Hausaufgabe 2

Von Fabian Wolter, Selin Kabak.

1.10 Komplizierte Partyplanung

1.10.1

- $H \Rightarrow F$
- $G \vee I$
- $F \oplus E$
- $E = G$
- $I \Rightarrow H \wedge G$

1.10.2

H	G	I	F	E	$H \Rightarrow F$	$G \vee I$	$F \oplus E$	$E = G$	$I \Rightarrow H \wedge G$
1	1	1	1	1	1	1	0	1	1
1	1	1	1	0	1	1	1	0	1
1	1	1	0	1	0	1	1	1	1
1	1	1	0	0	0	1	0	0	1
1	1	0	1	1	1	1	0	1	1
1	1	0	1	0	1	1	1	0	1
1	1	0	0	1	0	1	1	1	1
1	1	0	0	0	0	1	0	0	1
1	0	1	1	1	1	1	0	0	0
1	0	1	1	0	1	1	1	1	0
1	0	1	0	1	0	1	1	0	0
1	0	1	0	0	0	1	0	1	0
1	0	0	1	1	1	0	0	0	1
1	0	0	1	0	1	0	1	1	1
1	0	0	0	1	0	0	1	0	1
1	0	0	0	0	0	0	0	1	1
0	1	1	1	1	1	1	0	1	0
0	1	1	1	0	1	1	1	0	0
0	1	1	0	1	1	1	1	1	0
0	1	1	0	0	1	1	0	0	0
0	1	0	1	1	1	1	0	1	1
0	1	0	1	0	1	1	1	0	1
0	1	0	0	1	1	1	1	1	1
0	1	0	0	0	1	1	0	0	1
0	0	1	1	1	1	1	0	0	0
0	0	1	1	0	1	1	1	1	0
0	0	1	0	1	1	1	1	0	0
0	0	1	0	0	1	1	0	1	0

H	G	I	F	E	$H \Rightarrow F$	$G \vee I$	$F \oplus E$	$E = G$	$I \Rightarrow H \wedge G$
0	0	0	1	1	1	0	0	0	1
0	0	0	1	0	1	0	1	1	1
0	0	0	0	1	1	0	1	0	1
0	0	0	0	0	1	0	0	1	1

Nur Georg und Emma können kommen.

1.10.3

1. Hier ist bereits ein Fehler, denn ChatGPT hat hier etwas fehlverstanden. Aus *"Wenn Herr Meier kommt, dann kommt auch seine Frau."* schließt ChatGPT, dass Herr Meier auf jeden Fall kommt und somit auch seine Frau.
2. ChatGPT hat hier wieder den gleichen Fehler gemacht. Es hat gelesen *"Wenn Irvana kommt,..."* und hat daraus geschlossen, dass Irvana auf jeden Fall kommt. Zusätzlich hat ChatGPT keine Ahnung was mindestens heißt. In diesem Fall bedeutet es nicht, dass Georg auf jeden Fall kommt nur weil Irvana kommt.
3. Hier ist ein Folgefehler, da ChatGPT angenommen hat, dass Frau Meier kommt.
4. Ebenfalls ein Folgefehler, da ChatGPT angenommen hat, dass Georg kommt.

1.14

1.14.1

a) Entweder Alice kann in Java Programmieren, oder liebt Scala.

b) Alice, Bob und Charle lieben es alle in Python zu programmieren.

c) Alle Studierenden können in einer Programmiersprache programmieren.

d) Es gibt eine Programmiersprache, welche alle Studierenden können.

e) Es gibt eine Programmiersprache, die von Studierenden geliebt wird.

f) Alle Studierenden lieben alle Programmiersprachen.

1.14.2

a) $\forall_{m \in M} K(m, \text{Java}) \vee K(m, \text{Scala})$

b) $\neg \exists_{m \in M} L(m, \text{C++})$

c) $\neg K(\text{Bob, Java}) \Rightarrow K(\text{Bob, Scala})$

d) $\exists_{p \in P} \forall_{m \in M} K(m, p) \wedge L(m, p)$

e) $\forall_{m \in M} \exists_{p \in P} \neg K(m, p) \wedge \neg L(m, p)$

f) $\neg \exists_{m \in M} \exists_{p \in P} \neg K(m, p) \wedge \neg L(m, p)$

2.3

2.3.1

a) r_1 ist Typ 3, weil ein nonterminales Symbol auf der linken Seite auf ein terminales Symbol und ein nonterminales abbildet.

b) r_2 ist Typ 0, es mehr als ein nonterminales Symbol auf der linken Seite hat und weil die linke Seite länger ist als die rechte Seite.

c) r_3 ist Typ 2, weil ein nonterminales Symbol auf der linken Seite auf ein nonterminales Symbol abbildet.

d) r_4 ist Typ 1, es mehr als ein nonterminales Symbol auf der linken Seite hat und weil gleich viele Symbole von der linken Seite auf die rechte Seite abbilden.

e) r_5 ist Typ 1, es mehr als ein nonterminales Symbol auf der linken Seite hat und weil gleich viele Symbole von der linken Seite auf die rechte Seite abbilden.

f) r_6 ist keine gültige Regel, da die linke Seite keine nonterminalen Symbole enthält.

g) r_7 ist Typ 0, es mehr als ein nonterminales Symbol auf der linken Seite hat und da mehrere nonterminalen Symbole auf ein terminales abbilden.

h) r_8 ist keine gültige Regel, da die linke Seite keine nonterminalen Symbole enthält.

i) r_9 ist Typ 3, siehe r_1.

2.3.2

a) G_1 ist eine korrekte Grammatik mit Chomsky-Typ 3, da $N \to TN$.

b) G_2 ebenfalls.

c) G_3 ist eine korrekte Grammatik mit Chromsky-Typ 0, da bei Regel 2 gilt $TN \to T$.

d) G_4 ist keine korrekte Grammatik, da wir von einem terminalen Symbol nicht ableiten können.

e) G_5 ist keine korrekte Grammatik, da nie auf Regel A abgeleitet werden kann.

f) G_6 ist eine korrekte Grammatik mit Chomsky-Typ 1, da in der ersten Regel $NN \to NNN$.

2.4

2.4.1

a) ω_1: $S \Rightarrow \epsilon$

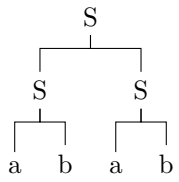
b) ω_2: $S \Rightarrow aSb \Rightarrow ab$

c) ω_3: $S \Rightarrow SS \Rightarrow^* aSbaSb \Rightarrow^* abab$

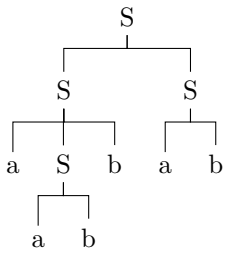
d) ω_4: $S \Rightarrow SS \Rightarrow^* aSbaSb \Rightarrow^* aaSbbab \Rightarrow aabbab$

2.4.2

ω_3:



ω_4:



2.4.3

a) Der Buchstabe c kommt nicht in unserem Alphabet vor.

b) Mit unseren Regeln können wir nur nur ein a erzeugen, wenn ein b folgt, und nur ein b erzeugen, wenn ein a voransteht.

c) Mit unseren Regeln können nur gleich viele a's und b's erzeugt werden.

2.7

2.7.1

a) $S \Rightarrow \epsilon$

b) $S \Rightarrow ABCS \Rightarrow^* abc$

c) $S \Rightarrow ABCS \Rightarrow BACS \Rightarrow^* bac$

d) $S \Rightarrow ABCS \Rightarrow ABCABCS \Rightarrow CBAABCS \Rightarrow^* cbaabc$

2.7.2

$\mathcal{L}(G_2) = \{(xyz)^n | x, y, z \text{ sind verschiedene Buchstaben aus } \Sigma, n \in \mathbb{N}^0\}$