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) JA!

```
            S
          –   B
        –   –   –
      –   G   –   S,F
    –   –   A   B   A
  E   E   E   D   E   D
  b   b   b   a   b   a
```

gelb unterlegt:
Parsingbaum, siehe zweite b-Lösung unten

(b) entweder „geometrisch“, CYK mit der „Technik der zwei Pyramidenkletterer mit dem Seil über der Pyramide“ (oben)

oder durch sonst wie gefundene explizite Ableitung aus der Grammatik (unten).

b – Ableitung aus Grammatik)
S –(2)→ EB –(8)→ bB –(8)→ bGA –(10)→ bEAA –(10)→ bbAA –(10)→ bbEDA –(10)→
bbEDA –(8)→ bbbDA –(7)→ bbbaA –(4)→ bbbaED –(8)→ bbbabD –(7)→ bbbaba

Die Regeln sind 1-10 nummeriert. Die angewendete Regel ist jeweils angegeben. Gezeigt wird eine Linksableitung, deren Parsingbaum in (b-CYK) steckt.

9. Aufgabe (Kellerautomaten)

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

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

Anfangszustand z0, akzeptierender Zustand ze

(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 (Kellerautomaten)

Gegeben sei die Sprache $L = \{a^m b^n \mid m > n \geq 1\}$.

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

Tipps: Der Kellerautomat kann im akzeptierenden Zustand bis zu einer gewissen Zahl weitere *b* akzeptieren, will dies aber beim untersten „a“ im Keller nicht tun, damit er nur *weniger* b’s als a’s akzeptiert. Wie können wir dieses unterste „a“ „markieren“, damit der Kellerautomat es erkennt?

b) Und wie ginge das Ganze mit Kelleralphabet $\Gamma = \{a\}$?

Lösungsbeispiele

a) besondere Zustände: Start z0, akzeptierend: z_{akz}, Fehler: z_{err}

(z0, a, \$, z0, A)	erstes a als A eingekellert
(z0, a, A, z0, aA)	zweites a …
(z0, a, a, z0, aa)	und weitere a einkellern
(z0, b, a, z_{akz}, ε)	noch OK: nach a’s mindestens ein b
(z0, b, A, z_{err}, ε)	UPPS: Das war ab, das ist nicht in L, Fehler
(z_{akz}, b, a, z_{akz}, ε)	OK solange weniger b’s als a‘s
(z_{akz}, b, A, z_{err}, ε)	UPPS: Das war ein $a^n b^n$, das ist nicht in L, Fehler

b) besondere Zustände: Start z0, akzeptierend: z_{akz}, Fehler: z_{err}

(z0, a, \$, z1, ε)	erstes a wird nur (per Zustand) gemerkt, nicht eingekellert
(z1, a, \$, z1, a)	zweites a …
(z1, a, a, z1, aa)	und weitere a einkellern
(z1, b, a, z_{akz}, ε)	noch OK: nach a’s mindestens ein b
(z1, b, \$, z_{err}, ε)	UPPS: Das war ab, das ist nicht in L, Fehler
(z_{akz}, b, a, z_{akz}, ε)	OK solange weniger b’s als a‘s
(z_{akz}, b, \$, z_{err}, ε)	UPPS: Das war ein $a^n b^n$, das ist nicht in L, Fehler

11. Aufgabe (Turing-Maschinen)

Eine Turing-Maschine mit Ein-/Ausgabe-Alphabet Σ = Bandalphabet $\Gamma = \{a,b,c\}$, 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 $\mapsto$ 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 $\mapsto$ a, (B inakzeptabel)
$\delta(z_1,a) = \delta(z_1,b)=(z_2,a,R)$	2. Zeichen $\mapsto$ a
$\delta(z_1,B)=(z_{zur},B,L)$	B $\mapsto$ zurück nach L
$\delta(z_2,a) = \delta(z_2,b)=(z_0,c,R)$	3. Zeichen $\mapsto$ c
$\delta(z_2,B)=(z_{zur},B,L)$	B $\mapsto$ 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 $\mapsto$ 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), $(z_{zur},\varepsilon,aacB)$, $(z_3,\varepsilon,BaacB)$, $(z_e,B,aacB)$

12. 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

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

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