Vorlesung Theoretische Informatik
Sommersemester 2017
Dr. B. Baumgarten

Übungen zur Wiederholung – quer durch den Stoff
Mit Lösungsbeispielen

Vollständigkeit wird nicht garantiert, und
einige sind klausuruntypisch umfangreich.
Erläuterungen (zusätzlich zur Lösung) stehen in BLAU.

1. Aufgabe (Automaten und Grammatiken)

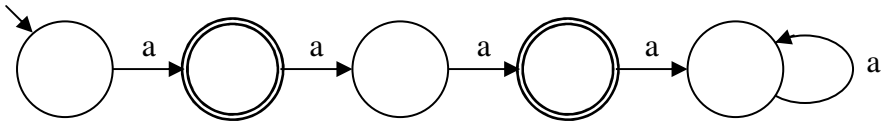
Definieren Sie die Sprache {*a*, *aaa*} mithilfe je eines der folgenden Beschreibungsmittel:

a) deterministischer endlicher Automat,
b) nichtdeterministischer endlicher Automat, der kein deterministischer endlicher Automat ist,
c) reguläre Grammatik,
d) kontextfreie aber nicht reguläre Grammatik,
e) kontextsensitive (nicht verkürzende) aber nicht kontextfreie Grammatik,
f) nicht kontextsensitive (also irgendwo verkürzende) Chomsky-Grammatik.

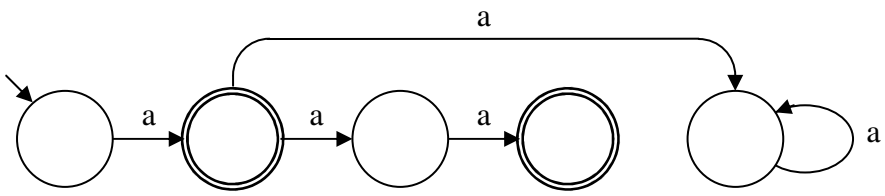
Bei Grammatiken genügen die Ableitungsregeln.

Lösungsbeispiele

a) (Zustandsnamen weggelassen, da irrelevant. Hier übrigens Minimalautomat)



b) (Zustandsnamen weggelassen, da irrelevant.
Geht z.B. auch ohne den Zustand ganz rechts.)



c) S → aA | a , A → aB , B → a

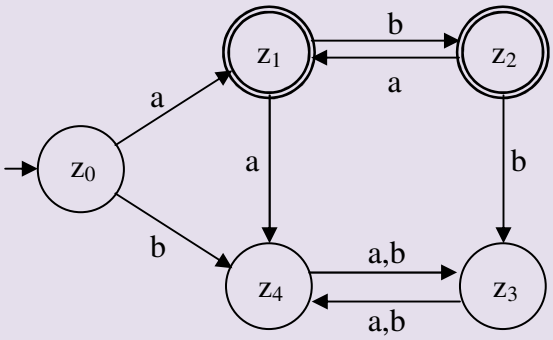
d) S → Aa | a , A → aa

e) S → a | aaA, aA → aa

f) S → aaa , aaa → a

2. Aufgabe (Minimalautomat)

Es sei der folgende endliche deterministische Automat *A* gegeben:



a) Zeigen oder widerlegen Sie, dass die Sprache $L(A)$ unendlich viele Wörter enthält.

b) Bestimmen Sie mit Hilfe des Minimierungsalgorithmus einen minimalen Automaten *B* mit $L(B) = L(A)$, und zeichnen Sie den Automaten *B*.

c) Geben Sie eine reguläre Grammatik *G* mit $L(G) = L(A)$ an.

Lösungsbeispiele

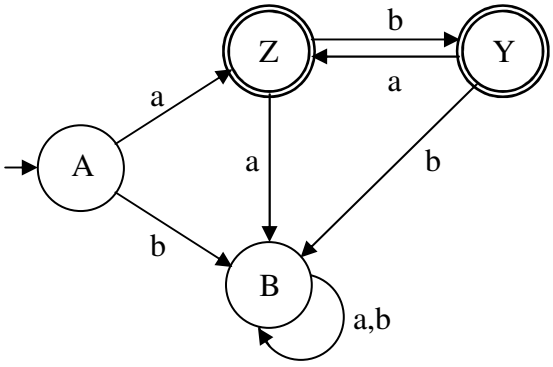
a) Der Automat akzeptiert u.a. (nämlich in z_1) alle $a(ba)^n$, n = 0,1,2,... , also unendlich viele Wörter.

b) Minimierungsalgorithmus

		a	b
z0	A	Z	A
z1	Z	A	Z
z2	Z	Z	A
z3	A	A	A
z4	A	A	A

		a	b
z0	A	Z	B
z1	Z	B	Y
z2	Y	Z	B
z3	B	B	B
z4	B	B	B

… also keine weitere Aufspaltung der Klassen, da in jeder Klasse alle Elemente die gleiche „Zukunft“ haben.



c) Startsymbol A

A → aZ | a | *bB*
Z → *aB* | bY | b
Y → aZ | a | *bB*
B → *aB* | *bB*

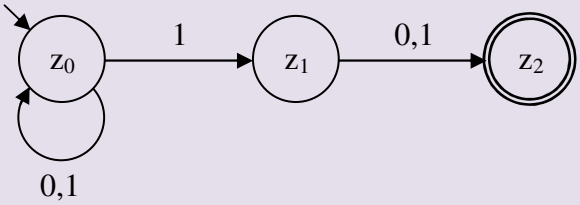
oder kürzer:
A → a | aZ
Z → b | bS

Die erste Grammatik ist nach Rezept gestrickt. Die 5 kursiven Regeln sind verzichtbar, da ein B am Ende immer bleibt, also daraus kein a-b-Wort mehr entstehen kann. Die zweite Grammatik entsteht durch Entfernung der verzichtbaren Regeln und die Beobachtung, dass A und Y „dasselbe tun“.

Zusatzübung: Leiten Sie aus der zweiten [einen ND-Automat, einen det. Automat und] einen Minimalautomat ab, den Sie mit dem in (b) vergleichen. Es sollte (bis auf evtl. die Zustandsnamen) der gleiche sein.

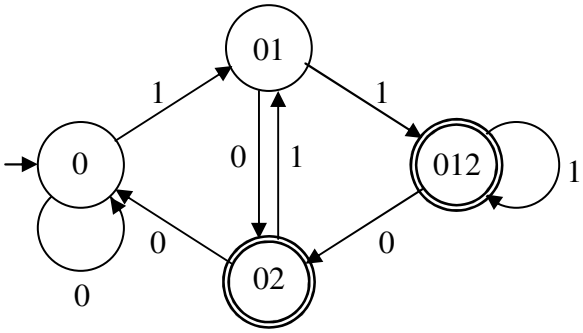
3. Aufgabe (Nichtdeterministischer Automat)

Konstruieren Sie einen deterministischen endlichen Automaten, der die gleiche Sprache akzeptiert wie der folgende NDA:



Ist der konstruierte Automat minimal?

Lösungsbeispiel



Wir wenden das Zustandsmengenrezept an. Zustand xy steht für $\{z_x,z_y\}$. Hier wurde die Konstruktion der erreichbaren Zustandsmengen gleich graphisch dargestellt. (Geht auch in einer Tabelle. Geht auch mit allen Zustandsmengen, einschließlich der unerreichbaren, wäre aber unnötige Schreib-/Malarbeit.) Begonnen wird wie im NDA, hier in z_0. Akzeptiert wird überall, wo der NDA akzeptieren könnte, also wo z_2 vorkommt.

Gezeichneter Automat minimal?

Minimierung beginnen:

		0	1
0	A	A	A
01	A	Z	Z
02	Z	A	A
012	Z	Z	Z

→

A
B
Z
Y

usw.

– ja, da die Minimierung einen Automaten mit 4 Zuständen ergibt, was wir ab „usw.“ wissen, denn A, B, Y, Z können nicht weiter zerlegt werden.

4. Aufgabe (Reguläre Sprachen)

Es seien $\Sigma = \{a,b\}$ und $L_1 = \Sigma^* \setminus \{ab\}$

a) Geben Sie einen deterministischen endlichen Automaten A mit $L(A) = L_1$ an.

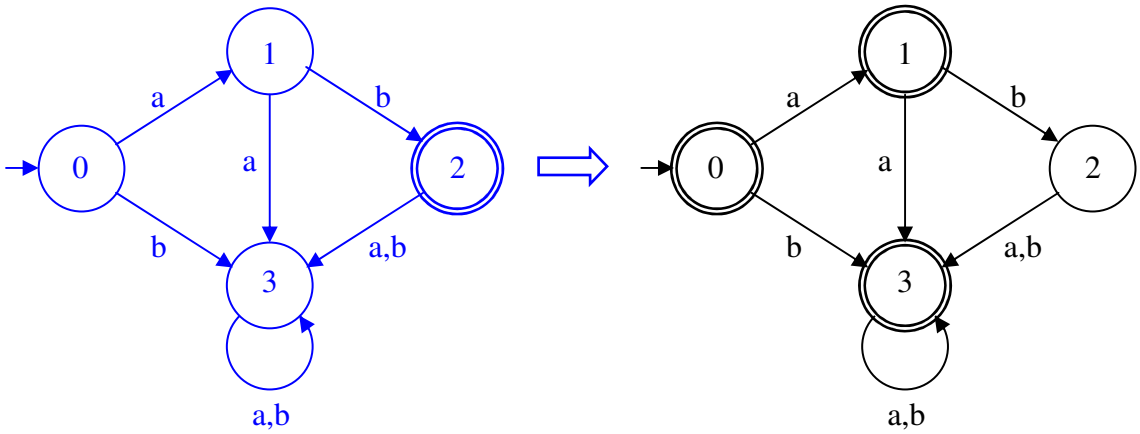
b) Geben Sie einen regulären Ausdruck R über Σ mit $L(R) = L_1$ an.

c) Geben Sie eine Sprache L_2 über Σ an, für die $L_1 \cdot L_2$ nicht regulär ist.
(ohne Begründung, aber erkennbar so, dass die Prüfung auf Zugehörigkeit zur Sprache unbeschränkten Speicher erfordert)

d) Geben Sie eine Sprache L_3 über Σ an, für die $L_1 \cap L_3$ nicht regulär ist.
(mit kurzer Begründung)

Lösungsbeispiele

a) Lösungsweg: Wir zeichnen den Automaten für {ab} und verwandeln ihn dann nach Rezept in den Automaten für das Komplement.



b) ε + a + b + (a+b) (a+b) (a+b) (a+b)* + aa + ba + bb
(d.h. alle Wörter der Längen 0, 1 oder >2, sowie die drei der Länge 2 ohne ab, oder auch anders, z.B.)
ε + a + b(a+b)* + a(a+b(a+b))(a+b)*
(d.h. nach einem Anfangs-a kann alles kommen außer nur ein b)

c) zum Beispiel $\{ a^n b^n \mid n>0 \}$
(Ein Algorithmus für das Wortproblem muss sich immer merken, wie viele a`s hintereinander zuletzt gelesen wurden; das erfordert unbegrenztes Gedächtnis.)

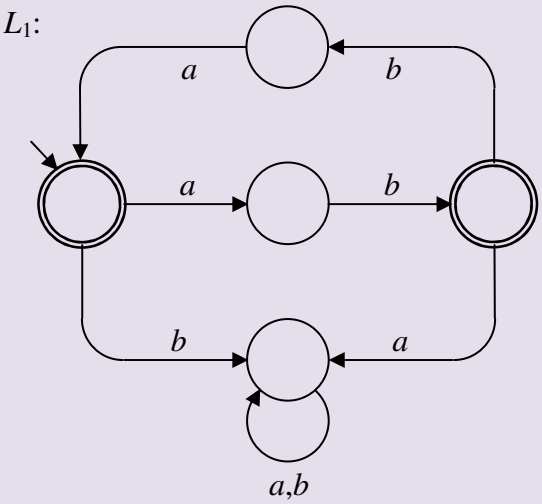
d) $L_3 := \{w \in \Sigma^* \mid \#_a(w) \neq \#_b(w)\}$. Wäre L_3 regulär, dann das Komplement $\{w \in \Sigma^* \mid \#_a(w) = \#_b(w)\}$ auch; das ist es aber nicht (erfordert unbegrenztes Gedächtnis). Also ist L_3 nicht regulär. Und $L_1 \cap L_3 = L_3 \setminus \{ab\} = L_3$.
alternativ ähnlich z.B. mit
$L_3 := \{b^n a^n \mid n \in \mathbb{N}\}$ (begründen … !)

5. Aufgabe (Reguläre Sprachen)

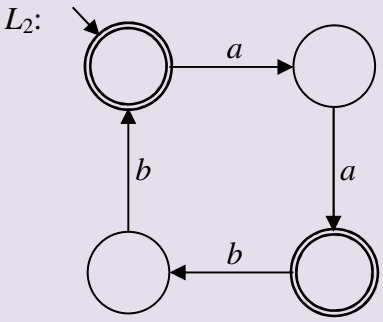
Im Folgenden sind 9 reguläre Sprachen definiert, und zwar durch

- Automaten (deterministische und nichtdeterministische), die sie akzeptieren:

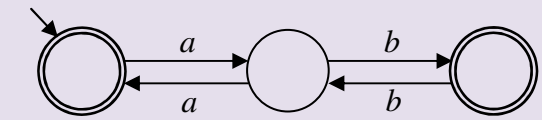
L_1:



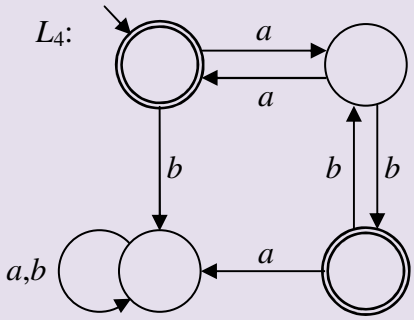
L_2:



L_3:



L_4:



- reguläre Ausdrücke:

L_5: $(aabb)^*(aa+\varepsilon)$
L_6: $(abba)^*(ab+\varepsilon)$

- Chomsky-Grammatiken über $\{a,b\}$ und mit Startsymbol S, die sie erzeugen:

L_7:
$S \to \varepsilon \mid T$
$T \to aA$
$A \to a \mid aT \mid b \mid bB$
$B \to bA$

L_8:
$S \to \varepsilon \mid T$
$T \to aA$
$A \to a \mid aB$
$B \to bC$
$C \to b \mid bT$

L_9:
$S \to \varepsilon \mid T$
$T \to aA$
$A \to b \mid bB$
$B \to bC$
$C \to a \mid aT$

Schreiben Sie jede Sprachbezeichnung (L_2, …, L_9) in genau eine der unteren Boxen, und zwar so, dass jeweils identische Sprachen in der gleichen Box und unterschiedliche Sprachen in verschiedenen Boxen stehen. Evtl. bleiben Boxen leer.

Tipp: Welche drei bis vier kürzesten Wörter werden jeweils akzeptiert bzw. produziert?

L_1	L_2	L_3	
L_6	L_5	L_4	
L_9	L_8	L_7	
ε, ab, abab, …	ε, aa, aabb, …	ε, aa, ab, …	

6. Aufgabe (Restsprachenautomat)

Sei L die Sprache aller Wörter über $\Sigma = \{ab\}$, die nirgendwo zwei aufeinanderfolgende a enthalten, d.h. $L = \{w \in \{a,b\}^* \mid$ es gibt keine $u,v \in \{a,b\}^*$ mit $w = uaav\}$.

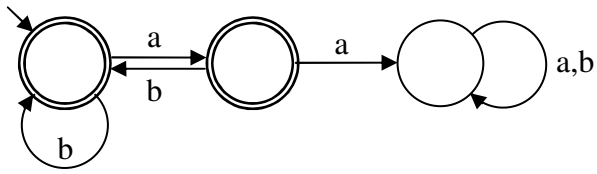
a) Zeichnen Sie einen endlichen Automaten für L mit möglichst wenigen Zuständen.

b) Geben Sie einen regulären Ausdruck für L an.

c) Tragen Sie reguläre Ausdrücke so in den Automaten aus (a) ein, dass Sie den Restsprachenautomat von L erhalten.

Lösungsbeispiel

a)



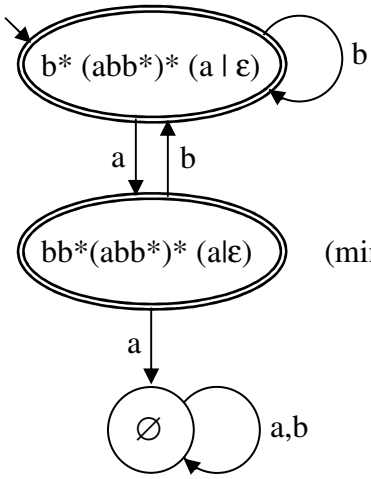
b) b* (abb*)* (a | ε)

evtl. *b*'s vor dem ersten *a*

null, ein- oder mehrmals: ein *a*, gefolgt von mind. einem *b*

nach den letzten *b*'s evtl. noch ein einzelnes *a*

c)



(mind. ein *b* vor dem ersten *a*, sonst wie oben)

Restsprachen-Schritte, z.B. vom obersten zum mittleren Zustand:

- Welche Wörter beginnen mit a?
  (abb*)* (a | ε) (außer ε)
- Welche bleiben nach Entfernung des Anfangs-a?
  Wenn ein abb* vorkam: Das bb* vom ersten abb*, dann immer noch beliebig viele abb* und dann (a | ε), also bb*(abb*)* (a|ε).
- Wenn kein abb* vorkam (also das einsame a hinten gestrichen wurde):
  ε, und das wird durch bb*(abb*)* (a|ε) (alle * null-mal wählen) mit abgedeckt.

7. Aufgabe (Pumping-Eigenschaft, kontextfreie Grammatik)

Es seien $\Sigma = \{a,b,c\}$. und $L = \{a^i b^j a^i c^k \mid i, j \geq 0, k \geq 1\}$.

a) Geben Sie ein 3- und ein 5-aufpumpbares Wort in L an, und begründen Sie Ihre Wahl.

b) Geben Sie ein Wort aus L an, das in L nicht 4-aufpumpbar ist, und begründen Sie Ihre Wahl.

c) Geben Sie eine kontextfreie Grammatik für L an.

Lösungsbeispiele

a) Für 3: cccc, für 5: cccccc, da das Pumpen (Streichen oder Vervielfachen) der „Schleife“ v im Wort, zerlegt in uvw mit $|uv| \leq 3$ bzw. 5 nicht-leere c-Folgen erzeugt, welche auch zur Sprache gehören (i=j=0).

b) aaaabaaaac, da 4-Pumpen (Vervielfachen oder Streichen eines nichtleeren Teilworts innerhalb der ersten 4 Zeichen) die erste Gruppe von a’s auch länger oder kürzer als die zweite Gruppe macht, also kein $a^i b^j a^i c^k$ ergibt.

c)

S → DC | C,	Zerlegung in $a^i b^j a^i$ und c^k
D → aDa | aa | B,	Zwiebelschalentechnik für die a’s rund um die evtl. b’s
B → b | Bb,	b’s, mindestens eins; Fall „kein b“ ist oben schon erledigt
C → c | cC	c’s, mindestens eins

8. Aufgabe (Chomsky-Normalform und CYK)

a) Transformieren Sie die Grammatik $G=[\Sigma,V,S,R]$ mit $\Sigma=\{a,b\}$, $V=\{A,B,C\}$ und den unten folgenden Regeln in eine Grammatik in Chomsky-Normalform mit gleicher Sprache:
S → aA | bB, A → Baa | ba | C, B → bCC | ab, C → A

b) Prüfen Sie nach, ob das Wort *bbbaba* zur Sprache von *G* gehört.

Lösungsbeispiel

a)

S	→	aA
S	→	bB
A	→	Baa
A	→	ba
A	→	C
B	→	bCC
B	→	ab
C	→	A

1. Zyklen und X>X raus:

S	→	aA
S	→	bB
A	→	Baa
A	→	ba
B	→	bAA
B	→	ab

2. X→Y überspringen: unnötig, erledigt

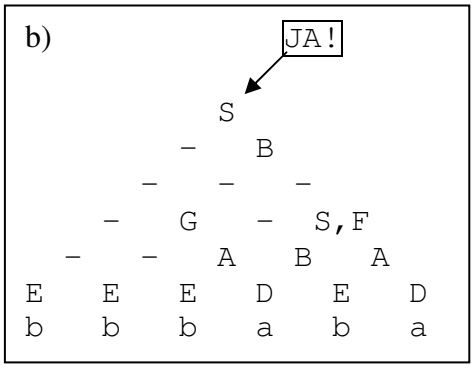
3. Variablen vor Konstanten schalten:

S	→	DA
S	→	EB
A	→	BDD
A	→	ED
B	→	EAA
B	→	DE
D	→	a
E	→	b

4. Lange rechte Seiten in Zweierschritte zerlegen:

S	→	DA | EB
A	→	FD | ED
B	→	GA | DE
D	→	a
E	→	b
F	→	BD
G	→	EA

b)



CYK-Beispiel für die Beschriftung der Felder, hier des Feldes mit S,F

Entweder „geometrisch“ mit der „Technik der zwei Pyramidenkletterer mit dem Seil über der Pyramide“ (vgl. Vorlesung bzw. Animation auf Webseite) oder ganz ausführlich so:

Das Feld ist die Spitze der Pyramide mit dem Sockel aba.
Gesucht werden daher Variablen (Nichtterminalzeichen),
aus denen aba bzw. DED (wegen D→a, E→b) ableitbar ist.
Das geht entweder über D-ED oder über DE-D,
und das geht über D-A (wegen A→ED) oder B-D (wegen B→DE),
und das leisten S (→DA) und F (→BD).

9. Aufgabe (Kellerautomaten)

Gegeben sei die Sprache $L = \{0^n 1^{2n} \mid n \in \mathbb{N}\}$

a) Leiten Sie aus einer kontextfreien Grammatik für L einen Kellerautomaten K_1 mit Kelleralphabet $\Gamma=\{S,0,1\}$ ab, der L beschreibt und nicht deterministisch ist.

b) Geben Sie die Übergangsfunktion eines deterministischen Kellerautomaten K_2 mit Kelleralphabet $\Gamma=\{a\}$ an, der die Sprache L beschreibt.

c) Demonstrieren Sie die Berechnung in K_2 für die Eingabe 10 und 01 bis zur Ablehnung und für die Eingabe 001111 bis zum Akzeptieren (in z_e).

Lösungsbeispiel

(a) Grammatikregeln: S → 011 | 0S11 (Zwiebelschalentechnik)

(z0, –, $, z1, S)	Grammatik im Keller, zunächst S hineinstecken,
(z1, –, S, z1, 011)	dann die Ableitungsschritte oben auf dem Stapel vollziehen
(z1, –, S, z1, 0S11)	dito
(z1, 0, 0, z1, ε)	Erzeugte Terminalzeichen mit Eingabewort verrechnen
(z1, 1, 1, z1, ε)	dito
(z1, –, $, ze, ε)	Ende wenn Keller wieder leer

(b) Hier z.B. folgende Methode: Pro 0 ein a in den Keller, pro a im Keller zweimal 1 verrechnen, genauer: Δ umfasst folgende Transitionen:

(z0, 0, $, z0, a)	pro 0 ein a in den Keller, erstmals
(z0, 0, a, z0, aa)	pro 0 ein a in den Keller, weitere
(z0, 1, a, z1, a)	verbraucht die erste 1, lässt a noch liegen
(z1, 1, a, z2, ε)	verbraucht die zweite (also eine „gerade“) 1 und das a
(z2, 1, a, z1, a)	verbraucht eine weitere „ungerade“ 1, lässt a noch liegen
(z2, –, $, ze, $)	Es kamen doppelt so viele Einsen wie a’s, also wie Nullen

Alternativ könnte man das a mit der ungeraden 1 löschen, dann aber noch eine gerade 1 ohne ein a verbrauchen.

Noch anders: Pro 0 zwei a’s in den Keller, dann eine 1 pro a verbrauchen:

(z0, 0, $, z0, aa)	pro 0 zwei a in den Keller, erstmals
(z0, 0, a, z0, aaa)	pro 0 zwei a in den Keller, weitere
(z0, 1, a, z1, ε)	verbraucht die erste (eine „ungerade“) 1 und das oberste a
(z1, 1, a, z2, ε)	verbraucht die nächste (also eine „gerade“) 1 und ein a
(z2, 1, a, z1, a)	verbraucht eine weitere „ungerade“ 1 und ein a
(z3, –, $, ze, $)	Es kamen so viele 1 wie a, also doppelt so viele wie Nullen

(c)

(z0, 10, $) ← stoppt mangels passendem Übergang

(z0, 01, $)
(z0, 1, a)
(z1, ε, a) ← stoppt mangels passendem Übergang

(z0, 001111, $)
(z0, 01111, a$)
(z0, 1111, aa$)
(z1, 111, aa$)
(z2, 11, a$)
(z1, 1, a$)
(z2, ε, $)
(ze, ε, $) ← akzeptiert und stoppt

Berechnung mit dem oberen Programm von (b)
(mit unterem ähnlich)

10. Aufgabe (Turing-Maschinen)

Eine Turing-Maschine mit Ein-/Ausgabe-Alphabet Σ = Bandalphabet Γ= {a,b,c}, Startzustand z_s und Endzustand z_e soll folgendes leisten: Bei jedem eingegebenen Wort *w* der Sprache (*a*+*b*)(*a*+*b*)* (also aus *a*'s und *b*'s und mit mindestens einem Zeichen) soll sie zwar jedes Zeichen mit *a* überschreiben, aber jedes dritte Zeichen stattdessen mit *c*. Das so entstandene Wort soll sie als Ergebnis liefern. Beispiel: abababb ↦ aacaaca.

a) Schreiben Sie eine passende Zustandsüberführungsfunktion δ mit möglichst wenigen Zuweisungen für diese Turing-Maschine.
b) Geben Sie die Folge der Konfigurationen Ihrer Turing-Maschine aus (a) an, wenn sie mit der Eingabe *abb* rechnet.

Lösungsbeispiel

(a)

$\delta(z_s,a) = \delta(z_s,b) = (z_1,a,R)$,	1. Zeichen ↦ a, (B inakzeptabel)
$\delta(z_1,a) = \delta(z_1,b)=(z_2,a,R)$	2. Zeichen ↦ a
$\delta(z_1,B)=(z_{zur},B,L)$	B ↦ zurück nach L
$\delta(z_2,a) = \delta(z_2,b)=(z_0,c,R)$	3. Zeichen ↦ c
$\delta(z_2,B)=(z_{zur},B,L)$	B ↦ zurück nach L
$\delta(z_0,a) = \delta(z_0,b)=(z_1,c,R)$	wie zs, erlaubt aber auch B
$\delta(z_0,B)=(z_{zur},B,L)$	B ↦ zurück nach L
$\delta(z_{zur},a)=(z_{zur},a,L)$	ganz nach links
$\delta(z_{zur},c)=(z_{zur},c,L)$	dito
$\delta(z_{zur},B)=(z_e,B,R)$	zurück auf Anfang vom Output, erledigt

b) Achtung: Konfiguration = (**Zustand**, was steht **links** von hier, was **beginnt** hier)!
(z_s,ε,aba),
Wegen $\delta(z_s,a) = (z_1,a,R)$ bleibt das a stehen, und der SL-Kopf geht nach rechts auf das b, und z_1 wird aktueller Zustand (usw.)
(z_1,a,ba), (z_2,aa,a), (z_0,aac,B), (z_{zur},aa,cB), (z_{zur},a,acB), $(z_{zur},\varepsilon,aacB)$, $(z_3,\varepsilon,BaacB)$, $(z_e,B,aacB)$

11. Aufgabe (Turing-Maschinen)

Es sei *M* eine Turing-Maschine mit der Zustandsmenge *Z*, dem Anfangszustand z_0, dem Endzustand z_e, dem Bandalphabet Γ und der Zustandsüberführungsfunktion δ. Ferner sei bekannt, dass *M* die folgende Funktion f über den natürlichen Zahlen (in Binärdarstellung) „in gutem Stil“ (nichts außer Output auf dem Band) berechnet:

$$f(x) = \begin{cases} 1, & \text{wenn } x \text{ durch 187 teilbar ist,} \\ 0 & \text{sonst.} \end{cases}$$

Geben Sie eine 1-Band Turing-Maschine M_0 an, die die folgende Funktion f_0 über den natürlichen Zahlen (in Binärdarstellung) berechnet:

$$f(x) = \begin{cases} 1, & \text{wenn } x \text{ nicht durch 187 teilbar ist,} \\ 0 & \text{sonst.} \end{cases}$$

Hinweis: Verwenden Sie die Komponenten von *M*.

Lösungsbeispiel

Neue Zustandsmenge $Z \cup \{z_f\}$, mit $z_f \notin Z$, Anfangszustand z_0, Endzustand z_f,

Bandalphabet Γ (muss ja mindestens 0, 1 enthalten)

Zustandsüberführungsfunktion δ':

wie δ, und zusätzlich

$\delta'(z_e,0) = (z_f,1,\mathrm{N})$

$\delta'(z_e,1) = (z_f,0,\mathrm{N})$

12. Aufgabe (Akzeptanz durch Kellerautomat)

Gegeben sei ein Kellerautomat K mit Startzustand z_0 und akzeptierendem Zustand z_a und den Transitionen:

$(z_0, -, \$, z_1, S)$,
$(z_1, -, S, z_1, 0S0)$, $(z_1, -, S, z_1, 1S1)$, $(z_1, -, S, z_1, \varepsilon)$
$(z_1, 0, 0, z_1, \varepsilon)$, $(z_1, 1, 1, z_1, \varepsilon)$, $(z_1, -, \$, z_a, \varepsilon)$

Welche Wörter über $\Sigma = \{0, 1\}$ werden mindestens (also ggf. auch andere) von K akzeptiert:

		JA	NEIN
(i)	alle Palindrome?	◯	◯
(ii)	alle Palindrome gerader Länge?	◯	◯
(iii)	alle Doppelwörter ww, $w \in \Sigma^*$?	◯	◯
(iv)	alle Binärzahlen mit Werten der Art $2^{2n}-1$, $n = 1, 2, 3, \ldots$	◯	◯
(v)	alle Binärzahlen mit Werten der Art $2^{2n}+1$, $n = 1, 2, 3, \ldots$	◯	◯
(vi)	alle Binärzahlen mit Werten der Art $2^{2n+1}+1$, $n = 1, 2, 3, \ldots$	◯	◯

Lösung

(i) NEIN: 1 z.B. wird nicht akzeptiert.

(ii) JA: Der Kellerautomat „spielt dafür im Keller eine Grammatik durch“, ähnlich wie beim konstruktiven Beweis, dass kontextfreie Sprachen durch Kellerautomaten akzeptiert werden. Sie ist mit den Regeln $S \rightarrow 0S0 \mid 1S1 \mid \varepsilon$ zwar nicht kontextfrei, erzeugt aber offensichtlich die $\{0, 1\}$-Palindrome gerader Länge.

(iii) NEIN: 1010 wird z.B. nicht akzeptiert.

(iv) JA: Alle diese sind Palindrome gerader Länge: $2^{2\cdot1}-1 \approx 11$, $2^{2\cdot2}-1 \approx 1111$ usw.

(v) NEIN: 101 mit Wert 2^2+1 wird z.B. nicht akzeptiert, hat ungerade Länge.

(vi) JA: Alle diese sind Palindrome gerader Länge: $2^{2\cdot1+1}+1 \approx 1001$, $2^{2\cdot2+1}+1 \approx 100001$ usw.

13. Aufgabe (Berechnung durch Turing-Maschine)

Gegeben sei eine Turingmaschine M mit Bandalphabet $\Sigma = \{0,1\}$, Startzustand z_a, Endzustand z_e und dem Programm:

$$\begin{aligned}
\delta(z_a, 0) &= (z_0, 0, R)\\
\delta(z_a, 1) &= (z_1, 1, R)\\
\\
\delta(z_0, 0) &= (z_0, 0, R)\\
\delta(z_0, 1) &= (z_0, 1, R)\\
\delta(z_0, B) &= (z_L, 0, L)\\
\\
\delta(z_1, 0) &= (z_1, 0, R)\\
\delta(z_1, 1) &= (z_1, 1, R)\\
\delta(z_1, B) &= (z_L, 1, L)\\
\\
\delta(z_L, 0) &= (z_L, 0, L)\\
\delta(z_L, 1) &= (z_L, 1, L)\\
\delta(z_L, B) &= (z_e, B, R)
\end{aligned}$$

Welchen Output produziert die Turingmaschine aus einem Input w, der ein nichtleeres Wort aus Nullen und Einsen ist?

Lösungsbeispiel

Das Ergebnis ist das Eingabewort, an das aber das erste Zeichen dieses Wortes angehängt wurde.

Durch z_0 oder z_1 merkt sich die TM, ob vorne 0 oder 1 steht. In beiden Zuständen geht die TM nach rechts und hängt hinten 0 bzw. 1 an. In z_L wird der Lese-Schreib-Kopf mit L-Bewegungen (und dem obligatorischen herunterfallen vom String mit Sprung zurück) auf das erste Zeichen des Ausgabewortes gesetzt.

14. Aufgabe (Turing-Reduzierbarkeit)

Es seien die Sprachen L_1, L_2 über dem Alphabet $\Sigma = \{0,1\}$ wie folgt gegeben:

$$L_1 = \{0^n \mid n \geq 1\},$$
$$L_2 = \{10^n \mid n \geq 1\}.$$

Zeigen Sie, dass das Wortproblem der Sprache L_1 auf das Wortproblem der Sprache L_2 Turing-reduzierbar ist.

Lösungsbeispiele

Der Algorithmus setzt als Ausgabe eine 1 vor die Eingabe.
Die Turing-Maschine mit Startzustand z_0, Endzustand z_e und mit

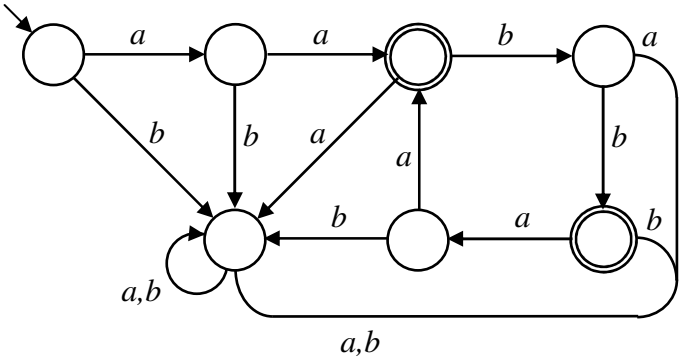
$\delta(z0,0) = (z0,0,L)$
$\delta(z0,1) = (z0,1,L)$
$\delta(z0,B) = (ze,1,N)$

berechnet die Funktion $f : \begin{cases} \Sigma^* \to \Sigma^* \\ w \mapsto 1\,w \end{cases}$, sodass genau dann $f(w) \in L_2$ gilt, wenn $w \in L_1$ gilt.

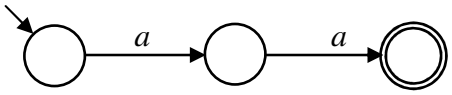
15. Aufgabe (Diverse Sprachbeschreibungen)

In der Folge werden zehn Sprachen, L_1 bis L_{10}, über $\Sigma = \{a,b\}$ beschrieben, die untereinander teilweise übereinstimmen. Dies geschieht nach mehreren Methoden:

- natürlich-sprachlich
  - L_1: Jedes Wort enthält *aa*.
  - L_2: Jedes Wort beginnt mit *aa*. Auf *aa* folgt stets zunächst *bb* oder nichts mehr. Auf *bb* folgt stets zunächst *aa* oder nichts mehr.
- als regulärer Ausdruck (+ wie | bedeuten „oder“)
  - L_3: $(a+b)^*aa(a+b)^*(aa)^*$
  - L_4: $aa\ (bbaa)^*(\varepsilon+bb)$
- als Automatensprache L_5 des folgenden endlichen Automaten:



- als Sprache L_6, die der folgende endliche nichtdeterministische Automat akzeptiert:



- als Sprache L_7 einer regulären Grammatik mit Startsymbol S:

  S → aR | bS R → a | aE | bS E → a | b | aE | bE

- als Sprache L_8 aller Wörter , für die eine Turing-Maschine das Ergebnis 1 berechnet, und zwar mit Anfangszustand z_0 und Endzustand z_e und folgender Zustandsüberführungsfunktion δ:

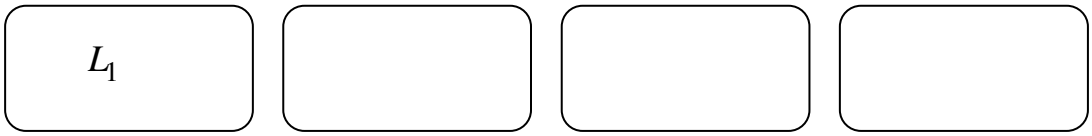
$$\delta(z_0,a)=(z_1,a,R),\quad \delta(z_0,b)=(z_0,b,R),$$
$$\delta(z_1,a)=(z_2,a,R),\quad \delta(z_1,b)=(z_0,b,R),$$
$$\delta(z_2,a)=(z_2,a,R),\quad \delta(z_2,b)=(z_2,b,R),\quad \delta(z_2,B)=(z_3,B,L),$$
$$\delta(z_3,a)=\delta(z_3,b)=(z_3,B,L),\quad \delta(z_3,B)=(z_e,1,N)$$

- als Sprache L_9 bzw. L_{10}, die ein Kellerautomat bei Endzustand z_e akzeptiert, mit Anfangszustand z_0 und folgender Transitionenmenge (links für L_9, rechts für L_{10}):

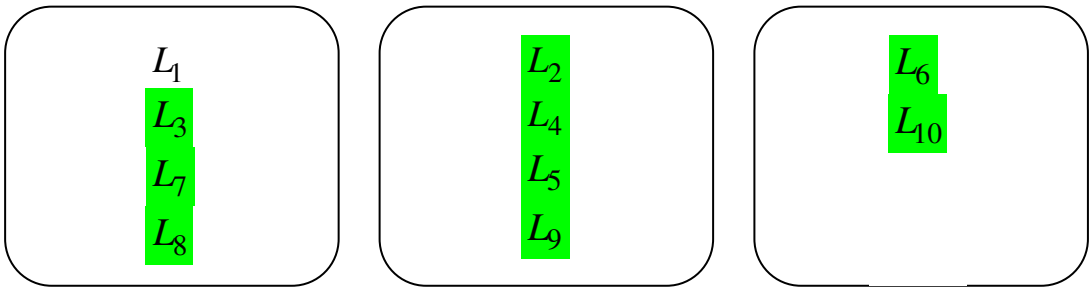
Δ_9 für L_9:
$(z_0,a,\$,z_0,a)$, $(z_0,a,a,z_1,\varepsilon)$,
$(z_1,b,\$,z_1,b)$, $(z_1,b,b,z_2,\varepsilon)$,
$(z_2,a,\$,z_2,a)$, $(z_2,a,a,z_1,\varepsilon)$,
$(z_1,-,\$,z_e,\varepsilon)$, $(z_2,-,\$,z_e,\varepsilon)$

Δ_{10} für L_{10}:
$(z_0,a,\$,z_0,1)$, $(z_0,a,1,z_0,2)$,
$(z_0,-,2,z_e,\varepsilon)$

Tragen Sie die Sprachen („L_n“) so in die folgenden Kästen ein, dass gleiche Sprachen im gleichen Kasten und unterschiedliche Sprachen in verschiedenen Kästen stehen. Es kann sein, dass nicht alle Kästen benutzt werden müssen.



Lösungsbeispiel

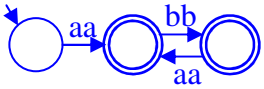


… aa …

oder (a+b)* aa (a+b)*

aa(!)-bb-aa-bb-…

oder



{ aa }

… wären anschauliche kurze (meist) formlose Kennzeichnungen, zum rascheren Einordnen.

Anmerkungen:

L3: Man sieht das garantierte aa mitten im Wort.
Das (aa)* am Ende ist wirkungslos und nur ein fieses Ablenkungsmanöver ☹, denn (a+b)*(aa)* = (a+b)* !

L4: *aa* (*bbaa*)* liefert nur L2-Wörter, aber nicht die mit bb endenden. Das repariert (ε+*bb*).

L5: Der Zustand links unten ist eine Falle/Mülltonne: Einmal drin wird nie mehr akzeptiert, und das geschieht genau dann, wenn von aabbaabb…. abgewichen wird.

L6: = {aa}, von L1 und L2 verschieden.

L7: R folgt auf a, E folgt auf aa. Alles (per E wie egal) darf kommen, sobald zwei a‘s nacheinander vorkamen, und erst nach zwei aufeinander folgenden a’s kann ein Wort enden, also L1.
Oder man zeichnet die Grammatik nach Rezept als Automat und enthält dann leicht erkennbar einen für L1.

L8: In z0 bzw. z1 bzw. z2 wurden gerade 0 bzw. 1 bzw. 2 a’s gelesen. In z0 bzw. z1 folgt auf b ein „neuer Versuch“ in z0.
In z2 läuft der Kopf ans Wortende und löscht in z3 rückwärts alles; dann wird 1 geschrieben, d.h. das Wort mit aa drin akzeptiert – L1!

L9: Das anfängliche a wird in z0 eingekellert, das nächste a leert den Keller und → z1.
Das erste b wird in z1 eingekellert, das nächste b leert den Keller und → z2.
z2 arbeitet praktisch wie z0.
Aufhören darf man bei leerem Keller (in z1, z2) aber nicht gleich am Anfang (in z0), also nach aa oder bb.

L10: Das erste a führt zu 1 im Keller, das zweite zu 2, und dann ist Schluss: {aa}