Lösungsblatt 2

Aufgabe 1

Geben Sie eine DTM M an, die die Nachfolgerfunktion

$$s\colon \mathbb{N} \to \mathbb{N}, n \mapsto n+1$$

berechnet.

Lösung

Eine Funktion $f\colon \mathbb{N}^k \to \mathbb{N}$ wird von einer DTM $M = (Q, \Sigma, \Gamma, \delta, s, \square, F)$ berechnet, wenn gilt:

$$f(m_1, \ldots, m_k) = n \iff \exists f \in F\colon s\,\mathrm{bin}(m_1)\#\ldots\#\,\mathrm{bin}(m_k) \vdash^* f\,\mathrm{bin}(n),$$

wobei $\mathrm{bin}\colon \mathbb{N} \to \{0,1\}^*$ die Funktion ist, die einer natürlichen Zahl ihre eindeutige Binärdarstellung ohne führende Nullen zuordnet, z. B. $\mathrm{bin}(0) = 0$, $\mathrm{bin}(6) = 110$ und $\mathrm{bin}(11) = 1011$.

Für die Nachfolgerfunktion wähle beispielsweise die DTM

$$M = (\{A, B, C, D\}, \{0,1\}, \{0, 1, \square\}, \delta, A, \square, \{D\})$$

mit folgender Überführungsfunktion δ:

δ	0	1	$\square$
A	$(A, 0, R)$	$(A, 1, R)$	$(B, \square, L)$
B	$(C, 1, L)$	$(B, 0, L)$	$(D, 1, N)$
C	$(C, 0, R)$	$(C, 1, L)$	$(D, \square, R)$
D	$(D, 0, N)$	$(D, 1, N)$	$(D, \square, N)$

Alternativ kann M auch grafisch als Zustandsübergangsdiagramm angegeben werden.

Aufgabe 2

Geben Sie ein LOOP-Programm P an, das die Maximumfunktion

$$\max\colon \mathbb{N}^2 \to \mathbb{N}, (m,n) \mapsto \begin{cases} n, & \text{falls } m \leq n \\ m & \text{sonst} \end{cases}$$

berechnet.

Lösung

```
P = x_0 := x_2 + 0;
    LOOP x_1 DO
      x_0 := x_0 - 1
    END;
    LOOP x_1 DO
      x_0 := x_0 + 1
    END
```

Aufgabe 3

Wir betrachten die ganzzahlige Division

$$\mathrm{div}\colon \mathbb{N}^2 \to \mathbb{N}, (m,n) \mapsto \left\lfloor \frac{m}{n} \right\rfloor,$$

wobei $\mathrm{div}(m,n)$ nur für $(m,n) \in \mathbb{N}^2$ mit $n \neq 0$ definiert sei.

1. Geben Sie ein WHILE-Programm P an, das div berechnet.
2. Geben Sie ein GOTO-Programm P an, das div berechnet.
3. Ist div
   (a) Turing-berechenbar?
   (b) LOOP-berechenbar?
4. Sei $f\colon \mathbb{N} \to \mathbb{N}$ eine Funktion, die nicht Turing-berechenbar ist. Für welche $c \in \mathbb{N}$ ist $cf\colon \mathbb{N} \to \mathbb{N}, n \mapsto cn$ Turing-berechenbar?

Lösung

1.
```
P = x_1 := x_1 + 1;
    WHILE x_1 ≠ 0 DO
      x_0 := x_0 + 1;
      LOOP x_2 DO
        x_1 := x_1 - 1
      END
    END;
    x_0 := x_0 - 1
```

2.
```
P = M_1: x_1 := x_1 + 1;
    M_2: IF x_1 = 0 THEN GOTO M_9;
    M_3: x_0 := x_0 + 1;
    M_4: x_3 := x_2 + 0;
    M_5: IF x_3 = 0 THEN GOTO M_2;
    M_6: x_1 := x_1 - 1;
    M_7: x_3 := x_3 - 1;
    M_8: GOTO M_5;
    M_9: x_0 := x_0 - 1;
    M_10: HALT
```

3. (a) Ja.

   Wir wissen aus den vorherigen Teilaufgaben, dass div WHILE- bzw. GOTO-berechenbar ist. Aus der Vorlesung (Einheit 8) wissen wir, dass solche Funktionen auch Turing-berechenbar sind.

   (b) Nein.

   Aus der Vorlesung (Einheit 7) wissen wir, dass alle LOOP-berechenbaren Funktionen total sind. Da div nicht total ist, kann sie nicht LOOP-berechenbar sein.

4. Nur für $c = 0$.

   Für $c = 0$ ist cf die konstante Nullfunktion. Diese ist Turing-berechenbar.

   Wäre cf für $c \geq 1$ Turing-berechenbar, dann könnte man mittels

   $$f(n) = \mathrm{div}(cf(n), c)$$

   eine DTM konstruieren, die f berechnet, was nicht möglich ist.

   Bemerkung: Eine solche Funktion f existiert, da es nur abzählbar viele Turingmaschine, aber überabzählbar viele Funktionen über natürliche Zahlen gibt.

Aufgabe 4

Wir erweitern LOOP-Programme um Anweisungen `HALT` und `ERROR`, die die Ausführung des Programms sofort abbrechen. Bei `HALT` ist der Rückgabewert der aktuelle Wert der Variable x_0. Bei `ERROR` ist das Resultat der Berechnung undefiniert.

1. Gibt es für jedes LOOP-Programm mit `HALT`-Anweisungen ein äquivalentes LOOP-Programm ohne `HALT`-Anweisungen?
2. Gibt es für jedes LOOP-Programm mit `ERROR`-Anweisungen ein äquivalentes LOOP-Programm ohne `ERROR`-Anweisungen?

Lösung

1. Ja.

   Sei P ein LOOP-Programm mit `HALT`-Anweisungen. Wir geben zwei mögliche Konstruktionen an, wie aus P ein äquivalentes LOOP-Programm ohne `HALT`-Anweisungen konstruiert werden kann.

   Konstruktion 1

   - Wähle eine in P nicht verwendete Variable x_n und setze zu Beginn der Ausführung ihren Wert mittels $x_n := x_n + 1$ auf 1.
   - Ersetze jede Anweisung der Form $x_i := x_j \pm c$ durch

     $$\texttt{LOOP}\ x_n\ \texttt{DO}\ x_i := x_j \pm c\ \texttt{END}.$$

   - Ersetze jede `HALT`-Anweisung durch $x_n := x_n - 1$.

   Konstruktion 2

- Wähle zwei in P nicht verwendete Variablen x_m und x_n und setze zu Beginn der Ausführung den Wert von x_n mittels $x_n := x_n + 1$ auf 1.
- Ersetze jede `HALT`-Anweisung durch

$$\texttt{LOOP } x_n \texttt{ DO } x_m := x_0 + 0\texttt{; } x_n := x_n - 1 \texttt{ END}.$$

- Füge am Ende des Programms die Anweisungen

$$\texttt{LOOP } x_n \texttt{ DO } x_m := x_0 + 0 \texttt{ END; } x_0 := x_m + 0$$

  hinzu.

2. Nein.

   Das Programm $P = \texttt{ERROR}$ berechnet die nirgends definierte Funktion $\Omega\colon \mathbb{N} \to \mathbb{N}$. Da diese nicht total ist, kann sie nicht von einem LOOP-Programm berechnet werden.

Aufgabe 5

Wir erweitern GOTO-Programme um `GOTOV`-Anweisungen. Beim Erreichen einer Anweisung der Form `GOTOV` x_i soll zur Marke M_{x_i} gesprungen werden. Falls die adressierte Marke nicht existiert, soll das Resultat der Berechnung undefiniert sein.

Gibt es für jedes GOTO-Programm mit `GOTOV`-Anweisungen ein äquivalentes GOTO-Programm ohne `GOTOV`-Anweisungen?

Lösung

Ja.

Sei $P = M_1\texttt{:}A_1\texttt{;}\ldots\texttt{;}M_n\texttt{:}A_n$ ein GOTO-Programm mit `GOTOV`-Anweisungen. Mit der folgenden Konstruktion kann aus P ein äquivalentes GOTO-Programm ohne `GOTOV`-Anweisungen konstruiert werden:

- Ersetze jede Anweisung der Form `GOTOV` x_i durch

$$\texttt{IF } x_i = 1 \texttt{ THEN GOTO } M_1\texttt{;}\ldots\texttt{;IF } x_i = n \texttt{ THEN GOTO } M_n\texttt{;GOTO } M_{n+1}.$$

- Füge am Ende des Programms $M_{n+1}\texttt{:GOTO } M_{n+1}$ hinzu.

Hinweis: Wie üblich bei GOTO-Programmen wurden hier überflüssige Markierungen weggelassen.

Aufgabe 6

Welche der folgenden Aussagen gelten für beliebige Funktionen $f, g\colon \mathbb{N} \to \mathbb{N}$?

1. Wenn f und g Turing-berechenbar sind, dann ist auch $f \circ g$ Turing-berechenbar.
2. Wenn f und $f \circ g$ Turing-berechenbar sind, dann ist auch g Turing-berechenbar.
3. Wenn g und $f \circ g$ Turing-berechenbar sind, dann ist auch f Turing-berechenbar.

Erinnerung: $f \circ g\colon \mathbb{N} \to \mathbb{N}, n \mapsto f(g(n))$ ist die *Komposition* von f und g.

Lösung

1. Die Aussage ist wahr.

   Seien f und g Turing-berechenbar und seien M_f und M_g DTMs, sodass f von M_f und g von M_g berechnet werden. Dann wird $f \circ g$ von $M_g; M_f$ berechnet (siehe Folie 6.7).

2. Die Aussage ist falsch.

   Wählt man f als die konstante Nullfunktion und g beliebig nicht Turing-berechenbar, dann ist $f \circ g = f$. Es gibt also Funktionen $f, g \colon \mathbb{N} \to \mathbb{N}$, sodass f und $f \circ g$ Turing-berechenbar sind, aber g nicht.

3. Die Aussage ist falsch.

   Wählt man g als die konstante Nullfunktion und f beliebig nicht Turing-berechenbar, dann gilt: $f \circ g \colon \mathbb{N} \to \mathbb{N}, n \mapsto f(0)$, d. h. $f \circ g$ ist konstant und somit Turing-berechenbar. Es gibt also Funktionen $f, g \colon \mathbb{N} \to \mathbb{N}$, sodass g und $f \circ g$ Turing-berechenbar sind, aber f nicht.