Musterlösung Blatt 6

Aufgabe 1

a) $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$

$\overline{L} = \{a^n b^m c^k \mid n \neq m \text{ oder } m \neq k\} \cup \overline{a^* . b^* . c^*}$

$= \{a^n b^m c^k \mid n \neq m\} \cup \{a^n b^m c^k \mid m \neq k\} \cup \overline{a^* . b^* . c^*}$

- Es reicht zu zeigen, dass jede der 3 Sprachen Kontextfrei ist.
- Wir zeigen die Kontextfreiheit der ersten Sprache.
  ↳ Aufgrund der Symmetrie ist die zweite auch Kontextfrei.
  ↳ Die dritte ist sogar regulär.

- Betrachte $L_1 = \{a^n b^m c^k \mid n \neq m\} = \{a^n b^m c^k \mid n < m\} \cup \{a^n b^m c^k \mid n > m\}$

L_1 wird durch folgende Grammatik erzeugt:

$S \to a.A.C \mid B.b.C$ // $a^n b^m / n > m$ oder $a^m b^m / n < m$

$A \to aA.b \mid a.A \mid \varepsilon$ // $a^n b^m \mid n \geq m$

$B \to a.B.b \mid B.b \mid \varepsilon$ // $a^n b^m \mid n \leq m$

$C \to C.C \mid c \mid \varepsilon$ // c^*

Bem: L_1 ließe sich weiterzerlegen wie folgt:

$L_1 = \{a^n b^m \mid n \neq m\} . c^*$

$= (\{a^n b^m \mid n < m\} \cup \{a^n b^m \mid n > m\}) . c^*$

- Aufgrund der Abschlusseigenschaften und Symmetrien reicht es aus die Kontextfreiheit von $\{a^n b^m \mid n < m\}$ zu zeigen.

b) • NFA: Sei $A = (Q, \Sigma, q_0, \rightarrow, Q_F)$ ein NFA

Idee: Wir drehen $\rightarrow$ um zu $\leftarrow$, und starten von den Finalzuständen. Für ~~letzteres~~ letzteres führen wir einen neuen Startzustand q_f ein, der ε-Transitionen zu allen $q \in Q_F$ hat.

~~$A' = (Q \cup \{q_f\}, \Sigma, q_f, \leftarrow, Q_F)$~~

$A' = (Q \cup \{q_f\}, \Sigma, q_f, \rightarrow', \{q_0\})$ mit

- $q_f \xrightarrow{\varepsilon}' q$, falls $q \in Q_F$
- $q \xrightarrow{a}' q'$, falls $q' \xrightarrow{a} q$

$\Rightarrow L(A') = \text{reverse}(L(A))$ □

• Grammatik: Sei $G = (N, \Sigma, P, S)$ eine CFG.

Idee: Wir drehen alle rechten Seiten von Produktionen um.

$G' = (N, \Sigma, P', S)$ mit

$X \xrightarrow[P']{} \text{reverse}(\alpha)$ gdw. $X \xrightarrow[P]{} \alpha$ $(\alpha \in (\Sigma \cup N)^*)$

• Beobachte, dass $\text{reverse}(\alpha.\beta) = \text{reverse}(\beta).\text{reverse}(\alpha)$ gilt.

↳ Damit folgt letztendlich $L(G') = \text{reverse}(L(G))$. □

• Sei G eine linkslineare Grammatik.

Es gilt $L(G) = \text{reverse}(\text{reverse}(L(G)) = \text{reverse}(L(G')) \in REG$

⇔ ↳ G' ist rechtslinear, also regulär (siehe Blatt 5)

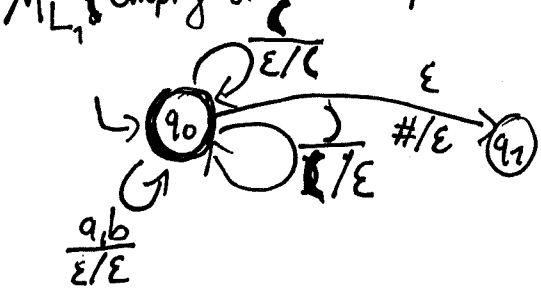
↳ $\text{reverse}(L(G'))$ ist somit auch regulär (siehe oben)

↳ $\text{reverse}(L(G')) = L(G)$ ist regulär.

Aufgabe 2

1. $L_1 = \{w \in \{a, b, (,)\}^* \mid w \text{ ist korrekt geklammert}\}$

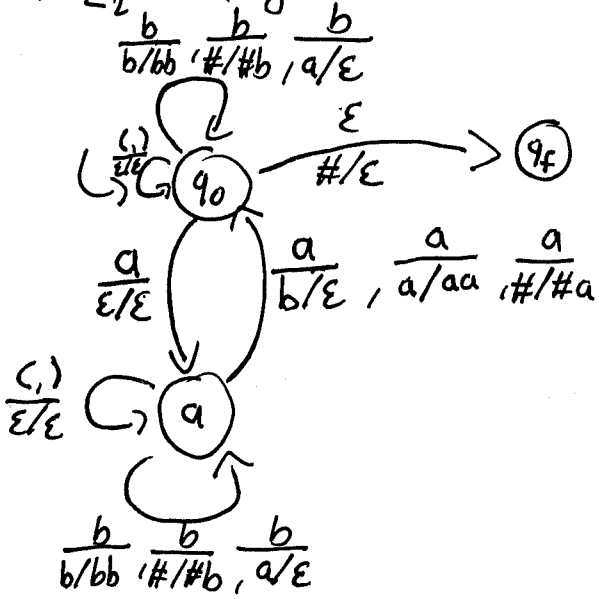
M_{L_1} (empty stack acceptance):



- Stackalphabet = {(}
- Initialsymbol #

2. $L_2 = \{w \in \{a, b, (,)\}^* \mid |w|_a = 2|w|_b\}$

M_{L_2} (empty stack):



- Stackalphabet = {a, b}
  - ↳ a steht für 2 gelesene a's
  - ↳ b steht für 1 gelesenes b.

3. $L = L_1 \cap L_2$

- Es gibt keinen PDA für L (beweisbar durch Pumping Lemma)
- Das intuitive Problem ist, dass sowohl für L_1 ein Stack notwendig ist als auch für L_2. Wir haben nur einen Stack.

Die Berechnungen von M_{L_1} und M_{L_2} lassen sich nicht auf einem einzigen Stack zusammenfassen. Eine Produktkonstruktion scheitert, denn Situationen wo ein Automat pushen will, der andere aber nicht, lassen sich nicht synchronisieren.

Aufgabe 3

G: $S \to aXbXc$
$X \to Y \mid YYY \mid a$
$Y \to bc \mid cb$

① Eliminiere ε-Produktionen (gibt es hier nicht)
② Führe neue NT ein mit $A \to a$, $B \to b$ und $C \to c$
③ Eliminiere Unit-Produktionen (hier ~~$X \to Y$ und $X \to A$~~)

~~$X \to BC \mid CB \mid YYY \mid a$~~

~~④ Ersetze zu lange Produktionen durch kürzere mit neuen NT:~~

$S \to AYBXC \mid AXBXC \mid AYBYC \mid AXBXC$

④ Füge neue Nichtterminale ein, um die Produktionen zu "kürzen":

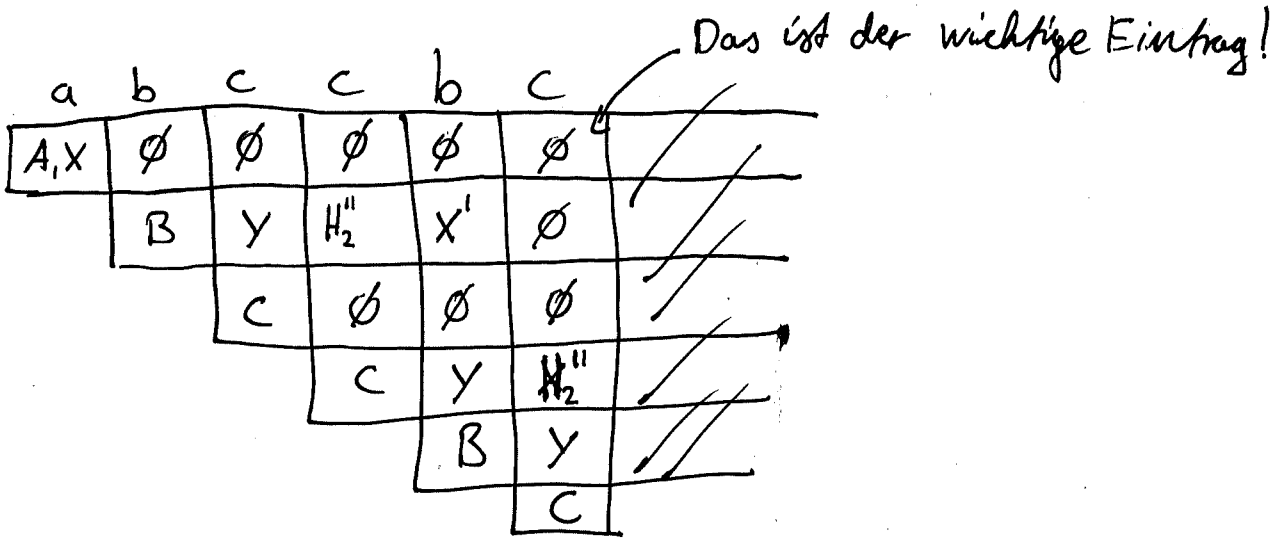
$S \to AH_1 \mid AH_2 \mid AH_3 \mid AH_4$
$H_1 \to YH_1'$, $H_1' \to B.H_1''$, $H_1'' \to XC$
$H_2 \to XH_2'$, $H_2' \to B.H_2''$, $H_2'' \to YC$
$H_3 \to Y.H_2'$ // Wir benutzen die obigen Substitutionen wieder.
$H_4 \to X.H_1'$ //

$X \to Y.X' \mid a$, $X' \to YY$

$Y \to BC \mid CB$

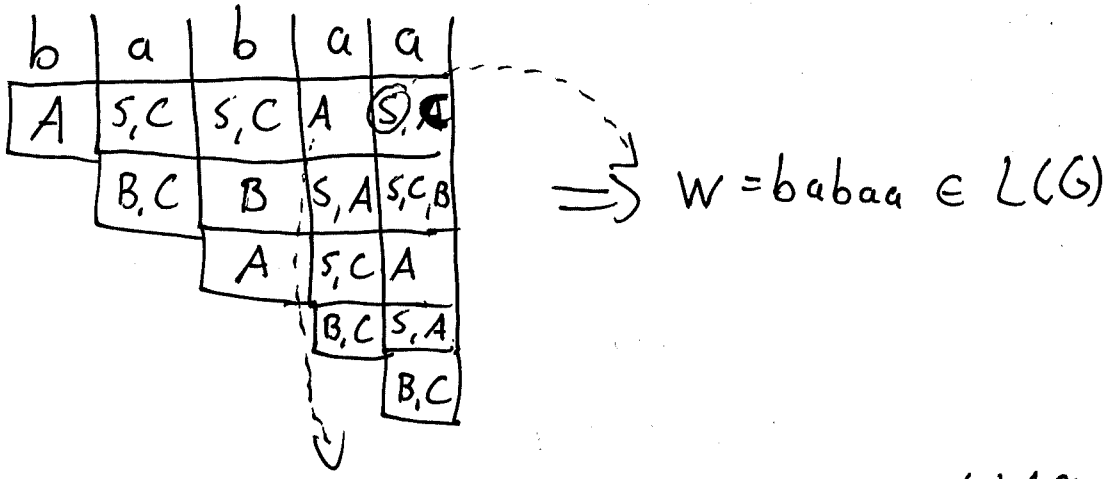
Auf dem nächsten Blatt gehts weiter.

$w = abccbc \in L(G)$?:



$\Rightarrow w \notin L(G)$

b)



$\Rightarrow w = babaa \in L(G)$

Dieselbe Tabelle zeigt auch, dass $w = baba \notin L(G)$ gilt.
(Wir vergessen einfach die letzte Spalte!)

Aufgabe 4

a) $G=(N,T,P,Prog)$ mit

$T=\{id, num, \underline{var}, \underline{if}, \underline{then}, \underline{else}, \underline{end}, \underline{while}, \underline{do}, ;, +, -, /, *, (,), <, >, =, ==\}$

$N=\{E, Prog\}$

und P gegeben durch

$E \rightarrow id \mid num \mid (E+E) \mid (E-E) \mid (E/E) \mid (E*E) \mid (E<E) \mid (E>E) \mid (E==E)$ // Ausdrücke

$Prog \rightarrow \varepsilon \mid \underline{var}\ id; \mid id = E; \mid \underline{if}\ E\ \underline{then}\ Prog\ \underline{end} \mid \underline{if}\ E\ \underline{then}\ Prog\ \underline{else}\ Prog\ \underline{end} \mid \underline{while}\ E\ \underline{do}\ Prog\ \underline{end} \mid Prog\ Prog$ // Programmsyntax

b) Wir fassen mehrere Ableitungsschritte zusammen:

$Prog \rightarrow Prog\ Prog \rightarrow Prog\ Prog\ Prog \rightarrow^{2} Prog^{5}$

~~$\rightarrow^{5}$ var id; id = E; var y; while E do Prog~~

$\rightarrow^{5}$ var id; id = E; var y; id = E; while E do Prog end

$\rightarrow^{4}$ var id; id = num; var y; id = (E-E); while id do Prog Prog end

$\rightarrow^{4}$ var id; id = num; var y; id = (id - num); while id do id = E; id = E; end

$\rightarrow^{2}$ ...; while id do id = (E-E); id = (E+E); end

$\rightarrow^{4}$ ...; while id do id = (id - num); id = (id + num); end

Wir benötigen 23 Ableitungsschritte, um das Programm abzuleiten.

c) Wir zeigen Nichtregularität mit Hilfe des Pumping-Lemmas.

- Sei $p_L \in \mathbb{N}$ beliebig
- Wähle $w = (^{p_L}\,\text{id}\,[+\,\text{num}\,)]^{p_L} \in L(G)$

  (Die eckigen Klammern sind zum Gruppieren. Sie sind nicht Teil des Wortes. Die runden hingegen schon!)

- Sei $w = xyz$ eine beliebige Zerlegung mit
  - $|xy| \leq p_L$
  - $|y| \geq 1$
- Es folgt $y = (^{\ell} \quad (1 \leq \ell \leq p_L)$
- Wähle $k = 0$, dann ist

  $xy^0z = xz = (^{p_L - \ell}\,\text{id}\,[+\,\text{num}\,)]^{p_L}$ kein Wort der Sprache!

  ↳ Mehr schließende Klammern als öffnende.

d) Es gibt viele Modellierungsmöglichkeiten. Die folgende ist nur eine davon.

$G' = (N', T', P', S)$ mit

$T' = T \cup \{$, return, function$\}$

$N' = N \cup \{S, \text{Fun}, \text{Funs}, \text{Params}, \text{Args}, \text{Funbody}\}$

$P' = P$ und folgende Regeln:

- $S \to \text{Funs Prog}$ $\quad \text{Funs} \to \text{Fun Funs} \mid \varepsilon$ // Funktionsliste
- Fun → function id (Params) Funbody end // Einzelne Funktion
- Params → ε | var id | var id, Params // Parameterliste
- Prog → id (Args); // Function call als Statement
- E → id (Args) // Function call als Expression
- Args → ε | E | E, Args // Argumentliste
- Funbody → return; | return E; | ... ← alles was Prog hat