Übersichtsfolien zur Übung

Theoretische Informatik Sommersemester 2014

Markus Kaiser

16. Mai 2014

Organisatorisches

- Markus Kaiser
  - Mail: tutor@zfix.org
  - Web: theo.zfix.org
  - Hg: tutor.zfix.org
- Meine Übungen
  - Gruppe 03: Montag, 16:05 - 17:35, 03.11.018
  - Gruppe 15: Donnerstag, 10:15 - 11:45, 00.08.038
- Hausaufgaben
  - Abgabe am Mittwoch, 10h
  - Rückgabe in der Übung
  - Kein Notenbonus
  - Trotzdem machen!
- Klausur
  - Endterm: Do 24.07. 11-14h
  - Wiederholung: Do 25.09. 11-14h

Was ist Theoinf?

Aus der VL

- Automatentheorie
  - Rechner mit endlichem oder kellerartigem Speicher
- Grammatiken
  - Syntax von Programmiersprachen
- Berechenbarkeitstheorie
  - Untersuchung der Grenzen, was Rechner prinzipiell können
- Komplexitätstheorie
  - Untersuchung der Grenzen, was Rechner mit begrenzten Ressourcen können

Alphabete

Definition

- Ein Alphabet Σ ist eine endliche Menge.
- Ein Wort über Σ ist eine endliche Folge von Zeichen.
- Eine Teilmenge $L \subseteq \Sigma^*$ ist eine formale Sprache

Definition (Operationen auf Sprachen)

- $AB := \{uv \mid u \in A \wedge v \in B\}$
- $A^{n+1} := A^n A, \qquad A^0 := \{\epsilon\}$
- $A^* := \bigcup_{n \in \mathbb{N}_0} A^n$

Definition (Grammatik)

Eine (Phrasenstruktur-)Grammatik $G = (V, \Sigma, P, S)$ ist ein 4-Tupel:

- V endlich viele Nichtterminale (Variablen)
- Σ ein Alphabet von Terminalen
- P endlich viele Produktionen $\subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$
- S ein Startsymbol (Axiom)

Ist $(l, r) \in P$, so schreibt man $l \to r$.

Beispiel

$\Sigma = \{0, 1\}$. Grammatik für alle Wörter ungerader Länge, bei denen alle Nullen vor der ersten Eins stehen und weniger Nullen als Einsen vorhanden sind.

Definition (Grammatik)

Eine (Phrasenstruktur-)Grammatik $G = (V, \Sigma, P, S)$ ist ein 4-Tupel:

- V endlich viele Nichtterminale (Variablen)
- Σ ein Alphabet von Terminalen
- P endlich viele Produktionen $\subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$
- S ein Startsymbol (Axiom)

Ist $(l, r) \in P$, so schreibt man $l \to r$.

Beispiel

$\Sigma = \{0, 1\}$. Grammatik für alle Wörter ungerader Länge, bei denen alle Nullen vor der ersten Eins stehen und weniger Nullen als Einsen vorhanden sind.

$$S \to 0S1 \mid S11 \mid 1$$

Sprachtypen

Sei $G = (V, \Sigma, P, S)$ eine Grammatik und $\alpha \to \beta \in P$ beliebig.

Definition (Monotonie)

G heißt (längen-)monoton, wenn für $\alpha \neq S$ gilt

$$|\alpha| \leq |\beta|$$

und falls $S \to \epsilon \in P$, dann kommt S nie auf der rechten Seite vor.

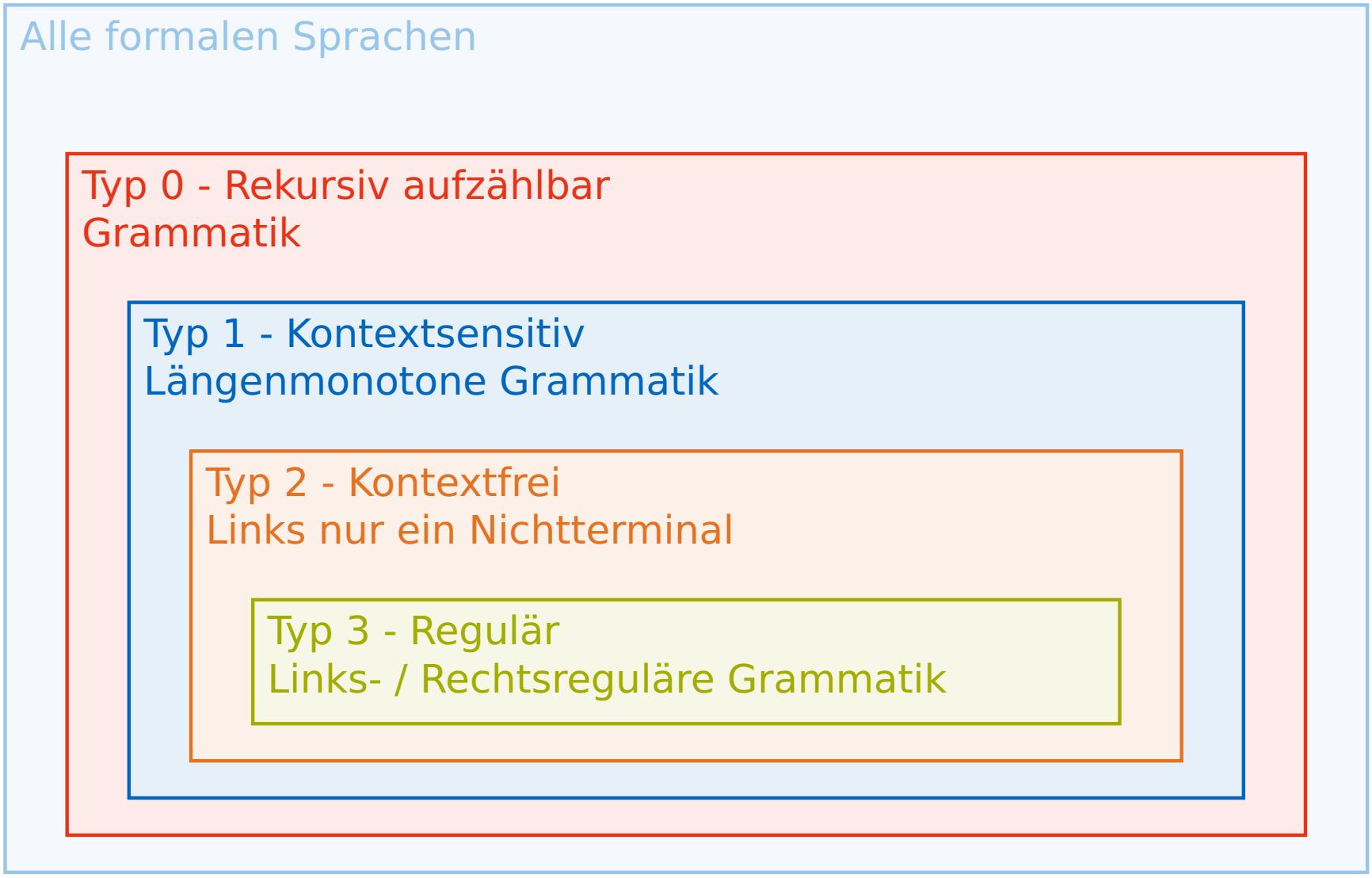
Definition (Chomsky-Typen)

Seien $A \in V$, $\gamma, \delta \in (V \cup \Sigma)^*$ und $\beta' \in (V \cup \Sigma)^+$.
Damit G vom Typ k ist, muss für α und β gelten

	α	β
Typ 0	beliebig	beliebig
Typ 1	$= \gamma A \delta$	$= \gamma \beta' \delta$
Typ 2	$\in V$	beliebig
Typ 3	$\in V$	$\in \Sigma^+ \cup \Sigma^* V$

Ab Typ 1 muss G auch monoton sein.

Chomsky-Hierarchie



Definition (Intuitive Berechenbarkeit)

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt intuitiv berechenbar, wenn es einen Algorithmus gibt, der bei Eingabe $(n_1, \ldots, n_k) \in \mathbb{N}^k$

- nach endlich vielen Schritten mit Ergebnis $f(n_1, \ldots, n_k)$ hält, falls $f(\ldots)$ definiert ist,
- und nicht terminiert, falls $f(\ldots)$ nicht definiert ist.

Churchsche These (nicht beweisbar)

Turing-Maschinen können genau alle intuitiv berechenbaren Funktionen berechnen.

Definition (Entscheidbarkeit)

Eine Menge A heißt entscheidbar gdw ihre charakteristische Funktion

$$\chi_A(x) := \begin{cases} 1 & \text{falls } x \in A \\ 0 & \text{falls } x \notin A \end{cases}$$

berechenbar ist.

Definition (Semi-Entscheidbarkeit)

Eine Menge A heißt semi-entscheidbar gdw

$$\chi'_A(x) := \begin{cases} 1 & \text{falls } x \in A \\ \bot & \text{falls } x \notin A \end{cases}$$

berechenbar ist.

Eindeutigkeit

Definition (kontextfreie Linksableitung)

Eine Ableitung

$$S \to^* xAz \to x\beta z \to^* w$$

heißt (kontextfreie) Linksableitung, wenn für jede Anwendung jeder Produktion $A \to \beta$ gilt, dass in x kein Nichtterminal vorkommt.

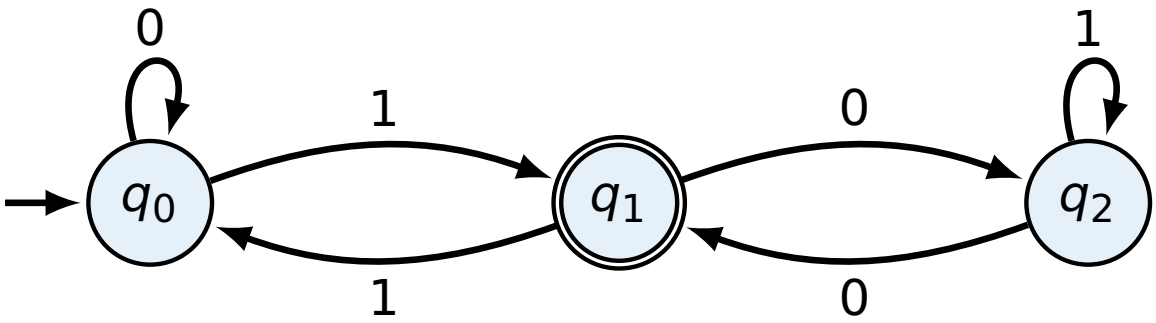
Definition (Eindeutigkeit)

- Eine Grammatik heißt eindeutig, wenn es für jedes Wort genau eine Linksableitung gibt.
- Eine Sprache heißt eindeutig, wenn es für sie eine eindeutige Grammatik gibt.

Definition (Deterministischer endlicher Automat)

Ein DFA ist ein Tupel $M = (Q, \Sigma, \delta, q_0, F)$ aus einer/einem

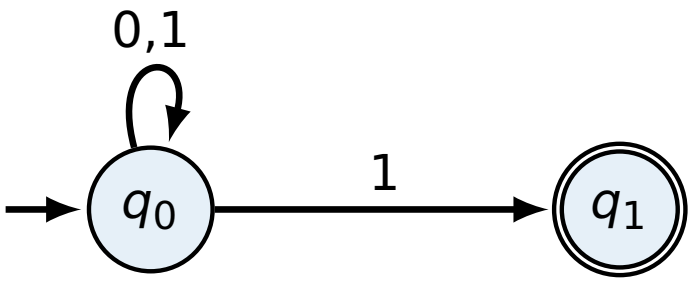
- endlichen Menge von Zuständen Q
- endlichen Eingabealphabet Σ
- totalen Übergangsfunktion $\delta : Q \times \Sigma \to Q$
- Startzustand $q_0 \in Q$
- Menge von Endzuständen $F \subseteq Q$



Definition (Nicht-Deterministischer endlicher Automat)

Ein NFA ist ein Tupel $N = (Q, \Sigma, \delta, S, F)$ mit

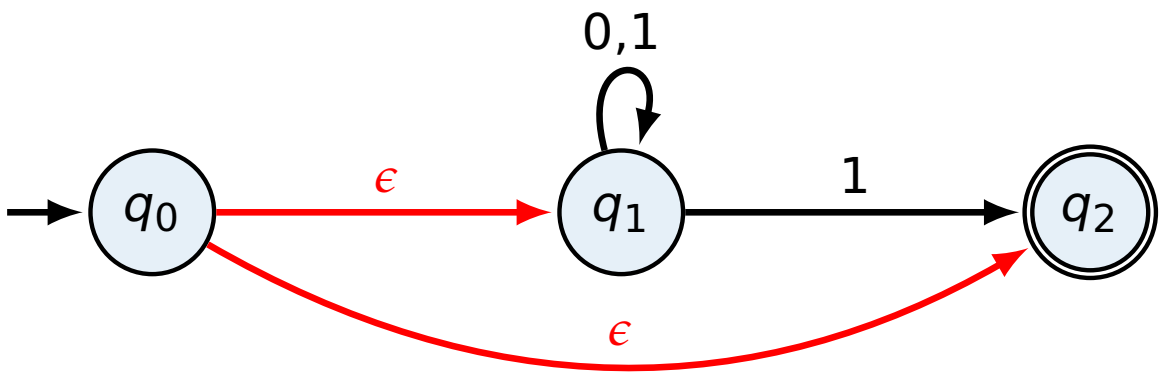
- Q, Σ, F wie ein DFA
- Menge von Startzuständen $S \subseteq F$
- Übergangsfunktion $\delta : Q \times \Sigma \to \mathcal{P}(Q)$



Definition (ϵ-Übergängen)

Ein ϵ-NFA ist ein Tupel $N = (Q, \Sigma, \delta, S, F)$ mit

- Q, Σ, F wie ein DFA
- Menge von Startzuständen $S \subseteq F$
- Übergangsfunktion $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$



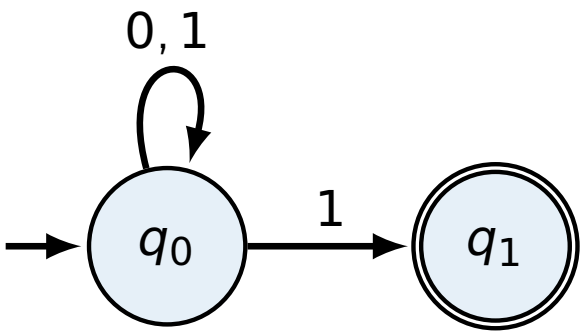
Potenzmengenkonstruktion

Konstruiere einen Automaten, der alle möglichen Pfade gleichzeitig berücksichtigt. Gegeben ein NFA $(Q, \Sigma, \delta, S, F)$, konstruiere einen DFA mit Zuständen aus $\mathcal{P}(Q)$.

- Starte in $\{S\}$
- Die Übergangsfunktion speichert alle möglichen Schritte

$$\overline{\delta} : \mathcal{P}(Q) \times \Sigma \rightarrow \mathcal{P}(Q)$$
$$(M, a) \mapsto \bigcup_{q \in M} \delta(q, a)$$

- M ist Endzustand wenn $F \cap M \neq \emptyset$



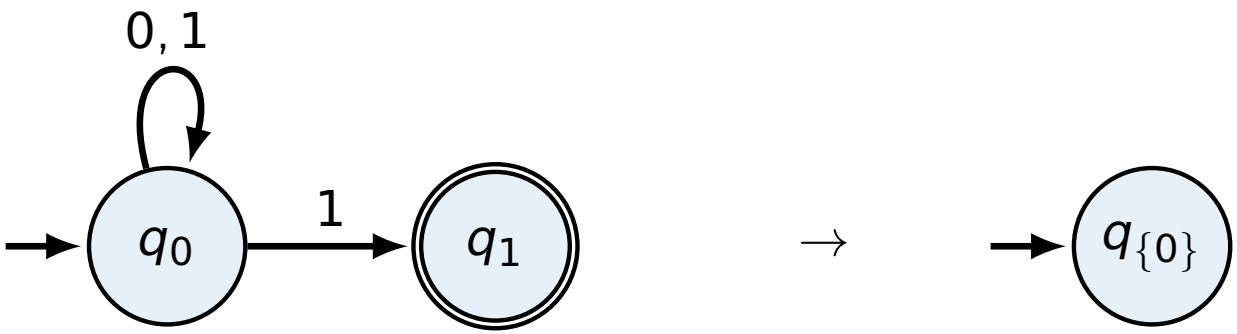
Potenzmengenkonstruktion

Konstruiere einen Automaten, der alle möglichen Pfade gleichzeitig berücksichtigt. Gegeben ein NFA $(Q, \Sigma, \delta, S, F)$, konstruiere einen DFA mit Zuständen aus $\mathcal{P}(Q)$.

- Starte in $\{S\}$
- Die Übergangsfunktion speichert alle möglichen Schritte

$$\overline{\delta} : \mathcal{P}(Q) \times \Sigma \rightarrow \mathcal{P}(Q)$$
$$(M, a) \mapsto \bigcup_{q \in M} \delta(q, a)$$

- M ist Endzustand wenn $F \cap M \neq \emptyset$



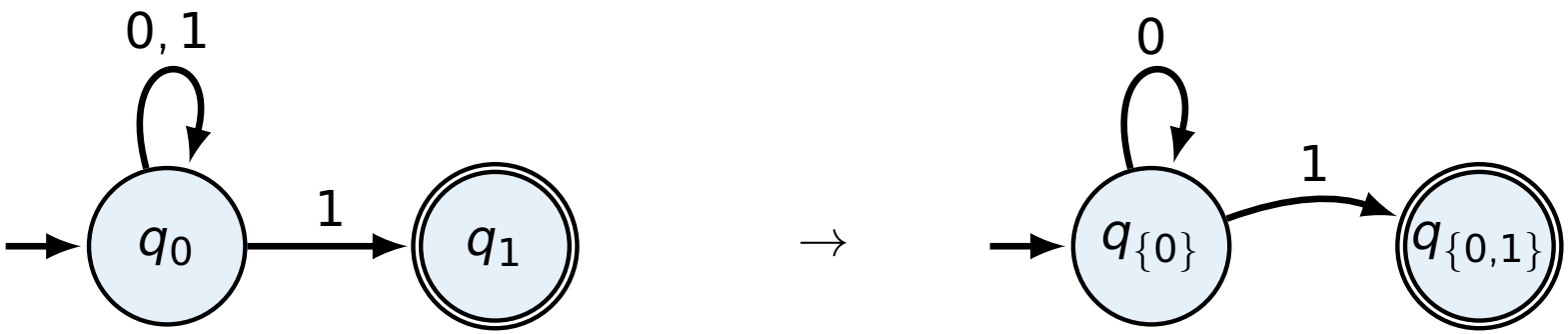
Potenzmengenkonstruktion

Konstruiere einen Automaten, der alle möglichen Pfade gleichzeitig berücksichtigt. Gegeben ein NFA $(Q, \Sigma, \delta, S, F)$, konstruiere einen DFA mit Zuständen aus $\mathcal{P}(Q)$.

- Starte in $\{S\}$
- Die Übergangsfunktion speichert alle möglichen Schritte

$$\overline{\delta} : \mathcal{P}(Q) \times \Sigma \rightarrow \mathcal{P}(Q)$$
$$(M, a) \mapsto \bigcup_{q \in M} \delta(q, a)$$

- M ist Endzustand wenn $F \cap M \neq \emptyset$



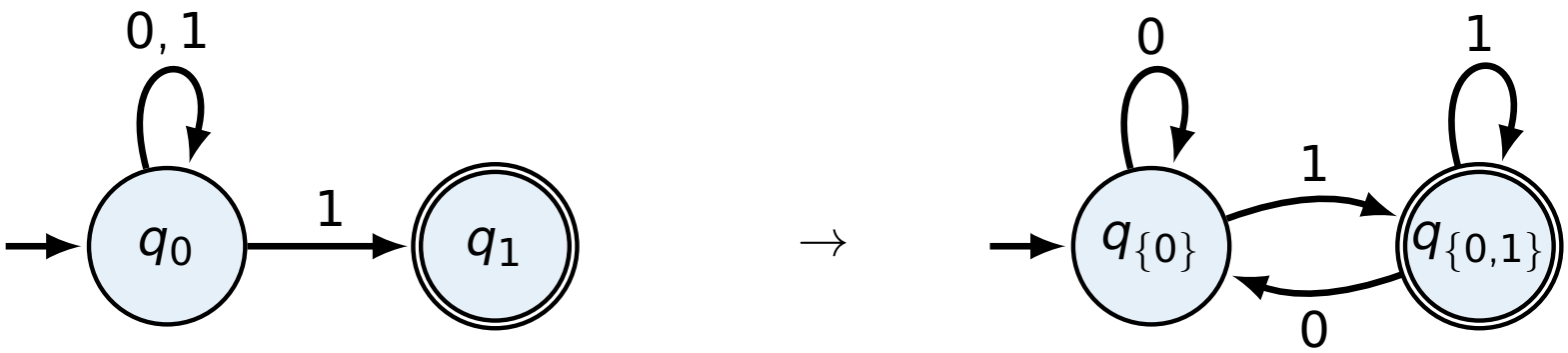
Potenzmengenkonstruktion

Konstruiere einen Automaten, der alle möglichen Pfade gleichzeitig berücksichtigt. Gegeben ein NFA $(Q, \Sigma, \delta, S, F)$, konstruiere einen DFA mit Zuständen aus $\mathcal{P}(Q)$.

- Starte in $\{S\}$
- Die Übergangsfunktion speichert alle möglichen Schritte

$$\overline{\delta} : \mathcal{P}(Q) \times \Sigma \rightarrow \mathcal{P}(Q)$$
$$(M, a) \mapsto \bigcup_{q \in M} \delta(q, a)$$

- M ist Endzustand wenn $F \cap M \neq \emptyset$



Definition (Regulärer Ausdruck)

Reguläre Ausdrücke sind induktiv definiert

- $\emptyset$ ist ein regulärer Ausdruck
- ϵ ist ein regulärer Ausdruck
- Für alle $a \in \Sigma$ ist a ein regulärer Ausdruck
- Sind α und β reguläre Ausdrücke, dann auch

  Konkatenation $\alpha\beta$
  Veroderung $\alpha \mid \beta$
  Wiederholung α^*

Analoge Sprachdefinition, z.b. $L(\alpha\beta) = L(\alpha)L(\beta)$

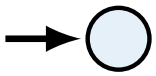
Beispiel

- $\alpha = (0|1)^*00$
- Worte bestehen aus einer beliebigen Folge von Einsen und Nullen gefolgt von zwei Nullen.
- $L(\alpha) \supseteq \{x \mid x \text{ Binärzahl}, x \mod 4 = 0\}$

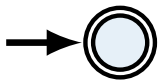
Thompson-Konstruktion

Für einen Ausdruck γ wird rekursiv mit struktureller Induktion ein ϵ-NFA konstruiert.

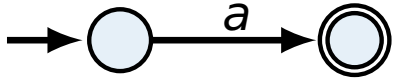
$\gamma = \emptyset$



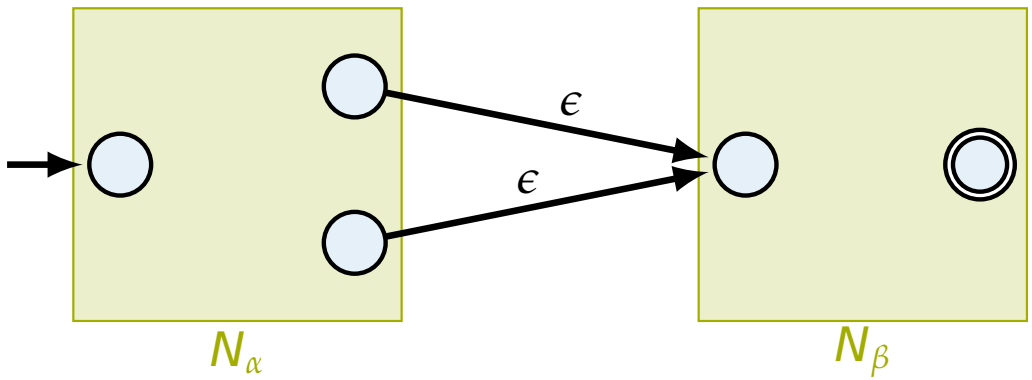
$\gamma = \epsilon$



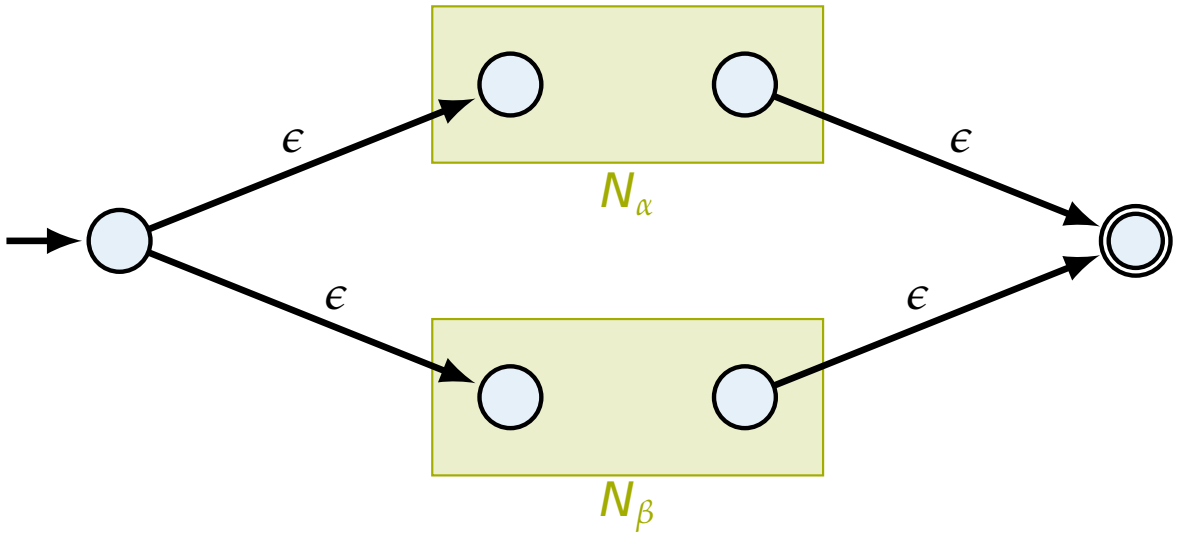
$\gamma = a \in \Sigma$



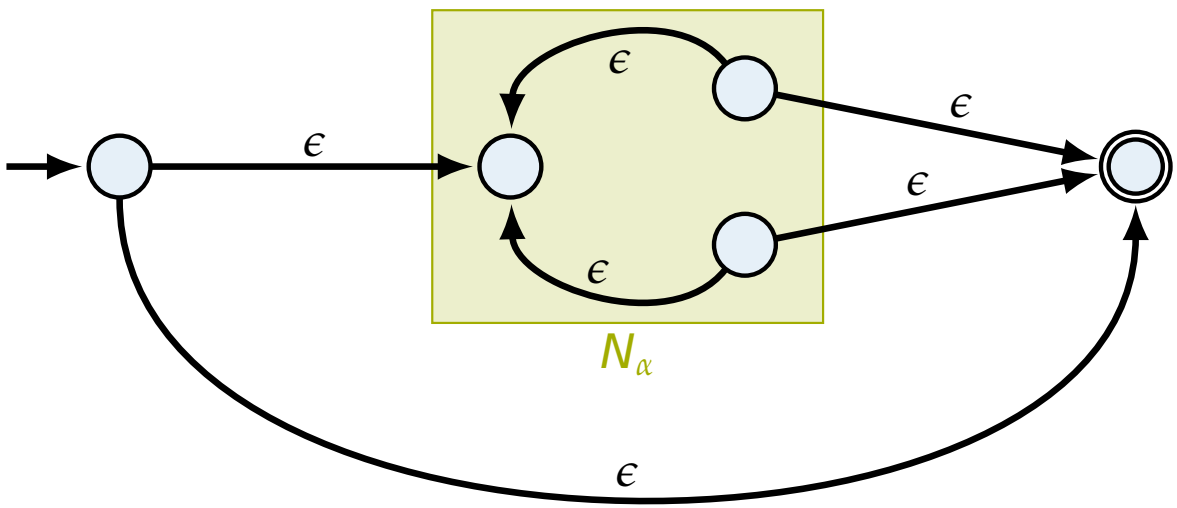
$\gamma = \alpha\beta$



$\gamma = \alpha \mid \beta$



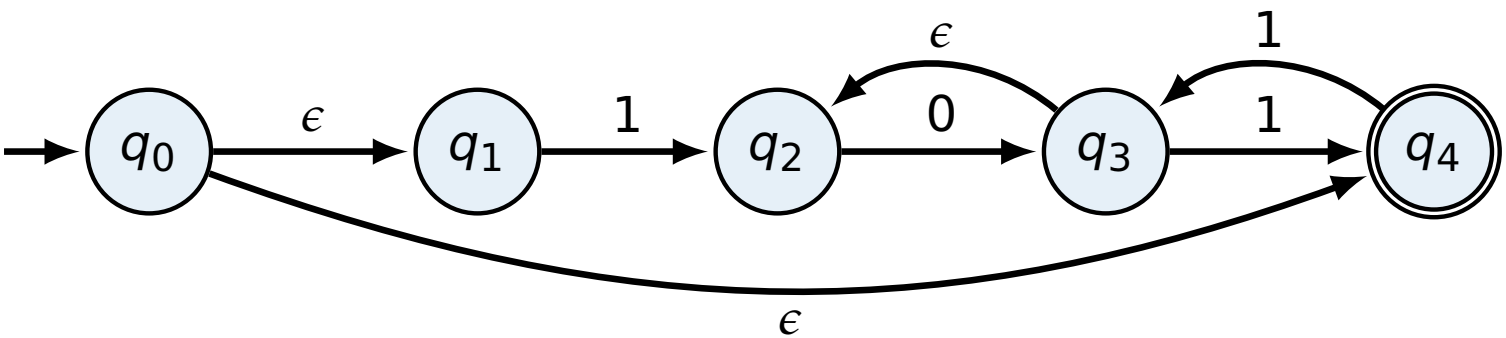
$\gamma = \alpha^*$



Idee

Entferne ϵ-Kanten durch das Bilden von ϵ-Hüllen.

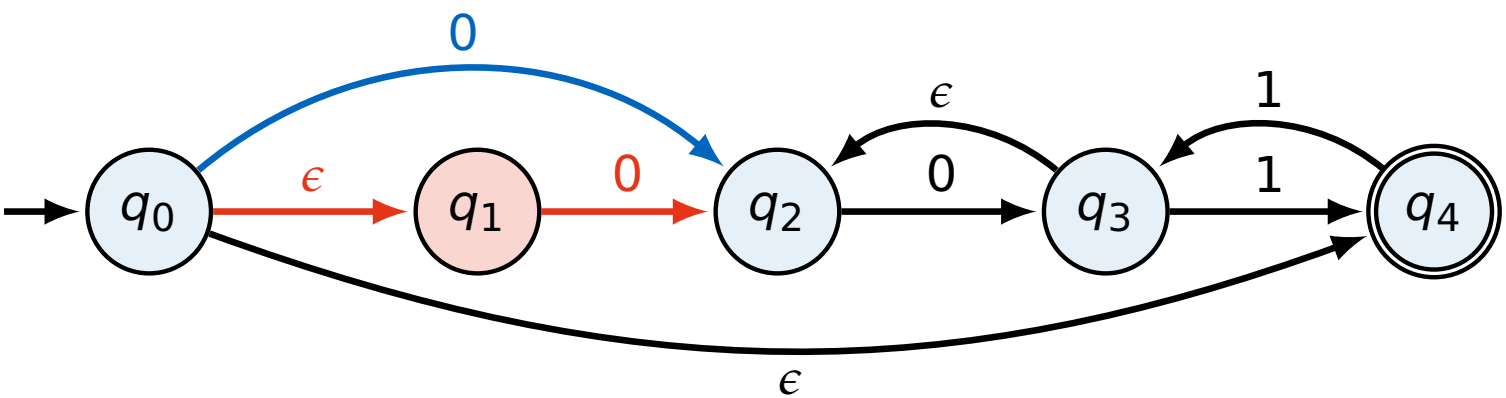
1. Entferne unnötige Knoten.
2. Für jeden Pfad der Form $\epsilon \ldots \epsilon a \epsilon \ldots \epsilon$ verbinde Anfangs- und Endknoten mit einer a-Kante.
3. Entferne alle ϵ-Kanten und unerreichbare Knoten.
4. Wurde das leere Wort akzeptiert mache den Anfangszustand zum Endzustand.



Idee

Entferne ϵ-Kanten durch das Bilden von ϵ-Hüllen.

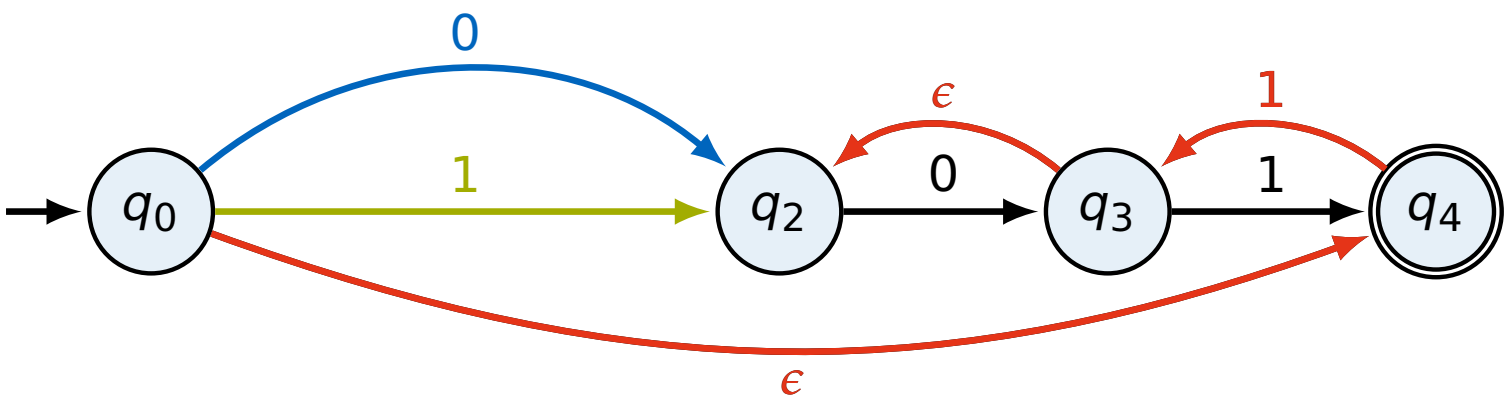
1. Entferne unnötige Knoten.
2. Für jeden Pfad der Form $\epsilon \ldots \epsilon a \epsilon \ldots \epsilon$ verbinde Anfangs- und Endknoten mit einer a-Kante.
3. Entferne alle ϵ-Kanten und unerreichbare Knoten.
4. Wurde das leere Wort akzeptiert mache den Anfangszustand zum Endzustand.



Idee

Entferne ϵ-Kanten durch das Bilden von ϵ-Hüllen.

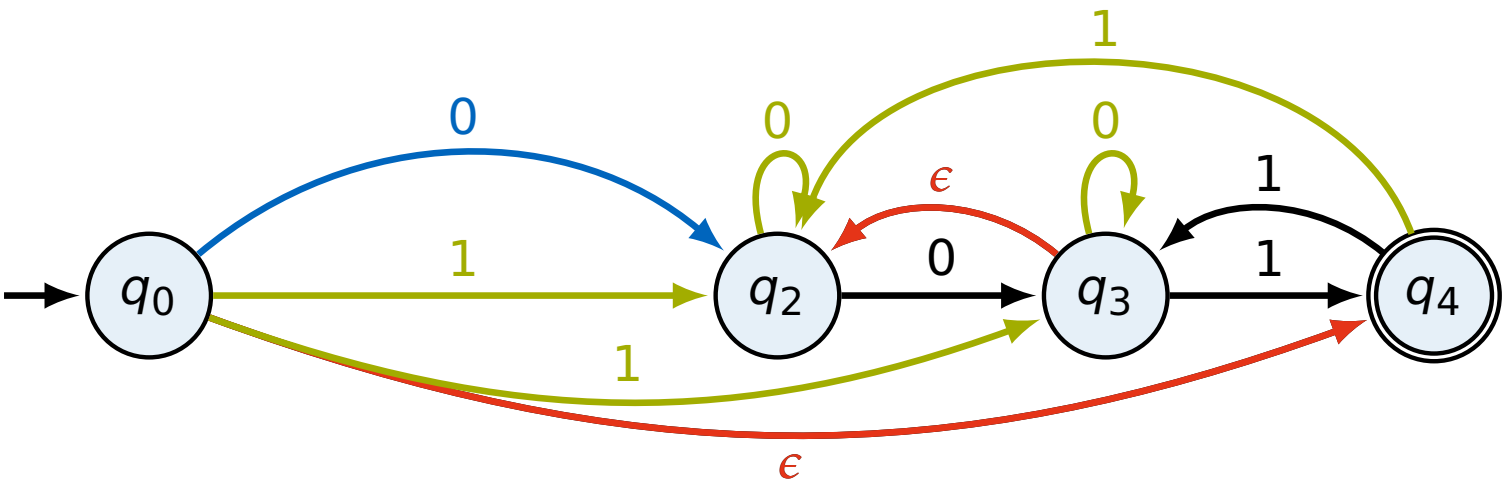
1. Entferne unnötige Knoten.
2. Für jeden Pfad der Form $\epsilon \ldots \epsilon a \epsilon \ldots \epsilon$ verbinde Anfangs- und Endknoten mit einer a-Kante.
3. Entferne alle ϵ-Kanten und unerreichbare Knoten.
4. Wurde das leere Wort akzeptiert mache den Anfangszustand zum Endzustand.



Idee

Entferne ϵ-Kanten durch das Bilden von ϵ-Hüllen.

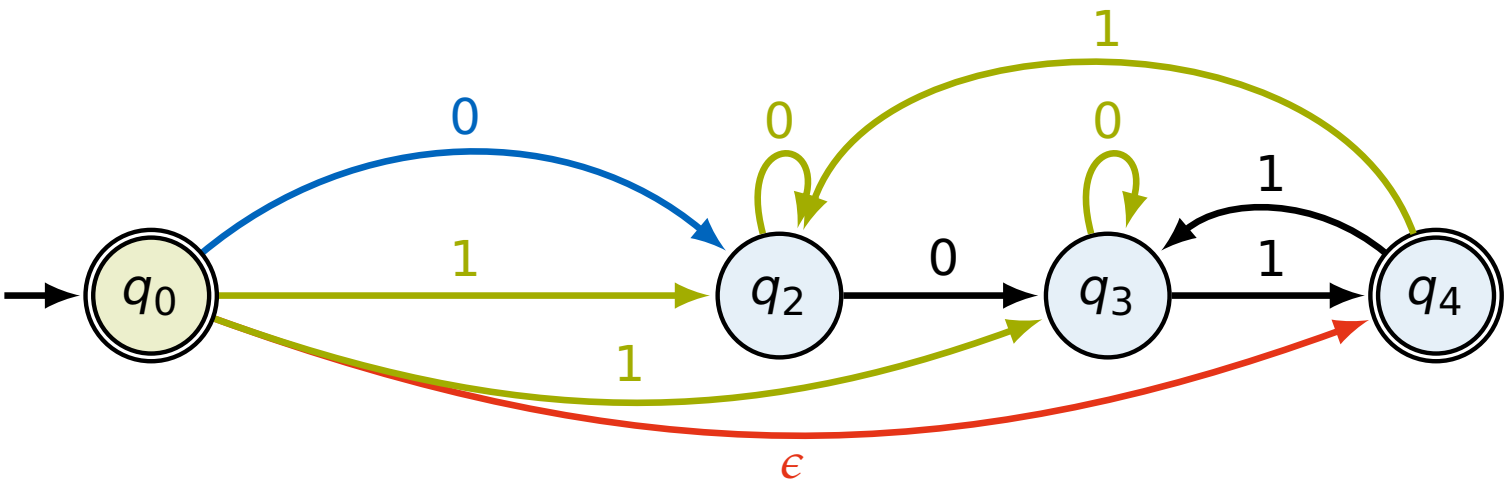
1. Entferne unnötige Knoten.
2. Für jeden Pfad der Form $\epsilon \ldots \epsilon a \epsilon \ldots \epsilon$ verbinde Anfangs- und Endknoten mit einer a-Kante.
3. Entferne alle ϵ-Kanten und unerreichbare Knoten.
4. Wurde das leere Wort akzeptiert mache den Anfangszustand zum Endzustand.



Idee

Entferne ϵ-Kanten durch das Bilden von ϵ-Hüllen.

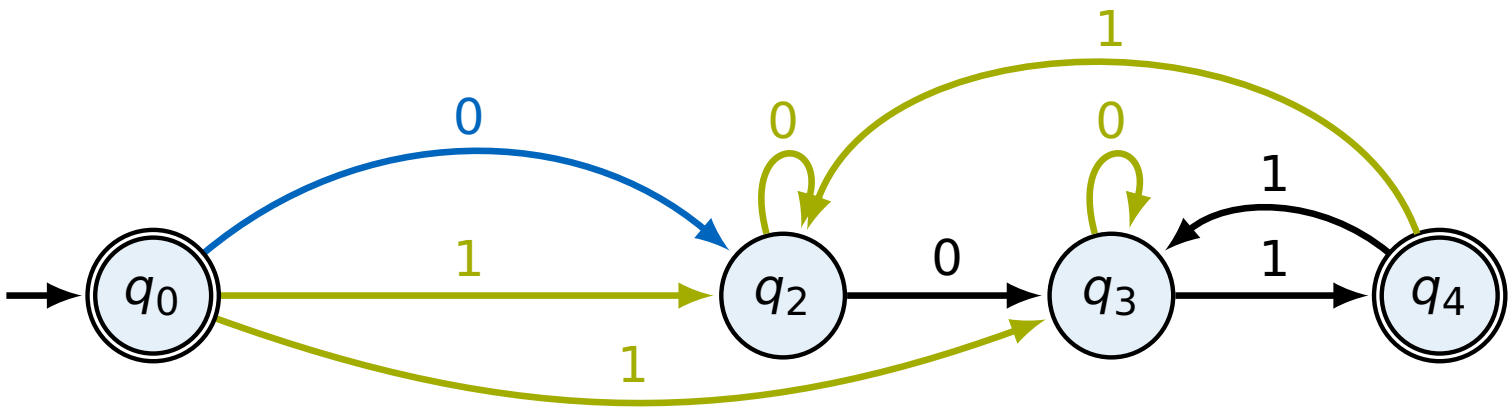
1. Entferne unnötige Knoten.
2. Für jeden Pfad der Form $\epsilon \ldots \epsilon a \epsilon \ldots \epsilon$ verbinde Anfangs- und Endknoten mit einer a-Kante.
3. Entferne alle ϵ-Kanten und unerreichbare Knoten.
4. Wurde das leere Wort akzeptiert mache den Anfangszustand zum Endzustand.



Idee

Entferne ϵ-Kanten durch das Bilden von ϵ-Hüllen.

1. Entferne unnötige Knoten.
2. Für jeden Pfad der Form $\epsilon \ldots \epsilon a \epsilon \ldots \epsilon$ verbinde Anfangs- und Endknoten mit einer a-Kante.
3. Entferne alle ϵ-Kanten und unerreichbare Knoten.
4. Wurde das leere Wort akzeptiert mache den Anfangszustand zum Endzustand.



Äquivalenzen

Definition (Äquivalente Worte)

Jede Sprache $L \subseteq \Sigma^*$ induziert eine Äquivalenzrelation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$

$$u \equiv_L v \iff (\forall w \in \Sigma^*. uw \in L \Leftrightarrow vw \in L)$$

Definition (Äquivalente Zustände)

Zwei Zustände im DFA A sind äquivalent wenn sie die selbe Sprache akzeptieren.

$$p \equiv_A q \iff \left(\forall w \in \Sigma^*. \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F\right)$$

Äquivalenzen

Definition (Äquivalente Worte)

Jede Sprache $L \subseteq \Sigma^*$ induziert eine Äquivalenzrelation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$

$$u \equiv_L v \iff (\forall w \in \Sigma^*. uw \in L \Leftrightarrow vw \in L)$$

Definition (Äquivalente Zustände)

Zwei Zustände im DFA A sind äquivalent wenn sie die selbe Sprache akzeptieren.

$$p \equiv_A q \iff \left(\forall w \in \Sigma^*. \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F\right)$$

Unterscheidbare Zustände

Definition (Unterscheidbarkeit)

Zwei Zustände sind unterscheidbar, wenn sie unterschiedliche Sprachen akzeptieren.

$$p \not\equiv_A q \iff \left(\exists w \in \Sigma^*.\hat{\delta}(p,w) \in F \wedge \hat{\delta}(q,w) \notin F\right)$$

Satz

Sind $\delta(p,a)$ und $\delta(q,a)$ unterscheidbar, dann auch p und q.

Unterscheidbare Zustände

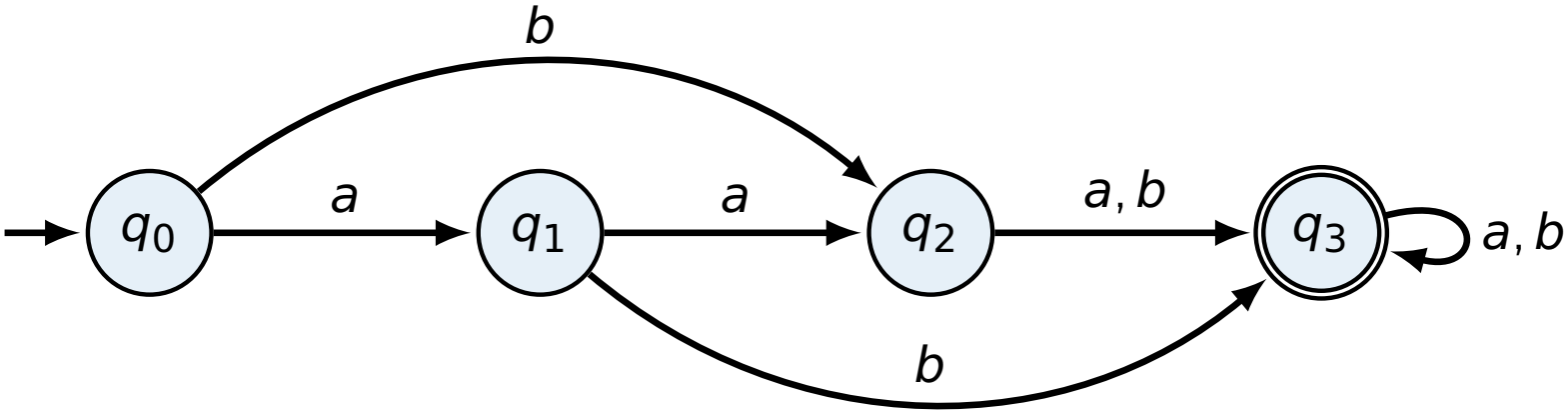
Definition (Unterscheidbarkeit)

Zwei Zustände sind unterscheidbar, wenn sie unterschiedliche Sprachen akzeptieren.

$$p \not\equiv_A q \iff \left(\exists w \in \Sigma^*.\hat{\delta}(p,w) \in F \wedge \hat{\delta}(q,w) \notin F\right)$$

Satz

Sind $\delta(p,a)$ und $\delta(q,a)$ unterscheidbar, dann auch p und q.



Unterscheidbare Zustände

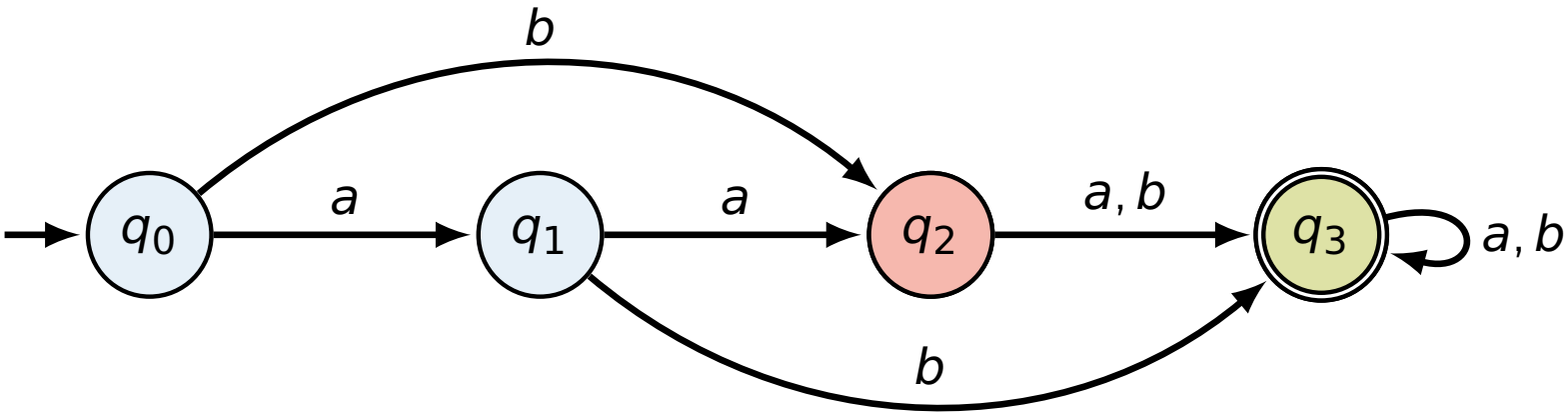
Definition (Unterscheidbarkeit)

Zwei Zustände sind unterscheidbar, wenn sie unterschiedliche Sprachen akzeptieren.

$$p \not\equiv_A q \iff \left(\exists w \in \Sigma^*.\hat{\delta}(p,w) \in F \wedge \hat{\delta}(q,w) \notin F\right)$$

Satz

Sind $\delta(p,a)$ und $\delta(q,a)$ unterscheidbar, dann auch p und q.



Unterscheidbare Zustände

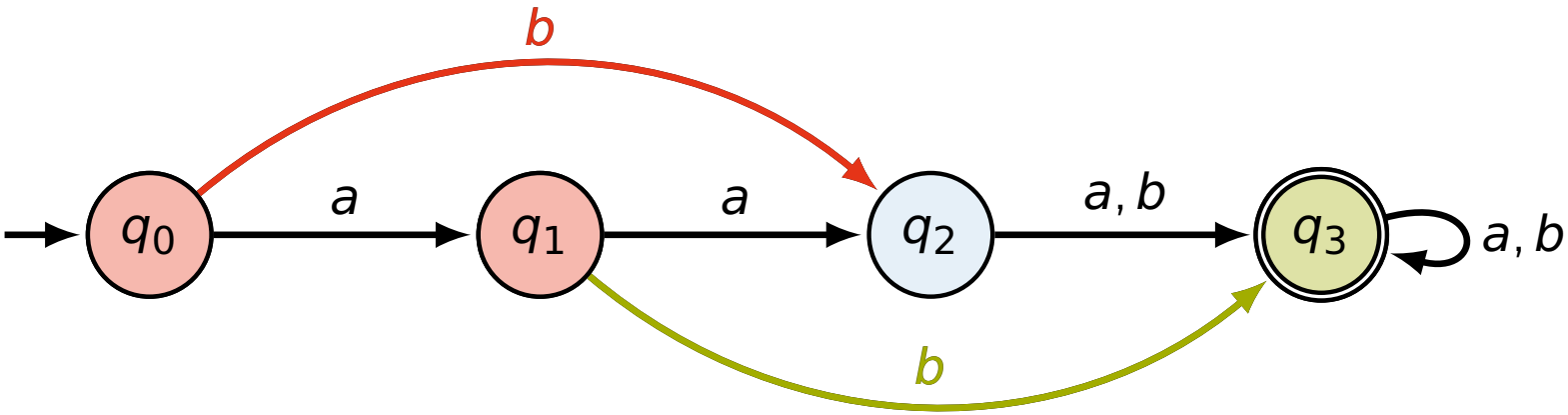
Definition (Unterscheidbarkeit)

Zwei Zustände sind unterscheidbar, wenn sie unterschiedliche Sprachen akzeptieren.

$$p \not\equiv_A q \iff \left(\exists w \in \Sigma^*.\hat{\delta}(p,w) \in F \wedge \hat{\delta}(q,w) \notin F\right)$$

Satz

Sind $\delta(p,a)$ und $\delta(q,a)$ unterscheidbar, dann auch p und q.

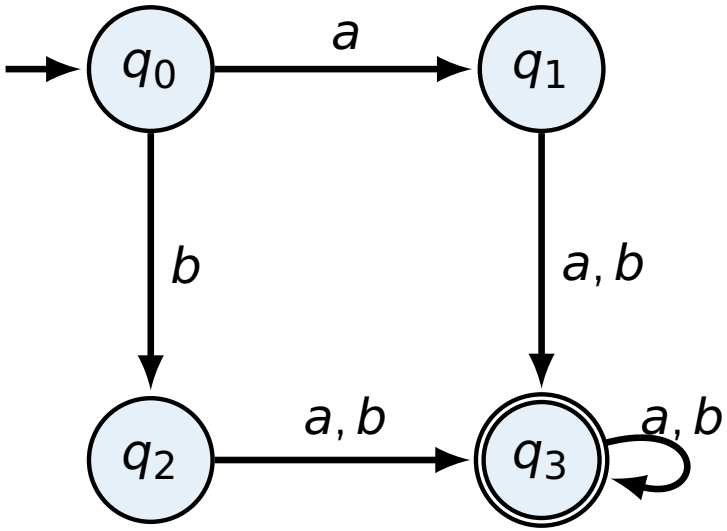


Quotientenautomat

1. Entferne alle von q_0 nicht erreichbaren Zustände
2. Berechne die unterscheidbaren Zustände
3. Kollabiere die äquivalenten Zustände

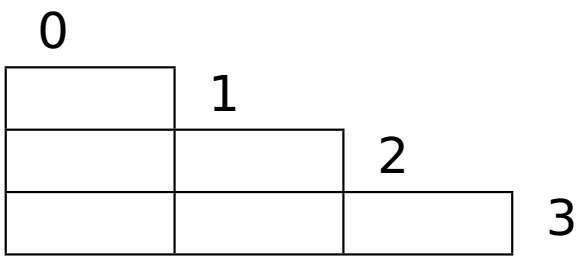
Quotientenautomat

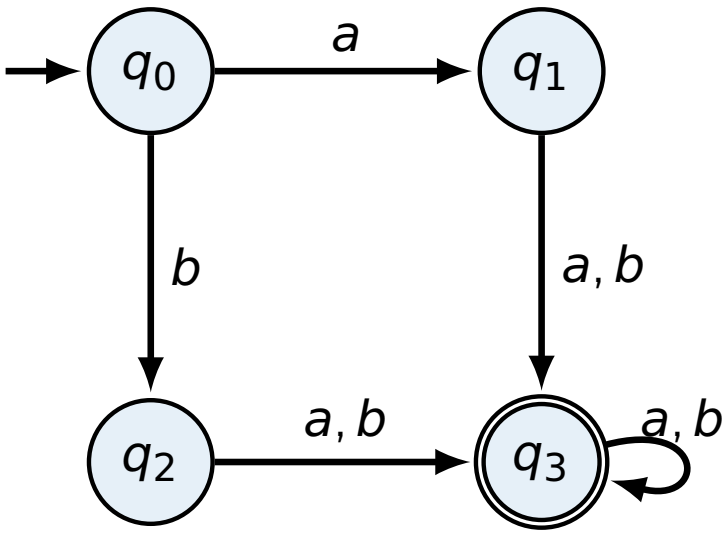
1. Entferne alle von q_0 nicht erreichbaren Zustände
2. Berechne die unterscheidbaren Zustände
3. Kollabiere die äquivalenten Zustände



Quotientenautomat

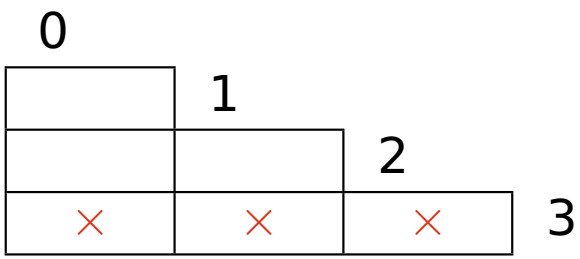
1. Entferne alle von q_0 nicht erreichbaren Zustände
2. Berechne die unterscheidbaren Zustände
3. Kollabiere die äquivalenten Zustände

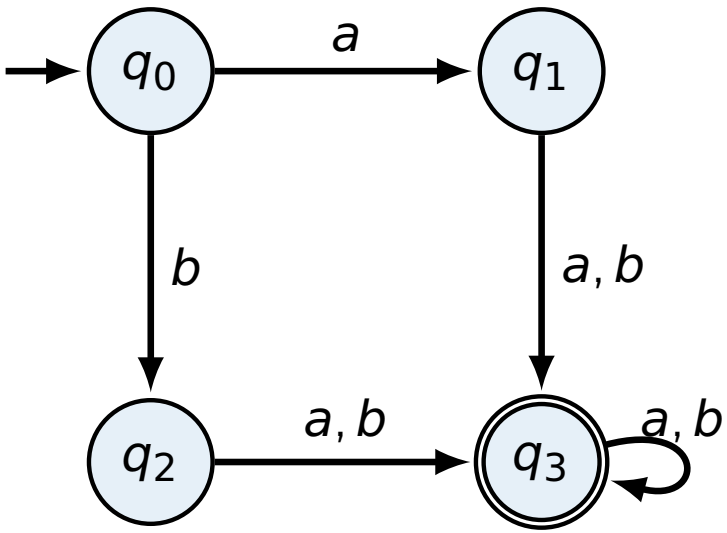




Quotientenautomat

1. Entferne alle von q_0 nicht erreichbaren Zustände
2. Berechne die unterscheidbaren Zustände
3. Kollabiere die äquivalenten Zustände

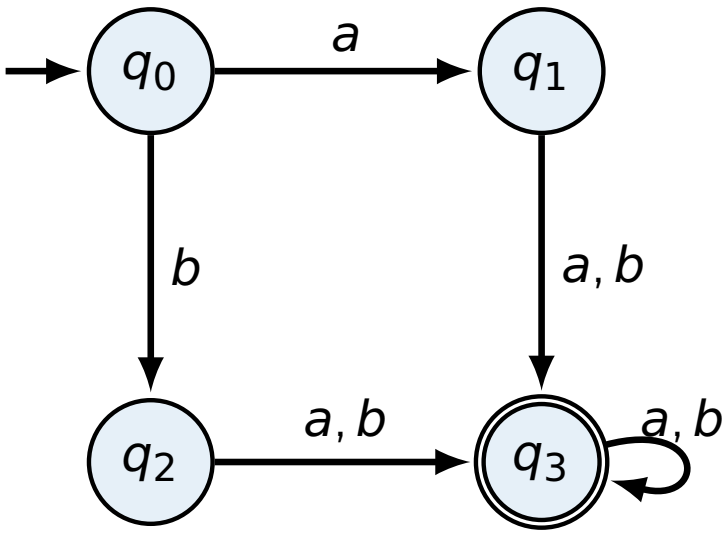




Quotientenautomat

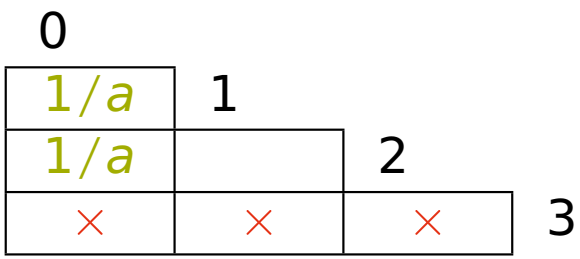
1. Entferne alle von q_0 nicht erreichbaren Zustände
2. Berechne die unterscheidbaren Zustände
3. Kollabiere die äquivalenten Zustände

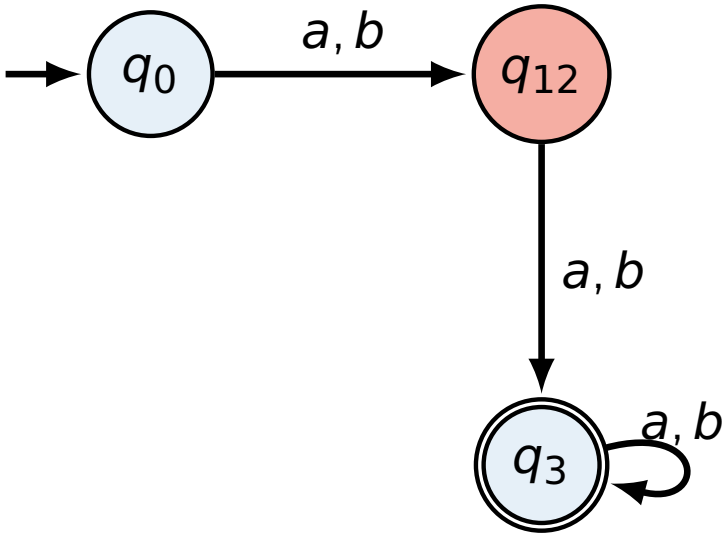
0	1	2	3
$1/a$			
$1/a$			
×	×	×	



Quotientenautomat

1. Entferne alle von q_0 nicht erreichbaren Zustände
2. Berechne die unterscheidbaren Zustände
3. Kollabiere die äquivalenten Zustände

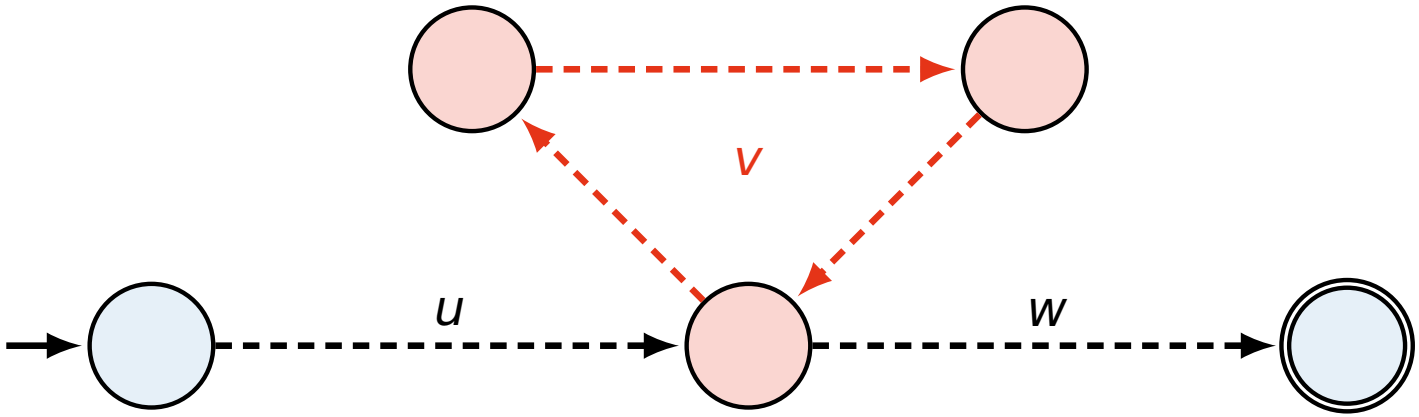




Satz (Pumping Lemma für reguläre Sprachen)

Sei $R \subseteq \Sigma^*$ regulär.
Dann gibt es ein $n > 0$, so dass sich jedes $z \in R$ mit $|z| \geq n$ so in $z = uvw$ zerlegen lässt, dass

- $v \neq \epsilon$
- $|uv| \leq n$
- $\forall i \geq 0. uv^i w \in R$



Eigenschaften von Symbolen

Definition

Sei $G = (V, \Sigma, P, S)$ eine CFG.
Ein Symbol $X \in V \cup \Sigma$ ist

nützlich	es gibt $S \to_G^* w \in \Sigma^*$ in der X vorkommt
erzeugend	es gibt $X \to_G^* w \in \Sigma^*$
erreichbar	es gibt $S \to_G^* \alpha X \beta$

Satz

Nützliche Symbole sind erzeugend und erreichbar. Aber nicht notwendigerweise umgekehrt.

$$S \to AB \mid a, \quad A \to b$$

Definition (Chomsky-Normalform)

Eine kontextfreie Grammatik ist in Chomsky-Normalform (CNF) genau dann wenn alle Produktionen die Form

$$A \to a \quad \text{oder} \quad A \to BC$$

haben.

Satz

Zu jeder CFG G existiert eine CFG G' in Chomsky-Normalform mit

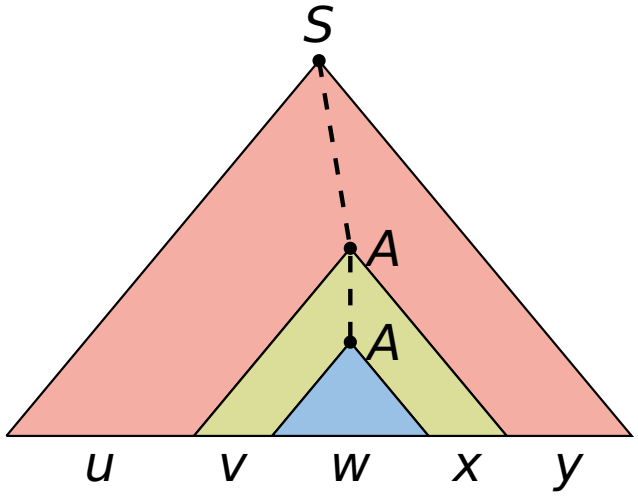
$$L(G') = L(G) \setminus \{\epsilon\}$$

Satz (Pumping Lemma für kontextfreie Sprachen)

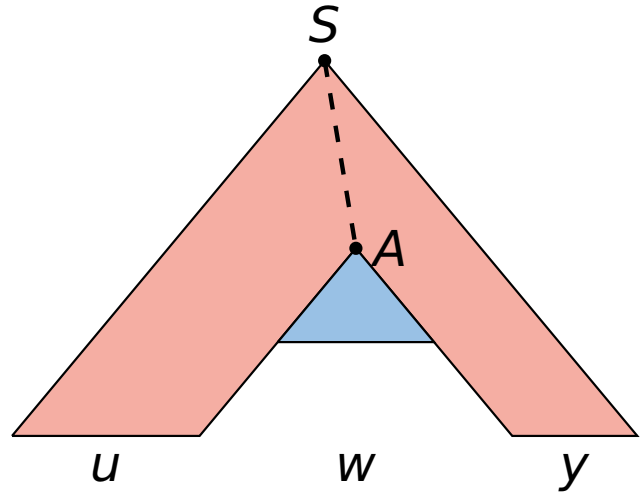
Sei $L \subseteq \Sigma^*$ kontextfrei.
Dann gibt es ein $n > 0$, so dass sich jedes $z \in L$ mit $|z| \geq n$ so in $z = uvwxy$ zerlegen lässt, dass

- $vx \neq \epsilon$
- $|vwx| \leq n$
- $\forall i \geq 0. uv^i wx^i y \in L$

$S \to^* uAy \to^* uvAxy \to^* uvwxy$



$S \to^* uAy \to^* uwy$



CNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Eliminiere ϵ-Produktionen
2. Eliminiere Kettenproduktionen
3. Ersetze Terminale durch Nichtterminale
4. Verkürze Ketten von Nichtterminalen der Länge ≥ 3

CNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Eliminiere ϵ-Produktionen
2. Eliminiere Kettenproduktionen
3. Ersetze Terminale durch Nichtterminale
4. Verkürze Ketten von Nichtterminalen der Länge ≥ 3

Sind $B \to \epsilon$ und $A \to \alpha B \beta$ in P, dann füge $A \to \alpha\beta$ hinzu. Entferne danach alle ϵ-Produktionen.

$$S \to Ab, \quad A \to aAA \mid \epsilon$$

wird zu:

$$S \to Ab \mid b$$
$$A \to aAA \mid aA \mid a$$

CNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Eliminiere ϵ-Produktionen
2. Eliminiere Kettenproduktionen
3. Ersetze Terminale durch Nichtterminale
4. Verkürze Ketten von Nichtterminalen der Länge ≥ 3

Sind $A \to B$ und $B \to \alpha$ in P, dann füge $A \to \alpha$ hinzu. Entferne danach alle Kettenproduktionen und unerreichbaren Symbole.

$$S \to A, \quad A \to a \mid B, \quad B \to bS$$

wird zu:

$$A \to a \mid bS$$
$$S \to a \mid bS$$

CNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Eliminiere ϵ-Produktionen
2. Eliminiere Kettenproduktionen
3. Ersetze Terminale durch Nichtterminale
4. Verkürze Ketten von Nichtterminalen der Länge ≥ 3

Ersetze jedes $a \in \Sigma$ in einer rechten Seite länger als 1 durch ein neues Nichtterminal.

$$S \to aa \mid Bb \mid b, \quad B \to \ldots$$

wird zu:

$$S \to X_a X_a \mid B X_b \mid b$$
$$X_a \to a, \quad X_b \to b$$

CNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Eliminiere ϵ-Produktionen
2. Eliminiere Kettenproduktionen
3. Ersetze Terminale durch Nichtterminale
4. Verkürze Ketten von Nichtterminalen der Länge ≥ 3

Ersetze jede Produktion der Form $A \to B_1 B_2 \dots B_k$ durch neue Nichtterminale mit Produktionen der Länge 2.

$$S \to X_a X_b B X_a, \quad X_a \to a, \quad X_b \to b, \quad B \to \dots$$

wird zu:

$$S \to X_a T_1$$
$$T_1 \to X_b T_2, \quad T_2 \to B X_a$$

Definition (Cocke-Younger-Kasami-Algorithmus)

Der CYK-Algorithmus entscheidet das Wortproblem für kontextfreie Grammatiken in Chomsky-Normalform in $\mathcal{O}(n^3)$.
Gegeben eine Grammatik $G = (V, \Sigma, P, S)$ in CNF und ein Wort $w = a_1 \dots a_n \in \Sigma^*$. Mit

$$V_{ij} := \{A \in V \mid A \to_G^* a_i \dots a_j\}$$

ist

$$w \in L(G) \Leftrightarrow S \in V_{1n}$$

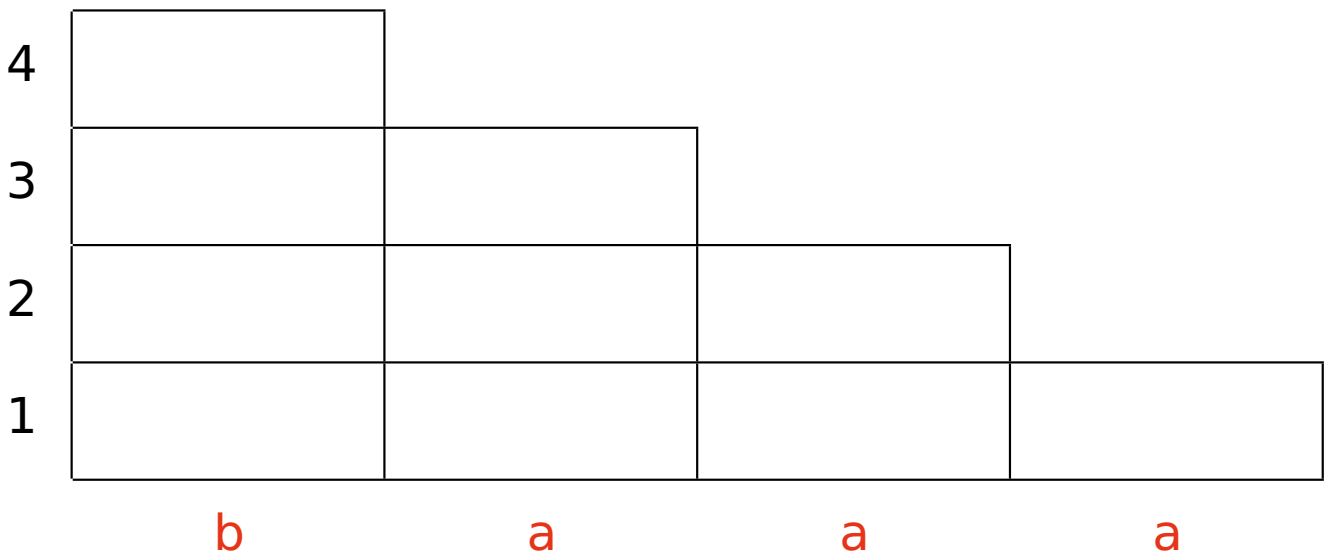
$$V_{ii} = \{A \in V \mid (A \to a_i) \in P\}$$
$$V_{ij} = \{A \in V \mid \exists k, B \in V_{ik}, C \in V_{k+1,j} \,.\, (A \to BC) \in P\}$$

CYK-Algorithmus

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

1. Initialisiere mit den V_{ii}.
2. Befülle die Tabelle von unten nach oben.

$$S \to AB \mid BC, \quad A \to BA \mid a, \quad B \to CC \mid b, \quad C \to AB \mid a$$

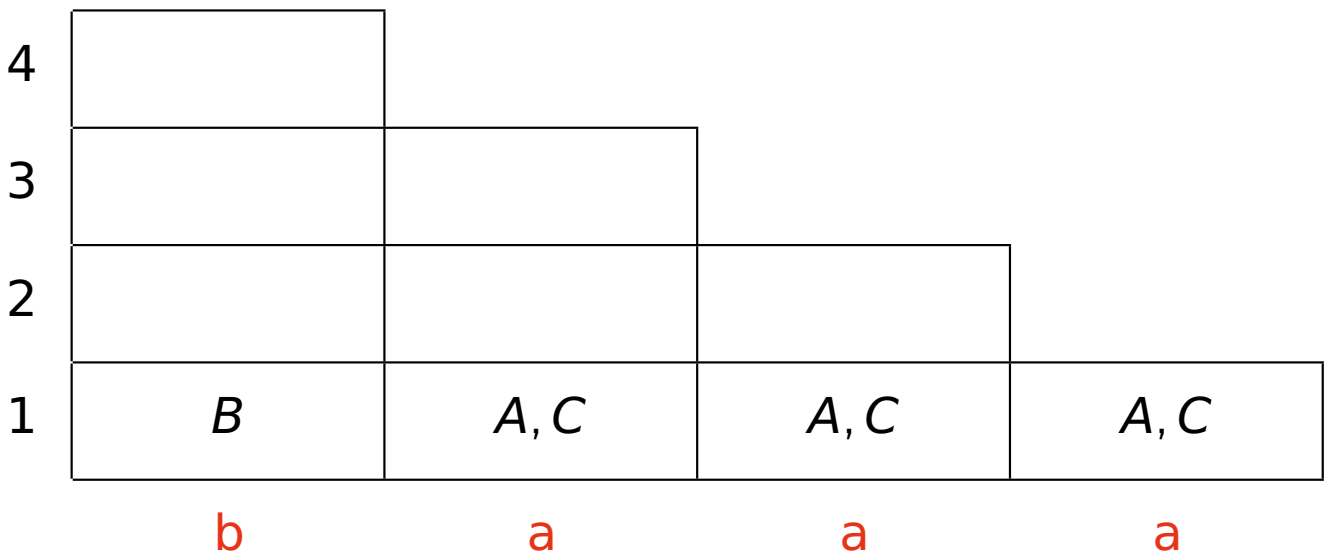


CYK-Algorithmus

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

1. Initialisiere mit den V_{ii}.
2. Befülle die Tabelle von unten nach oben.

$$S \to AB \mid BC, \quad A \to BA \mid a, \quad B \to CC \mid b, \quad C \to AB \mid a$$

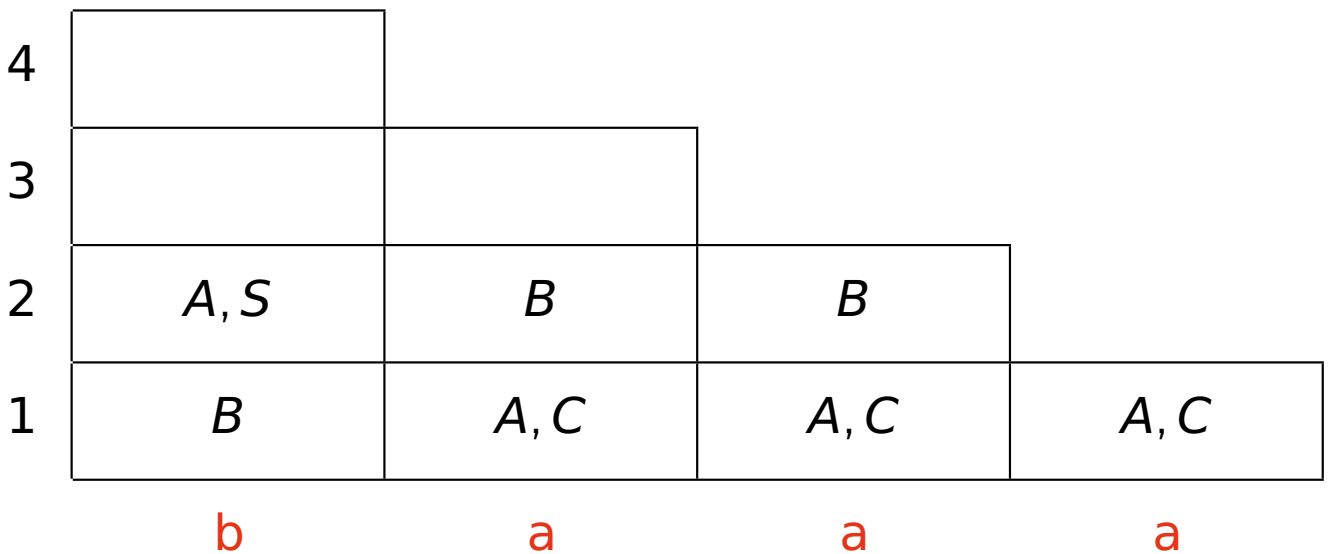


CYK-Algorithmus

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

1. Initialisiere mit den V_{ii}.
2. Befülle die Tabelle von unten nach oben.

$$S \to AB \mid BC, \quad A \to BA \mid a, \quad B \to CC \mid b, \quad C \to AB \mid a$$



CYK-Algorithmus

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

1. Initialisiere mit den V_{ii}.
2. Befülle die Tabelle von unten nach oben.

$$S \to AB \mid BC, \quad A \to BA \mid a, \quad B \to CC \mid b, \quad C \to AB \mid a$$

4				
3	$\emptyset$	S, A, C		
2	A, S	B	B	
1	B	A, C	A, C	A, C
	b	a	a	a

CYK-Algorithmus

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

1. Initialisiere mit den V_{ii}.
2. Befülle die Tabelle von unten nach oben.

$$S \to AB \mid BC, \quad A \to BA \mid a, \quad B \to CC \mid b, \quad C \to AB \mid a$$

4	$S, \ldots$			
3	$\emptyset$	S, A, C		
2	A, S	B	B	
1	B	A, C	A, C	A, C
	b	a	a	a

Induktive Sprachdefinition

Die induktive Definition zu einer Grammatik G ergibt sich direkt aus ihren Produktionen. Dabei werden kleinere Worte zu größeren Worten zusammengesetzt, die Definition erfolgt bottom-up.

Beispiel

Mit den Produktionen $S \to 0S1 \mid S11 \mid 1$:

$$\begin{aligned} & & 1 \in L_G(S) \\ u \in L_G(S) & \implies & 0u1 \in L_G(S) \\ u \in L_G(S) & \implies & u11 \in L_G(S) \end{aligned}$$

Also z.B:

$$1 \in L_G(S) \implies 010 \in L_G(S) \implies 01011 \in L_G(S)$$

Satz (Ogden Lemma für kontextfreie Sprachen)

Sei $L \subseteq \Sigma^*$ kontextfrei.
Dann gibt es ein $n > 0$, so dass für jedes $z \in L$ mit $|z| \geq n$ gilt:
Für jede Markierung M von mindestens n Buchstaben in z gibt es eine Zerlegung $z = uvwxy$ mit

- $|vx|_M \geq 1$
- $|vwx|_M \leq n$
- $\forall i \geq 0. uv^i wx^i y \in L$

Beispiel (Markierung)

Sei $w = abaabaaa$ ein Wort. Dann ist

$$w = abaabaaa$$

eine Markierung mit $|w|_M = 3$.

Definition (Greibach-Normalform)

Eine kontextfreie Grammatik ist in Greibach-Normalform (GNF) genau dann wenn alle Produktionen außer $S \to \epsilon$ die Form

$$A \to a\alpha \quad \text{mit} \quad a \in \Sigma, \alpha \in V^*$$

haben.

Satz

Zu jeder CFG G existiert eine CFG G' in Greibach-Normalform mit

$$L(G') = L(G)$$

Einsetzen von Produktionen

Satz (Einsetzen von Produktionen)

Enthält eine CFG die Produktionen

$$A \to \alpha_1 B \alpha_2$$
$$B \to \beta_1 \mid \cdots \mid \beta_k$$

so ändert sich die erzeugte Sprache nicht, wenn man B in A einsetzt.

$$A \to \alpha_1 \beta_1 \alpha_2 \mid \cdots \mid \alpha_1 \beta_k \alpha_2$$

Beispiel

Die Grammatik

$$S \to a \mid aBc$$
$$B \to b \mid bS$$

ist äquivalent zur Grammatik

$$S \to a \mid abc \mid abSc$$

Linksrekursive Produktionen

Definition (Linksrekursive Produktion)

Man nennt eine Produktion linksrekursiv, wenn sie die Form

$$A \to A\alpha_1 \mid \cdots \mid A\alpha_k \mid \beta_1 \mid \cdots \mid \beta_l \quad \text{mit} \quad \alpha_i, \beta_i \in (V \cup \Sigma)^+$$

hat, wobei die β_i nicht mit A beginnen.

Satz (Ersetzen von linksrekursiven Produktionen)

Sei A eine linksrekursive Produktion einer CFG.
Dann ändert sich die erzeugte Sprache nicht, wenn wir A ersetzen durch

$$A \to \beta_1 \mid \cdots \mid \beta_l \mid \beta_1 B \mid \cdots \mid \beta_l B$$
$$B \to \alpha_1 \mid \cdots \mid \alpha_k \mid \alpha_1 B \mid \cdots \mid \alpha_k B$$

B ist niemals linksrekursiv.

GNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Nummeriere Nichtterminale
2. Mache Prduktionen aufsteigend und nicht rekursiv
3. Setze Produktionen absteigend ein

GNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Nummeriere Nichtterminale
2. Mache Prduktionen aufsteigend und nicht rekursiv
3. Setze Produktionen absteigend ein

Benenne alle Nichtterminale beliebig um in $A_1, \ldots, A_{|V|}$.

$$S \to Ab, \quad A \to aAS \mid \epsilon$$

wird zu

$$A_1 \to A_2 b$$
$$A_2 \to aA_2A_1$$

GNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Nummeriere Nichtterminale
2. Mache Prduktionen aufsteigend und nicht rekursiv
3. Setze Produktionen absteigend ein

Betrachte alle Produktionen $A_l \to \ldots$ in aufsteigender Reihenfolge.

- Existieren Produktionen der Form $A_l \to A_r\alpha$ mit $r < l$, dann setze A_r in A_l ein.

  $$A_1 \to A_2 \mid a \mid b$$
  $$A_2 \to A_1A_1$$

  wird zu

  $$A_1 \to a \mid b$$
  $$A_2 \to A_2A_1 \mid aA_1 \mid bA_1$$

- Entferne danach alle linksrekursiven A_l-Produktionen.

GNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Nummeriere Nichtterminale
2. Mache Prduktionen aufsteigend und nicht rekursiv
3. Setze Produktionen absteigend ein

Betrachte alle Produktionen $A_l \to \ldots$ in aufsteigender Reihenfolge.

- Existieren Produktionen der Form $A_l \to A_r\alpha$ mit $r < l$, dann setze A_r in A_l ein.
- Entferne danach alle linksrekursiven A_l-Produktionen.

$$A_2 \to A_2A_1 \mid aA_1 \mid bA_1$$

wird zu

$$A_2 \to aA_1 \mid bA_1 \mid aA_1A_3 \mid bA_1A_3$$
$$A_3 \to A_1 \mid A_1A_3$$

GNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Nummeriere Nichtterminale
2. Mache Prduktionen aufsteigend und nicht rekursiv
3. Setze Produktionen absteigend ein

Betrachte alle Produktionen $A_l \to \ldots$ in absteigender Reihenfolge.

- Existieren Produktionen der Form $A_l \to A_r\alpha$ mit $r > l$, dann setze A_r in A_l ein.

$$A_1 \to a \mid b \mid A_2$$
$$A_2 \to aA_1 \mid bA_1 \mid aA_1A_3 \mid bA_1A_3$$
$$A_3 \to bA_3 \mid c$$

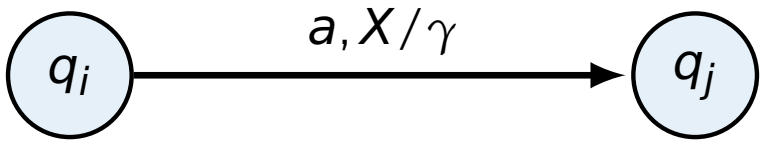
A_1 wird zu

$$A_1 \to a \mid b \mid aA_1 \mid bA_1 \mid aA_1A_3 \mid bA_1A_3$$

Definition (Kellerautomat)

Ein PDA (Push-Down-Automaton) ist ein Tupel $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ aus einer/einem

- endlichen Menge von Zuständen Q
- endlichen Eingabealphabet Σ
- endlichen Kelleralphabet Γ
- Übergangsfunktion $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to P(Q \times \Gamma^*)$
- Startzustand $q_0 \in Q$
- Kellerinitialisierung $Z_0 \in \Gamma$
- Menge von Endzuständen $F \subseteq Q$



Definition (Kellerautomat)

Ein PDA (Push-Down-Automaton) ist ein Tupel $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ aus einer/einem

- Übergangsfunktion $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to P(Q \times \Gamma^*)$

Definition (Akzeptanz)

Ein PDA P akzeptiert $w \in \Sigma^*$ mit Endzustand gdw

$$\exists f \in F, \gamma \in \Gamma^*.(q_0, w, Z_0) \to_P^* (f, \epsilon, \gamma)$$

Ein PDA P akzeptiert $w \in \Sigma^*$ mit leerem Keller gdw

$$\exists q \in Q.(q_0, w, Z_0) \to_P^* (q, \epsilon, \epsilon)$$

Definition (Kellerautomat)

Ein PDA (Push-Down-Automaton) ist ein Tupel $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ aus einer/einem

- Übergangsfunktion $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to P(Q \times \Gamma^*)$

Beispiel

PDA akzeptierend mit leerem Keller zu $L = \{a^n b^n \mid n \in \mathbb{N}_0\}$.