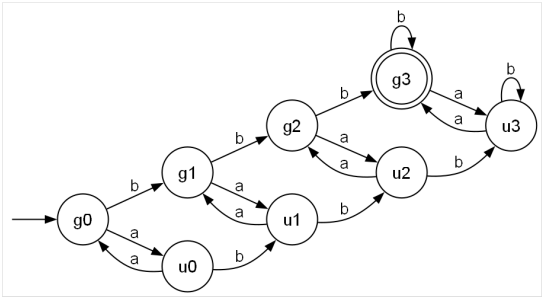
1. (3 Punkte) Gegeben ist das Alphabet $\Sigma = \{a, b\}$.
   (a) Wie viele Wörter der Länge 3 gibt es über Σ? Gib alle Wörter mit höchstens 2 Zeichen an.
   (b) L sei die Sprache der Wörter mit genau einem b. Gib alle Wörter von L mit höchstens 3 Zeichen an. Liegt ε in L?
   (c) Ist L endlich? Ist Σ^* endlich?

Lösung:
(a) $2^3 = 8$. Höchstens 2 Zeichen: $\varepsilon, a, b, aa, ab, ba, bb$.
(b) b, ab, ba, aab, aba, baa. ε enthält kein b, liegt also nicht in L.
(c) Beide sind unendlich: L enthält z.B. $a^n b$ für jedes n.

2. (3 Punkte) Konstruiere einen DEA über $\Sigma = \{a, b\}$, der die Sprache der Wörter mit einer geraden Anzahl von a und mindestens 3 b akzeptiert. Gib auch die Zustandsübergangstabelle an.

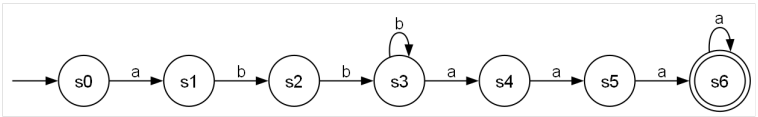
Lösung: Der Zustand merkt sich zwei Dinge: ob die Anzahl der a gerade (g) oder ungerade (u) ist und wie viele b schon gelesen wurden (0, 1, 2, 3 oder mehr).
Ein möglicher Automat:



	a	b
→ g0	u0	g1
g1	u1	g2
g2	u2	g3
∗ g3	u3	g3
u0	g0	u1
u1	g1	u2
u2	g2	u3
u3	g3	u3

3. (3 Punkte) Konstruiere einen DEA über $\Sigma = \{a, b\}$, der die Sprache $L = \{ab^n a^m : n \geq 2, m \geq 3\}$ akzeptiert.

Lösung: Ein möglicher Automat:

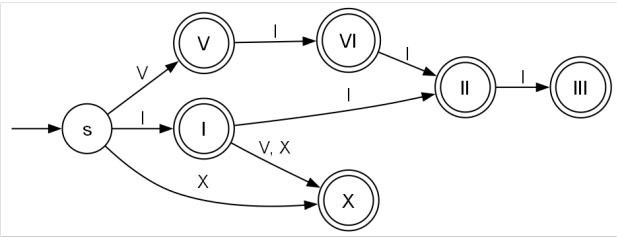


	a	b
→ s0	s1	–
s1	–	s2
s2	–	s3
s3	s4	s3
s4	s5	–
s5	s6	–
∗ s6	s6	–

Alle fehlenden Übergänge (–) führen in einen Fehlerzustand F, der nicht eingezeichnet ist.

4. (4 Punkte) Konstruiere einen DEA über $\Sigma = \{I, V, X\}$ mit höchstens sieben Zuständen (ohne Fehlerzustand), der genau die römischen Zahlen von 1 bis 10 akzeptiert, also $\{I, II, III, IV, V, VI, VII, VIII, IX, X\}$.

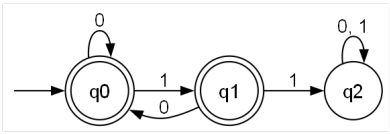
Lösung: Ein möglicher Automat:



	I	V	X
→ s	I	V	X
∗ I	II	X	X
∗ II	III	–	–
∗ III	–	–	–
∗ V	VI	–	–
∗ VI	II	–	–
∗ X	–	–	–

Der Zustand II steht für II und VII, der Zustand X für IV, IX und X: Danach darf jeweils dasselbe folgen.

5. (3 Punkte) Der DEA akzeptiert alle Wörter über $\{0, 1\}$ ohne zwei aufeinanderfolgende Einsen.



(a) Welche Sprache akzeptiert der DEA, wenn man Endzustände und Nicht-Endzustände vertauscht?

(b) Der NEA mit den Zuständen A, B, C, D aus dem Notebook akzeptiert alle Wörter, die 010 enthalten. Vertauscht man auch hier die Endzustände, akzeptiert er dann genau die Wörter ohne 010? Prüfe das Wort 010.

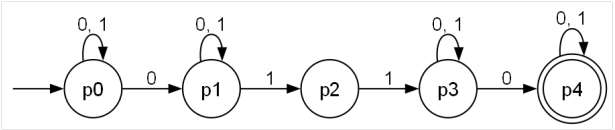
Lösung:

(a) Alle Wörter, die 11 enthalten. Der DEA landet bei jedem Wort in genau einem Zustand, das Vertauschen kehrt die Entscheidung für jedes Wort um.

(b) Nein. Beim Wort 010 kann der NEA in A bleiben (Weg A, A, A, A) oder nach D gehen. A ist nach dem Tausch ein Endzustand, also wird 010 akzeptiert, obwohl es 010 enthält. Bei einem NEA genügt ein Weg zum Endzustand, deshalb funktioniert der Tausch nur beim DEA (mit Übergängen für alle Zeichen).

6. (3 Punkte) Die Sprache L besteht aus allen Wörtern über $\Sigma = \{0, 1\}$, die mindestens zwei Nullen enthalten, zwischen denen der Teilstring 11 vorkommt. Konstruiere einen endlichen Automaten (es darf ein NEA sein), der genau L akzeptiert.

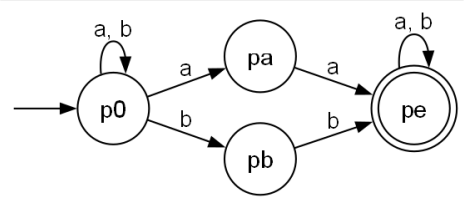
Lösung: Ein möglicher Automat:



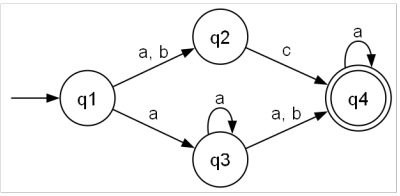
Der NEA rät die erste Null, dann die Stelle der 11, dann die zweite Null.

7. (3 Punkte) Die Sprache L besteht aus allen Wörtern über $\Sigma = \{a, b\}$, die aa oder bb enthalten. Konstruiere einen endlichen Automaten, der genau L erkennt.

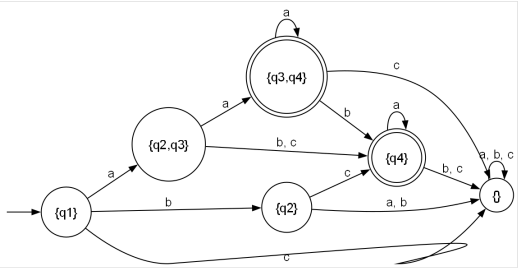
Lösung: Ein möglicher Automat:



8. (4 Punkte) Konstruiere zu folgendem NEA über dem Alphabet $\Sigma = \{a, b, c\}$ mit der Potenzmengenkonstruktion einen DEA, der dieselbe Sprache akzeptiert. Gib die Zustandsübergangstabelle an und zeichne das Diagramm.

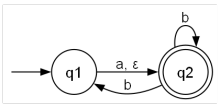


Lösung: Ein möglicher Automat:

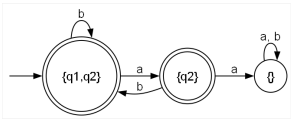


	a	b	c
→ {q1}	{q2,q3}	{q2}	{}
{q2,q3}	{q3,q4}	{q4}	{q4}
{q2}	{}	{}	{q4}
{}	{}	{}	{}
∗ {q3,q4}	{q3,q4}	{q4}	{}
∗ {q4}	{q4}	{}	{}

9. (3 Punkte) Konstruiere zu folgendem NEA über dem Alphabet $\Sigma = \{a, b\}$ mit der Potenzmengenkonstruktion einen DEA, der dieselbe Sprache akzeptiert. Gib die Zustandsübergangstabelle an und zeichne das Diagramm.



Lösung: Ein möglicher Automat:



	a	b
→ ∗ {q1,q2}	{q2}	{q1,q2}
∗ {q2}	{}	{q1,q2}
{}	{}	{}

10. (4 Punkte) Die Sprache L besteht aus allen Wörtern über $\Sigma = \{0, 1\}$ mit der folgenden Eigenschaft: Im Wort kommt sowohl 0110 als auch 101 als Teilwort vor. Gib einen regulären Ausdruck für L an.

Lösung: $L = (0 \cup 1)^*(0110(0 \cup 1)^*101 \cup 101(0 \cup 1)^*0110 \cup 01101 \cup 10110)(0 \cup 1)^*$

Entweder kommt 0110 zuerst und später 101 oder umgekehrt. Die beiden Teilwörter können sich auch überlappen: 01101 und 10110 enthalten beide.

11. (2 Punkte) Die Sprache L besteht aus allen Wörtern über $\Sigma = \{a, b\}$ mit der folgenden Eigenschaft: Das Wort enthält aa oder bb. Gib einen regulären Ausdruck für L an.

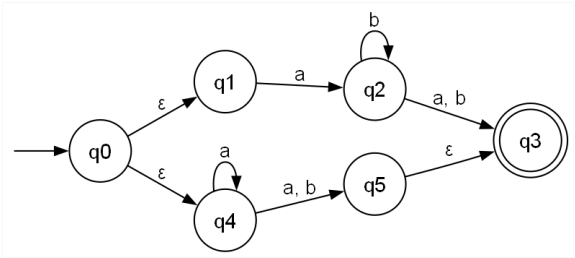
Lösung: $L = (a \cup b)^*(aa \cup bb)(a \cup b)^*$

12. (3 Punkte) Die Sprache L besteht aus allen Wörtern über $\Sigma = \{a, b\}$ mit der folgenden Eigenschaft: Das Wort enthält höchstens einmal aa und nie bb. Gib einen regulären Ausdruck für L an.

Lösung: $L = (\varepsilon \cup b)(ab)^*(\varepsilon \cup a)(ab)^*(\varepsilon \cup a)$

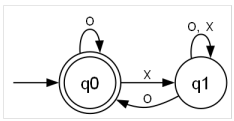
Ohne bb wechseln sich a und b ab, nur einmal dürfen zwei a aufeinanderfolgen. (aaa enthält aa zweimal.)

13. (3 Punkte) Gegeben ist der folgende endliche Automat A über $\Sigma = \{a, b\}$. Gib einen regulären Ausdruck an, der die Sprache $L(A)$ beschreibt und dabei höchstens zweimal das Vereinigungssymbol $\cup$ enthält.



Lösung: $L(A) = (ab^* \cup a^*)(a \cup b)$

14. (3 Punkte) Gegeben ist der folgende endliche Automat A über $\Sigma = \{o, x\}$. Gib einen regulären Ausdruck an, der die Sprache $L(A)$ beschreibt.



Lösung: $L(A) = \varepsilon \cup (o \cup x)^* o$

$L(A)$ enthält das leere Wort und alle Wörter, die auf o enden. Auch richtig ist z.B. $(o^* x (x \cup o)^* o)^* \cup o^*$.

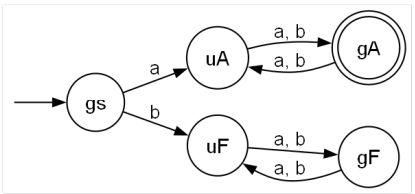
15. (4 Punkte) L_1 ist die Sprache der Wörter über $\{a, b\}$ mit gerader Länge, L_2 die Sprache der Wörter, die mit a beginnen. Die DEAs dafür haben die Tabellen

	a	b
$\rightarrow * \mathtt{g}$	u	u
u	g	g

	a	b
$\rightarrow \mathtt{s}$	A	F
$* \mathtt{A}$	A	A
F	F	F

Konstruiere den Produktautomaten für $L_1 \cap L_2$: Seine Zustände sind Paare aus je einem Zustand der beiden DEAs. Gib nur die erreichbaren Zustände an. Welche Zustände sind Endzustände?

Lösung: Ein möglicher Automat:



	a	b
$\rightarrow$ gs	uA	uF
uA	gA	gA
uF	gF	gF
$*$ gA	uA	uA
gF	uF	uF

Endzustand ist nur gA: gerade Länge (g) und mit a begonnen (A).

16. (3 Punkte) Der DEA aus der Aufgabe zum Komplement (keine zwei Einsen hintereinander) hat 3 Zustände und liest das Wort $w = 0101$.
    (a) Gib die Folge der Zustände an. Warum muss ein Zustand doppelt vorkommen?
    (b) Zerlege $w = xyz$, so dass y das Teilwort zwischen dem ersten und dem zweiten Besuch dieses Zustands ist.
    (c) Prüfe, dass xz, xy^2z und xy^3z akzeptiert werden.

Lösung:
(a) q0, q0, q1, q0, q1. Bei 4 Zeichen werden 5 Zustände durchlaufen, es gibt aber nur 3 verschiedene.
(b) Der Zustand q0 kommt zuerst nach 0 und wieder nach 1 Zeichen vor: $x = \varepsilon$, $y = 0$, $z = 101$.
(c) 101, 00101, 000101 werden akzeptiert: Die Schleife über y führt zurück in denselben Zustand.

17. (3 Punkte) Zeige mit dem Pumping-Lemma, dass die Sprache $L = \{ww \mid w \in \{0,1\}^*\}$ nicht regulär ist.

Lösung: Angenommen, L ist regulär, und p ist die Zahl aus dem Pumping-Lemma. Wähle $w = 0^p10^p1 \in L$ (mit $w = 0^p1$ zweimal). Für jede Zerlegung $w = xyz$ mit $|xy| \leq p$ und $|y| \geq 1$ besteht y nur aus Nullen der ersten Hälfte, $y = 0^k$ mit $k \geq 1$. Dann ist $xz = 0^{p-k}10^p1$. Läge es in L, müsste jede Hälfte genau eine 1 enthalten und beide Hälften gleich sein, also $0^{p-k}1 = 0^p1$. Das geht nicht, weil $k \geq 1$. Widerspruch, also ist L nicht regulär.

18. (3 Punkte) Zeige mit dem Pumping-Lemma, dass die Sprache $L = \{0^k10^j \mid k > j\}$ nicht regulär ist.

Lösung: Angenommen, L ist regulär, und p ist die Zahl aus dem Pumping-Lemma. Wähle $w = 0^{p+1}10^p \in L$. Für jede Zerlegung mit $|xy| \leq p$ besteht y nur aus Nullen vor der 1, $y = 0^k$ mit $k \geq 1$. Pumpt man ab ($i = 0$), ist $xz = 0^{p+1-k}10^p$. Vorne stehen jetzt höchstens p Nullen, also nicht mehr als hinten, das Wort liegt nicht in L. Widerspruch, also ist L nicht regulär.

19. (4 Punkte) (a) Wann akzeptiert ein NEA ein Wort? Warum kann man einen NEA trotzdem mit einem gewöhnlichen Programm ohne Raten simulieren?

    (b) Was sagt der Satz von Kleene?

    (c) Eine Sprache erfüllt die Bedingungen des Pumping-Lemmas. Ist sie dann regulär?

    (d) Warum kann kein endlicher Automat prüfen, ob ein Ausdruck korrekt geklammert ist? Welches Modell kann es?

Lösung:

(a) Wenn es mindestens einen Weg durch den Automaten gibt, der nach dem Lesen des ganzen Wortes in einem Endzustand endet. Man verfolgt alle Wege gleichzeitig, indem man sich die Menge der möglichen Zustände merkt. Das ist die Idee der Potenzmengenkonstruktion.

(b) DEAs, NEAs und reguläre Ausdrücke (und reguläre Grammatiken) beschreiben genau dieselben Sprachen, die regulären Sprachen.

(c) Nicht unbedingt. Das Pumping-Lemma gibt nur eine Eigenschaft an, die jede reguläre Sprache hat. Man kann damit zeigen, dass eine Sprache nicht regulär ist, aber nicht, dass sie regulär ist.

(d) Der Automat müsste zählen, wie viele Klammern noch offen sind, und diese Zahl kann beliebig groß werden. Mit endlich vielen Zuständen geht das nicht (Beweis mit dem Pumping-Lemma und $(^p)^p$). Ein Kellerautomat kann es: Er legt für jede öffnende Klammer ein Zeichen auf den Stapel und nimmt für jede schließende eines weg.

20. (4 Punkte) Schreibe ein Programm, das prüft, ob ein regulärer Ausdruck eine Sprache richtig beschreibt. Es soll für alle Wörter über $\{a, b\}$ mit höchstens 10 Zeichen vergleichen, ob `re.fullmatch` mit dem Ausdruck dasselbe Ergebnis liefert wie eine direkte Prüfung der Eigenschaft, und die Wörter ausgeben, bei denen es nicht stimmt. Teste es mit der Sprache der Wörter, die *aa* oder *bb* enthalten, und einmal mit dem falschen Ausdruck `(a|b)*aa(a|b)*`.

Lösung:

```
import itertools
import re

def woerter(alphabet, n):
    for laenge in range(n + 1):
        for t in itertools.product(alphabet, repeat=laenge):
            yield ''.join(t)

def abweichungen(muster, eigenschaft, alphabet='ab', n=10):
    return [w for w in woerter(alphabet, n)
            if (re.fullmatch(muster, w) is not None) != eigenschaft(w)]

def eigenschaft(w):
    return 'aa' in w or 'bb' in w

print(len(abweichungen('(a|b)*(aa|bb)(a|b)*', eigenschaft)))
falsch = abweichungen('(a|b)*aa(a|b)*', eigenschaft)
print(len(falsch), falsch[:3])
```

Ausgabe: 0 Abweichungen für den richtigen Ausdruck, 354 für den falschen, z.B. `bb, abb, bba`: Diese Wörter enthalten *bb*, aber kein *aa*.