Prof. Javier Esparza
Philipp Czerner, Martin Helfrich

Technische Universität München
Lehrstuhl für Theoretische Informatik

Einführung in die Theoretische Informatik

Sommersemester 2021 – Hausaufgabenblatt 8

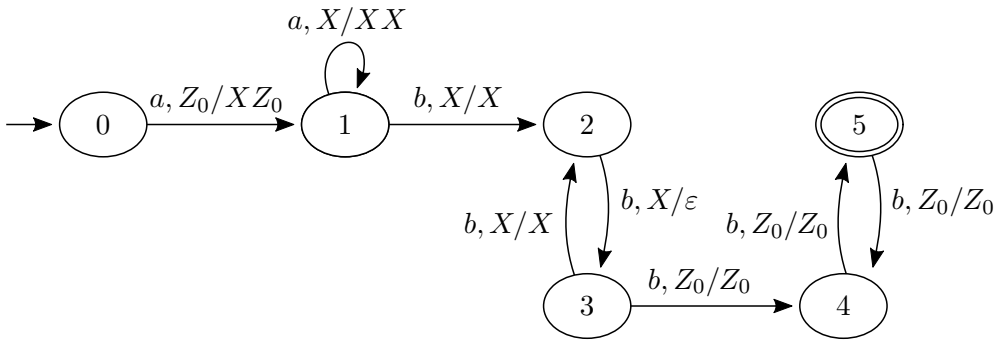
- Sei $\Phi := \{\{1\}, \{2\}, \{3, 4\}, \{5\}\}$. Nach dem Abgabedatum werden wir für jede Menge $A \in \Phi$ eine zufällige Aufgabe $a \in A$ wählen und korrigieren.
- Es werden diese Aufgaben korrigiert: H8.1, H8.2, H8.4, H8.5

Aufgabe H8.1. (AT) 1+1+1 Punkte

Diese Hausaufgabe wird mit Automata Tutor bearbeitet und abgegeben. Falls Sie es noch nicht bereits gemacht haben, folgen Sie den Schritten in Ü1.2, um ein Konto zu erstellen. Achten Sie darauf, dass Sie sich, wie dort beschrieben, mit Ihrer TUM-Kennung anmelden. Ansonsten können wir Ihnen die Punkte nicht gutschreiben.

Bearbeiten Sie die Hausaufgaben H8.1 (a–c). **Achtung:** Während Sie für die Aufgaben aus dem Übungsblatt beliebig viele Versuche hatten, haben Sie für jede Hausaufgabe nur 5 Versuche. Sie bekommen nur dann einen Punkt, wenn Sie die Aufgabe nach 5 Versuchen vollständig (also mit 10/10 Punkten) gelöst haben. Ihr konstruierter Automat darf nicht zu viele Zustände haben (siehe Aufgabenstellung). Wenn Sie einen ε-Übergang angeben wollen, geben Sie statt ε bitte `E` ein (siehe Hinweisbox über Canvas) und beim Entfernen eines Kellersymbols, lassen Sie den Teil hinter "/" bitte leer (Beispiel: "`a`, X/").

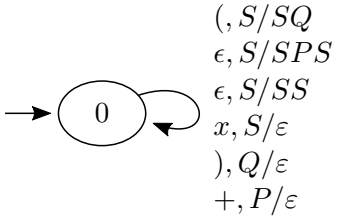
Lösungsskizze. **(a)** Ein DPDA für $\{a^n b^{2m} : 0 < n < m\}$



(b) Um einen PDA für die Sprache der Grammatik $S \to (S) \mid S+S \mid SS \mid x$ zu erstellen, formen wir die Grammatik zunächst um.

$$S \to (SQ \mid SPS \mid SS \mid x$$
$$Q \to) \qquad P \to +$$

Den PDA erstellen wir nun mit dem Verfahren aus der Vorlesung. Das initiale Kellersymbol ist S, und der PDA akzeptiert über leeren Keller.



(c) Vorab kurz die Anmerkung, dass man für diese Aufgabe den DPDA auch direkt hinschreiben kann. In der folgenden Lösung präsentieren wir eine Möglichkeit, wie man strukturiert vorgehen kann – dies ist aber nicht notwendig.

Für diesen Aufgabenteil formen wir die Grammatik zunächst um, mit dem Ziel, dass man nur mit Kenntnis des nächsten Zeichen des Wortes bereits entscheiden kann, welche Produktion verwendet werden soll. Die Produktionen $S \to S + S \mid SS$ stellen hier jedoch ein Problem dar, da man sie erst nach dem Einlesen von S unterscheiden kann. (Produktionen solcher Form bezeichnet man auch als *linksrekursiv*, für mehr Details sei auf die Vorlesung Compilerbau verwiesen.) Also führen wir eine neue Variable T ein, zu der wir die Produktionen von S, die nicht mit S beginnen, bewegen, und terminieren die Rekursion direkt in T:

$$S \to T + S \mid TS \mid T$$
$$T \to (S) \mid x$$

Die Produktionen von S beginnen allerdings alle mit T, sodass wir immer noch nicht zwischen ihnen unterscheiden können. Stattdessen führen wir eine neue Variable ein, sodass S nur noch eine Produktion hat.

$$S \to TR$$
$$R \to +S \mid S \mid \varepsilon$$
$$T \to (S) \mid x$$

Anschließend können wir die Produktionen $R \to S$ umformen, indem wir $S \to TR$ und $T \to (S) \mid x$ einsetzen. Bei R und T setzen wir S ebenfalls ein.

$$S \to TR$$
$$R \to +TR \mid (TR)R \mid xR \mid \varepsilon$$
$$T \to (TR) \mid x$$

Nun hat jede Variable entweder genau eine Produktion, oder jede Produktion beginnt mit unterschiedlichen Terminalsymbolen. Jede Produktion? Nein! Die Produktion $R \to \varepsilon$ leistet erbitterten Widerstand. Wir wollen die Grammatik nun weiter umformen, sodass die Produktion $R \to \varepsilon$ nur am Ende des Wortes vorkommen kann (und wir das Wort mit dieser Produktion also direkt akzeptieren).

Wir beobachten, dass nach R immer entweder das Ende des Wortes oder) kommt. Wir führen also eine neue Variable K ein, die R und dann eine schließende Klammer einliest.

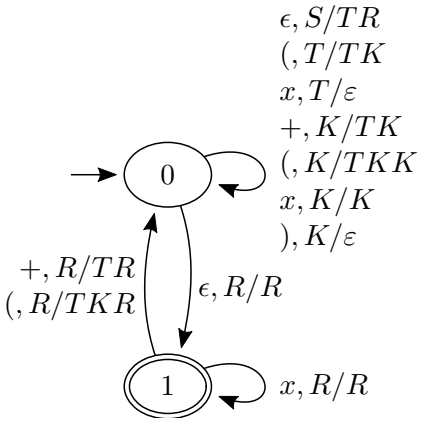
$$S \to TR$$
$$R \to +TR \mid (TKR \mid xR \mid \varepsilon$$
$$T \to (TK \mid x$$
$$K \to R)$$

Damit R nun tatsächlich immer nur am Ende des Wortes steht, müssen wir die Produktion $K \to R)$ ersetzten, und die dadurch entstandenen $R)$ in den Produktionen von K

wieder durch K.

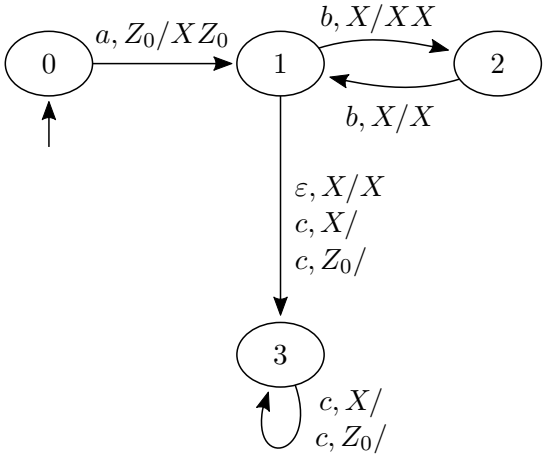
$$
\begin{aligned}
S &\to TR \\
R &\to +TR \mid (TKR \mid xR \mid \varepsilon \\
T &\to (TK \mid x \\
K &\to +TK \mid (TKK \mid xK \mid)
\end{aligned}
$$

Hieraus lässt sich ein DPDA erstellen. Der DPDA akzeptiert über Finalzustände. Wir haben dafür gesorgt, dass R stets an der untersten Stelle des Kellers steht, sodass wir ein Wort genau dann akzeptieren, wenn wir R das einzige Zeichen im Keller ist. Also fügen wir eine ϵ-Transition ein, die feststellt, ob dies der Fall ist.

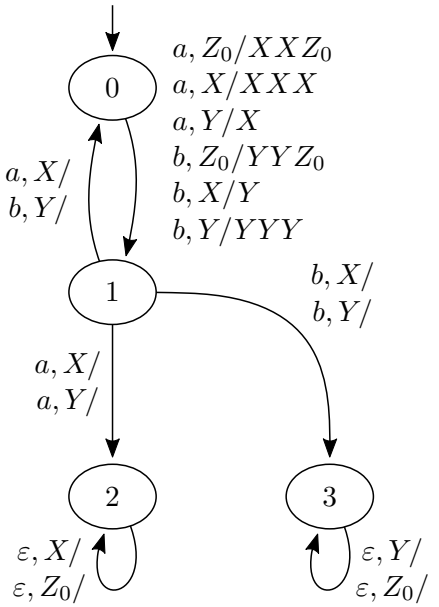


Aufgabe H8.2. (*Sprachfindung*) 1+1+1+1 Punkte

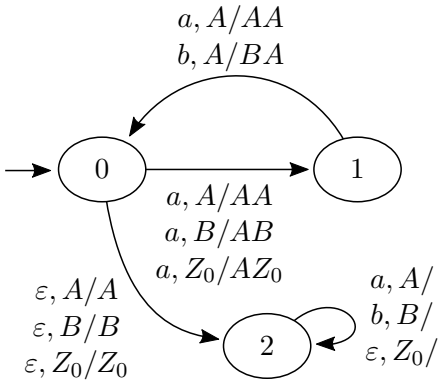
Geben Sie für die folgenden Kellerautomaten jeweils die akzeptierte Sprache in Mengennotation an. Die PDAs akzeptieren über leeren Keller.



(a)

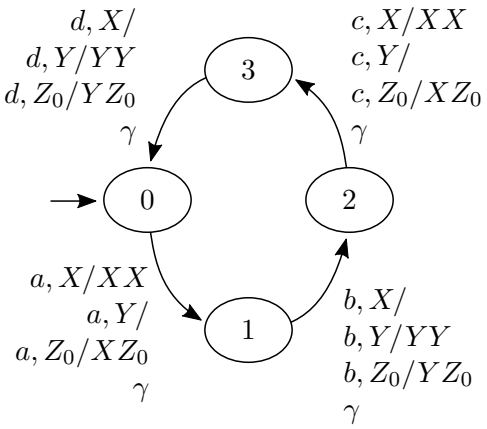


(b)



(c)

$\gamma := (\ \varepsilon, Z_0/Z_0 \quad \varepsilon, X/X \quad \varepsilon, Y/Y\ )$



(d)

Lösungsskizze.

(a) $\{ab^{2n}c^{n+2} : n \in \mathbb{N}\}$

(b) $\{wxy : w \in \{aa, bb\}^*, x, y \in \{a, b\}, f(y)(f(w) + 2f(x)) \geq 0\},\ f(w) := |w|_a - |w|_b$. Alternativ:

$$\begin{aligned} &\{waa : w \in \{aa, bb\}^*, \tfrac{1}{2}|w|_a + 1 \geq \tfrac{1}{2}|w|_b\} \\ \cup\ &\{wba : w \in \{aa, bb\}^*, \tfrac{1}{2}|w|_a - 1 \geq \tfrac{1}{2}|w|_b\} \\ \cup\ &\{wab : w \in \{aa, bb\}^*, \tfrac{1}{2}|w|_a + 1 \leq \tfrac{1}{2}|w|_b\} \\ \cup\ &\{wbb : w \in \{aa, bb\}^*, \tfrac{1}{2}|w|_a - 1 \leq \tfrac{1}{2}|w|_b\} \end{aligned}$$

(c) $\{ww^R : w \in L((ab|aa)^*)\}$

(d) Die leere Sprache, also $\emptyset$. Es gab einen Fehler in der Aufgabe, die Sprache hätte eigentlich $\{w \in \{a, b, c, d\}^* : |w|_a + |w|_c = |w|_b + |w|_d\}$ sein sollen (aber Z_0 wird nie entfernt).

Aufgabe H8.3. (*Kontextfreie Residualsprachen*) 1+3+1 Punkte

Sei Σ ein Alphabet. Zur Erinnerung: Für eine Sprache $L \subseteq \Sigma^*$ und ein Wort $w \in \Sigma^*$ ist $L^w := \{v : wv \in L\}$ die Residualsprache von L bezüglich w (siehe Ü5.6).

(a) Sei G folgende Grammatik:

$$\begin{aligned} S &\to aC \mid Da \mid BC & C &\to aS \mid BC \\ B &\to BB \mid b & D &\to CB \mid \varepsilon \end{aligned}$$

Bestimmen Sie eine kontextfreie Grammatik für $L(G)^a$.

(b) Sei $H = (V, \Sigma, P, S)$ eine beliebige kontextfreie Grammatik und $c \in \Sigma$. Beschreiben Sie ein allgemeines Verfahren, um eine kontextfreie Grammatik $H' = (V', \Sigma, P', S')$ mit $L(H') = L(H)^c$ zu konstruieren.

(c) Führen Sie Ihr Verfahren auf der Grammatik G aus, um eine Grammatik für $L(G)^b$ zu bestimmen.

Hinweise: Sie dürfen auch (b) zur Lösung von (a) verwenden. Bitte beachten Sie, dass wir aus der Vorlesung nur wissen, dass die Greibach-Normalform existiert, wir kennen jedoch kein Verfahren, sie zu erzeugen. (Streng genommen wissen wir auch nicht, ob ein solches Verfahren existiert.) Lösungen, die sich auf die GNF berufen, sind wahrscheinlich inkorrekt.

Lösungsskizze. **(a)** Wir fügen zu G ein neues Startsymbol S' und folgende Produktionen hinzu:

$$S' \to C \mid SBa \mid \varepsilon$$

Begründung: $L_G(aC)^a = L_G(C)$ ist klar, also $S' \to C$. Da $\varepsilon \in L_G(D)$, gilt $a \in L(G)$ und wir fügen $S' \to \varepsilon$ hinzu. Davon abgesehen gibt es die Produktion $D \to CB$, die genau dann mit einem a beginnt, wenn wir $C \to aS$ ableiten, da $L_G(B) \subseteq L(b^+)$. Also gilt $S \to DA \to CBa \to aSBa$ und wir fügen $S' \to SBa$ hinzu. Wir angemerkt gilt $L_G(B) \subseteq L(b^+)$, also erzeugt die Produktion $S \to BC$ keine Wörter, die mit a beginnen.

Insgesamt ergibt sich also folgende Grammatik:

$$\begin{array}{ll} S' \to C \mid SBa \mid \varepsilon & \\ S \to aC \mid Da \mid BC & C \to aS \mid BC \\ B \to BB \mid b & D \to CB \mid \varepsilon \end{array}$$

(b) Wir konvertierten H zuerst zur Chomsky-Normalform. (Beachten Sie, dass es keinen Einfluss auf $L(H)^c$ hat, ob $\varepsilon \in L(H)$.) Als Variablen verwenden wir $V' := V \uplus \{X_c : X \in V\}$, und das Startsymbol ist nun $S' := S_c$. Die Idee ist, dass $L_{H'}(X_c) = L_H(X)^c$ für jedes $X \in V$ gilt, dass also X_c genau die Wörter erzeugt, die der Residualsprache der von X erzeugen Wörter entspricht.

Die Produktionen P' konstruieren wir folgendermaßen:

- Alle Produktionen aus P sind in P' enthalten.
- Für $(X \to c) \in P$ fügen wir $X_c \to \varepsilon$ zu P' hinzu.
- Für $(X \to YZ) \in P$ fügen wir $X_c \to Y_cZ$ zu P' hinzu.

(c) Zuerst konvertieren wir G zur Chomsky-Normalform.

$$\begin{array}{ll} S \to AC \mid DA \mid BC \mid a & C \to AS \mid BC \\ B \to BB \mid b & D \to CB \\ A \to a & \end{array}$$

Nun fügen wir folgende Produktionen hinzu (S_b ist das neue Startsymbol).

$$\begin{array}{ll} S_b \to A_bC \mid D_bA \mid B_bC & C_b \to A_bS \mid B_bC \\ B_b \to B_bB \mid \varepsilon & D_b \to C_bB \end{array}$$

Wir merken an, dass das Symbol A_b nicht erzeugend ist – alle Produktionen, die es involvieren, könnten also auch gelöscht werden.

Aufgabe H8.4. (*Twice the ↑, double the ↓*) 2+3 Punkte

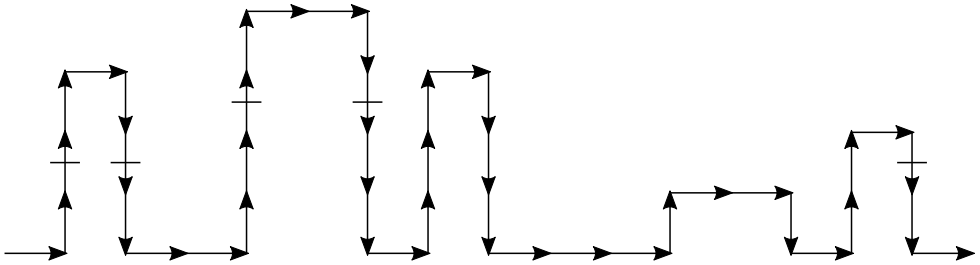
Durch die steigenden Bevölkerungszahlen in jüngerer Vergangenheit wachsen auch die Orte Estlingen und Oberfeld langsam zusammen. Dies stellt die Bebauungsplanerstellungskommission der Gemeindeverwaltung Oberfeld vor Probleme, die (teils kuriosen)

traditionellen Estlinger Bauweisen mit den wissenschaftlich fundierten Oberfelder Vorgaben zu vereinen. Nun benötigen sie Ihre Hilfe bei einem besonders kniffeligen Fall.

Ähnlich zu Übungsaufgabe Ü5.7 betrachten wir „Skylines“. Sei also $\Sigma := \{\uparrow, \rightarrow, \downarrow, \Updownarrow, \Downarrow\}$ ein Alphabet, wobei wir Wörter aus Σ^* als Linien illustrieren. Beispielsweise stellen wir

$$\rightarrow\uparrow\Updownarrow\uparrow\rightarrow\downarrow\Downarrow\downarrow\rightarrow\rightarrow\uparrow\uparrow\Updownarrow\Updownarrow\uparrow\rightarrow\rightarrow\downarrow\Downarrow\Downarrow\downarrow\rightarrow\uparrow\uparrow\uparrow\rightarrow\downarrow\downarrow\downarrow\rightarrow\rightarrow\rightarrow\uparrow\rightarrow\rightarrow\downarrow\rightarrow\uparrow\uparrow\rightarrow\Downarrow\downarrow\rightarrow$$

über folgende Linie dar:



Die Pfeile $\Updownarrow$ und $\Downarrow$ stellen jeweils Fenster dar. Formal definieren wir *Skylines* induktiv: ε ist eine Skyline, jedes Wort in $\bigcup_{n,m>0}\{\uparrow,\Updownarrow\}^n\{\rightarrow\}^m\{\downarrow,\Downarrow\}^n$ ist eine Skyline, und für Skylines u, v sind $u{\rightarrow}v$, $u{\rightarrow}$ und ${\rightarrow}u$ auch Skylines.

Alle PDAs in dieser Aufgabe akzeptieren über Finalzustände.

(a) Konstruieren Sie einen PDA, der genau die Skylines akzeptiert, die folgenden Einschränkungen erfüllen:

- Alle Gebäude haben ungerade Höhe und Fenster befinden sich nur in ungeraden Stockwerken. (Das linke Haus in obigem Beispiel erfüllt dies.)
- Die Fenster auf gegenüberliegenden Seiten des Hauses müssen übereinstimmen, und alle Häuser müssen mindestens Breite 2 haben.

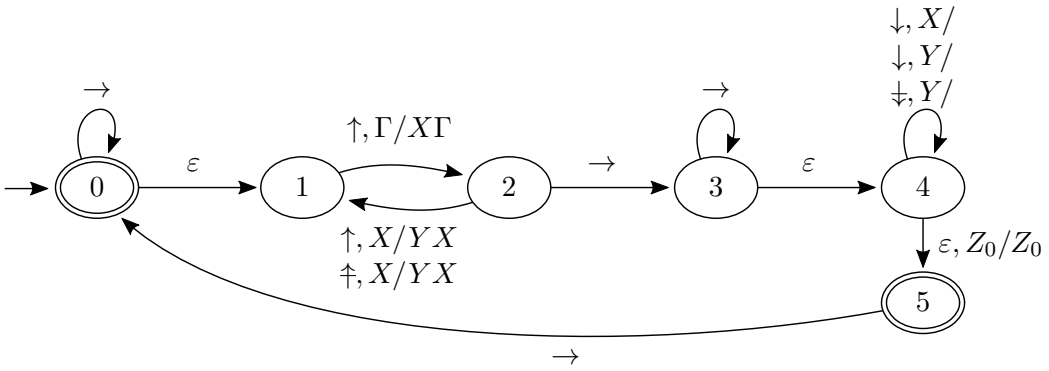
Hinweise: Sie dürfen die (b) verwenden. Falls Sie dies tun, ist es nicht notwendig, M auszurechnen – M_1 und M_2 genügen. Um Ihre Notation zu vereinfachen, können Sie alternativ das Alphabet $\Sigma' := \{o, r, u, o', u'\}$ benutzen.

Sie sind mittlerweile leicht genervt von „traditionellen“ Estlinger Bräuchen, und beschließen, der Bebauungsplanerstellungskommission ein Werkzeug an die Hand zu geben, mit dem sie sich entsprechende PDAs auch selbst zusammenstellen kann.

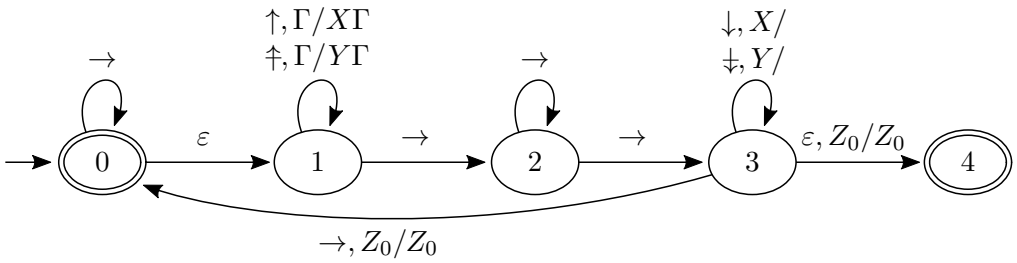
Sei $f : \Sigma \to \mathbb{N}$ eine Funktion und $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ ein PDA. Wir sagen, dass M *Signatur* f hat, wenn für alle $(q, c, Z) \in Q \times \Sigma \times \Gamma$ sowohl $|\alpha| = f(c)$ für alle $(q', \alpha) \in \delta(q, c, Z)$ gilt, also auch $|\alpha| = 1$ für alle $(q', \alpha) \in \delta(q, \epsilon, Z)$. In anderen Worten sorgt jede Transition, die ein Zeichen $c \in \Sigma$ einliest, dafür, dass genau $f(c) - 1$ Zeichen auf den Stapel geschrieben werden, und ϵ-Transitionen dürfen die Größe des Stapels nicht verändern.

(b) Seien M_1, M_2 beliebige PDAs, die beide Signatur f haben. Konstruieren Sie einen PDA M mit $L_F(M) = L_F(M_1) \cap L_F(M_2)$, und begründen Sie, wieso Ihre Konstruktion funktioniert.

Lösungsskizze. **(a)** Zuerst konstruieren wir uns einen PDA für die erste Einschränkung:



Und nun einen für die zweite:



Beide PDAs haben die Signatur f mit $f(\rightarrow) = 1$, $f(\uparrow) = f(\Uparrow) = 2$ und $f(\downarrow) = f(\Downarrow) = 0$. Über (b) konstruieren wir dann einen PDA für ihren Schnitt.

(b) Diese Konstruktion ist ähnlich zur Produktkonstruktion für DFAs. Wir müssen jedoch auch die Stapel beider PDAs gleichzeitig simulieren, und haben nur einen Stapel zur Verfügung. Hier benutzen wir, dass die PDAs M_1 und M_2 die gleiche Signatur haben, was dazu führt, dass ihre Stapel stets gleich groß sind.

Formal: Seien $M_1 = (Q_1, \Sigma, \Gamma_1, q_{01}, Z_{01}, \delta_1, F_1)$ und $M_2 = (Q_2, \Sigma, \Gamma_2, q_{02}, Z_{02}, \delta_2, F_2)$ zwei PDAs mit Signatur f. Aus technischen Gründen passen wir M_1 und M_2 zuerst so an, dass es in jedem Zustand eine ϵ-Transition zu sich selbst gibt. Für $i \in \{1, 2\}$ und $q \in Q_i, X \in \Gamma_i$ fügen wir also $qX \xrightarrow{\epsilon} qX$ in δ_i ein. $(*)$

Wir definieren $Q := Q_1 \times Q_2, \Gamma := \Gamma_1 \times \Gamma_2, q_0 := (q_{01}, q_{02}), Z_0 := (Z_{01}, Z_{02})$ und $F := F_1 \times F_2$. Sei $a \in \Sigma \cup \{\varepsilon\}$. Für jede Transition $(q_1 X_1 \xrightarrow{a} r_1 \alpha) \in \delta_1$ und $(q_2 X_2 \xrightarrow{a} r_2 \beta) \in \delta_2$ gilt $|\alpha| = f(a) = |\beta|$ (da M_1 und M_2 beide Signatur f haben). Wir setzen also $\alpha_1 ... \alpha_k := \alpha$ und $\beta_1 ... \beta_k := \beta$, und fügen folgende Transition zu δ hinzu.

$$(q_1, q_2)(X_1, X_2) \xrightarrow{a} (r_1, r_2)(\alpha_1, \beta_1)(\alpha_2, \beta_2) \cdots (\alpha_k, \beta_k)$$

Aufgrund der Definition von δ wird ein Schritt von M wird immer aus gültigen Schritten von M_1 und M_2 bestehen. Falls M in einen Endzustand übergeht, sind sowohl M_1 als auch M_2 in einem Endzustand, das Wort muss also in beiden Sprachen liegen. Andererseits, wenn sowohl M_1 als auch M_2 für ein gegebenes Wort w mit irgendwelchen Übergängen in einen Endzustand können, können wir diese Übergänge kombinieren und in M ausführen, damit wir also auch alle Wörter im Schnitt von M_1 und M_2 akzeptieren. Falls nur einer der PDAs eine ϵ-Transition ausführt, kann der andere eine der nutzlosen Transitionen, die wir in $(*)$ hinzugefügt haben, ausführen.

Wichtig! Wir wissen aus der Vorlesung, dass kontextfreie Sprachen nicht unter Durchschnitt abgeschlossen sind, deswegen funktioniert diese Konstruktion (wie vorher mehrmals erwähnt) nur wegen der Bedingung an die Signatur! In anderen Fällen ist ein Argument, dass wir einen PDA für den Durchschnitt bauen können, inkorrekt.

Bonusaufgabe H8.5. (*Semilineare Mengen*) 2+2 Bonuspunkte

Sei $n \in \mathbb{N}$ beliebig. Wir nennen eine Menge $M \subseteq \mathbb{N}^n$ *linear*, wenn es Vektoren $r, p_1, ..., p_k \in \mathbb{N}^n$ gibt, sodass

$$M = \{r + \lambda_1 p_1 + ... + \lambda_k p_k : \lambda_1, ..., \lambda_k \in \mathbb{N}\}$$

Eine Menge S ist *semilinear*, wenn $S = M_1 \cup ... \cup M_l$, wobei $M_1, ..., M_l \subseteq \mathbb{N}^n$ lineare Mengen sind, und $l \in \mathbb{N}_{\geq 0}$. Sei nun $n := 2$ und $S \subseteq \mathbb{N}^2$ eine beliebige semilineare Menge.

Zeigen Sie, dass $L := \{w \in \{a, b\}^* : (|w|_a, |w|_b) \in S\}$ kontextfrei ist, indem Sie

(a) geeignet einen PDA M konstruieren und

(b) die Korrektheit von M beweisen.

Lösungsskizze. **(a)** Seien $M_1, ..., M_l$ lineare Mengen mit $S = M_1 \cup ... \cup M_l$ sind (siehe Definition von semilinearen Mengen). Es gilt

$$L = \{w \in \{a, b\}^* : (|w|_a, |w|_b) \in \bigcup_i M_i\} = \bigcup_i \{w \in \{a, b\}^* : (|w|_a, |w|_b) \in M_i\}$$

Insbesondere genügt es also einen PDA für den Spezialfall, dass S eine lineare Menge ist, zu konstruieren. Für den allgemeinen Fall bilden wir dann die Vereinigung von endlich vielen PDAs.

Sei also nun S linear. Wir können also $r, p_1, ..., p_k \in \mathbb{N}^2$ wählen, sodass

$$S = \{r + \lambda_1 p_1 + ... + \lambda_k p_k : \lambda_1, ..., \lambda_k \in \mathbb{N}\}$$

gilt. Wir zeigen, dass L kontextfrei ist, indem wir einen PDA für L erstellen.

Die Idee des PDAs ist, dass er eine Sequenz an Vektoren $v \in \{p_1, ..., p_k\} \subseteq \mathbb{N}^2$ „rät“. Er merkt sich daraufhin, dass er v_2 zusätzliche b einlesen muss, indem er entsprechende Zeichen in den Keller schreibt, und liest v_1 viele a ein. Wir verwenden den Keller also als Zähler, wobei X jeweils „$+1$“ repräsentiert, und Y „-1“. (Wir üblich sorgen wir dafür, dass sich X und Y nicht vermischen.)

Beispiel: Sei $p_1 := (2, 1)$ und $p_2 := (1, 1)$. Beim Einlesen des Wortes *aabba* geht der Automat folgendermaßen vor:

- Der Automat wird initialisiert.

  Keller: Z_0, Restwort: *aabba*
- Der PDA rät $v = p_1 = (2, 1)$, schreibt X auf den Keller und liest aa ein.

  Keller: XZ_0, Restwort: *bba*
- Der PDA liest zwei b ein und dekrementiert den Zähler zweimal.

  Keller: YZ_0, Restwort: a
- Schließlich rät der PDA $v = p_2 = (1, 1)$, erhöht den Zähler um 1, und liest ein a ein.

  Keller: Z_0, Restwort: ε

Nun definieren wir den Automaten $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ formal. Wir wählen hierfür $P := \max\{v_1, v_2 : v \in \{r, p_1, ..., p_k\}\}$ als den größten Eintrag in einem der Vektoren, die S festlegen. Unsere Zustände sind $Q := \{0, ..., P\} \times \{0, ..., P\}$, wobei Zustand

$(i,j) \in Q$ bedeutet, dass wir i-mal ein a einlesen wollen, und j-mal ein b. Als Kelleralphabet verwenden wir $\Gamma := \{Z_0, X, Y\}$, der Startzustand ist $q_0 := r$ und die Menge an akzeptierenden Zuständen $F := \emptyset$, da wir mit leerem Keller akzeptieren wollen.

Es verbleibt, die Übergangsfunktion δ zu beschreiben. Zuerst kümmern wir uns darum, dass wir die korrekte Anzahl an b einlesen: Wenn wir in einem Zustand $(i,j) \in Q$ sind mit $j > 0$, wollen wir den Zählen um j erhöhen. Dafür erhöhen wir den Zähler um 1 und gehen zu Zustand $(i, j-1)$.

$$\begin{aligned} &(i,j)Z_0 \xrightarrow{\varepsilon} (i,j{-}1)XZ_0 \\ &(i,j)X \xrightarrow{\varepsilon} (i,j{-}1)XX \qquad\qquad \text{für } i \in \{0,...,P\} \text{ und } j \in \{1,...,P\} \\ &(i,j)Y \xrightarrow{\varepsilon} (i,j{-}1)\,\varepsilon \end{aligned} \tag{1}$$

Die Zustände $(i,0) \in Q$ können dann beliebig viele b einlesen, und den Zählen jeweils um 1 senken.

$$\begin{aligned} &(i,0)Z_0 \xrightarrow{b} (i,0)YZ_0 \\ &(i,0)X \xrightarrow{b} (i,0)\,\varepsilon \qquad\qquad \text{für } i \in \{0,...,P\} \\ &(i,0)Y \xrightarrow{b} (i,0)YY \end{aligned} \tag{2}$$

Um die richtige Anzahl an a einzulesen benutzen wir den Zustand direkt.

$$(i,0) \xrightarrow{a} (i{-}1,0) \qquad \text{für } i \in \{1,...,P\} \tag{3}$$

Hier verwenden wir eine Kurzschreibweise und notieren $q \xrightarrow{c} s$ für $q, s \in Q$, $c \in \Sigma \cup \{\varepsilon\}$, anstatt $qW \xrightarrow{c} sW$ für jedes $W \in \Gamma$ zu schreiben.

Sobald alle a eingelesen sind, wir also den Zustand $(0,0)$ erreicht haben, kann der Automat sich einen beliebigen Vektor $v \in \{p_1, ..., p_k\}$ aussuchen und zu dem entsprechenden Zustand übergehen.

$$(0,0) \xrightarrow{\varepsilon} v \qquad \text{für } v \in \{p_1, ..., p_k\} \tag{4}$$

Schließlich kann der PDA akzeptieren, wenn er in Zustand $(0,0)$ im Keller nur Z_0 stehen hat, der Zähler also 0 ist.

$$(0,0)Z_0 \xrightarrow{\varepsilon} (0,0)\,\varepsilon \tag{5}$$

(b) Nun beweisen wir, dass obiger Automat korrekt ist. Wie üblich, teilen wir diesen Beweis in zwei Richtungen auf und zeigen $L(M) \subseteq L$ und $L(M) \supseteq L$. Für die folgenden Beweise ist ein wenig Notation hilfreich, wir definieren also $f : \Sigma^* \to \mathbb{N}^2$ als $f(w) := (|w|_a, |w|_b)$ und $g : \Gamma^* \to \mathbb{N}$ als $g(\alpha) := |w|_X - |w|_Y$. Außerdem schreiben wir $e_2 := (0,1)$.

„$\subseteq$“: Sei $w \in \Sigma^**$ beliebig. Wir zeigen, dass für jede Konfiguration $C := (q, w', \alpha)$ von M, die von der Startkonfiguration (r, w, Z_0) erreicht werden kann, $H(C) := q + e_2 g(\alpha) + f(w) - f(w') \in S$ gilt. Aus dieser Aussage folgt $L(M) \subseteq L$ direkt, da ein Wort nur in einer Konfiguration $C^* := ((0,0), \varepsilon, \varepsilon)$ akzeptiert wird, und $H(C^*) = f(w) \in S$ impliziert $w \in L$. Der Beweis erfolgt mithilfe von Induktion über die Anzahl ausgeführter Transitionen.

Für die Induktionsbasis erhalten wir $C = (q, w', \alpha) = (r, w, Z_0)$, und somit $H(C) = r \in S$. Für den Induktionsschritt sei nun $C = (q, u, \alpha)$ eine beliebige Konfiguration

mit $H(C) \in S$, und $C' = (q', u', \alpha')$ eine Konfiguration mit $C \to_M C'$. Nun müssen wir lediglich alle Transitionen durchgehen: (1) inkrementiert $g(\alpha)$ und senkt q_2 um 1, (2) dekrementiert $g(\alpha)$ und erhöht $-f(u)_2$ um 1, (3) reduziert q_1 und vergrößert $-f(u)_1$, und (5) ändert keinen der Terme. In diesen Fällen gilt also (mit Induktionsannahme)

$$H(C') = q' + e_2 g(\alpha') + f(w) - f(u') = q + e_2 g(\alpha) + f(u) - f(w) = H(C) \in S$$

Es bleibt noch Transition (4), hier gilt $q = (0, 0)$ und $q' \in \{p_1, ..., p_k\}$ und somit:

$$H(C') = H(C) + q' \in S$$

Hier verwenden wir wieder die Induktionsannahme, und $x + v \in S$ für alle $x \in S$ und $v \in \{p_1, ..., p_k\}$.

„$\supseteq$“: Wir zeigen zunächst $L \cap L(a^*b^*) \subseteq L(M)$. Hierzu beobachten wir, dass jede Konfiguration $C = (q, a^{q_1}w, \alpha)$, mit $q \in Q$, $w \in \Sigma^*$, $\alpha \in \Gamma^*$ eine Konfiguration $((0, 0), w, \alpha')$ mit $g(\alpha') = g(\alpha) + q_2$ erreichen kann, indem wir von C aus q_2-mal Transition (1) ausführen, und q_1-mal Transition (3).

Sei nun $x \in S$ beliebig. Wir können also eine Folge $v^{(1)}, v^{(2)}, ..., v^{(m)} \in \{p_1, ..., p_k\}$ wählen, sodass $x = r + v^{(1)} + ... + v^{(m)}$. Für $\lambda := r_1 + v_1^{(1)} + ...v_1^{(m)}$ wenden wir nun unsere Beobachtung m-mal an:

$$\begin{aligned}
& (r, && a^{x_1} && b^{x_2}, && Z_0) \\
= \; & (r, && a^{r_1+v_1^{(1)}+...v_1^{(m)}} && b^{r_2+v_2^{(1)}+...v_2^{(m)}}, && Z_0) \\
\to_M^* \; & ((0,0), && a^{v_1^{(1)}+...v_1^{(m)}} && b^{r_2+v_2^{(1)}+...v_2^{(m)}}, && X^{r_1}Z_0) \\
\to_M^* \; & ((0,0), && a^{v_1^{(1)}+...v_1^{(m)}} && b^{r_2+v_2^{(1)}+...v_2^{(m)}}, && X^{r_2}Z_0) \\
\to_M^* \; & (v^{(1)}, && a^{v_1^{(1)}+...v_1^{(m)}} && b^{r_2+v_2^{(1)}+...v_2^{(m)}}, && X^{r_2}Z_0) \\
\to_M^* \; & ((0,0), && a^{v_1^{(2)}+...v_1^{(m)}} && b^{r_2+v_2^{(1)}+...v_2^{(m)}}, && X^{r_2+v_2^{(1)}}Z_0) \\
\to_M^* \; & \cdots \\
\to_M^* \; & ((0,0), && && b^{r_2+v_2^{(1)}+...v_2^{(m)}}, && X^{r_2+v_2^{(1)}+...v_2^{(m)}}Z_0)
\end{aligned}$$

Anschließend müssen wir lediglich Transition (4) x_2-mal anwenden und können dann das Wort mit Transition (5) akzeptieren. Insbesondere gilt damit $a^{x_1}b^{x_2} \in L(M)$ für jedes $x \in S$, und somit $L \cap L(a^*b^*) \subseteq L(M)$.

Um den Beweis abzuschließen, zeigen wir, dass für $w \in L(M)$ mit $ab \in w$ auch $w' \in L(M)$ gilt, wobei w' das Wort ist, das wir erhalten, wenn wir ein ab in w durch ba ersetzen. Da sich ein a nur über Transition (3) einlesen lässt, muss der PDA vor Einlesen des a in einem Zustand der Form $(i, 0)$ sein, von dem aus wir Transition (2) ausführen können (ohne den Zustand zu ändern). Da wir den Keller aus (auch negativen) Zähler verwenden, können wir die Transition (2), die b einliest, nach vorne verschieben, und die übrigen Transitionen lassen sich immer noch ausführen. Am Ende der Ausführung (vor Transition (5)) muss der Zähler dann den gleichen Wert anzeigen, und wir können w' ebenfalls über Transition (5) akzeptieren.