Grundbegriffe der Informatik
Aufgabenblatt 4

Matr.nr.:							

Nachname:

Vorname:

Tutorium: Nr.

Name des Tutors:

Ausgabe: 07. Dezember 2016

Abgabe: 22. Dezember 2016, 16:00 Uhr
im GBI-Briefkasten im Untergeschoss
von Gebäude 50.34

Lösungen werden nur korrigiert, wenn sie

- rechtzeitig,
- in Ihrer eigenen Handschrift,
- mit dieser Seite als Deckblatt und
- in der oberen **linken** Ecke zusammengeheftet

abgegeben werden.

Vom Tutor auszufüllen:

erreichte Punkte

Blatt 4:	/ 36	(4 ECTS: 16)
Blätter 1 – 4:	/ 137,5	(4 ECTS: 106,5)

Mit **[nicht 4ECTS]** gekennzeichnete Aufgaben werden von Studenten, die den 4ECTS Schein der Vorlesung machen wollen, bitte nicht bearbeitet.

Aufgabe 4.1 (2 + 2 + 2 + 2 + 8 = 16 Punkte)
Es sei $G = (N, T, \mathtt{S}, P)$ die Grammatik mit den Nichtterminalsymbolen $N = \{\mathtt{S}, \mathtt{E}, \mathtt{T}, \mathtt{Z}, \mathtt{Q}, \mathtt{C}, \mathtt{D}\}$, den Terminalsymbolen $T = \{\mathtt{0}, \mathtt{1}\}$ und den Produktionen

$$\begin{aligned} P = \{&\mathtt{S} \to \mathtt{E} \mid \mathtt{Z} \mid \mathtt{Q}, \\ &\mathtt{E} \to \mathtt{1E} \mid \mathtt{0T} \mid \epsilon, \\ &\mathtt{T} \to \mathtt{1T} \mid \mathtt{0E}, \\ &\mathtt{Z} \to \mathtt{1Z} \mid \mathtt{0E} \mid \epsilon, \\ &\mathtt{Q} \to \mathtt{CD}, \\ &\mathtt{C} \to \mathtt{0C} \mid \mathtt{0}, \\ &\mathtt{D} \to \mathtt{1D} \mid \mathtt{1}\} \end{aligned}$$

a) Leiten Sie aus dem Startsymbol das Wort 11010111010 ab. Geben Sie dabei jeden Ableitungsschritt an.
b) Leiten Sie aus dem Startsymbol das Wort 11000110110 ab. Geben Sie dabei jeden Ableitungsschritt an.
c) Zeichnen Sie einen Ableitungsbaum für das Wort 0000001111111.
d) Zeichnen Sie einen zweiten Ableitungsbaum für das Wort 0000001111111, der sich von dem Ableitungsbaum aus Teilaufgabe c) unterscheidet.
e) Geben Sie die Sprache $L(G)$ an, die die Grammatik erzeugt.

Lösung 4.1
Die Grammatik besteht aus drei Teilgrammatiken, die mit den Symbolen E, D und Z beginnen.

E Alle Wörter aus $\{0,1\}^*$ mit einer geraden Anzahl an 0 oder das leere Wort.

D Alle Wörter aus $\{0,1\}^*$ mit einer ungeraden Anzahl an 0 oder das leere Wort.

Z Alle Wörter der Form $\{0\}^+\{1\}^+$.

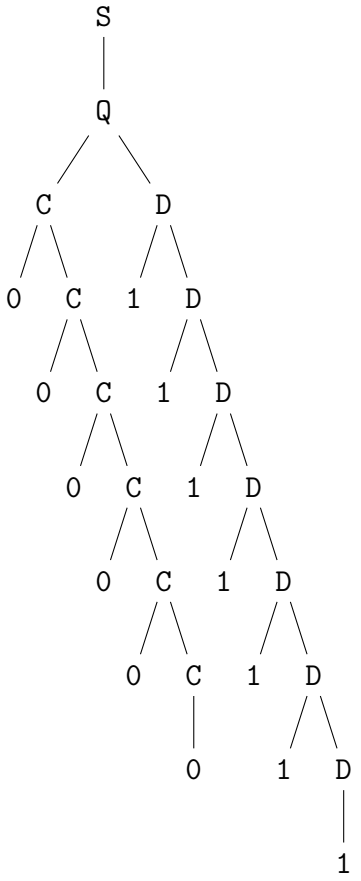
a) Da das Wort 4, also eine gerade Anzahl, 0 enthält, kann es über E aus S wie folgt abgeleitet werden:

$$\begin{aligned} &\mathtt{S} \Rightarrow \mathtt{E} \Rightarrow \mathtt{1E} \Rightarrow \mathtt{11E} \Rightarrow \mathtt{110T} \Rightarrow \mathtt{1101T} \Rightarrow \mathtt{11010E} \Rightarrow \mathtt{110101E} \Rightarrow \mathtt{1101011E} \\ &\quad \Rightarrow \mathtt{11010111E} \Rightarrow \mathtt{110101110T} \Rightarrow \mathtt{1101011101T} \Rightarrow \mathtt{11010111010E} \Rightarrow \mathtt{11010111010}. \end{aligned}$$

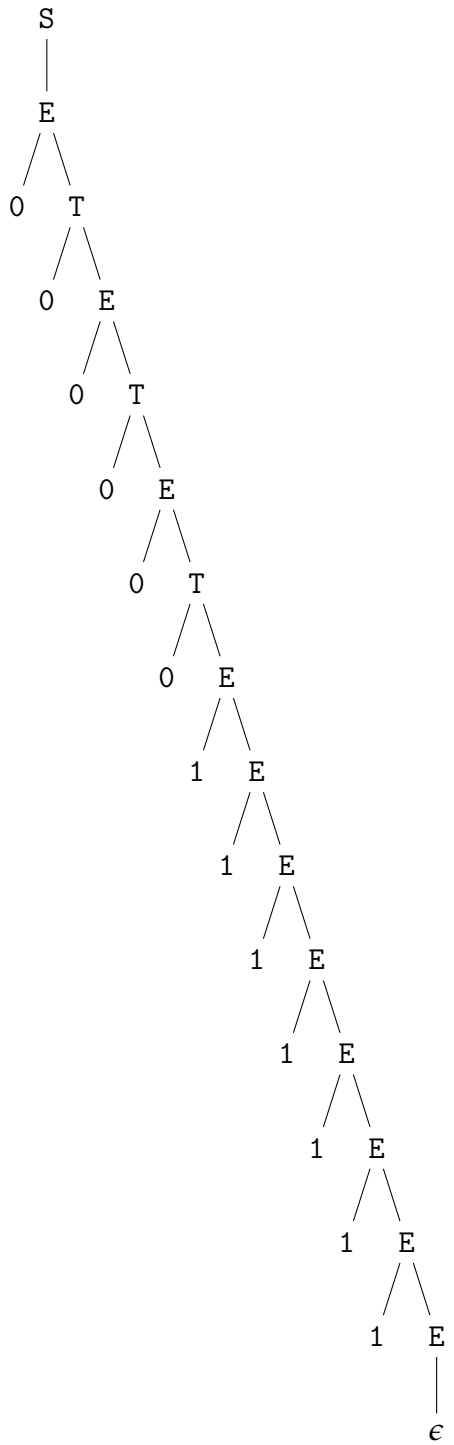
b) Da das Wort 5, also eine ungerade Anzahl, 0 enthält, kann es über Z aus S wie folgt abgeleitet werden:

$$\begin{aligned} &\mathtt{S} \Rightarrow \mathtt{Z} \Rightarrow \mathtt{1Z} \Rightarrow \mathtt{11Z} \Rightarrow \mathtt{110E} \Rightarrow \mathtt{1100T} \Rightarrow \mathtt{11000E} \Rightarrow \mathtt{110001E} \Rightarrow \mathtt{1100011E} \\ &\quad \Rightarrow \mathtt{11000110T} \Rightarrow \mathtt{110001101T} \Rightarrow \mathtt{1100011011T} \Rightarrow \mathtt{11000110110E} \Rightarrow \mathtt{11000110110} \end{aligned}$$

c)



d)



e)

$$
\begin{aligned}
L(G) = \quad & \{x | x \in \{0,1\}^* \text{ und } x \text{ hat eine gerade Anzahl an '0'}\} \\
& \cup \{x | x \in \{0,1\}^* \text{ und } x \text{ hat eine ungerade Anzahl an '0'}\} \\
& \cup \{0x1y | x \in \{0\}^* \text{ und } y \in \{1\}^*\} \\
& \cup \epsilon \\
= \quad & \{0,1\}^*
\end{aligned}
$$

Aufgabe 4.2 (4+8=12 Punkte) [nicht 4ECTS]

a) Schreiben Sie ein MIMA-Programm, das Ihnen den Wert $2^{23} - 24 = 8.388.584$ in Speicherzelle `y` ablegt. Schreiben Sie das Programm in die entsprechende Vorlage am Ende des Übungslatts.
Hinweis: Beachten Sie die Anzahl an Bits, die Sie für die Repräsentation im Zweierkomplement zur Darstellung dieser Zahl benötigen!

b) Implementieren Sie den MIMA-Befehl `EQL` *adr* mit Hilfe der anderen MIMA-Befehle nach. Schreiben Sie dazu ein MIMA-Programm in die entsprechende Vorlage am Ende des Übungsblatts. Verwenden Sie nicht mehr Instruktionen als die Vorlage vorsieht. Zusätzlich zu den Speicheradressen in der Vorlage haben Sie noch zwei zusätzliche Speicherzellen zur Verfügung, die Sie über die Adressen `x` und `y` adressieren können.

Lösung 4.2

a)

Speicheradresse	Befehl	Kommentar
0	`LDC` *1*	
1	`RAR`	jetzt steht in Akk 1000 0000 0000 0000 0000 0000 = 2^{23}
2	`STV` *y*	
3	`LDC` *23*	
4	`NOT`	Akku jetzt $= -24$
5	`ADD` *y*	Akku jetzt $= 2^{23} - 24$
6	`STV` *y*	

b) Idee des Algorithmus: XOR der zu vergleichenden Werte. Wenn Werte gleich, sind alle Bits 0. Wenn nicht gleich, dann mindestens ein Bit 1. JMN prüft das höchste Bit ab, wenn 1, dann setze Akku auf 0 und Befehl ist fertig; rotiere mit RAR alle 24 Bits durch. Wenn nie JMN, dann setze Akku auf -1.

Speicheradresse	Befehl	Kommentar
0	`XOR` *adr*	Falls Speicher(addr) und Akku gleich, dann steht in Akku 0000 0000 0000 0000 0000 0000
1	`STV` x	
2	`LDC` *1*	
3	`RAR`	jetzt steht in Akk 1000 0000 0000 0000 0000 0000 = 2^{23}
4	`STV` y	
5	`LDC` *23*	
6	`NOT`	Akku jetzt $= -24$
7	`ADD` y	Akku jetzt $= 2^{23} - 24$
8	`STV` y	In y jetzt Zähler, der, wenn 24mal erhöht, dazu führt, dass höchstwertige Bit 1 wird.
9	`LDV` x	
10	`JMN` *20*	höchste Bit 1, dann nicht gleich, setze Akku auf 0
11	`RAR`	rotiere zu nächstem Bit
12	`STV` x	
13	`LDC` *1*	
14	`ADD` y	Zähler erhöht
15	`STV` y	und abgespeichert
16	`JMN` *18*	alle 24 Bits durchrotiert, keine 1 dabei also Akku auf -1 setzen
17	`JMP` *9*	nächste Iteration
18	`LDC` *0*	alle Bits durchrotiert, ohne, dass auch nur eines 1 war
	`NOT`	also setze Akku auf -1
19	`JMP` *21*	fertig
20	`LDC` *0*	Haben ein Bit gefunden, das auf 1 steht, also nicht gleich, also setze Akku auf 0
21		Hier geht das Programm nach Ausführung von nachimplementiertem `EQU` *d*ann weiter

Aufgabe 4.3 (2+4+2=8 Punkte) [nicht 4ECTS]

So wie in der Vorlesung sei im Speicher eine Liste von Werten gegeben. Die Länge der Liste stehe an der Speicheradresse *len*. Der erste Wert der Liste steht an Speicheradresse *list*. Die weiteren Werte der Liste stehen in den direkt folgenden Speicherzellen.

Folgendes Programm haben Sie in der Vorlesung kennegelernt, um

$$M(sum) = \sum_{i=0}^{M(len)-1} M(list[i])$$

zu berechnen:

```
start:  LDC 0
        STV sum
        STV cnt
        LDC list
        STV ref
next:   LDIV ref
        ADD sum
        STV sum
        LDC 1
        ADD cnt
        STV cnt
        LDC 1
        ADD ref
        STV ref
        LDV cnt
        EQL len
        JMN done
        JMP next
done:   HALT
```

Es sei nun der folgende Speicherausschnitt mit einer Liste gegeben:

len:	3
list:	11
	22
	33
cnt:	
ref:	
sum:	

a) Schreiben Sie die Befehle der Reihe nach hin, wie sie bei der Ausführung des Programms auf der gegeben Liste ausgeführt werden. Wieviele Befehle werden insgesamt ausgeführt?
b) Verändern Sie das Programm so, dass nachwievor

$$M(sum) = \sum_{i=0}^{M(len)-1} M(list[i])$$

berechnet wird, aber weniger Befehle ausgeführt werden müssen. Schreiben Sie das Programm in die Vorlage am Ende des Übungsblattes.
c) Schreiben Sie für Ihr Programm aus Teilaufgabe b) hin, welche Befehle in welcher Reihenfolge jetzt ausgeführt werden, wenn das Programm auf die gegeben Liste angewendet wird. Wie viele Befehle werden jetzt ausgeführt?

Lösung 4.3

a) Es werden 44 Befehle ausgeführt:

Nr.	Befehl	sum	cnt	ref
1	`LDC` 0			
2	`STV` *sum*	0		
3	`STV` *cnt*	0	0	
4	`LDC` *list*	0	0	
5	`STV` *ref*	0	0	list
6	`LDIV` *ref*	0	0	list
7	`ADD` *sum*	0	0	list
8	`STV` *sum*	11	0	list
9	`LDC` 1	11	0	list
10	`ADD` *cnt*	11	0	list
11	`STV` *cnt*	11	1	list
12	`LDC` 1	11	1	list
13	`ADD` *ref*	11	1	list
14	`STV` *ref*	11	1	list+1
15	`LDV` *cnt*	11	1	list+1
16	`EQL` *len*	11	1	list+1
17	`JMN` *done*	11	1	list+1
18	`JMP` *next*	11	1	list+1
19	`LDIV` *ref*	11	1	list+1
20	`ADD` *sum*	11	1	list+1
21	`STV` *sum*	33	1	list+1
22	`LDC` 1	33	1	list+1
23	`ADD` *cnt*	33	1	list+1
24	`STV` *cnt*	33	2	list+1
25	`LDC` 1	33	2	list+1
26	`ADD` *ref*	33	2	list+1
27	`STV` *ref*	33	2	list+2
28	`LDV` *cnt*	33	2	list+2
29	`EQL` *len*	33	2	list+2
30	`JMN` *done*	33	2	list+2
31	`JMP` *next*	33	2	list+2
32	`LDIV` *ref*	33	2	list+2
33	`ADD` *sum*	33	2	list+2
34	`STV` *sum*	66	2	list+2
35	`LDC` 1	66	2	list+2
36	`ADD` *cnt*	66	2	list+2
37	`STV` *cnt*	66	3	list+2
38	`LDC` 1	66	3	list+2
39	`ADD` *ref*	66	3	list+2
40	`STV` *ref*	66	3	list+3
41	`LDV` *cnt*	66	3	list+2
42	`EQL` *len*	66	3	list+2
43	`JMN` *done*	66	3	list+2
44	`HALT`	66	3	list+2

b)

```
start:  LDC 0
        STV sum
        STV cnt
        LDC list
        STV ref
next:   LDIV ref
        ADD sum
        STV sum
        LDC 1
        ADD cnt
        STV cnt
        EQL len
        JMN done
        LDC 1
        ADD ref
        STV ref
        LDV cnt
        JMP next
done:   HALT
```

c) Es werden 40 Befehle ausgeführt:

Nr.	Befehl	sum	cnt	ref
1	`LDC` 0			
2	`STV` *sum*	0		
3	`STV` *cnt*	0	0	
4	`LDC` *list*	0	0	
5	`STV` *ref*	0	0	list
6	`LDIV` *ref*	0	0	list
7	`ADD` *sum*	0	0	list
8	`STV` *sum*	11	0	list
9	`LDC` 1	11	0	list
10	`ADD` *cnt*	11	0	list
11	`STV` *cnt*	11	1	list
12	`EQL` *len*	11	1	list
13	`JMN` *done*	11	1	list
14	`LDC` 1	11	1	list
15	`ADD` *ref*	11	1	list
16	`STV` *ref*	11	1	list+1
17	`LDV` *cnt*	11	1	list+1
18	`JMP` *next*	11	1	list+1
19	`LDIV` *ref*	11	1	list+1
20	`ADD` *sum*	11	1	list+1
21	`STV` *sum*	33	1	list+1
22	`LDC` 1	33	1	list+1
23	`ADD` *cnt*	33	1	list+1
24	`STV` *cnt*	33	2	list+1
25	`EQL` *len*	33	2	list+1
26	`JMN` *done*	33	2	list+1
27	`LDC` 1	33	2	list+1
28	`ADD` *ref*	33	2	list+1
29	`STV` *ref*	33	2	list+2
30	`LDV` *cnt*	33	2	list+2
31	`JMP` *next*	33	2	list+2
32	`LDIV` *ref*	33	2	list+2
33	`ADD` *sum*	33	2	list+2
34	`STV` *sum*	66	2	list+2
35	`LDC` 1	66	2	list+2
36	`ADD` *cnt*	66	2	list+2
37	`STV` *cnt*	66	3	list+2
38	`EQL` *len*	66	3	list+2
39	`JMN` *done*	66	3	list+2
40	`HALT`	66	3	list+2

Vorlage für Aufgabe 4.2a)

Speicheradresse	Befehl
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Vorlage für Aufgabe 4.2b)

Speicheradresse	Befehl
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	

19	
20	
21	
22	
23	
24	
25	

Vorlage für Aufgabe 4.3b)

start:	LDC 0 STV *sum* STV *cnt* LDC *list* STV *ref*
next:	
done:	HALT