Algorithmen und Berechnungskomplexität II SS 16

Universität Bonn, Institut für Informatik, Abteilung I

6. Aufgabenblatt zur Vorlesung **Abgabe: Mi. 08.06.** (09^{00})

Aufgabe 24: Turingmaschine konstruieren (4 Punkte)

Für $n \geq 1$ sei $\Sigma = \{a_1, a_2, \ldots a_n\}$ ein beliebiges Alphabet. Konstruieren Sie eine deterministische 1-Band Turingmaschine M die, gestartet auf leerem Band, Σ auf dem Band ausgibt. Das heißt nach endlich vielen Schritten hält M, der Kopf zeigt auf das erste Bandquadrat und der Bandinhalt ist $\$a_1\#a_2\#\ldots\#a_n$. Ihre Turingmaschine darf maximal 5 Zustände besitzen.

Aufgabe 25: DFA vs DTM (4 Punkte)

Zeigen Sie: Zu jedem deterministischen, endlichen Automaten A gibt es eine deterministische Turingmaschine T, die das Verhalten von A simuliert.

Aufgabe 26: Fleißige Biber (4 Punkte)

In dieser Aufgabe betrachten wir spezielle Turingmaschinen, welche ein beidseitig unbeschränktes Band verwenden. Das (Band)-Alphabet besteht nur aus $\{0, 1\}$. Es gibt lediglich einen Endzustand. Der Kopf muss in jedem Schritt bewegt werden. Zu Beginn ist jedes Bandquadrat mit 0 initialisiert, der Kopf zeigt auf ein beliebiges Bandquadrat. Wie viele solcher Turingmaschinen gibt es, wenn n die Anzahl der Zustände bezeichnet? Begründen Sie Ihre Antwort.

Ein fleißiger Biber B mit n Zuständen ist eine solche Turingmaschine, die darüber hinaus folgende Eigenschaften besitzt.

1. B hält nach endlich vielen Schritten.
2. Keine andere derartige, nach endlich vielen Schritten haltende Turingmaschine mit n Zuständen, schreibt mehr Einsen auf das Band als B.

Finden Sie einen fleißigen Biber für $n = 2$ und geben Sie die Überführungsfunktion als Diagramm an. Notieren Sie die Konfigurationsfolge.

Hinweis: Ein fleißiger Biber für $n = 2$ schreibt genau 4 Einsen auf das Band.

Aufgabe 27: GOTO-Programme (Präsenzaufgabe)

Betrachten Sie die (eingeschränkte) Programmiersprache **GOTO**. Diese erlaubt lediglich die Verwendung von Additionen, Wertzuweisungen und Sprungmarken. Ein GOTO-Programm besteht aus Variablen $x_1, x_2, \ldots$, Konstanten $c \in \mathbb{N}_0$, Trennsymbolen ; und :=, Operatoren $+, -$ und den Schlüsselwörtern IF , THEN GOTO und HALT.

Notiert wird ein GOTO-Programm als endliche Folge von Anweisungen A_i und Sprungmarken M_i. Für Anweisungen gibt es folgende Möglichkeiten:

$$\begin{array}{lcl} M_1 & : & A_1; \\ M_2 & : & A_2; \\ \vdots & & \\ M_k & : & A_k; \end{array}$$

1. **Wertzuweisung** von Variablen und Konstanten, d.h.
   - $x_i := c$
   - $x_i := x_j$
   - $x_i := x_j + c$
   - $x_i := x_j - c$

   sind gültige Anweisungen.
2. **Bedingter Sprung**, d.h. "IF $x_i = c$ THEN GOTO M_j" ist eine gültige Anweisung.
3. **Stopp**, d.h. HALT ist eine gültige Anweisung.

Ausgeführt wird das Programm beginnend mit der Anweisung A_1. Nachdem die Anweisung A_i ausgeführt wurde wird stets mit der Anweisung A_{i+1} weitergemacht, sofern A_i nicht durch einen bedingten Sprung die nächste durchzuführende Anweisung bestimmt. Das Programm terminiert sofort, wenn die Stopp-Anweisung ausgeführt wird. Eingaben stehen in den ersten i Variablen, die Ausgabe in x_1.

Eine Funktion f heißt *GOTO-berechenbar*, wenn ein GOTO-Programm formuliert werden kann, welches f berechnet. Mit $\mathcal{F}_{\text{GOTO}}$ bezeichnen wir die Klasse/Menge der GOTO-berechenbaren Funktionen. Zeigen Sie, dass jede WHILE-berechenbare Funktion auch GOTO-berechenbar ist, d.h. es gilt $\mathcal{F}_{\text{WHILE}} \subseteq \mathcal{F}_{\text{GOTO}}$.