Stephan Brumme, SST, 3.FS, Matrikelnr. 70 25 44
mail@stephan-brumme.com

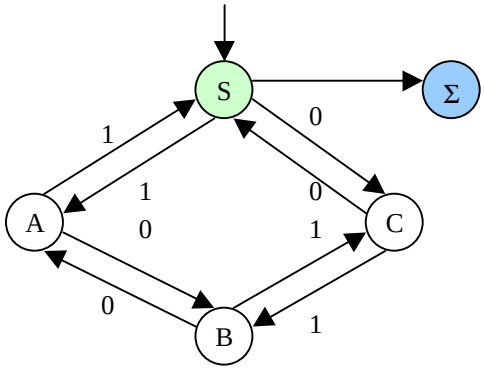
Aufgabe 1

Der endliche Automat besteht aus zwei Terminalen: 0 und 1. Auf diesen aufbauend existieren vier Nichtterminale, die im wesentlichen folgende Zustände repräsentieren:

Nichtterminal	Zustand
S	stabil, d.h. gerade Anzahl von Nullen und Einsen
A	ungerade Anzahl von Einsen, gerade Anzahl Nullen
B	ungerade Anzahl von Einsen und Nullen
C	gerade Anzahl von Einsen, ungerade Anzahl Einsen

Der grün schattierte Zustand S ist gleichzeitig der Startzustand, nur er kann in den terminierten Zustand Σ übergehen, der in der Graphik blau ist. Sollte der Eingabestrom abbrechen, während der Automat sich im Zustand A, B oder C befindet, so ist dies ein Fehler, da dann das Eingabewort nicht der Grammatik entspricht. Die Ableitungsregeln lassen sich 1:1 in einen Graphen umsetzen:

Ausgangszustand	0	1
S	C	A
A	B	S
B	A	C
C	S	B



Ich gehe stets davon aus, dass die gegebenen Zeichenketten von links nach rechts verarbeitet werden.
Die Eingabefolge 010011011100 erzeugt dann die folgende Ableitung aus der Grammatik:

010011011100
S → 0,C
→ 01,B
→ 010,A
→ 0100,B
→ 01001,C
→ 010011,B
→ 0100110,A
→ 01001101,S
→ 010011011,A
→ 0100110111,S
→ 01001101110,C
→ 010011011100,S
→ Σ

Die Zeichenkette ist genau dann gültig gemäß der Grammatik, wenn der Automat sich im Zustand S befindet. Somit erhält man für 0110110101:

0110110101		
(0110110101,S)	→	(110110101,C)
	→	(10110101,B)
	→	(0110101,C)
	→	(110101,S)
	→	(10101,A)
	→	(0101,S)
	→	(101,C)
	→	(01,B)
	→	(1,A)
	→	S

Auch diese Folge enthält eine gerade Anzahl von Nullen und Einsen.

Aufgabe 2

Das Startsymbol ist *EXPR*. Die jeweils nächste Ersetzung ist rot geschrieben, beim Startsymbol EXPR spare ich mir das:

v		
EXPR	→	TERM
	→	FACT
	→	v

(v)		
EXPR	→	TERM
	→	FACT
	→	(EXPR)
	→	(TERM)
	→	(FACT)
	→	(v)

v*v		
EXPR	→	TERM
	→	TERM*FACT
	→	FACT*FACT
	→	v*FACT
	→	v*v

v+v*v		
EXPR	→	TERM
	→	EXPR+TERM
	→	TERM+TERM
	→	FACT+TERM
	→	v+TERM
	→	v+TERM*FACT
	→	v+FACT*FACT
	→	v+v*FACT
	→	v+v*v

v*(v+v)		
EXPR	→	TERM
	→	TERM*FACT
	→	FACT*FACT
	→	v*FACT
	→	v*(EXPR)
	→	v*(EXPR+TERM)
	→	v*(TERM+TERM)
	→	v*(FACT+TERM)
	→	v*(v+TERM)
	→	v*(v+FACT)
	→	v*(v+v)

v+v*v/v		
EXPR	→	TERM
	→	EXPR+TERM
	→	TERM+TERM
	→	FACT+TERM
	→	v+TERM
	→	v+TERM*FACT
	→	v+FACT*FACT
	→	v+v*FACT
	→	v+v*v
	→	???

Der letzte Ausdruck deckt eine entscheidende Schwäche der Grammatik auf: sie ist nicht-deterministisch. Eine mögliche Lösung wäre, statt der grau schattierten Zeilen folgende Ersetzungen vorzunehmen:

v+v*v/v		
	→	v+TERM/FACT
	→	v+TERM*FACT/FACT
	→	v+FACT*FACT/FACT
	→	v+v*FACT/FACT
	→	v+v*v/FACT
	→	v+v*v/v

Aufgabe 3

a) Alle drei Sprachen enthalten nur Wörter, die aus einer Folge von a‘s, anschließend einer Folge von b’s und schließlich c’s bestehen: $L=\{a^+b^+c^+\}$.
Sie unterscheiden sich lediglich in der Anzahl der einzelnen Buchstaben, die ich im folgenden mit H(a), H(b) und H(c) beschreiben möchte.

G_a ist eine Grammatik, die beliebige H(a), H(b), H(c) größer 0 erlaubt, wobei H(a)=H(b)=H(c) ist: $L(G_a)=\{a^nb^nc^n|\forall n>0\}$

G_b erzeugt eine Sprache, die aus Wörter besteht, welche ebenso viele a’s wie b’s enthalten: H(a)=H(b). Der Wert H(c) ist beliebig, aber min. 1: $L(G_b)=\{a^nb^nc^m|\forall n,m>0\}$

G_c stellt eine Grammatik dar, die beliebige H(a), H(b) und H(c) größer 0 zulässt: $L(G_c)=\{a^nb^mc^k|\forall n,m,k>0\}$

b) G_a: Chomsky-Typ 1, kontextsensitive Spache
G_b: Chomsky-Typ 2, kontextfreie Spache
G_c: Chomsky-Typ 3, reguläre Spache

c) Die jeweils nächsten Ersetzungen sind zur besseren Übersicht rot markiert.
Für die drei Grammatiken entsteht bei der Eingabefolge *aaabbbccc*:

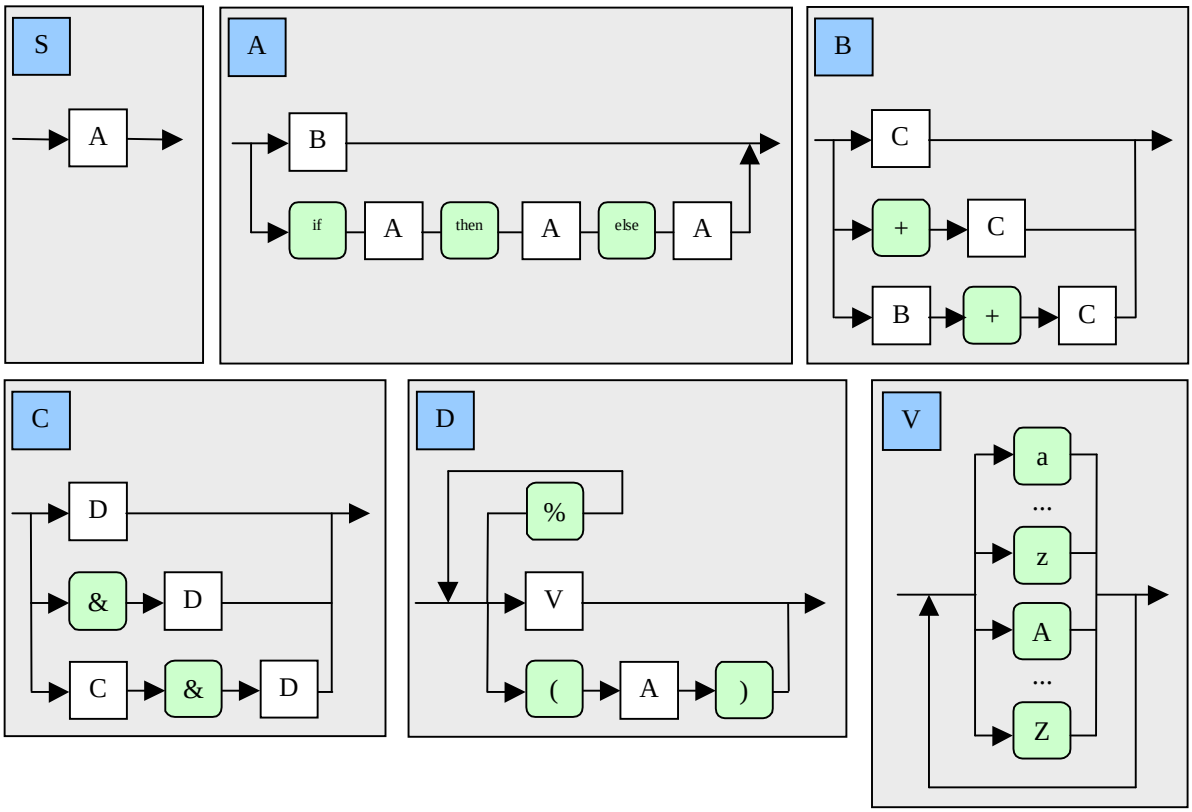
G_a	
S	→ Abc
	→ aABbc
	→ aAbBc
	→ aaABbBc
	→ aaABbbcc
	→ aaaBbbcc
	→ aaabBbcc
	→ aaabbBcc
	→ aaabbbccc

G_b	
S	→ AB
	→ aAbB
	→ aaAbbB
	→ aaabbbB
	→ aaabbbBc
	→ aaabbbBcc
	→ aaabbbccc

G_c	
C	→ Cc
	→ Ccc
	→ Bccc
	→ Bbccc
	→ Bbbccc
	→ Abbbccc
	→ Aabbbccc
	→ Aaabbbccc
	→ aaabbbccc

Aufgabe 4

Ich fange zuerst mit einer graphischen Darstellung der Backus-Naur-Form an, die Terminale sind in grünen Kästchen mit abgerundeten Ecken:



(xyz)
S → A
→ B
→ C
→ D
→ (A)
→ (B)
→ (C)
→ (D)
→ (V)
→ (xV)
→ (xyV)
→ (xyz)

abc+efg
S → A
→ B
→ B+C
→ C+C
→ D+C
→ V+C
→ aV+C
→ abV+C
→ abc+C
→ abc+D
→ abc+V
→ abc+eV
→ abc+efV
→ abc+efg

x+x+x+
S → ???

Diese Formel kann in der Grammatik nicht dargestellt werden, da alle Nichtterminale letztlich auf V reduziert werden, dieses aber verlangt, dass das letzte Terminal stets ein Buchstabe ist.

+x+x+x
S → A
→ B
→ B+C
→ B+C+C
→ +C+C+C
→ +D+D+D
→ +V+V+V
→ +x+x+x

Aus Platzgründen habe ich C, D und V gleichzeitig reduziert.