Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Übungsblatt 7

Vorlesung Theoretische Grundlagen der Informatik im WS 20/21

Ausgabe: 5. Februar 2021
Abgabe: 16. Februar 2021, 11:30 Uhr (digital im ILIAS)

Aufgabe 1

(3 + 1 + 1 = 5 Punkte)

Sei $G = (\Sigma, V, S, R)$ mit $\Sigma = \{\mathsf{a}, \mathsf{b}\}$ und $V = \{S, A, B, C, D, E\}$ die durch folgende Regelmenge gegebene Grammatik:

$$\begin{aligned} S &\to \mathsf{b}E \mid EA \\ A &\to CB \mid A\mathsf{a} \\ B &\to S \mid \mathsf{b} \\ C &\to CA \mid AD \\ D &\to D\mathsf{a} \mid \varepsilon \\ E &\to \mathsf{b}B \mid \varepsilon \end{aligned}$$

(a) Identifizieren Sie alle nutzlosen Variablen in G mit dem Verfahren aus der Vorlesung. Geben Sie die Grammatik G' an, die durch Entfernen der nutzlosen Variablen entsteht.

(b) Welche Sprache erzeugt G? Welchen Chomsky-Typ hat $L(G)$? Ist $L(G)$ endlich?

(c) In der Vorlesung wurde ein Verfahren vorgestellt, um zu entscheiden, ob eine gegebene kontextfreie Grammatik endlich ist. Dabei werden zunächst alle nutzlosen Variablen entfernt. Warum ist das nötig? Geben Sie ein Beispiel an, wo das Verfahren ein falsches Ergebnis liefert, wenn die nutzlosen Variablen vorher nicht entfernt werden.

Lösung:

(a) **Schritt 1:** Berechne die Menge V' der Variablen, die ein Wort erzeugen können.

I. Initialisiere $Q = V' = \{B, D, E\}$.
II. Entnehme B aus Q. Ersetze in allen Regeln B durch b:

$$\begin{aligned} S &\to \mathsf{b}E \mid EA \\ A &\to C\mathsf{b} \mid A\mathsf{a} \\ B &\to S \mid \mathsf{b} \\ C &\to CA \mid AD \\ D &\to D\mathsf{a} \mid \varepsilon \\ E &\to \mathsf{bb} \mid \varepsilon \end{aligned}$$

Nun ist $Q = \{D, E\}$.

III. Entnehme D aus Q. Ersetze in allen Regeln D durch ε:

$$\begin{aligned} S &\to \mathsf{b}E \mid EA \\ A &\to C\mathsf{b} \mid A\mathsf{a} \\ B &\to S \mid \mathsf{b} \\ C &\to CA \mid A \\ D &\to \mathsf{a} \mid \varepsilon \\ E &\to \mathsf{bb} \mid \varepsilon \end{aligned}$$

Nun ist $Q = \{E\}$.

IV. Entnehme E aus Q. Ersetze in allen Regeln E durch ε:

$$\begin{aligned} S &\to \mathsf{b} \mid A \\ A &\to C\mathsf{b} \mid A\mathsf{a} \\ B &\to S \mid \mathsf{b} \\ C &\to CA \mid A \\ D &\to \mathsf{a} \mid \varepsilon \\ E &\to \mathsf{bb} \mid \varepsilon \end{aligned}$$

Nun ist $V' = \{S, B, D, E\}$ und $Q = \{S\}$.

V. Entnehme S aus Q. Ersetze in allen Regeln S durch b:

$$\begin{aligned} S &\to \mathsf{b} \mid A \\ A &\to C\mathsf{b} \mid A\mathsf{a} \\ B &\to \mathsf{b} \\ C &\to CA \mid A \\ D &\to \mathsf{a} \mid \varepsilon \\ E &\to \mathsf{bb} \mid \varepsilon \end{aligned}$$

Nun ist Q leer und Schritt 1 endet mit $V' = \{S, B, D, E\}$.

A und C sind nutzlos und können entfernt werden:

$$\begin{aligned} S &\to \mathsf{b}E \\ B &\to S \mid \mathsf{b} \\ D &\to D\mathsf{a} \mid \varepsilon \\ E &\to \mathsf{b}B \mid \varepsilon \end{aligned}$$

Schritt 2: Berechne die Menge V'' der Variablen, die von S erreichbar sind.

I. Initialisiere $V'' = \{S\}$.

II. Über S lässt sich E erreichen: $V'' = \{S, E\}$.

III. Über E lässt sich B erreichen: $V'' = \{S, B, E\}$.

IV. Über B lassen sich keine neuen Variablen erreichen. D ist also nutzlos.

Durch Entfernen der nutzlosen Variablen entsteht die Grammatik $G' = (\Sigma, V', S, R')$ mit $V' = \{S, B, E\}$ und Regelmenge R':

$$\begin{aligned} S &\to \mathtt{b}E \\ B &\to S \mid \mathtt{b} \\ E &\to \mathtt{b}B \mid \varepsilon \end{aligned}$$

(b) $L(G) = \mathtt{b}(\mathtt{bb})^*$. Diese Sprache hat Chomsky-Typ 3 und ist nicht endlich.

(c) Betrachte die Grammatik $G = (\Sigma, V, S, R)$ mit $V = \{S, A\}$ und Regelmenge R:

$$\begin{aligned} S &\to \varepsilon \\ A &\to AA \end{aligned}$$

Die nutzlose Variable A erzeugt einen Zyklus, aber $L(G) = \{\varepsilon\}$ ist endlich.

Aufgabe 2 (2 + 2 + 1 = 5 Punkte)

In dieser Aufgabe betrachten wir Kellerautomaten mit Reset-Zustand (RPDA). Diese arbeiten wie normale nichtdeterministische Kellerautomaten (NPDA), verfügen aber zusätzlich über einen *Reset-Zustand* $r \in Q$. Wenn ein RPDA in den Zustand r übergeht, wird der Stackinhalt zurückgesetzt (d.h. der Stack wird geleert und das Initialsymbol Z_0 auf den Stack gelegt) und der Lesekopf springt zurück zum ersten Zeichen der Eingabe. Der RPDA bleibt dabei im Zustand r. Sie können in dieser Aufgabe davon ausgehen, dass sowohl bei NPDAs als auch bei RPDAs das Ende der Eingabe mit einem speziellen Symbol markiert ist.

(a) Beschreiben Sie, wie die Sprache $L = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \in \mathbb{N}\}$ von einem RPDA erkannt werden kann.

(b) Seien zwei beliebige kontextfreie Sprachen L_1 und L_2 gegeben. Beschreiben Sie, wie die Sprache $L_1 \cap L_2$ von einem RPDA erkannt werden kann.

(c) Wie kann das Maschinenmodell erweitert werden, sodass der Schnitt von beliebig (aber endlich) vielen kontextfreien Sprachen erkannt werden kann?

Die Grundstruktur eines Kellerautomaten soll dabei erhalten bleiben. Lösungen der Art „Baue den Kellerautomaten zu einer Turing-Maschine um“ werden nicht akzeptiert.

Lösung:

(a) Prüfe zunächst, dass die Eingabe mit einem Präfix der Form $\mathtt{a}^n\mathtt{b}^n\mathtt{c}$ beginnt. Schreibe für jedes gelesene $\mathtt{a}$ ein X auf den Stack. Sobald das erste $\mathtt{b}$ gelesen wird, wechsle in einen neuen Zustand und entferne für jedes gelesene $\mathtt{b}$ ein X vom Stack. Der Stack sollte mit dem Lesen des letzten $\mathtt{b}$ leer werden und darauf sollte ein $\mathtt{c}$ folgen. Ist das nicht der Fall, wechsle in einen Müllzustand, der noch den Rest der Eingabe liest und dann ablehnt. Ansonsten gehe in den Reset-Zustand über. Laufe nun bis zum ersten $\mathtt{b}$ und prüfe dann, ob der Rest der Eingabe die Form $\mathtt{b}^n\mathtt{c}^n$ hat. Dies geschieht analog zum Prüfen von $\mathtt{a}^n\mathtt{b}^n\mathtt{c}$.

(b) Da L_1 eine kontextfreie Sprache ist, gibt es einen NPDA, der L_1 mit akzeptierendem Endzustand erkennt. Simuliere diesen NPDA. Wenn er das Wortende erreicht und sich in einem akzeptierenden Zustand befindet, gehe in den Reset-Zustand über. Wechsle nun mit einem ε-Übergang in den Startzustand des NPDA für L_2 und simuliere diesen.

(c) Statt einem Reset-Zustand $r \in Q$ gibt es eine Menge $R \in Q$ von Reset-Zuständen. Ein solcher Automat kann den Schnitt von bis zu $|R| + 1$ kontextfreien Sprachen erkennen.

Aufgabe 3 (3 + 1 = 4 Punkte)

Sei L eine kontextfreie Sprache. Die Funktionen α und β sind wie folgt definiert.

$$\alpha(L) = \{yx \mid xy \in L\} \tag{1}$$
$$\beta(L) = \{yxz \mid xyz \in L\} \tag{2}$$

Zeigen Sie:

- Die Menge der kontextfreien Sprachen ist abgeschlossen unter α.

  Lösung:
  Sei L eine kontextfreie Sprache. Dann existiert ein NPDA A, der L durch leeren Stack erkennt. Betrachte die Abarbeitung eines Wortes xy durch A. Vor dem Lesen des ersten Zeichens von y sei A in einem Zustand q und der Stackinhalt sei $Z_1, Z_2, \ldots, Z_m$ (es wurde also zuerst Z_1 und zuletzt Z_m auf den Stack gelegt).

  Konstruiere wie folgt einen NPDA B, der $\alpha(L)$ erkennt. Die Idee ist, A auf y zu simulieren und sich dabei zu merken, welcher durch die Abarbeitung von x erzeugte Stackinhalt und Zustand zu einer erfolgreichen Abarbeitung von y führen würde. Bei der Abarbeitung des Wortes yz durch B raten wir zunächst durch ε-Übergänge den Zustand q und speichern diese Wahl. Es kann nun sein, dass A bei der weiteren Abarbeitung von Zeichen aus y die Symbole $Z_1, Z_2, \ldots, Z_m$ vom Stack entfernen würde. Diese Symbole liegen aber nicht auf dem Stack von B. Würde A ein Symbol Z_i vom Stack entfernen, so merkt sich B dies, indem stattdessen ein Symbol Z'_i auf den Stack gelegt wird. So wird B nach Abarbeitung von y die Symbole $Z'_m, Z'_{m-1}, \ldots, Z'_1$ auf den Stack gelegt haben. Errate mit dem Nichtdeterminismus den Übergang von y zu x und gehe in den Startzustand über. Arbeite nun x ab. Wenn A durch Abarbeiten von x in den Zustand q gelangt und den Stackinhalt $Z_1, Z_2, \ldots, Z_m$ erzeugt, so wird das Zeichen Z_1 zuerst auf den Stack gelegt. Bei der Abarbeitung von x durch B wird nicht Z_1 auf den Stack gelegt, sondern überprüft, ob Z'_1 das oberste Symbol auf dem Stack ist. Falls ja, wird es entfernt. Falls nein, so passt die Abarbeitung von x nicht zu der von y und das Wort wird abgelehnt. So wird sukzessive der gesamte Stackinhalt $Z'_m, Z'_{m-1}, \ldots, Z'_1$ abgearbeitet. Am Ende muss man überprüfen, ob der Stack leer ist und ob man jetzt tatsächlich im Zustand q, den man sich zu diesem Zweck gemerkt hatte, angelangt ist.

- Die Menge der kontextfreien Sprachen ist nicht abgeschlossen unter B.

  Hinweis: Sie dürfen das Ergebnis aus Aufgabe 4 nutzen.

  Lösung:
  Betrachte die Sprache $L_1 = \{a^n b^n c^m d^m \mid n, m \geq 1\}$, welche kontextfrei ist. Wir führen die Annahme, dass $\beta(L_1)$ kontextfrei sei zum Widerspruch. Aufgrund der vielen verschiedenen Fälle, wie xyz ein Wort aus L_1 partitionieren kann, ist die Sprache $\beta(L_1)$ nicht ohne Weiteres geschlossen anzugeben. Wir vermeiden dieses Problem wie folgt. Man kann leicht zeigen, dass die Sprache $L_2 = \{b^w c^x a^y d^z \mid w, x, y, z \geq 1\}$ regulär ist. Nach Aufgabe 4 wäre dann auch die

Sprache $L_3 = \beta(L_1) \cap L_2 = \{b^n c^m a^n d^m \mid n, m \geq 1\}$ kontextfrei. Mit dem Pumpinglemma für kontextfreie Sprachen kann aber zeigen, dass L_3 nicht kontextfrei ist (wähle ein Wort so, dass der gepumpte Teil nicht sowohl b als auch a, bzw. sowohl c als auch d enthalten kann). Damit kann $\beta(L_1)$ also nicht kontextfrei sein.

Aufgabe 4 (3 Punkte)

Sei L_1 eine kontextfreie Sprache und sei L_2 eine reguläre Sprache. Zeigen Sie, dass $L_1 \cap L_2$ eine kontextfreie Sprache ist.

Lösung:
Weil L_1 kontextfrei ist, existiert ein NPDA A_1, der L_1 erkennt. Weil L_2 regulär ist, existiert ein DEA A_2, der L_2 erkennt. Schneide den „Automatenteil“ von A_1 mit A_2 (Produktautomat). Offenbar akzeptiert dieser neue NPDA ein Wort genau dann, wenn es sowohl von A_1 als auch A_2 erkannt wird. Also erkennt der neue NPDA gerade die Sprache $L_1 \cap L_2$.

Aufgabe 5 ($1 + 2 + 1 = 4$ Punkte)

Wir betrachten die Buchstaben $7 \times B, 5 \times A, 4 \times L, 2 \times E, 1 \times T$.

(a) Geben Sie einen Huffman-Code für die gegebene Verteilung an. Verwenden Sie die `0`, wenn Sie zu einem häufigeren Buchstaben absteigen und die `1`, wenn Sie zu einem weniger häufigen Buchstaben absteigen.

(b) Wir ziehen mit Zurücklegen fünf Buchstaben und kodieren das so entstandene Wort mit der Huffman-Codierung. Wie lang ist die Codierung erwartet?

(c) Decodieren Sie das Wort `111110110`. Ist das Ergebnis eindeutig?

Lösung:

(a) $B \mapsto \texttt{0}, A \mapsto \texttt{10}, L \mapsto \texttt{110}, E \mapsto \texttt{1110}, T \mapsto \texttt{1111}$

(Falls Teilbaum mit höherer Gesamthäufigkeit statt zu den häufigeren Einzelbuchstaben gewählt wurde: $B \mapsto \texttt{1}, A \mapsto \texttt{01}, L \mapsto \texttt{000}, E \mapsto \texttt{0010}, T \mapsto \texttt{0011}$. Dann ist die (c) nicht decodierbar.)

oder

$B \mapsto \texttt{00}, A \mapsto \texttt{01}, L \mapsto \texttt{10}, E \mapsto \texttt{110}, T \mapsto \texttt{111}$

(b) $(\frac{7}{19} \cdot 1 + \frac{5}{19} \cdot 2 + \frac{4}{19} \cdot 3 + \frac{2}{19} \cdot 4 + \frac{1}{19} \cdot 4) \cdot 5 \approx 10,8$

(c) Tal oder Tee, je nach gewählter Codierung. Nicht eindeutig.

Aufgabe 6 ($1 + 6 = 7$ Punkte)

Für eine Menge U von Variablen bezeichnen wir eine Klauselmenge $\mathcal{C}$ als

- 3-SAT-*Klauselmenge*, wenn jeden Klausel höchstens drei Literale enthält,
- UNIQUE-3-SAT-*Klauselmenge*, wenn zusätzlich jede Variable höchstens ein Mal in $\mathcal{C}$ vorkommt (insbesondere kommt nur eines von u und $\bar{u}$ vor und keine Klausel enthält eine Variable doppelt).
- 2-SAT-*Klauselmenge*, wenn jede Klausel höchstens zwei Literale enthält.

Daraus ergeben sich folgende SAT-Varianten:

Gegeben: Menge U von Variablen und Klauselmenge über U wie folgt:

- für 2-SAT: eine 2-SAT-Klauselmenge
- für 3-SAT: eine 3-SAT-Klauselmenge
- für UNIQUE-3-SAT: eine UNIQUE-3-SAT-Klauselmenge
- für UNIQUE-3-2-SAT: eine UNIQUE-3-SAT-Klauselmenge und eine 2-SAT-Klauselmenge

Frage: Existiert eine erfüllende Wahrheitsbelegung?

Von 3-SAT wissen Sie bereits, dass das Problem $\mathcal{NP}$-vollständig ist. Ebenso wissen Sie, wie man 2-SAT in Polynomialzeit lösen kann.

(a) Zeigen Sie, dass UNIQUE-3-SAT in $\mathcal{P}$ liegt.

(b) Zeigen Sie, dass UNIQUE-3-2-SAT $\mathcal{NP}$-vollständig ist. Geben Sie dabei explizit an, welches Problem Sie auf welches reduzieren. Verwenden Sie für die Reduktion das Problem 3-SAT.

Lösung:

(a) Für jede Variable $u \in U$: Setze u auf `wahr`, falls das bejahte Literal u in einer Klausel vorkommt und auf `falsch` sonst. Da nur eines von u und $\bar{u}$ vorkommt, ist die Variablenbelegung wohldefiniert und alle Klauseln enthalten nur Literale, die zu `wahr` ausgewertet werden. Folglich hat das Problem UNIQUE-3-SAT keine Nein-Instanzen. Es kann in konstanter Zeit „Ja“ ausgegeben werden bzw. in linearer Zeit eine erfüllende Belegung.

(b) UNIQUE-3-2-SAT ist ein Spezialfall von 3-SAT, also auch in $\mathcal{NP}$.

Wir geben eine polynomielle Reduktion von 3-SAT auf UNIQUE-3-2-SAT an.

Konstruktion. Sei $(U_3, \mathcal{C}_3)$ eine 3-SAT Instanz. Wir konstruieren eine UNIQUE-3-2-SAT-Instanz $(U, \mathcal{C}, \mathcal{C}')$, die genau dann erfüllbar ist, wenn $(U_3, \mathcal{C}_3)$ erfüllbar ist. Hierbei bezeichnet $\mathcal{C}$ die UNIQUE-3-SAT-Klauselmenge und $\mathcal{C}'$ die 2-SAT-Klauselmenge. Wir ersetzen jedes Vorkommen einer Variable in $\mathcal{C}_3$ durch eine neue Variable. Zusätzlich führen wir mit der 2-SAT-Klauselmenge Implikationen ein, die sicherstellen, dass die neuen Variablen alle den gleichen Wahrheitswert erhalten.

Sei $u \in U_3$ eine Variable, die $n \geq 1$ Mal in $\mathcal{C}_3$ vorkommt. Wir verwenden n neue Variablen $u_0, \ldots, u_{n-1} \in U$. Im Folgenden nehmen wir alle Indizes modulo n. Sei $\{(\bar{u}_i \vee u_{i+1}) \colon i \in \{0, \ldots, n-1\}\} \in \mathcal{C}'$. Für jede Klausel $C_3 \in \mathcal{C}_3$, die u enthält, führen wir eine entsprechende Klausel $C \in \mathcal{C}$ ein. Wir erhalten C, indem wir in C_3 das Literal u durch ein u_i bzw. $\bar{u}$ durch ein $\bar{u}_i$ ersetzen (für ein $i \in \{0, \ldots, n-1\}$) und zwar so, dass jede Variable u_i nur ein Mal in $\mathcal{C}$ vorkommt.

Die Instanz $(U, \mathcal{C}, \mathcal{C}')$ kann in Polynomialzeit konstruiert werden, da $|U| \leq 3|\mathcal{C}_3|$, $|\mathcal{C}| = |\mathcal{C}_3|$ und $|\mathcal{C}'| = |U| \leq 3|\mathcal{C}_3|$.

Korrektheit. Wir zeigen, dass $(U, \mathcal{C}, \mathcal{C}')$ genau dann erfüllbar ist, wenn (U_3, C_3) erfüllbar ist.

$\Longleftarrow$ Sei $(U_3, \mathcal{C}_3)$ erfüllbar. Sei $t_3 \colon U_3 \to \{\texttt{wahr}, \texttt{falsch}\}$ eine erfüllende Wahrheitsbelegung für $\mathcal{C}_3$. Wir konstruieren eine Wahrheitsbelegung $t \colon U \to \{\texttt{wahr}, \texttt{falsch}\}$, die alle Klauseln in $\mathcal{C} \cup \mathcal{C}'$ erfüllt. Für jede Variable $u \in U_3$, die n Mal in $\mathcal{C}_3$ vorkommt, setzen wir $t(u_i) = t_3(u)$ für jedes $i \in \{0, \ldots, n-1\}$. Alle Klauseln in $\mathcal{C}'$ sind erfüllt, da jede Klausel ein bejahtes und ein negiertes Literal enthält, das zur gleichen Variable aus U_3 gehört. Alle Klauseln in $\mathcal{C}$ sind erfüllt, da jede Klausel in $\mathcal{C}$ einer Klausel in $\mathcal{C}_3$ entspricht.

$\Longrightarrow$ Sei nun $(U, \mathcal{C}, \mathcal{C}')$ erfüllbar. Sei $t \colon U \to \{\texttt{wahr}, \texttt{falsch}\}$ eine erfüllende Wahrheitsbelegung für $\mathcal{C} \cup \mathcal{C}'$. Sei $u \in U_3$ eine Variable, die n Mal in $\mathcal{C}_3$ vorkommt. Die Klausel $(\bar{u}_i \vee u_{i+1})$ ist äquivalent zur der Implikation $t(u_i) \implies t(u_{i+1})$. Folglich impliziert die Klauselmenge $\{(\bar{u}_i \vee u_{i+1}) \colon i \in \{0, \ldots, n-1\}\}$, dass $t(u_0) \iff \cdots \iff t(u_{n-1})$. Wir konstruieren eine Wahrheitsbelegung $t_3 \colon U_3 \to \{\texttt{wahr}, \texttt{falsch}\}$, die alle Klauseln in $\mathcal{C}_3$ erfüllt. Wir setzen $t_3(u) = t(u_i)$ für ein beliebiges $i \in \{0, \ldots, n-1\}$. Da jede Klausel in $\mathcal{C}_3$ einer Klausel in $\mathcal{C}$ entspricht, sind alle Klauseln in $\mathcal{C}_3$ erfüllt.