Ergänzungsblatt 2

Aufgabe 1

Geben Sie eine DTM M an, die die Nachfolgerfunktion

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

berechnet.

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.

Aufgabe 3

Wir betrachten die ganzzahlige Division

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

wobei $\operatorname{div}(m,n)$ 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?

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?

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?

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.