Gruppe **17**
Berechenbarkeit & Komplexität

Philipp Hochmann, 356148
Felix Kiunke, 357322

Blatt 7

Aufgabe 7.1

(a)

RAM-Befehl: MULT 1 ($c(0) := c(0) \cdot c(1)$)

Äquivalentes RAM-Programm mit eingeschränktem Befehlssatz (mit $l = k + 1$, $m = k + 2$ der Lesbarkeit halber):

		Anmerkungen
1	STORE k	Speichere ersten Faktor in $c(k)$
2	GOTO 15	Springe zur Speicherung von $c(1)$ in $c(l)$ und Prüfung $c(k) \neq 0$
3	GOTO 10	Äußere Schleife: Springe zur Prüfung $c(l) \neq 0$
4	LOAD m	Innere Schleife: $c(k) > 0$ und $c(l) > 0$, also addiere 1 auf $c(m)$
5	CADD 1	
6	STORE m	
7	LOAD l	Subtrahiere 1 von $c(l)$
8	CSUB 1	
9	STORE l	
10	LOAD l	Prüfe, ob $c(l) > 0$, wenn ja, wiederhole die „innere Schleife“
11	IF $c(0) \neq 0$ THEN GOTO 4	
12	LOAD k	Äußere Schleife: Subtrahiere 1 von $c(k)$
13	CSUB 1	
14	STORE k	
15	LOAD 1	Setze $c(l)$ zurück auf $c(1)$
16	STORE l	
17	LOAD k	Prüfe, ob $c(k) > 0$, wenn ja, wiederhole die „äußere Schleife“
18	IF $c(0) \neq 0$ THEN GOTO 3	
19	LOAD m	Lies das Ergebnis aus $c(m)$ in $c(0)$

Das Programm besteht aus zwei Schleifen. In der äußeren wird der erste Faktor $c(0)$ in $c(k)$ heruntergezählt und somit die innere Schleife $c(0)$-mal ausgeführt. In der inneren Schleife wird der zweite Faktor $c(1)$ in $c(l)$ heruntergezählt und jedesmal 1 auf $c(m)$ unseren „Akkumulator“ addiert, also insgesamt $c(m) := c(m) + c(l)$. Damit wird $c(l)$ $c(0)$-mal addiert, also ist nach dem letzten Durchlauf der äußeren Schleife $c(m) = c(0) \cdot c(1)$. Zum Schluss wird das Ergebnis aus $c(m)$ in das Akkumulatorregister $c(0)$ geladen.

(b)

Idee: man legt für jedes der $k-1$ Register, die von 0 verschieden sind, einen eigenen Codeabschnitt an, der dieses Register lädt. $c(k-1)$ ist das letzte Register, in dem ein Wert ungleich 0 stehen kann. Wenn in $c(i) \geq k$, wird nichts geladen.

1: LOAD i
2: CSUB 1
3: IF c(0) $\neq$ 0 THEN GOTO 6
4: LOAD 1
5: GOTO $4k+1$
$\vdots$
$4 \cdot l + 2$: CSUB 1
$4 \cdot l + 3$: IF c(0) $\neq$ 0 THEN GOTO $4 \cdot (l+1) + 2$
$4 \cdot l + 4$: LOAD l
$4 \cdot l + 5$: GOTO $4k+1$
$\vdots$
$4(k-1) + 2$: CSUB 1
$4(k-1) + 3$: IF c(0) $\neq$ 0 THEN GOTO $4k+1$
$4(k-1) + 4$: LOAD $k-1$
$4k+1$: *hier geht das Programm weiter*

Aufgabe 7.2

(a)

Da LOOP-Programme berechenbar sind und durch die festgelegte Anzahl Iterationen (wie in der VL gezeigt) immer terminieren, kann ein gegebenes LOOP-Programm einfach mit der Eingabe x in endlicher Zeit simuliert werden. Die Ausgabe kann dann mit y verglichen werden. Da y endlich ist, kann dieser Vergleich ebenfalls in endlicher Zeit stattfinden. Also ist das Problem entscheidbar.

(b)

Wie in der Vorlesung gezeigt, ist jedes eingeschränkte RAM-Programm in ein WHILE-Programm konvertierbar; ebenso sind Turingmaschinen und RAM mit eingeschränktem Befehlssatz äquivalent. Damit lässt sich jede Turingmaschine als RAM und somit auch als WHILE-Programm darstellen.

Ob eine Turingmaschine zu einer Eingabe x die Ausgabe y produziert, ist wie in der VL gezeigt nicht entscheidbar nach Satz von Rice. Also ist auch das entsprechende WHILE-Programm nicht entscheidbar. Damit ist das gegebene Problem im Allgemeinen nicht entscheidbar.

Aufgabe 7.3

(a)

Diese Aussage trifft zu, da die Sprache A_{LOOP} entscheidbar ist. Sie ist insbesondere nicht schwieriger, als das Halteproblem. Eine Reduktion sähe so aus, dass eine Abbildung $\langle P \rangle$ simuliert. LOOP-Programme haben eine feste Laufzeit und es ist daher entscheidbar, ob bei Eingabe 0 das Ergebnis 1 ist. Wenn ja, wird auf $\langle M_1 \rangle$ abgebildet, wenn nicht, auf $\langle M_2 \rangle$, wobei M_1 immer hält, und M_2 nie.

(b)

A_{LOOP} ist entscheidbar. Gäbe es eine Reduktion auf H, wäre somit das Halteproblem entscheidbar. Aus der Vorlesung ist bekannt, dass das Halteproblem nicht entscheidbar ist. Deshalb stimmt die Aussage nicht.

Aufgabe 7.4

$(IA) A(m+1, 0) = A(m, 1) > A(m, 0)$
$\quad A(1, n) = n + 2 > n + 1 = A(0, n)$

(IV) Die Bedingung gelte für (m', n') mit m' < m oder m' ≤ m und n' < n
(IS) A(m+1, n) = A(m, A(m+1, n-1)) > A(m, A(m, n-1)) > A(m-1, A(m, n-1)) = A(m, n)