Universität Siegen
Lehrstuhl Theoretische Informatik
Markus Lohrey

Musterlösung zu Übungsblatt 12

Aufgabe 1. Geben Sie (formal) Turingmaschinen M_1 bzw. M_2 an, die die Funktionen $f_i : \mathbb{N}^2 \to \mathbb{N}$ mit

$$f_i(n_1, n_2) = n_i \quad (i = 1, 2)$$

berechnen.

Lösung zu Aufgabe 1.
Zu Beginn steht $\text{bin}(n_1)\#\text{bin}(n_2)\#$ auf dem Band. Nach einem Durchlauf der TM für $f_i, i \in \{1, 2\}$ soll nur noch $\text{bin}(n_i)$ auf dem Band stehen.

Beachte: Der Lesekopf der TM muss nach dem Durchlauf auf dem ersten Symbol von $\text{bin}(n_1)$ bzw. $\text{bin}(n_2)$ stehen.

TM für f_1

- $M_1 = (Z_1, \Sigma_1, \Gamma_1, \Box, \delta_1, z_0, \{z_e\})$
- $Z_1 = \{z_0, z_1, z_2, z_3, z_e\}$
- $\Sigma_1 = \{0, 1, \#\}$
- $\Gamma_1 = \Sigma \cup \{\Box\}$

Dabei ist δ_1 wie folgt definiert:

z_0: Überspringen von $\text{bin}(n_1)$

- $\delta_1(z_0, x) = (z_0, x, R)$ für alle $x \in \{0, 1\}$
- $\delta_1(z_0, \#) = (z_1, \#, R)$

Wir durchlaufen das Band von links nach rechts, bis wir auf ein $\#$ treffen. Dann gehen wir in z_1 über um den restlichen Bandinhalt zu löschen.

z_1: Löschen des restlichen Bandinhaltes

- $\delta_1(z_1, x) = (z_1, \Box, R)$ für alle $x \in \{0, 1, \#\}$

- $\delta_1(z_1, \Box) = (z_2, \Box, L)$

Alle Symbole von $\text{bin}(n_2)\#$ werden mit $\Box$ überschrieben. Wenn das rechte Bandende erreicht ist, gehen wir in z_2 über um zurück zum linken Bandende zu laufen.

<u>z_2, z_3: Zurück zum linken Bandende</u>

- $\delta_1(z_2, \Box) = (z_2, \Box, L)$
- $\delta_1(z_2, \#) = (z_3, \Box, L)$
- $\delta_1(z_3, x) = (z_3, x, L)$ für alle $x \in \{0, 1\}$
- $\delta_1(z_3, \Box) = (z_e, \Box, R)$

Hier brauchen wir zwei Zustände, da die $\Box$'s rechts und links von $\text{bin}(n_1)$ unterschiedlich behandelt werden.

Das mittlere $\#$ haben wir zu Beginn mit z_0 stehen gelassen um diesen Übergang zu markieren.

TM für f_2

- $M_2 = (Z_2, \Sigma_2, \Gamma_2, \Box, \delta_2, z_0, \{z_e\})$
- $Z_2 = \{z_0, z_1, z_2, z_e\}$
- $\Sigma_2 = \{0, 1, \#\}$
- $\Gamma_2 = \Sigma_2 \cup \{\Box\}$

Dabei ist δ_2 wie folgt definiert:

<u>z_0: Löschen von $\text{bin}(n_1)$</u>

- $\delta_2(z_0, x) = (z_0, \Box, R)$ für alle $x \in \{0, 1\}$
- $\delta_2(z_0, \#) = (z_1, \Box, R)$

<u>z_1: Löschen des letzten $\#$</u>

- $\delta_2(z_1, x) = (z_1, x, R)$ für alle $x \in \{0, 1\}$
- $\delta_2(z_1, \#) = (z_2, \Box, L)$

z_2: Zurück zum Anfang von $\text{bin}(n_2)$

- $\delta_2(z_2, x) = (z_2, x, L)$ für alle $x \in \{0, 1\}$
- $\delta_2(z_2, \square) = (z_e, \square, R)$

Aufgabe 2. Wahr oder falsch?

(a) Das folgende LOOP-Programm terminiert nicht.

$$x_1 := 5;\ \textsc{Loop}\ x_1\ \textsc{Do}\ x_1 := x_1 + 1;\ \textsc{End}$$

(b) Das folgende WHILE-Programm berechnet die Funktion $f(x) = 0$.

$$\textsc{While}\ x \neq 0\ \textsc{Do}\ x := x - 2; x := x + 1;\ \textsc{End}$$

Lösung zu Aufgabe 2.

(a) falsch

Der LOOP wird $x_1 = 5$ mal ausgeführt, das veränderte x_1 wird nicht erneut gelesen. Grundsätzlich gilt: LOOP-Programme terminieren immer.

(b) falsch

Das Programm berechnet $f(x) = \begin{cases} 0 & \text{falls } x = 0 \\ \text{undef} & \text{falls } x \geq 1 \end{cases}$

Fall 1, $x = 0$

Das Programm terminiert mit einem Rückgabewert von 0.

Fall 2, $x \geq 1$

Das Programm terminiert nicht. Falls $x \geq 2$, dekrementiert ein Durchlauf der Schleife den Wert um 1 bis $x = 1$ erreicht ist.

Für $x = 1$ ist der Wert nach einem Durchlauf der Schleife aber wieder 1, da wir beim Subtrahieren nicht ins negative, sondern nur bis zur 0 gehen (also $a - b = 0$ für $b \geq a$). Somit ergibt $x := x - 2$ zuerst $x = 0$ und anschließend erhalten wir durch $x := x + 1$ wiederum $x = 1$, was zu einer Endlosschleife führt.

Aufgabe 3.

(a) Schreiben Sie ein LOOP-Programm, das für eine Zahl n die n-te *Fibonacci-Zahl* berechnet.

(b) Schreiben Sie ein LOOP-Programm, das die Funktion $f(x, y) = x^y$ für $x \neq 0$ berechnet .

(c) Schreiben Sie ein LOOP-Programm, das die Funktion $f(x, y) = \max(x, y)$ berechnet.

(d) Schreiben Sie ein LOOP-Programm, das die Funktion $f(x, y, z) = \min(x, y, z)$ berechnet.

(e) Schreiben Sie ein WHILE-Programm, das für eine gegebene Zahl $n \geq 2$ den kleinsten Teiler p von n mit $p \geq 2$ ausgibt.

(f) Schreiben Sie ein WHILE-Programm, das die Funktion $f(n) = \lceil\sqrt{n}\rceil$ berechnet.

(g) Schreiben Sie ein GOTO-Programm für Aufgabe 3(c).

Lösung zu Aufgabe 3.

(a) Wir verwenden die folgende Definition der Fibonacci-Zahlen:

$$\mathrm{fib}(n) = \begin{cases} 0 & \text{falls } n = 0 \\ 1 & \text{falls } n = 1 \\ \mathrm{fib}(n-2) + \mathrm{fib}(n-1) & \text{falls } n \geq 2 \end{cases} \tag{1}$$

Zu Beginn steht n in x_1.

In x_1 und x_2 speichern wir $\mathrm{fib}(i)$ und $\mathrm{fib}(i+1)$, beginnend mit $i = 0$. Mit jeder Iteration setzen wir $x_1 := x_2$ und $x_2 := x_1 + x_2$.

$$\begin{array}{l} x_4 := x_1; \\ x_1 := 0; \\ x_2 := 1; \\ \textsc{Loop}\ x_4\ \textsc{Do} \\ \quad x_3 := x_1 + x_2; \\ \quad x_1 := x_2; \\ \quad x_2 := x_3 \\ \textsc{End} \end{array}$$

Beachte: Das Ergebnis steht immer in x_1! (Siehe Folie 374).

(b) $f(x, y) = x^y$. Zu Beginn steht x in x_1 und y in x_2.

$$
\begin{aligned}
&x_3 := x_1;\\
&x_1 := 1;\\
&\textsc{Loop}\ x_2\ \textsc{Do}\\
&\quad x_1 := x_1 \cdot x_3\\
&\textsc{End}
\end{aligned}
$$

(c) $f(x, y) = \max(x, y)$. Zu Beginn steht x in x_1 und y in x_2

$$
\begin{aligned}
&x_3 := x_2;\\
&x_3 := x_3 - x_1; \qquad // \textsc{Loop}\ x_1\ \textsc{Do}\ x_3 := x_3 - 1\ \textsc{End};\\
&\textsc{Loop}\ x_3\ \textsc{Do}\ x_1 := x_2\ \textsc{End}
\end{aligned}
$$

Im Kommentar wird $x_3 - x_1$ $(y - x)$ nochmal ursprünglich berechnet, analog zur Addition aus der Vorlesung.

Eine Besonderheit dabei ist, dass $y - x = 0$, falls $x \geq y$.

Das heißt, dass im Fall $y > x$ der zweite Loop (mindestens einmal) ausgeführt wird und der „Rückgabewert“ mit y (x_2) überschrieben wird.

(d) $f(x, y, z) = \min(x, y, z)$. Zu Beginn steht x in x_1, y in x_2 und z in x_3.

$$
\begin{aligned}
&x_4 := x_2;\\
&x_4 := x_4 - x_3;\\
&\textsc{Loop}\ x_4\ \textsc{Do}\ x_2 := x_3\ \textsc{End};\\
&x_4 := x_1;\\
&x_4 := x_4 - x_2;\\
&\textsc{Loop}\ x_4\ \textsc{Do}\ x_1 := x_2\ \textsc{End};
\end{aligned}
$$

Wir verwenden eine ähnliche Technik wie bei Aufgabenteil (c).

Mit den ersten drei Zeilen prüfen wir, ob $x_2 > x_3$. Falls ja, ist $x_4 > 0$ und wir überschreiben x_2 mit x_3 ($x_2 := \min(y, z)$).

Dann wiederholen wir den Vorgang mit x_1 und x_2. Am Ende steht in x_1 der Wert $\min(x, \min(y, z)) = \min(x, y, z)$.

(e) Wir definieren zunächst einige Funktionen. Dies ist auch in der Klausur möglich, solange nur die erlaubte Syntax verwendet wird.

Definiere wie oben bereits verwendet $x_i := x_j - x_k$ (für $i \neq k$) durch

$x_i := x_j$;
LOOP x_k DO $x_i := x_i - 1$ END

Definiere $x_i := x_j \bmod x_k$ (für $i \neq j$ und $i \neq k$) durch

$x_i := 0$;
LOOP x_j DO
  $x_i := x_i + 1$;
  $x_h := x_k - x_i$;   // h verschieden von i, j, k
  IF $x_h = 0$ THEN $x_i := 0$ END;
END

Idee: Wir inkrementieren x_i x_j-mal um 1. Nach jedem Durchlauf der Schleife prüfen wir mit Hilfe der Hilfsvariable x_h, ob $x_i = x_k$ gilt (also $x_k - x_i = 0$). Falls ja, setzen wir x_i auf 0 zurück.

Zu Beginn steht n in x_1. Falls p ein Teiler von n ist, gilt $n \bmod p = 0$.

$x_2 := 1$;   // Speicher für $x_1 \bmod x_3$
$x_3 := 1$;   // Zähler für den Teiler p
WHILE $x_2 \neq 0$ DO
  $x_3 := x_3 + 1$;
  $x_2 := x_1 \bmod x_3$
END;
$x_1 := x_3$ // p in x_1 kopieren

(f) $f(n) = \lceil \sqrt{n} \rceil$, n steht in x_1.

```
x₂ := 1;                          // WHILE breaker
x₃ := 0;                          // Counter c
x₄ := 1;                          // Speicher für c²
x₅ := 1;                          // Speicher für n − c²
WHILE x₂ ≠ 0 DO
   x₃ := x₃ + 1;
   x₄ := x₃ · x₃;
   x₅ := x₁;
   x₅ := x₅ − x₄;
   IF x₅ = 0 THEN x₂ := 0 END
END;
x₁ := x₃
```

Idee: Wir zählen c hoch und berechnen nach jedem Schritt $n - c^2$. Im Fall $c^2 \geq n$ (also $c = \lceil \sqrt{n} \rceil$) ist x_5 dann 0, wir beenden die WHILE-Schleife und speichern c im x_1 (dem Rückgabewert).

Alternativ können wir uns den "WHILE breaker" sparen und direkt im WHILE prüfen, ob $c^2 \geq n$.

```
x₂ := 0;                          // Counter c
x₃ := 0;                          // Speicher für c²
x₄ := x₁;                         // Speicher für n − c²
WHILE x₄ ≠ 0 DO
   x₂ := x₂ + 1;
   x₃ := x₂ · x₂;
   x₄ := x₁;
   x₄ := x₄ − x₃;
END;
x₁ = x₂
```

(g)

$$
\begin{aligned}
& x_3 := x_1; \\
& x_4 := x_2; \\
\\
M_1 : \ & \textsc{If } x_3 = 0 \textsc{ Then Goto } M_2 \textsc{ End}; \\
& \textsc{If } x_4 = 0 \textsc{ Then Halt End}; \\
& x_3 := x_3 - 1; \\
& x_4 := x_4 - 1; \\
& \textsc{Goto } M_1; \\
\\
M_2 : \ & x_1 := x_2; \\
& \textsc{Halt};
\end{aligned}
$$

Idee: Zu Beginn steht x in x_1 und y in x_2. Wir zählen parallel beide Zahlen (x in x_3 und y in x_4) runter.

Falls x_3 zuerst 0 wird, gilt $\max(x, y) = y$ und wir springen zu M_2, um x_1 (den Rückgabewert) mit x_2 zu überschreiben, und beenden das Programm.

Wird x_4 zuerst 0, ist $\max(x, y) = x$. Da x schon in x_1 steht, können wir das Programm direkt beenden.