Theoretische Informatik

Prof.Dr.Christian Wagenknecht

Hochschule Zittau/Görlitz
Fachbereich Informatik

10. Mai 2007

Formale Sprachen und Automaten: Einführung, Teil 2

Kontrolle der Studienaufgabe

Geben Sie eine Grammatik für arithmetische Ausdrücke an. Die Ausdrücke sollen nur die Variablen a,b und c, sowie die Operatoren $+, -, *$ und $/$ für die entsprechenden zweistelligen Operationen enthalten dürfen. Außerdem können Klammern (und) verwendet werden, wobei bei deren Einsatz auf korrekte Klammerung geachtet werden muss. Beispiele: $a + b$, $a + (b * c)$, $a/(b - c * c)$.

Noam Chomsky, geb. 1928

Allg. Regelgestalt: $\alpha \longrightarrow \beta \mid \alpha \in (N \cup T)^* \setminus T^*$ und $\beta \in (N \cup T)^*$

Typ	Bezeichnung	Regelgestalt
0	unbeschränkt (Phrasenstrukturgrammatik; Semi-Thue-System – Norwegischer Mathematiker A. THUE, 1863-1922)	keine Einschränkung
1	kontextsensitiv	längenmonotone Regeln, d.h. $\|\alpha\| \leq \|\beta\|$; Ausnahme: $s \longrightarrow \varepsilon$ darf vorkommen, allerdings nur dann, wenn s in keiner Regel auf der rechten Seite vorkommt
2	kontextfrei	wie Typ 1 aber zusätzlich $\alpha \in N$; Regeln der Form $\alpha \longrightarrow \varepsilon$ sind erlaubt
3	regulär	wie Typ 2 aber zusätzlich haben *alle* Regeln *entweder* die Form $\alpha \longrightarrow x$ und ggf. $\alpha \longrightarrow xA$ (rechtslinear) *oder* $\alpha \longrightarrow x$ und ggf. $\alpha \longrightarrow Ax$ (linkslinear), wobei $x \in T$ und $A \in N$

Noam Chomsky, geb. 1928

Allg. Regelgestalt: $\alpha \longrightarrow \beta \mid \alpha \in (N \cup T)^* \setminus T^*$ und $\beta \in (N \cup T)^*$

Typ	Bezeichnung	Regelgestalt
0	unbeschränkt (Phrasenstrukturgrammatik; Semi-Thue-System – Norwegischer Mathematiker A. THUE, 1863-1922)	keine Einschränkung
1	kontextsensitiv	längenmonotone Regeln, d.h. $\|\alpha\| \leq \|\beta\|$; Ausnahme: $s \longrightarrow \varepsilon$ darf vorkommen, allerdings nur dann, wenn s in keiner Regel auf der rechten Seite vorkommt
2	kontextfrei	wie Typ 1 aber zusätzlich $\alpha \in N$; Regeln der Form $\alpha \longrightarrow \varepsilon$ sind erlaubt
3	regulär	wie Typ 2 aber zusätzlich haben *alle* Regeln *entweder* die Form $\alpha \longrightarrow x$ und ggf. $\alpha \longrightarrow xA$ (rechtslinear) *oder* $\alpha \longrightarrow x$ und ggf. $\alpha \longrightarrow Ax$ (linkslinear), wobei $x \in T$ und $A \in N$

Noam Chomsky, geb. 1928

Allg. Regelgestalt: $\alpha \longrightarrow \beta \mid \alpha \in (N \cup T)^* \setminus T^*$ und $\beta \in (N \cup T)^*$

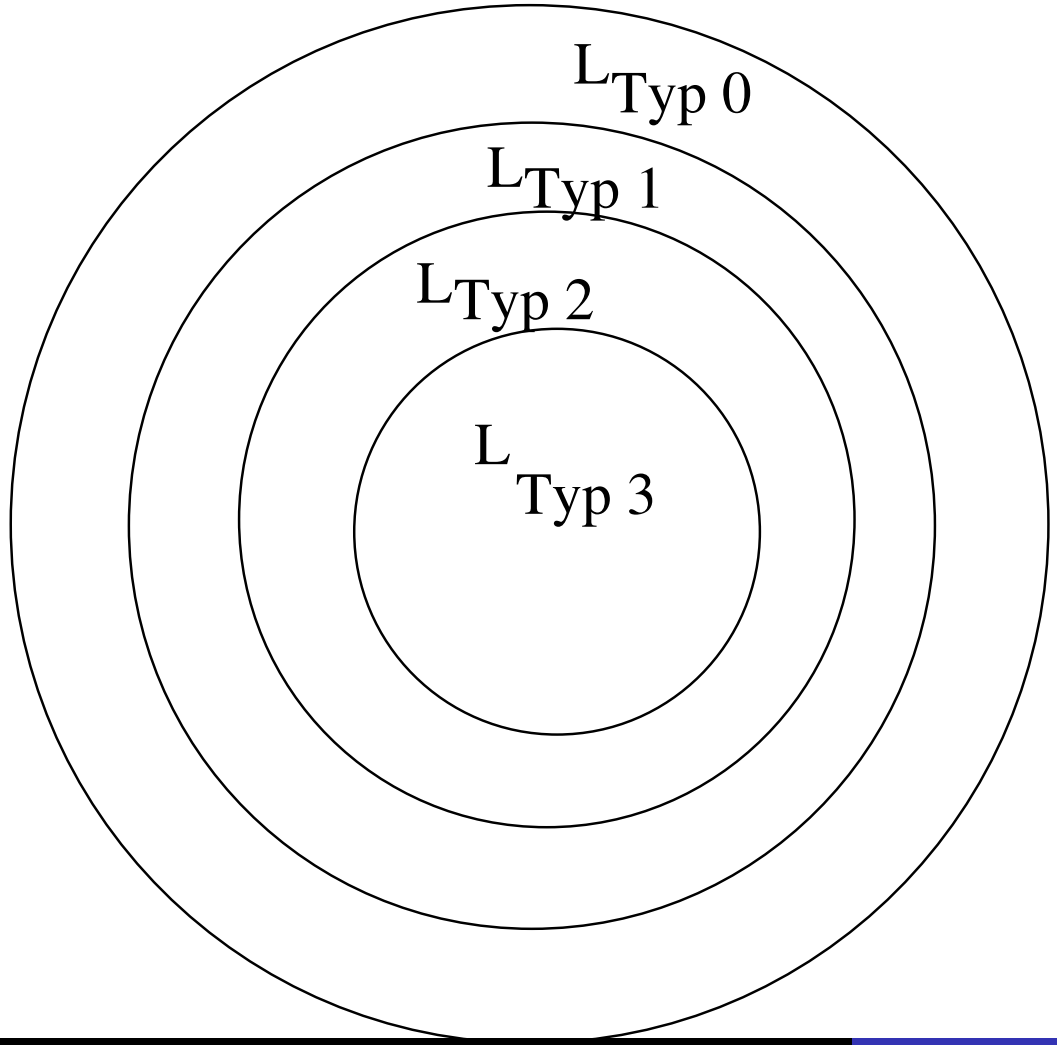
Typ	Bezeichnung	Regelgestalt
0	unbeschränkt (Phrasenstrukturgrammatik; Semi-Thue-System – Norwegischer Mathematiker A. THUE, 1863-1922)	keine Einschränkung
1	kontextsensitiv	längenmonotone Regeln, d.h. $\|\alpha\| \leq \|\beta\|$; Ausnahme: $s \longrightarrow \varepsilon$ darf vorkommen, allerdings nur dann, wenn s in keiner Regel auf der rechten Seite vorkommt
2	kontextfrei	wie Typ 1 aber zusätzlich $\alpha \in N$; Regeln der Form $\alpha \longrightarrow \varepsilon$ sind erlaubt
3	regulär	wie Typ 2 aber zusätzlich haben *alle* Regeln *entweder* die Form $\alpha \longrightarrow x$ und ggf. $\alpha \longrightarrow xA$ (rechtslinear) *oder* $\alpha \longrightarrow x$ und ggf. $\alpha \longrightarrow Ax$ (linkslinear), wobei $x \in T$ und $A \in N$

Noam Chomsky, geb. 1928

Allg. Regelgestalt: $\alpha \longrightarrow \beta \mid \alpha \in (N \cup T)^* \setminus T^*$ und $\beta \in (N \cup T)^*$

Typ	Bezeichnung	Regelgestalt
0	unbeschränkt (Phrasenstrukturgrammatik; Semi-Thue-System – Norwegischer Mathematiker A. THUE, 1863-1922)	keine Einschränkung
1	kontextsensitiv	längenmonotone Regeln, d.h. $\|\alpha\| \leq \|\beta\|$; Ausnahme: $s \longrightarrow \varepsilon$ darf vorkommen, allerdings nur dann, wenn s in keiner Regel auf der rechten Seite vorkommt
2	kontextfrei	wie Typ 1 aber zusätzlich $\alpha \in N$; Regeln der Form $\alpha \longrightarrow \varepsilon$ sind erlaubt
3	regulär	wie Typ 2 aber zusätzlich haben *alle* Regeln *entweder* die Form $\alpha \longrightarrow x$ und ggf. $\alpha \longrightarrow xA$ (rechtslinear) *oder* $\alpha \longrightarrow x$ und ggf. $\alpha \longrightarrow Ax$ (linkslinear), wobei $x \in T$ und $A \in N$

Hierarchie von Sprachklassen

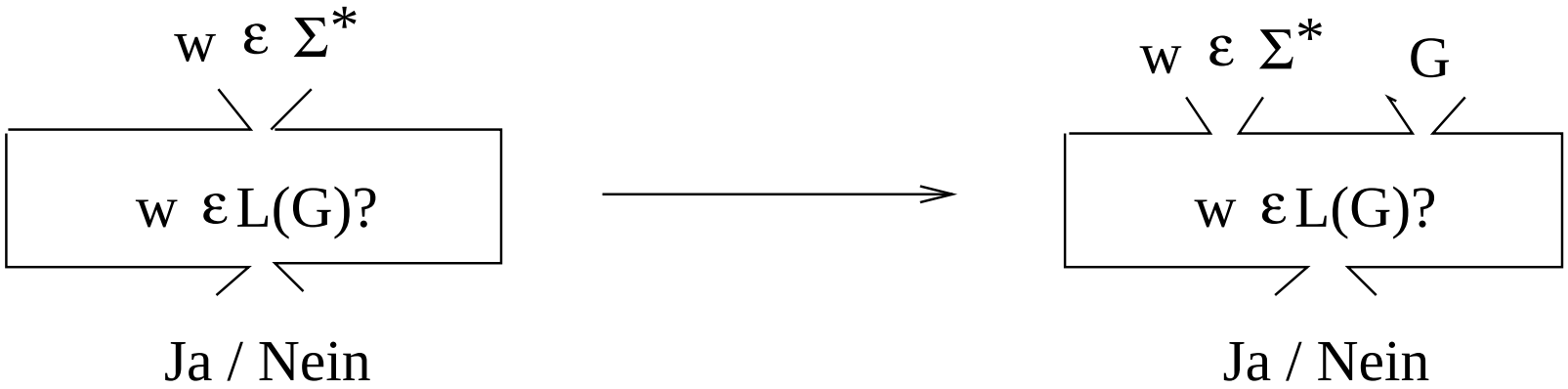


Das (allgemeine) Wortproblem

Verallgemeinerung der Frage $w \in_? L(G)$?: freie Wahl einer Grammatik/Sprache aus einer der vier Klassen (Chomsky-Typ-0,1,2,3)

Das (allgemeine) Wortproblem

Verallgemeinerung der Frage $w \in_? L(G)$?: freie Wahl einer Grammatik/Sprache aus einer der vier Klassen (Chomsky-Typ-0,1,2,3)



P(w) = Ja / Nein

P(w,G) = Ja / Nein

Entscheidbarkeit des Wortproblems

Kann die Frage $w \in_? L(G)$? für *jede beliebige* Grammatik G und *jedes beliebige* Wort $w \in T^*$ entschieden (d.h. entweder mit JA oder NEIN beantwortet) werden? Nein!

Satz

Das Wortproblem ist für Chomsky-Typ-0-Sprachen nicht allgemein entscheidbar. Für alle anderen Typen (1,2,3) ist dessen allgemeine Entscheidbarkeit gesichert.

Entscheidbarkeit des Wortproblems

Kann die Frage $w \in_? L(G)$? für *jede beliebige* Grammatik G und *jedes beliebige* Wort $w \in T^*$ entschieden (d.h. entweder mit JA oder NEIN beantwortet) werden? Nein!

Satz

Das Wortproblem ist für Chomsky-Typ-0-Sprachen nicht allgemein entscheidbar. Für alle anderen Typen (1,2,3) ist dessen allgemeine Entscheidbarkeit gesichert.

Entscheidbarkeit des Wortproblems

Kann die Frage $w \in_? L(G)$? für *jede beliebige* Grammatik G und *jedes beliebige* Wort $w \in T^*$ entschieden (d.h. entweder mit JA oder NEIN beantwortet) werden? Nein!

Satz

Das Wortproblem ist für Chomsky-Typ-0-Sprachen nicht allgemein entscheidbar. Für alle anderen Typen (1,2,3) ist dessen allgemeine Entscheidbarkeit gesichert.

Entscheidbarkeit des Wortproblems

Kann die Frage $w \in_? L(G)$? für *jede beliebige* Grammatik G und *jedes beliebige* Wort $w \in T^*$ entschieden (d.h. entweder mit JA oder NEIN beantwortet) werden? Nein!

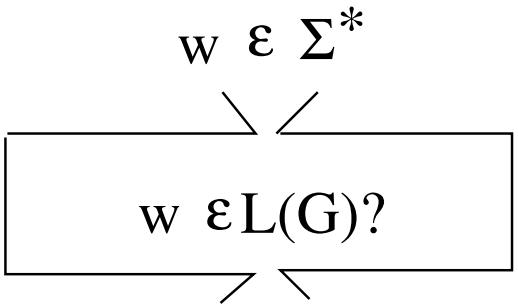
Satz

Das Wortproblem ist für Chomsky-Typ-0-Sprachen nicht allgemein entscheidbar. Für alle anderen Typen (1,2,3) ist dessen allgemeine Entscheidbarkeit gesichert.

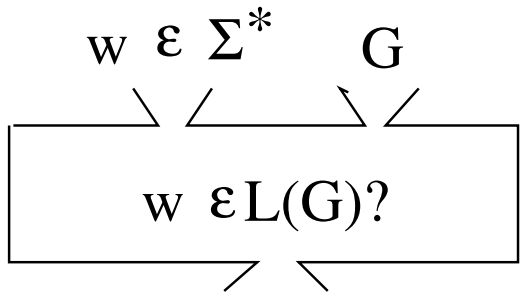
Beweis.

Typ-0-Sprachen fehlt im Allg. die Längenmonotonie, so dass kein stets terminierendes Prüfverfahren angegeben werden kann.
Für Typ-1-Sprachen: Simultanableitung + längenmonotone Regeln
$\rightarrow$ Abbruch, wenn $s_i = w$ (Erfolg) bzw. $\mid s_i \mid > \mid w \mid$. □

Konsequenzen für die Praxis



Parser für eine konkrete PS



z.B. Parser für eine durch DTD/Scheme def. XML–Sprache

Parser für ksS mgl., aber praktisch unbrauchbar

Effiziente Parser (kubischer bzw. linearer Aufwand) gibt es für bestimmte Untermengen kfS, z.B.: LR(k)-, LALR(k)-, LL(k)-Sprachen.

Reguläre Ausdrücke

Komprimierteste Darstellungsform für Typ-3-Sprachen – Muster

Beispiele: $a^*(b+c)d^+$, $[0-9]^*.[0-9]^*$; in Editoren: $test?.doc$, $*.bat$, $[abc]*.*$, $[\hat{}adt]*.*$

Verschiedendste Notationsformen $\rightarrow$ *Definition* regulärer Ausdrücke notwendig

Reguläre Ausdrücke

Komprimierteste Darstellungsform für Typ-3-Sprachen – Muster

Beispiele: $a^*(b+c)d^+$, $[0-9]^*.[0-9]^*$; in Editoren: $test?.doc$, $*.bat$, $[abc]*.*$, $[\hat{}adt]*.*$

Verschiedendste Notationsformen $\rightarrow$ *Definition* regulärer Ausdrücke notwendig

Reguläre Ausdrücke

Komprimierteste Darstellungsform für Typ-3-Sprachen – Muster

Beispiele: $a^*(b+c)d^+$, $[0-9]^*.[0-9]^*$; in Editoren: $test?.doc$, $*.bat$, $[abc]*.*$, $[\hat{}adt]*.*$

Verschiedendste Notationsformen $\rightarrow$ *Definition* regulärer Ausdrücke notwendig

Definition regulärer Ausdrücke (rA)

Definition

1. $\emptyset$ ist ein rA und es gilt $L(\emptyset) = \emptyset$.
2. ε ist ein rA und es gilt $L(\varepsilon) = \{\varepsilon\}$.
3. Für jedes Zeichen $a \in A$ ist a ein rA und es gilt $L(a) = \{a\}$.
4. Sind α und β rA so auch
   $(\alpha + \beta)$, und es gilt $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$.
   $(\alpha \cdot \beta)$ und es gilt $L(\alpha \cdot \beta) = L(\alpha) \circ L(\beta)$.
   (α^*) und es gilt $L(\alpha^*) = L(\alpha)^*$.

Klammersparregeln: Prioritäten der Operationen gemäß Zahlenrechnen

regulärer Ausdruck $\rightleftarrows$ reguläre Sprache (Mengen)

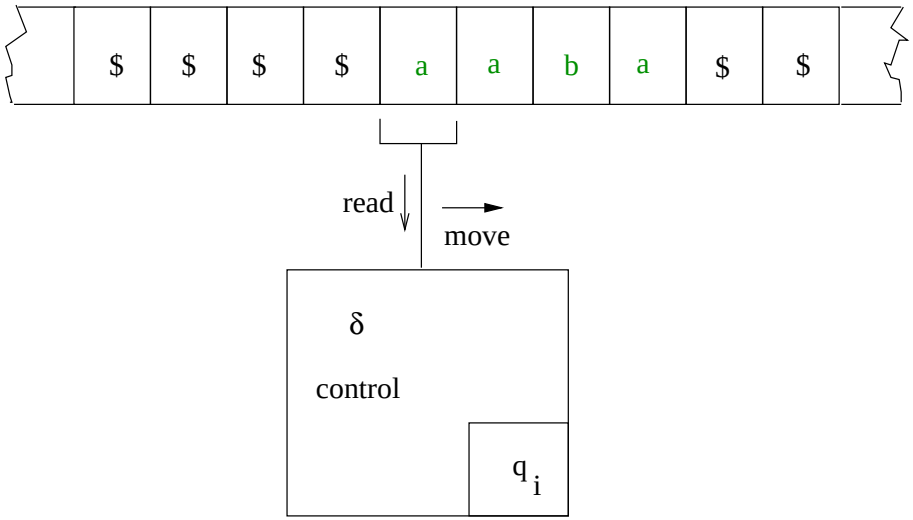
$$
\begin{aligned}
L(a+bc) &= L(a) \cup L(bc) \\
&= L(a) \cup L(b) \circ L(c) \\
&= \{a\} \cup \{b\} \circ \{c\} \\
&= \{a\} \cup \{bc\} \\
&= \{a, bc\}
\end{aligned}
$$

$$
\begin{aligned}
L((a+b)c) &= \{ac, bc\} \\
L(a^*b) &= \{b, ab, a^2b, a^3b, \ldots\}
\end{aligned}
$$

Aufgabe: Geben Sie eine reguläre Grammatik für a^*b an.

Aufbau und Arbeitsweise

Endlicher Automat (*abstrakter* Automat - Bitte nicht nachbauen!)



Start: in q_0

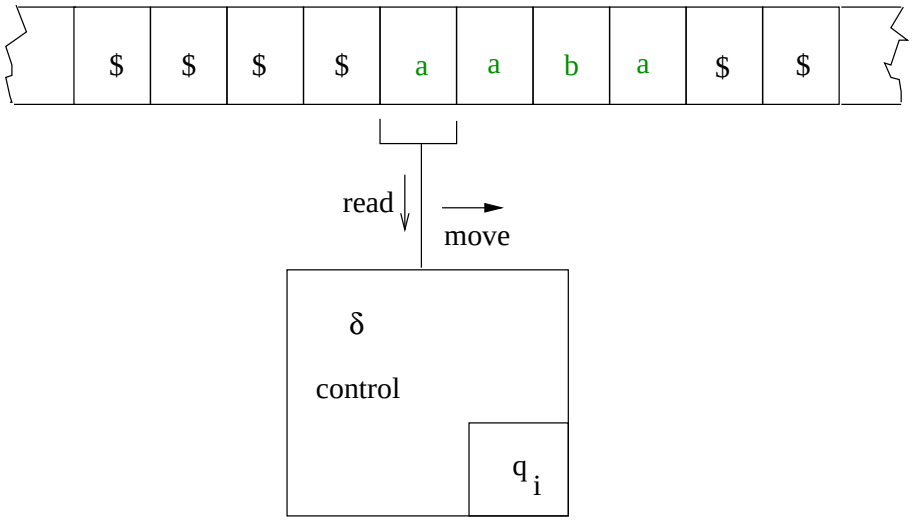
Stopp: in q_i, wenn Eingabewort vollständig gescannt

Arbeitstakt: read, Folgezustand, move

$q_i \in E$: Wort wird akzeptiert
$q_i \notin E$: Wort wird nicht akzeptiert

Aufbau und Arbeitsweise

Endlicher Automat (*abstrakter* Automat - Bitte nicht nachbauen!)



Start: in q_0

Stopp: in q_i, wenn Eingabewort vollständig gescannt

Arbeitstakt: read, Folgezustand, move

$q_i \in E$: Wort wird akzeptiert
$q_i \notin E$: Wort wird nicht akzeptiert

Überführungsfunktion

Totale Funktion δ, mit $\delta(q_i, a) = q_j$

δ	a_0	a_1	$\ldots$	a_n
q_0	$\ldots$	q_r	$\ldots$	
q_1	$\ldots$			
$\vdots$	$\vdots$	$\ldots$		
q_m	$\ldots$			

Jedes Tabellenfeld enthält genau einen Eintrag (Zustand).

Überführungsfunktion

Totale Funktion δ, mit $\delta(q_i, a) = q_j$

δ	a_0	a_1	$\ldots$	a_n
q_0	$\ldots$	q_r	$\ldots$	
q_1	$\ldots$			
$\vdots$	$\vdots$	$\ldots$		
q_m	$\ldots$			

Jedes Tabellenfeld enthält genau einen Eintrag (Zustand).

Überführungsfunktion

Totale Funktion δ, mit $\delta(q_i, a) = q_j$

δ	a_0	a_1	$\ldots$	a_n
q_0	$\ldots$	q_r	$\ldots$	
q_1	$\ldots$			
$\vdots$	$\vdots$	$\ldots$		
q_m	$\ldots$			

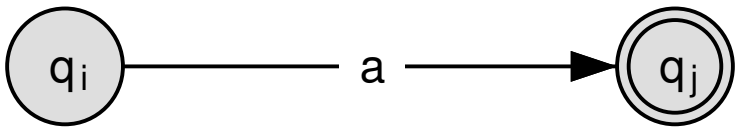
Jedes Tabellenfeld enthält genau einen Eintrag (Zustand).

Überführungsfunktion

Totale Funktion δ, mit $\delta(q_i, a) = q_j$

δ	a_0	a_1	$\ldots$	a_n
q_0	$\ldots$	q_r	$\ldots$	
q_1	$\ldots$			
$\vdots$	$\vdots$	$\ldots$		
q_m	$\ldots$			

Jedes Tabellenfeld enthält genau einen Eintrag (Zustand).



$q_j \in E$

Definition

Definition

Ein EA ist ein Quintupel $M = (Q, \Sigma, \delta, q_0, E)$, mit

- Q ... endliche Menge der Zustände,
- Σ ... Eingabealphabet, $Q \cap \Sigma = \emptyset$,
- δ ... Überführungsfunktion (totale Funktion!), $Q \times \Sigma \longrightarrow Q$,
- q_0 ... Startzustand, $q_0 \in Q$, und
- E ... endliche, nichtleere Menge der Endzustände ($E \subseteq Q$).

Definition

Definition

Ein EA ist ein Quintupel $M = (Q, \Sigma, \delta, q_0, E)$, mit

- Q ... endliche Menge der Zustände,
- Σ ... Eingabealphabet, $Q \cap \Sigma = \emptyset$,
- δ ... Überführungsfunktion (totale Funktion!), $Q \times \Sigma \to Q$,
- q_0 ... Startzustand, $q_0 \in Q$, und
- E ... endliche, nichtleere Menge der Endzustände ($E \subseteq Q$).

Definition

Definition

Ein EA ist ein Quintupel $M = (Q, \Sigma, \delta, q_0, E)$, mit

- Q ... endliche Menge der Zustände,
- Σ ... Eingabealphabet, $Q \cap \Sigma = \emptyset$,
- δ ... Überführungsfunktion (totale Funktion!), $Q \times \Sigma \rightarrow Q$,
- q_0 ... Startzustand, $q_0 \in Q$, und
- E ... endliche, nichtleere Menge der Endzustände ($E \subseteq Q$).

Definition

Definition

Ein EA ist ein Quintupel $M = (Q, \Sigma, \delta, q_0, E)$, mit

- Q ... endliche Menge der Zustände,
- Σ ... Eingabealphabet, $Q \cap \Sigma = \emptyset$,
- δ ... Überführungsfunktion (totale Funktion!), $Q \times \Sigma \rightarrow Q$,
- q_0 ... Startzustand, $q_0 \in Q$, und
- E ... endliche, nichtleere Menge der Endzustände ($E \subseteq Q$).

Definition

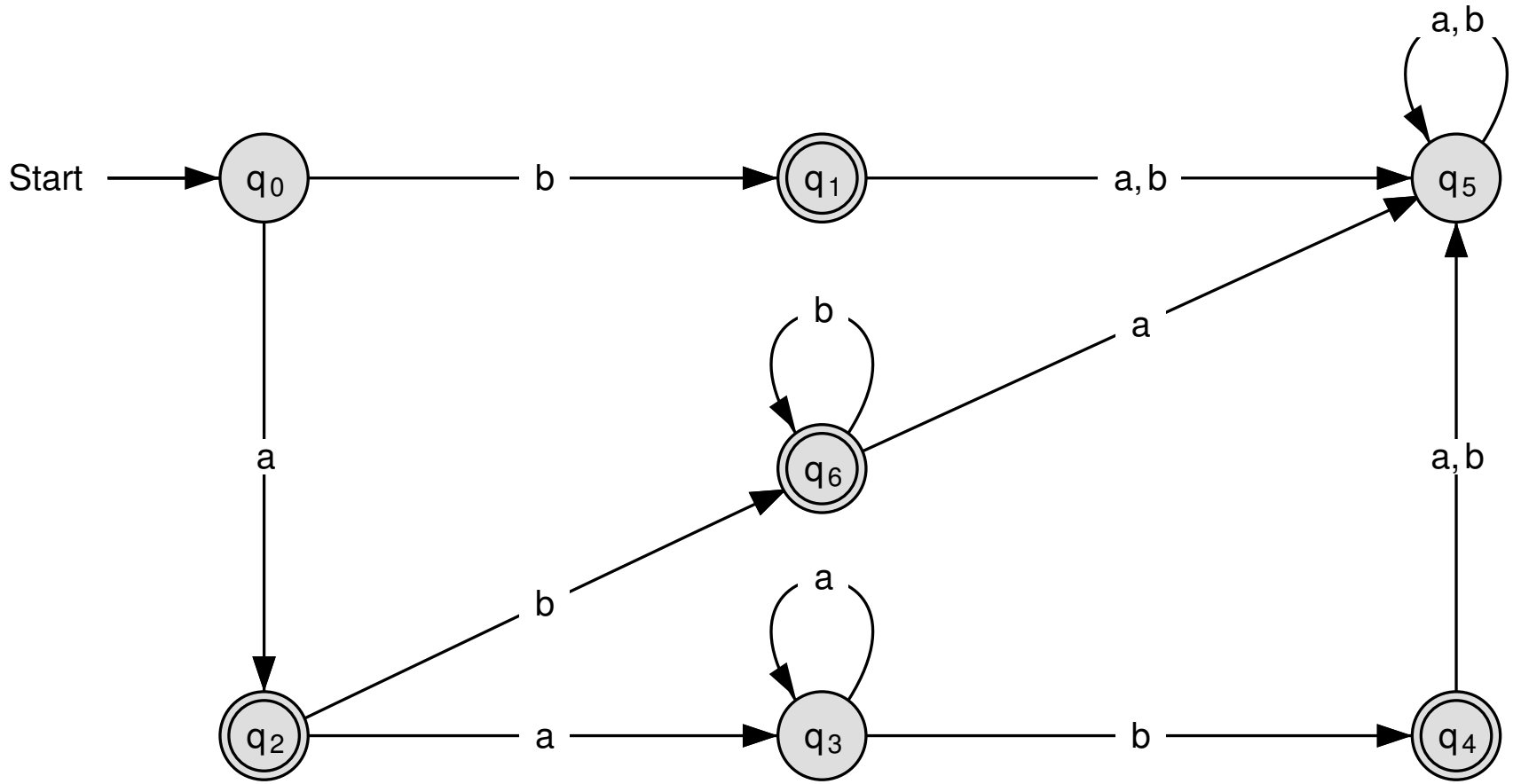
Definition

Ein EA ist ein Quintupel $M = (Q, \Sigma, \delta, q_0, E)$, mit

- Q ... endliche Menge der Zustände,
- Σ ... Eingabealphabet, $Q \cap \Sigma = \emptyset$,
- δ ... Überführungsfunktion (totale Funktion!), $Q \times \Sigma \to Q$,
- q_0 ... Startzustand, $q_0 \in Q$, und
- E ... endliche, nichtleere Menge der Endzustände ($E \subseteq Q$).

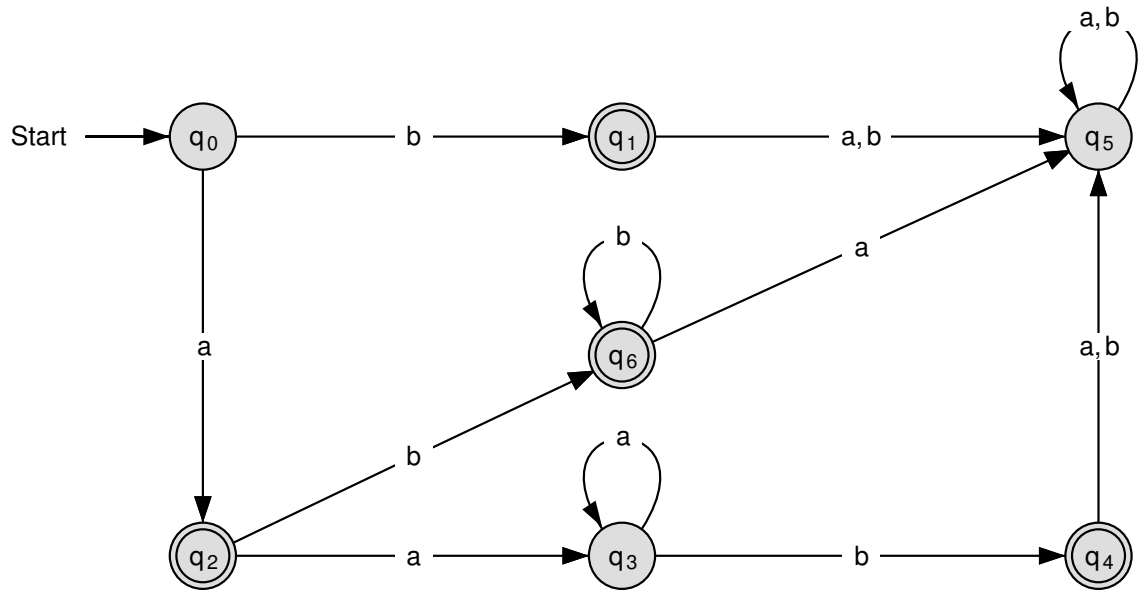
Beispiel

$$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{a, b\}, \delta, q_0, \{q_1, q_2, q_4, q_6\}))$$



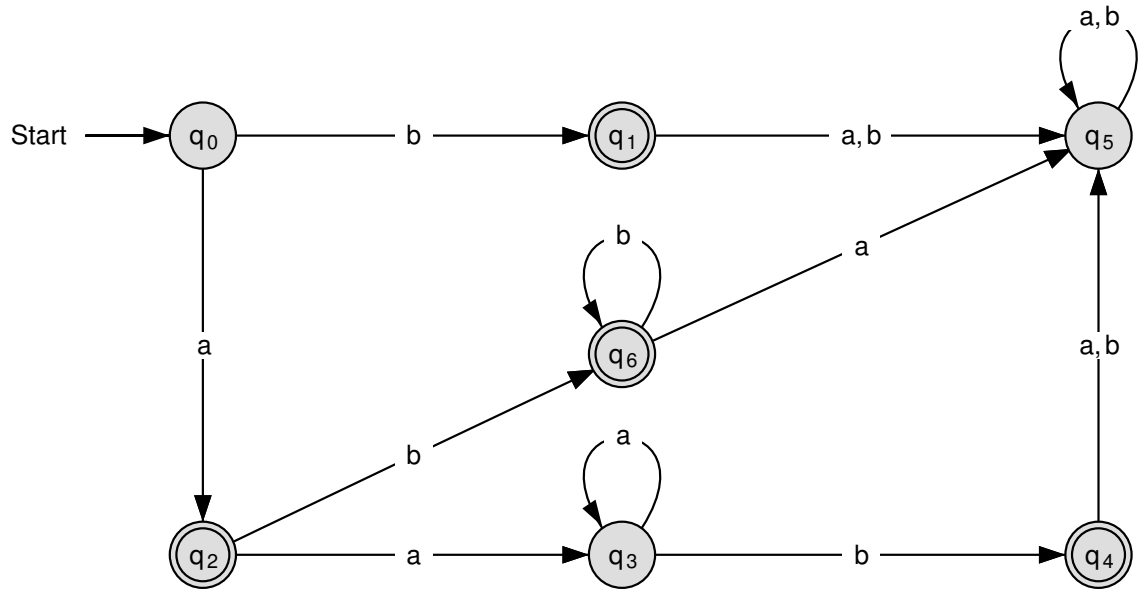
Aufgabe

Aufgabe: Geben Sie δ als Tabelle an.



Aufgabe

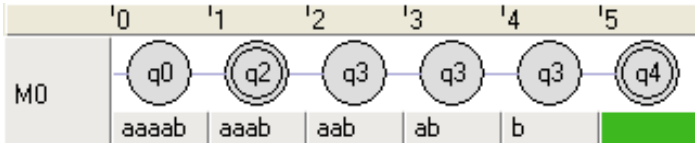
Aufgabe: Geben Sie δ als Tabelle an.



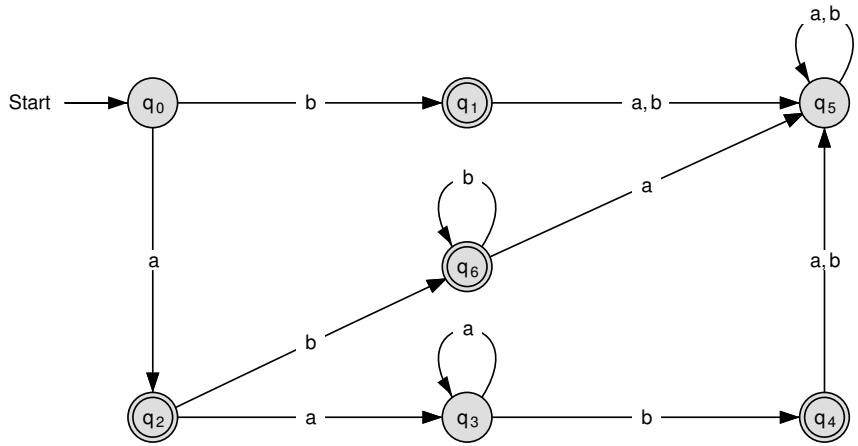
Lösung

δ	a	b
q_0	q_2	q_1
q_1	q_5	q_5
q_2	q_3	q_6
q_3	q_3	q_4
q_4	q_5	q_5
q_5	q_5	q_5
q_6	q_5	q_6

Simulation von M für aaaab

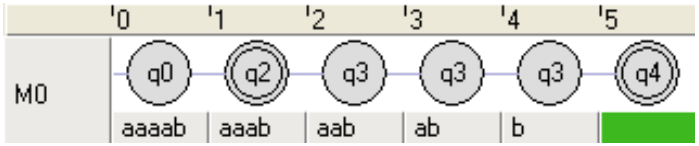


Zustand	Restwort
q_0	$aaaab$
q_2	$aaab$
q_3	aab
q_3	ab
q_3	b
q_4	ε

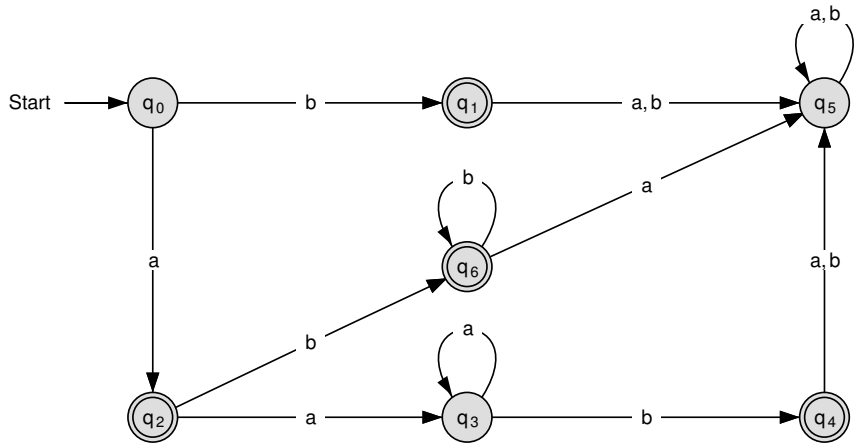


Welche Sprache akzeptiert der Automat?
$L(M) =$

Simulation von M für aaaab

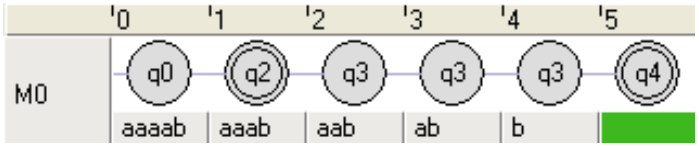


Zustand	Restwort
q_0	*aaaab*
q_2	*aaab*
q_3	*aab*
q_3	*ab*
q_3	*b*
q_4	ε

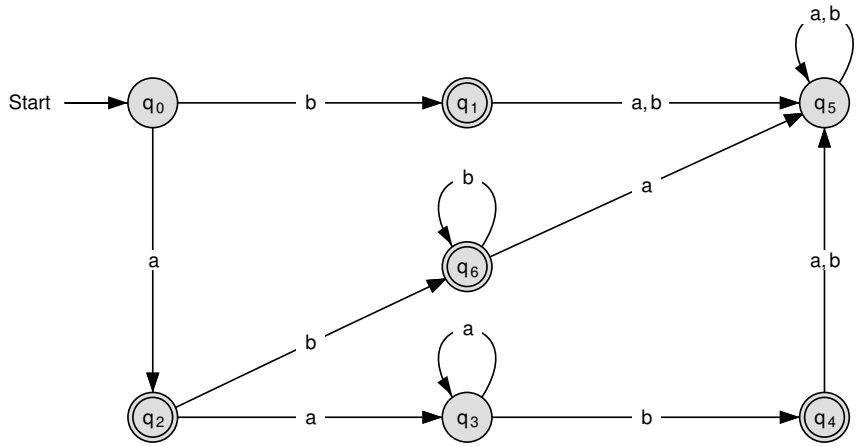


Welche Sprache akzeptiert der Automat?
$L(M) =$

Simulation von M für aaaab

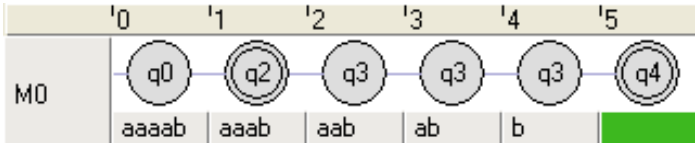


Zustand	Restwort
q_0	*aaaab*
q_2	*aaab*
q_3	*aab*
q_3	*ab*
q_3	*b*
q_4	ε

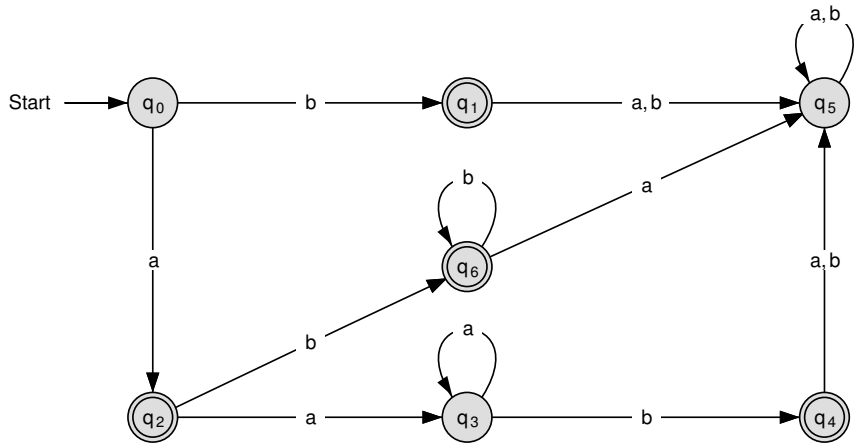


Welche Sprache akzeptiert der Automat?
$L(M) =$

Simulation von M für aaaab

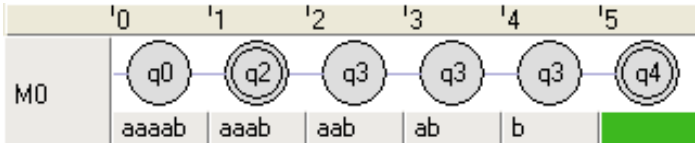


Zustand	Restwort
q_0	*aaaab*
q_2	*aaab*
q_3	*aab*
q_3	*ab*
q_3	*b*
q_4	ε

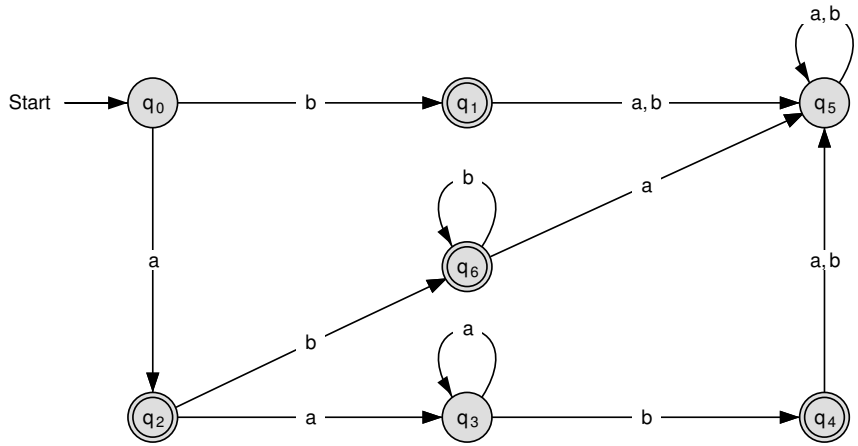


Welche Sprache akzeptiert der Automat?
$L(M) =$

Simulation von M für aaaab

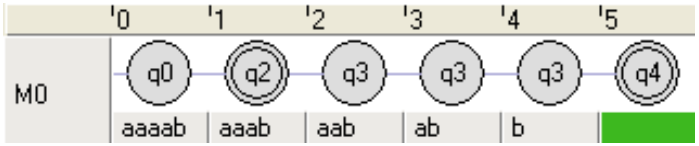


Zustand	Restwort
q_0	$aaaab$
q_2	$aaab$
q_3	aab
q_3	ab
q_3	b
q_4	ε

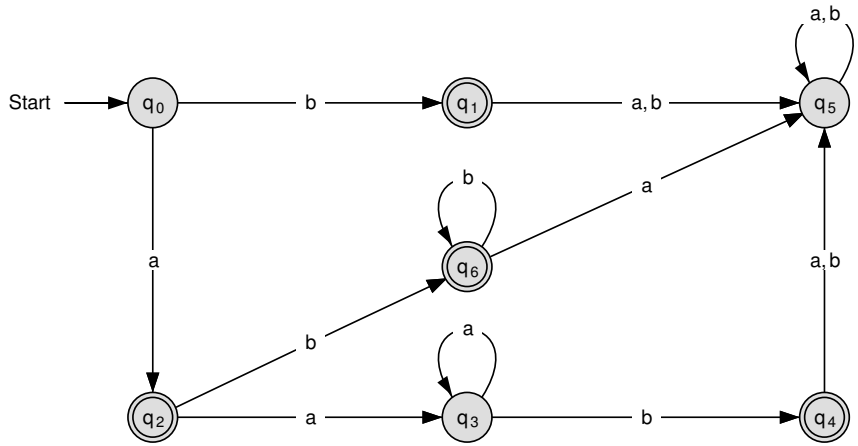


Welche Sprache akzeptiert der Automat?
$L(M) =$

Simulation von M für aaaab

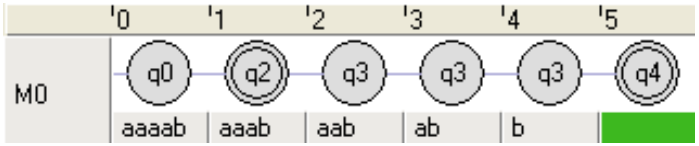


Zustand	Restwort
q_0	$aaaab$
q_2	$aaab$
q_3	aab
q_3	ab
q_3	b
q_4	ε

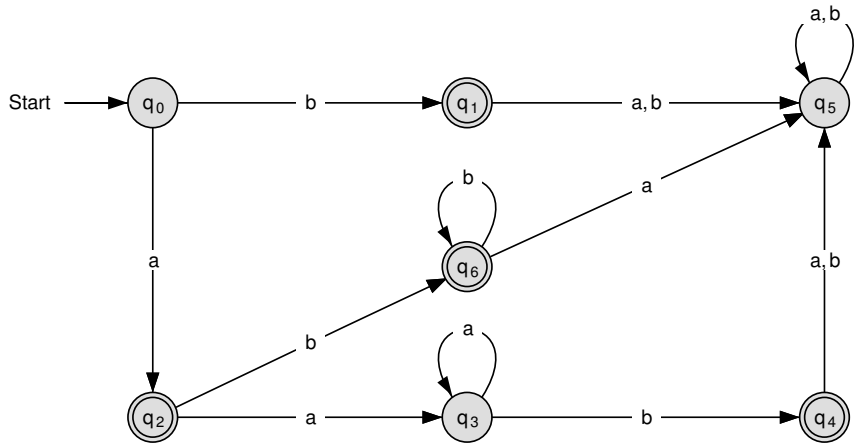


Welche Sprache akzeptiert der Automat?
$L(M) =$

Simulation von M für aaaab

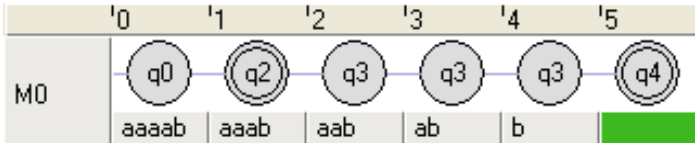


Zustand	Restwort
q_0	$aaaab$
q_2	$aaab$
q_3	aab
q_3	ab
q_3	b
q_4	ε

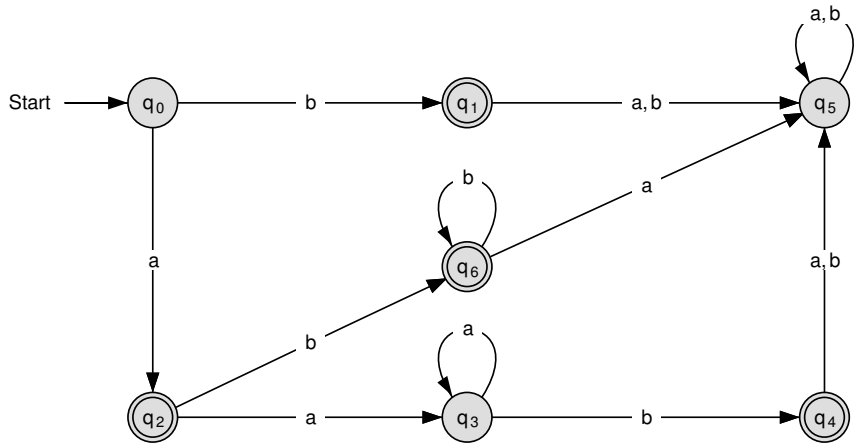


Welche Sprache akzeptiert der Automat?
$L(M) =$

Simulation von M für aaaab

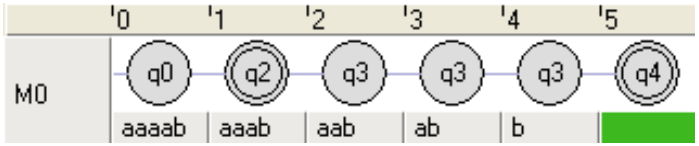


Zustand	Restwort
q_0	$aaaab$
q_2	$aaab$
q_3	aab
q_3	ab
q_3	b
q_4	ε

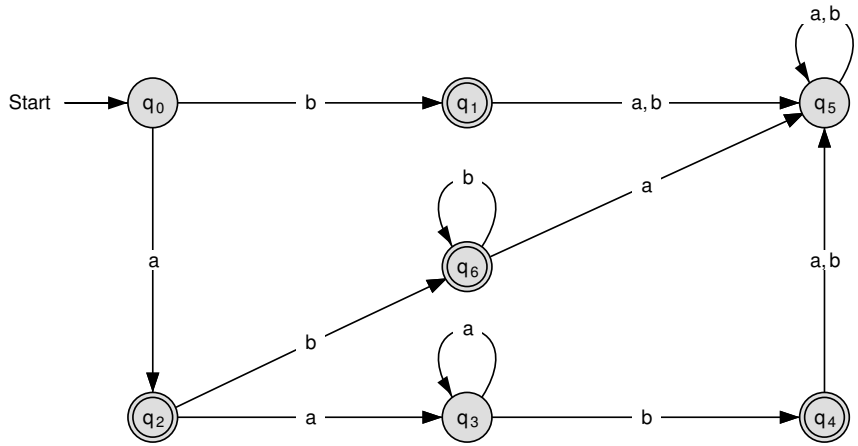


Welche Sprache akzeptiert der Automat?
$L(M) =$

Simulation von M für aaaab



Zustand	Restwort
q_0	$aaaab$
q_2	$aaab$
q_3	aab
q_3	ab
q_3	b
q_4	ε



Welche Sprache akzeptiert der Automat?
$L(M) = a^*b + ab^*$

Konfigurationenfolge und Sprache $L(M)$

Konfigurationenfolge: $[q_{i_1}, r_{k_1}] \vdash [q_{i_2}, r_{k_2}] \vdash [q_{i_3}, r_{k_3}] \vdash \ldots \vdash [q_{i_n}, r_{k_n}]$

Die von einem EA M akzeptierte (erkannte, beschriebene, definierte) Sprache $L(M)$:

$$L(M) = \{w \mid w \in \Sigma^* \text{ und } [q_0, w] \vdash^* [q_e, \varepsilon] \text{ und } q_e \in E\}$$

Konfigurationenfolge und Sprache $L(M)$

Konfigurationenfolge: $[q_{i_1}, r_{k_1}] \vdash [q_{i_2}, r_{k_2}] \vdash [q_{i_3}, r_{k_3}] \vdash \ldots \vdash [q_{i_n}, r_{k_n}]$

Die von einem EA M akzeptierte (erkannte, beschriebene, definierte) Sprache $L(M)$:

$$L(M) = \{w \mid w \in \Sigma^* \text{ und } [q_0, w] \vdash^* [q_e, \varepsilon] \text{ und } q_e \in E\}$$

Umwandlungen: EA $\rightleftarrows$ reguläre Grammatik

Satz

Jede von einem EA akzeptierte Sprache ist regulär.

Beweis.

Teil 1: EA $\longrightarrow$ G
Teil 2: G $\longrightarrow$ EA

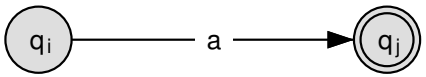
Umwandlung: EA $\longrightarrow$ reguläre Grammatik

Beweis.

Teil 1: EA $\longrightarrow$ G

Konstruiere reguläre Grammatik $G = (N, T, P, s)$ aus EA $M = (Q, \Sigma, \delta, q_0, E)$: $N = Q$, $T = \Sigma$, $s = q_0$.

P ergibt sich aus δ und E:



entspricht $q_i \longrightarrow aq_j$

und zusätzlich $q_i \longrightarrow a$, da $q_j \in E$. □

Aufgabe

Aufgabe: Wenden Sie das Prinzip zur Konstruktion einer äquivalenten Grammatik aus einem gegebenen EA auf M (aus obigem Beispiel) an.

Optimieren Sie anschließend die erzeugte Regelmenge.

Aufgabe

Aufgabe: Wenden Sie das Prinzip zur Konstruktion einer äquivalenten Grammatik aus einem gegebenen EA auf M (aus obigem Beispiel) an.

Optimieren Sie anschließend die erzeugte Regelmenge.

$$P = \{q_0 \rightarrow bq_1 \mid b \mid aq_2 \mid a,\ q_2 \rightarrow bq_6 \mid b \mid aq_3,\ q_3 \rightarrow aq_3 \mid bq_4 \mid b,\ q_6 \rightarrow bq_6 \mid b,\}$$

Motivation für die Einführung von NEAs

- entstehen bei dem Versuch aus einer rG einen äquivalenten EA zu konstruieren
- verwenden das Konzept des Nichtdeterminismus (s. Arbeitsweise)
- einfachere Überführungsgraphen gegenüber DEA
- Allerdings: gleiche Mächtigkeit wie DEA (kann man beweisen)

Motivation für die Einführung von NEAs

- entstehen bei dem Versuch aus einer rG einen äquivalenten EA zu konstruieren
- verwenden das Konzept des Nichtdeterminismus (s. Arbeitsweise)
- einfachere Überführungsgraphen gegenüber DEA
- Allerdings: gleiche Mächtigkeit wie DEA (kann man beweisen)

Motivation für die Einführung von NEAs

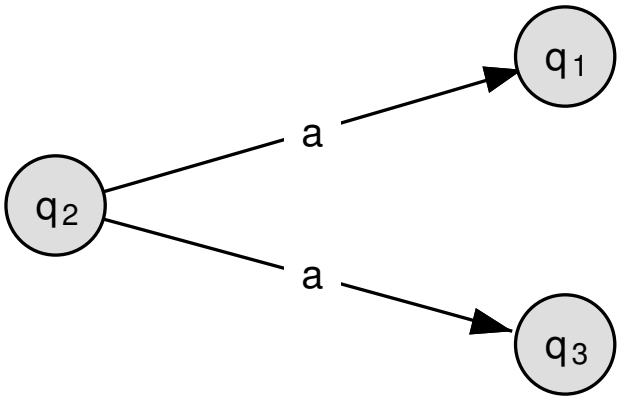
- entstehen bei dem Versuch aus einer rG einen äquivalenten EA zu konstruieren
- verwenden das Konzept des Nichtdeterminismus (s. Arbeitsweise)
- einfachere Überführungsgraphen gegenüber DEA
- Allerdings: gleiche Mächtigkeit wie DEA (kann man beweisen)

Motivation für die Einführung von NEAs

- entstehen bei dem Versuch aus einer rG einen äquivalenten EA zu konstruieren
- verwenden das Konzept des Nichtdeterminismus (s. Arbeitsweise)
- einfachere Überführungsgraphen gegenüber DEA
- Allerdings: gleiche Mächtigkeit wie DEA (kann man beweisen)

Konstruktionsversuch $G \longrightarrow DEA$ führt zum NEA

$q_2 \longrightarrow aq_1 \mid aq_3 \mid \ldots$ führen strikt zu



Überführungsfunktion

Totale Funktion δ, mit
$\delta(q_i, a) = \{q_j, q_k, \ldots, q_r\},\ q_j, q_k, \ldots, q_r \in Q$

δ	a_0	a_1	$\ldots$	a_n
q_0	$\ldots$	$\{q_r\}$	$\ldots$	
q_1	$\ldots$	$\{q_r, q_k\}$		
$\vdots$	$\vdots$	$\ldots$		
q_m	$\ldots$		$\emptyset$	

Jedes Tabellenfeld enthält genau einen Eintrag: eine Menge von Zuständen.

Überführungsfunktion

Totale Funktion δ, mit
$\delta(q_i, a) = \{q_j, q_k, \ldots, q_r\},\ q_j, q_k, \ldots, q_r \in Q$

δ	a_0	a_1	$\ldots$	a_n
q_0	$\ldots$	$\{q_r\}$	$\ldots$	
q_1	$\ldots$	$\{q_r, q_k\}$		
$\vdots$	$\vdots$	$\ldots$		
q_m	$\ldots$		$\emptyset$	

Jedes Tabellenfeld enthält genau einen Eintrag: eine Menge von Zuständen.

Überführungsfunktion

Totale Funktion δ, mit
$\delta(q_i, a) = \{q_j, q_k, \ldots, q_r\},\ q_j, q_k, \ldots, q_r \in Q$

δ	a_0	a_1	$\ldots$	a_n
q_0	$\ldots$	$\{q_r\}$	$\ldots$	
q_1	$\ldots$	$\{q_r, q_k\}$		
$\vdots$	$\vdots$	$\ldots$		
q_m	$\ldots$		$\emptyset$	

Jedes Tabellenfeld enthält genau einen Eintrag: eine Menge von Zuständen.

Aufbau und Arbeitsweise

Aufbau/Definition: wie DEA, aber Unterschiede in δ

Arbeitsweise: falls für a mehrere Folgezustände, dann Automat klonen und parallele Fortsetzung

Start: in q_0

Stopp und Akzeptanz: Eingabewort von einem geklonten Automaten vollständig gescannt und in einem Endzustand

Stopp und Nichtakzeptanz: Eingabewort von allen geklonten Automaten vollständig gescannt und keiner in Endzustand

Aufbau und Arbeitsweise

Aufbau/Definition: wie DEA, aber Unterschiede in δ

Arbeitsweise: falls für a mehrere Folgezustände, dann Automat klonen und parallele Fortsetzung

Start: in q_0

Stopp und Akzeptanz: Eingabewort von einem geklonten Automaten vollständig gescannt und in einem Endzustand

Stopp und Nichtakzeptanz: Eingabewort von allen geklonten Automaten vollständig gescannt und keiner in Endzustand

Aufbau und Arbeitsweise

Aufbau/Definition: wie DEA, aber Unterschiede in δ

Arbeitsweise: falls für a mehrere Folgezustände, dann Automat klonen und parallele Fortsetzung

Start: in q_0

Stopp und Akzeptanz: Eingabewort von einem geklonten Automaten vollständig gescannt und in einem Endzustand

Stopp und Nichtakzeptanz: Eingabewort von allen geklonten Automaten vollständig gescannt und keiner in Endzustand

Aufbau und Arbeitsweise

Aufbau/Definition: wie DEA, aber Unterschiede in δ

Arbeitsweise: falls für a mehrere Folgezustände, dann Automat klonen und parallele Fortsetzung

Start: in q_0

Stopp und Akzeptanz: Eingabewort von einem geklonten Automaten vollständig gescannt und in einem Endzustand

Stopp und Nichtakzeptanz: Eingabewort von allen geklonten Automaten vollständig gescannt und keiner in Endzustand

Aufbau und Arbeitsweise

Aufbau/Definition: wie DEA, aber Unterschiede in δ

Arbeitsweise: falls für a mehrere Folgezustände, dann Automat klonen und parallele Fortsetzung

Start: in q_0

Stopp und Akzeptanz: Eingabewort von einem geklonten Automaten vollständig gescannt und in einem Endzustand

Stopp und Nichtakzeptanz: Eingabewort von allen geklonten Automaten vollständig gescannt und keiner in Endzustand

Konfigurationenfolge und Sprache $L(M)$

Die von einem EA M akzeptierte (erkannte, beschriebene, definierte) Sprache $L(M)$:
$L(M) = \{w \mid w \in \Sigma^* \text{ und } [q_0, w] \vdash^* [q_e, \varepsilon] \text{ und } q_e \in E\}$ (mind. ein Erfolgsautomat)

$L(M) = \{w \mid w \in \Sigma^* \text{ und } \hat{\delta}(\{q_0\}, w) \cap E \neq \emptyset\}$.
$\hat{\delta} : \wp(Q) \times \Sigma^* \to \wp(Q)$, mit

$$\begin{aligned} \hat{\delta}(Q', \varepsilon) &= Q', \text{ für alle } Q' \subseteq Q \\ \hat{\delta}(Q', aw) &= \bigcup_{q \in Q'} (\hat{\delta}(\delta(q, a), w)) \end{aligned}$$

Konfigurationenfolge und Sprache $L(M)$

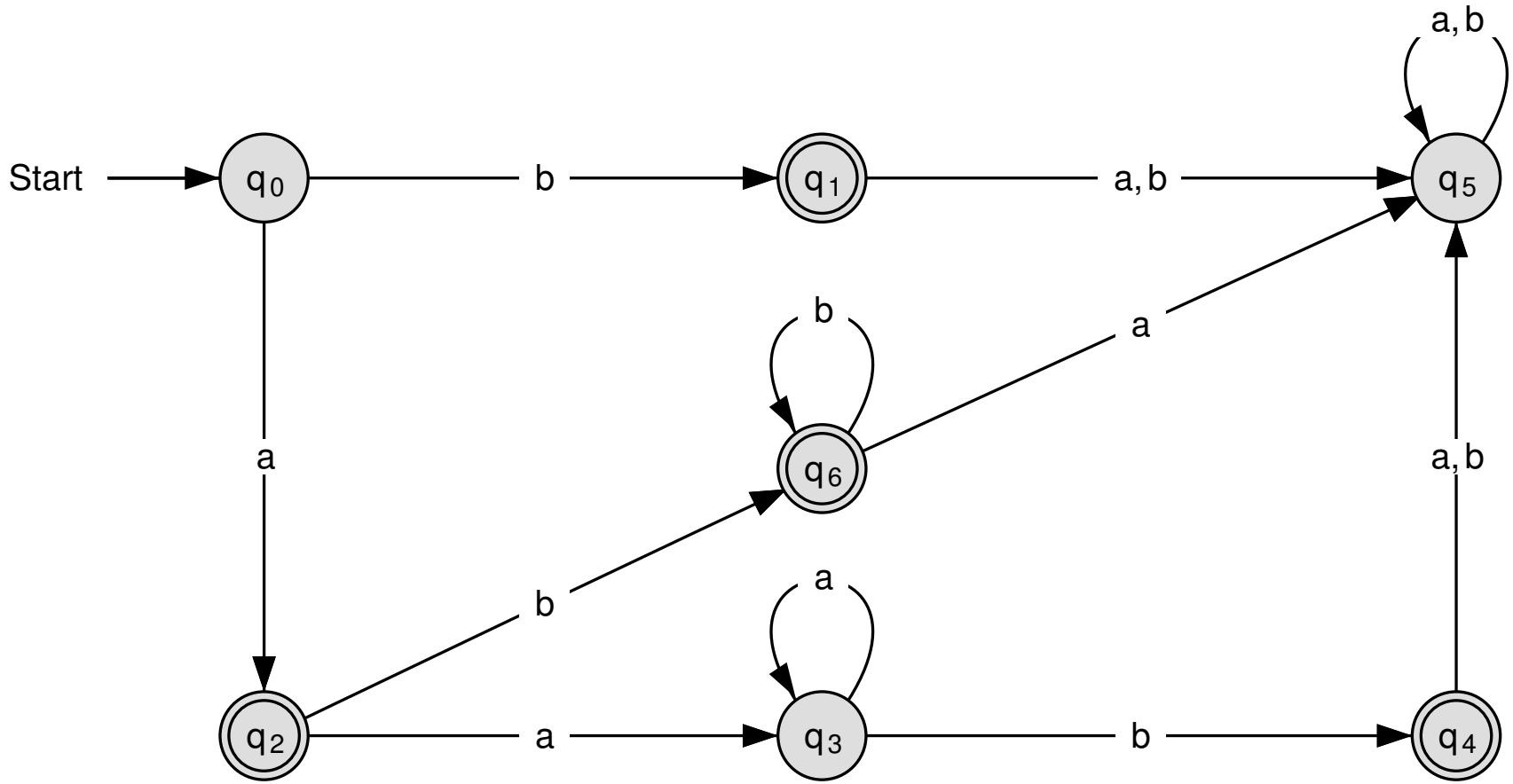
Die von einem EA M akzeptierte (erkannte, beschriebene, definierte) Sprache $L(M)$:
$L(M) = \{w \mid w \in \Sigma^* \text{ und } [q_0, w] \vdash^* [q_e, \varepsilon] \text{ und } q_e \in E\}$ (mind. ein Erfolgsautomat)

$L(M) = \{w \mid w \in \Sigma^* \text{ und } \hat{\delta}(\{q_0\}, w) \cap E \neq \emptyset\}$.
$\hat{\delta} : \wp(Q) \times \Sigma^* \to \wp(Q)$, mit

$$\begin{aligned} \hat{\delta}(Q', \varepsilon) &= Q', \text{ für alle } Q' \subseteq Q \\ \hat{\delta}(Q', aw) &= \bigcup_{q \in Q'} (\hat{\delta}(\delta(q, a), w)) \end{aligned}$$

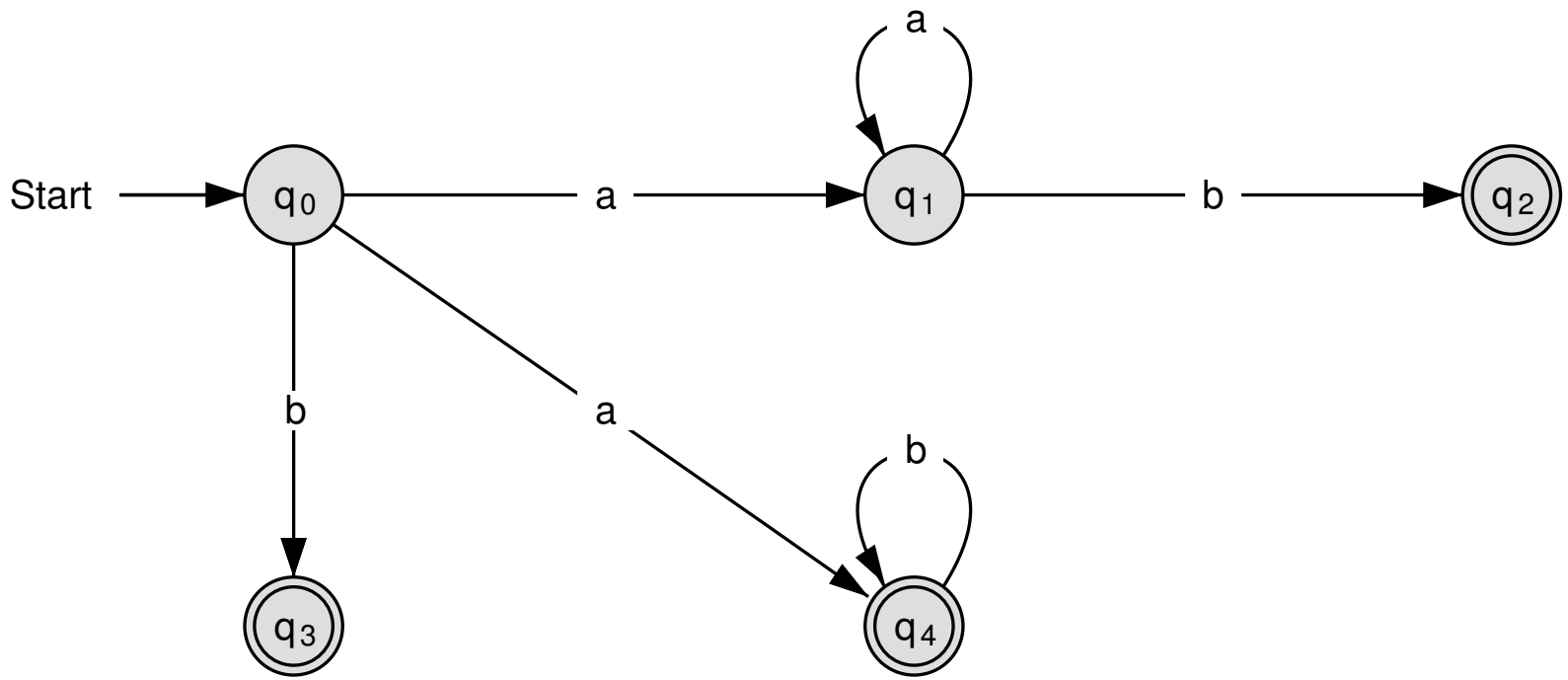
Beispiel: DEA für $a^*b + ab^*$

$$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{a, b\}, \delta, q_0, \{q_1, q_2, q_4, q_6\}))$$



Beispiel: NEA für $a^*b + ab^*$

$$M = (\{q_0, q_1, q_3, q_4, q_2\}, \{a, b\}, \delta, q_0, \{q_3, q_4, q_2\}))$$



Fazit

Typ-3-Grammatik, DEA und NEA sind äquivalente Beschreibungsmittel für reguläre Sprachen.

Aufgaben für die CÜ

mit AutoEdit (Teil von AtoCC), s. Aufgabensammlung