10 Übung zu Informatik zum 8.7.2010 Blatt 10

10.1

(A) Hält ein Algorthimus bei jeder Eingabe ungerader Länge?
A=$\{$i—$\varphi_i(x)$ ist total, x ungerade$\}$
A respektiert Funktionen: $i, j \in \mathbb{N}_0$, $i \in A \Rightarrow \varphi_i(x)$ ist total bei x ungerade. Da $\varphi_i = \varphi_j$ ist auch φ_j total bei x ungerade. $\Rightarrow \varphi_j \in A$
Nach Satz von Rice ist (A) entscheidbar, wenn $A = \emptyset$ oder $A = \mathbb{N}_0$
$A \neq \emptyset$, da Turing-Maschinen existieren, die immer (für alle Eingaben) anhalten, auch bei ungeraden Eingaben.
$A \neq \mathbb{N}_0$, da Turing-Maschinen existieren, die niemals (für keine Eingabe) anhalten, auch bei ungeraden Eingaben.
Also ist (A) nicht entscheidbar

(B) Hält eine beliebige Turingmaschine beginnend mit leerem Eingabeband innerhalb der ersten 100 Schritte?
Lösungsalgorithmus: man führe die ersten 100 Schritte der Turing-Maschine aus, gebe Wahr zurück, wenn Sie darin anhält, Falsch, wenn nicht. Mit diesem Algorithmus kann Für alle Turingmaschinen nach 100 Schritten entschieden werden, ob sie anhalten oder nicht.
(B) ist also entscheidbar.

(C) Gibt ein beliebiger Algorithmus bei geeigneter Eingabe eine Zahl aus, die ein Palindrom ist?
C=$\{i|\exists x : \varphi_i(x)$ ergibt Palindrom$\}$
C respektiert Funktionen: $i, j \in \mathbb{N}_0$, $i \in C \Rightarrow \exists x : \varphi_i(x)$ erzeugt Palindrom. Da $\varphi_i = \varphi_j$ erzeugt auch $\varphi_j(x)$ ein Palindrom bei x. $\Rightarrow \varphi_j \in C$
$C \neq \emptyset$, da es Turing-Maschinen gibt, die für alle Eingaben ein Palindrom erzeugen
$C \neq \mathbb{N}_0$, da es Turing-Maschinen gibt, die für keine Eingabe ein Palindrom erzeugen
Nach Rice ist (C) also nicht entscheidbar.

(D) Liefert ein Algorithmus, der für wenigstens eine Eingabe nicht hält, jede natürliche Zahl als Ausgabewert? D=$\{i|$Wertebereich von φ_i ist $\mathbb{N}_0$, $\exists$ k:$\varphi_i(k)$ nicht total $\}$
D respektiert Funktionen: $i, j \in \mathbb{N}_0$, $i \in D \Rightarrow \varphi_i(\mathbb{N}_0) = \mathbb{N}_0$ und φ_i hält für min. eine Eingabe nicht. Da $\varphi_i = \varphi_j$ ist auch $\varphi(\mathbb{N}_0) = \mathbb{N}_0$ und φ_j hält für min. eine Eingabe nicht. $\Rightarrow \varphi_j \in D$
$D \neq \mathbb{N}_0$, da min. eine Turing-Maschine existiert, die für alle Eingaben hält.
$D \neq \emptyset$, da beispielsweise die Funktion $f : \mathbb{N} \to \mathbb{N}_0$, $f(x) = x - 1$, $\mathbb{N}_0$ als Ausgabe enthält und für $x = 0$ nicht definiert ist, also nicht total für dieses Element.
Damit ist nach Rice (D) nicht entscheidbar.

10.2

Die Rückgabe soll jeweils in x_0 stehen, nachdem das Programm ausgeführt wurde.

a) Es wird geprüft, ob x_1 größer als x_2 ist. Ist dies der Fall, ist x_1 das maximale Element, andernfalls x_2. Darum wird in den jeweiligen Zweigen der IF-Bedingung der Ausgabe x_0 im ersten Fall x_1 und sonst x_2 zugewiesen.

IF $x_1 > x_2$ THEN $x_0 := x_1$;
ELSE $x_0 := x_2$;
FI

b) Grundidee ist, die größte natürliche Zahl anzugeben, deren Quadrat kleiner gleich x1 ist, diese darauf zu prüfen, ob sie die exakte Wurzel ist und im negativen Fall um 1 zu erhöhen, um die aufgerundete Wurzel zu erhalten.
Zunächst müsste man also die ersten $\lceil \frac{x_1}{2} \rceil$ Zahlen durchlaufen, um sicher zu sein, das darunter die gesuchte Wurzel (aufgerundet) ist. Da Division für diese Aufgabe nicht erlaubt und deren Einführung etwa so aufwändig wie der Kompromiss einfach die ersten x_1 Zahlen zu durchlaufen ist, wird letzteres gemacht.
Mit dieser Beschränkung zählt man x_0 hoch, bis dessen Quadrat größer ist als x_1.

$x_0 := 0$;
LOOP x_1 DO
IF $x_0 \cdot x_0 < x_1$ THEN $x_0 := x_0 + 1$; FI
OD

c) Hier macht man zunächst eine Fallunterscheidung, da man bei $x_2 = 0$ eine 0 ausgeben muss,. Andernfalls nähert man sich hier ähnlich wie in b) dem Produkt aus x_1 und x_2 mit x_0, jedoch hier von oben, da man nun den abgerundeten Wert der Division benötigt und das Ergebnis, sofern Quotient nicht ganzzahlig, in jedem Fall überlaufen.
Man setzt also x_0 im Fall $x_2 \neq 0$ auf x_1 und zieht solange davon 1 ab, bis das Produkt mit x_2 kleiner als x_1 ist.

$x_0 := 0$;
IF $x2 > 0$ THEN
$x_0 := x_1$;
LOOP x_1 DO
IF $x_0 \cdot x_2 > x_1$ THEN $x_0 := x_0 \dot{-} 1$; FI
OD
FI

10.3

Man kann das Problem darauf zurückführen, dass man 5 Variabeln hat, die entweder dafür stehen, 2 mal 1L oder 1 mal 2L geschöpft zu haben.
Ist dem Betrachter egal, in welcher Reihenfolge er die 10L mit den beiden Behältnissen ausschöpft, so kann er es wiefolgt leeren:

11	11	11	11	11
2	11	11	11	11
2	2	11	11	11
2	2	2	11	11
2	2	2	2	11
2	2	2	2	2

Ergibt 6 Möglichkeiten

Ist die Reihenfolge nicht egal, so kann die Antwort auf die Konstruktion von Wahrheitstabellen zurückgeführt werden, in der auch jede Zusammenstellung der Variabeln eine Zeile erhalten muss, und so kann man durch die Zurückführung auf 5 Variabeln, die entweder 1*2L oder 2*1L sind, die bekannte Formel für die Anzahl der Zeilen mit n Variabeln anwenden:
2^n mit n=5, also 32 Möglichkeiten bei Beachtung der Reihenfolge.