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

Übungsblatt 7

Vorlesung Theoretische Grundlagen der Informatik im WS 21/22

Ausgabe: 28. Januar 2022
Abgabe: 11. Februar 2022 (digital im ILIAS)

Bitte bearbeiten Sie die Aufgaben **handschriftlich** und laden Sie eine eingescannte PDF-Version im Übungsmodul Ihrer ILIAS-Tutoriumsgruppe hoch! Beschriften Sie Ihren handschriftlichen Aufschrieb gut sichtbar mit Name und Matrikelnummer. Nicht handschriftliche oder unbeschriftete Abgaben werden nicht akzeptiert!

Aufgabe 1

(3 + 1 + 1 = 5 Punkte)

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

$$\begin{aligned} S &\to A \mid E\mathsf{a} \\ A &\to B\mathsf{a} \mid \mathsf{c}E\mathsf{c} \\ B &\to AD \mid \mathsf{b}E \mid \varepsilon \\ C &\to C\mathsf{a} \mid AB \\ D &\to \mathsf{c} \\ E &\to E\mathsf{b} \mid ED \end{aligned}$$

(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) Ist G' minimal in dem Sinne, dass es keine Grammatik mit weniger Variablen gibt, die $L(G')$ erzeugt? Begründen Sie Ihre Antwort.

Lösung:

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

I. Initialisiere $Q = V' = \{B, D\}$.

II. Entnehme B aus Q. Ersetze in allen Regeln B durch ε:

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

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

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

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

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

IV. Entnehme A aus Q. Ersetze in allen Regeln A durch a:

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

Nun ist $Q = \{S, C\}$.

V. Entnehme S aus Q. Die Regelmenge ändert sich nicht, da S in keiner Regel auf der rechten Seite vorkommt. Nun ist $Q = \{C\}$.

VI. Entnehme C aus Q. Ersetze in allen Regeln C durch a:

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

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

E ist nutzlos und kann entfernt werden:

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

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

I. Initialisiere $V'' = \{S\}$.
II. Über S lässt sich A erreichen: $V'' = \{S, A\}$.
III. Über A lässt sich B erreichen: $V'' = \{S, A, B\}$.
IV. Über B lässt sich D erreichen: $V'' = \{S, A, B, D\}$.
V. Über D lassen sich keine neuen Variablen erreichen. C ist also nutzlos.

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

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

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

(c) Nein, $L(G')$ kann auch mit nur einer Variable erzeugt werden, z.B. durch $G'' = (\Sigma, \{S\}, S, \{S \to S\mathsf{ca} \mid \mathsf{a}\})$.

Aufgabe 2

(2 + 3 = 5 Punkte)

Sei $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$. Geben Sie zu den folgenden Sprachen jeweils einen deterministischen Kellerautomaten an, der die Sprache erkennt. Erklären Sie jeweils kurz die Arbeitsweise des Kellerautomaten.

(a) $L_1 = \{\mathsf{a}^i\mathsf{b}^j\mathsf{c}^k \mid i, j, k \in \mathbb{N}_0, i + k = j\}$

(b) $L_2 = \{uv \mid u \in \{\mathsf{a}, \mathsf{b}\}^*, v \in \{\mathsf{c}\}^* \text{ und } |u|_\mathsf{a} + 2 \cdot |u|_\mathsf{b} = 5 \cdot |v|_\mathsf{c}\}$, wobei $|w|_x$ die Anzahl der Vorkommen von $x \in \Sigma$ in $w \in \Sigma^*$ bezeichnet.

Lösung:

(a) Wir können L_1 auch als $\{\mathsf{a}^i\mathsf{b}^i\mathsf{b}^k\mathsf{c}^k \mid i, k \in \mathbb{N}_0\}$ schreiben. Sei $M_1 =$

$(\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \Sigma, \{A, B\}, q_0, Z_0, \delta, \{q_0, q_3, q_6\})$ mit

$$\begin{aligned}
\delta(q_0, \mathsf{b}, Z_0) &= (q_4, BZ_0) \\
\delta(q_0, \mathsf{a}, Z_0) &= (q_1, AZ_0) \\
\delta(q_1, \mathsf{a}, A) &= (q_1, AA) \\
\delta(q_1, \mathsf{b}, A) &= (q_2, \epsilon) \\
\delta(q_2, \mathsf{b}, A) &= (q_2, \epsilon) \\
\delta(q_2, \epsilon, Z_0) &= (q_3, Z_0) \\
\delta(q_3, \mathsf{b}, Z_0) &= (q_4, BZ_0) \\
\delta(q_4, \mathsf{b}, B) &= (q_4, BB) \\
\delta(q_4, \mathsf{c}, B) &= (q_5, \epsilon) \\
\delta(q_5, \mathsf{c}, B) &= (q_5, \epsilon) \\
\delta(q_5, \epsilon, Z_0) &= (q_6, Z_0)
\end{aligned}$$

ein Kellerautomat, der mit akzeptierendem Endzustand akzeptiert. M_1 pusht zunächst die gelesenen as auf den Stack und löscht für jedes gelesene b ein Symbol vom Stack, bis der Stack leer ist. Das wiederholt M_1 für den zweiten Teil des Wortes mit b statt a und c statt b.

(b) Wir definieren eine Funktion $f\colon \Sigma \to \{1, 2, 5\}$, die jedem Zeichen ihren Wert zuweist, also $f(\mathsf{a}) = 1$, $f(\mathsf{b}) = 2$ und $f(\mathsf{c}) = 5$. Die Idee ist, im ersten Teil die Werte immer auf 5-er Schritte aufzusummieren und auf den Stack zu legen und im zweiten Teil des Wortes die 5en wieder vom Stack zu löschen.

Sei $M_1 = (\{q_0, q_1, q_2, q_3\}, \Sigma, \{1, 2, 3, 4, 5\}, q_0, Z_0, \delta, \{q_0, q_3\})$. Zunächst wird der erste gelesene Wert auf den Stack gelegt:

$$\begin{aligned}
\delta(q_0, \mathsf{a}, Z_0) &= (q_1, 1Z_0) \\
\delta(q_0, \mathsf{b}, Z_0) &= (q_1, 2Z_0)
\end{aligned}$$

Dann werden die gelesenen Werte auf den Wert im Stack addiert. Ist die Summe höchstens 5, wird der Wert im Stack einfach durch die Summe ersetzt:

$$\delta(q_1, x, Z) = (q_1, (f(x) + Z))) \ \ \forall x \in \{\mathsf{a}, \mathsf{b}\}, Z \in \Gamma \setminus \{Z_0\} \text{ mit } f(x) + Z \leq 5$$

Ist die Summe größer als 5, wird eine 5 und der Rest der Summe auf den Stack gelegt:

$$\delta(q_1, x, Z) = (q_1, (f(x) + Z - 5)\, 5)) \ \ \forall x \in \{\mathsf{a}, \mathsf{b}\}, Z \in \Gamma \setminus \{Z_0\} \text{ mit } f(x) + Z > 5$$

Wird ein c gelesen, geht der Kellerautomat in den zweiten Zustand über. Im zweiten Teil wird für jedes gelesene c eine 5 vom Stack gelöscht:

$$\begin{aligned}
\delta(q_1, \mathsf{c}, 5) &= (q_2, \epsilon) \\
\delta(q_2, \mathsf{c}, 5) &= (q_2, \epsilon) \\
\delta(q_2, \epsilon, Z_0) &= (q_3, Z_0)
\end{aligned}$$

Sobald der Stack leer ist, geht der Kellerautomat in den akzeptierenden Zustand q_3 über.

Aufgabe 3 (1 + 1 + 2 + 2 + 1 = 7 Punkte)

Ein *Input-Driven Deterministic Pushdown Automaton (IDPA)* ist ein etwas modifizierter deterministischer Kellerautomat. Dabei ist für jedes Symbol des Alphabets Σ vorher festgelegt, ob bei der Abarbeitung des Symbols etwas auf den Stack gelegt oder vom Stack gelöscht wird. Zusätzlich zum Alphabet Σ betrachten wir also ein *Pushdown-Alphabet* $\widehat{\Sigma} = (\Sigma_{push}, \Sigma_{pop}, \Sigma_{-})$ mit $\Sigma = \Sigma_{push} \dot\cup \Sigma_{pop} \dot\cup \Sigma_{-}$. Das Pushdown-Alphabet $\widehat{\Sigma}$ partitioniert Σ in drei untereinander paarweise disjunkte Mengen.

Formal besteht ein IDPA über dem Pushdown-Alphabet $\widehat{\Sigma} = (\Sigma_{push}, \Sigma_{pop}, \Sigma_{-})$ wie ein deterministischer Kellerautomat aus $(Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ mit $\Sigma = \Sigma_{push} \dot\cup \Sigma_{pop} \dot\cup \Sigma_{-}$. Außerdem hat ein Zustandsübergang $\delta(q, a, Z) = (q', \gamma)$ mit $q, q' \in Q$, $a \in \Sigma$ und $\gamma \in \Gamma^*$ die folgenden Einschränkungen:

- Ist $a \in \Sigma_{push}$, gilt $\gamma = Z'Z$ für ein $Z' \in \Gamma \setminus \{Z_0\}$, es wird also genau ein neues Symbol auf den Stack gelegt.
- Ist $a \in \Sigma_{pop}$, gilt $Z \neq Z_0$ und $\gamma = \varepsilon$, es wird also das oberste Symbol auf dem Stack gelöscht, außer es ist das Initialisierungssymbol.
- Ist $a \in \Sigma_{-}$, gilt $\gamma = Z$, der Stack bleibt also unverändert.

Der IDPA hat keine ε-Übergänge und akzeptiert mit akzeptierendem Endzustand.

(a) Beschreiben Sie, wie die Sprache der korrekten Klammerausdrücke über dem Alphabet $\Sigma = \{[,]\}$ von einem IDPA erkannt werden kann.

(b) Geben Sie eine kontextfreie Sprache an, die von keinem IDPA erkannt werden kann. Begründen Sie.

(c) Seien L_1 und L_2 zwei Sprachen über einem Alphabet Σ, die von den IDPAs M_1 bzw. M_2 über demselben Pushdown-Alphabet $\widehat{\Sigma} = (\Sigma_{push}, \Sigma_{pop}, \Sigma_{-})$ erkannt werden. Zeigen Sie, dass es einen IDPA $M_\cap$ gibt, der $L_1 \cap L_2$ erkennt.

(d) Die Menge der Sprachen, die von einem deterministischen Kellerautomaten erkannt werden können, sind nicht unter Durchschnittbildung abgeschlossen. Erklären Sie kurz, warum die Konstruktion aus (c) für IDPAs funktioniert, aber nicht für deterministische Kellerautomaten.

(e) Begründen Sie, dass die Menge der Sprachen, die von IDPAs erkannt werden, auch unter Komplement und Vereinigung abgeschlossen sind.

Lösung:

(a) Wir betrachten den IDPA $M = (\{s, q_1\}, \Sigma, \{[, \widetilde{[}\}, s, Z_0, \delta, \{s\})$ über dem Pushdown-Alphabet $\widehat{\Sigma} = (\Sigma_{push}, \Sigma_{pop}, \Sigma_{-})$ mit $\Sigma_{push} = \{[\}$, $\Sigma_{pop} = \{]\}$ und $\Sigma_{-} = \emptyset$. M arbeitet gleich wie ein deterministischer Kellerautomat: Wird eine offene Klammer abgearbeitet, wird diese auf den Stack gelegt. Bei einer schließenden Klammer wird eine Klammer vom Stack gelöscht. Ist dies nicht möglich, wird das Wort abgelehnt. Mit dem Symbol $\widetilde{[}$ wird die unterste Klammer auf dem Stack markiert, damit M merkt, wann diese gelöscht wird. M akzeptiert, wenn beim letzten Abarbeitungsschritt die unterste Klammer vom Stack gelöscht wird.

Formal sind die Zustandsübergänge von M wie folgt definiert:

$$\begin{aligned}\delta(s, [, Z_0) &= (q_1, \widetilde{[}Z_0)\\ \delta(q_1, [, Z) &= (q_1, [Z)\ \forall Z \in \Gamma\\ \delta(q_1,], [) &= (q_1, \varepsilon)\\ \delta(q_1,], \widetilde{[}) &= (s, \varepsilon)\end{aligned}$$

(b) Die kontextfreie Sprache $L = \{\mathsf{a}^n\#\mathsf{a}^n \mid n \in \mathbb{N}\}$ kann nicht von einem IDPA erkannt werden. Ist $\mathsf{a} \in \Sigma_- \cup \Sigma_{pop}$, besteht der Stack eines IDPAs nach Abarbeitung der ersten Hälfte nur aus dem Initialisierungssymbol, damit hat ein IDPA keine Informationen über die Größe von n. Ist $\mathsf{a} \in \Sigma_{push}$, muss jedes Mal, wenn ein a gelesen wird, ein Symbol auf den Stack gelegt werden. Dadurch kann zwar die Information über die Größe von n durch die Höhe des Stacks nach Abarbeitung der ersten Hälfte beschrieben werden, allerdings kann darauf in der zweiten Hälfte nicht zugegriffen werden, da auch hier weiter Symbole auf den Stack gelegt werden müssen.

(c) Seien $M_1 = (Q_1, \Sigma, \Gamma_1, q_0^1, Z_0^1, \delta_1, F_1)$ und $M_2 = (Q_2, \Sigma, \Gamma_2, q_0^2, Z_0^2, \delta_2, F_2)$ Wir konstruieren $M_\cap$ als den Produktautomaten von M_1 und M_2 mit $M_\cap = (Q_1 \times Q_2, \Sigma, \Gamma_1 \times \Gamma_2, (q_0^1, q_0^2), (Z_0^1, Z_0^2), \delta_\cap, F_1 \times F_2)$. Dabei ist $\delta_\cap((q_1, q_2), a, (Z_1, Z_2))$ für $\delta_1(q_1, a, Z_1) = (q_1', \gamma_1)$ und $\delta_2(q_2, a, Z_2) = (q_2', \gamma_2)$ wie folgt definiert:

- Ist $a \in \Sigma_{push}$, gilt $\gamma_1 = Z_1' Z_1$ für ein $Z_1' \in \Gamma_1 \setminus \{Z_0^1\}$ und $\gamma_2 = Z_2' Z_2$ für ein $Z_2' \in \Gamma_2 \setminus \{Z_0^2\}$. M muss also das Symbol (Z_1', Z_2') auf den Stack legen, damit ist $\delta_\cap((q_1, q_2), a, (Z_1, Z_2)) = ((q_1', q_2'), (Z_1', Z_2')(Z_1, Z_2))$.
- Ist $a \in \Sigma_{pop}$, löschen M_1 und M_2 ein Symbol vom Stack. M muss also das Symbol (Z_1, Z_2) vom Stack löschen, damit ist $\delta_\cap((q_1, q_2), a, (Z_1, Z_2)) = ((q_1', q_2'), \varepsilon)$.
- Ist $a \in \Sigma_-$, bleibt in M_1 und M_2 jeweils der Stack unverändert, also ist $\delta_\cap((q_1, q_2), a, (Z_1, Z_2)) = ((q_1', q_2'), (Z_1, Z_2))$.

Der konstruierte Kellerautomat ist also ein IDPA und akzeptiert ein Wort $w \in \Sigma^*$ genau dann, wenn M_1 und M_2 das Wort akzeptieren.

(d) Die Stacks zweier IDPAs über dem gleichen Pushdown-Alphabet $\widehat{\Sigma}$ sind bei der Abarbeitung des gleichen Wortes synchronisiert und legen gleichzeitig ein Symbol auf den Stack bzw. löschen eins, da diese Aktionen durch das Eingabewort bereits vorgegeben sind. Das heißt, die Symbole in gleicher Höhe der zwei Stacks können als Paare betrachtet werden, da diese immer die gleiche Höhe in ihren Stacks haben. Bei zwei deterministischen Kellerautomaten ist das nicht der Fall, da beispielsweise der eine Automat in einem Abarbeitungsschritt etwas vom Stack löschen kann, während der andere Automat etwas auf den Stack legen kann.

(e) Sei M ein IDPA, der eine Sprache L erkennt. Dann kann die Komplementsprache $\overline{L} = \Sigma \setminus L$ erkannt werden, indem das Akzeptanzverhalten der Endzustände invertiert wird.

Seien L_1 und L_2 zwei Sprachen, die von den IDPAs M_1 bzw. M_2 über dem gleichen Pushdown-Alphabet erkannt werden. Dann kann auch $L_1 \cup L_2$ von einem IDPA erkannt werden, da $L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$ und die Menge der Sprachen, die von IDPAs erkannt werden können, abgeschlossen unter Komplement und Vereinigung ist.

Aufgabe 4 (2 + 4 (+2) = 6 (8) Punkte)

(a) Sei $k \in \mathbb{N}$ eine beliebige, aber feste Konstante. Ein k-beschränkter Kellerautomat ist ein deterministischer Kellerautomat, dessen Stack zu jedem Zeitpunkt höchstens k Symbole enthalten darf. Zeigen Sie, dass k-beschränkte Kellerautomaten genau die Menge der regulären Sprachen erkennen können.

(b) Ein Queue-Automat ist wie ein Kellerautomat definiert, besitzt aber statt eines Stacks eine Queue. Das heißt, bei einem Abarbeitungsschritt kann das vorderste Symbol in der Queue entfernt und neue Symbole hinten in der Queue eingefügt werden. Zusätzlich besitzt der Queue-Automat einen Queue-Lesekopf, der in jedem Schritt ein Element der Queue lesen und sich nach links/rechts bewegen oder stehen bleiben kann.

Welche Klasse von Sprachen können Queue-Automaten erkennen? Beweisen Sie Ihre Behauptung.

(c) *Bonusaufgabe:* Zeigen Sie, dass ein Queue-Automat, der keinen Queue-Lesekopf hat (also nur das vorderste Zeichen lesen kann), dieselbe Klasse von Sprachen wie der Automat in (b) erkennt.

Lösung:

(a) Ein k-beschränkter Kellerautomat, der seinen Stack ignoriert, ist ein deterministischer endlicher Automat, also können k-beschränkte Kellerautomaten reguläre Sprachen erkennen.

Umgekehrt zeigen wir, dass jeder k-beschränkte Kellerautomat durch einen deterministischen endlichen Automaten simuliert werden kann. Da der Stack nur begrenzte Kapazität k hat, gibt es nur $|Q| \cdot |\Gamma|^k$ Kombinationen von Zustand des Automaten und Konfiguration des Stacks. Da es also nur eine endliche Anzahl an Kombinationen gibt, können wir diese als Zustände eines endlichen Automaten kodieren.

(b) Ein Queue-Automat ist genauso mächtig wie eine Turingmaschine, erkennt also genau die semi-entscheidbaren Sprachen.

Eine 2-Band-Turingmaschine kann einen Queue-Automaten wie folgt simulieren: Auf Band 1 steht das Eingabewort. Band 2 repräsentiert den Inhalt der Queue, wobei das vorderste Symbol in der Queue an vorderster Stelle auf dem Band steht. Das Symbol, das gerade vom Queue-Lesekopf gelesen wird, wird entsprechend markiert. Kopf 1 geht das Eingabewort von links nach rechts durch. Um den aktuellen Übergang zu bestimmen, wird zunächst auf Band 2 das markierte Symbol gesucht und gelesen. Dann wird die Markierung aktualisiert und ggf. eine push- oder pop-Operation ausgeführt. Bei einer push-Operation wird Kopf 2 auf das letzte Symbol von Band 2 bewegt und das neue Symbol wird dahinter auf das Band geschrieben. Bei einer pop-Operation wird Kopf 2 auf das erste Symbol von Band 2 bewegt, das dann gelöscht wird. Da 2-Band-Turingmaschinen genauso mächtig sind wie 1-Band-Turingmaschinen, ist eine Turingmaschine mindestens so mächtig wie ein Queue-Automat.

Umgekehrt kann jede Turingmaschine von einem Queue-Automaten simuliert werden. Die Idee ist, den Bandinhalt und die Position des Kopfes in der Queue zu speichern. Dazu führen wir für jedes Zeichen $a \in \Sigma$ ein zusätzliches Zeichen $\widehat{a}$ ein, womit wir in der Queue die Kopfposition markieren können. Außerdem wird das Ende des Bands durch ein zusätzliches Zeichen repräsentiert.

Zu Beginn wird das Eingabewort in die Queue geschrieben. Die Elemente in einer Queue können „durchrotiert“ werden, indem wiederholt das vorderste Element entfernt und hinten wieder eingefügt wird. Damit ist die Repräsentation des Bandinhalts in der Queue also zyklisch.

Soll der Kopf nach rechts bewegt werden, rotieren wir den Inhalt der Queue so lange, bis das Zeichen mit dem Kopfsymbol an vorderster Stelle in der Queue ist. Dieses Zeichen wird aus

der Queue entfernt und ohne Kopfsymbol am Ende wieder eingefügt. Das nächste Zeichen wird ebenfalls aus der Queue entfernt und mit Kopfsymbol am Ende wieder eingefügt.

Soll der Kopf nach links bewegt werden, rotieren wir den Inhalt der Queue so lange, bis das Zeichen mit dem Kopfsymbol an zweiter Stelle in der Queue ist. Das können wir feststellen, indem wir mit dem Queue-Lesekopf das Zeichen an dieser Stelle lesen. Das erste Zeichen (ohne Kopfsymbol) wird aus der Queue entfernt und mit Kopfsymbol am Ende wieder eingefügt. Das zweite Zeichen wird ebenfalls aus der Queue entfernt und ohne Kopfsymbol am Ende wieder eingefügt.

(c) Bei der Kopfbewegung nach links müsste der Queue-Automat eigentlich die ersten beiden Zeichen in der Queue betrachten. Das kann umgangen werden, indem wir für jedes Zeichen neue Zustände einführen, die das erste Zeichen in der Queue „puffern“. Dieses wird also durch einen Zustand kodiert und befindet sich in der Reihenfolge des Bandinhalts vor dem ersten und nach dem letzten Zeichen der Queue. Auch hier kann der Inhalt der Queue durchgeschoben werden: Das vorderste Zeichen in der Queue wird aus der Queue entfernt und in den Zustand kodiert. Das Zeichen, das vom Zustand kodiert wurde, wird hinten in der Queue wieder eingefügt. Wir können also die Übergänge abhängig von zwei Zeichen definieren, wobei eins vom aktuellen Zustand kodiert ist und eins sich an erster Stelle in der Queue befindet.

Bei einer Kopfbewegung nach rechts muss sich das Zeichen mit dem Kopf an vorderster Stelle in der Queue befinden. Das Zeichen, das vom Zustand kodiert wird, wird mit Kopfsymbol hinten eingefügt. Das erste Zeichen wird aus der Queue entfernt und ohne Kopfsymbol durch den Zustand kodiert. Bei einer Kopfbewegung nach links muss das Zeichen mit dem Kopf durch den Zustand kodiert sein. Dieses wird ohne Kopfsymbol hinten eingefügt. Das erste Zeichen wird aus der Queue entfernt und mit Kopfsymbol durch den Zustand kodiert.

Aufgabe 5 (2 + 3 + 2 = 7 Punkte)

Eine Instanz des Post'schen Korrespondenzproblems (PKP) ist eine endliche Menge von Wortpaaren $K = \{(x_1, y_1), \dots (x_n, y_n)\}$ über einem endlichen Alphabet Σ. Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$ für alle $i = 1, \dots, n$. Eine Lösung ist eine Folge von Indizes $i_1, \dots, i_k \in \{1, \dots, n\}$, für die

$$x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$$

gilt. Dieses Problem ist nicht entscheidbar.

Im Folgenden sollen Sie zeigen, dass es nicht entscheidbar ist, ob der Schnitt von zwei gegebenen kontextfreien Sprachen wieder kontextfrei ist. Betrachten Sie dazu die folgende Sprache L_K über dem Alphabet $\Gamma = \Sigma \cup \{\#, 1, \dots, n\}$ zu einer gegebenen PKP-Instanz K:

$$L_K = \left\{ i_k \dots i_1 x_{i_1} \dots x_{i_k} \# y^R_{i_k} \dots y^R_{i_1} i_1 \dots i_k \;\middle|\; x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k},\ i_1, \dots, i_k \in \{1, \dots, n\} \right\}$$

(a) Geben Sie zwei kontextfreie Sprachen L_1 und L_2 an, sodass $L_K = L_1 \cap L_2$ gilt. Zeigen Sie, dass L_1 und L_2 kontextfrei sind.

(b) Zeigen Sie, dass L_K genau dann kontextfrei ist, wenn die PKP-Instanz K keine Lösung hat. Sie können dabei verwenden, dass ein Wort $i_k \dots i_1 x_{i_1} \dots x_{i_k} \# y^R_{i_k} \dots y^R_{i_1} i_k \dots i_1$ genau dann in L_k ist, wenn $i_1 \dots i_k$ eine Lösung von K ist.

Hinweis 1: Verwenden Sie für eine Beweisrichtung das Pumping-Lemma für kontextfreie Sprachen.

Hinweis 2: Hat die Instanz K eine Lösung $i_1 \dots i_k$, dann sind auch $(i_1 \dots i_k)^m$ für alle $m \in \mathbb{N}_+$ Lösungen von K, da wir Lösungen konkatenieren können, um neue Lösungen zu erhalten.

(c) Folgern Sie, dass es unentscheidbar ist, ob der Schnitt von zwei gegebenen kontextfreien Sprachen kontextfrei ist.

Lösung:

(a) Sei $G_1 = (\Gamma, S_1, \{S, A, B\}, R_1)$ eine kontextfreie Grammatik mit

$$\begin{aligned} R_1 = \{ & S \to A\#B \\ & A \to 1Ax_1 \mid \ldots \mid nAx_n \mid 1x_1 \mid \ldots \mid nx_n \\ & B \to y_1^R B1 \mid \ldots \mid y_n^R Bn \mid y_1^R 1 \mid \ldots \mid y_n^R n\} \end{aligned}$$

Sei $G_2 = (\Gamma, S_1, \{S, A, B\}, R_1)$ eine kontextfreie Grammatik, die genau die Spiegelsprache erkennt mit

$$R_1 = \{S \to aSa \mid a \in \Gamma\} \cup \{S \to \varepsilon\}$$

Dann ist $L(G_1) \cap L(G_2) = L_K$.

(b) Sei K eine PKP Instanz über dem Alphabet Σ. Hat K keine Lösung, dann ist L_K genau die leere Sprache und damit kontextfrei.

Sei $i_1 \ldots i_k$ eine Lösung von K. Wir nehmen an, dass L_K kontextfrei ist. Dann existiert nach dem Pumping-Lemma ein $m \in \mathbb{N}$, sodass für jedes Wort $z \in L_K$ mit $|z| \geq m$ eine Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq m$, so dass $uv^iwx^iy \in L_K$ für alle $i \geq 0$. Wähle

$$z = (i_k \ldots i_1)^m (x_{i_1} \ldots x_{i_k})^m \#(y_{i_k}^R \ldots y_{i_1}^R)^m (i_1 \ldots i_k)^m \, .$$

Dieses Wort ist wegen Hinweis 2 in L_K enthalten. Wegen $|vwx| \leq m$ kann vx höchstens zwei der vier Teile enthalten. Enthält vwx Indizes (ist also am Anfang oder am Ende von z), ist uv^0wx^0y kein Palindrom und damit nicht in L_2. Enthält vwx keine Indizes (ist also in der Mitte von z), passen in uv^0wx^0y die x_i oder die y_i nicht zu den Indizes, ist also kein Wort in L_1. Das ergibt mit dem Pumping-Lemma ein Widerspruch zur Annahme, dass L_K kontextfrei sei.

(c) Wir nehmen an, dass eine Turingmaschine M existiert, die entscheidet, ob der Schnitt zweier gegebener kontextfreier Grammtiken kontextfrei ist.

Daraus wollen wir eine Turingmaschine M_K bauen, die PKP entscheidet. Gegeben eine PKP-Instanz K, konstruieren wir die zwei Grammatiken G_1 und G_2 wie in (a). Auf G_1 und G_2 simuliert M_K die Turingmaschine M. Akzeptiert M, dann ist $L(G_1) \cap L(G_2)$ kontextfrei und M_K lehnt ab. Lehnt M ab, ist $L(G_1) \cap L(G_2)$ und nach (b) existiert eine Lösung für die PKP-Instanz und M_K akzeptiert. Damit wäre also auch PKP entscheidbar, was aber ein Widerspruch ist.

Aufgabe 6 (1 + 1 = 2 Punkte)

Die folgende Tabelle gibt eine Häufigkeitsverteilung für die Zeichen des Alphabets $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}, \mathsf{g}, \mathsf{h}\}$ an.

a	b	c	d	e	f	g
12%	22%	5%	15%	14%	28%	4%

(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) Was ist die erwartete Länge der Kodierung eines Wortes in Abhängigkeit von dessen Länge n, wenn sie die obige Verteilung und Ihre Huffman-Kodierung zugrunde legen?

Lösung:

(a) $\mathsf{a} \mapsto \mathtt{110}, \mathsf{b} \mapsto \mathtt{10}, \mathsf{c} \mapsto \mathtt{1110}, \mathsf{d} \mapsto \mathtt{000}, \mathsf{e} \mapsto \mathtt{001}, \mathsf{f} \mapsto \mathtt{01}, \mathsf{g} \mapsto \mathtt{1111}$

(b) $(0.12 \cdot 3 + 0.22 \cdot 2 + 0.05 \cdot 4 + 0.15 \cdot 3 + 0.14 \cdot 3 + 0.28 \cdot 2 + 0.04 \cdot 4) \cdot n = 2.59n$