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

Übungsblatt 6

Vorlesung Theoretische Grundlagen der Informatik im WS 20/21

Ausgabe: 26. Januar 2020
Abgabe: 9. Februar 2020, 11:30 Uhr (digital im ILIAS)

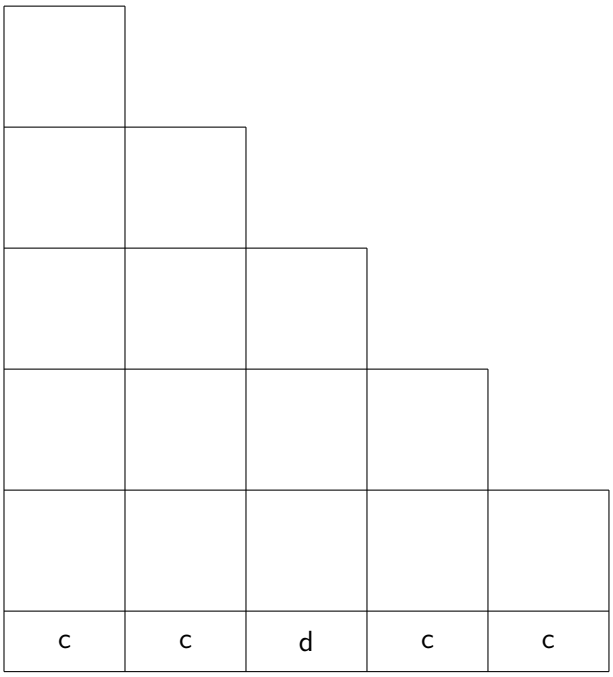
Aufgabe 1

(2 + 1 + 2 + 1 = 6 Punkte)

Gegeben sei die Grammatik $G = (\Sigma, V, S, R)$ mit Terminalen $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$, Nichtterminalen $V = \{S, A, B, C, D\}$, Startsymbol S und Produktionen

$$\begin{aligned} R = \{ & S \to AD \\ & A \to CB \mid \mathsf{a} \mid \mathsf{c} \\ & B \to AD \mid \mathsf{b} \mid \mathsf{d} \\ & C \to DB \mid \mathsf{c} \\ & D \to AC \mid \mathsf{c} \mid \mathsf{d}\} \end{aligned}$$

(a) Überprüfen Sie mit dem CYK-Algorithmus, ob das Wort ccdcc in der Sprache $L(G)$ enthalten ist.



(b) Geben Sie einen Syntaxbaum für das Wort ccdcc an.

(c) Entwerfen Sie einen Algorithmus, der ausgibt, ob ein gegebenes Wort w ein Zeuge dafür ist, dass die Grammatik mehrdeutig (d.h. nicht eindeutig) ist. Falls w ein solcher Zeuge ist, geben Sie zwei Syntaxbäume aus.

Hinweis: Erweitern Sie dazu den CYK-Algorithmus. Berechnen Sie dabei rekursiv für jedes Symbol X in Zelle V_{ij} der Tabelle bis zu zwei Syntaxbäume, die mögliche Ableitungen von X auf das Teilwort w_{ij} repräsentieren.

(d) Ist die Grammatik G eindeutig? Begründen Sie.

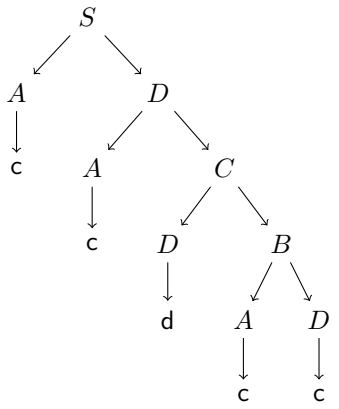
Lösung:

(a) Vergleiche folgende Tabelle:

V				
V	S, A, B, D			
A, C, D	S, B, D	C		
S, B, D	S, A, B, C	$\emptyset$	S, B, D	
A, C, D	A, C, D	B, D	A, C, D	A, C, D
c	c	d	c	c

Das Wort ist also in der Sprache enthalten.

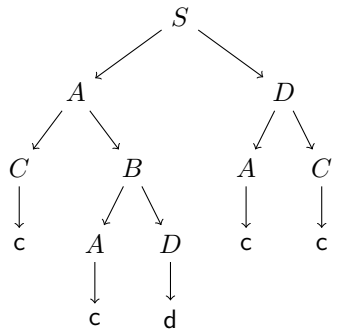
(b) Syntaxbaum für ccdcc:



(c) Für $i = 1$ gibt es genau einen Syntaxbaum; dieser hat die Form $X \to w_j$. Ansonsten lassen sich die möglichen Syntaxbäume von X auf w_{ij} rekursiv aus den bereits berechneten Bäumen

zusammensetzen. Prüfe dafür für alle $i \leq k < j$, $Y \in V_{ik}$ und $Z \in V_{k+1\,j}$, ob die Regel $X \to YZ$ existiert. Falls ja, müssen die bereits berechneten Bäume für $Y \in V_{ik}$ und $Z \in V_{k+1\,j}$ zusammengesetzt werden. Bei y Bäumen für Y und z Bäumen für Z entstehen $y \cdot z$ mögliche Bäume für X. Diese bestehen aus der Wurzel X, an die links der Baum für Y und rechts der Baum für Z angehängt wird. Falls $y \cdot z > 2$, genügt es zwei beliebige Bäume zu speichern. Die Syntaxbäume für w sind dann am Ende die berechneten Bäume für $S \in V_{1n}$.

(d) Die Grammatik ist nicht eindeutig. Zum Beispiel hat das Wort ccdcc noch einen weiteren Syntaxbaum:



Aufgabe 2

(5 + 1 = 6 Punkte)

Sei eine Grammatik G durch $V = \{S, A, B, C, D\}, \Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$ und folgende Regelmenge R gegeben:

$$
\begin{aligned}
S &\to A \mid \mathsf{a}B \mid \mathsf{a}C \\
A &\to B \mid C \mid \mathsf{c}A\mathsf{d} \\
B &\to S \mid B\mathsf{a} \\
C &\to D \mid \mathsf{c} \\
D &\to \mathsf{d} \mid \mathsf{d}DD \mid \varepsilon
\end{aligned}
$$

(a) Verwenden Sie das Verfahren aus der Vorlesung, um G in eine äquivalente Grammatik in erweiterter Chomsky-Normalform zu überführen. Es genügt dabei, wenn sie nach jedem Schritt bzw. jeder Phase das jeweilige Ergebnis angeben.

(b) In Schritt 4, Phase 2 wird beim Ersetzen von Kettenregeln der Form $A \to B$ in umgekehrter topologischer Sortierung vorgegangen. Warum ist dies nötig? Geben Sie eine Grammatik und eine Sortierung der Variablen an, für die das Verfahren nicht zum korrekten Ergebnis führt.

Lösung:

(a) **Schritt 1:** Ersetze gemischte Produktionen mit Terminal- und Nichtterminalsymbolen auf der rechten Seite.

$$
\begin{aligned}
S &\to A \mid Y_{\mathsf{a}}B \mid Y_{\mathsf{a}}C \\
A &\to B \mid C \mid Y_{\mathsf{c}}AY_{\mathsf{d}} \\
B &\to S \mid BY_{\mathsf{a}} \\
C &\to D \mid \mathsf{c} \\
D &\to \mathsf{d} \mid Y_{\mathsf{d}}DD \mid \varepsilon \\
Y_{\mathsf{a}} &\to \mathsf{a} \\
Y_{\mathsf{c}} &\to \mathsf{c} \\
Y_{\mathsf{d}} &\to \mathsf{d}
\end{aligned}
$$

Schritt 2: Ersetze Produktionen mit Länge ≥ 3.

$$
\begin{aligned}
S &\to A \mid Y_{\mathsf{a}}B \mid Y_{\mathsf{a}}C \\
A &\to B \mid C \mid Y_{\mathsf{c}}Z_1 \\
B &\to S \mid BY_{\mathsf{a}} \\
C &\to D \mid \mathsf{c} \\
D &\to \mathsf{d} \mid Y_{\mathsf{d}}Z_2 \mid \varepsilon \\
Y_{\mathsf{a}} &\to \mathsf{a} \\
Y_{\mathsf{c}} &\to \mathsf{c} \\
Y_{\mathsf{d}} &\to \mathsf{d} \\
Z_1 &\to AY_{\mathsf{d}} \\
Z_2 &\to DD
\end{aligned}
$$

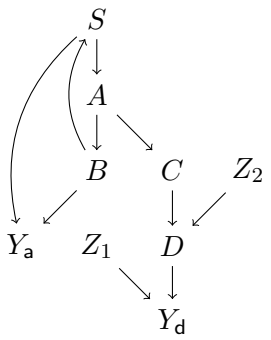
Schritt 3: Ersetze Regeln der Form $A \to \varepsilon$.

Die Menge der Variablen, die sich auf ε ableiten lassen, ist $V' = \{S, A, B, C, D, Z_2\}$. Das folgt aus den Ableitungsketten $B \to S \to A \to C \to D \to \varepsilon$ und $Z_2 \to DD \overset{*}{\to} \varepsilon$. Y_{a}, Y_{c} und Y_{d} sind offensichtlich nicht in V', während sich Z_1 wegen des Vorkommens von Y_{d} auf der rechten Seite nur in Wörter ableiten lässt, die mindestens ein d enthalten.

$$
\begin{aligned}
S &\to A \mid Y_{\mathsf{a}} \mid Y_{\mathsf{a}}B \mid Y_{\mathsf{a}}C \\
A &\to B \mid C \mid Y_{\mathsf{c}}Z_1 \\
B &\to S \mid Y_{\mathsf{a}} \mid BY_{\mathsf{a}} \\
C &\to D \mid \mathsf{c} \\
D &\to \mathsf{d} \mid Y_{\mathsf{d}} \mid Y_{\mathsf{d}}Z_2 \\
Y_{\mathsf{a}} &\to \mathsf{a} \\
Y_{\mathsf{c}} &\to \mathsf{c} \\
Y_{\mathsf{d}} &\to \mathsf{d} \\
Z_1 &\to Y_{\mathsf{d}} \mid AY_{\mathsf{d}} \\
Z_2 &\to D \mid DD
\end{aligned}
$$

Schritt 4: Ersetze Kettenregeln der Form $A \to B$.

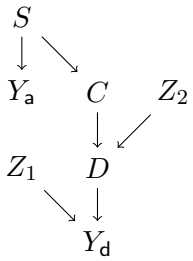
Abhängigkeitsgraph:



Es existiert eine zyklische Abhängigkeit $S \to A \to B \to S$. Wir entfernen also A und B und ersetzen alle Vorkommen davon durch S.

$$
\begin{aligned}
S &\to Y_{\mathsf{a}} \mid Y_{\mathsf{a}}S \mid Y_{\mathsf{a}}C \mid C \mid Y_{\mathsf{c}}Z_1 \mid SY_{\mathsf{a}} \\
C &\to D \mid \mathsf{c} \\
D &\to \mathsf{d} \mid Y_{\mathsf{d}} \mid Y_{\mathsf{d}}Z_2 \\
Y_{\mathsf{a}} &\to \mathsf{a} \\
Y_{\mathsf{c}} &\to \mathsf{c} \\
Y_{\mathsf{d}} &\to \mathsf{d} \\
Z_1 &\to Y_{\mathsf{d}} \mid SY_{\mathsf{d}} \\
Z_2 &\to D \mid DD
\end{aligned}
$$

Es entsteht folgender neuer Abhängigkeitsgraph:



Eine mögliche topologische Sortierung ist $S, Y_{\mathsf{a}}, C, Z_2, Z_1, D, Y_{\mathsf{d}}$.

$$
\begin{aligned}
S &\to \mathsf{a} \mid Y_{\mathsf{a}}S \mid Y_{\mathsf{a}}C \mid \mathsf{d} \mid Y_{\mathsf{d}}Z_2 \mid \mathsf{c} \mid Y_{\mathsf{c}}Z_1 \mid SY_{\mathsf{a}} \\
C &\to \mathsf{d} \mid Y_{\mathsf{d}}Z_2 \mid \mathsf{c} \\
D &\to \mathsf{d} \mid Y_{\mathsf{d}}Z_2 \\
Y_{\mathsf{a}} &\to \mathsf{a} \\
Y_{\mathsf{c}} &\to \mathsf{c} \\
Y_{\mathsf{d}} &\to \mathsf{d} \\
Z_1 &\to \mathsf{d} \mid SY_{\mathsf{d}} \\
Z_2 &\to \mathsf{d} \mid Y_{\mathsf{d}}Z_2 \mid DD
\end{aligned}
$$

Schritt 5: Stelle die Ableitung $S \overset{*}{\to} \varepsilon$ wieder her.

Das Endergebnis ist eine Grammatik G' mit $V' = \{S', S, C, D, Y_{\mathsf{a}}, Y_{\mathsf{c}}, Y_{\mathsf{d}}, Z_1, Z_2\}, \Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$ und Regelmenge R':

$$
\begin{aligned}
S' &\to S \mid \varepsilon \\
S &\to \mathsf{a} \mid Y_{\mathsf{a}}S \mid Y_{\mathsf{a}}C \mid \mathsf{d} \mid Y_{\mathsf{d}}Z_2 \mid \mathsf{c} \mid Y_{\mathsf{c}}Z_1 \mid SY_{\mathsf{a}} \\
C &\to \mathsf{d} \mid Y_{\mathsf{d}}Z_2 \mid \mathsf{c} \\
D &\to \mathsf{d} \mid Y_{\mathsf{d}}Z_2 \\
Y_{\mathsf{a}} &\to \mathsf{a} \\
Y_{\mathsf{c}} &\to \mathsf{c} \\
Y_{\mathsf{d}} &\to \mathsf{d} \\
Z_1 &\to \mathsf{d} \mid SY_{\mathsf{d}} \\
Z_2 &\to \mathsf{d} \mid Y_{\mathsf{d}}Z_2 \mid DD
\end{aligned}
$$

(b) Betrachte z.B. die Grammatik G mit $V = \{A, B, C\}$, $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, Startsymbol A und Regelmenge R:

$$
\begin{aligned}
A &\to B \mid \mathsf{a} \\
B &\to C \mid \mathsf{b} \\
C &\to AB \mid \mathsf{c}
\end{aligned}
$$

Schritt 1 bis Schritt 4, Phase 1 lassen diese Grammatik unverändert. In Schritt 4, Phase 2 ist die einzige mögliche topologische Sortierung A, B, C. Wird beim Ersetzen der Kettenregeln aber z.B. in der Reihenfolge A, B, C vorgegangen, erhalten wir folgendes Ergebnis:

$$
\begin{aligned}
A &\to C \mid \mathsf{b} \mid \mathsf{a} \\
B &\to AB \mid \mathsf{c} \mid \mathsf{b} \\
C &\to AB \mid \mathsf{c}
\end{aligned}
$$

Es gibt danach also immer noch eine Kettenregel $A \to C$, bei der C noch durch AB ersetzt werden müsste. Das wurde aber nicht gemacht, weil A vor C abgearbeitet wurde.

Aufgabe 3

(2 + 2 = 4 Punkte)

Geben Sie für die folgenden Sprachen jeweils eine Grammatik vom angegebenen Typ an. Erläutern Sie die Idee der von Ihnen angegebenen Regeln.

(a) $L = \{a^n b^{2n} c^m \mid n \geq 0, m > 0\}$ – kontextfrei

(b) $L = \{a^n b^{2n} c^{3n} \mid n \geq 0\}$ – kontextsensitiv

Lösung:
$G = (\Sigma, V, S, R)$ mit $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ und $V = \{S, A, B, C, A', B', C', X\}$

(a)

$$
\begin{array}{rll}
R = \{S & \to XC & \text{Aufteilung in unabhängige Teile für } \mathsf{a}^n\mathsf{b}^{2n} \text{ und } \mathsf{c}\text{'s} \\
C & \to \mathsf{c}C \mid \mathsf{c} & \text{Mindestens ein } \mathsf{c} \\
X & \to \mathsf{a}X\mathsf{bb} \mid \varepsilon & \text{Erzeugen von } \mathsf{a}^n\mathsf{b}^{2n}\}
\end{array}
$$

(b)

$$
\begin{array}{rll}
R = \{ \quad S & \to X \mid \varepsilon & n = 0 \\
X & \to XABBCCC \mid A'BBCCC & \text{Erzeugen der Variablen in entsprechendem Verhältnis} \\
CA & \to AC & \text{Sortieren} \\
BA & \to AB & \\
CB & \to BC & \\
A'A & \to A'A' & \text{Überprüfen der Sortierung von links nach rechts,} \\
A'B & \to A'B' & \text{wobei } A', B', C' \text{ nicht mehr beweglich} \\
B'B & \to B'B' & \\
B'C & \to B'C' & \\
C'C & \to C'C' & \\
A' & \to \mathsf{a} & \text{Ersetzen durch Terminale} \\
B' & \to \mathsf{b} & \\
C' & \to \mathsf{c}\} &
\end{array}
$$

Aufgabe 4

(2 + 4 = 6 Punkte)

Zeigen Sie, dass die kontextsensitiven Sprachen den Sprachen der Klasse $\mathsf{NTAPE}(n)$ entsprechen[1], indem Sie in zwei Schritten vorgehen.

(a) Sei L eine kontextsensitive Sprache. Dann existiert eine nichtdeterministische Turing-Maschine $\mathcal{M}$, die L mit linearem Platzbedarf akzeptiert.

(b) Sei $\mathcal{M}$ eine nichtdeterministische Turing-Maschine, die die Sprache $L(\mathcal{M})$ mit linearem Platzbedarf akzeptiert. Dann existiert eine kontextsensitive Grammatik G, so dass $L(G) = L(\mathcal{M})$ gilt.

Lösung:

(a) Sei L eine kontextsensitive Sprache und G eine Typ-1-Grammatik, die L erzeugt. Zeige, dass L von einer nichtdeterministischen Turing-Maschine $\mathcal{M}$ mit linearem Platzbedarf entschieden werden kann. Die Grammatik G hat konstante Größe und lässt sich damit in $\mathcal{O}(1)$ Platz auf das Band von $\mathcal{M}$ schreiben. Sei nun w eine Eingabe für $\mathcal{M}$. Es gilt zu entscheiden, ob $w \in L$ ist. Im Falle $w = \varepsilon$ gilt $w \in L$ genau dann, wenn G die Ableitungsregel $S \to \varepsilon$ enthält. Andernfalls wird wie folgt vorgegangen. Enthält w ein Teilwort v, sodass G eine Regel $u \to v$ enthält, ist $u \to v$ eine *mögliche* Ableitung. Die NTM $\mathcal{M}$ sucht in jedem Schritt erst alle möglichen Ableitungen und wählt danach eine nichtdeterministisch aus. Dann wird v durch u ersetzt. Steht irgendwann nur noch S auf dem Band, hält $\mathcal{M}$ und akzeptiert. Da $\mathcal{M}$ nur Ableitungen vornimmt, die durch G kodiert werden, gilt $w \in L$, falls $\mathcal{M}$ akzeptiert. Umgekehrt akzeptiert $\mathcal{M}$, falls $w \in L$ gilt, z.B. indem die Ableitungen, die von S zu w führen, in umgekehrter Reihenfolge vorgenommen werden. Man beachte, dass die Eingabe beim umgekehrten Anwenden der Ableitungen immer kürzer wird. Zusammen mit der in $\mathcal{O}(1)$ kodierten Grammatik wird so also höchstens linear viel Platz benötigt.

(b) Wir gehen für diese Aufgabe davon aus, dass die NTM statt einem separaten Read-Only-Eingabeband eine separate Eingabe*spur* auf dem Arbeitsband hat. Die NTM hat also nur

[1] Vergleiche Vorlesung 13, Folie 20.

einen Kopf und ein zweispuriges Band, wobei auf Spur 1 die Eingabe steht und der Kopf nur auf Spur 2 schreiben darf.

Sei nun $L \in \mathsf{NTAPE}(n)$ und $\mathcal{M} = (Q, \Sigma, \Sigma \times \Gamma, s, \delta, F)$ eine NTM, die L akzeptiert. Das Bandalphabet besteht aus Tupeln (σ, γ), wobei $\sigma \in \Sigma$ das Eingabesymbol auf Spur 1 ist und $\gamma \in \Gamma$ das „Arbeitssymbol“ auf Spur 2. Übergänge haben die Form $(q, \sigma, \gamma) \mapsto (\hat{q}, \hat{\gamma}, d)$. Dabei ist q der aktuelle Zustand, σ das aktuell gelesene Eingabesymbol, γ das aktuell gelesene Arbeitssymbol. Diese werden abgebildet auf einen neuen Zustand $\hat{q}$, ein neues Arbeitssymbol $\hat{\gamma}$ und eine Kopfbewegung $d \in \{L, N, R\}$.

Wir konstruieren eine Grammatik G, die die Sprache L erzeugt. Das Terminalalphabet von G sei gerade Σ. Die Idee ist es, die Kopfbewegungen und Zustandsübergänge von $\mathcal{M}$ als Ableitungen in G zu kodieren. Dazu kodieren wir die aktuelle Bandkonfiguration als Kette von Nichtterminalsymbolen. Jedes Nichtterminalsymbol ist ein Tripel $(q, \sigma, \gamma) \in (Q \cup \{\bot\}) \times (\Sigma \cup \{\sqcup\}) \times (\Gamma \cup \{\sqcup\})$, das ein Feld auf dem Band repräsentiert. Dabei steht q für den Zustand der Kontrolleinheit bzw. $\bot$, falls der Kopf nicht auf diesem Feld steht. Das Eingabesymbol auf Spur 1 wird durch σ repräsentiert und das Arbeitssymbol auf Spur 2 durch γ.

Durch folgende Regeln lässt sich die Anfangskonfiguration für jedes mögliche Eingabewort erzeugen:

$$\begin{aligned} & S \;\to\; (\bot, \sqcup, \sqcup)\; A\; (\bot, \sqcup, \sqcup) \\ \forall \sigma \in \Sigma: \quad & A \;\to\; (s, \sigma, \sqcup) \mid (s, \sigma, \sqcup)\; B \\ \forall \sigma \in \Sigma: \quad & B \;\to\; BB \mid (\bot, \sigma, \sqcup) \end{aligned}$$

Man beachte, dass der Kopf von $\mathcal{M}$ zunächst im Zustand s auf dem linkesten Symbol der Eingabe steht. Die Eingabe wird links und rechts von je einem leeren Feld der Form $(\bot, \sqcup, \sqcup)$ begrenzt.

Die Idee ist nun, dass sich diese Anfangskonfiguration genau dann zum Eingabewort w ableiten lässt, wenn w von $\mathcal{M}$ akzeptiert wird. Füge deshalb für jeden Übergang $(q, \sigma, \gamma) \mapsto (\hat{q}, \hat{\gamma}, d)$ und alle $\sigma' \in \Sigma$,$\gamma' \in \Gamma$ folgende Regeln hinzu:

$$\begin{aligned} (\bot, \sigma', \gamma')\; (q, \sigma, \gamma) \;&\to\; (\hat{q}, \sigma', \gamma')\; (\bot, \sigma, \hat{\gamma}) && \text{falls } d = L \\ (q, \sigma, \gamma) \;&\to\; (\hat{q}, \sigma, \hat{\gamma}) && \text{falls } d = N \\ (q, \sigma, \gamma)\; (\bot, \sigma', \gamma') \;&\to\; (\bot, \sigma, \hat{\gamma})\; (\hat{q}, \sigma', \gamma') && \text{falls } d = R \end{aligned}$$

Sobald ein akzeptierender Zustand $f \in F$ erreicht ist, muss das ursprüngliche Wort entpackt werden.

$$\begin{aligned} (f, \sigma, \gamma) \;&\to\; \sigma \\ (\bot, \sigma, \gamma)\; \sigma' \;&\to\; \sigma\; \sigma' \\ \sigma'\; (\bot, \sigma, \gamma) \;&\to\; \sigma'\; \sigma \end{aligned}$$

Ist $\varepsilon \in L$, fügen wir außerdem noch die Regel $S \to \varepsilon$ hinzu.

Aufgabe 5

(1 + 2 = 3 Punkte)

Zeigen Sie, dass die Klasse der Typ-1-Sprachen unter folgenden Operationen abgeschlossen ist:

(a) Vereinigung

(b) Schnitt

Lösung:

(a) Seien zwei Typ-1-Grammatiken $G_1 = (\Sigma, V_1, S_1, R_1)$ und $G_2 = (\Sigma, V_2, S_2, R_2)$ gegeben. O.B.d.A. seien $V_1 \cap V_2 = \emptyset$. Die Grammatik $G = (\Sigma, V, S, R)$ mit $V = V_1 \cup V_2 \cup \{S\}$ und $R = R_1 \cup R_2 \cup \{S \to S_1, S \to S_2\}$ erzeugt $L(G_1) \cup L(G_2)$. Falls die Regel $S_1 \to \varepsilon$ oder $S_2 \to \varepsilon$ existiert, muss man diese noch entfernen und durch die Regel $S \to \varepsilon$ ersetzen. Die resultierende Grammatik hat Typ 1.

(b) Gegeben seien zwei Typ-1-Sprachen L_1 und L_2. Dann existieren NTMs $\mathcal{M}_1$ und $\mathcal{M}_2$ für L_1 bzw. L_2, die jeweils bei Eingabelänge n höchstens Platzbedarf n haben. Folgende NTM $\mathcal{M}$ erkennt $L_1 \cap L_2$ mit Platzbedarf n: Simuliere zunächst L_1. Falls L_1 ablehnt, lehne ab. Ansonsten lösche den Bandinhalt und simuliere L_2. Also ist $L_1 \cap L_2 \in \mathsf{NTAPE}(n)$ und hat somit Typ 1.

Aufgabe 6 (4 Punkte)

Zeigen Sie, dass Sie für kontextsensitive Grammatiken ohne Beschränkung der Allgemeinheit fordern können, dass in jeder Ableitungsregel sowohl links als auch rechts höchstens zwei Zeichen stehen.

Lösung:
Wir zeigen zuächst, wie man die linken Seiten aller Ableitungsregeln auf Länge höchstens 2 beschränkt. Betrachte dazu eine Regel $AB\gamma \to XY\delta$ mit γ, δ in V^* und $1 \leq |\gamma| \leq |\delta|$. Entferne diese Regel und führe die neue Variable $\square$ und folgende Ableitungsregeln ein.

$$AB \to X\square$$
$$\square\gamma \to Y\delta$$

So hat man die Regel $AB\gamma \to XY\delta$ mit $k = |AB\delta|$ durch Regeln ersetzt, deren linke Seite höchstens Länge $k-1$ hat. Durch wiederholtes Anwenden dieser Ersetzung haben die linken Seiten aller Regeln Länge höchstens 2.

Um die rechten Seiten zu beschränken gehen wir wie in Schritt 2 der Umformung einer kontextfreien Grammatik in Chomsky-Normalform vor. Betrachte eine Regel $\alpha \to B_1 \dots B_m$. Entferne diese Regel und führe neue Variablen $C_1, \dots, C_{m-2}$ und folgende Ableitungsregeln ein.

$$\begin{aligned} \alpha &\to B_1 C_1 \\ C_i &\to B_{i+1} C_{i+1} \text{ für } 1 \leq i \leq m-3 \\ C_{m-2} &\to B_{m-1} B_m \end{aligned}$$

Aufgabe 7 (2 + 2 = 4 Punkte)

Zeigen Sie, dass folgende Sprachen nicht kontextfrei sind:

(a) $L_1 = \{w \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}^* \mid |w|_\mathsf{a} = |w|_\mathsf{b} = |w|_\mathsf{c}\}$, wobei $|w|_x$ angibt, wie oft x in w vorkommt.

(b) $L_2 = \left\{\mathsf{a}^i\mathsf{b}^i\mathsf{c}^j \mid i \geq j\right\}$.

Hinweis: Dazu können Sie das Pumping-Lemma für kontextfreie Sprachen verwenden, das in der 15. Vorlesung am 28.1. behandelt wird.

Lösung:

(a) Nehme an, dass L_1 kontextfrei ist. Dann existiert nach dem Pumping-Lemma ein $n \in \mathbb{N}$, sodass für jedes Wort z mit $|z| \geq n$ eine Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$, so dass $uv^iwx^iy \in L_1$ für alle $i \geq 0$. Wähle $z = \mathsf{a}^n\mathsf{b}^n\mathsf{c}^n$. Wegen $|vwx| \leq n$ kann vx kann nicht gleichzeitig a's und c's enthalten. Das Wort uv^2wx^2y enthält also nicht gleich viele a's und c's, liegt also nicht in L_1, was mit dem Pumping-Lemma ein Widerspruch zur Annahme, dass L_1 kontextfrei sei, ergibt.

(b) Widerlege die Annahme, dass L_2 kontextfrei ist, mit Ogdens Lemma. Wähle n als Konstante aus Ogdens Lemma. Betrachte $z = \mathsf{a}^{n+1}\mathsf{b}^{n+1}\mathsf{c}^{n+1} \in L_2$ und markiere alle b's. Betrachte Zerlegung $z = uvwxy$ gemäß Ogdens Lemma. Dann muss mindestens ein b zu vx gehören. Da höchstens n markierte Buchstaben zu vwx gehören dürfen, kann vx nicht gleichzeitig a's und c's enthalten. Das Wort uv^0wx^0y enthält dann mehr a's als b's oder mehr c's als b's und ist deswegen nicht in L_2 enthalten. Widerspruch.