ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
https://courses.ite.inf.ethz.ch/theo_inf_21

Lösungsvorschläge – Blatt 8

Zürich, 26. November 2021

Lösung zu Aufgabe 22

(a) Um $L_{\mathrm{H}}^{\complement} \leq_{\mathrm{EE}} L_{\mathrm{all}}$ zu beweisen, geben wir einen Algorithmus F an, der eine Eingabe x für $L_{\mathrm{H}}^{\complement}$ zu einer Eingabe $f(x)$ für L_{all} transformiert. Zunächst entscheidet F, ob x von der Form $\mathrm{Kod}(M)\#w$ für eine Turingmaschine M und ein Wort $w \in \{0,1\}^*$ ist. Falls nicht, dann gibt F die Kodierung einer Turingmaschine M_{all} aus, die jede Eingabe direkt akzeptiert, also gilt $f(x) = \mathrm{Kod}(M_{\mathrm{all}})$. Ansonsten berechnet F die Kodierung einer Turingmaschine M', die sich wie folgt verhält. Auf der Eingabe y simuliert M' die Maschine M auf w für $|y|$ Schritte. (Falls die Simulation nach weniger Schritten in den akzeptierenden oder verwerfenden Zustand führt, natürlich entsprechend weniger.) Falls M auf w in dieser Simulation von maximal $|y|$ Schritten terminiert, dann verwirft M' die Eingabe y. Andernfalls akzeptiert M' die Eingabe y.

Die Ausgabe von F ist $f(x) = \mathrm{Kod}(M')$.

Wir beweisen nun die Korrektheit dieser Reduktion, also $x \in L_{\mathrm{H}}^{\complement} \iff f(x) \in L_{\mathrm{all}}$ für alle $x \in \{0,1,\#\}^*$.

Nehmen wir zunächst an, dass $x \in L_{\mathrm{H}}^{\complement}$ gilt und unterscheiden zwei Teilfälle. Falls x nicht von der Form $\mathrm{Kod}(M)\#w$ für eine Turingmaschine M und ein Wort $w \in \{0,1\}^*$ ist, dann haben wir $f(x) = \mathrm{Kod}(M_{\mathrm{all}})$ und somit $f(x) \in L_{\mathrm{all}}$. Falls hingegen $x = \mathrm{Kod}(M)\#w$ für eine Turingmaschine M und ein Wort $w \in \{0,1\}^*$, dann hält M nicht auf w, weil $x \in L_{\mathrm{H}}^{\complement}$ gilt. Dies impliziert wiederum, dass M' jede Eingabe $y \in \Sigma^*$ akzeptiert, da die Simulation von M auf w nie endet, insbesondere nicht innerhalb von $|y|$ Schritten. Es folgt also, dass $f(x) \in L_{\mathrm{all}}$ gilt.

Nehmen wir nun an, dass $x \notin L_{\mathrm{H}}^{\complement}$, also $x \in L_{\mathrm{H}}$. Entsprechend ist x von der Form $\mathrm{Kod}(M)\#w$ für eine Turingmaschine M und ein Wort $w \in \{0,1\}^*$ und M hält auf w nach einer gewissen Anzahl k von Schritten. Sei y ein beliebiges Wort in Σ^k. Dann wird M' auf der Eingabe y eine Simulation von M auf w für $|y| = k$ Schritte durchführen, in der M terminiert. Entsprechend verwirft M' die Eingabe y, also gilt $y \notin L(M')$ und somit $f(x) = \mathrm{Kod}(M') \notin L_{\mathrm{all}}$.

(b) Es gilt

$$(L_{\text{infinite}})^{\complement} = \{w \in \{0,1\}^* \mid w \neq \text{Kod}(M') \text{ für alle TM } M'\} \\ \cup \{\text{Kod}(M) \mid \text{es gibt eine Eingabe } x, \text{ so dass } M \text{ auf } x \text{ hält}\}\,.$$

Wir beschreiben im Folgenden eine nichtdeterministische Turingmaschine N mit $L(N) = (L_{\text{infinite}})^{\complement}$. Es gibt also für jedes Wort aus $w \in (L_{\text{infinite}})^{\complement}$ eine endliche, akzeptierende Berechnung von N auf w. Die NTM N überprüft für eine Eingabe w zunächst, ob $w = \text{Kod}(M)$ für irgendeine TM M. Falls nicht, dann akzeptiert N die Eingabe w. Falls $w = \text{Kod}(M)$ gilt für eine TM M, dann wählt N nichtdeterministisch ein Wort x über dem Eingabealphabet von M und simuliert M deterministisch auf x.

Falls M auf dem Wort x hält, dann akzeptiert N ihre Eingabe w. Falls M auf x unendlich lange rechnet, dann ist auch die Berechnung von N auf w unendlich.

Offenbar akzeptiert N alle Wörter, die nicht die Kodierung einer Turingmaschine sind. Falls die Eingabe w die Kodierung einer TM M ist, dann ist $w \in (L_{\text{infinite}})^{\complement}$ genau dann, wenn es ein Wort x gibt, so dass M auf x hält. Also gibt es dann eine akzeptierende Berechnung von N auf w, in der N genau dieses Wort x nichtdeterministisch wählt. Falls $w \in L_{\text{infinite}}$ gilt, dann gibt es keine akzeptierende Berechnung von N auf w. Also ist N eine NTM, die $(L_{\text{infinite}})^{\complement}$ akzeptiert, damit gilt $(L_{\text{infinite}})^{\complement} \in \mathcal{L}_{\text{RE}}$.

Lösung zu Aufgabe 23

Die Idee hinter der Konstruktion von A ist es, jeweils 12 Schritte von M in 6 Schritten von A zu simulieren. Dafür werden jeweils 12 Felder des Arbeitsbandes von M zu einem Feld von A zusammengefasst. Die gleiche Komprimierung wird auch auf die Eingabe angewendet, hierfür nutzt A ihr zweites Arbeitsband. Wir bemerken, dass A eine konstante Anzahl von Berechnungsschritten von M in einem Schritt simulieren kann, wenn sie die von M in diesen Schritten gelesenen Symbole vorher in ihren Zuständen gespeichert hat. Wir bezahlen also für die Verkürzung der Laufzeit mit einer erheblichen Vergrösserung des Arbeitsalphabets und der Zustandsmenge.

Die MTM A hat auf ihrem Eingabeband dieselbe Eingabe wie M. Um die Berechnung zu verkürzen, komprimiert A zunächst diese Eingabe auf ihrem zweiten Arbeitsband. Hierfür liest sie jeweils 12 Felder des Eingabebandes und schreibt das 12-Tupel der gelesenen Symbole auf ein Feld des zweiten Arbeitsbandes. Dies benötigt auf einer Eingabe der Länge n offenbar $n+1$ Schritte, weil der Kopf auf dem Eingabeband im $(n+1)$-ten Schritt die rechte Endmarkierung \$ erreicht. Danach bewegt A ihren Kopf auf dem zweiten Arbeitsband zurück an den Anfang. Weil auf diesem Arbeitsband jetzt $\lceil n/12 \rceil$ Felder belegt sind, benötigt dies $\lceil n/12 \rceil$ Schritte. Insgesamt braucht A also für dieses Preprocessing $\frac{13n}{12} + c_1$ Schritte, für eine kleine Konstante c_1, z.B. $c_1 = 2$.

Für die eigentliche Simulation von M arbeitet A in *Runden* von jeweils bis zu 6 Berechnungsschritten. In jeder Runde simuliert A dabei 12 Berechnungsschritte von M. Damit ergibt sich eine Laufzeit für die Simulation von höchstens

$$\left\lceil \frac{\text{Time}_M(n)}{12} \right\rceil \cdot 6 \leq \frac{\text{Time}_M(n)}{2} + c_2\,,$$

für eine kleine Konstante c_2, z.B. $c_2 = 6$.

Zu Beginn einer jeden Runde liest A die Inhalte ihrer beiden Arbeitsbänder auf dem aktuellen Feld und den beiden Nachbarfeldern links und rechts und speichert diese jeweils

$3 \cdot 12 = 36$ Felder des Eingabebandes und des Arbeitsbandes von M in ihren Zuständen. Hierfür sind 4 Schritte nötig: Ein Schritt nach links, zwei Schritte nach rechts und ein Schritt zurück auf die ursprüngliche Position. Damit hat A jetzt genügend Informationen, um 12 Schritte von M in ihren Zuständen zu simulieren, denn M kann sich in 12 Schritten um höchstens 12 Positionen bewegen und A kennt nun mindestens 12 Felder links und 12 Felder rechts von der aktuellen Position von M. In den 12 simulierten Schritten kann M nur Inhalte in zwei der drei von A gelesenen Zwölferblöcken ändern. Diese Änderungen kann A nun in höchstens 2 Schritten auf ihren Bändern umsetzen: Wir beschreiben nur die Änderung des ersten Arbeitsbandes, die Änderung der Kopfposition auf dem simulierten Eingabeband kann dann analog umgesetzt werden.

Im fünften Schritt der Runde ändert A den Inhalt des aktuellen Arbeitsbandfeldes entsprechend der 12 Berechnungsschritte von M und bewegt den Kopf nach links oder rechts, falls auch in dem dort simulierten Bereich des Bandes von M Änderungen nötig sind. Im sechsten Schritt führt A die Änderungen auf dem Nachbarfeld durch und kehrt gegebenenfalls zurück. Wenn auf beiden Arbeitsbändern von A nur auf der aktuellen Position Änderungen nötig sind, dann wird der sechste Schritt nicht benötigt.

Insgesamt ergibt sich also für die Zeit, die A für die gesamte Berechnung benötigt, die behauptete obere Schranke.

Lösung zu Aufgabe 24

Die Grammatik $G = (\{S, A, X\}, \{0, 1, 2\}, P, S)$ mit

$$P = \{S \to AS2, S \to X, AX \to 0X1, A0 \to 0A, X \to \lambda\}$$

erzeugt die Sprache L. Sie arbeitet nach der folgenden Idee: Mit der Regel $S \to AS2$ werden zunächst gleich viele As und Zweien erzeugt. Die As dienen hier als Platzhalter für Nullen und Einsen. Mit der Regel $AX \to 0X1$ wird ein A in eine Null und eine Eins umgewandelt. Die Regel $A0 \to 0A$ dient nun dazu, die erzeugte Null über alle As hinweg nach links zu schieben. Erst wenn die erzeugte Null mindestens einmal nach links verschoben wurde, ist die Regel $AX \to 0X1$ wieder anwendbar. Falls die Regel $X \to \lambda$ angewendet wird, obwohl noch As vorhanden sind, dann lässt sich kein Terminalwort erzeugen, dies ist also keine gültige Ableitung. Mit $S \Rightarrow X \Rightarrow \lambda$ lässt sich auch das leere Wort ableiten. Eine Ableitung für das Wort 000111222 kann wie folgt aussehen:

$$\begin{aligned} S &\Rightarrow AS2 \Rightarrow AAS22 \Rightarrow AAAS222 \Rightarrow AAAX222 \\ &\Rightarrow AA0X1222 \Rightarrow A0AX1222 \Rightarrow 0AAX1222 \Rightarrow 0A0X11222 \\ &\Rightarrow 00AX11222 \Rightarrow 000X111222 \Rightarrow 000111222\,. \end{aligned}$$