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:

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

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:

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

, 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 ϵ 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

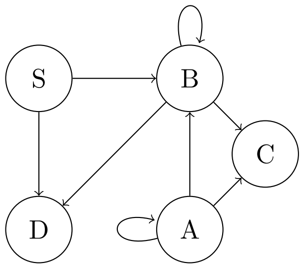
5.2.1

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, die eine Regel mit nur Terminalsymbolen enthalten hinzugefügt. Darauf folgend werden solange Variablen, die Regeln mit nur Variablen aus der Menge V_e 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 ausgehend vom Startsymbol S im Graphen ermittelt werden.

A und dessen Regeln werden entfernt.

Daraus ergibt sich die Grammatik G_0 mit:

$$\begin{array}{lllll} S & \rightarrow & Bb & |Da & \\ B & \rightarrow & bBD & |Bb & |C \\ C & \rightarrow & c & |D & |B \\ D & \rightarrow & a & & \end{array}$$

5.2.2

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

$$\begin{array}{lllll} S & \rightarrow & BBW_b & |W_bW_c & \\ A & \rightarrow & B & |AW_a & |W_c \\ B & \rightarrow & BABAW_a & |BW_b & |C \\ C & \rightarrow & W_c & |A & |B \end{array}$$

Dabei wird für jedes vorkommende Terminalsymbol σ eine Variable W_σ 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:

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

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 Konkatenationen durch Regeln ersetzen unter Aufrecherhaltung der Bedingung.

5.2.3

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

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

Die Menge V' ergibt sich durch das Hinzufügen von Variablen mit Regeln die das Terminalsymbol ϵ 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 ϵ entfernt und für jede Regel mit Variablen aus V' auf der rechten Seite neue Regeln ohne diese Variablen angefügt.

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

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

Die Menge U ergibt sich durch Variablen, die in einer Regel auf nur eine Variable verweisen, die nicht gleich ihnen selbst ist.
Abschließend werden die Einheitsregeln durch die von ihnen erreichten nicht-Einheitsregeln ersetzt.
Zusätzlich werden alle nicht erreichbaren Regeln entfernt.

5.3

5.3.1

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

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

Wir betrachten die Aussage:

$\forall k \in \mathbb{N}, w \in L(G)$, mit w ist durch eine Ableitung der Länge k aus G abgeleitet : $w = a^n b^m, (m, n \in \mathbb{N}, m \geq 2n)$

Wir führen einen Induktionsbeweis über die Länge der Ableitung(sschritte) k.

I.A.

$k = 1$
Es existiert genau eine Ableitung der Länge 1 von G mit:

$$S \Rightarrow_G abb$$

, damit muss keine Fallunterscheidung betrachtet werden.
damit muss w gewählt werden mit $w \stackrel{\text{def}}{=} abb$
es gilt:
$\#_a(w) = 1$
$\#_b(w) = 2 \geq 2 * 1 = 2 * \#_a(w)$
Damit gilt die Aussage für k=1

I.V.

Die Aussage gelte für $k' \in \mathbb{N}$, beliebig aber fest.

I.S.

$k = k' + 1$

Die Einzige Regel, die einen vorherigen Ableitungsschritt erlaubt ist:

$$S \to aSbb$$

, damit muss keine Fallunterscheidung betrachtet werden.
Wir betrachten nun das abgeleitete Wort $w \overset{\text{def}}{=} avbb$, mit v als ein Wort, dessen Ableitung die Länge k' hat.
Nach I.V. gilt: $\#_b(v) \geq 2 * \#_a(v)$
Ferner gilt:
$\#_b(w) = 2 + \#_b(v)$
$\#_a(w) = 1 + \#_a(v)$
Und damit dann auch:
$\#_a(w) \leq 2 * \#_a(w) = 2 * (1 + \#_a(v)) = 2 + 2 * \#_a(v)) \overset{I.V.}{\leq} 2 + \#_b(v) = \#_b(w)$

Damit wurde die Aussage auch für $k = k' + 1$ gezeigt und gilt damit für alle $k \in \mathbb{N}$

5.3.2

Annahme $L \subseteq L(G)$:

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

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

w ist in L, da $w \equiv a^n b^m$, mit $n = 1, m = 3, m \geq 2n$ gilt.
w ist nicht in $L(G)$, da G $abbb$ nicht ableiten kann.
Offensichtlich ist die Wortlänge bei steigender Ableitungslänge monoton steigend.

Ableitung der Länge 1:
$S \Rightarrow_G abb$
$abbb \not\equiv abb, |abbb| > |abb|$

Ableitung der Länge 2:
$S \Rightarrow_G aSbb$
$\Rightarrow_G aabbbb$
$abbb \not\equiv aabbbb, |abbb| < |aabbbb|$

Damit gilt $abbb \notin L(G)$

Damit gilt die Aussage nicht.