IWI Fakultät für Informatik
und Wirtschaftsinformatik
Lehrgebiet für Grundlagen der Informatik
Prof. Dr. Heiko Körner

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES
Näher dran.

3. Übung zur Vorlesung Theoretische Informatik II

Musterlösungen

Aufgabe 1: Sei $M = (\{z_0, \ldots, z_5\}, \{0,1\}, \{0,1,\square\}, \delta, z_0, \square, \{z_5\})$ eine DTM mit folgender Turingtafel:

	0	1	$\square$
z_0	$(z_5, 0, N)$	$(z_1, 1, R)$	–
z_1	$(z_1, 0, R)$	$(z_1, 1, R)$	$(z_2, \square, L)$
z_2	$(z_2, 1, L)$	$(z_3, 0, L)$	–
z_3	$(z_3, 0, L)$	$(z_3, 1, L)$	$(z_4, \square, R)$
z_4	$(z_4, \square, R)$	$(z_5, 1, N)$	$(z_5, 0, N)$
z_5	–	–	–

Zunächst testet M im Zustand z_0, ob als Eingabe die Zahl 0 vorliegt. In diesem Fall lässt M das Band unverändert und stoppt sofort wieder. Ansonsten sucht M in den Zustand z_1 nach dem rechten Ende der Zahl, ohne sie dabei zu verändern. Anschließend beginnt in z_2 der Vorgang der binären Subtraktion. Im Fall eines nicht gesetzten Bits muss dieses durch eine Eins ersetzt werden, und aufgrund des Übertrags stellt sich die gleiche Aufgabe ein Bit weiter links erneut. Wenn jedoch eine Eins gefunden wird, so muss diese nur noch durch einen Null ersetzt werden, und M wechselt nach z_3, um den Kopf wieder nach links zu fahren. Nun kann es sein, dass bei dem bislang skizzierten Verfahren eine führende Null entsteht (wenn man z.B. eine Eins von $4 = 100_2$ subtrahiert). Deshalb wird nach dem Finden des höchsten Bits eine evtl. führende Null in z_4 gelöscht. Allerdings kann dabei (wenn die Eingabe 1 war), „aus Versehen“ das komplette Ergebnis (also die einzelne Null) überschrieben werden. Auch dies wird in z_4 bei Auffinden eines Blanks entsprechend korrigiert. Abschließend wechselt M in den Endzustand z_5, wobei der Kopf wie üblich unter dem ersten Zeichen der Ausgabe stehen bleibt.

Aufgabe 2: Die folgenden TMs sind alle von der Form $M = (Z, \{0,1\}, \{0,1,\square\}, \delta, z_0, \square, E)$, wobei die Zustandmengen Z aus den Turingtafeln ersichtlich sind. Der jeweils letzte Zustand mit dem höchsten Index ist immer der Endzustand.

a) In der binären Zahlendarstellung lässt sich eine Multiplikation mit 2 durch das Anhängen einer Null an die Kodierung von n realisieren. Nur im Fall $n = 0$ muss die Kodierung unverändert bleiben. Mit der nachstehenden Turingtafel erfüllt deshalb M offenbar das Gewünschte.

	0	1	$\square$
z_0	$(z_3, 0, N)$	$(z_1, 1, R)$	–
z_1	$(z_1, 0, R)$	$(z_1, 1, R)$	$(z_2, 0, L)$
z_2	$(z_2, 0, L)$	$(z_2, 1, L)$	$(z_3, \square, R)$
z_3	–	–	–

Im Zustand z_1 fährt sie mit dem Kopf nach rechts und hängt die Null an, im Zustand z_2 fährt sie mit dem Kopf zurück, und im Zustand z_3 (dies sei der einzige Endzustand) akzeptiert sie dann. Der Zustand z_0 dient für die Sonderabfrage des Falls $n = 0$.

b) Im Fall der ganzzahlige Division durch 4 verfolgt M eine ähnliche Strategie. Diesmal müssen jedoch die letzten beiden binären Stellen von n gestrichen werden.

	0	1	$\square$
z_0	$(z_0, 0, R)$	$(z_0, 1, R)$	$(z_1, \square, L)$
z_1	$(z_2, \square, L)$	$(z_2, \square, L)$	–
z_2	$(z_3, \square, L)$	$(z_3, \square, L)$	$(z_5, 0, N)$
z_3	$(z_4, 0, L)$	$(z_4, 1, L)$	$(z_5, 0, N)$
z_4	$(z_4, 0, L)$	$(z_4, 1, L)$	$(z_5, \square, R)$
z_5	–	–	–

Im Startzustand z_0 fährt M mit dem Kopf nach rechts, im Zustand z_1 löscht sie die erste Ziffer, und im Zustand z_2 die zweite. Falls beim Streichen der beiden Ziffern nur noch das leere Band zurückbleibt, ist das Ergebnis 0, so dass M dann eine einzelne Null erzeugen muss. Dies wird in den Zuständen z_2 und z_3 sichergestellt. Im Zustand z_4 fährt der Kopf dann wieder nach links zurück, und der Endzustand z_5 akzeptiert.

c) Der ganzzahlige Zweierlogarithmus einer binär kodierten Zahl ist leichter zu berechnen als es zunächst vielleicht den Anschein hat. Es handelt sich hierbei nämlich lediglich um die um Eins verminderte Anzahl der Binärziffern von n. Mit Hilfe der nachstehenden Turingtafel erzeugt M links von der Eingabe n die Ausgabe $\lfloor \log_2 n \rfloor$, indem sie Ziffer für Ziffer der Kodierung von n streicht und dafür jedesmal binär eine Eins auf das Ergebnis aufaddiert. Um die Ein– und Ausgabe voneinander zu trennen, verwendet M dazwischen ein Trennzeichen #.

	0	1	#	$\square$
z_0	$(z_0, 0, N)$	$(z_1, \#, R)$	–	–
z_1	$(z_1, 0, R)$	$(z_1, 1, R)$	–	$(z_2, \square, L)$
z_2	$(z_3, \square, L)$	$(z_3, \square, L)$	$(z_6, \square, L)$	–
z_3	$(z_3, 0, L)$	$(z_3, 1, L)$	$(z_4, \#, L)$	–
z_4	$(z_5, 1, R)$	$(z_4, 0, L)$	–	$(z_5, 1, R)$
z_5	$(z_5, 0, R)$	$(z_5, 1, R)$	$(z_1, \#, R)$	–
z_6	$(z_7, 0, L)$	$(z_7, 1, L)$	–	$(z_8, 0, N)$
z_7	$(z_7, 0, L)$	$(z_7, 1, L)$	–	$(z_8, \square, R)$
z_8	–	–	–	–

Beachten Sie, dass der Logarithmus für $n = 0$ nicht definiert ist. In diesem Fall bleibt M im Zustand z_0 „hängen“, ohne jemals zu stoppen. Andernfalls überschreibt M in z_0 die führende Eins mit dem Trennzeichen (dadurch nimmt gleichzeitig die Anzahl der Ziffern von n um Eins ab) und bewegt sich mit dem Kopf nach rechts. Im Zustand z_1 sucht M das rechte Ende der noch vorhandenen Ziffern von n und löscht im Zustand z_2 eine weitere davon. Im nächsten Zustand z_3 bewegt M den Kopf wieder nach links bis über das Trennzeichen hinweg, d.h. M steht dann am rechten Ende der Ergebniszahl (bzw. unter einem Blank, falls überhaupt noch nichts beim Ergebnis geschrieben wurde). Der Zustand z_4 erledigt die binäre Addition von einer Eins. Im Zustand z_5 wird der Kopf

wieder nach rechts bis über das Trennzeichen bewegt, so dass M im Zustand z_1 mit der nächsten Iteration beginnen kann.

Beim Löschen der jeweils nächsten Ziffer im Zustand z_2 wird gleichzeitig abgefragt, ob überhaupt noch eine solche Ziffer vorhanden ist. Falls nicht, wird das Trennzeichen entfernt, und M bewegt in den Zuständen z_6 und z_7 ihren Kopf unter die führende Binärziffer des Ergebnisses. Gleichzeitig stellt z_6 sicher, dass eine einzelne Null auf das Band gedruckt wird, falls noch keine Ergebniszahl erzeugt wurde (dies kann allerdings nur im Fall $n = 1$ passieren). Beim einzigen Endzustand z_8 endet dann die Berechnung.

Aufgabe 3: Eine 2–Band DTM $M = (\{z_0, z_1, z_2, z_3, z_4\}, \{a, b\}, \{a, b, \square\}, \delta, z_0, \square, \{z_4\})$ kann z.B. auf folgender Idee aufbauen. Zuerst wird die Eingabe von links nach rechts auf das zweite Band kopiert. Anschließend wird der zweite Kopf zurück nach links gefahren, während der erste Kopf hinter dem rechtesten Symbol der Eingabe verharrt. Anschließend wird das Wort nochmals von links nach rechts kopiert (diesmal von dem zweiten auf das erste Band), wobei gleichzeitig die Kopie wieder gelöscht wird. Danach muss lediglich der Schreib–Lese–Kopf des ersten Bandes nochmals nach links gefahren werden (die Position des zweiten Kopfes ist egal).

a) Die folgende Turingtafel realisiert diese Idee. Dabei bedeutet eine Spaltenbeschriftung wie $a, \square$, dass vom ersten Band (mit der Ein– und Ausgabe) ein a und vom zweiten Band ein Blank gelesen wurde. Für alle fehlenden Kombinationen (z.B. a, b) ist δ unabhängig vom Zustand nicht definiert.

   Aus Platzgründen wurde bei den Tabelleneinträgen auf die öffnenden und schließenden Klammern verzichtet, d.h. „z_0, a, a, R, R“ steht z.B. für das 5–Tupel (z_0, a, a, R, R).

	$a, \square$	$b, \square$	$\square, \square$	$\square, a$	$\square, b$
z_0	z_0, a, a, R, R	z_0, b, b, R, R	$z_1, \square, \square, N, L$	–	–
z_1	–	–	$z_2, \square, \square, N, R$	$z_1, \square, a, N, L$	$z_1, \square, b, N, L$
z_2	–	–	$z_3, \square, \square, L, N$	$z_2, a, \square, R, R$	$z_2, b, \square, R, R$
z_3	$z_3, a, \square, L, N$	$z_3, b, \square, L, N$	$z_4, \square, \square, R, N$	–	–
z_4	–	–	–	–	–

   Im Zustand z_0 erfolgt die erste Kopie, im Zustand z_1 wird der zweite Kopf zurückgefahren, und im Zustand z_2 erfolgt die zweite Kopie. Nach dem zweiten Zurückfahren durch den Zustand z_3 akzeptiert M im Zustand z_4.

b) Im Zustand z_0 verweilt M für genau $n + 1$ Schritte, weil sie das erste Blank hinter der Eingabe lesen muss, um dessen Ende zu erkennen. Genauso benötigt die TM $n + 1$ Schritte, um den zweiten Kopf zurückzufahren sowie weitere $n+1$ Schritte, um die zweite Kopie anzufertigen. Das Zurückfahren über das nun doppelt so lange Wort benötigt abschließend nochmals $2n + 1$ Schritte, so dass M insgesamt $5n + 4$ Schritte ausgeführt.

c) Wegen $5n+4 = O(n)$ arbeitet M in linearer Zeit. Dies ist optimal, da allein das Abtasten der Eingabe schon linear viel Zeit benötigt.

Aufgabe 4: Hier ist z.B. $M = (\{z_0, z_1, \ldots, z_9\}, \{0, 1\}, \{0, 1, \square\}, \delta, z_0, \square, \{z_9\})$ mit der Turingtafel

	0	1	$\square$
z_0	$(z_1, 0, N)$	–	–
z_1	$(z_2, \square, R)$	–	$(z_9, \square, N)$
z_2	$(z_3, \square, R)$	$(z_4, \square, R)$	–
z_3	$(z_3, 0, R)$	$(z_4, 0, R)$	–
z_4	$(z_4, 0, R)$	$(z_4, 1, R)$	$(z_5, \square, L)$
z_5	$(z_6, \square, L)$	–	–
z_6	$(z_7, \square, L)$	$(z_8, \square, L)$	–
z_7	$(z_7, 0, L)$	$(z_8, 0, L)$	–
z_8	$(z_8, 0, L)$	$(z_8, 1, L)$	$(z_1, \square, R)$
z_9	–	–	–

eine passende Turingmaschine für L.

Der Startzustand z_0 dient lediglich dazu, das leere Wort als Eingabe zu verwerfen. Ansonsten funktioniert M wie folgt. Angesetzt auf eine Eingabe der Form $0^n1^{2n}0^n$ mit $n \geq 1$ überschreibt M im Zustand z_1 zunächst die führende Null, d.h. danach steht das Wort $0^{n-1}1^{2n}0^n$ auf dem Band. In den beiden Folgezuständen z_2 und z_3 löscht M auch die nächste Null (sofern vorhanden) und überschreibt damit die erste Eins, oder aber löscht die folgende Eins direkt. In beiden Fällen steht danach das Wort $0^{n-1}1^{2n-1}0^n$ auf dem Band. Danach wechselt M in den Zustand z_4 und bewegt dort den Kopf an das rechte Ende der Eingabe. Die weiteren Zustände z_5, z_6, und z_7 erledigen die gleiche Arbeit wie z_1, z_2 und z_3 in gespiegelter Form, d.h. jetzt wird am rechten Wortende eine Null gelöscht und die am weitesten rechts stehende Eins mit einer entsprechend verschobenen Null (sofern vorhanden) überschrieben. Danach steht auf dem Band das Wort $0^{n-1}1^{2n-2}0^{n-1} = 0^{n-1}1^{2(n-1)}0^{n-1}$. Der Zustand z_8 stellt noch anschließend den Kopf an das linke Wortende zurück. Insgesamt wird so ein gültiges Eingabewort auf das nächstkürzere Wort aus der Sprache L reduziert, und die Verarbeitung startet von vorn.

Der Lauf von M endet nur dann erfolgreich, wenn zum Schluss das gesamte Band gelöscht wurde. Bei Ungereimtheiten jeglicher Art stösst M auf undefinierte Übergänge und akzeptiert deshalb nicht.

Aufgabe 5: Sei $M = (\{z_0, z_1, z_2, z_3\}, \{a, b\}, \{a, b, \square\}, \delta, z_0, \square, \{z_3\})$ eine DTM mit folgender Turingtafel:

	a	b	$\square$
z_0	(z_0, a, R)	(z_0, b, R)	$(z_1, \square, L)$
z_1	$(z_2, \square, L)$	$(z_2, \square, L)$	$(z_1, \square, N)$
z_2	(z_2, a, L)	(z_2, b, L)	$(z_3, \square, R)$
z_3	–	–	–

Der Kopf von M sucht in z_0 das rechte Ende der Eingabe. In z_1 wird anschließend das letzte Zeichen durch ein Blank überschrieben. Beachten Sie, dass M für immer in z_1 verbleibt, falls kein zu überschreibendes letztes Zeichen vorgefunden wird. In diesem Fall war die Eingabe nämlich das leere Wort, und auf diesem ist f nicht definiert. In z_2 bewegt sich der Kopf nach links zurück, und M wechselt abschließend in den Endzustand z_3.

Aufgabe 6: Sei $M = (\{z_0, z_1, z_2, z_4\}, \{a, b\}, \{a, b, \square\}, \delta, z_0, \square, \{z_4\})$ eine DTM mit folgender Turingtafel:

	a	b	$\square$
z_0	(z_1, a, R)	(z_2, b, R)	$(z_0, \square, N)$
z_1	(z_1, a, R)	(z_2, b, R)	(z_3, a, L)
z_2	(z_1, a, R)	(z_2, b, R)	(z_3, b, L)
z_3	(z_3, a, L)	(z_3, b, L)	$(z_4, \square, R)$
z_4	$-$	$-$	$-$

Zunächst testet M im Zustand z_0, ob das Eingabewort leer ist. Die Funktion f ist für das leere Wort nicht definiert, so dass M „hängen“ bleibt. Ansonsten sucht M in den Zuständen z_1 und z_2 das rechte Ende der Eingabe, wobei das Wort nicht verändert wird. M befindet sich dabei immer im Zustand z_1, sofern zuletzt ein a gelesen wurde, und andernfalls in z_2. Somit weiß M, mit welchem Symbol das erste Blank hinter dem Eingabewort zu überschreiben ist. In z_3 bewegt sich der Kopf nach links zurück, und M wechselt abschließend in den Endzustand z_4.