Grundlagen der Theoretischen Informatik
Lösungshinweise zur Klausur am 05.02.2000

Aufgabe 1

a) Induktionsanfang: Für i = 0 gilt die Behauptung trivialerweise.
Induktionsschritt: Sei b, c, d $\geq$ i + 1. Dann gilt
$ES^{7(i+1)}(2,(a,b,\mathrm{c},d,0,\ldots))$
$= ES^{7i}(2,(a+1, b-1, \mathrm{c}-1, \mathrm{d}-1, 0, \ldots))$
$(2, (a+1+i, b-1-i, c-1-\mathrm{i}, d-1-\mathrm{i}, 0, \ldots))$.

Die erste Gleichung ist direkt nachprüfbar, die zweite gilt nach Induktionshypothese, denn $b-1$, c -1, d $-1 \geq$ i.

b) $f_M(x,y,z) = min(x,y,z)$:
Setzt man $a = 0$, b = x, c = y, d = z und i = $min(x, \mathrm{y}, z)$, so gilt laut a):
$ES^{7i}(2,(0, x, y, z, 0, \ldots) = (2, (i, x-i, y-i, z-i, 0, \ldots))$, wobei mindestens eine der Zahlen x $-i$, y $-i$, $z-$ i gleich 0 ist. Damit geht einer der Tests positiv aus, und die Maschine hält mit Ausgabe i (gemäß Ein- und Ausgabecodierurig der Registermaschinen).

Aufgabe 2

a) μ ist surjektiv, denn: Für x $\geq$ 0 ist $\mu(2x)$ = x, für x < 0 ist $\mu(-2x-1)$ = x.

b) Man muß berechenbare Funktionen f, g : N $\rightarrow$ N finden mit $\mu = \nu_Z f$ und $\nu_Z = \mu g$.
Für f muß also, ausführlich geschrieben, gelten:

$$\mathrm{p(n)} = \nu_Z(f(n)) = \pi_1(f(n)) - \pi_2(f(n)).$$

Setzt man

$$f(n) = \begin{cases} \left\langle \frac{n}{2}, 0 \right\rangle, n \text{ gerade} \\ \left\langle 0, \frac{n+1}{2} \right\rangle, \mathrm{n} \text{ ungerade,} \end{cases}$$

so ist diese Bedingung erfüllt. f ist berechenbar, da der Test auf Teilbarkeit, die Division, die Cantorsche Paarungsfunktion und ihre Umkehrungen berechenbar sind.

Für g muß gelten:

$$\pi_1(n) - \pi_2(n) = \nu_Z(n) = \mu(g(n)) = \begin{cases} \frac{g(n)}{2}, & \text{g(n) gerade} \\ \frac{g(n)+1}{2}, & g(n) \text{ ungerade} \end{cases}$$

Wir setzen

$$g(n) = \begin{cases} 2(\pi_1(n) - \pi_2(n)), & \pi_1(n) \geq \pi_2(n) \\ -2(\pi_1(n) - \pi_2(n)) - 1, & \pi_1(n) < \pi_2(n) \end{cases}$$

g ist berechenbar, Begründung wie oben.

c) Aus dem Kurs ist bereits bekannt, dass die Subtraktion (ν_Z, ν_Z, ν_Z)-berechenbar ist. Wegen $\nu_Z \equiv \mu$ ist nichts weiter zu zeigen.

Aufgabe 3

a) Man kann eine Registermaschine konstruieren, die bei Eingabe n folgendes tut:

- i, x, t mit (i, $x, t\rangle$ = n berechnen,
- $h(i,$ x, $t)$ berechnen - dabei ist h die berechenbare (!) Funktion aus dem Rechenzeit-Lemma,
- wenn $h(i, x, t) > 0$, $j := h(i, x, t) - 1$ setzen. In diesem Fall ist $j = \varphi_i(x)$.
- (i, $j\rangle$ ausgeben, andernfalls ein festes (ie, je) $\in A$ ausgeben.

Zu jedem $\langle i, j\rangle$ mit $j \in$ Bild (φ_i) gibt es ein x mit $j = \varphi_i(x)$, also auch ein n, für das die Maschine $\langle i, j\rangle$ ausgibt. Bezeichnet f die von der Maschine berechnete Funktion, so ist also Bild (f) = A. Nach einer Charakterisierung der rekursiv aufzählbaren Mengen folgt die Behauptung.

b) $A_0 = \{i \mid 0 \in \text{Bild } (\varphi_i)\}$ist nach dem Satz von Rice nicht rekursiv. Wir zeigen $A_0 \leq A$. Sei dazu r(i) = $\langle i, 0)$. r ist trivialerweise berechenbar, und es gilt:

$$i \in A_0 \Leftrightarrow (i, 0) \in A \Leftrightarrow r(i) \in A.$$

Aufgabe 4

a) Für eine Eingabe x der Länge n genügt es, alle y $<$ x daraufhin zu testen, ob y^2 = x ist. Bei naivem Durchprobieren wären das bis zu 2^n Versuche. Da y^2 aber monoton in y ist, kann man jeweils testen, ob y^2 < x oder $y^2 >$ x gilt und binäre Suche anwenden. Dadurch kommt man mit log 2^n = n Versuchen aus. Die Vergleiche brauchen nur O(n) Zeit. Mit einem Multiplikationsverfahren mit $O(n^2)$ Zeitbedarf für jedes Quadrieren kommt man auf die $O(n^3)$-Schranke.

b) Im Kurs wurde ein $O(n^{\log_2 3})$-Zeit-Multiplikationsverfahren vorgestellt. Alles andere ist wie in a).

c) Ja, nach dem Zeithierarchiesatz.
Es gilt $lim_{n\to\infty} \frac{n^{\log_2 3} \cdot \log(n^{log_2 3})}{n^3} = 0$,
und n^3 ist zeitkonstruierbar, somit ist der Satz anwendbar.

Aufgabe 5

Die im Hinweis vorgegebene Reduktionsfunktion ist offensichtlich in Polynomialzeit berechenbar. Sei *F* eine Formel und G die aus *F* konstruierte Eingabe von ALMOST-SAT.

Wir müssen zeigen:

F erfüllbar gdw. G erfüllbar bis auf höchstens eine Klausel („fast erfüllbar“).

Sei *F* erfüllbar. Dann benutzen wir in G dieselbe Belegung wie in *F* und setzen v = 1. Dann ist einzig die Klausel $\neg v$ unerfüllt.

Sei umgekehrt G fast erfüllbar. Wir betrachten eine entsprechende Belegung. Ist darin v = 0, so muß jede Klausel der Form F_i **V** v ein Litera1 mit dem Wert **1** enthalten, sonst wären F_i und F_i **V** v beide unerfüllt.

Das heißt aber, die Belegung erfüllt *F*. Ist v = 1, so ist die Klausel $\neg v$ unerfüllt. Da es nur eine unerfüllte Klausel gibt, muß wiederum jedes F_i erfüllt sein.

Aufgabe 6

Man kann solch ein w aus v erhalten, indem man zunächst v spiegelt und „(„ durch “)“ ersetzt und umgekehrt. Jetzt stehen nur noch eventuell vorkommende Symbole * links von dem Klammerpaar, auf das sie sich beziehen, statt rechts. Daraus ergibt sich fast unmittelbar der Entwurf einer Turingmaschine.

(1) Eingabe von Band 1 in umgekehrter Reihenfolge auf Band 2 kopieren.

(2) Auf Band 2 alle „ („ durch “)“ ersetzen und umgekehrt.

(3) Auf Band 2 jedes Symbol * durch # ersetzen. (Soll einen noch nicht bewegten * symbolisieren.)

(4) Auf Band 2 ein # suchen und solange mit dem jeweils rechts davon stehenden Symbol vertauschen, bis genausoviele „ („ wie “)“überstrichen wurden. Dann steht # an der richtigen Stelle und wird wieder durch * ersetzt. Die Differenz Anzahl der „ („ minus Anzahl der “)“ wird dabei mittels einer Strichliste auf Band 3 verwaltet.

(4) wiederholen, bis kein # mehr vorhanden ist.

(5) Schließlich Band 2 auf Band 0 kopieren.

Aufgabe 7

a) Dies ist bereits klar durch die Form der Regeln. Auf der linken Seite steht nur ein Nichtterminalsymbol.

b) Wenn u = ε bzw. $v = \varepsilon$, so gilt offensichtlich L(G) = cv^* bzw. u^*, also ist L(G) regulär.

Es bleigt zu zeigen, dass L(G) nicht regulär ist, wenn u, $v \neq \varepsilon$. In diesem Fall ist L(G) = $\{u^n c v^n \mid n \in \mathbb{N}\}$.

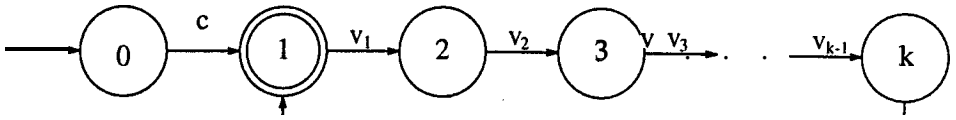
Mit dem Pumping-Lemma für reguläre Mengen wurde im Kurs gezeigt, dass $\{ a^n b^n \mid n \in \mathrm{IN}\}$ nicht regulär ist, wobei a, b einzelne Symbole sind.

Die Beweisidee lässt sich auch auf unsere Sprache anwenden:

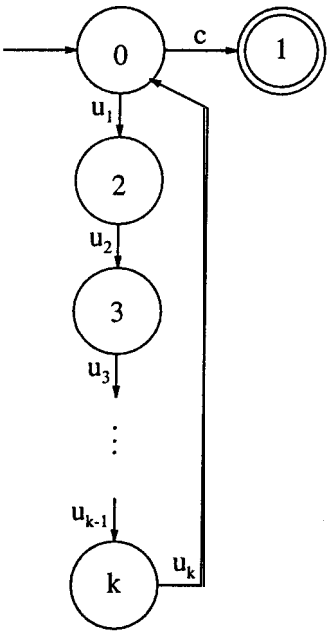
Sei n die Zahl aus dem Pumping-Lemma.

Es gilt $u^n v^n \in$ L(G). Wir wählen $t = \varepsilon$, $z = u^n$, $\bar{t} = cv^n$. Dann kann z in p, q, r zerlegt werden mit $z = pqr$, $q \neq \varepsilon$, und $Vi \geq 0$. $tpq^i r\bar{t} \in$ L(G), insbesondere $tpqr\bar{t} \in L(G)$, im Widerspruch zur Gestalt der Worte in L(G).

c) Sei v = $v_1 \ldots v_k (v_i \in$ C). Dann wird cv^* akzeptiert von



Sei u = $u_1 \ldots u_k (u_i \in$ C). Dann wird u^*c akzeptiert von



Wenn u, v $\neq \varepsilon$, so existiert kein endlicher Automat, der L(G) akzeptiert, da L(G) nicht regulär ist.