Lehrstuhl für Softwaretechnik und Programmiersprachen
Professor Dr. Michael Leuschel

Heinrich Heine Universität Düsseldorf

Compilerbau - Wintersemester 2021/22

Übungsblatt 3 - Musterlösung

Aufgabe 3.1

1. (a)

$$S \vdash (L) \vdash (L,S) \vdash (L,S,S) \vdash (S,S,S) \vdash ((L),S,S) \vdash ((L,S),S,S)$$
$$\vdash ((S,S),S,S) \vdash ((a,S),S,S) \vdash ((a,a),S,S) \vdash ((a,a),a,S)$$
$$\vdash ((a,a),a,(L)) \vdash ((a,a),a,(S)) \vdash ((a,a),a,(a))$$

(b)

$$S \vdash (L) \vdash (L,S) \vdash (L,(L)) \vdash (L,(S)) \vdash (L,(a)) \vdash (L,S,(a))$$
$$\vdash (L,a,(a)) \vdash (S,a,(a)) \vdash ((L),a,(a)) \vdash ((L,S),a,(a))$$
$$\vdash ((L,a),a,(a)) \vdash ((S,a),a,(a)) \vdash ((a,a),a,(a))$$

(c) Die Grammatik ist eindeutig. Am Anfang gibt es nur zwei Möglichkeiten: Entweder der String besteht nur aus einem a, oder er startet mit einer Klammer, entsprechend ist es eindeutig, welche Produktion zuerst angewandt werden muss. Hat man die Regel $S \vdash (L)$ angewandt, so kann man nun zählen wieviele „Elemente“ diese Liste enthält, deren Klammern man gerade erzeugt hat. Entsprechend oft muss man dann in einer Linksableitung die Regel $L \vdash L,S$ (jeweils auf das führende L) anwenden, bevor man die Regel $L \vdash S$ anwendet, um so auf eine Ableitung der Form

$$S \vdash (L) \vdash (L,S) \vdash \cdots \vdash (L,S,\cdots,S) \vdash (S,S,\cdots,S)$$

zu kommen. Rekursiv kann man dasselbe Argument nun wieder auf die gerade erzeugten S-Nichtterminale anwenden.

(d) Informal beschreibt diese Grammatik eine Sprache von „Listen“, wobei eine Liste entweder a ist oder von der Form $(L_1, L_2, \ldots, L_n)$, wobei die L_i wiederum Listen sind.

2. (a)

$$S \vdash aSbS \vdash aaSbSbS \vdash aabSbS \vdash aabbS \vdash aabbaSbS \vdash aabbabS \vdash aabbab$$

(b)

$$S \vdash aSbS \vdash aSbaSbS \vdash aSbaSb \vdash aSbab \vdash aaSbSbab \vdash aaSbbab \vdash aabbab$$

(c) Die Grammatik ist mehrdeutig. Der String *aabbab* hat etwa neben der in (a) gegebenen Linksableitung noch folgende Linksableitung:

$$S \vdash aSbS \vdash aaSbSbS \vdash aabSbS \vdash aabbSaSbS \vdash aabbaSbS \vdash aabbabS \vdash aabbab$$

(d) Die Grammatik generiert die Sprache aller Wörter in a und b mit gleich vielen as wie bs.

3. (a)

$$\begin{aligned}
bexpr &\vdash bterm \\
&\vdash bfactor \\
&\vdash (bexpr) \\
&\vdash (bexpr \textbf{ or } bterm) \\
&\vdash (bterm \textbf{ or } bterm) \\
&\vdash (bfactor \textbf{ or } bterm) \\
&\vdash ((bexpr) \textbf{ or } bterm) \\
&\vdash ((bexpr \textbf{ or } bterm) \textbf{ or } bterm) \\
&\vdash ((bterm \textbf{ or } bterm) \textbf{ or } bterm) \\
&\vdash ((bfactor \textbf{ or } bterm) \textbf{ or } bterm) \\
&\vdash ((\textbf{true or } bterm) \textbf{ or } bterm) \\
&\vdash ((\textbf{true or } bterm \textbf{ and } bfactor) \textbf{ or } bterm) \\
&\vdash ((\textbf{true or } bfactor \textbf{ and } bfactor) \textbf{ or } bterm) \\
&\vdash ((\textbf{true or false and } bfactor) \textbf{ or } bterm) \\
&\vdash ((\textbf{true or false and true}) \textbf{ or } bterm) \\
&\vdash ((\textbf{true or false and true}) \textbf{ or } bfactor) \\
&\vdash ((\textbf{true or false and true}) \textbf{ or not } bfactor) \\
&\vdash ((\textbf{true or false and true}) \textbf{ or not false})
\end{aligned}$$

(b)

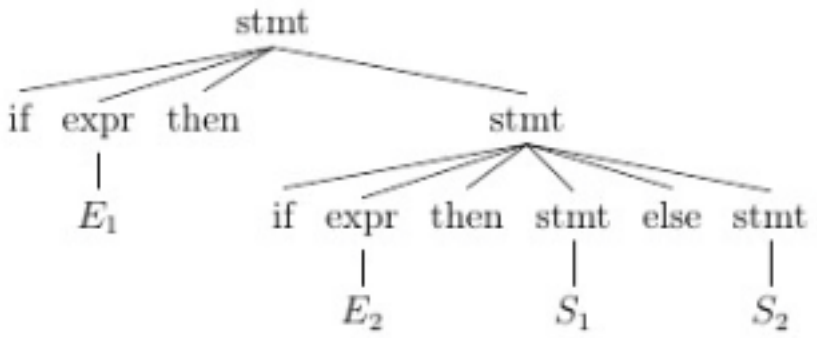
$$
\begin{aligned}
\mathit{bexpr} &\vdash \mathit{bterm} \\
&\vdash \mathit{bfactor} \\
&\vdash (\mathit{bexpr}) \\
&\vdash (\mathit{bexpr}\ \textbf{or}\ \mathit{bterm}) \\
&\vdash (\mathit{bexpr}\ \textbf{or}\ \mathit{bfactor}) \\
&\vdash (\mathit{bexpr}\ \textbf{or not}\ \mathit{bfactor}) \\
&\vdash (\mathit{bexpr}\ \textbf{or not false}) \\
&\vdash (\mathit{bterm}\ \textbf{or not false}) \\
&\vdash (\mathit{bfactor}\ \textbf{or not false}) \\
&\vdash ((\mathit{bexpr})\ \textbf{or not false}) \\
&\vdash ((\mathit{bexpr}\ \textbf{or}\ \mathit{bterm})\ \textbf{or not false}) \\
&\vdash ((\mathit{bexpr}\ \textbf{or}\ \mathit{bterm}\ \textbf{and}\ \mathit{bfactor})\ \textbf{or not false}) \\
&\vdash ((\mathit{bexpr}\ \textbf{or}\ \mathit{bterm}\ \textbf{and true})\ \textbf{or not false}) \\
&\vdash ((\mathit{bexpr}\ \textbf{or}\ \mathit{bfactor}\ \textbf{and true})\ \textbf{or not false}) \\
&\vdash ((\mathit{bexpr}\ \textbf{or false and true})\ \textbf{or not false}) \\
&\vdash ((\mathit{bterm}\ \textbf{or false and true})\ \textbf{or not false}) \\
&\vdash ((\mathit{bfactor}\ \textbf{or false and true})\ \textbf{or not false}) \\
&\vdash ((\textbf{true or false and true})\ \textbf{or not false})
\end{aligned}
$$

(c) Die Grammatik ist eindeutig. Argument ähnlich wie oben.

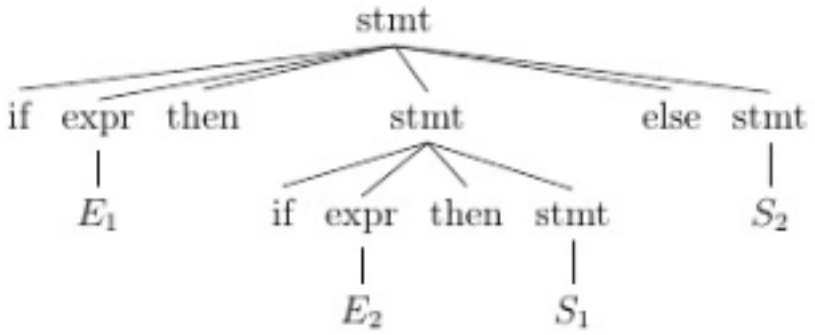
(d) Die Sprache besteht aus allen wohlgeformten Booleschen Ausdrücken, wobei ein Boolescher Ausdruck entweder **true** oder **false** oder von der Form (A), **not** A, A **or** B oder A **and** B ist, wobei A und B wieder Boolesche Ausdrücke sind.

Aufgabe 3.2

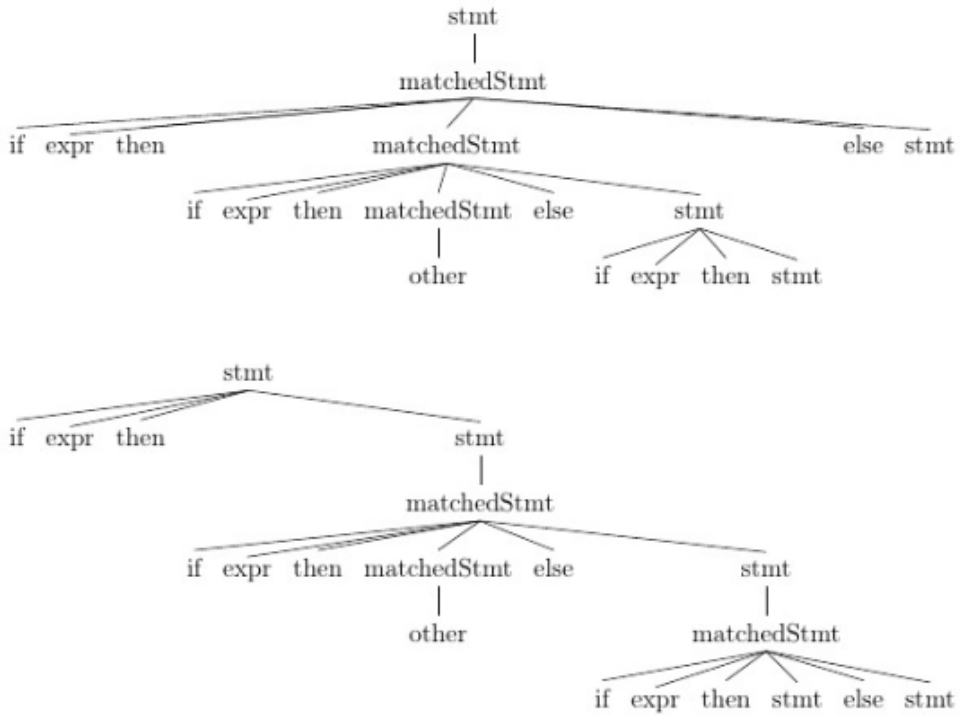
1. Entweder



oder



2. Nicht alle stmt-Knoten in den Blättern wurden noch weiter bis auf **other** runtergebrochen.



Aufgabe 3.3
Die erste Regel der jeweiligen Grammatik ist immer die Startregel.

(a)

$$
\begin{aligned}
U &\to K \ \wedge \ U \mid K \\
K &\to (\ O\ ) \mid N \\
O &\to N \ \vee \ O \mid N \\
N &\to \neg\, V \mid V \\
V &\to \mathbf{v} \mid \mathbf{w} \mid \mathbf{x} \mid \mathbf{y} \mid \mathbf{z}
\end{aligned}
$$

(b)

$$
\begin{aligned}
&U \rightarrow K \ \vee \ U \mid K \\
&K \rightarrow (\ O\ ) \mid N \\
&O \rightarrow N \ \wedge \ O \mid N \\
&N \rightarrow \neg\, V \mid V \\
&V \rightarrow \mathbf{v} N \\
&N \rightarrow N\ D \mid D \\
&D \rightarrow \mathbf{0} \mid \mathbf{1} \mid \mathbf{2} \mid \mathbf{3} \mid \mathbf{4} \mid \mathbf{5} \mid \mathbf{6} \mid \mathbf{7} \mid \mathbf{8} \mid \mathbf{9}
\end{aligned}
$$