Gruppe **17**
Berechenbarkeit & Komplexität

Philipp Hochmann, 356148
Felix Kiunke, 357322

Blatt 2

Aufgabe 2.1

$$\langle M \rangle = 111 \text{ code}(1) \text{ } 11 \text{ code}(2) \text{ } 11 \text{ code}(3) \text{ } 11 \text{ code}(4) \text{ } 11 \text{ code}(5) \text{ } 11 \text{ code}(6) \text{ } 111$$

$$\begin{aligned} \text{wobei code}(1) &= 0101000100100, \\ \text{code}(2) &= 010010101000, \\ \text{code}(3) &= 01000100100010, \\ \text{code}(4) &= 00010101010, \\ \text{code}(5) &= 000100100010010, \\ \text{code}(6) &= 00010001010001000 \end{aligned}$$

Aufgabe 2.2

Angenommen, $\mathcal{M}$ hält. Das heißt, das während der Ausführung keine Konfiguration doppelt angenommen werden kann. Würde eine Konfiguration doppelt vorkommen, würde $\mathcal{M}$ nicht halten (Analog Pumping Lemma). Insbesondere wird die Anfangskonfiguration nicht wieder angenommen. Es gilt also, die Anzahl der Konfigurationen zu berechnen, um eine obere Schranke der Schritte, die $\mathcal{M}$ macht, angeben zu können.

Die Anzahl der Schritte kann genau mit der Formel

$$(|Q| - 1) \cdot |\Gamma|^{s(n)} \cdot s(n) + 1$$

berechnet werden.

Anzahl der Zustände ohne Endzustand $\times$ Anzahl der Bandbelegung $\times$ Anzahl der Lesekopfpositionen + 1 (für den Schritt in den Endzustand)

Aufgabe 2.3

Sei $L = \{w\#w \mid w \in \{0,1\}^*\}$ über $\Sigma = \{0, 1, \#\}$.

(a)

Wir konstruieren die Turingmaschine $M = (Q, \Sigma, \Gamma, B, q_0, \bar{q}, \delta)$ mit

$$\begin{aligned} \Sigma &= \{0, 1, \#\}, \\ \Gamma &= \{0, 1, \#, B\}. \end{aligned}$$

Die Turingmaschine operiert folgendermaßen (High-Level-Beschreibung folgt weiter unten):

1. Lies und merke (im Zustand) das erste Zeichen a und ersetze es dann durch B
2. Gehe nach rechts, bis ein # oder ein B erreicht wird. Wird zuerst ein B erreicht, schreibe 0 und gehe in den Endzustand $\bar{q}$ über
3. Gehe weiter nach rechts, bis zum ersten Zeichen, das kein # ist
4. Wenn das Zeichen dem Zeichen a nicht entspricht, schreibe 0 und gehe in den Endzustand $\bar{q}$ über
5. Wenn das Zeichen ein b ist und a ein # ist, schreibe 1 und gehe in den Endzustand $\bar{q}$ über
6. Ansonsten schreibe # und gehe einen Schritt nach rechts
7. Lies und merke (im Zustand) das Zeichen b und ersetze es durch #
8. Gehe nach links bis zum ersten B und dann einen Schritt nach rechts
9. Wenn das Zeichen dem gemerkten Zeichen b nicht entspricht, schreibe 0 und gehe in den Endzustand $\bar{q}$ über
10. Ansonsten schreibe B und gehe einen Schritt nach rechts.
11. Fahre fort mit Schritt 1

High-Level-Beschreibung: Die Maschine merkt sich immer ein Zeichen in Form verschiedener Zustände und vergleicht es dann mit dem entsprechenden Zeichen auf der anderen Seite des #. Jedes gelesene Zeichen wird links durch ein B und rechts durch ein # ersetzt. Das erleichtert es, einen möglichst kurzen Weg zurückzulegen. Außerdem wird, bevor der „Rückweg“ gegangen wird, das jeweils nächste Zeichen (einen Schritt weiter rechts) eingelesen, um weitere Strecken zu vermeiden. Wenn die Zeichen nicht übereinstimmen oder ein sonstiger ungültiger Fall (z.B. kein # im Wort, unterschiedliche Längen des linken und rechten Teils, ...) auftritt, wird sofort 0 geschrieben und abgebrochen, also das Wort abgelehnt.

Analyse: Da die Turingmaschine nur auf dem Eingabewort operiert und maximal eine Zelle rechts davon verwendet, ist der Speicherbedarf $\mathcal{O}(n)$ mit $n =$ Länge des Eingabewortes.

Um die Buchstaben links und rechts des # zu vergleichen, bewegt sich der Lesekopf alle zwei Zeichen um $\lceil \frac{n}{2} \rceil$. Da die konstanten Faktoren von der $\mathcal{O}$-Notation „verschluckt“ werden, diese Bewegung des Lesekopfs aber proportional oft zur Zeichenanzahl ausgeführt wird, ist der Zeitbedarf $\mathcal{O}(n^2)$.

(b)

Wir konstruieren eine 2-Band-Turingmaschine, die folgendermaßen funktioniert:

1. Lies die Zeichen der Eingabe von links nach rechts ein, bis entweder # oder B gelesen wird. Schreibe dabei jedes gelesene Zeichen (außer dem zuletzt gelesenen) von links nach rechts auf das zweite Band
2. Wenn ein B erreicht wurde, schreibe 0 aufs erste Band und gehe in den Endzustand über

3. Ansonsten gehe auf dem ersten Band einen Schritt nach rechts. Wenn dort ein # steht, ist die Eingabe ungültig. Schreibe dann 0 und gehe in den Endzustand über

4. Gehe auf dem zweiten Band nach links bis zum ersten B und gehe dann wieder einen Schritt nach rechts

5. Vergleiche die Zeichen auf beiden Bändern. Unterscheiden sie sich, schreibe sofort 0 aufs erste Band und gehe in den Endzustand über. Sind beide Zeichen B, schreibe 1 aufs erste Band und gehe in den Endzustand über (dann ist das Wort in L). Gehe einen Schritt nach rechts und wiederhole dies bis ein Endzustand erreicht wird

Analyse: Die 2-Band-Maschine braucht nur den Platz, den das Eingabewort braucht, und auf dem zweiten Band zusätzlich maximal ebensoviel Platz (wenn das Wort kein # enthält). Der Speicherbedarf ist also $2n$ bzw. $\mathcal{O}(n)$ mit $n =$ Länge des Eingabewortes.

Der Zeitbedarf für Schritt 1-3 ist im Worst Case $\mathcal{O}(n)$. Das Zurückgehen in Schritt 4 ist ebenfalls $\mathcal{O}(n)$. Gleiches gilt für den Vergleich in Schritt 5. Damit ist der Gesamtzeitbedarf $\mathcal{O}(n)$.

Aufgabe 2.4

An der 0. Position des Bandes, also ganz links vom Eingabewort, steht ein Sentinel-Element, welches den Zugriff auf das Band regelt, es sozuagen nach links absperrt. Man simuliert die linke Seite nun über das kartesische Produkt (die resultierende TM ist eine Zweispurmaschine): Jede Zelle des Bandes rechts vom Sentinel-Element enthalte ein Tupel. Links davon sind nur Blanks. Der erste Eintrag ist das Element auf dem rechten Teil des Bandes, der zweite das Element, was spiegelbildlich auf der linken Seite stehen würde. Die Turingmaschine speichert über zwei Zustände, ob gerade auf dem ersten oder zweiten Element des Tupels operiert wird. Wird der Lesekopf auf das Sentinel-Element geschoben, wird er direkt wieder nach rechts auf das erste richtige Symbol geschoben. Dabei wird in den jeweils anderen Zustand übergegangen, da nun die jeweils mit den anderen Elementen der Tupel gearbeitet werden muss. Dadurch wird ein Übergang auf die andere Bandhälfte simuliert.

Dabei entsteht kein Zeitverlust.