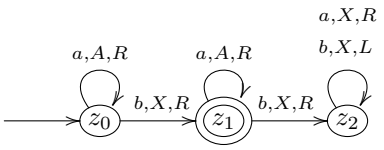
FGI-1 – Formale Grundlagen der Informatik I

Logik, Automaten und Formale Sprachen

Musterlösung 5 : Turing-Maschinen

Präsenzteil am 5.-8. Mai – Abgabe am 12.-15. Mai 2015

Präsenzaufgabe 5.1: Gegeben sei die folgende Turing-Maschine M. Beschreiben Sie die Funktionsweise von M und bestimmen Sie $L(M)$.



Lösung: Die TM M liest zunächst as, ersetzt diese durch As (was aber hier nicht weiter wichtig ist) und bewegt dabei den Lese-/Schreibkopf nach rechts. Trifft sie auf ein b, so wird b durch X ersetzt (was ebenfalls unwichtig ist) und in den Zustand z_1 gewechselt, in dem das Wort akzeptiert wird. M akzeptiert also gerade die Sprache $a^* \cdot b \cdot (a+b)^*$ (also jene Wörter aus as und bs, die mindestens ein b enthalten). Man beachte, dass es nicht wichtig ist, dass w nicht zu Ende gelesen wurde oder dass, wie hier, die TM gar nicht (im Endzustand) anhält. Ein Eingabewort w wird akzeptiert, sobald die TM bei einer Rechnung auf w (irgendwann!) in einen Endzustand gelangt. Man kann stets eine äquivalente TM konstruieren, die den Endzustand nicht mehr verlässt und dort wirklich hält, einfach indem man alle Kanten, die aus einem Endzustand herausführen, entfernt.

Der Vollständigkeit halber aber noch zur weiteren Funktionsweise der TM: In z_1 können nun wieder beliebig viele as gelesen werden, wobei der Kopf wieder nach rechts wandert und die as durch As ersetzt werden. Folgt ein zweites b, wird der Endzustand verlassen und nach z_2 gewechselt (allerdings wird das Wort dennoch akzeptiert, da wir ja bereits einen Endzustand besucht hatten!). In z_2 können nun weiter as gelesen werden. Kommt jedoch ein (drittes) b, so wird der Kopf nach links bewegt und der Kopf steht nun in jedem Fall auf einem X. Da die TM weiterhin in z_2 ist und hier kein entsprechender Übergang (eine Kante, die vom Eingabeband ein X liest) möglich ist, hält die TM. Die TM hält also auf allen Wörtern. Bei Worten, die (mindestens) drei b enthalten, aber nicht auf dem dritten b enden, werden allerdings alle folgenden Buchstaben nicht mehr betrachtet. Akzeptiert werden, wie oben besprochen, alle Worte, die mindestens ein b enthalten. Der Zustand z_2 sowie die Kanten aus z_1 heraus (inkl. der Schleife an z_1) können entfernt werden, ohne dass sich an der akzeptierten Sprache etwas ändert.

Präsenzaufgabe 5.2:

1. Konstruieren Sie eine TM, die die Funktion $f : \{0,1\}^* \to \{0,1\}^*$ mit $f(w) = [w]$ berechnet, wobei $[w]$ wie folgt definiert ist: $[\lambda] = \lambda$ und $[w'0] = [w']1$ und $[w'1] = [w']0$ (d.h. 0 wird zu 1 und 1 zu 0, bspw. ist $[001] = 110$). Begründen Sie, dass Ihre TM das Gewünschte leistet.

2. Beschreiben Sie die Arbeitsweise einer TM, die die Sprache

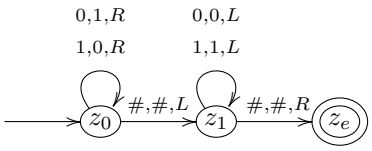
$$L = \{w\$[w] \mid w \in \{0,1\}^*\}$$

akzeptiert. (Sie sollen hier die Arbeitsweise nur beschreiben. Die Angabe eines Zustandsübergangsdiagramms ist nicht nötig!)

Lösung:

1. Eine TM, die die angegebene Funktion berechnet, liest bei Eingabe von w einmal von links nach rechts über w rüber und wandelt dabei 0 in 1 und 1 in 0 um. Rechts angekommen (erkennbar daran, dass ein # erreicht wird), läuft der Kopf wieder komplett nach links (ohne an der Bandinschrift noch etwas zu ändern), bis er am Anfang des Wortes (jetzt $[w]$) angekommen ist. Dort wird dann in einen Endzustand gewechselt, so dass wir von der Konfiguration $z_0 w$ in die Konfiguration $z_e[w]$ gelangt sind, wobei z_0 der Start-, z_e der Endzustand ist.

   Eine TM, die obige Idee umsetzt, ist wie folgt gegeben:



   In z_0 wird, wie in der Idee beschrieben, über das Eingabewort w gelesen und jede 0 in 1 und jede 1 in 0 gewandelt. Ist der Lese-/Schreibkopf einmal über das Wort gewandert, so steht er nun auf dem Symbol #. Er bewegt sich nach links und wir wechseln in den Zustand z_1. In z_1 wird nach links über das Wort gelesen, ohne es weiter zu ändern. Sind wir wieder am Anfang (d.h. wir haben wieder # erreicht), so wird noch einmal der Kopf nach rechts bewegt, so dass er nun auf dem ersten Symbol von $[w]$ steht. Gleichzeitig wird in den Zustand z_e gewechselt und die TM hält.

2. Um $L = \{w\$[w] \mid w \in \{0,1\}^*\}$ zu akzeptiert, kann man auf die eben konstruierte TM zurückgreifen. Eine Möglichkeit wäre die folgende: Die Eingabe muss von der Form $w\$v$ sein. Ansonsten kann man gleich ablehnen. Die Frage ist dann, ob $v = [w]$ gilt. Hierzu wird zunächst mit der obigen TM w in $[w]$ umgewandelt. Dabei muss obige TM so abgewandelt werden, dass nicht bei # aufgehört wird, sondern bei \$. Man hat dann $[w]\$v$ auf dem Band stehen. Die TM kann nun das erste Symbol von $[w]$ lesen, sich das gelesene Symbol im Zustand merken, das Symbol streichen (durch # ersetzen) und zum ersten Symbol hinter dem \$ gehen. Ist dies identisch (wir hatten uns das Symbol im Zustand gemerkt, daher ist dieser Vergleich jetzt möglich), wird es gelöscht und dann nach ganz links gegangen und das zweite Symbol von $[w]$ gelesen und gelöscht. (Man findet das Symbol, da links daneben das erste Mal ein # auftritt). Dann wird wieder nach rechts bis zum \$ gegangen. Von dort weiter nach rechts (über das # rüber) zum ersten Symbol. Dies muss dann wieder identisch zum eben gelöschten sein. Nun geht man wieder nach links usw. Ein Problem gibt es hierbei:

Wenn man rechts von dem \$ ein Symbol sucht und keines findet, terminiert die TM nicht, sondern wandert mit dem LSK immer weiter nach rechts. Um dies zu verhindern, muss man zu Anfang einmal ganz nach rechts gehen und das Ende des Wortes markieren (mit einem ansonsten nicht benutzen Symbol wie z.B. X). Wenn nun bei dem abwechselnden Löschvorgang irgendwann links von dem \$ sofort ein # kommt, weiß man, dass $[w]$ wurde komplett abgearbeitet. Man muss dann noch einmal nach rechts gehen und überprüfen, ob auch das v komplett gelöscht wurde. Dies erkennt man daran, dass zwischen dem \$ und dem X nur Blanks (d.h. #) stehen. Ist dies der Fall akzeptiert die TM. (Man beachte, dass die Symbole von v stets nur gelöscht werden, wenn sie mit dem gerade gelöschten Symbol von $[w]$ übereinstimmen.)

Will man das w nicht zerstören, könnte man, statt die 0 und die 1 durch # zu löschen, bpsw. die 0 durch $\overline{0}$ und die 1 durch $\overline{1}$ ersetzen. Dann hat man die Symbole auch als "bearbeitet" markiert, aber sie nicht gelöscht. Später lässt sich hieraus, falls nötig, das ursprüngliche w wiedergewinnen.

Präsenzaufgabe 5.3: Beschreiben Sie die Arbeitsweise einer TM, die die Funktion $f(x, y) = x \cdot y$ berechnet. Dabei sollen x und y natürliche Zahlen sein, die in binärer Kodierung vorliegen.

Beschreiben Sie anschließend die Arbeitsweise einer TM, die die Sprache $L = \{(x, y, z) \mid [x]_2 \cdot [y]_2 = [z]_2\}$ akzeptiert. Dabei sind x, y, z wie eben natürliche Zahlen in Binärkodierung und $[x]_2$ etc. meint die natürliche Zahl, die man erhält, wenn man x als Binärkodierung interpretiert, also z.B. $[1001]_2 = 9$. Nutzen Sie hier insb. Ihre eben konstruierte TM, die die Funktion $f(x, y) = x \cdot y$ berechnet!

Lösung: Wir beschreiben eine Drei-Band-TM, die f berechnet: Wir testen zunächst, ob $x = 0$ ist. Falls ja, schreiben wir 0 auf das Band, löschen alles andere und sind fertig. Ansonsten lesen wir zunächst x vom ersten Band und notieren dies auf dem zweiten Band. Auf dem dritten Band notieren wir eine 0.

Nun rechnen wir auf dem zweiten Band $x - 1$. Wir gehen ans rechte Ende von x und lesen von rechts nach links über x. Aus einer 0 machen wir eine 1 und gehen weiter nach links. Treffen wir auf eine 1 machen wir hieraus eine 0 und hören auf. Wir haben x um 1 reduziert.

Auf dem dritten Band addieren wir nun y zu der bisherigen Zahl hinzu. Das machen wir wie folgt: Wir gehen auf dem ersten Band an den rechten Rand von y und auf dem dritten Band an den rechten Rand der bisherigen Zahl. Nun lesen wir beide Zahlen von rechts nach links und aktualisieren die Zahl auf dem dritten Band. (Wobei wir uns den Überhang, sollte er existieren, im Zustand merken. Haben wir bspw. 0 und 1 und keinen Überhang, so notieren wir eine 1 und machen weiter. Haben wir 0 und 1 und einen Überhang, so notieren wir eine 0, wechseln aber in jenen Zustand, in dem wir uns merken, dass wir jetzt keinen Überhang mehr haben.) Dabei kann es passieren, dass die Zahl auf dem dritten Band nach links wächst. Haben wir nur ein einseitig unendliches Band, so müssen wir stets zu Beginn dieser Operation die Zahl auf dem dritten Band so um ein Feld verschieben, dass ganz links ein freies Feld ist (sofern dort nicht noch eines ist).

Dies wird nun so lange wiederholt, bis die Zahl auf dem zweiten Band den Wert 0 erreicht. Dann sind wir fertig und auf dem dritten Band steht das Ergebnis.

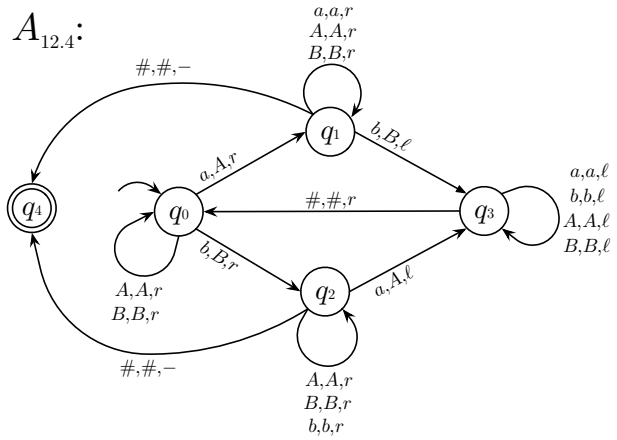
Die Arbeitsweise einer TM für die Sprache L ist sehr ähnlich. Eine Mehrband TM kann zunächst $x \cdot y =: c$ auf die eben beschriebene Weise berechnen. Dann wird das berechnete c noch mit dem z auf dem Eingabeband verglichen. Sind sie identisch, wird akzeptiert, sonst nicht. (Vorher ist ggf. eine Syntaxüberprüfung nötig, ob die Eingabe überhaupt von der Form (x, y, z) ist, aber dazu muss die TM nur einmal von links nach rechts über die Eingabe lesen.)

Übungsaufgabe 5.4: Geben Sie eine DTM an, die die Sprache $L = \{w \in \{a,b\}^* \mid |w|_a \neq |w|_b\}$ akzeptiert. Zeichnen Sie dazu das Zustandsübergangsdiagramm und begründen Sie, dass Ihre TM das Gewünschte leistet.

von
5

Lösung: Es gibt viel Möglichkeiten, diese Sprache zu akzeptieren. Man könnte z.B. auch einen Zähler implementieren, mit dem man zuerst die Anzahl der as ermittelt. Der Zähler wird zu Anfang auf 0 gesetzt. Jedes gefundene a wird dann gelöscht und der Zähler um eins erhöht. Im Anschluss werden die bs gelöscht und der Zähler wieder um eins verringert. Erreicht der Zähler 0 und sind dann alle as und bs gelöscht, wird nicht akzeptiert. Hat der Zähler einen Wert ungleich 0 wird akzeptiert (wobei akzeptiert werden kann, sobald der Zähler negativ werden würde). Dabei ist es u.U. erforderlich, das Eingabewort zunächst mit zwei Begrenzern zu umschließen, so dass man ermitteln kann, dass man alle as und bs gelöscht hat.

Wir machen das hier aber anders. Das Zustandsdiagramm unserer TM $A_{12.4}$ (die Benennung ist nur aus historischen Gründen) ist unten abgebildet. Es werden von links nach rechts das erste auftretenden Symbol a (bzw. b) und nachfolgend ein zweites, passendes Symbol b (bzw. a) durch jeweiliges Umwandeln in Großbuchstaben markiert. Das wird solange, beginnend am linken Rand des Eingabewortes, wiederholt, bis das paarweise Markieren Erfolg hatte, also nicht akzeptiert werden darf. Akzeptiert werden darf nur, wenn überzählige Symbole existieren. Nach diesen wird also gesucht.



Im Startzustand wird der *LSK* ohne Schreibaktion nach rechts über alle Großbuchstaben bis auf das erste Feld mit einem Kleinbuchstaben geführt. Das gefundene kleine Symbol wird durch den entsprechenden Großbuchstaben ersetzt und bei a (bzw. b) in den Zustand q_1 (bzw. q_2) gewechselt. Immer wenn der Zustand q_1 (bzw. q_2) erreicht wurde, muss noch nach dem entsprechenden Kleinbuchstaben b (bzw. a) gesucht werden. Bei Erfolg wird dieser mit Übergang in den Zustand q_3 markiert. In q_3 wandert der *LSK* zum Anfang der Eingabe zurück und beginnt die Markierungssuche von Neuem. Im Zustand q_0 besitzt die Bandinschrift also immer eine identische Anzahl von A's und B's. Wenn alle Kleinbuchstaben dann paarweise durch Großschreibung markiert wurden, wird im Zustand q_0 das leere Feld # am rechten Ende der Bandinschrift gefunden und die DTM blockiert. Im Eingabewort waren gleich viele a's und b's und es darf nicht akzeptiert werden.

Gibt es jedoch noch ein überzähliges a (bzw. b), so erreicht man aus q_0 einen der Zustände q_1 oder in q_2 und der *LSK* steht dann dort auf dem unbeschriebenen Feld hinter der Bandinschrift und es

wird dann (und nur dann) in den Endzustand gewechselt. Hat das Eingabewort nicht die gleiche Anzahl von a's und b's, kann der Endzustand also erreicht werden.

Die TM oben akzeptiert also jene Wörter $w \in \{a, b\}^*$ mit $|w|_a \neq |w|_b$ und auch nur genau diese.

Übungsaufgabe 5.5: Geben Sie eine kontextfreie Grammatik für

von
3

$$L_1 = \{a^{i+j}b^{i+k}c^k \mid i, j, k \geq 0\}$$

an und zeigen Sie, dass Ihre Grammatik das Gewünschte leistet (indem sie zwei Mengeninklusionen zeigen).

Lösung: Da $a^{i+j}b^{i+k}c^k = a^j a^i b^i b^k c^k$ gilt, werden also links beliebig viele a's generiert (unabhängig vom Rest des Wortes). Dann gibt es einen Teil, der das Teilwort $a^i b^i$ generiert und einen Teil, der das Teilwort $b^k c^k$ generiert. Nimmt man dies zusammen, so erhält man die folgende Grammatik G_1 mit den Produktionen

$$\begin{array}{lcl} S & \longrightarrow & AVH \\ A & \longrightarrow & aA \mid \lambda \\ V & \longrightarrow & aVb \mid \lambda \\ H & \longrightarrow & bHc \mid \lambda \end{array}$$

Wir wollen $L(G_1) = L_1$ zeigen. Sei $w = a^j a^i b^i b^k c^k \in L_1$ mit $i, j, k \geq 0$, dann wird w wie folgt aus S abgeleitet

$$\begin{array}{lcl} S & \Longrightarrow & AVH \\ & \stackrel{j}{\Longrightarrow} & a^j AVH \Longrightarrow a^j VH \\ & \stackrel{i}{\Longrightarrow} & a^j a^i Vb^i H \Longrightarrow a^j a^i b^i H \\ & \stackrel{k}{\Longrightarrow} & a^j a^i b^i b^k Hc^k \Longrightarrow a^j a^i b^i b^k c^k \end{array}$$

Wobei $\stackrel{i}{\Longrightarrow}$ besagt, dass eine Regel i-mal angewendet wurde. Daraus folgt $w \in L(G_1)$.

Sei umgekehrt $w \in L(G_1)$, dann kann jede Ableitung nur mit der Produktion $S \longrightarrow AVH$ beginnen. Anschliessend wird A zu a^j abgeleitet, V zu $a^i b^i$ und H zu $b^k c^k$, was man auch leicht per Induktion beweisen kann. Insgesamt ergibt sich so, dass auch $w \in L_1$ gelten muss, die Mengen also gleich sind.

Übungsaufgabe 5.6:

von
4

1. Zeigen Sie, dass eine TM M einen PDA A simulieren kann, indem Sie beschreiben, was eine TM M tun muss, damit $L(M) = L(A)$ gilt (oder wahlweise auch $L(M) = N(A)$).
2. Beschreiben Sie dann eine TM, die $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ akzeptiert.

Mit beiden Teilaufgaben zusammen (und da wir wissen, dass L oben nicht kontextfrei ist) haben Sie gezeigt, dass CF eine echte Teilfamilie der Familie der von Turing-Maschinen akzeptierten Sprachen ist!

(Sie brauchen in dieser Aufgabe kein Zustandsübergangsdiagramm zeichnen. Es genügt, wenn Sie die Arbeitsweise Ihrer TM beschreiben und die Korrektheit begründen.)

Lösung:

1. Wir beschreiben die Arbeitsweise einer TM M mit drei extra Arbeitsbändern.

   Sei A nun ein PDA. Wir konstruieren M wie folgt. M schreibt zunächst auf das erste Arbeitsband den Automaten A. Auf das zweite Arbeitsband wird die Startkonfigurationen $(z, w, \bot)$ des PDA geschrieben, wobei w das Eingabewort ist, was die TM auf dem Eingabeband abliest.

   Nun betrachtet M die Konfiguration auf dem zweiten Arbeitsband. Sie liest das erste Symbol von w, das oberste Symbol vom Keller und den Zustand, in dem der PDA gerade ist. Dann schaut sie auf dem ersten Arbeitsband nach, welche Zustandsübergänge möglich sind und schreibt die entstehenden Konfigurationen auf das dritte Arbeitsband. Ist sie damit fertig, wird der Inhalt des zweiten Arbeitsbandes gelöscht und der Inhalt des dritten Arbeitsbandes auf das zweite Arbeitsband übertragen. Das dritte Band ist nun leer, auf dem zweiten Arbeitsband stehen alle Konfigurationen, in denen der PDA nichtdeterministisch sein könnte, und auf dem ersten Arbeitsband steht weiterhin die Beschreibung des PDA.

   Die TM wiederholt dies nun. Sie betrachtet nacheinander alle Konfigurationen auf dem zweiten Band, berechnet für jede die nichtdeterministischen Nachfolgekonfigurationen und schreibt diese auf das dritte Band. Dann wird das zweite Band gelöscht und der Inhalt des dritten Bandes auf das zweite Band übertragen.

   Wird dabei irgendwann einmal eine Konfiguration erreicht, bei der der Keller leer ist und das Eingabewort zu Ende gelesen wurde, so akzeptiert die TM. Damit akzeptiert die TM genau die Wörter, die auch der PDA akzeptiert und es gilt $L(M) = L(A)$.

   (Anmerkung: Um die Aktionen "das zweite Arbeitsband löschen" und "den Inhalt des dritten Arbeitsbandes auf das zweite Arbeitsband kopieren" ausführen zu können, ist es nützlich den Inhalt des Bandes mit zwei speziellen Markern zu markieren (z.B. mit α und β). Dann kennt man stets Anfang und Ende. Außerdem könnte man, statt den PDA zu Anfang auf das erste Band zu schreiben, diesen auch direkt in die Zustandsübergänge der TM kodieren. Für den Beweis macht das aber keinen Unterschied.)

2. Sei A_1 eine TM für die Sprache $L_1 := \{a^n b^n \mid n \in \mathbb{N}\}$, die nach der ersten Teilaufgabe existiert, da L_1 von einem PDA akzeptiert werden kann. (Alternativ kann man sich auch schnell eine TM überlegen, die L_1 akzeptiert, z.B. indem sie einen Zähler für die as und einen zweiten für die bs hat oder indem sie wiederholt über das Band liest und dabei as und bs streicht.) Sei ferner A_2 eine TM für die Sprache $L_2 := \{b^n c^n \mid n \in \mathbb{N}\}$, die wir durch einfaches Austauschen von a durch b und von b durch c aus der TM A_1 erhalten.

Eine TM A, die die Sprache $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ akzeptiert, arbeitet nun wie folgt: Bei Eingabe eines Wortes w testen wir zunächst ähnlich wie in der ersten Aufgabe, ob das Wort von der Form $a^*b^*c^*$ ist. Falls nein, lehnen wir sofort ab. Falls ja, hat w die Form $w = a^i b^j c^k$. Wir schreiben nun das Teilwort $a^i b^j$ auf ein zweites Arbeitsband und überprüfen, mit der TM A_1, ob $i = j$ gilt. Falls nein, lehnen wir ab. Falls ja, wissen wir, dass $i = j$ ist. Dann löschen wir das zweite Arbeitsband und schreiben das Teilwort $b^j c^k$ auf dieses. Nun überprüfen wir noch mit der TM A_2, ob auch $j = k$ gilt. Falls ja, akzeptieren wir, da dann das Wort $a^i b^j c^k$ mit $i = j = k$ in der Sprache L ist.

Man sieht an dieser Beschreibung, dass wir damit auch nur Worte dieser Art akzeptieren können und damit gilt $L(A) = L$.

Informationen und Unterlagen zur Veranstaltung unter:

`http://www.informatik.uni-hamburg.de/TGI/lehre/vl/SS15/FGI1`