Theoretische Informatik: Endliche Automaten, Formale Sprachen und Grammatiken

Marko Livajusic

24. November 2024

Inhaltsverzeichnis

1. Deterministische Endliche Automaten

1.1 Transduktor

Definition 1 *Ein Transduktorautomat* $\mathcal{T} : \{\Sigma, A, Z, z_0, \delta, \lambda\}$ *ist ein deterministicher endlicher Automat ohne einen Endzustand.*

$$\begin{aligned}
\mathbf{\Sigma} &: \text{Eingabealphabet} \\
\mathbf{A} &: \text{Ausgabealphabet} \\
\mathbf{Z} &: \text{Zustandsmenge} \\
\mathbf{z_0} \in Z &: \text{Startzustand} \\
\delta : \Sigma \times Z \to Z &: \text{Überführungsfunktion} \\
\lambda : \Sigma \times Z \to A^* &: \text{Ausgabefunktion}
\end{aligned}$$

1.1.1 Mealy-Automat

Definition 2 *Ein Mealy-Automat* [1] *ist ein Transduktor, dessen Ausgabe von der* ***Überführungsfunktion*** δ *und vom aktuellen* ***Zustand*** z_n *abhängig ist.*

1.2 Akzeptor

Definition 3 *Ein Akzeptor* $\mathcal{A} : \{\Sigma, Z, z_0, \delta, F\}$ *ist ein deterministicher endlicher Automat, der die Eingabe überprüft und keine Ausgabe besitzt. Er lässt sich wie folgt beschreiben:*

[1] für die Klausur irrelevant.

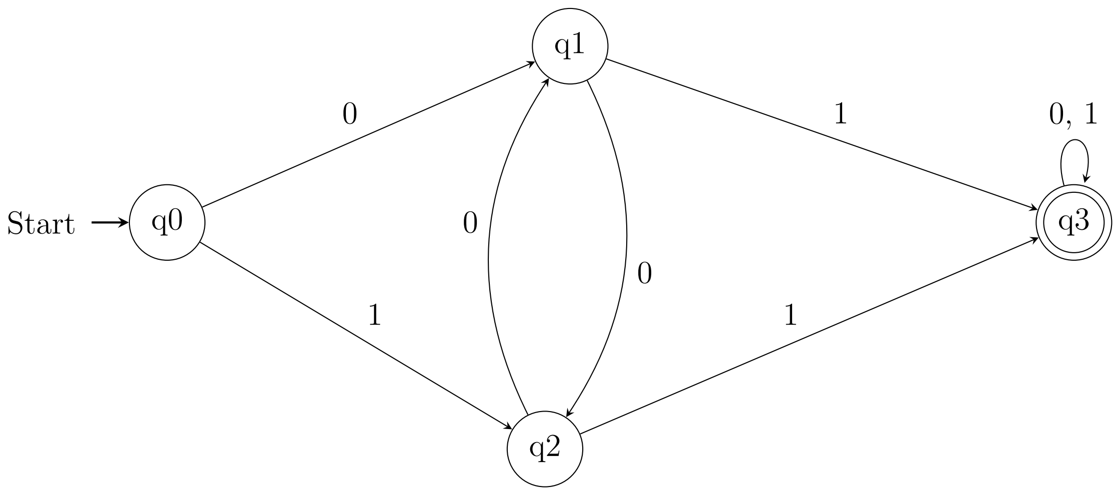
$$\begin{aligned}\Sigma &: \text{Eingabealphabet}\\ Z &: \text{Zustandsmenge}\\ z_0 &: \text{Startzustand}\\ \delta &: \text{Überführungsfunktion}\\ F &: \text{Endzustandsmenge}\end{aligned}$$

1.2.1 Moore-Automat

Definition 4 *Ein Moore-Automat ist ein Transduktor, dessen Ausgabe vom aktuellen* ***Zustand*** z_n *abhängig ist.*

1.2.2 Minimierung von DEAs

Zu minimieren sei folgender DEA:



Diagonale als äquivalent markieren:

Zustand	q_0	q_1	q_2	q_3
q_0	$\equiv$			
q_1		$\equiv$		
q_2			$\equiv$	
q_3				$\equiv$

Felder, wo ein Zustand auf einen Endzustand trifft, streichen

Zustand	q_0	q_1	q_2	q_3
q_0	$\equiv$			
q_1		$\equiv$		
q_2			$\equiv$	
q_3	X	X	X	$\equiv$

Eine Übergangstabelle mit übrigen Zuständen erstellen. Die Zustandspaare, die auf einen bereits gestrichenen Zustandspaar abgebildet werden, streichen

Zustand	0	1
(q_0, q_1)	(q_1, q_2)	$(\mathbf{q_2}, \mathbf{q_3})$
(q_0, q_2)	(q_1, q_1)	$(\mathbf{q_2}, \mathbf{q_3})$
(q_1, q_2)	(q_2, q_1)	(q_3, q_3)

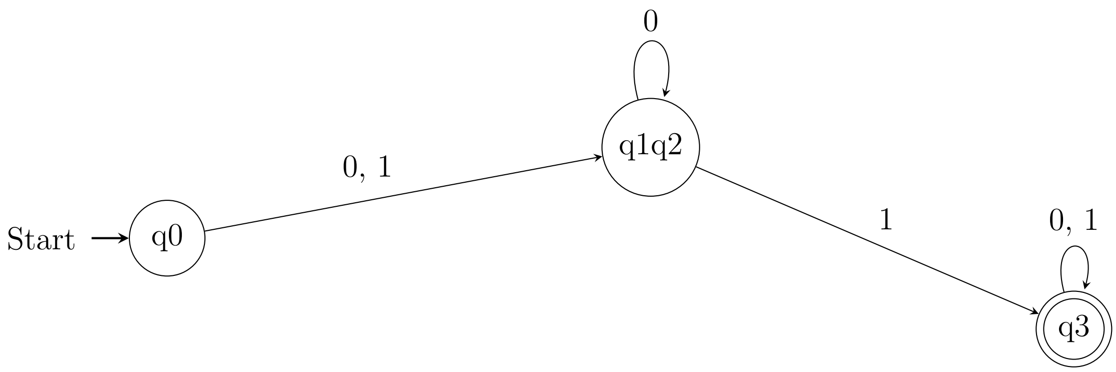
Die neue Tabelle sieht dann so aus:

Zustand	q_0	q_1	q_2	q_3
q_0	$\equiv$			
q_1	X	$\equiv$		
q_2	X		$\equiv$	
q_3	X	X	X	$\equiv$

Die leeren Felder als äquivalent markieren:

Zustand	q_0	q_1	q_2	q_3
q_0	$\equiv$			
q_1	X	$\equiv$		
q_2	X	$\equiv$	$\equiv$	
q_3	X	X	X	$\equiv$

Spaltenweise die Zustände zusammenfassen:



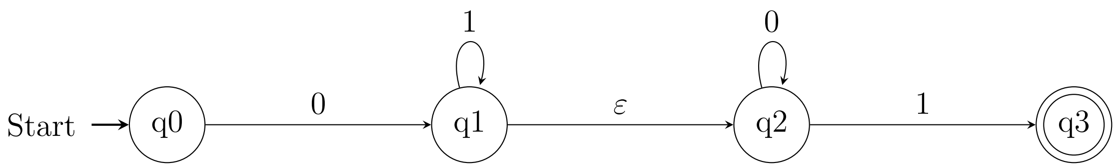
2. Nichtdeterministische Endliche Automaten

2.1 ϵ-NEAs

Definition 5 *Ein ϵ-NEA ist ein Akzeptor, der ϵ-Übergänge besitzt und deshalb mit dem leeren Wort Zustände wechseln kann.*

2.1.1 ϵ-NEA$\rightarrow$ NEA

Gegeben sei folgendes Zustandsdiagramm eines ϵ-NEA, welches in einen NEA umgewandelt werden soll:



Zuerst wird eine leere Übergangstabelle erstellt:

Zustand	0	1
q_0		
q_1		
q_2		
q_3		

Danach wird für jedes Eingabesymbol eine Tabelle mit der ϵ-Hülle erstellt:

Zustand	ϵ*	0	ϵ*
q_0			

Wie oben zu sehen ist, wird zuerst der Startzustand q_0 eingetragen. Danach wird die ϵ-Hülle des Zustands q_0 berechnet und eingetragen.

Definition 6 *Eine ϵ-Hülle ist die Menge aller Zustände, die ein Zustand q_n mit dem leeren Wort ϵ erreichen kann.*

Da im vorigen Beispiel q_0 mit dem leeren Wort keinen anderen Zustand als sich selbst erreichen kann, wird für dessen ϵ-Hülle q_0 eingetragen.

Die nächte Spalte steht für den Zustand, der erreicht wird, wenn bei q_0 das Eingabesymbol 0 eingegeben wird. Dies ist in diesem Beispiel der Zustand q_1:

Zustand	ϵ^*	0	ϵ^*
q_0	q_0	$\mathbf{q_1}$	

Die letzte Spalte bezieht sich auf die ϵ-Hülle des Zustands aus der mittleren Spalte, welcher hier fettgedruckt steht. Die ϵ-Hülle von q_1 ist dabei $\{q_1, q_2\}$. Diese wird ebenfalls eingetragen:

Zustand	ϵ^*	0	ϵ^*
q_0	q_0	q_1	$\{q_1, q_2\}$

Diese ϵ-Hülle $\{q_1, q_2\}$ repräsentiert dabei die Zustände, die q_0 bei der Eingabe von 0 erreicht werden. Deshalb können diese in die Übergangstabelle eingetragen werden:

Zustand	0	1
q_0	$\{q_1, q_2\}$	
q_1		
q_2		
q_3		

Dieser Vorgang wird für alle Zustände durchgeführt, sowohl für die Eingabe von 0 als auch von 1. Die Tabellen sehen nach dem Algorithmus wie folgt aus:

Zustand	ϵ^*	0	ϵ^*
$\{q_0\}$	$\{q_0\}$	$\{q_1\}$	$\{q_1, q_2\}$
$\{q_1\}$	$\{q_1\}$ $\{q_1, q_2\}$	$\emptyset$ $\{q_2\}$	$\emptyset$ $\{q_2\}$
$\{q_2\}$	$\{q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_3\}$	$\{q_3\}$	$\emptyset$	$\emptyset$

Zustand	ϵ*	1	ϵ*
$\{q_0\}$	$\{q_0\}$	$\emptyset$	$\emptyset$
$\{q_1\}$	$\{q_1\}$ $\{q_2\}$	$\{q_1\}$ $\{q_3\}$	$\{q_1, q_2\}$ $\{q_3\}$
$\{q_2\}$	$\{q_2\}$	$\{q_3\}$	$\{q_3\}$
$\{q_3\}$	$\{q_3\}$	$\emptyset$	$\emptyset$

Zustand	0	1
$\{q_0\}$	$\{q_1, q_2\}$	$\emptyset$
$\{q_1\}$	$\{q_2\}$	$\{q_1, q_2, q_3\}$
$\{q_2\}$	$\{q_2\}$	$\{q_3\}$
$\{q_3\}$	$\emptyset$	$\emptyset$

Noch sollen die Endzustände ermittelt werden. Zu den Endzuständen gehört der Endzustand aus dem ϵ-NEAund die Zustände, die durch das leere Wort ϵ in den ursprünglichen Endzustand gelangen können. Deshalb wird in diesem Fall nur q_3 der Endzustand. Gezeichnet sieht das neue Zustandsdiagramm wie folgt aus:

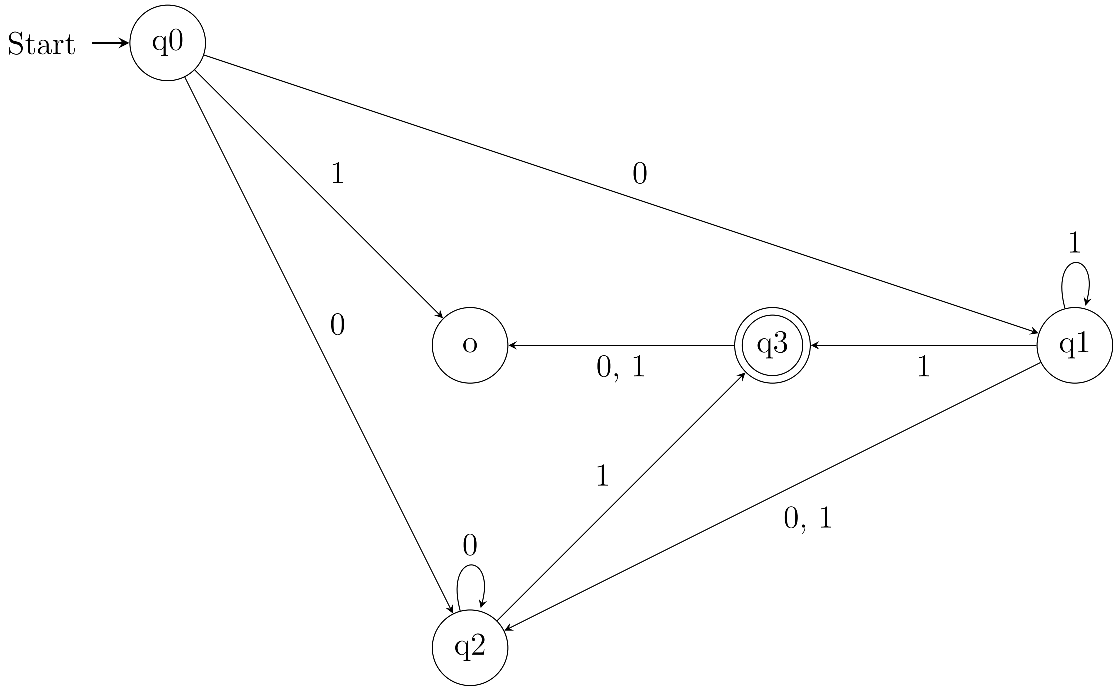
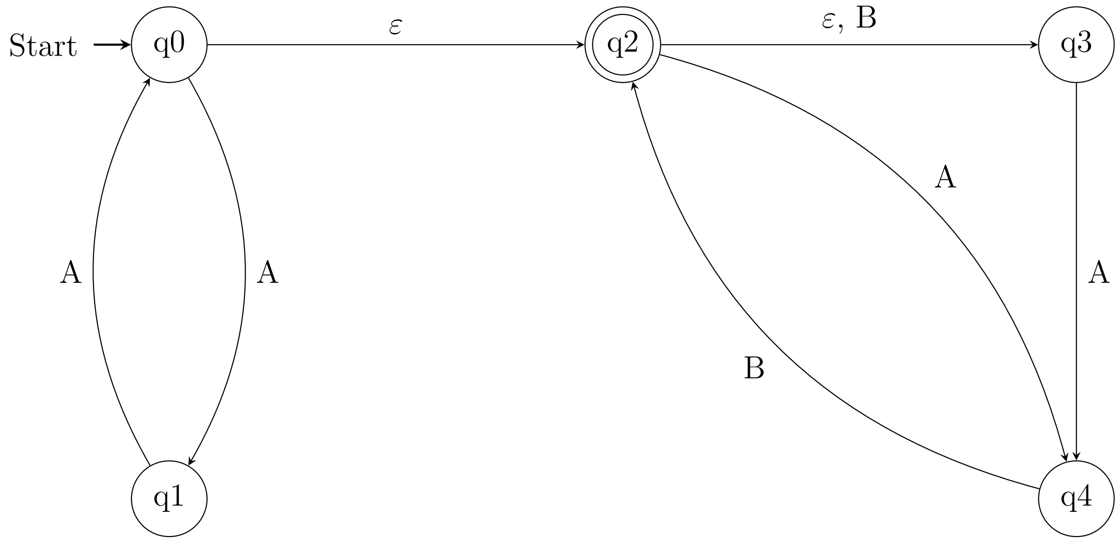


Abbildung 2.1: Der neue NEA, ohne ϵ-Übergänge.

„o“ steht hier für die leere Menge $\emptyset$.

2.1.2 ϵ-NEA$\rightarrow$ DEA

Es sei folgendes Zustandsdiagramm eines ϵ-NEAs gegeben:



Die Umwandlung in ein DEA geschieht wie üblich mit der Potenzmengenkonstruktion:

Zustand	A	B
$\rightarrow \{q_0\}$	$\{q_1, q_4\}$	$\{q_3\}$
$\{q_1, q_4\}$	$\{q_0\}$	$\{q_2*\}$
$\{q_3\}$	$\{q_4\}$	$\emptyset$
$\{q_2*\}$	$\{q_4\}$	$\{q_3\}$
$\{q_4\}$	$\emptyset$	$\{q_2\}$
$\emptyset$	$\emptyset$	$\emptyset$

Anschlißend wird das neue Zustandsdiagramm des DEAs gezeichnet. qE repräsentiert dabei die leere Menge $\emptyset$.

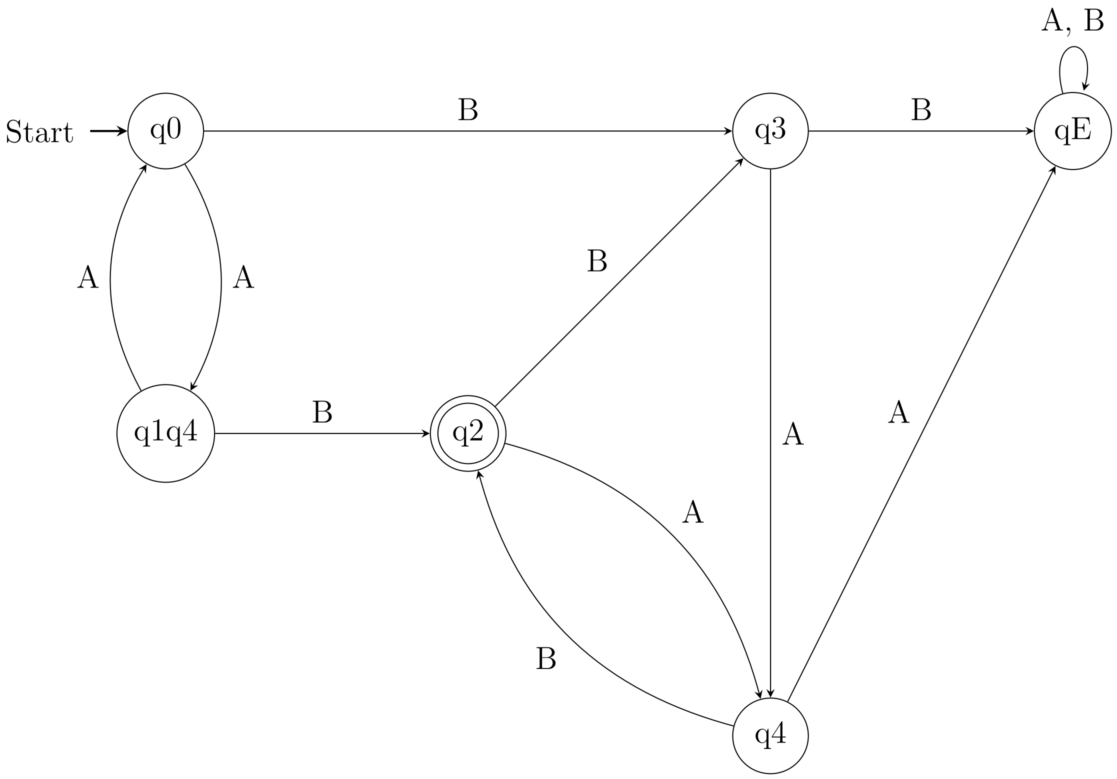
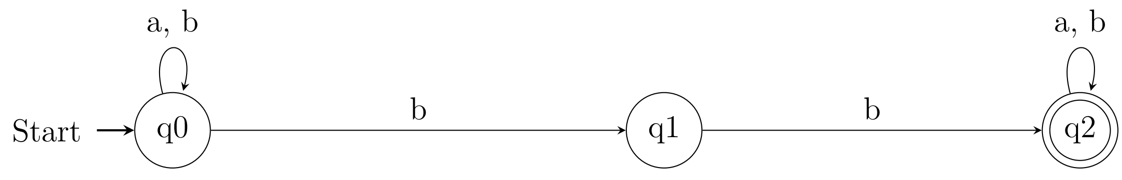


Abbildung 2.2: Umwandlung von ϵ-NEA zu DEA. Dieser ist jedoch nicht zwangsläufig optimal bzw. minimal.

2.2 NEA → DEA (Potenzmengenkonstruktion)

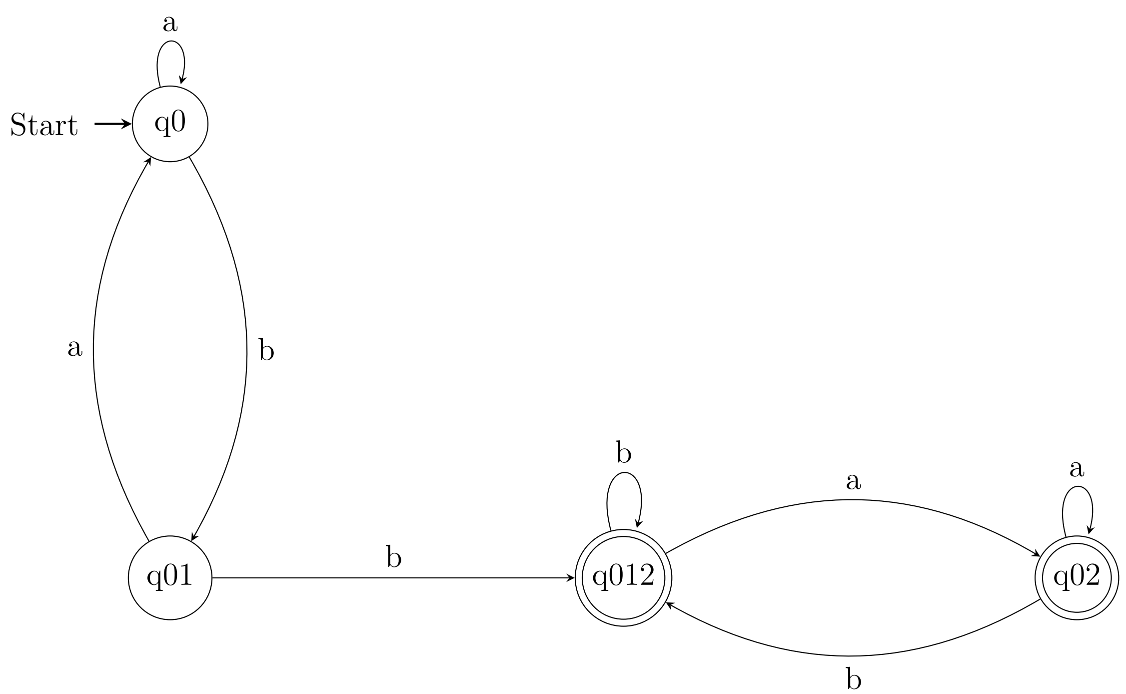
Dieser NEA soll in einen DEA umgewandelt werden:



Vorgehen: Es wird zuerst eine Übergangstabelle aufgestellt und geschaut, welche Zustände neu auftreten.

Zustand	a	b
$\rightarrow \{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0, q_1, q_2{*}\}$
$\{q_0, q_1, q_2\}^*$	$\{q_0, q_2{*}\}$	$\{q_0, q_1, q_2{*}\}$
$\{q_0, q_2\}^*$	$\{q_0, q_2{*}\}$	$\{q_0, q_1, q_2{*}\}$

Danach wird aus dieser Übergangstabelle der DEA gezeichnet:



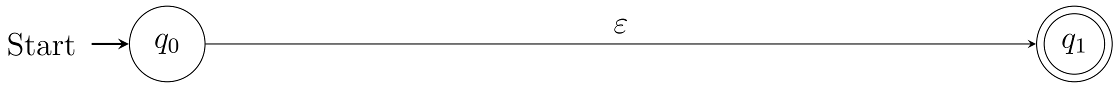
3. Reguläre Ausdrücke

$+$: wiederhole das Zeichen davor n-mal, wobei $\mathbf{n} > \mathbf{0}$
$*$: wiederhole das Zeichen davor n-mal, wobei $\mathbf{n} \geq \mathbf{0}$

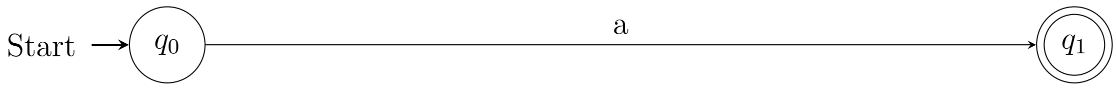
3.1 RegEx $\rightarrow$ ϵ-NEA

3.1.1 $R = \emptyset$

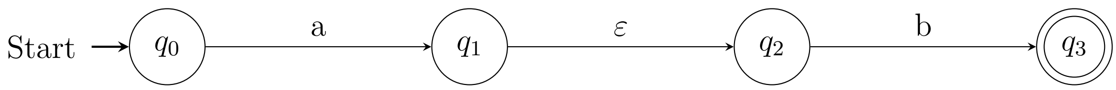
3.1.2 $R = \epsilon$



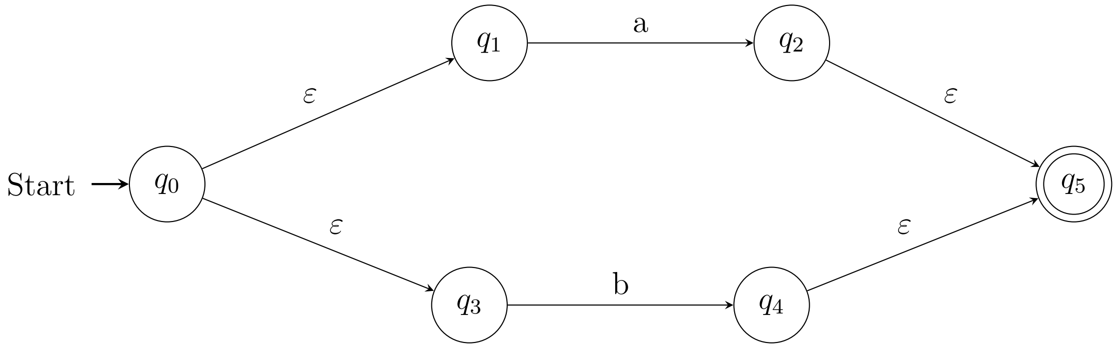
3.1.3 $R = a$



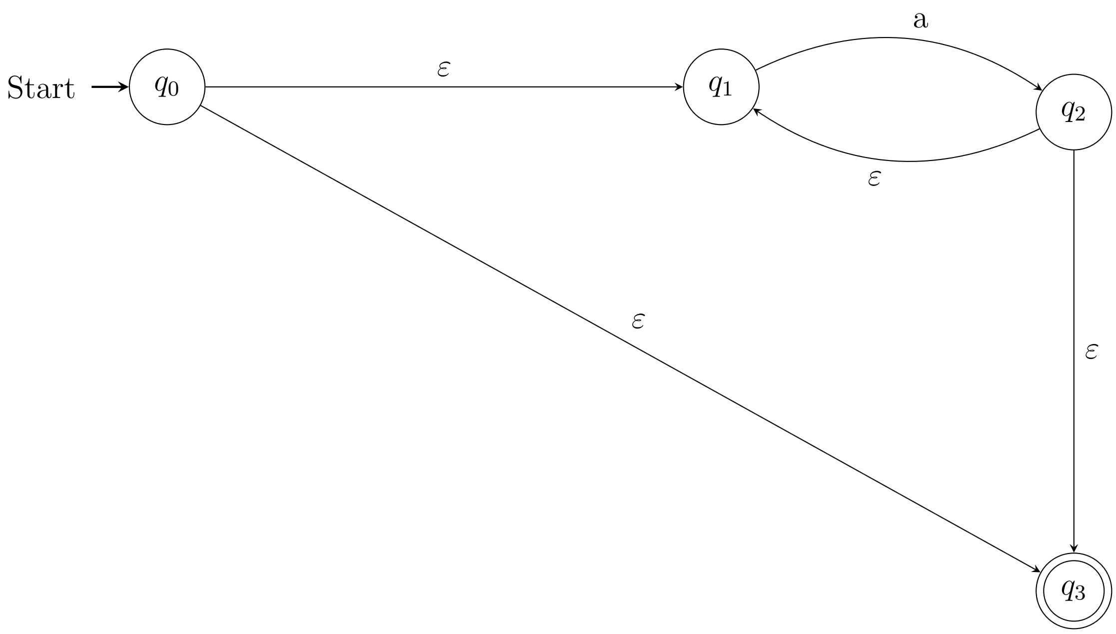
3.1.4 $R = ab$



3.1.5 $R = a|b$



3.1.6 $R = a^*$



Beispiel 1 *Es soll der reguläre Ausdruck* (0|1)*01 *in einen* ϵ*-NEA umgewandelt werden.*

Teil 2: Formale Sprachen

4. Formale Sprachen

4.1 Reguläre Sprachen

Definition 7 *Eine Sprache L ist dann regulär, wenn diese sich darstellen lässt mithilfe eines:*

1. *nichtdeterministischen endlichen Automatens*
2. *deterministischen endlichen Automatens*
3. *regulären Ausdrucks.*

4.2 Q3.2: Grammatiken

Definition 8 *Eine Grammatik G ist ein 4-Tupel $G = \{N, T, P, S\}$, wobei*

- *N das **Nichtterminalalphabet***
- *T das **Terminalalphabet***
- *P die **Produktionen***
- *S das **Startsymbol** ist.*

4.2.1 Typ 3 Grammatik (regulär)

Eine Grammatik G ist dann *regulär*, wenn in den Produktionen P

- links ein Nichtterminal und rechts ein oder mehrere Terminale vorkommen gefolgt von maximal einem Nichtterminal

4.3 Ableitung

Gegeben sei folgende Grammatik:

$$
\begin{aligned}
T &= \{x, y, z\} \\
N &= \{S, M, A, V\} \\
P &= \{ \\
S &\rightarrow A|M|V \\
A &\rightarrow (S + S) \\
M &\rightarrow (S \cdot S) \\
V &\rightarrow x|y|z \\
&\}
\end{aligned}
$$

Wie wird das Wort $(x \cdot (y + z))$ gebildet?

$$
\begin{aligned}
S \Rightarrow M \Rightarrow (S \cdot S) \\
\Rightarrow (v \cdot S) \Rightarrow (x \cdot S) \Rightarrow (x \cdot A) \Rightarrow \\
(x \cdot (S + S)) \Rightarrow (x \cdot (v + S)) \Rightarrow (x \cdot (y + S)) \Rightarrow (x \cdot (y + v)) \Rightarrow (x \cdot (y + z))
\end{aligned}
$$

4.3.1 Ableitungsbaum

Dies kann man auch mit einem Ableitungsbaum darstellen:

4.3.2 Syntaxdiagramme: Regeln

1. 1 Syntaxdiagramm $\hat{=}$ 1 Produktionsregel, wobei das Syntaxdiagramm der Name der Produktionsregel ist
2. Nichtterminale: eckig
3. Terminale: rund

4.4 Kontextfreie Sprachen

Gegeben sei folgende kontextfreie Grammatik:

$$
\begin{aligned}
&N = \{A, B, S\}\\
&T = \{a, b, \epsilon\}\\
&S = S\\
&P = \{\\
&\quad S \to AB\\
&\quad S \to ABA\\
&\quad A \to aA\\
&\quad A \to a\\
&\quad B \to Bb\\
&\quad B \to \epsilon\\
&\}
\end{aligned}
$$

4.4.1 Chomsky-Normalform

Definition 9 *Die Chomsky-Normalform ist eine Normalform für kontextfreie Grammatiken und ist die Voraussetzung für den* **??**.

Gegeben sei folgende Grammatik, die in die Chomsky-Normalform gebracht werden sollte:

$$
\begin{aligned}
&G = (N, T, P, S)\\
&N = \{A, B\}\\
&T = \{0\}\\
&P = \{\\
&\quad A \to BAB|B|\epsilon\\
&\quad B \to 00|\epsilon\}
\end{aligned}
$$

Um eine Grammatik G in die Chomsky-Normalform zu bringen, müssen 4 Regeln befolgt werden:

1. Wähle ein neues Startsymbol.
2. Eliminiere ϵ
3. Eliminiere *unit rules*, d.h. Nichtterminal auf ein Nichtterminal, bspw. $S \to A$
4. Verändere alle Regeln, wo mehr als ein Terminal vorkommt, bspw. $S \to 00$
5. Verändere alle Regeln, wo mehr als zwei Nichtterminale vorkommen, bspw. $S \to AB$

4.4.2 CYK-Algorithmus

Mit dem CYK-Algorithmus lässt sich sagen, ob ein Wort ω in einer kontextfreien Sprache liegt. Die Voraussetzung für den CYK-Algorithmus ist die **??**.

Beispiel 2 *Sei G eine Grammatik mit Produktionsregeln P, die definiert sind als:*

$$\begin{aligned} S &\to BC|AC|BA \\ A &\to AA|BB|a \\ B &\to BA|b \\ C &\to AC|c \end{aligned}$$

Nun bestimme man, ob das Wort ababac in $L(G)$ liegt.

.

$a \quad b \quad a \quad b \quad a \quad c$

Die unterste Zeile ist die 1. Zeile. Fangen wir (von links) mit dem ersten Feld der ersten Zeile, so sehen wir, dass ein Nichtterminalsymbol gesucht ist, welches das Wort a ableitet. Schaut man auf die Grammatik, so sieht man, dass lediglich die Produktionsregel A das Wort a ableitet, weshalb sie in das untere Feld eingetragen wird: .

$$\begin{array}{cccccc} \{A\} & \{A\} & \{B\} & \{B\} & \{B\} & \{C\} \\ a & b & a & b & a & c \end{array}$$

5. Registermaschine

5.1 Häufige Operationen

5.1.1 $R_1 == R_2$

Es gilt: $R_1 - R_2 = 0 \wedge R_2 - R_1 = 0$

```
load #10
store 1
load #2
store 2

load 1
sub 2
store 3

load 2
sub 1
store 4

load 3
jzero second_check
goto not_equal // else case

second_check:   load 4
jzero equal // R1-R2 UND R2-R1 sind 0

not_equal: END
equal: END
```

5.2 $R_1 < R_2$

Es gilt: $R_2 - R_1 \neq 0$

```
load #10
```

```
store 1
load #2
store 2

load 2
sub 1
jnzero proceed
end

proceed:    do_stuff
            end
```

5.3 $R_1 > R_2$

Es gilt: $R_1 - R_2 \neq 0$

```
load #10
store 1
load #2
store 2

load 1
sub 2
jnzero proceed
end

proceed:    do_stuff
            end
```