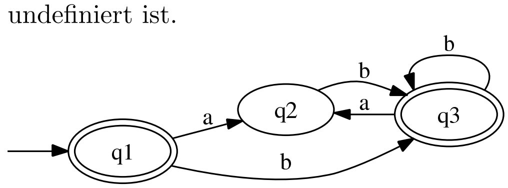
Lösungen zu den Aufgaben zur Vorlesung “*Formale Sprachen*”

Aufgabe 1: Eine mögliche Lösung sehen Sie nachstehend:

```
%%

%class Kasse
%standalone

%unicode

%{
    double mCount = 0;
%}

%eof{
    System.out.println("Total: " + mCount);
%eof}

%%

([1-9][0-9]*|[0-9])\.[0-9][0-9]  { mCount += new Double(yytext()); }
.|\n                             { /* skip */                       }
```

Aufgabe 2:

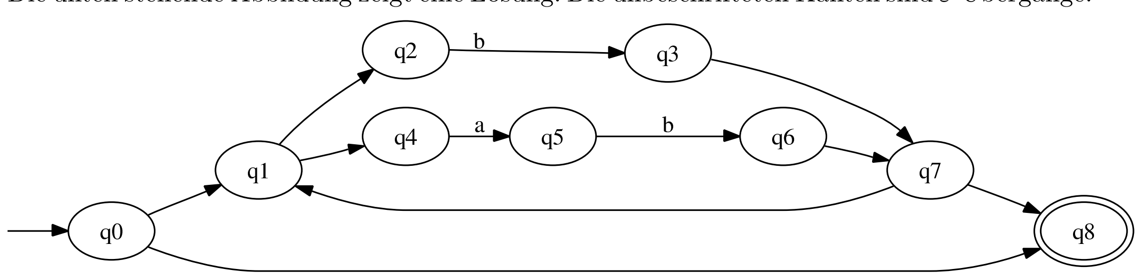
(a) Eine mögliche Lösung ist unten gezeigt. Beachten Sie, dass bei dieser Lösung der Wert $\delta(q_2, a)$ undefiniert ist.



Bemerkung: Der Zustand q_2 ist der einzige Zustand, der nicht-akzeptierend ist. Wir gelangen sowohl von q_1 als auch von q_3 in diesen Zustand, wenn wir ein “a” lesen. Lesen wir anschließend ein “b”, so gelangen wir wie vorgeschrieben in den akzeptierenden Zustand q_3 in dem wir dann noch beliebig viele “b”s einlesen können. Falls wir im Zustand q_2 ein “a” lesen, so stirbt der Automat, aber das ist auch richtig so, denn auf jedes “a” soll ja ein “b” folgen.

(b) Die Sprache wird von dem regulären Ausdruck $(b + ab)^*$ beschrieben.

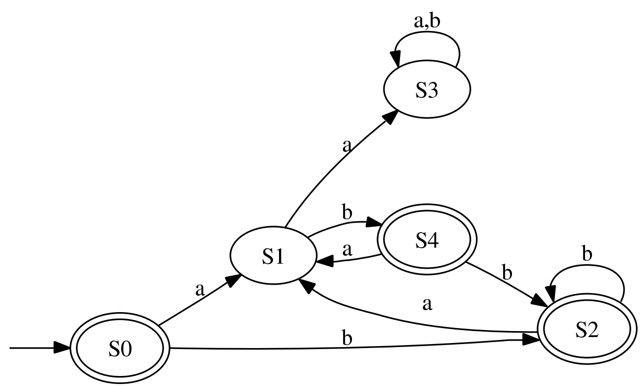
(c) Die unten stehende Abbildung zeigt eine Lösung. Die unbeschrifteten Kanten sind ε-Übergänge.



(d) Wir definieren die Zustände wir folgt:

1. $S_0 = \{q_0, q_1, q_2, q_4, q_8\}$,
2. $S_1 = \Delta(S_0, a) = \{q_5\}$,
3. $S_2 = \Delta(S_0, b) = \{q_3, q_7, q_1, q_2, q_4, q_8\}$,
4. $S_3 = \Delta(S_1, a) = \{\}$,
5. $S_4 = \Delta(S_1, b) = \{q_6, q_7, q_1, q_2, q_4, q_8\}$,
6. $\Delta(S_2, a) = \{q_5\} = S_1$,
7. $\Delta(S_2, b) = \{q_3, q_7, q_1, q_2, q_4, q_8\} = S_2$,
8. $\Delta(S_3, a) = \{\}$,
9. $\Delta(S_3, b) = \{\}$,
10. $\Delta(S_4, a) = \{q_5\} = S_1$,
11. $\Delta(S_4, b) = \{q_3, q_7, q_1, q_2, q_4, q_8\} = S_2$.

Der Startzustand ist S_0, die Zustände S_0, S_2 und S_4 sind akzeptierend. Die folgende Abbildung zeigt den deterministischen Automaten.



(e) Wir gehen wir im Skript beschrieben vor und erstellen eine Tabelle.

1. Im ersten Schritt erkennen wir, dass die akzeptierenden Zustände S_0, S_2 und S_4 von den nicht-akzeptierenden Zuständen S_1 und S_3 unterscheidbar sind. Damit hat der erste Entwurf unserer Tabelle die folgende Gestalt:

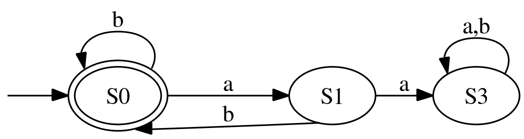
	S_0	S_1	S_2	S_3	S_4
S_0	$\sim$	1		1	
S_1	1	$\sim$	1		1
S_2		1	$\sim$	1	
S_3	1		1	$\sim$	1
S_4		1		1	$\sim$

2. Als nächstes erkennen wir, dass die Zustände S_1 und S_3 unterscheidbar sind, denn es gilt
$$\delta(S_1, b) = S_4, \quad \delta(S_3, b) = S_3 \quad \text{und} \quad S_4 \not\sim S_3.$$
Auf der anderen Seite sehen wir, dass die Zustände S_0 und S_2 nicht unterscheidbar sein können, denn es gilt
$$\delta(S_0, a) = S_1 = \delta(S_2, a) \quad \text{und } \delta(S_0, b) = S_2 = \delta(S_2, b).$$
Wir folgern $S_0 \sim S_2$. Genauso sehen wir, dass $S_0 \sim S_4$ ist, denn es gilt
$$\delta(S_0, a) = S_1 = \delta(S_4, a) \quad \text{und } \delta(S_0, b) = S_2 = \delta(S_4, b).$$

Aus $S_0 \sim S_2$ und $S_0 \sim S_4$ folgt sofort $S_2 \sim S_4$, Damit haben wir jetzt alle möglichen Äquivalenzen untersucht und unsere Tabelle hat die folgende Form:

	S_0	S_1	S_2	S_3	S_4
S_0	$\sim$	1	$\sim$	1	$\sim$
S_1	1	$\sim$	1	2	1
S_2	$\sim$	1	$\sim$	1	$\sim$
S_3	1	2	1	$\sim$	1
S_4	$\sim$	1	$\sim$	1	$\sim$

Die Zustände S_0, S_2 und S_4 sind also paarweise äquivalent. Die nachstehende Abbildung zeigt die Lösung.



Aufgabe 3: Wir führen den Beweis indirekt und nehmen an, dass L_P eine reguläre Sprache wäre. Nach dem Pumping-Lemma gibt es dann eine natürliche Zahl n, so dass gilt:

$$\forall s \in L_P : \big(|s| \geq n \rightarrow \exists u, v, w \in \Sigma^* : s = uvw \wedge v \neq \varepsilon \wedge |uv| \leq n \wedge (\forall h \in \mathbb{N} : uv^h w \in L_P)\big).$$

In Worten heißt dass: Für alle $s \in L_P$, für die $|s| \geq n$ gilt, gibt es eine Zerlegung

$$s = uvw,$$

so dass $v \neq \varepsilon$ und $|uv| \leq n$ gilt. Wir setzen nun

$$s := a^n b^{2 \cdot n} a^n.$$

Dann gilt $s^r = s$ und damit $s \in L_P$. Weiter ist $|s| = 4 \cdot n \geq n$. Also gibt es nach dem Pumping-Lemma eine Zerlegung von s der Form

$$a^n b^{2 \cdot n} a^n = uvw$$

mit den Eigenschaften $v \neq \varepsilon$ und $|uv| \leq n$. Aus $|uv| \leq n$ folgt, dass der String uv ganz in dem String a^n enthalten ist. Damit müssen die Strings u, v und w folgende Form haben:

$$u = a^i, \quad v = a^j \quad \text{und} \quad w = a^k b^{2 \cdot n} a^n,$$

wobei dann

$$i + j + k = n$$

gelten muss. Wegen $v \neq \varepsilon$ wissen wir, dass $j > 0$ ist. Wir betrachten jetzt den String $uw = uv^0 w$, der nach Aussage des Pumping-Lemmas in der Sprache L_P liegen muss. Es gilt

$$uw = a^i a^k b^{2 \cdot n} a^n.$$

Aus $i + j + k = n$ und $j > 0$ folgt offenbar

$$i + k \neq n$$

und damit ist uw keine Palindrom.

Aufgabe 4:

(a) Wir setzen $G = \langle \{S, T\}, \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, R, S\rangle$, wobei die Regeln R wie folgt gegeben sind:

$$S \to \mathtt{a}S\mathtt{c} \mid T,$$

$$T \to \mathtt{b}T\mathtt{c} \mid \varepsilon.$$

Die möglichen Ableitungen von T haben die Form

$$T \Rightarrow \mathtt{b}T\mathtt{c} \Rightarrow \mathtt{b}^2T\mathtt{c}^2 \Rightarrow \cdots \Rightarrow \mathtt{b}^lT\mathtt{c}^l \Rightarrow \mathtt{b}^l\mathtt{c}^l,$$

Dies zeigt, dass für die durch T beschriebene Sprache folgendes gilt:

$$L(T) = \{\mathtt{b}^l\mathtt{c}^l \mid l \in \mathbb{N}\}.$$

Die möglichen Ableitungen von S haben die Form

$$S \Rightarrow \mathtt{a}S\mathtt{c} \Rightarrow \mathtt{a}^2S\mathtt{c}^2 \Rightarrow \cdots \Rightarrow \mathtt{a}^kS\mathtt{c}^k \Rightarrow \mathtt{a}^kT\mathtt{c}^k \Rightarrow^* \mathtt{a}^k\mathtt{b}^l\mathtt{c}^l\mathtt{c}^k.$$

Dies zeigt, dass für die durch S beschriebene Sprache folgendes gilt:

$$L(S) = \{\mathtt{a}^k\mathtt{b}^l\mathtt{c}^{l+k} \mid l \in \mathbb{N}\}.$$

(b) Wir können die Sprache L_2 auf die Sprache L_1 zurück führen, denn wenn $h \geq k + l$ ist, heißt dies nichts anderes als dass es ein $j \in \mathbb{N}$ gibt, so dass $h = k + l + j$ gilt. Damit haben wir

$$L_2 = \{\mathtt{a}^k\mathtt{b}^l\mathtt{c}^{l+k}\mathtt{c}^j \mid l \in \mathbb{N} \wedge j \in \mathbb{N}\}.$$

Dies zeigt, dass die Wörter aus L_2 aus Wörtern von L_1 bestehen, an die noch der Buchstabe "$\mathtt{c}$" beliebig oft herangehängt wird. Dies führt zu folgender Grammatik:

$$G = \langle \{\widehat{S}, S, T, C\}, \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, R, \widehat{S}\rangle,$$

wobei die Regeln wie folgt gegeben sind:

$$\widehat{S} \to SC,$$

$$S \to \mathtt{a}S\mathtt{c} \mid T,$$

$$T \to \mathtt{b}T\mathtt{c} \mid \varepsilon,$$

$$C \to \mathtt{c}C \mid \varepsilon.$$

Aufgabe 5: Im Skript hatten wir gesehen, dass Regeln der Form

$$A \rightarrow A\beta_1 \mid \cdots \mid A\beta_k \mid \gamma_1 \mid \cdots \mid \gamma_l$$

durch Einführen einer neuen Variablen L in die folgenden nicht-links-rekursiven Regeln überführt werden können:

$$\begin{array}{lcl} A & \rightarrow & \gamma_1\, L \;\mid\; \gamma_2\, L \;\mid\; \cdots \;\mid\; \gamma_l\, L \\ L & \rightarrow & \beta_1\, L \;\mid\; \beta_2\, L \;\mid\; \cdots \;\mid\; \beta_k\, L \;\mid\; \varepsilon \end{array}$$

Wir führen also drei neue syntaktische Variablen L_S, L_A und L_B ein und erhalten dann die folgenden Regeln:

$$\begin{array}{lcl} S & \rightarrow & aBSL_S \mid L_S \\ L_S & \rightarrow & abL_S \mid \varepsilon \\[1ex] A & \rightarrow & bL_A \\ L_A & \rightarrow & aAL_A \mid \varepsilon \\[1ex] B & \rightarrow & aL_B \\ L_B & \rightarrow & aBL_B \mid \varepsilon \end{array}$$

Aufgabe 6:

(a) Die Grammatik $G = \langle \{E\}, \{N\}, R, E \rangle$, bei der die Regeln R wie folgt gegeben sind, leistet das Gewünschte:

$$\begin{array}{lcl} E & \rightarrow & \texttt{'+'}\; E\; E \\ & \mid & \texttt{'-'}\; E\; E \\ & \mid & \texttt{'*'}\; E\; E \\ & \mid & \texttt{'/'}\; E\; E \\ & \mid & \texttt{N} \end{array}$$

(b) Ein ANTLR-Programm könnte wie folgt aussehen:

```
grammar Program;

program : expr { System.out.println($expr.result); } ;

expr returns [int result]
     : '+' e1 = expr e2 = expr { $result = $e1.result + $e2.result; }
     | '-' e1 = expr e2 = expr { $result = $e1.result - $e2.result; }
     | '*' e1 = expr e2 = expr { $result = $e1.result * $e2.result; }
     | '/' e1 = expr e2 = expr { $result = $e1.result / $e2.result; }
     | NUMBER { $result = new Integer($NUMBER.text); }
     ;

NUMBER: ('0'..'9')|('1'..'9')('0'..'9')*;
WS    : (' '|'\t'|'\n'|'\r') { skip(); };
```

Aufgabe 7:

(a) Es sei $\Sigma = \{a, b\}$. Wir definieren die Sprachen L_1 und L_2 wie folgt:

$$L_1 = \{a^k b^l \mid k, l \in \mathbb{N}\} \quad \text{und} \quad L_2 = \{a^k b^l \mid k, l \in \mathbb{N} : k = l\}.$$

Dann gilt offenbar

$$L_2 = L_1 \cap (\Sigma^* \backslash L).$$

Die Sprache L_1 ist offenbar regulär, denn sie wird durch den regulären Ausdruck $a^* b^*$ beschrieben. Die Sprache L_2 ist identisch mit der Sprache

$$\{a^k b^k \mid k \in \mathbb{N}\}$$

und von dieser Sprache haben wir bereits im Skript gezeigt, dass sie nicht regulär ist. Wäre nun L regulär, so wäre auch das Komplement von L, die Sprache $\Sigma^* \backslash L$ regulär. Da der Schnitt zweier regulärer Sprache ebenfalls regulär ist, wäre dann auch L_2 regulär, im Widerspruch zu dem in der Vorlesung gezeigten Ergebnis.

(b) Wir geben eine Grammatik an, die diese Sprache erzeugt:

$$\begin{aligned} S &\rightarrow A \mid B \\ A &\rightarrow aG \mid aA \\ B &\rightarrow Gb \mid Bb \\ G &\rightarrow \varepsilon \mid aGb \end{aligned}$$

Die Idee hinter dieser Grammatik ist folgende:

1. $L(G) = \{a^k b^l \mid k, l \in \mathbb{N} : k = l\}$,
2. $L(A) = \{a^k b^l \mid k, l \in \mathbb{N} : k > l\}$,
3. $L(B) = \{a^k b^l \mid k, l \in \mathbb{N} : k < l\}$,
4. $L(S) = \{a^k b^l \mid k, l \in \mathbb{N} : k < l \vee k > l\}$.

Aufgabe 8: Wir führen den Beweis indirekt und nehmen an, $L_{\mathbb{P}}$ wäre regulär. Nach dem Pumping-Lemma gibt es dann eine Zahl n, so dass es für alle Strings $s \in L_{\mathbb{P}}$, deren Länge größer-gleich n ist, eine Zerlegung

$$s = uvw$$

mit den folgenden drei Eigenschaften gibt:

(a) $v \neq \varepsilon$,

(b) $|uv| \leq n$ und

(c) $\forall h \in \mathbb{N} : uv^h w \in L_{\mathbb{P}}$.

Wir wählen nun eine Primzahl p, die größer-gleich $n+2$ ist und setzen $s = \mathtt{a}^p$. Dann gilt $|s| = p \geq n$ und die Voraussetzung des Pumping-Lemmas ist erfüllt. Wir finden also eine Zerlegung von $\mathtt{a}^p$ der Form

$$\mathtt{a}^p = uvw$$

mit den oben angegebenen Eigenschaften. Aufgrund der Gleichung $s = uvw$ können die Teilstrings u, v und w nur aus dem Buchstaben "$\mathtt{a}$" bestehen. Also gibt es natürliche Zahlen x, y, und z so dass gilt:

$$u = \mathtt{a}^x, \quad v = \mathtt{a}^y \quad \text{und} \quad w = \mathtt{a}^z.$$

Für x, y und z gilt dann Folgendes:

(a) $x + y + z = p$,

(b) $y \neq 0$,

(c) $x + y \leq n$,

(d) $\forall h \in \mathbb{N} : x + h \cdot y + z \in \mathbb{P}$.

Setzen wir in der letzten Gleichung für h den Wert $(x + z)$ ein, so erhalten wir

$$x + (x + z) \cdot y + z \in P.$$

Wegen $x + (x + z) \cdot y + z = (x + z) \cdot (1 + y)$ hätten wir dann

$$(x + z) \cdot (1 + y) \in \mathbb{P}.$$

Dass kann aber nicht sein, denn wegen $y > 0$ ist der Faktor $1 + y$ von 1 verschieden und wegen $x + y \leq n$ und $x + y + z = p$ und $p \geq n + 2$ wissen wir, dass $z \geq 2$ ist, so dass auch der Faktor $(x + z)$ von 1 verschieden ist. Damit kann das Produkt $(x + z) \cdot (1 + y)$ aber keine Primzahl mehr sein und wir haben einen Widerspruch zu der Annahme, dass $L_{\mathbb{P}}$ regulär ist.

Aufgabe 9: Die Grammatik

$$G = \langle \{E_1, E_2, \cdots, E_n\}, \{\textsc{Number}, \textsc{Var}, \#_1, \cdots, \#_n\}, R, E_1 \rangle,$$

deren Regeln wie folgt gegeben sind, leistet das Gewünschte:

$$\begin{array}{lcll}
E_k & \to & E_k \; \#_k \; E_{k+1} & \\
 & \mid & E_{k+1} & \text{für alle } k \in \{1, \cdots, k-1\} \\[1ex]
E_n & \to & \textsc{Var} & \\
 & \mid & \textsc{Number} &
\end{array}$$

Hier habe ich vorausgesetzt, dass die Operatoren $\mathtt{\#}_k$ linksassoziativ sind.