1. Wie wir in der Vorlesung gesehen haben ist die Menge der Funktionen $\mathbb{N} \to \mathbb{N}$ überabzählbar.

   (a) Ändern Sie das dortige Argument so ab, daß es zeigt, daß sogar schon die Menge aller Funktionen $\mathbb{N} \to \{0,1\}$ überabzählbar ist.

   (b) Zeigen Sie, daß die Menge aller Teilmengen von $\mathbb{N}$ überabzählbar ist. Nehmen Sie hierfür an, daß es eine Aufzählung $A_1, A_2, \ldots$ aller Teilmengen von $\mathbb{N}$ gibt und betrachten $B = \{n \mid n \notin A_n\}$

   (c) Was haben die beiden oberen Teilaufgaben gemeinsam?

   Lösung

   (a) Sei das Setup wie in den Folien, aber g definiert als

   $$g(n) = \begin{cases} 0 & \text{falls } f_n(n) = 1 \\ 1 & \text{falls } f_n(n) = 0 \ . \end{cases}$$

   Dann ist $g : \mathbb{N} \to \{0,1\}$, also vom Typ her korrekt. Der Rest funktioniert wie bisher: Nehmen wir an es gibt ein k so daß $f_k \equiv g$, so ist

   $$f_k(k) = g(k) \ .$$

   Aber nach Definition von g ist entweder die linke Seite 0 und die rechte Seite 1 oder anderstherum; also ein Widerspruch.

   (b) Sei B wie oben, und nehmen wir an es ist auf der Liste, also es gibt k mit $A_k = B$. Dann gilt:

   $$k \in B \iff k \in A_k \iff k \notin B \ ;$$

   also auch hier ein Widerspruch.

   (c) Die Ergebnisse lassen sich leicht ineinander überführen, bzw. aus dem jeweils anderen erlangen, da $\mathcal{P}(\mathbb{N})$ und $\{f : \mathbb{N} \to \{0,1\}\}$ gleichmächtig sind: die Abbildung die jeder Menge $A \subseteq \mathbb{N}$ ihre charakteristische Funktion zuweist ist eine Bijektion.

2. Geben Sie eine Turing-Maschine an, die ein Eingabewort w verdoppelt; also ww als Ausgabe hat. (D.h. realisieren Sie die Idee zu M_3 aus der Vorlesung.)

   Lösung: Wie in der Vorlesung angesprochen konstruieren wir eine TM, die in drei Phasen arbeitet. Sei Σ das Eingabealphabet und $\hat{\Sigma} = \{\hat{x} \mid x \in \Sigma\}$ eine Kopie davon mit Markierungen und $\Gamma = \Sigma \cup \hat{\Sigma} \cup \{\#, B\}$.

   $$M_3 = (\{s, p_1, \ldots, p_4, r_B, m, h\} \cup \{q_x, r_x \mid x \in \Sigma\}, \Sigma, \Gamma, s, \{h\}, \delta)$$

   (a) $w \rightsquigarrow w\#$.

   $$\begin{aligned} \delta(s, x) &= (s, x, R) \quad \text{für } x \in \Sigma \\ \delta(s, B) &= (p_1, \#, L) \\ \delta(p_1, x) &= (p_1, x, L) \quad \text{für } x \in \Sigma \\ \delta(p_1, B) &= (q, B, R) \quad \text{für } x \in \Sigma \end{aligned}$$

D.h. wir bewegen uns ans rechte Ende des Wortes (im Startzustand s), schreiben ein # und bewegen uns zurück an den Anfang.

(b) In der nächsten Phase markieren wir das erste Zeichen x, merken es uns über den passenden Zustand q_x und schreiben es ans Ende des Wortes. Danach gehen wir bis zum ersten nicht markierten Zeichen und wiederholen die Prozedur. Finden wir kein unmarkiertes Zeichen links von # entfernen wir alle Markierungen und bewegen uns ans Ende des Wortes

$$\begin{gathered}
\delta(q, x) = (q_x, \hat{x}, R) \quad \text{für } x \in \Sigma \\
\delta(q_x, y) = (q_x, y, R) \quad \text{für } y \in \Sigma \cup \{\#\} \\
\delta(q_x, B) = (p_2, x, L) \\
\delta(p_2, y) = (p_2, y, L) \quad \text{für } y \in \Sigma \cup \{\#\} \\
\delta(p_2, y) = (q, y, R) \quad \text{für } y \in \hat{\Sigma} \\
\delta(q, \#) = (p_3, \#, L) \\
\delta(p_3, \hat{x}) = (p_3, x, L) \quad \text{für } x \in \Sigma \\
\delta(p_3, B) = (p_4, B, R)
\end{gathered}$$

(c) In der letzten Phase bewegen wir uns ans rechte Ende des Wortes und fangen an jeden Buchstaben um eins nach links zu bewegen, bis wir das # überschrieben haben

$$\begin{gathered}
\delta(p_4, x) = (p_4, x, R) \quad \text{für } x \in \Sigma \cup \{\#\} \\
\delta(p_4, B) = (r_B, B, L) \\
\delta(r_x, y) = (r_y, x, L) \quad \text{für } x \in \Sigma \cup \{B\} \text{ und } y \in \Sigma \\
\delta(r_x, \#) = (m, x, L) \quad \text{für } x \in \Sigma \\
\delta(m, x) = (m, x, L) \quad \text{für } x \in \Sigma \\
\delta(m, B) = (h, B, R)
\end{gathered}$$

3. Konstruieren Sie „die“ Turingmaschine $T_{bin?}$ aus der Vorlesung.

Lösung: Eine passende Maschine ist jetzt:

$M_{bin?} = (\{q_1, \ldots, q_k, p_1, \ldots, p_k, r_1, \ldots, r_k, m, h\}, \{0, 1, \#\}, \{0, 1, \#, B\}, q_1, \{h\}, \delta)$

wobei δ gegeben ist durch

$$\begin{gathered}
\delta(q_i, 1) = (p_i, 1, R) \quad i = 1 \ldots k \\
\delta(q_i, 0) = (r_i, 0, R) \quad i = 1 \ldots k-1 \\
\delta(r_i, \#) = (q_{i+1}, \#, R) \quad \text{für } i = 1 \ldots k-1 \\
\delta(p_i, x) = (p_i, x, R) \quad \text{für } x = 0, 1 \text{ und } i = 1 \ldots k \\
\delta(p_i, \#) = (q_{i+1}, \#, R) \quad \text{für } i = 1 \ldots k-1 \\
\delta(r_k, B) = (m, B, L) \\
\delta(p_k, B) = (m, B, L) \\
\delta(m, x) = (m, x, L) \quad \text{für } x = 0, 1, \# \\
\delta(m, B) = (h, B, R)
\end{gathered}$$

(Man beachte, daß die Maschine nie ein Zeichen überschreibt!) Die Idee ist, daß wir im Index des Zustandes festhalten wie viele Eingaben wir eigentlich erwarten. Ist das erste Zeichen einer Binärzahl 1 gehen wir in den Zustand p_i, in dem wir einfach das nächste # suchen. Lesen wir eine 0 gehen wir in den Zustand r_i, welcher auf alle Fälle ein # als nächstes Symbol erwartet. Haben wir ein # gefunden gehen wir in den Zustand q_{i+1} und wiederholen die Prozedur. In den Zuständen r_k und p_k suchen wir natürlich nach dem Blanksymbol am rechten Ende des Wortes. Zum Schluss bewegen wir uns wie gewohnt in die Ausgangsposition zurück.

4. Konstruieren Sie eine Turingmaschine, die die Funktion $f_- : \mathbb{N} \to \mathbb{N}$ definiert durch $f_-(n) = \max\{0, n-1\}$ berechnet.

   Lösung: Wie in der Vorlesung erläutert können wir, dank $T_{bin?}$ annehmen, daß das Eingabewort wirklich eine Binärzahl ist.
   Eine passende Maschine ist jetzt:

   $$M = (\{s, q, p, k, m, h\}, \{0, 1\}, \{0, 1, B\}, s, \{h\}, \delta)$$

   wobei δ gegeben ist durch

   $$\begin{aligned}
   \delta(s, 0) &= (s, 0, R) \\
   \delta(s, 1) &= (s, 1, R) \\
   \delta(s, B) &= (q, B, L) \\
   \delta(q, 1) &= (p, 0, L) \\
   \delta(q, 0) &= (q, 1, L) \\
   \delta(q, B) &= (k, B, R) \\
   \delta(k, 1) &= (h, 0, N) \\
   \delta(p, 0) &= (p, 0, L) \\
   \delta(p, 1) &= (p, 1, L) \\
   \delta(p, B) &= (m, B, R) \\
   \delta(m, 0) &= (h, B, R) \\
   \delta(m, 1) &= (h, 1, N)
   \end{aligned}$$

   Zur Erklärung: Wie so oft bewegen wir uns im Zustand s ans rechte Ende des Wortes und gehen in den Zustand q. Finden wir eine 1, so ersetzen wir sie durch eine 0, ansonsten ersetzen wir 0-en durch 1-en, bis wir eine 1 finden (was wir immer werden, es sei den die Eingabe war 0). Danach gehen wir in den Zustand p, in dem wir ans linke Ende des Wortes gehen. Dort müssen wir eventuell noch eine 0 am Anfang des Wortes löschen.
   Ist die Eingabe 0 gewesen, und nur dann, finden wir im Zustand q ein B. Ausserdem haben wir in diesem Fall dummerweise die 0 mit einer 1 überschrieben. Also gehen wir in den Zustand k und berichtigen dies, bevor wir in den Endzustand h gehen.

5. In der Vorlesung haben wir den folgenden Satz kennengelernt:

Satz. *Seien für* $i = 1, 2$ $T_i = (Q_i, \Sigma, \Gamma, B, s_i, F_i, \delta_i)$.

Des Weiteren nehmen wir an, daß jede akzeptierende Berechnung von T_1 *in einer Konfiguration der Form* $(\alpha, q, a_1 a_2 \dots a_n \beta)$ *endet, wobei* $\alpha, \beta \in B^*$ *und* $a_1, \dots, a_n \in \Sigma$.

Dann gilt

$$[\![T_1; T_2]\!] = [\![T_2]\!] \circ [\![T_1]\!] \ .$$

Zeigen Sie, daß die umrandete Bedingung notwendig ist.

Lösung: Sicherlich lässt sich auch ein realistischeres Beispiel finden. Allerdings lässt folgende Konstruktion das Problem gut erkennen: Sei $T_1 = (\{s, p, h\}, \{0, 1\}, \{0, 1, B\}, B, s, \{h\}, \delta_1)$ mit δ definiert durch

$$\begin{aligned} \delta(s, x) &= (p, x, L) \quad \text{für } x = 0, 1 \\ \delta(p, B) &= (h, 0, R) \end{aligned}$$

D.h. Die Maschine T_1 macht nichts als eine 0 links vom Eingabewort hinzuschreiben. Da

$$\text{init}_{T_1}(w) = (\varepsilon, s, w) \vdash_{T_1} (\varepsilon, p, Bw) \vdash_{T_1} (0, h, w)$$

ist $[\![T_1]\!] = \text{id}$.

Sei jetzt $T_2 = (\{q, m, f\}, \{0, 1\}, \{0, 1, B\}, q, \{f\}, \delta_2)$ wobei δ definiert durch

$$\begin{aligned} \delta(q, x) &= (m, x, L) \quad \text{für } x = 0, 1 \\ \delta(m, B) &= (f, B, R) \end{aligned}$$

Auch diese Maschine berechnet, wie man leicht sehen kann, die Identität. Startet T_2 allerdings nicht in der Startkonfiguration sondern steht in der Zelle links der Startposition ein Symbol, welches nicht das Blanksymbol ist, verwirft die Rechnung.

Zusammen haben wir egal für welches Eingabewort w:

$$[\![T_1; T_2]\!](w) = \bot$$

aber

$$([\![T_2]\!] \circ [\![T_1]\!])(w) = (\text{id} \circ \text{id})(w) = w \ .$$