ein Arbeitsband a als Ausgabeband für M_f benutzt werde.

(2) Auf dem Arbeitsband a gehe der Lese-/Schreibkopf zurück auf das Feld links von dem geschriebenen Wort.

(3) Sie simuliert die Turingmaschine M_g mit Eingabe 0^n, wobei wieder das Arbeitsband a als Ausgabeband für M_g benutzt werde und zusätzlich anstelle von 0'en immer 1'en geschrieben werden sollen. Außerdem soll die Maschine an der gleichen Stelle mit dem Schreiben beginnen wie in Schritt (1).

(4) Sie kopiert alle 0'en von Arbeitsband a auf das Ausgabeband.

Diese Maschine M_h berechnet bei Eingabe 0^n die Ausgabe $0^{h(n)}$. Bei Eingabe 0^n schreibt sie nämlich in Schritt (1) $0^{f(n)}$ auf das Arbeitsband a. In Schritt (2) wandert der Lese-/Schreibkopf von Arbeitsband a zurück auf die Position links von dem geschriebenen Wort $0^{f(n)}$. In Schritt (3) werden $g(n)$ dieser 0'en durch 1'en überschrieben. Dann stehen bei Beginn von Schritt (4) noch $f(n) - g(n) = h(n)$ Nullen auf dem Arbeitsband a. Diese werden auf die Ausgabe kopiert.
Eine Zeitabschätzung für die einzelnen Schritte liefert:

(1) $O(f)$.

(2) $O(f)$.

(3) $O(g)$.

(4) $O(h)$.

Wegen $O(g) \subseteq O(f) = O(h)$ arbeitet die Maschine insgesamt in Zeit $O(h)$. Die Maschine M_h zeigt also, dass h zeitkonstruierbar ist.

Aufgabe 6

Wir nehmen an, es gelte $\mathbf{P} = \mathbf{NP}$. Eine Sprache ist genau dann **NP**-vollständig, wenn sie in **NP** ist und **NP**-hart ist. Wir müssen also zeigen, dass L in **NP** liegt und **NP**-hart ist.
"$L \in \mathbf{NP}$": Ganz ohne die Annahme $\mathbf{P} = \mathbf{NP}$ kann die Sprache L sogar deterministisch in konstanter Zeit erkannt werden. Dazu muss man nur eine Turingmaschine konstruieren, deren Lese-/Schreibkopf auf dem Eingabeband einen Schritt nach rechts geht, und die prüft, ob dort ein Zeichen aus Σ^* steht. Das heißt, es ist $L \in \mathrm{ZEIT}(1) \subseteq \mathbf{P} \subseteq \mathbf{NP}$.
"L ist **NP**-hart": Sei K eine beliebige Sprache in **NP**. Wir müssen zeigen,

dass $K \leq_{\text{pol}} L$ gilt. Aus der Annahme $\mathbf{P} = \mathbf{NP}$ folgt, dass es eine (deterministische) Turingmaschine M mit $K = L(M)$ gibt, die in Polynomzeit arbeitet. Wegen $L(M) = \{w \in \Sigma^* \mid f_M(w) = \varepsilon\}$ gilt

$$w \in K \iff f_M(w) \in L$$

für alle $w \in \Sigma^*$. Da M in Polynomzeit arbeitet, ist $f_M \in \mathbf{FP}$. Also gilt $K \leq_{\text{pol}} L$. Wir haben gezeigt, dass aus der Annahme $\mathbf{P} = \mathbf{NP}$ folgt, dass L **NP**-hart ist.

Aufgabe 7

Wir verwenden die im Beweis von Lemma 11.2.6 eingeführten Transformationen und geben jeweils nur die Regelmenge an. Wir starten mit der Regelmenge R:

$$\begin{aligned} S &\longrightarrow \mathtt{a}T \mid \mathtt{ab} \\ T &\longrightarrow U \mid \mathtt{b} \\ U &\longrightarrow \mathtt{ba}T \end{aligned}$$

Normieren der Terminalregeln liefert:

$$\begin{aligned} S &\longrightarrow \mathtt{a}T \mid \mathtt{ab}E \\ T &\longrightarrow U \mid \mathtt{b}E \\ U &\longrightarrow \mathtt{ba}T \\ E &\longrightarrow \varepsilon \end{aligned}$$

Verkürzung der Regellängen, zweimal angewendet, liefert:

$$\begin{aligned} S &\longrightarrow \mathtt{a}T \mid \mathtt{a}F \\ T &\longrightarrow U \mid \mathtt{b}E \\ U &\longrightarrow \mathtt{b}G \\ E &\longrightarrow \varepsilon \\ F &\longrightarrow \mathtt{b}E \\ G &\longrightarrow \mathtt{a}T \end{aligned}$$

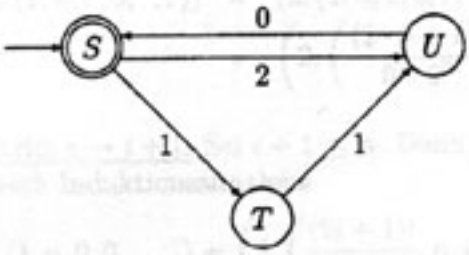
Elimination längentreuer Regeln liefert:

$$\begin{aligned} S &\longrightarrow \mathtt{a}T \mid \mathtt{a}F \\ T &\longrightarrow \mathtt{b}G \mid \mathtt{b}E \\ U &\longrightarrow \mathtt{b}G \\ E &\longrightarrow \varepsilon \\ F &\longrightarrow \mathtt{b}E \\ G &\longrightarrow \mathtt{a}T \end{aligned}$$

Diese Regelmenge sei R' genannt. Die Grammatik $G' = (\Pi', \Sigma, R', S)$ mit $\Pi' := \{S, T, U, E, F, G\}$ ist eine rechtslineare Normalformgrammatik und erfüllt $L(G') = L(G)$.

Aufgabe 8

Einen endlichen Automaten A mit $L(A) = L(G)$ und mit genau drei Zuständen kann man nach Lemma 11.3.5 direkt aus der Grammatik G ablesen:



Aufgabe 9

Angenommen, die Sprache L sei kontextfrei. Dann gibt es eine natürliche Zahl $p \in \mathbf{N}$ mit den im Pumping-Lemma 12.4.3 genannten Eigenschaften. Wir wählen eine Zahl n mit $2^n > p$ und wählen irgendein Wort $z \in \Sigma^*$ der Länge 2^n. Dann gilt $z \in L$ und $\lg(z) \geq p$. Nach dem Pumping–Lemma gibt es Worte $u, v, w, x, y \in \Sigma^*$ mit $z = uvwxy$, $vx \neq \varepsilon$, $\lg(vwx) \leq p$ und $\forall i \in \mathbf{N}.\, uv^iwx^iy \in L$. Wir erhalten einerseits

$$2^n = \lg(z) < \lg(uv^2wx^2y)$$

wegen $vx \neq \varepsilon$, andererseits

$$\lg(uv^2wx^2y) = \lg(z) + \lg(vx) \leq 2^n + p < 2^n + 2^n = 2^{n+1}$$

wegen $\lg(vx) \leq \lg(vwx) \leq p$. Also ist

$$2^n < \lg(uv^2wx^2y) < 2^{n+1}.$$

Damit kann $\lg(uv^2wx^2y)$ keine Zweierpotenz sein. Dies steht im Widerspruch zu $uv^2wx^2y \in L$. Also war die Annahme, dass L kontextfrei ist, falsch.