Technische Universität Chemnitz Sommersemester 2002
Professur Theoretische Informatik
Prof. Dr. A. Goerdt, Dipl.-Inf. F. Schädlich, Dipl.-Inf. A. Lanka

11. Übung „Theoretische Informatik II“
Lösungsvorschläge

Aufgabe 1

Wir schalten der gegebenen TM M einfach eine andere TM M' vor, die auf das leere Band die Zeichenkette 010 schreibt und dann in die Überführungsfunktion von M wechselt. Dann hält M' genau dann auf dem leeren Band, wenn M bei der Eingabe 010 hält.

Aufgabe 2

$L_1 \setminus L_2$ ist nicht immer rekursiv aufzählbar (= semientscheidbar, Typ-0-Sprache, Turing-berechenbar, While- / Goto-berechenbar).

Zum Beweis genügt es ein Beispiel anzugeben, wo $L_1 \setminus L_2$ nicht rekursiv aufzählbar ist. Die Beweisidee besteht darin, dass mit $L_1 \setminus L_2$ das Komplement $\overline{L_2}$ für $L_1 = \Sigma^*$ darstellbar ist. Eine Sprache L_2 ist genau dann entscheidbar, wenn sie und ihr Komplement semientscheidbar (rekursiv aufzählbar) sind.

Dies wird für ein nichtentscheidbares Problem auf einen Widerspruch geführt.

Sei $L_1 = \Sigma^*$ (ist rekursiv aufzählbar).
Sei L_2 rekursiv aufzählbar (= semientscheidbar), aber nicht entscheidbar (z. B. PCP).
Angenommen $L_1 \setminus L_2 \ = \ \Sigma^* \setminus L_2 \ = \ \overline{L_2}$ sei rekursiv aufzählbar.
$\Rightarrow$ L_2 und $\overline{L_2}$ sind rekursiv aufzählbar.
$\Rightarrow$ L_2 ist entscheidbar.
$\Rightarrow$ Widerspruch zur Wahl von L_2.

Aufgabe 3

Gegeben sei eine Turing-Maschine B_n, welche BB(n) Einsen schreibt und dann stoppt.

Angenommen, es gäbe eine Turing-Maschine C mit c Zuständen, die BB(n) berechnet.

Diese Maschine wird so in eine Turing-Maschine C' mit c' Zuständen umgeformt, so dass vor dem Stoppen (Übergang in den Endzustand) noch eine 1 am Bandende angehängt wird. D. h. C' berechnet $\mathrm{BB}(n)+1$. Sei A eine Turing-Maschine, die die Zahl n binär aufs Band schreibt $(0 = \Box 1, 1 = 11, \Box = \Box\Box)$ und den Schreib-Lese-Kopf danach an der 1. Stelle der Zahl positioniert. Diese Maschine A sollte sich mit ca. $2 \cdot \log n$ Zuständen realisieren lassen. Sei B eine Turing-Maschine, die binäre Zahlen $(0 = \Box 1, 1 = 11, \Box = \Box\Box)$ in unäre umwandelt und den Kopf wieder an der ersten Stelle positioniert. Diese Maschine habe b Zustände.

Die 3 Turing-Maschinen werden nun hintereinandergeschaltet:
$\Rightarrow$ $A; B; C$ hat die gleiche Funktionalität wie $(\mathrm{BB}(n) + 1)$ und hat dabei $\log_2 n + c' + b$ Zustände.

Für hinreichend große n ist dies jedoch weniger als n.

Die Turing-Maschine wird nun so gewählt, daß sie die Anzahl n von Zuständen der Maschine ABC' auf das Band schreibt. Die TM ABC' schreibt nun genau $\mathrm{BB}(n) + 1)$ Einsen auf das leere Band. Dies ist eine 1 mehr, als C für $\mathrm{BB}(n)$ ausgerechnet hat. $\Rightarrow$ Widerspruch

Bemerkung:
Für das Verständnis des Beweises sind die folgenden beiden Sätze hilfreich. Der Beweis der Richtigkeit verbleibt als Übung.

- Es gibt eine Turing-Maschine, die die unäre Zahl 1^n auf das leere Band schreibt, für die gilt: $\Gamma = \{1, \Box\},\ |Z| = O(\log_2 n)$
- Jede Turing-Maschine kann von einer Turing-Maschine mit dem Bandalphabet $\Gamma = \{1, \Box\}$ simuliert werden. Dabei ist die Anzahl der Zustände entsprechend zu vergrößern und die Überführungsfunktion anzupassen.

Aufgabe 4

Wir gehen davon aus, daß Eingabe der TM in den Registern der RAM steht. Wenn auf einer Bandposition das Zeichen a_j steht, so soll im entsprechenden Register die Zahl j stehen.

Wir benutzen zwei weitere Register, um uns die Kopfposition der TM zu merken. Eines für das Vorzeichen und eines für den Betrag der Position. Um auch auf den Teil des Bandes zugreifen zu können, der links von der Eingabe steht, benutzen wir jedes Register für zwei Bandpositionen, nämlich i und $-i$. Jedes Register ist in seinem Inhalt durch das Bandalphabet begrenzt. Also können wir aus dem entsprechenden Register j den Bandinhalt von Position i errechnen „`c(j) MOD b`“ bzw. bei $-i$ „`c(j) DIV b`“. Da wir nicht wissen, wie lang unsere Eingabe ist, sollte die Kopfposition vor der Eingabe stehen. Den Platz für diese beiden Register haben wir allerdings nicht. Wir müssen uns also zuerst Platz schaffen und alle Register zwei Stellen nach rechts verschieben. Dabei entsteht das Problem, daß wir nicht wissen, wann die Eingabe beendet wird. Deswegen kodieren wir die ersten k Register in $c(1)$, um genug Platz zu haben, das Ende der Eingabe zu bestimmen und alles zwei Plätze nach rechts zu verschieben. Steht in den ersten Registern $i_1, i_2, \ldots, i_k$, so speichern wir in $c(1)$ einfach $[i_1 i_2 \ldots i_k]_b$. Also als eine Zahl zur Basis b. Jetzt haben wir uns beliebig Platz geschaffen, um die Registerinhalte um einen beliebigen Wert nach rechts zu verschieben und unsere Kopfposition (und andere wichtige Informationen, die auftreten können) zu speichern. Beginnen wir nun mit der Simulation.

Bei einem Zustandswechsel, springen wir einfach an die entsprechende Stelle des RAM-Programms und ändern die Inhalte der Register passend zu den Bandmodifikationen der TM. Dafür bietet sich die indirekte Adressierung an. Danach ändern wir das Register, das die Kopfposition enthält, entsprechend.

Im uniformen Kostenmaß brauchen wir für jeden Schritt der TM eine konstante Zahl von Simulationsschritten. Da die TM $t(n)$-zeitbeschränkt ist, haben wir $O(t(n))$ viele Simulationsschritte. Dazu kommen noch $O(n)$ viele Schritte um die Eingabe zu verschieben. Im logarithmischen Kostenmaß, müssen wir bedenken, daß die Maschine jedes Mal die Kopfposition aktualisieren muß. Die größte mögliche Position ist $\max(t(n), n)$, also kostet eine Kopfaktualisierung höchstens $O(\log(\max(t(n), n)))$, was wir durch $O(\log(t(n) + n))$ ausdrücken. Damit ergibt sich als Aufwand im logarithmischen Kostenmaß $O((t(n) + n) \cdot \log(t(n) + n))$.