FGI-1 – Formale Grundlagen der Informatik I

Logik, Automaten und Formale Sprachen

Musterlösung 6 : Berechenbarkeit

Präsenzteil am 12.-15. Mai – Abgabe am 19.-22. Mai 2015

Präsenzaufgabe 6.1: In den Folien ist die Überführungsfunktion der Turing-Maschine durch

$$\delta : (Z \times \Gamma) \to (\Gamma \times \{L, R, H\} \times Z)$$

gegeben. Im Skript aber durch

$$\delta : (Z \times \Gamma) \to (\Gamma \times \{L, R\} \times Z)$$

(wobei wir hier noch Y durch Γ ersetzt haben). In einem Fall kann der Lese-/Schreibkopf also bei einem Kantenübergang seine Postion halten, in dem anderen Fall nicht. Zeigen Sie, dass dies keinen Unterschied macht, dass es also zu jeder TM, bei der der LSK seine Position Halten kann, eine äquivalente TM gibt, bei der er das nicht kann, und umgekehrt.

Lösung: Die eine Richtung ist ganz leicht: Sei M eine TM, die ihren LSK nicht halten kann. Dann kann eine TM M', die ihren LSK halten kann, genau wie M konstruiert werden. Ihre zusätzliche Fähigkeit benutzt M' dann gar nicht. Da die TMs völlig gleich sind, gilt $L(M) = L(M')$.

Die andere Richtung ist interessanter. Sei M eine TM, die ihren LSK halten kann. Wir wollen eine äquivalente TM M' konstruieren, die bei jedem Kantenübergang ihren LSK bewegt.

Zunächst sei M' wie M aufgebaut. Seien $c_1, \ldots, c_m$ die Kanten von M. Sei nun $c_i = (z, x, X, H, z') \in K$ ein Kantenübergang von M, bei dem der LSK nicht bewegt wird. Wir löschen diese Kante in M' und fügen einen neuen Zustand z_{c_i} sowie eine Kante (z, x, X, R, z_{c_i}) hinzu. Dann fügen wir noch für jedes $y \in \Gamma$ eine Kante (z_{c_i}, y, y, L, z') hinzu. So verfahren wir für jede Kante in M, bei der der LSK nicht bewegt wird.

Es gilt nun $L(M) = L(M')$, da das Stehenbleiben des LSK von M nun durch eine Rechts-Links-Bewegung nachgeahmt werden kann. Genauer: Sei $w \in L(M)$. Wenn bei der Erfolgsrechnung keine Kante benutzt wird, in der M den LSK nicht bewegt, so ist in M' die gleiche Erfolgsrechnung möglich. Wird eine solche Kante $c = (z, x, X, H, z')$ benutzt, so kann M' zunächst die Kante (z, x, X, R, z_c) benutzen. Eine Bewegung des LSK nach rechts ist bei einem einseitig unendlichen Band immer möglich (hätten wir hier eine Links-Bewegung benutzt, hätte dies am linken Rand nicht geklappt). Auf dem Feld, auf dem der LSK nun ist, ist ein Symbol $y \in \Gamma$ (dies kann auch $\#$ sein) und mit der Kante (z, y, y, L, z') ist M' nun wieder in der gleichen Konfiguration wie M und die Rechnung kann fortgesetzt werden. Damit kann M' die Rechnung von M nachvollziehen und es gilt $w \in L(M')$. Ist umgekehrt $w \in L(M')$, so kann man ähnlich verfahren. Falls dann ein Zustand z_c besucht wird, der aufgrund einer Kante c von M eingeführt wurde, so kann dieser wieder verlassen werden. Die beiden Kanten, mit denen der Zustand z_c besucht und wieder verlassen wurde, entsprechen dabei gerade der Kante c in M. Es gilt dann auch $w \in L(M)$.

Man beachte, dass es wichtig ist, für *jede* Kante c_i in M mit obigen Eigenschaften einen Zustand z_{c_i} wie in der obigen Konstruktion einzuführen. Führt man bspw. für zwei Kanten (z, x, X, H, z') und (z, u, U, H, z'') mit $z' \neq z''$ nur einen neuen Zustand z_c ein, sowie zwei Kanten (z, x, X, R, z_c) und (z, u, U, R, z_c) und nun für jedes $y \in \Gamma$ Kanten (z_c, y, y, L, z') sowie (z_c, y, y, L, z''), so könnte man mit (z, x, X, R, z_c) den Zustand z verlassen und mit (z_c, y, y, L, z'') nach z'' gelangen. Die Kante (z, x, X, H, z'') war aber gar nicht in M vorhanden! Man merkt sich über den Zustand z_{c_i} also explizit welche Kante in M benutzt wurde.

Präsenzaufgabe 6.2: Zeigen Sie, dass

$$DFA_\emptyset = \{\langle A\rangle \mid A \text{ ist ein DFA und } L(A) = \emptyset\}$$

entscheidbar ist.

Lösung: Um bei Vorlage eines DFA A zu entscheiden, ob $L(A)$ leer ist, also kein Wort akzeptiert wird, muss man überprüfen, ob ein Endzustand von A erreichbar ist. Nur wenn dies nicht möglich ist, gilt $L(A) = \emptyset$. Ist ein Endzustand bspw. über die Kanten mit der Beschriftung $x_1, x_2, \ldots, x_j$ erreichbar, so ist $x_1 x_2 \cdots x_j$ ein Wort, das A akzeptiert.

Um zu ermitteln, ob ein Endzustand erreichbar ist, kann die TM wie folgt verfahren: Zuerst markiert sie den Startzustand. Im nächsten Schritt werden alle Zustände markiert, die von dem Startzustand aus mit einer einzelnen Kante erreichbar sind. Dann werden immer alle markierten Zustände durchgegangen und die von ihnen aus über eine Kante erreichbaren Zustände markiert. Dies wird wiederholt, bis keine neuen Zustände mehr markiert werden. Ist unter den so markierten Zuständen kein Endzustand, so gilt $L(A) = \emptyset$ und die TM akzeptiert die Eingabe A. Sonst lehnt sie ab. Die TM hält immer, da DFAs nur endlich viele Zustände enthalten und obiges Verfahren daher terminiert. Außerdem akzeptiert sie genau die DFAs, die keine Wörter akzeptieren. Damit entscheidet sie $DFA_\emptyset$.

(Obiges Verfahren ist nicht besonders effizient, da auch Zustände erneut betrachtet werden, die bereits betrachtet wurden. Effizienter wäre es, diese mit einer zweiten Markierung zu versehen.)

Präsenzaufgabe 6.3: Rudi Ratlos möchte zeigen, dass das folgende Problem `Print0` unentscheidbar ist:

> Gegeben eine TM M. Wenn M mit dem leeren Band gestartet wird, so gibt M irgendwann einmal 0 aus.

Leider hat Rudi keine Ahnung, was er tun soll. Können Sie ihm erklären, was er tun muss? Können Sie dann den Beweis sogar führen?

Lösung: Eine üblich Methode um zu zeigen, dass ein neues Problem (hier `Print0`) unentscheidbar ist, ist die folgende: Man nimmt an, das Problem wäre entscheidbar und konstruiert dann unter Nutzung des Algorithmus, der das Problem entscheidet (der nach Annahme existiert), einen neuen Algorithmus, der ein Problem entscheidet, von dem man bereits weiß, dass es unentscheidbar ist. Wegen dieses Widerspruchs (ein unentscheidbares Problem als entscheidbar nachzuweisen) kann die ursprüngliche Annahme nicht stimmen und das neue Problem muss also auch unentscheidbar sein.

Zur Illustration wollen wir noch zeigen, dass `Print0` unentscheidbar ist. Nehmen wir also an, `Print0` wäre entscheidbar, d.h. es gibt eine TM M_{Pr0}, die bei Vorlage einer TM M entscheidet, ob diese bei Eingabe des leeren Wortes eine 0 ausgibt oder nicht.

Zu jeder TM A und jedem Wort w kann leicht (und algorithmisch, d.h. durch eine TM) eine TM $M_{A,w}$ konstruiert werden, die, auf dem leeren Band gestartet, A auf w simuliert. Hält A auf w an, so gibt $M_{A,w}$ eine 0 aus. (Hält A nicht an, so simuliert $M_{A,w}$ die TM A einfach weiter, gibt aber nichts aus, insb. keine 0.)

Wir konstruieren nun eine TM M_H, die das Halteproblem entscheidet (also das Problem, bei Vorlage einer TM A und eines Wortes w zu entscheiden, ob M auf w anhält oder nicht). M_H arbeitet wie folgt: Bei Eingabe $\langle A, w\rangle$ wird zunächst die TM $M_{A,w}$ konstruiert. Diese wird aber nun nicht etwa gestartet, sondern M_{Pr0} als Eingabe vorgelegt. M_{Pr0} akzeptiert nun genau dann, wenn $M_{A,w}$, gestartet auf dem leeren Band, eine 0 ausgibt, was genau dann passiert, wenn A auf w hält. Die TM M_H entscheidet damit das Halteproblem. (Man beachte wie wichtig es dabei ist, dass M_{Pr0} das Problem, ob eine TM eine 0 zurückliefert oder nicht, *entscheidet*.)

Da wir aber bereits wissen, dass das Halteproblem nicht entscheidbar ist, kann die angenommene Existenz von M_{Pr0} nicht stimmen (alle anderen Schritte gehen nämlich ohne weitere Annahmen, insb. die Konstruktion von $M_{A,w}$) und damit muss `Print0` unentscheidbar sein.

Aus der Unentscheidbarkeit von `Print0` kann man schnell auch auf die Unentscheidbarkeit weiterer Probleme schließen; bspw. auf die des sehr ähnlichen Problems, ob eine Java-Routine den Wert 0 zurückliefert. Auch dieses Problem ist unentscheidbar, d.h. es gibt keinen Algorithmus, der bei Vorlage eines Java-Programms und eines Methodennamens entscheidet, ob die Methode den Wert 0 zurückliefert.

Ein weiteres sehr ähnliches Beispiel findet man im Skript im Beweis von Theorem 12.29, wo im Grunde genommen das Gleiche gemacht wird, nur mit 1 als Rückgabewert.

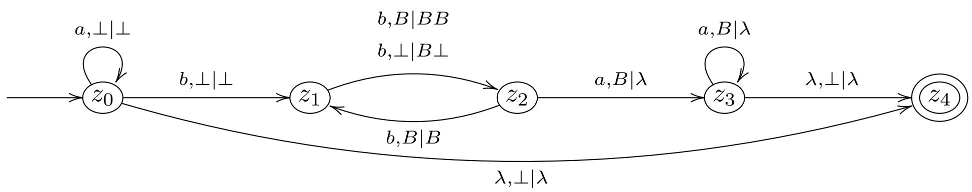
Übungsaufgabe 6.4: Sei $L := \{a^n b^{2m} a^m \mid n, m \geq 0\}$.

von
4

1. Konstruieren Sie einen PDA A, der L im Endzustand akzeptiert und begründen Sie kurz, dass Ihr Automat das Gewünschte leistet.
2. Konstruieren Sie eine kontextfreie Grammatik G mit $L(G) = L$ und begründen Sie kurz, dass Ihre Grammatik das Gewünschte leistet.

Lösung:

1. Ein möglicher Automat A mit $L(A) = L$ (und sogar auch $N(A) = L$):



   In z_0 können zunächst beliebig viele as gelesen werden. Die Kante von z_0 nach z_4 ist zudem dazu da, jene Worte aus L_1 mit $m = 0$ zu akzeptieren und zwar sowohl für den Fall $n = 0$, als auch für den Fall $n > 0$. Auf diese Weise wird also die Menge $\{a\}^* \subset L$ akzeptiert.

   Mit der Kante nach z_1 wird nun ein b gelesen, der Kellerinhalt aber noch nicht verändert. Dies geschieht erst mit der Kante $(z_1, b, \perp, B\perp, z_2)$. Durch die Kanten (z_1, b, B, BB, z_2) und (z_2, b, B, B, z_1) zwischen z_1 und z_2 können nun die $2m$ bs gelesen werden. Man beachte, dass dabei nur m viele Bs auf dem Keller abgelegt werden, da nur beim Lesen jedes *zweiten* b von der Eingabe ein B auf den Keller geschrieben wird. In z_2 hat A also eine gerade Anzahl $2m$ von bs gelesen (und vorne weg noch beliebig viele as in z_0) und der Kellerinhalt ist gerade $B^m\perp$. Durch den Übergang nach z_3 und dann in z_3 kann nun für jedes B im Keller ein a vom Eingabeband gelesen werden, d.h. maximal m viele. Zuletzt wird mittels der Kante $(z_3, \lambda, \perp, \lambda, z_4)$ in den Endzustand gewechselt. Ist das Wort zu Ende gelesen, so wird es hier akzeptiert. Dies geschieht dann sowohl im Endzustand als auch mit leerem Keller.

   Jedes Wort, das der Automat akzeptiert, hat also die Form $a^n b^{2m} a^m$, d.h. es ist $L(A) \subseteq L$. Andersherum sieht man schnell ein, dass Worte dieser Form vom Automaten akzeptiert werden. Somit gilt auch $L \subseteq L(A)$ und damit insgesamt $L = L(A)$.

2. Um eine Grammatik anzugeben, macht man sich zunächst klar, dass n und m nicht voneinander abhängen. Wir konstruieren daher den ersten Block as von einem Symbol aus und dann dei $b^{2m}a^m$ von einem anderen Symbol aus. Dies geht dann ähnlich wie bei $a^i b^i$. Die Grammatik G ist durch folgende Produktionen gegeben:

$$\begin{array}{rcl} S & \longrightarrow & XY \\ X & \longrightarrow & aX \mid \lambda \\ Y & \longrightarrow & bbYa \mid \lambda \end{array}$$

   Wir haben bereits oft gesehen, dass $w \in \{a\}^*$ für jedes w mit $X \stackrel{*}{\Longrightarrow} w$ gilt. Ferner sieht man schnell, dass $w = b^{2i}a^i, i \geq 0$ für ein w mit $Y \stackrel{*}{\Longrightarrow} w$ gilt. Da wir zunächst nur die Produktion $S \to XY$ benutzen können, folgt daraus, dass G jedes Wort aus L generieren kann und auch nur genau diese.

von
4

Übungsaufgabe 6.5:

1. Zeigen Sie, dass die Familie der entscheidbaren Sprachen gegenüber Vereinigung abgeschlossen ist, d.h. wenn L_1 und L_2 entscheidbare Sprachen sind, dann auch $L_1 \cup L_2$.
2. Zeigen Sie, dass die Familie der aufzählbaren Sprachen gegenüber Vereinigung abgeschlossen ist.
3. Ist die Familie der aufzählbaren Sprachen gegenüber Komplementbildung abgeschlossen? Begründen Sie Ihre Antwort!

Lösung:

1. Sind L_1 und L_2 entscheidbar, dann gibt es TMs A_1 und A_2, die L_1 bzw. L_2 entscheiden. Wichtig ist nun, dass A_1 und A_2 auf allen Eingaben anhalten (aber nur jene akzeptieren, die in "ihrer" Sprache sind). Eine TM A, die $L_1 \cup L_2$ entscheidet, arbeitet nun wie folgt: Bei Eingabe von w, wird w kopiert und A_1 auf w ausgeführt. Akzeptiert A_1, so akzeptiert auch A. Akzeptiert A_1 nicht, so wird A_2 auf w gestartet. Akzeptiert A_2, so akzeptiert A. Akzeptiert aber auch A_2 nicht, so ist w weder in L_1 (A_1 hatte ja nicht akzeptiert), noch in L_2 und A lehnt ab. Damit akzeptiert A genau die Worte, die in L_1 oder L_2 sind und lehnt alle anderen ab. Außerdem hält A stets, da auch A_1 und A_2 stets halten. Damit entscheidet A die Sprache $L_1 \cup L_2$.
2. Seien L_1 und L_2 aufzählbar und A_1 und A_2 Turing-Maschinen, die L_1 bzw. L_2 akzeptieren. Ein Hintereinandersausführen von A_1 und A_2 wie eben ist hier nicht möglich, denn sollte w nicht von A_1 akzeptiert werden, so könnte A_1 unendlich lange laufen. Wir wären dann nie in der Lage A_2 zu starten (wüssten aber auch nie, ob A_1 unendlich lange läuft und nicht akzeptiert oder nur noch ein bisschen mehr Zeit braucht und dann doch noch akzeptiert). Hier hilft nun ein anderer Trick: Eine TM A arbeitet wie folgt: Auf einem Band simuliert sie A_1 bei Eingabe w, auf einem anderen Band A_2 bei Eingabe w. Nun simuliert sie aber *immer abwechselnd* einen Schritt von A_1, dann einen Schritt von A_2, dann wieder einen Schritt von A_1 usw. Akzeptiert nun A_1 oder A_2 die Eingabe w, so tun sie dies nach endlich vielen Schritten. A bemerkt dies, stoppt beide Simulationen und akzeptiert. Akzeptieren beide nicht, so akzeptiert auch A nicht. Läuft dabei mindestens eine unendlich lange, so tut dies auch A. Dies ist aber nicht schlimm, da A ja $L_1 \cup L_2$ nur akzeptieren soll und nicht entscheiden! So konstruiert akzeptiert A gerade die Worte, die in L_1 oder L_2 sind, hält aber nicht zwingend auf allen Eingaben. Es gilt also $L(A) = L_1 \cup L_2$ und $L_1 \cup L_2$ ist aufzählbar.
3. Nein. Wir wissen, dass die Familie der entscheidbaren Sprachen eine echte Teilmenge der Familie der aufzählbaren Sprachen ist (z.B. ist das Halteproblem aufzählbar, aber nicht entscheidbar). Zudem wissen wir, dass, wenn eine Sprache L und ihre Komplement aufzählbar ist, die Sprache L bereits entscheidbar ist (wir haben das in der Vorlesung gemacht, die Idee ist im Grunde genommen die gleiche wie die in der vorherigen Teilaufgabe). Wären die aufzählbaren Sprachen nun gegenüber Komplementbildung abgeschlossen, so würde aus letzterem folgen, dass alle aufzählbaren Sprachen entscheidbar wären. Damit wären die Sprachfamilien gleich. Dies steht aber im Widerspruch zu ersterem, dass es Sprachen gibt, die aufzählbar, aber nicht entscheidbar sind.

Übungsaufgabe 6.6: Sei

von 4

$$L := \{\langle M, w, n\rangle \mid \text{ die NTM } M \text{ hat auf } w \text{ mindestens } n \text{ Erfolgsrechnungen}\}.$$

Zeigen Sie, dass L nicht entscheidbar ist.

Lösung: Wir nehmen an, dass L entscheidbar ist und entscheiden damit das Halteproblem. Da wir wissen, dass das Halteproblem nicht entscheidbar ist, haben wir den gewünschten Widerspruch und können schlussfolgern, dass L auch nicht entscheidbar ist.

Sei hierzu M_L eine TM, die L entscheidet. Es gilt also $L(M_L) = L$ und M_L hält auf jeder Eingabe. Sei ferner H das Halteproblem also $H = \{\langle M, w\rangle \mid M \text{ hält auf } w\}$. Da H aufzählbar ist, gibt es eine TM M_H mit $L(M_H) = H$ (bspw. kann M_H bei Eingabe $\langle M, w\rangle$ einfach M auf w simulieren und sollte M anhalten 1 ausgeben).

Wir konstruieren nun eine TM M', die das Halteproblem *entscheidet* wie folgt: Bei Eingabe von $\langle M, w\rangle$ konstruiert M' das Tupel $\langle M_H, \langle M, w\rangle, 1\rangle$ und legt dies M_L vor. M' antworte dann genauso wie M_L. Da M_L stets hält, tut dies auch M'. Wir behaupten nun, dass $L(M') = H$ gilt, womit (mit der Aussage zum Halten von eben) M' das Halteproblem entscheiden würde.

Sei $\langle M, w\rangle \in L(M')$, so antwortet M' also positiv, woraus folgt, dass M_L auch positiv geantwortet hat, dass also $\langle M_H, \langle M, w\rangle, 1\rangle \in L$ gilt. Dies wiederum bedeutet, dass (mindestens) eine Rechnung existiert mit der M_H das Wort $\langle M, w\rangle$ akzeptiert, was bedeutet, dass M auf w anhält. Folglich gilt auch $\langle M, w\rangle \in H$.

Ist andersherum $\langle M, w\rangle \in H$, so gibt es (mindestens) eine akzeptierende Rechnung von M_H auf $\langle M, w\rangle$, da M_H ja die Sprache H akzeptiert. Damit ist dann aber $\langle M_H, \langle M, w\rangle, 1\rangle \in L$ und wird somit insb. von M_L akzeptiert. Dies wiederum bedeutet, dass M' nach Konstruktion $\langle M, w\rangle$ akzeptiert und damit ist auch $\langle M, w\rangle \in L(M')$ und wir sind fertig.

Informationen und Unterlagen zur Veranstaltung unter:

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