# GTI Übungsblatt 5
# Tutor: Marko Schmellenkamp
# ID: MS1
# Übung: Mi 16-18

Max Springenberg, 177792

## 5.1

### 5.1.1

Eine mögliche Lösung die die Grammatik $G$ mit:

$$\begin{array}{lcll} S & \to & \epsilon & |S' \\ S' & \to & D & |H \\ D & \to & aDbb & |abb \\ H & \to & aaHb & |aab \end{array}$$

Ein Beweis war nicht erforderlich, die Motivation hinter der Struktur der Grammatik ist:

Ein Wort kann entweder halb oder doppelt soviele $b$'s, wie $a$'s haben.
$L(G)$ enthält das leere wort, da $|\epsilon| = 0$ gilt und 0 unter Multiplikation idempotent ist ($2 * \#_a(\epsilon) = 2 * 0 = \#_b(\epsilon)$)
Der Fall für doppelt soviele $b$'s wie $a$'s ist durch die Variable $D$ abgedeckt, der Fall für halb soviel durch die Variable $H$.
Von dem Startsymbol aus kann nur genau eine, oder keine der beiden Variablen erreicht werden, die Ableitungsbäume der Variablen haben keine Gemeinsamen Variablen.

### 5.1.2

Eine mögliche Lösung ist die Grammatik $G$ mit:

$$\begin{array}{lcllllll} V & \to & str & |num & |true & |false & |null & |A & |O \\ O & \to & \{\} & |\{V_o\} \\ V_o & \to & str : V & |V_o, V_o \\ A & \to & [] & |[V_a] \\ V_a & \to & V & |V_a, V_a \end{array}$$

, mit dem Startsymbol $V$.
Ein Beweis wurde nicht gefordert, die Motivation hinter der Konstruktion der Grammatik ist:

Es sollten genau die gültigen JSON *values* beschrieben werden, damit ist das leere Wort $\epsilon$ nicht in der Sprache $L(G)$, da anstelle dessen in JSON *null* verwendet wird.
Ein *value* kann aus einem der Terminalsymbolen $num, str, truefalse, null$ oder einem Array oder Objekt bestehen.
Neben den trivialen Regeln für die Terminalsymbole ergeben sich die Variablen O, für Objekte, und A, für Arrays, durch die vorgeschriebene Syntax.
Werte in einem Array sind durch je ein Komma getrennt und können sämtliche *values* enthalten.
Wertepaare in einem Objekt haben *str* als Schlüssel, ein *value* als Wert und sind wie auch im Array durch je einem Komma getrennt.

## 5.2

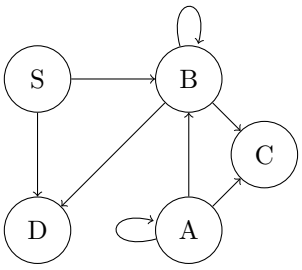
### 5.2.1

Die Menge $V_e$ ergibt sich zu:
$V_e = \{C, D, S, A, , B\}$
Nicht erzeugende Variablen sind: $E$

Die Menge wurde gebildet, indem zuerst alle Variablen mit regeln, die eine Regel mit nur Terminalsymbolen enthalten hinzugefügt. Darauf folgend solange Variablen, die Regeln mit nur Variablen aus der Menge enthalten, hinzugefügt, bis keine mehr gefunden werden.

$E$ und Alle Regeln, die $E$ enthalten werden aus der Grammatik $G_0$ entfernt.

Der Erreichbarkeitsgraph, der erzeugenden Variablen ergibt sich zu:




nicht erreichbare Variablen sind: $A$

Dies kann z.B. durch BFS oder DFS auf der Startsymbol S im Graphen ermittelt werden.

A und alle Regeln, die A enthalten werden entfernt.

Daraus ergibt sich die Grammatik $G_0$ mit:

| | | | | |
|---|---|---|---|---|
| $S$ | $\rightarrow$ | $Bb$ | $\vert Da$ | |
| $B$ | $\rightarrow$ | $bBD$ | $\vert Bb$ | $\vert C$ |
| $C$ | $\rightarrow$ | $c$ | $\vert D$ | $\vert B$ |
| $D$ | $\rightarrow$ | $a$ | | |

**5.2.2**

CNF 2:
Die Grammatik $G'$ ergibt sich nach CNF2 zu:

| | | | | |
|---|---|---|---|---|
| $S$ | $\rightarrow$ | $BBW_b$ | $\vert W_bW_c$ | |
| $A$ | $\rightarrow$ | $B$ | $\vert AW_a$ | $\vert W_c$ |
| $B$ | $\rightarrow$ | $BABAW_a$ | $\vert BW_b$ | $\vert C$ |
| $C$ | $\rightarrow$ | $W_c$ | $\vert A$ | $\vert B$ |

Dabei wird für jedes vorkommende Terminalsymbol $\sigma$ eine Variable $W_\sigma$ eingeführt, die als einzige Regel das jeweilige Terminalsymbol enthält. Danach werden alle vorherigen Vorkommnisse des Terminalsymbols durch die jeweilige Variable ersetzt.

CNF 3:
die Grammatik $G_1$ ergibt sich nach CNF3 zu:

$$
\begin{array}{lllll}
S & \to & BS_1 & |W_bW_c & \\
S_1 & \to & BW_b & & \\
A & \to & B & |AW_a & |W_c \\
B & \to & BB_1 & |BW_b & |C \\
B_1 & \to & AB_2 & & \\
B_2 & \to & BB_3 & & \\
B_3 & \to & AW_a & & \\
C & \to & W_c & |A & |B \\
W_a & \to & a & & \\
W_b & \to & b & & \\
W_c & \to & c & &
\end{array}
$$

Dabei werden die rechten Seiten so verkürzt, dass die Bedingung gilt, dass nur je zwei Variablen konkateniert vorkommen. Dies geschieht durch einfügen von Variablen, die Teilregeln ersetzen unter dieser Bedingung.

### 5.2.3

CNF4: Die Grammatik $G'$ ergibt sich nach CNF4 zu:
$V' = \{C, B, A\}$

$$
\begin{array}{lllllll}
S & \to & W_bA & |W_b & |BW_c & |W_c & \\
A & \to & B & |AW_a & |W_a & |W_c & \\
B & \to & BW_a & |W_a & |BW_b & |W_b & |C \\
C & \to & W_c & |A & |B & & \\
W_a & \to & a & & & & \\
W_b & \to & b & & & & \\
W_c & \to & c & & & &
\end{array}
$$

Die Menge $V'$ ergibt sich durch das Hinzufügen von Variablen mit Regeln die das Terminalsymbol $\epsilon$ enthalten und anschließend Variablen mit Regeln die zu Variablen führen, die in der Menge $V'$ enthalten sind, bis keine mehr gefunden werden.
Anschließend werden alle regeln mit $\epsilon$ entfernt und für jede Regel mit Variablen aus $V'$ neue Regeln ohne diese Variablen angefügt.

CNF5:
Die Grammatik $G_3$ ergibt sich nach CNF5 zu:
$U = \{(A, B), (A, W_a), (A, W_c), (A, W_b), (A, C)$
$, (B, W_a), (B, W_b), (B, C), (B, W_c), (B, A)$
$, (C, W_c), (C, A), (C, W_a), (C, B), (C, W_b)\}$

$$
\begin{array}{lllllllll}
S & \to & W_bA & |b & |BW_c & |c & & & \\
A & \to & BW_a & |a & |BW_b & |b & |c & |A & |AW_a \\
B & \to & BW_a & |a & |BW_b & |b & |c & |AW_a & \\
W_a & \to & a & & & & & & \\
W_b & \to & b & & & & & & \\
W_c & \to & c & & & & & &
\end{array}
$$

Die Menge U ergibt sich durch Variablen, die in einer Regel auf nur eine Variable verweisen, die nicht gleich ihnen selbst ist.

## 5.3

### 5.3.1

gegeben:
Grammatik $G$ mit:
$S \to aSbb|abb$

Sprache $L$ mit:
$L = \{a^n b^m | n, m \in \mathbb{N} \wedge m \geq 2n\}$

z.z.: $\forall w \in L(G) : w = a^n b^m, n, m \in \mathbb{N}, m \geq 2n)$

Wir führen eine Induktion über die Wortlänge $n = |w|$, mit der Menge der Wortlängen von $L(G)$:

$$N_L = \{3n | n \in \mathbb{N}\}$$

Ich nehme an, dass ich für $N_L$ keinen Induktionsbeweis führen muss, da die Wortlänge aller Wörter aus der Sprache der Grammatik mit der Einzigen Regel $S \to abb|aSbb$ gleich 3 beim kleinsten, 3+3 beim nächst kleinsten Element ist und S induktiv definiert ist.
Es geht mehr darum $k$ aus einer sinnvolleren Menge als $\mathbb{N}$ zu wählen.

Aussage:

$$\forall k \in N_L : w = a^n b^m, w \in L(G), |w| = n :, (n, m \in \mathbb{N}, m \geq 2n)$$

I.A.

$k = 3$, da $abb$ kleinstes Element der Sprache L(G) ist

$\nexists w \in L(G) : w \not\equiv abb \wedge |w| = 3$ , da $S \Rightarrow^G abb$ die einzige Ableitung für Wörter der Länge 3 ist.

Für $w = abb$:
$\#_a(w) = 1, \#_b(w) = 2$
$2 = 2 * 1$
damit gilt auch $2 \geq 2 * 1$

Dadurch wurde gezeigt, dass die Aussage für k=3 gilt.

I.V.
Die Aussage gelte für $k' \in N_L$ beliebig, aber fest.

I.S.

$k = k' + 3$

definiert seien:
$w_k$, mit: $|w| = k, w_k \in L(G)$
$w_{k'}$, mit: $|w| = k', w_{k'} \in L(G)$

nach der Ableitungsregel von $S$ gilt:
$w_k = aw_{k'}bb$

nach der I.V. gilt:
$\#_b(w_{k'}) \geq 2 * \#_a(w_{k'})$

Daraus folgt:
$2 * \#_a(w_k) = 2 * (1 + \#_a(w_{k'})) = 2 + 2 * \#_a(w_{k'}) \overset{I.V.}{\leq} 2 + \#_b(w_{k'}) = \#_b(w_k)$

Damit wurde die Aussage für beliebige $k \in N_L$ gezeigt.

**5.3.2**

Annahme $L \subseteq L(G)$:

Daraus würde folgen:
$\forall w \in L : w \in L(G)$

$w \overset{\text{def}}{=} abbb$
$w \in L, w \notin L(G) \lightning$

w ist nicht in $L(G)$, da $L(G)$ keine Wörter der Länge 4 enthält.

Damit gilt die Aussage nicht.