Übersichtsfolien zur Übung

Theoretische Informatik Sommersemester 2013

Markus Kaiser

July 15, 2013

Organisatorisches

- Mail: tutor@zfix.org
- Web: theo.zfix.org
- Hg: tutor.zfix.org

- Wann?
  - Dienstag 10:15-11:45 00.08.038
  - Dienstag 12:10-13:45 00.08.038
- Hausaufgaben
  - Abgabe am Montag 14h, allein
  - Rückgabe in der richtigen Übung
  - Notenbonus für 40% der Punkte, 40% in der zweiten Hälfte
- Klausur
  - Endterm: Mi 31.07. 11.30-14h
  - Wiederholung: Do 26.09. 11-13.30h

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

Alphabet

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)

- $AB = \{uv \mid u \in A \wedge v \in B\}$
- $A^n = \{w_1 \dots w_n \mid w_1 \dots w_n \in A\}, \qquad A^0 = \{\epsilon\}$
- $A^* = \bigcup_{n \in \mathbb{N}_0} A^n$

Definition (Deterministischer endlicher Automat)

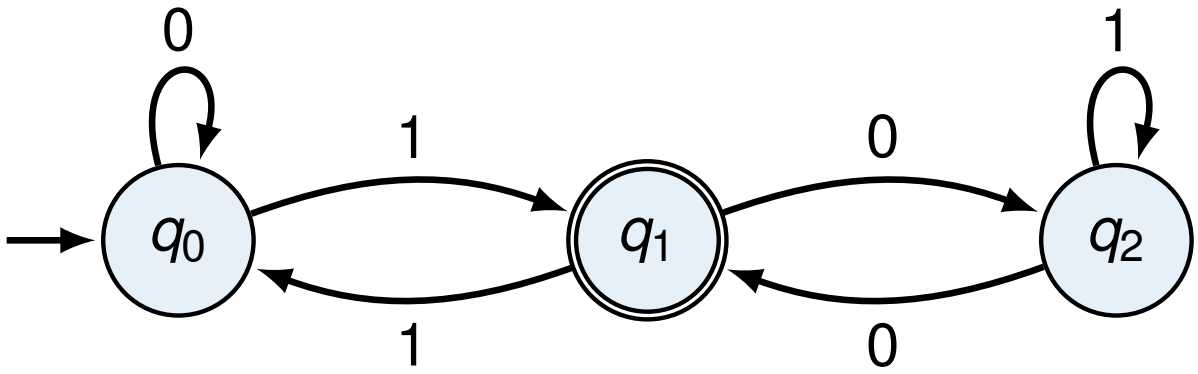
Ein DFA ist ein Tupel $M = (Q, \Sigma, \delta, q_0, F)$ aus einer/einem

- 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 (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)

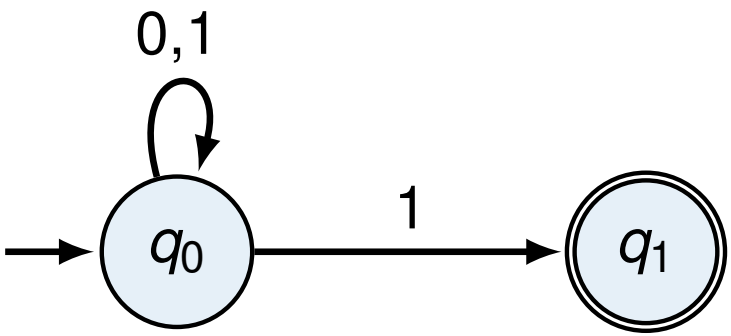
Ein NFA ist ein Tupel $N = (Q, \Sigma, \delta, q_0, F)$ mit

- Q, Σ, q_0, F wie ein DFA
- Übergangsfunktion $\delta : Q \times \Sigma \to P(Q)$

Definition (Nicht-Deterministischer endlicher Automat)

Ein NFA ist ein Tupel $N = (Q, \Sigma, \delta, q_0, F)$ mit

- Q, Σ, q_0, F wie ein DFA
- Übergangsfunktion $\delta : Q \times \Sigma \to P(Q)$



Definition (NFA mit ϵ-Übergängen)

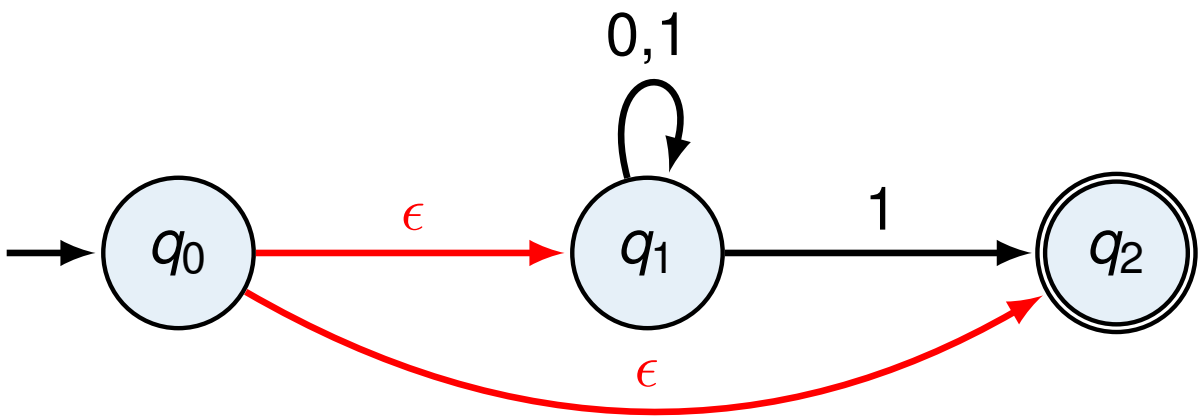
Ein ϵ-NFA ist ein Tupel $N = (Q, \Sigma, \delta, q_0, F)$ mit

- Q, Σ, q_0, F wie ein DFA
- Übergangsfunktion $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to P(Q)$

Definition (NFA mit ϵ-Übergängen)

Ein ϵ-NFA ist ein Tupel $N = (Q, \Sigma, \delta, q_0, F)$ mit

- Q, Σ, q_0, F wie ein DFA
- Übergangsfunktion $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to P(Q)$



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. $L(\alpha\beta) = L(\alpha)L(\beta)$

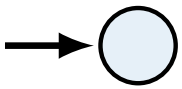
Beispiel

$\alpha = (0|1)^*00$ $\qquad L(\alpha) = \{x \mid x \text{ Binärzahl}, x \mod 4 = 0\}$

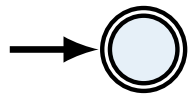
Idee (Kleene)

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

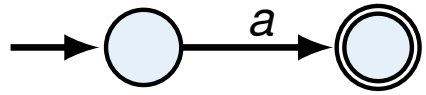
$\gamma = \emptyset$



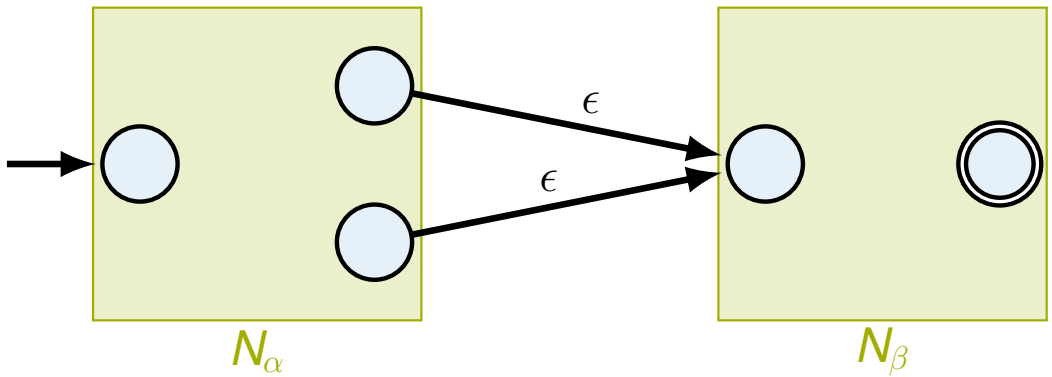
$\gamma = \epsilon$



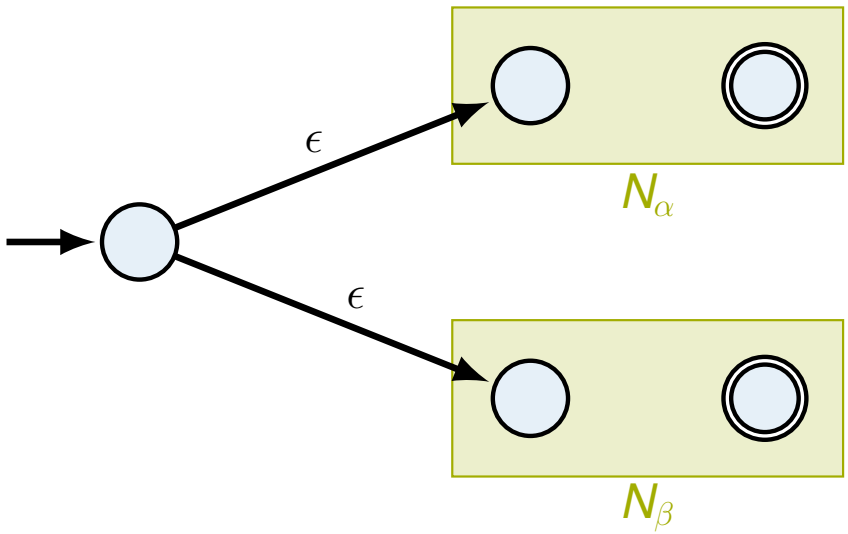
$\gamma = a \in \Sigma$



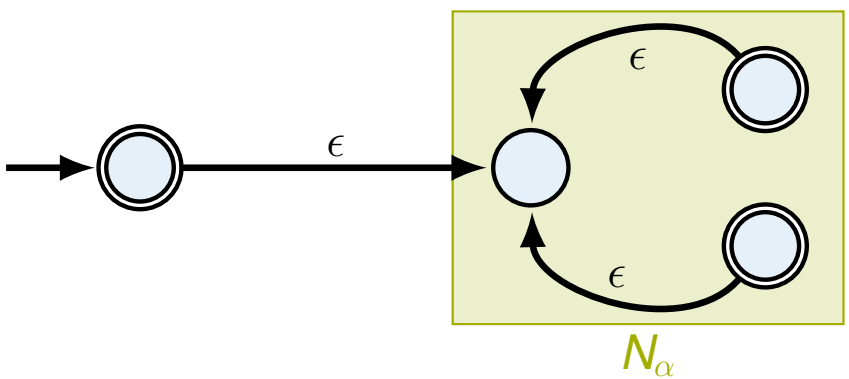
$\gamma = \alpha\beta$



$\gamma = \alpha \mid \beta$



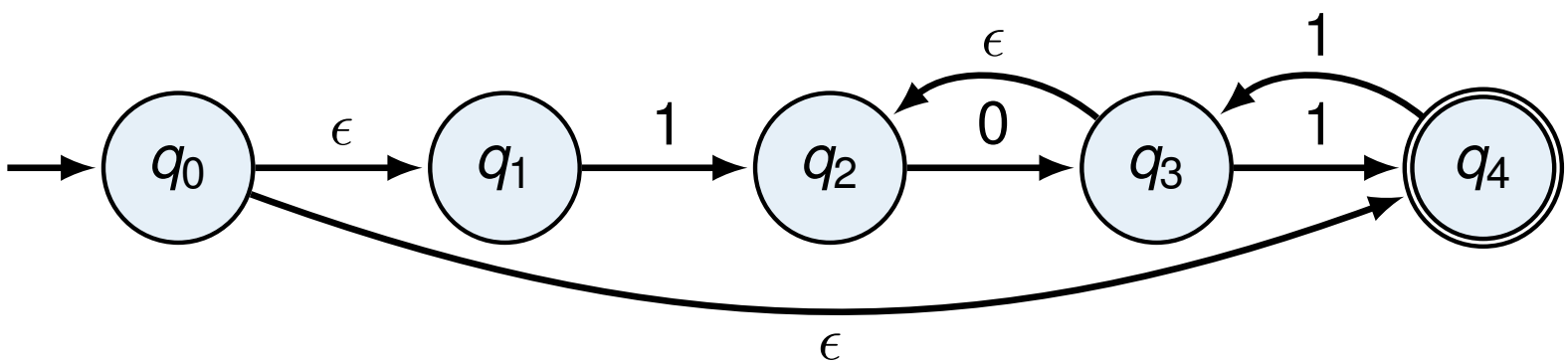
$\gamma = \alpha^*$



Idee

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

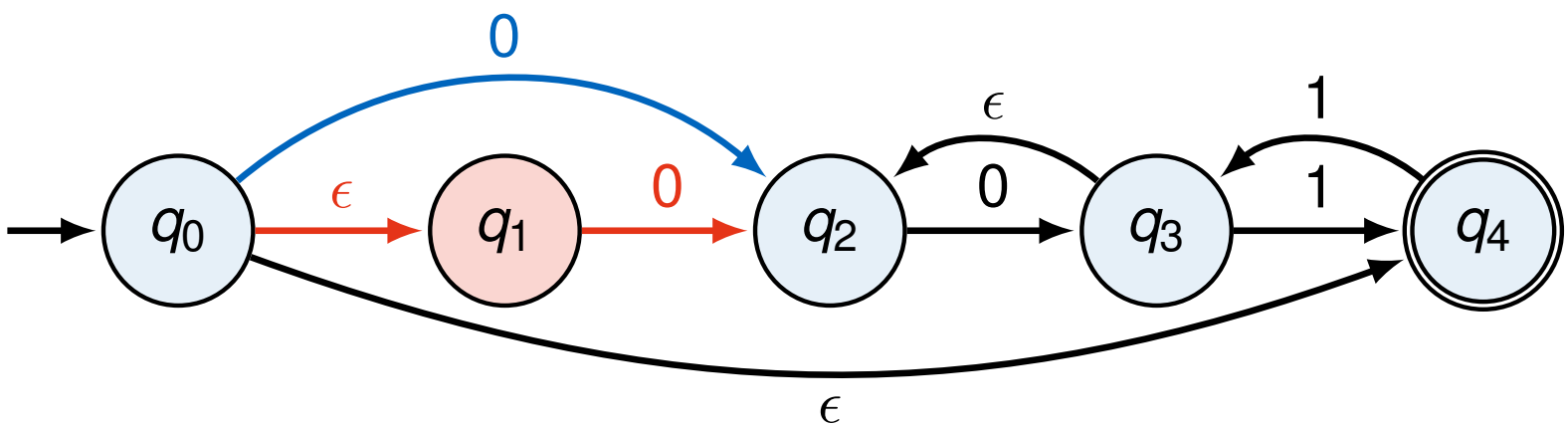
1. Entferne unnötige Knoten.
2. Für jeden Pfad der Form $\epsilon \ldots \epsilon a \epsilon \ldots \epsilon$ verbinde Anfangs- und Endknoten mit einer a-Kante.
3. Entferne alle ϵ-Kanten und unerreichbare Knoten.
4. Wurde das leere Wort akzeptiert mache den Anfangszustand zum Endzustand.



Idee

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

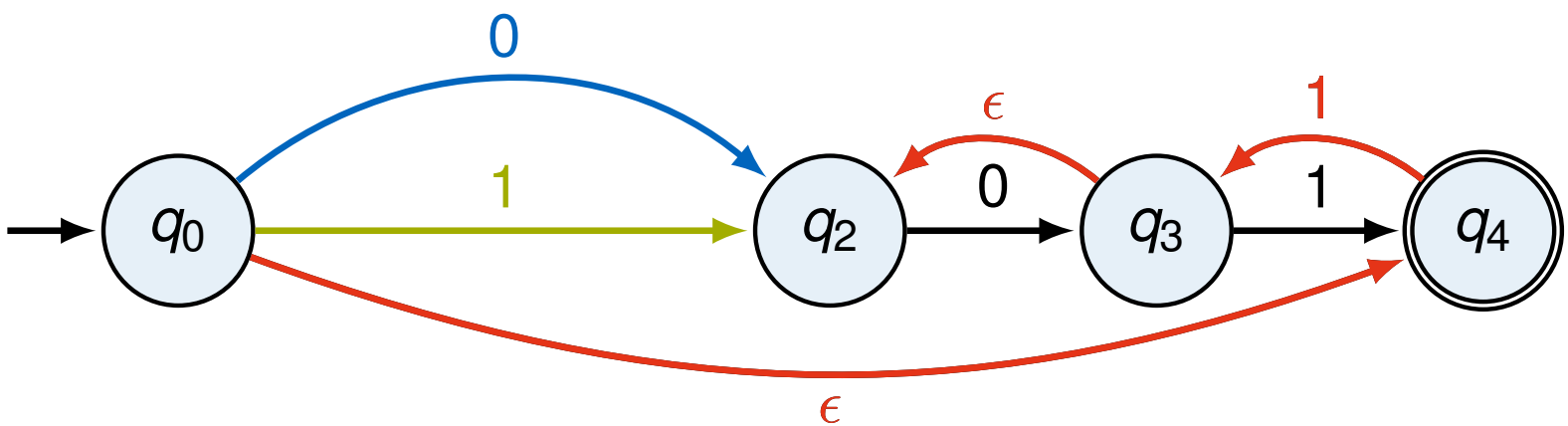
1. Entferne unnötige Knoten.
2. Für jeden Pfad der Form $\epsilon \ldots \epsilon a \epsilon \ldots \epsilon$ verbinde Anfangs- und Endknoten mit einer a-Kante.
3. Entferne alle ϵ-Kanten und unerreichbare Knoten.
4. Wurde das leere Wort akzeptiert mache den Anfangszustand zum Endzustand.



Idee

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

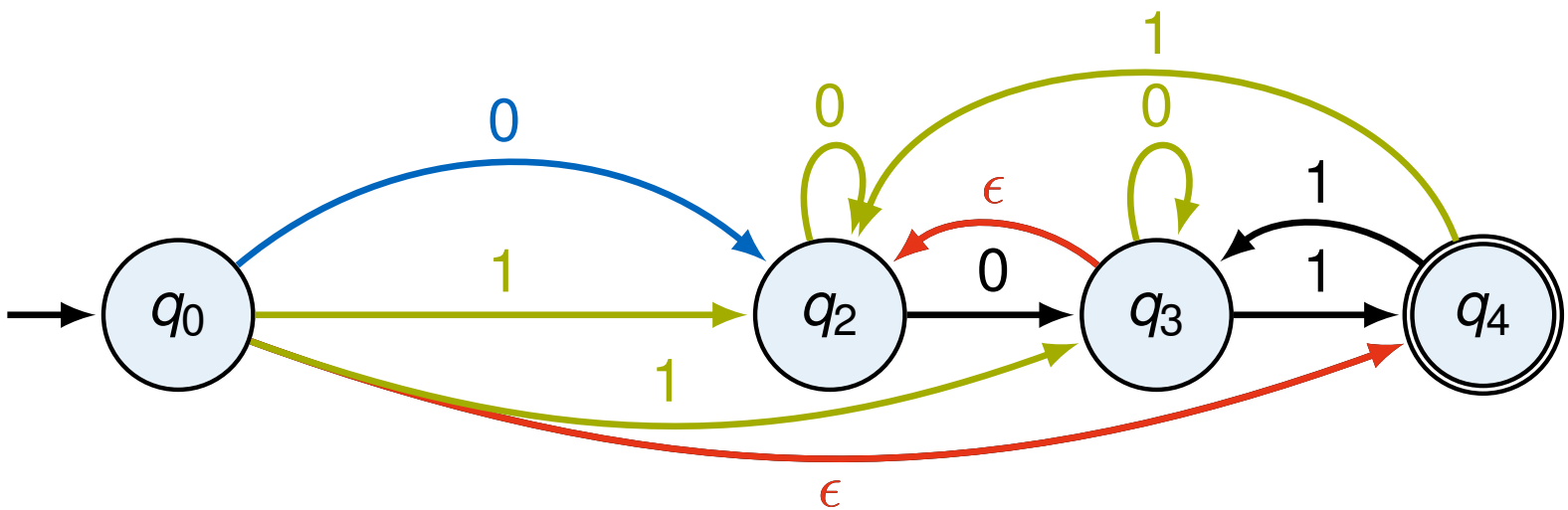
1. Entferne unnötige Knoten.
2. Für jeden Pfad der Form $\epsilon \ldots \epsilon a \epsilon \ldots \epsilon$ verbinde Anfangs- und Endknoten mit einer a-Kante.
3. Entferne alle ϵ-Kanten und unerreichbare Knoten.
4. Wurde das leere Wort akzeptiert mache den Anfangszustand zum Endzustand.



Idee

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

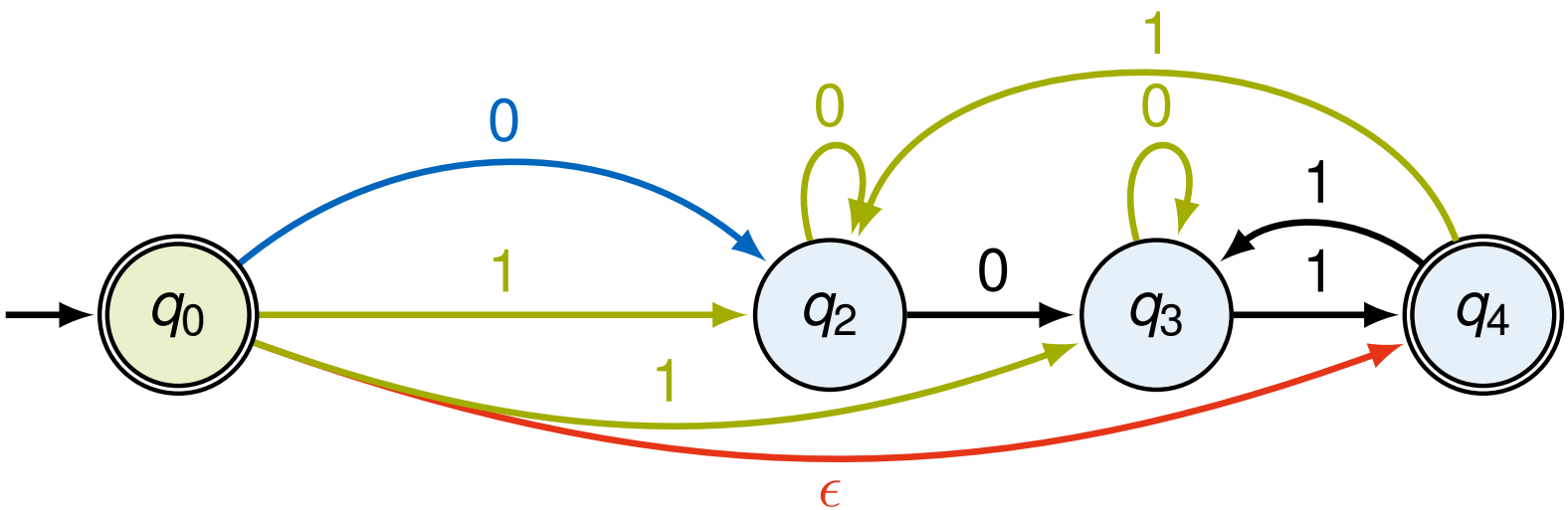
1. Entferne unnötige Knoten.
2. Für jeden Pfad der Form $\epsilon \ldots \epsilon a \epsilon \ldots \epsilon$ verbinde Anfangs- und Endknoten mit einer a-Kante.
3. Entferne alle ϵ-Kanten und unerreichbare Knoten.
4. Wurde das leere Wort akzeptiert mache den Anfangszustand zum Endzustand.



Idee

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

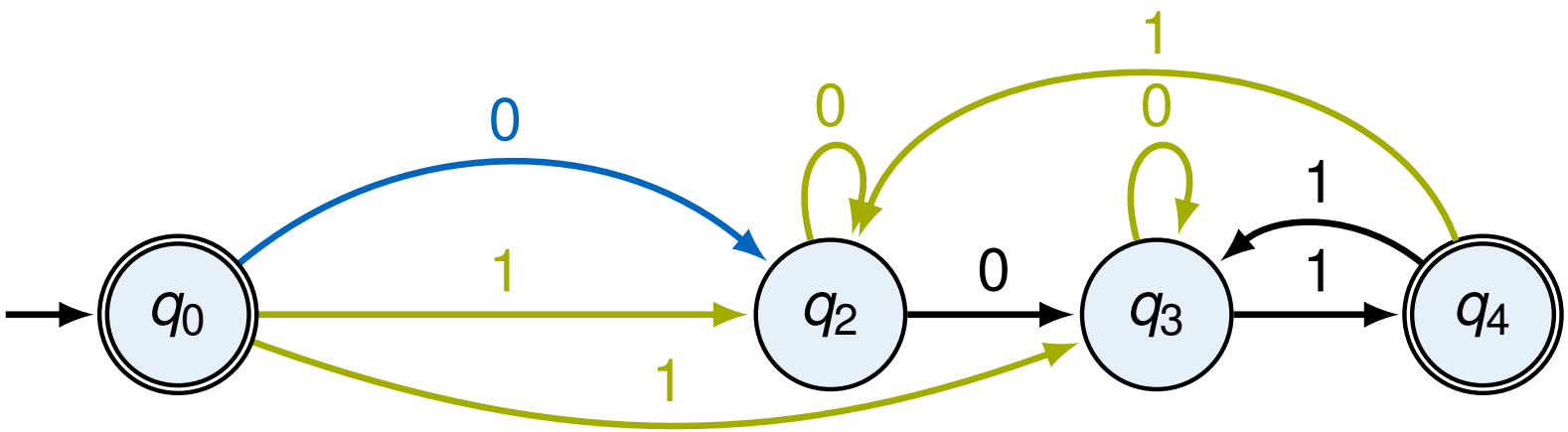
1. Entferne unnötige Knoten.
2. Für jeden Pfad der Form $\epsilon \ldots \epsilon a \epsilon \ldots \epsilon$ verbinde Anfangs- und Endknoten mit einer a-Kante.
3. Entferne alle ϵ-Kanten und unerreichbare Knoten.
4. Wurde das leere Wort akzeptiert mache den Anfangszustand zum Endzustand.



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.



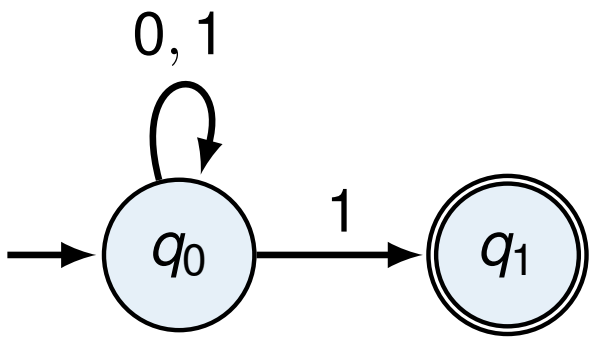
Idee (Potenzmengenkonstruktion)

Konstruiere aus einem NFA $N = (Q, \Sigma, \delta, q_0, F)$ einen DFA $D = (P(Q), \Sigma, \overline{\delta}, \{q_0\}, F_M)$ mit Zuständen aus $P(Q)$.

- $\overline{\delta} : P(Q) \times \Sigma \to P(Q)$

$$\overline{\delta}(S, a) := \bigcup_{q \in S} \delta(q, a)$$

- $F_M := \{S \subseteq Q \mid S \cap F \neq \emptyset\}$



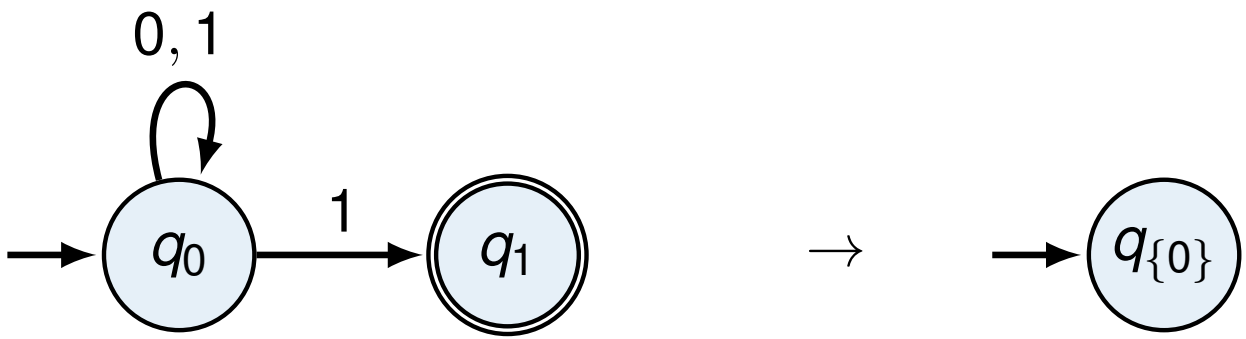
Idee (Potenzmengenkonstruktion)

Konstruiere aus einem NFA $N = (Q, \Sigma, \delta, q_0, F)$ einen DFA $D = (P(Q), \Sigma, \overline{\delta}, \{q_0\}, F_M)$ mit Zuständen aus $P(Q)$.

- $\overline{\delta} : P(Q) \times \Sigma \to P(Q)$

$$\overline{\delta}(S, a) := \bigcup_{q \in S} \delta(q, a)$$

- $F_M := \{S \subseteq Q \mid S \cap F \neq \emptyset\}$



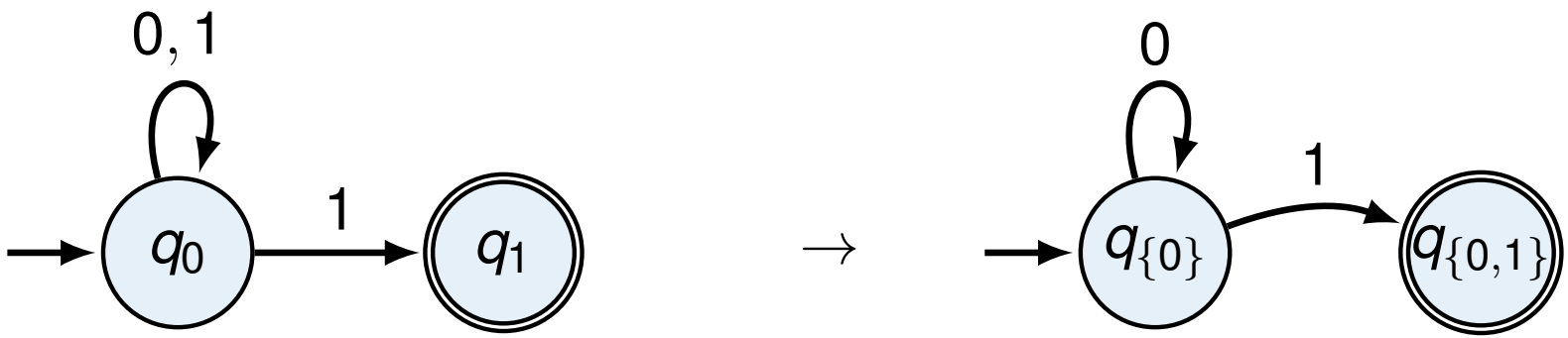
Idee (Potenzmengenkonstruktion)

Konstruiere aus einem NFA $N = (Q, \Sigma, \delta, q_0, F)$ einen DFA $D = (P(Q), \Sigma, \overline{\delta}, \{q_0\}, F_M)$ mit Zuständen aus $P(Q)$.

- $\overline{\delta} : P(Q) \times \Sigma \to P(Q)$

$$\overline{\delta}(S, a) := \bigcup_{q \in S} \delta(q, a)$$

- $F_M := \{S \subseteq Q \mid S \cap F \neq \emptyset\}$



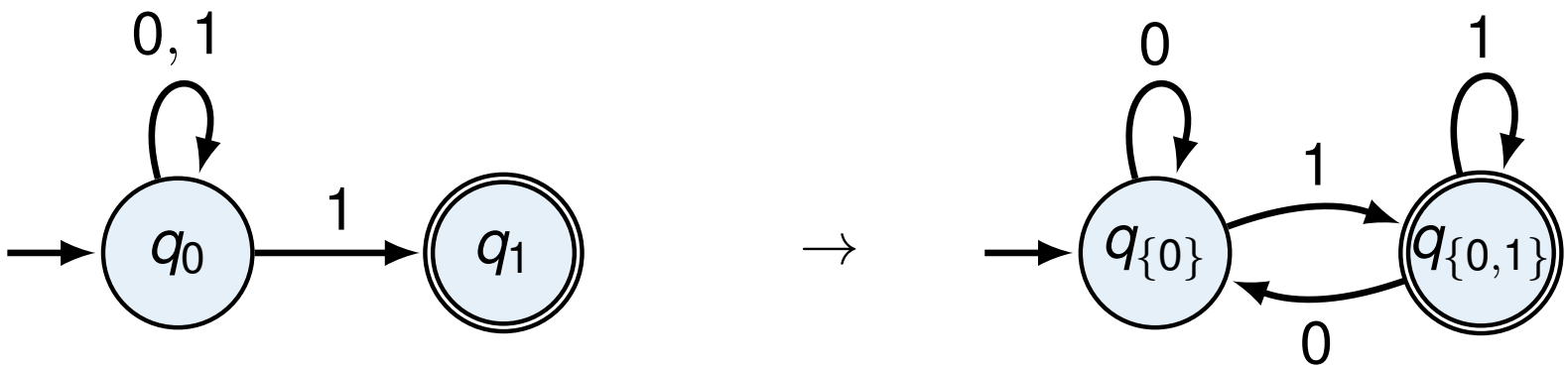
Idee (Potenzmengenkonstruktion)

Konstruiere aus einem NFA $N = (Q, \Sigma, \delta, q_0, F)$ einen DFA $D = (P(Q), \Sigma, \overline{\delta}, \{q_0\}, F_M)$ mit Zuständen aus $P(Q)$.

- $\overline{\delta} : P(Q) \times \Sigma \to P(Q)$

$$\overline{\delta}(S, a) := \bigcup_{q \in S} \delta(q, a)$$

- $F_M := \{S \subseteq Q \mid S \cap F \neq \emptyset\}$



Produktautomat

Satz

Sind $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ *und* $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ *DFAs, dann ist der Produkt-Automat*

$$M := (Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F_1 \times F_2)$$
$$\delta((q_1, q_2), a) := (\delta_1(q_1, a), \delta_2(q_2, a))$$

ein DFA, der $L(M_1) \cap L(M_2)$ *akzeptiert.*

Satz

Die regulären Ausdrücke $\mathfrak{R}$ über einem Alphabet Σ bilden mit Konkatenation $\circ$ und Veroderung $\mid$ einen Halbring $\langle \mathfrak{R}, \mid, \circ, \emptyset, \epsilon \rangle$.

- *Assoziative Operationen*
- *Veroderung kommutativ*
- *Distributivität:* $\alpha(\beta \mid \gamma) \equiv \alpha\beta \mid \alpha\gamma$
- $\emptyset$ *neutral bezüglich Oder*
- ϵ *neutral bezüglich Konkatenation*

Beispiel

$$1\psi \mid 0\phi \mid \psi \equiv 0\phi \mid (1 \mid \epsilon)\psi$$

Satz (Ardens Lemma)

Sind A, B und X Sprachen mit $\epsilon \notin A$, dann gilt

$$X = AX \cup B \Longrightarrow X = A^*B$$

Speziell gilt für reguläre Ausdrücke

$$X \equiv \alpha X \mid \beta \Longrightarrow X \equiv \alpha^* \beta$$

Beispiel

$$\psi \equiv 0\psi \mid (1 \mid \epsilon)\phi \Longrightarrow \psi \equiv 0^*(1 \mid \epsilon)\phi$$

Idee

Erzeuge ein Gleichungssystem aus allen Zuständen.

1. Ausdruck für jeden Zustand
2. Auflösen nach X_0 mit Algebra und Ardens Lemma

$$
\begin{aligned}
X_0 &\equiv 1X_0 \mid 0X_1 \\
&\equiv 1X_0 \mid 00^*(\epsilon \mid 1X_0) \\
&\equiv (1 \mid 00^*1)X_0 \mid 00^* \\
&\equiv (1 \mid 00^*1)^*(00^*)
\end{aligned}
$$

$$
\begin{aligned}
X_1 &\equiv 1X_0 \mid 0X_1 \mid \epsilon \\
&\equiv 0X_1 \mid (\epsilon \mid 1X_0) \\
&\equiv 0^*(\epsilon \mid 1X_0)
\end{aligned}
$$

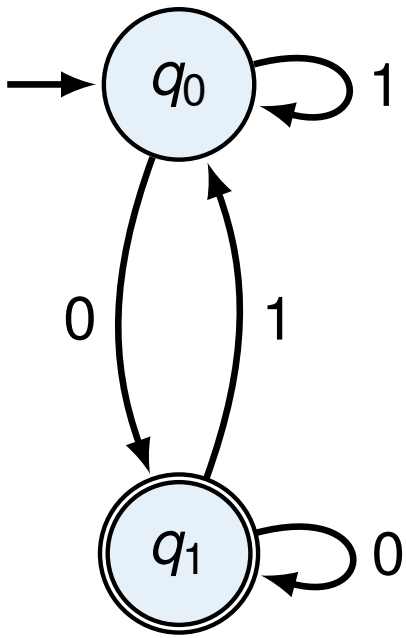
Idee

Erzeuge ein Gleichungssystem aus allen Zuständen.

1. Ausdruck für jeden Zustand
2. Auflösen nach X_0 mit Algebra und Ardens Lemma

$$
\begin{aligned}
X_0 &\equiv 1X_0 \mid 0X_1 \\
&\equiv 1X_0 \mid 00^*(\epsilon \mid 1X_0) \\
&\equiv (1 \mid 00^*1)X_0 \mid 00^* \\
&\equiv (1 \mid 00^*1)^*(00^*)
\end{aligned}
$$

$$
\begin{aligned}
X_1 &\equiv 1X_0 \mid 0X_1 \mid \epsilon \\
&\equiv 0X_1 \mid (\epsilon \mid 1X_0) \\
&\equiv 0^*(\epsilon \mid 1X_0)
\end{aligned}
$$



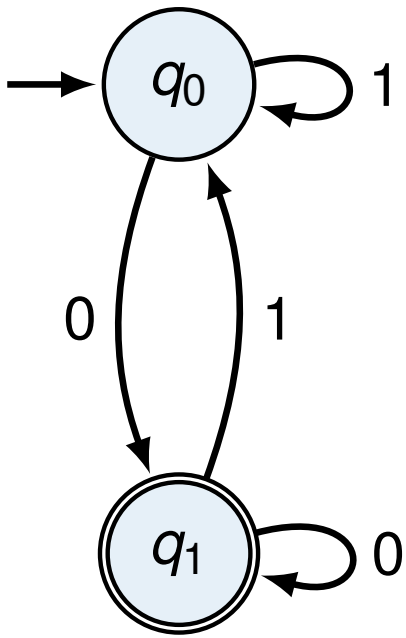
Idee

Erzeuge ein Gleichungssystem aus allen Zuständen.

1. Ausdruck für jeden Zustand
2. Auflösen nach X_0 mit Algebra und Ardens Lemma

$$
\begin{aligned}
X_0 &\equiv 1X_0 \mid 0X_1 \\
&\equiv 1X_0 \mid 00^*(\epsilon \mid 1X_0) \\
&\equiv (1 \mid 00^*1)X_0 \mid 00^* \\
&\equiv (1 \mid 00^*1)^*(00^*)
\end{aligned}
$$

$$
\begin{aligned}
X_1 &\equiv 1X_0 \mid 0X_1 \mid \epsilon \\
&\equiv 0X_1 \mid (\epsilon \mid 1X_0) \\
&\equiv 0^*(\epsilon \mid 1X_0)
\end{aligned}
$$



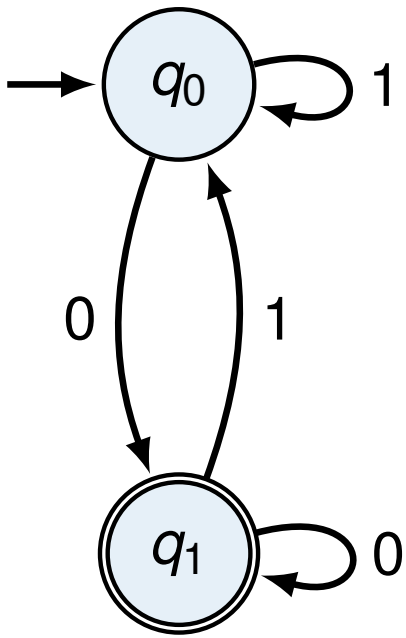
Idee

Erzeuge ein Gleichungssystem aus allen Zuständen.

1. Ausdruck für jeden Zustand
2. Auflösen nach X_0 mit Algebra und Ardens Lemma

$$
\begin{aligned}
X_0 &\equiv 1X_0 \mid 0X_1 \\
&\equiv 1X_0 \mid 00^*(\epsilon \mid 1X_0) \\
&\equiv (1 \mid 00^*1)X_0 \mid 00^* \\
&\equiv (1 \mid 00^*1)^*(00^*)
\end{aligned}
$$

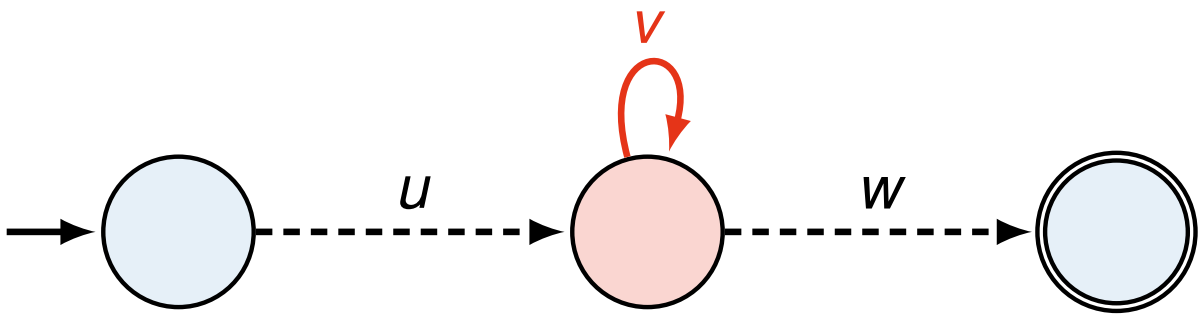
$$
\begin{aligned}
X_1 &\equiv 1X_0 \mid 0X_1 \mid \epsilon \\
&\equiv 0X_1 \mid (\epsilon \mid 1X_0) \\
&\equiv 0^*(\epsilon \mid 1X_0)
\end{aligned}
$$



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

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



Nichtregularität beweisen

Idee

Gegenbeispiel fürs Pumpinglemma suchen.

$$\forall n \in \mathbb{N}_0 \exists z \in L. |z| \geq n \, \forall u, v, w. \, z = uvw \text{ nicht pumpbar}$$

Beispiel

Ist $L = \{a^i b^i \mid i \in \mathbb{N}_0\}$ regulär?

1. Sei n PL-Zahl
2. Wähle $z = a^n b^n$
3. Dann ist $z = uvw$ mit $|uv| \leq n$, hier: $v = a^k$ mit $k > 0$
4. Dann ist $uv^0 w \notin L$
5. Damit ist L nicht regulär.

Nichtregularität beweisen

Idee

Gegenbeispiel fürs Pumpinglemma suchen.

$$\forall n \in \mathbb{N}_0 \exists z \in L. |z| \geq n \, \forall u, v, w.\ z = uvw \text{ nicht pumpbar}$$

Beispiel

Ist $L = \{a^i b^i \mid i \in \mathbb{N}_0\}$ regulär?

1. Sei n PL-Zahl
2. Wähle $z = a^n b^n$
3. Dann ist $z = uvw$ mit $|uv| \leq n$, hier: $v = a^k$ mit $k > 0$
4. Dann ist $uv^0 w \notin L$
5. Damit ist L nicht regulär.

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.

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

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.

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

Idee

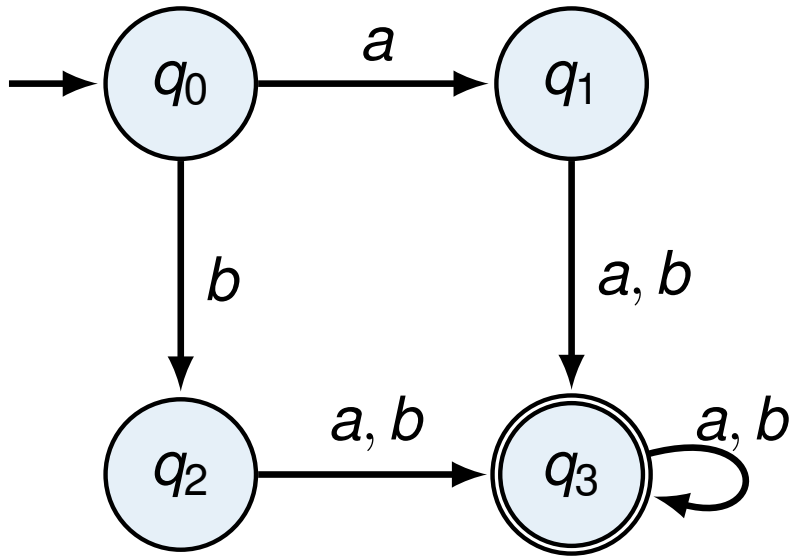
Erzeuge den Quotientenautomaten.

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

Idee

Erzeuge den Quotientenautomaten.

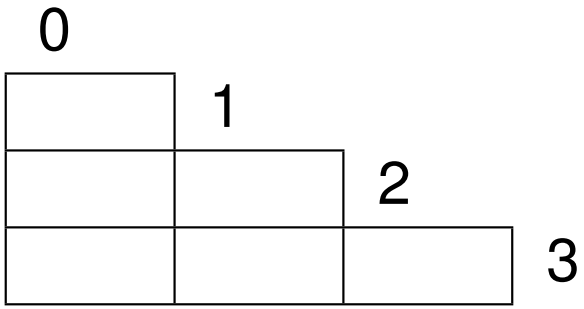
1. Entferne alle von q_0 nicht erreichbaren Zustände
2. Berechne die unterscheidbaren Zustände
3. Kollabiere die äquivalenten Zustände

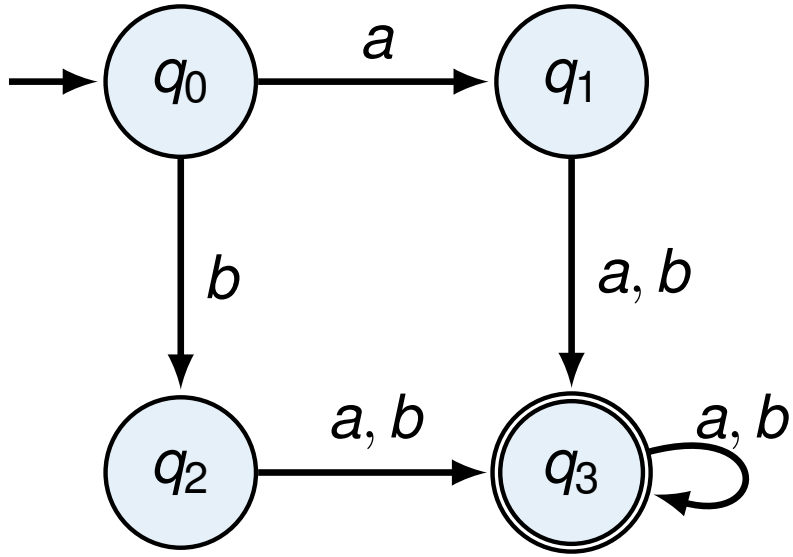


Idee

Erzeuge den Quotientenautomaten.

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

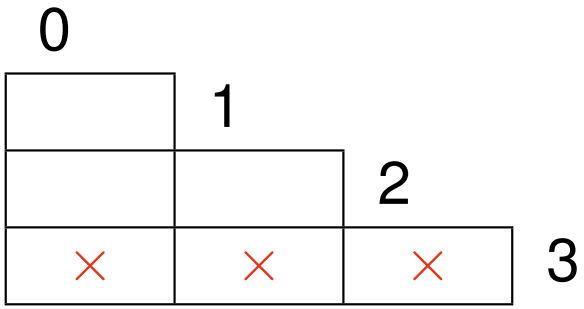


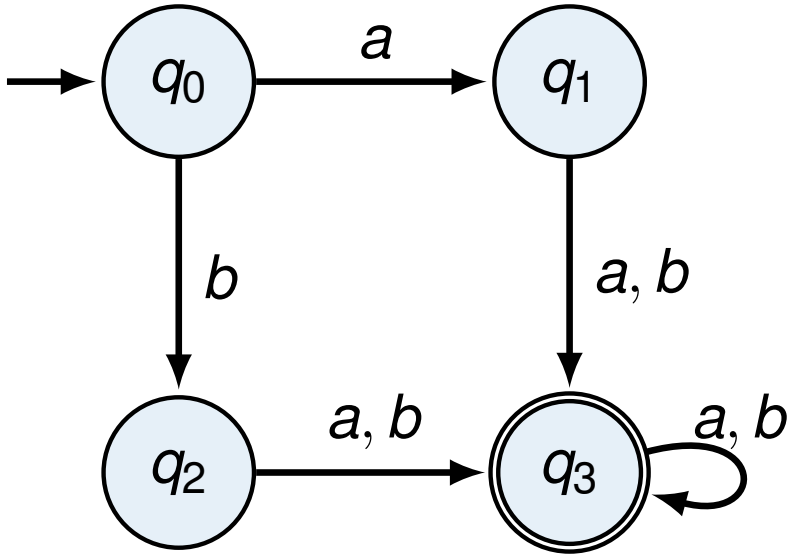


Idee

Erzeuge den Quotientenautomaten.

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

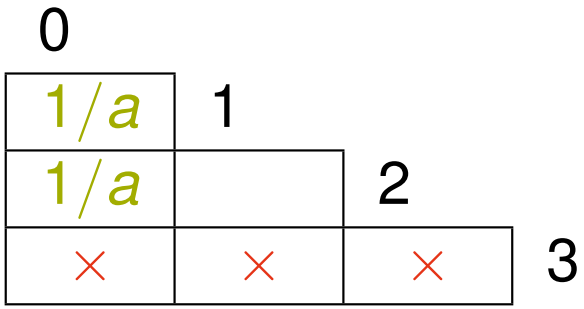


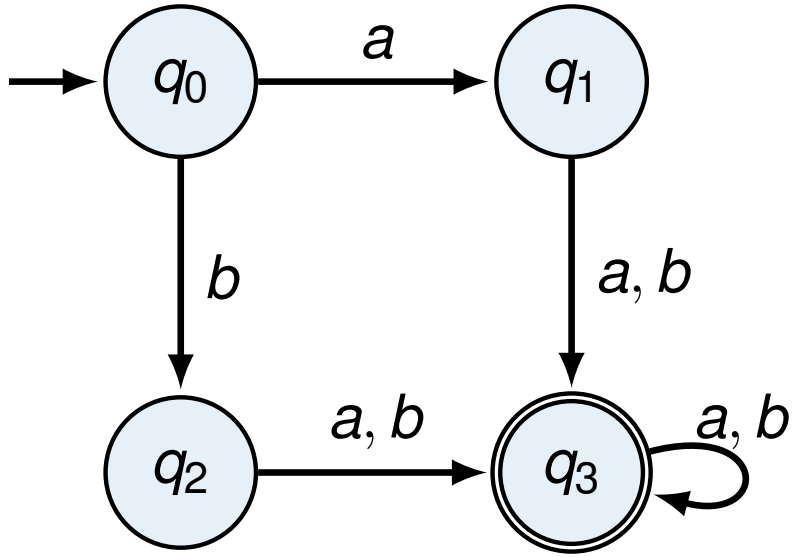


Idee

Erzeuge den Quotientenautomaten.

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

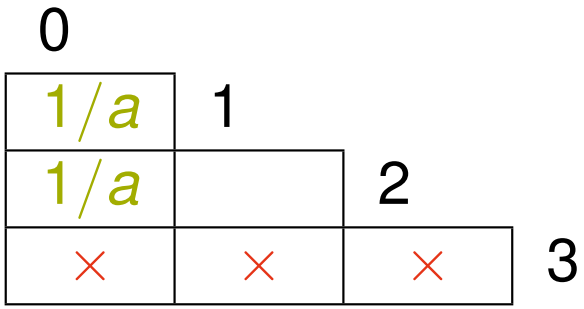


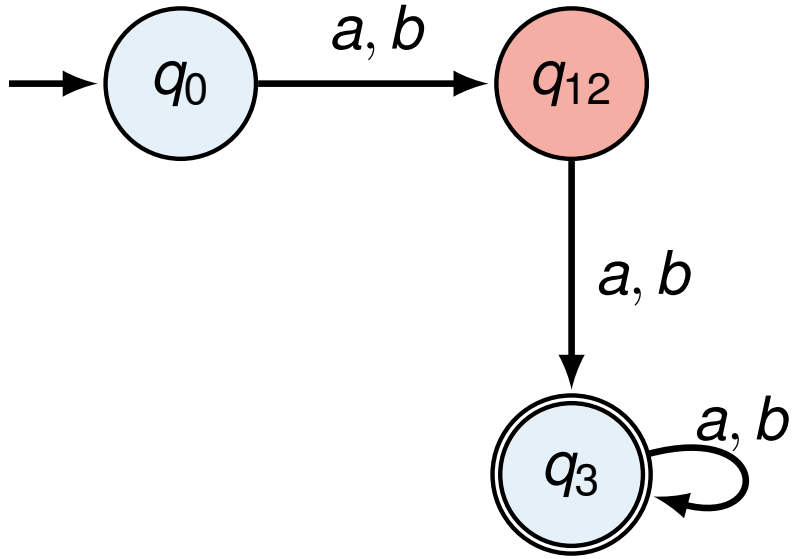


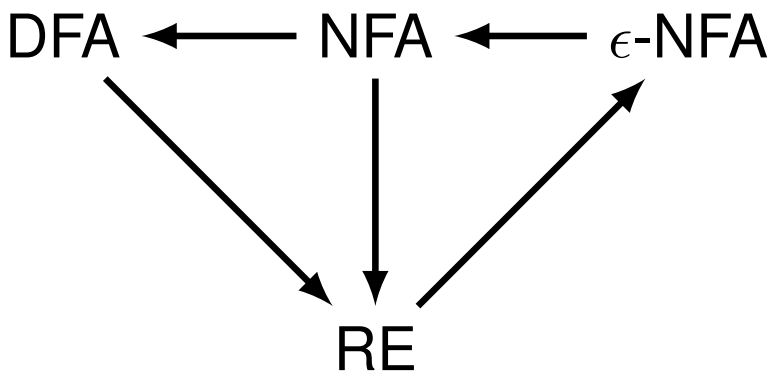
Idee

Erzeuge den Quotientenautomaten.

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







Satz

Für eine Darstellung D einer regulären Sprache ist entscheidbar:

Wortproblem *Gegeben w, gilt $w \in L(D)$?*

Leerheitsproblem *Ist $L(D) = \emptyset$?*

Endlichkeitsproblem *Ist $|L(D)| < \infty$?*

Äquivalenzproblem *Gilt $L(D_1) = L(D_2)$?*

Definition (Kontextfreie Grammatik)

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

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

Beispiel (Vorbereitung 3)

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

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

Definition (Kontextfreie Grammatik)

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

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

Beispiel (Vorbereitung 3)

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

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

Ableitungsrelation

Definition (Ableitungsrelation)

Eine CFG G induziert eine Ableitungsrelation $\to_G$ auf Wörtern über $V \cup \Sigma$:

$$\alpha \to_G \beta$$

gdw es eine Regel $A \to \gamma$ in P mit Wörtern α_1, α_2 gibt, so dass

$$\alpha = \alpha_1 A \alpha_2 \quad \text{und} \quad \beta = \alpha_1 \gamma \alpha_2$$

Beispiel (Vorbereitung 3)

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

$$S \to_G 0S1 \to_G 00S11 \to_G 00S1111 \to_G 0011111$$
$$\Rightarrow S \to_G^* 0011111$$

Kontextfreie Sprache

Definition (Kontextfreie Sprache)

Eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ erzeugt die Sprache

$$L(G) := \{w \in \Sigma^* \mid S \to_G^* w\}$$

Eine Sprache $L \subseteq \Sigma^*$ heißt kontextfrei gdw es eine kontextfreie Grammatik G gibt mit $L = L(G)$.

Induktive Sprachdefinition

Die induktive Definition ($\Longrightarrow$) erzeugt Wörter bottom-up, setzt also kleine Wörter zu größeren zusammen.

Beispiel (Vorbereitung 3)

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:

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

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

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

Idee

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

Idee

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.

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

neu:

$$S \to b$$

$$A \to aA \mid a$$

CNF Konstruktion

Idee

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.

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

neu:

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

Idee

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.

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

neu:

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

CNF Konstruktion

Idee

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.

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

neu:

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

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

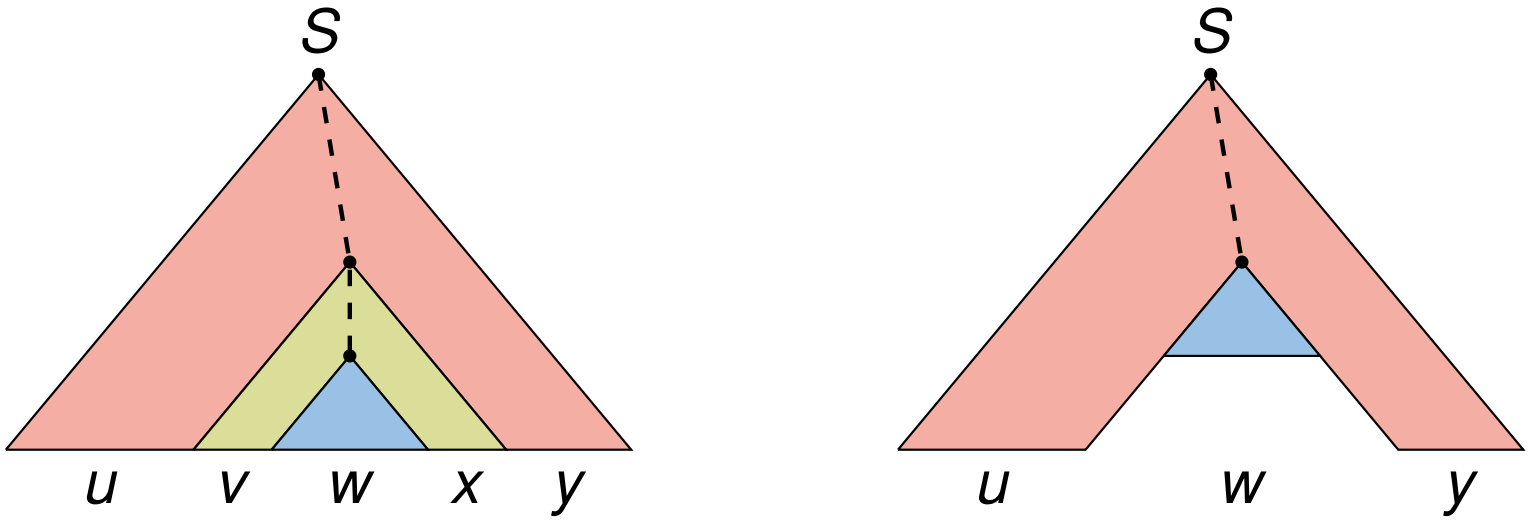
Nützliche Symbole sind erzeugend und erreichbar. Aber nicht notwendigerweise umgekehrt.

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

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

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



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

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

ist

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

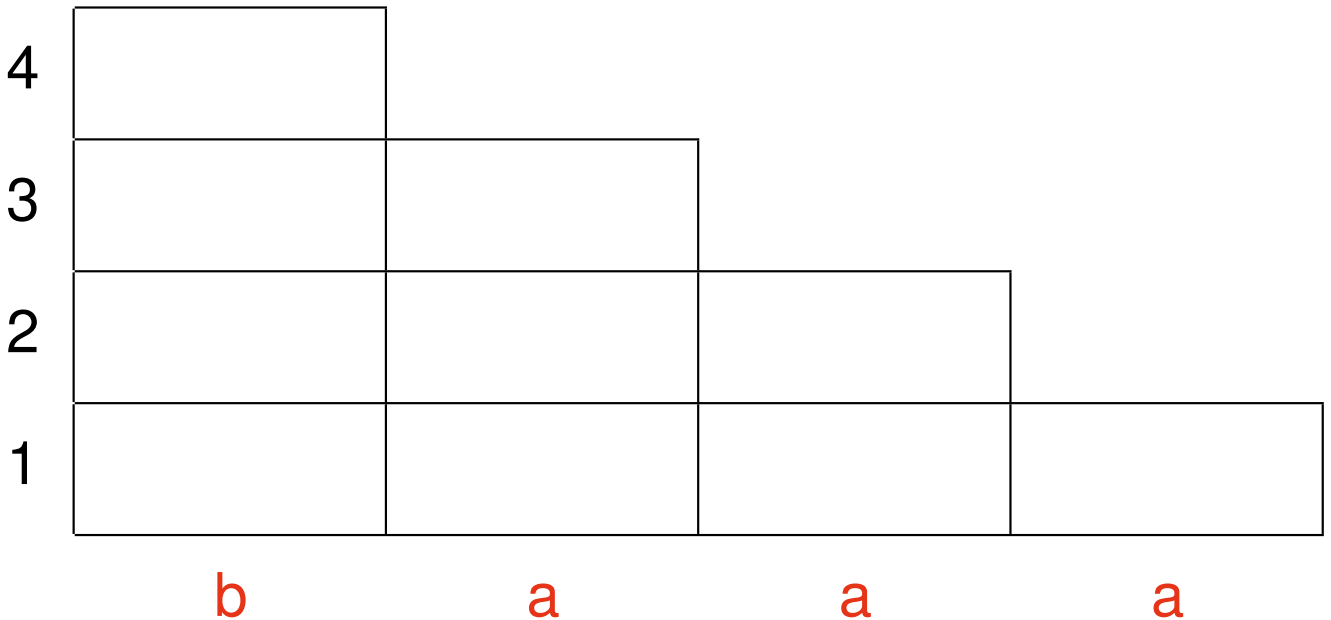
$$V_{ii} = \{A \in V \mid (A \to a_i) \in P\}$$
$$V_{ij} = \left\{A \in V \mid \exists k, B \in V_{ik}, C \in V_{k+1,j} \,.\, (A \to BC) \in P\right\}$$

Idee

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

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

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

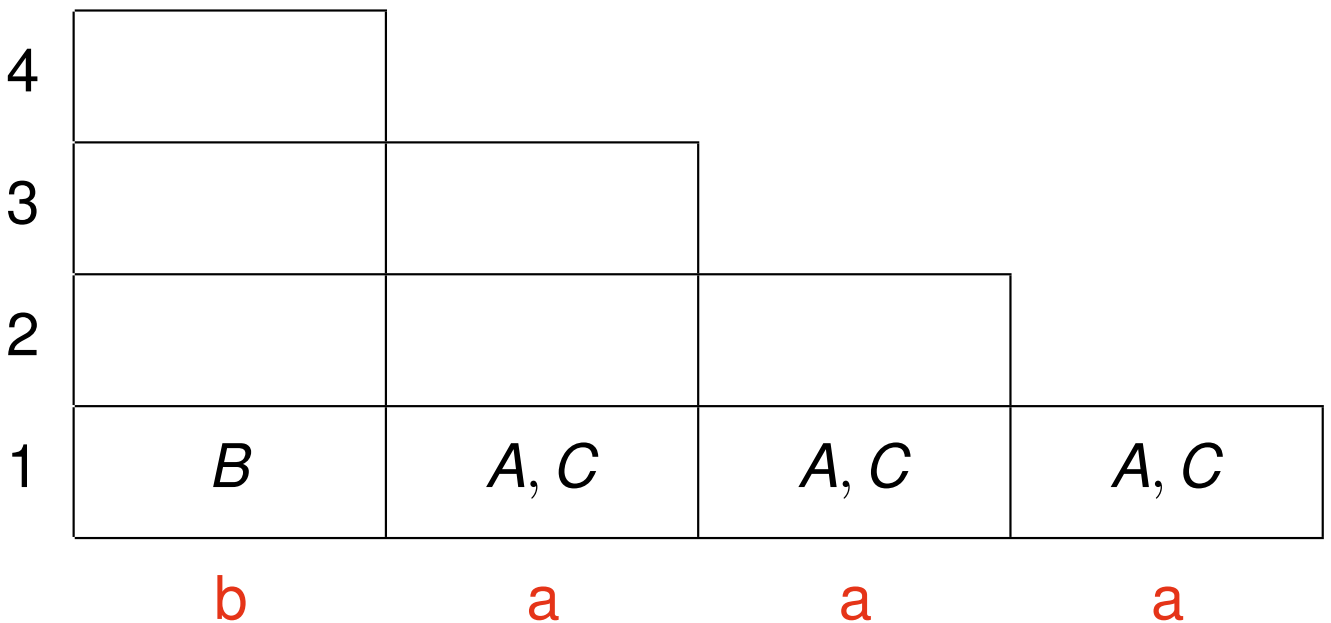


Idee

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

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

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



Idee

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				
2	A	B	B	
1	B	A, C	A, C	A, C
	b	a	a	a

Idee

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	B	B	
1	B	A, C	A, C	A, C
	b	a	a	a

Idee

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

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

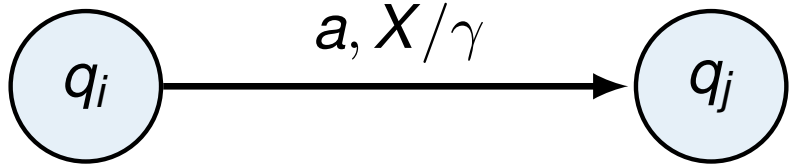
$S \to AB \mid BC, \quad A \to BA \mid a, \quad B \to CC \mid b, \quad C \to AB \mid a$

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

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

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

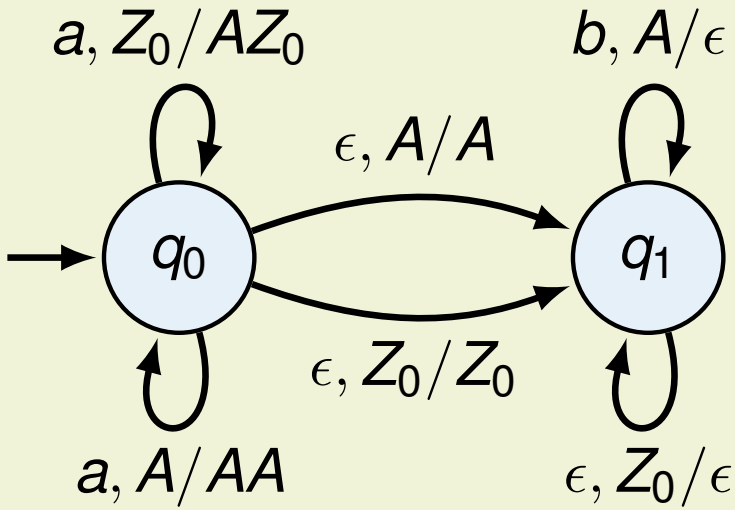
Definition (Kellerautomat)

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 $L = \{a^n b^n \mid n \in \mathbb{N}\}$.



Kontextfreie Sprachen

- Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
REG	ja	ja	ja	ja	ja
CFL	nein	ja	nein	ja	ja

- Entscheidbarkeit

	Wortproblem	Leerheit	Äquivalenz	Schnittproblem
DFA	$\mathcal{O}(n)$	ja	ja	ja
CFG	$\mathcal{O}(n^3)$	ja	nein	nein

Definition (Turingmaschine)

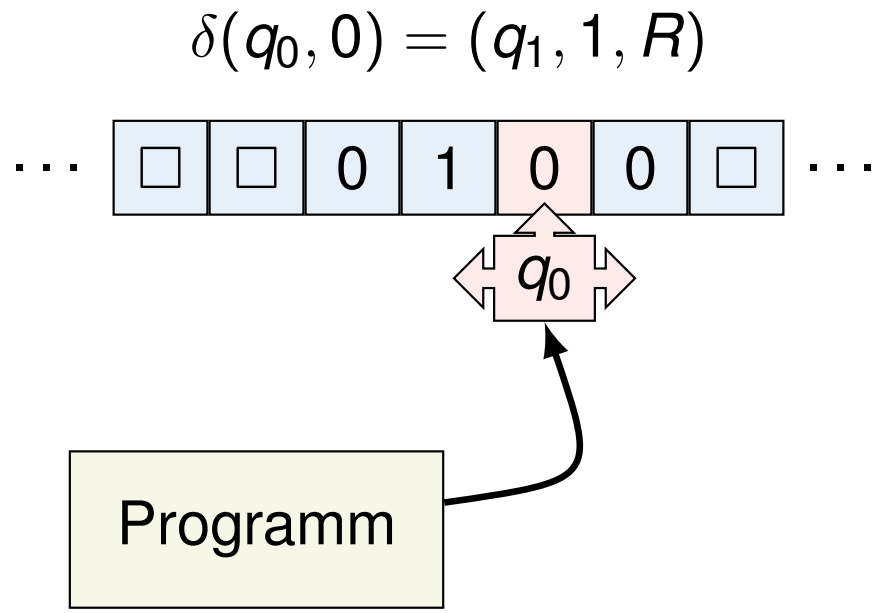
Eine deterministische Turingmaschine (TM) ist ein Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ aus einer/einem

- endlichen Menge von Zuständen Q
- endlichen Eingabealphabet Σ
- endlichen Bandalphabet Γ mit $\Sigma \subset \Gamma$
- Übergangsfunktion $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$
- Startzustand $q_0 \in Q$
- Leerzeichen $\square \in \Gamma \setminus \Sigma$
- Menge von Endzuständen $F \subseteq Q$

Definition (Turingmaschine)

Eine deterministische Turingmaschine (TM) ist ein Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ aus einer/einem

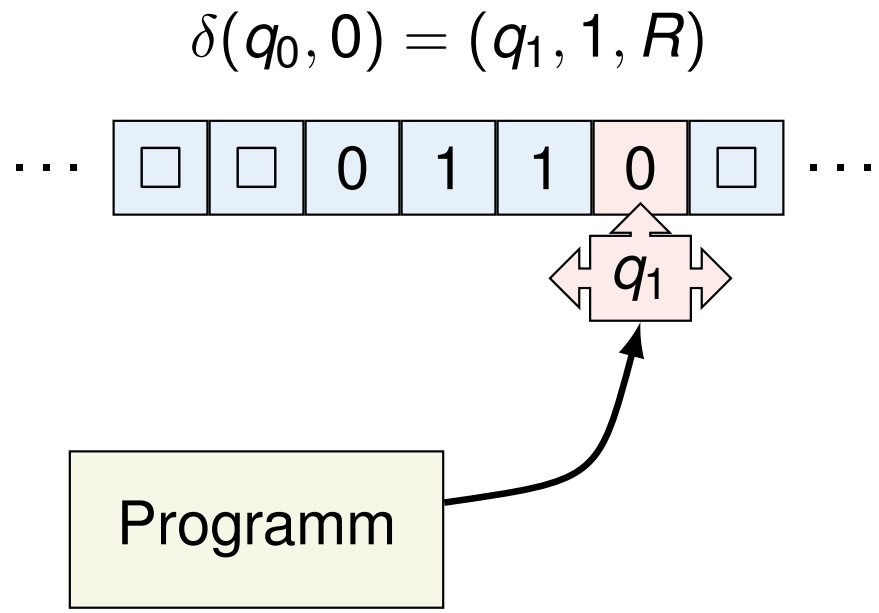
- Übergangsfunktion $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$



Definition (Turingmaschine)

Eine deterministische Turingmaschine (TM) ist ein Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ aus einer/einem

- Übergangsfunktion $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$



Definition (Nichtdeterministische Turingmaschine)

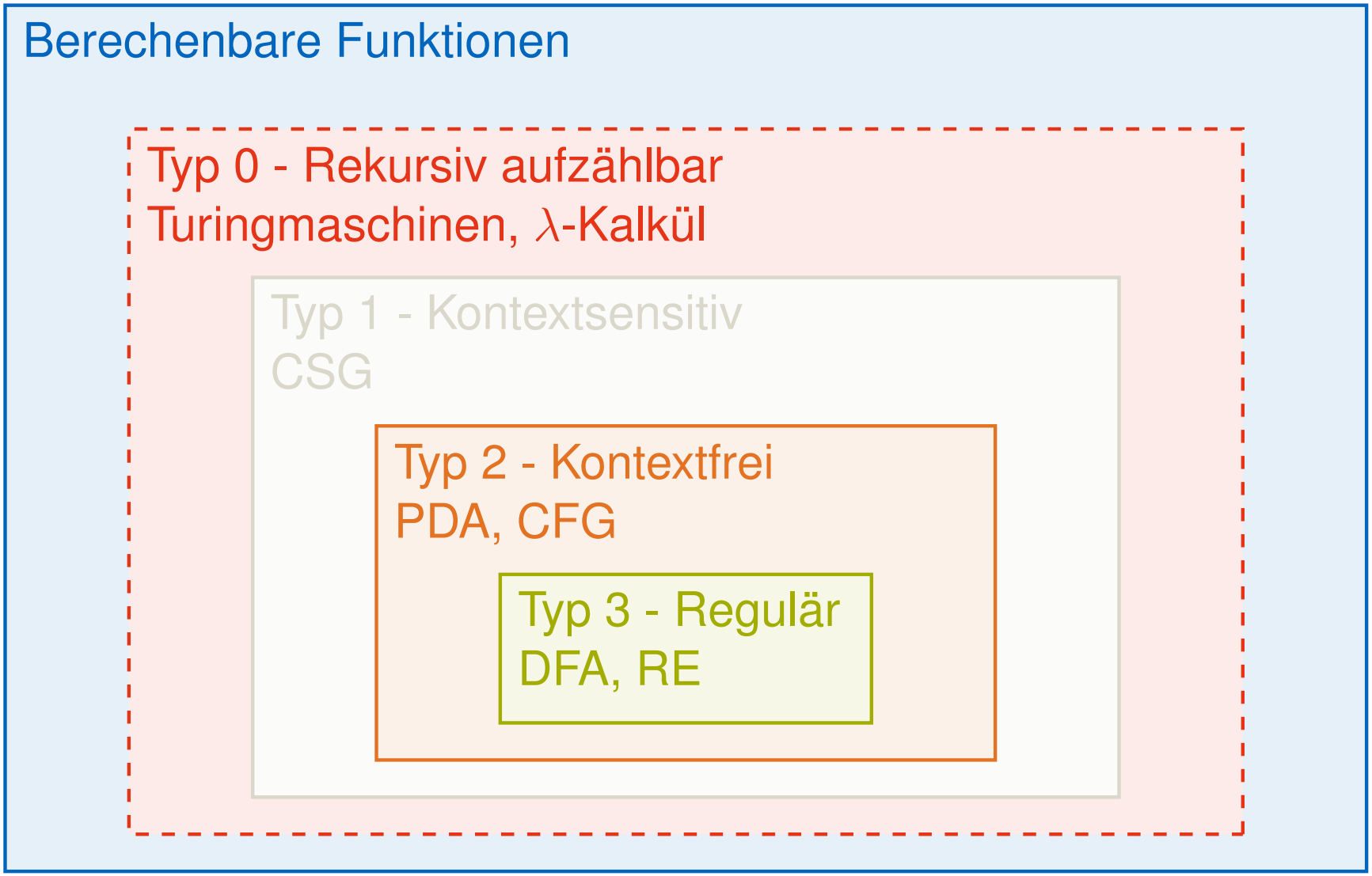
Eine nichtdeterministische Turingmaschine (TM) ist ein Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ aus einer/einem

- ...
- Übergangsfunktion $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R, N\})$
- ...

Satz

Zu jeder nichtdeterministischen TM N gibt es eine deterministische TM M mit $L(N) = L(M)$.

Chomsky-Hierarchie



Definition (LOOP-Programm)

Syntax von LOOP-Programmen.
Es ist $X \in \{x_0, x_1, \ldots\}$ und $C \in \mathbb{N}$.

$$
\begin{aligned}
P \to\ & X := X + C \\
\mid\ & X := X - C \\
\mid\ & P; P \\
\mid\ & \textbf{LOOP } X \textbf{ DO } P \textbf{ END} \\
\mid\ & \textbf{IF } X = 0 \textbf{ DO } P \textbf{ ELSE } Q \textbf{ END}
\end{aligned}
$$

- Ausgabe steht in x_0, Eingaben in $x_1, \ldots, x_n$, Rest ist 0.
- **LOOP** x_i **DO** P **END** führt P genau n mal aus, wobei n der Anfangswert von x_i ist. Zuweisungen an x_i in P ändern die Anzahl der Durchläufe nicht.

Definition (Basisfunktionen)

Primitiv Rekursiv sind:

- Die konstante Funktion 0
- Die Nachfolgerfunktion $s(n) = n + 1$
- Die Projektionsfunktion $\pi_i^k : \mathbb{N}^k \to \mathbb{N}, i \in [k]$

$$\pi_i^k(x_1, \ldots, x_k) = x_i$$

Definition (Komposition)

Sind g und h_i PR und $\bar{x} = (x_1, \ldots, x_n)$, dann ist auch f PR:

$$f(\bar{x}) = g(h_1(\bar{x}), \ldots, h_k(\bar{x}))$$

Basisfunktionen und Komposition

Schon PR sind:

- Konstante: 0
- Nachfolger: $s(n) = n + 1$
- Projektion: $\pi_i^k : \mathbb{N}^k \to \mathbb{N}$
- Komposition: $f(\bar{x}) = g(h_1(\bar{x}), \ldots, h_k(\bar{x}))$

Definition (Primitive Rekursion)

Das Schema der primitiven Rekursion erzeugt aus g und h die Funktion f:

$$f(0, \bar{x}) = g(\bar{x})$$
$$f(m + 1, \bar{x}) = h(f(m, \bar{x}), m, \bar{x})$$

U.a. diese Programme sind laut Vorlesung oder Übung PR:

- $pred(x) = \max\{0, x - 1\}$
- $add(x, y) = x + y$
- $x \dot{-} y = \max\{0, x - y\}$
- $mult(x, y) = x \cdot y$
- $div(x, y) = x \div y$ (Ganzzahldivision)
- Die restliche einfache Arithmetik...

- $tower(n) = 2^{2^{2^{\cdot^{\cdot^{\cdot}}}}}$ mit $tower(4) = 2^{16}$
- $sqr(x) = x^2$, $sqrt(x) = \sqrt{x}$
- $c(x), p_1(x), p_2(x)$ (Cantorsche Paarungsfunktion)
- $ifthen(n, a, b) = \begin{cases} a & n \neq 0 \\ b & n = 0 \end{cases}$

Erweitertes PR-Schema

Definition (Erweitertes PR-Schema)

Das erweiterte Schema der primitiven Rekursion erlaubt

$$f(0, \bar{x}) = t_0$$
$$f(m+1, \bar{x}) = t$$

wobei

- t_0 enthält nur PR-Funktionen und die x_i
- t enthält nur $f(m, \bar{x})$, PR Funktionen, m und die x_i.

Satz

Das erweiterte Schema der primitiven Rekursion führt nicht aus PR heraus.

Beschränkte Operationen

Definition

Ein Prädikat P ist PR, wenn es eine PR Funktion $\hat{P}$ gibt mit

$$\hat{P}(x) = 1 \iff P(x)$$

Definition (Beschränkte Operationen)

Ist P PR, dann auch

- der beschränkte max-Operator

$$\max\{x \leq n \mid P(x)\}, \quad \max\{\emptyset\} = 0$$

- der beschränkte Existenzquantor

$$\exists x \leq n.P(x)$$

Definition (μ-Operator)

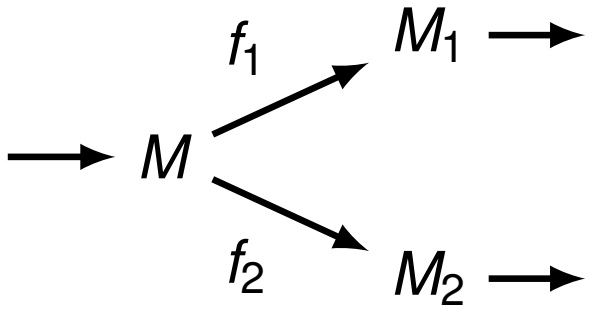
Sei $f : \mathbb{N}^{k+1} \to \mathbb{N}$ eine Funktion.
Der μ-Operator definiert eine neue Funktion $\mu f : \mathbb{N}^k \to \mathbb{N}$:

$$(\mu f)(\bar{x}) := \begin{cases} \min\{n \in \mathbb{N} \mid f(n, \bar{x}) = 0\} & \text{falls } n \text{ existent}^* \\ \bot & \text{sonst} \end{cases}$$

- *Für alle $m \leq n$ muss f definiert sein: $f(m, \bar{x}) \neq \bot$
- PR + μ = μ-Rekursion
- In Pseudocode:

$$\mu f(\bar{x}) = \mathit{find}(0, \bar{x})$$
$$\mathit{find}(n, \bar{x}) = \textbf{if } f(n, \bar{x}) = 0 \textbf{ then } n \textbf{ else } \mathit{find}(n+1, \bar{x})$$

Sind f_1 und f_2 Endzustände von M, so bezeichnet



eine Fallunterscheidung.

Beispiel (Band=0?)

$$\delta(q_0, 0) = (q_0, 0, R)$$
$$\delta(q_0, \square) = (ja, \square, L)$$
$$\delta(q_0, a) = (nein, a, N) \qquad \text{für } a \neq 0, \square$$

Definition (WHILE-Programm)

Syntax von WHILE-Programmen.
Es ist $X \in \{x_0, x_1, \ldots\}$ und $C \in \mathbb{N}$.

$$
\begin{aligned}
P \rightarrow\ & X := X + C \\
\mid\ & X := X - C \\
\mid\ & P; P \\
\mid\ & \textbf{WHILE } X \neq 0 \textbf{ DO } P \textbf{ END} \\
\mid\ & \textbf{LOOP } X \textbf{ DO } P \textbf{ END} \\
\mid\ & \textbf{IF } X = 0 \textbf{ DO } P \textbf{ ELSE } Q \textbf{ END}
\end{aligned}
$$

- Ausgabe steht in x_0, Eingaben in $x_1, \ldots, x_n$, Rest ist 0.
- Semantik wie erwartet.

Definition (GOTO-Programm)

Syntax von GOTO-Programmen.
Es ist $X \in \{x_0, x_1, \ldots\}$ und $C \in \mathbb{N}$.
Alle Anweisungen haben eine Markierung $M_1 : A_1; M_2 : A_2$.

$$
\begin{aligned}
P \rightarrow\ & X := X + C \\
\mid\ & X := X - C \\
\mid\ & P; P \\
\mid\ & \textbf{GOTO}\ M_i \\
\mid\ & \textbf{IF}\ X = 0\ \textbf{GOTO}\ M_i \\
\mid\ & \textbf{HALT}
\end{aligned}
$$

- Ausgabe steht in x_0, Eingaben in $x_1, \ldots, x_n$, Rest ist 0.

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.

Definition (Reduzierbarkeit)

Eine Menge $A \subseteq \Sigma^*$ ist reduzierbar auf eine Menge $B \subseteq \Gamma^*$ gdw es eine totale und berechenbare Funktion $f : \Sigma^* \to \Gamma^*$ gibt mit

$$\forall w \in \Sigma^*. w \in A \iff f(w) \in B$$

Wir schreiben dann $A \leq B$.

Intuition:

- B ist mindestens so schwer zu lösen wie A
- Ist A unlösbar, dann auch B.
- Ist B lösbar, dann erst recht A.

Spezielles Halteproblem

Definition (Spezielles Halteproblem)

Gegeben ein Wort $w \in \{0,1\}^*$.
Hält M_w bei Eingabe w?

$$K := \{w \mid M_w[w] \downarrow\}$$

Satz

Das spezielle Halteproblem ist nicht entscheidbar.

- Hält eine Turingmaschine mit sich selbst als Eingabe?
- w ist die Gödelisierung von M_w.
- K ist semi-entscheidbar, $\overline{K}$ nicht.

Allgemeines Halteproblem

Definition (Allgemeines Halteproblem)

Gegeben Wörter $w, x \in \{0,1\}^*$.
Hält M_w bei Eingabe x?

$$H := \{w\#x \mid M_w[x] \downarrow\}$$

Satz

Das allgemeine Halteproblem ist nicht entscheidbar.

- Es ist $K \leq H$. Warum?

Definition (Rekursiv aufzählbar)

Eine Menge A heißt rekursiv aufzählbar wenn $A = \emptyset$ oder es eine berechenbare totale Funktion $f : \mathbb{N} \to A$ gibt, so dass

$$A = \{f(0), f(1), \ldots\} = \bigcup_{n \in \mathbb{N}} \{f(n)\}$$

Äquivalent:

- A rekursiv aufzählbar
- A semi-entscheidbar, also χ'_A berechenbar
- $A = L(M)$ für eine TM M
- A ist Bild oder Urbild einer berechenbaren Funktion

Satz von Rice

Satz (Rice)

Sei F eine Menge berechenbarer Funktionen.
Sei weder $F = \emptyset$ noch $F =$ alle ber. Funktionen (F nicht trivial).
Dann ist unentscheidbar, ob die von einer gegebenen TM M_w berechnete Funktion in F ist, also ob $\varphi_w \in F$.

- Nicht-triviale semantische Eigenschaften von Programmen sind unentscheidbar.
- Termination ist unentscheidbar.

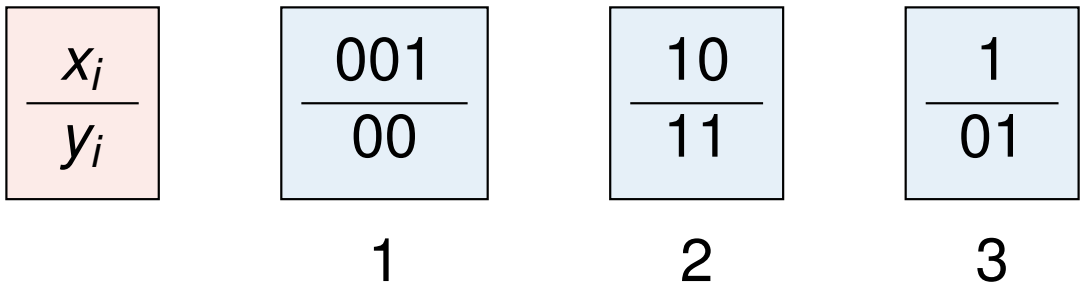
Rice-Shapiro:

- Termination ist nicht semi-entscheidbar.
- Nicht-Termination ist nicht semi-entscheidbar.

Definition (Postsches Korrespondenzproblem)

Gegeben endliche Folge $(x_1, y_1), \ldots, (x_k, y_k)$ mit $x_i, y_i \in \Sigma^+$.
Gibt es eine Folge von Indizes $i_1, \ldots, i_n \in \{1, \ldots, k\}$ mit

$$x_{i_1}, \ldots, x_{i_n} = y_{i_1}, \ldots, y_{i_n}$$



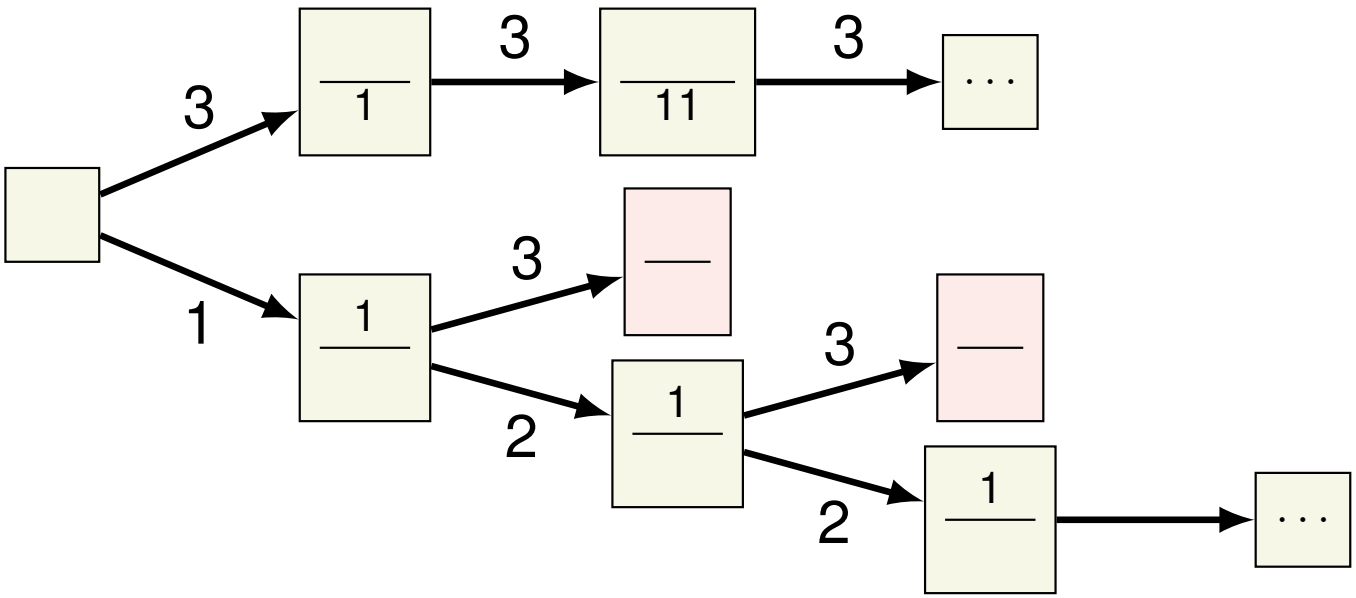
Satz

Das PCP ist unentscheidbar, aber semi-entscheidbar.

Idee

Mögliche Lösungen aufzählen, richtige Lösungen identifizieren

x_i / y_i	001 / 00	01 / 10	1 / 11
	1	2	3

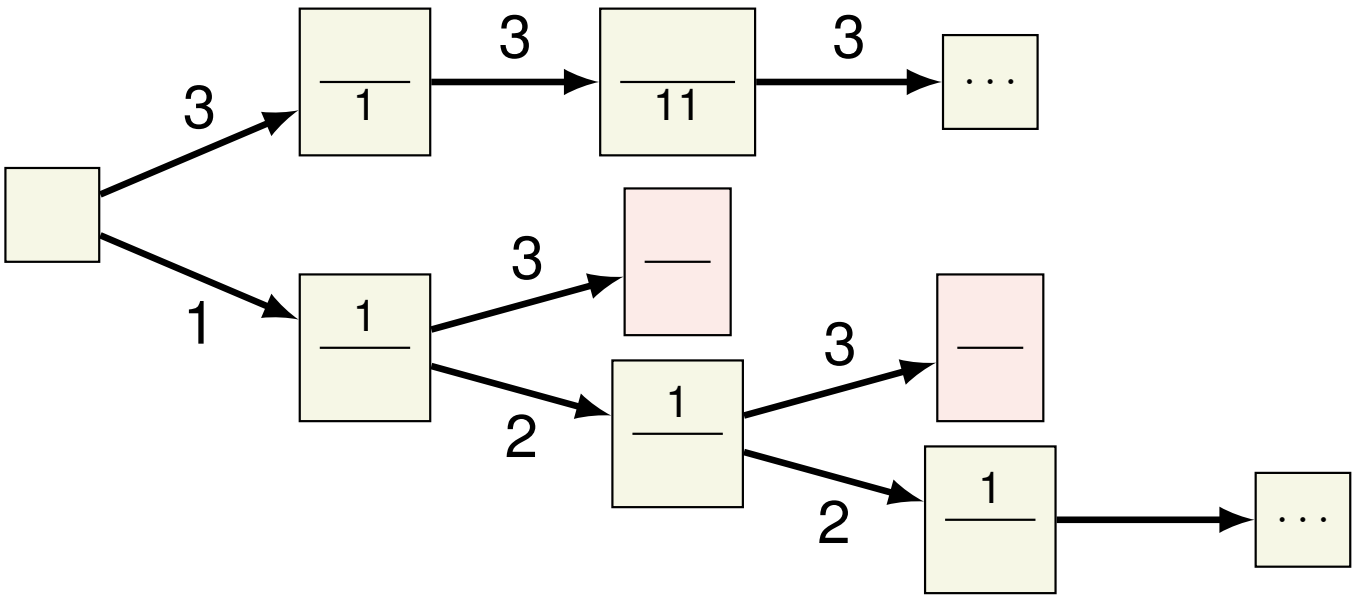


$L = \{(12^*3)^+\}$

Idee

Mögliche Lösungen aufzählen, richtige Lösungen identifizieren

x_i / y_i	001 / 00	01 / 10	1 / 11
	1	2	3



$$L = \{(12^*3)^+\}$$

Definition (*TIME*)

Wir bezeichnen die minimale Anzahl der Schritte, bis eine DTM M mit Eingabe w hält als $time_M(w) \in \mathbb{N} \cup \{\infty\}$.

Sei $f : \mathbb{N} \to \mathbb{N}$ total. Dann ist

$$TIME(f(n)) := \{A \subseteq \Sigma^* \mid \exists DTM\ M.A = L(M) \wedge \forall w \in \Sigma^*.time_M(w) \leq f(|w|)\}$$

die Klasse der in Zeit $f(n)$ von einer DTM entscheidbaren Sprachen.

- $TIME(\mathcal{O}(n))$ enthält alle "linearen Probleme".
- Also alle Probleme, für die ein Linearzeitalgorithmus existiert.

Definition (*NTIME*)

Wir bezeichnen die minimale Anzahl der Schritte, bis eine NTM M mit Eingabe w hält als $ntime_M(w) \in \mathbb{N}$.

$$ntime_M(w) := \begin{cases} \text{minimale Schrittanzahl} & \text{falls } w \in L(M) \\ 0 & \text{falls } w \notin L(M) \end{cases}$$

Dann ist

$$NTIME(f(n)) := \{A \subseteq \Sigma^* \mid \exists NTM\ M.A = L(M) \wedge \forall w \in \Sigma^*.ntime_M(w) \leq f(|w|)\}$$

die Klasse der in Zeit $f(n)$ von einer NTM entscheidbaren Sprachen.

Definition

P ist die Menge aller von einer DTM in polynomieller Zeit entscheidbaren Sprachen.

$$P := \bigcup_{p \text{ Polynom}} TIME(p(n)) = \bigcup_{k \in \mathbb{N}} TIME(\mathcal{O}(n^k))$$

Definition

NP ist die Menge aller von einer NTM in polynomieller Zeit entscheidbaren Sprachen.

$$NP := \bigcup_{p \text{ Polynom}} NTIME(p(n)) = \bigcup_{k \in \mathbb{N}} NTIME(\mathcal{O}(n^k))$$

Verifikator

Definition (Verifikator)

Sei M eine DTM mit $L(M) \subseteq \{w\#c \mid w \in \Sigma^*, c \in \Delta^*\}$.

- Falls $w\#c \in L(M)$, dann heißt c Zertifikat für w.
- M ist ein polynomiell beschränkter Verifikator für

$$\{w \in \Sigma^* \mid \exists c \in \Delta^*. w\#c \in L(M)\}$$

falls $time_M(w\#c) \leq p(|w|)$ für ein Polynom p.

- NTM rät Lösung (Zertifikat), DTM probiert sie aus.
- Verifizieren (wahrscheinlich) einfacher als Lösung finden.

Satz

$A \in NP$ gdw es einen pol. beschränkten Verifikator für A gibt.

Definition (Polynomielle Reduzierbarkeit)

Eine Menge $A \subseteq \Sigma^*$ ist polynomiell reduzierbar auf eine Menge $B \subseteq \Gamma^*$ gdw es eine totale und von einer DTM in polynomieller Zeit berechenbare Funktion $f : \Sigma^* \to \Gamma^*$ gibt mit

$$\forall w \in \Sigma^*.w \in A \iff f(w) \in B$$

Wir schreiben dann $A \leq_P B$.

- Die Relation $\leq_P$ ist transitiv.
- P und NP sind nach unten abgeschlossen:

$$A \leq_P B \in P/NP \implies A \in P/NP$$

Definition (NP-Schwere)

Eine Sprache L heißt NP-schwer (NP-hart) wenn sich alle Sprachen in NP auf L reduzieren lassen.

$$\forall A \in NP.A \leq_P L$$

Definition (NP-Vollständigkeit)

Eine Sprache L heißt NP-vollständig wenn L NP-schwer ist und $L \in NP$.

Fragen:

- Gibt es überhaupt NP-vollständige Sprachen?
- Gibt es eine NP-vollständige Sprache in P?

Definition (Aussagenlogik)

Syntax der Aussagenlogik.

$$\text{Formeln} \quad F \to \neg F \mid (F \wedge F) \mid (F \vee F) \mid X$$
$$\text{Variablen} \quad X \to x \mid y \mid z \mid \ldots$$

Definition (SAT)

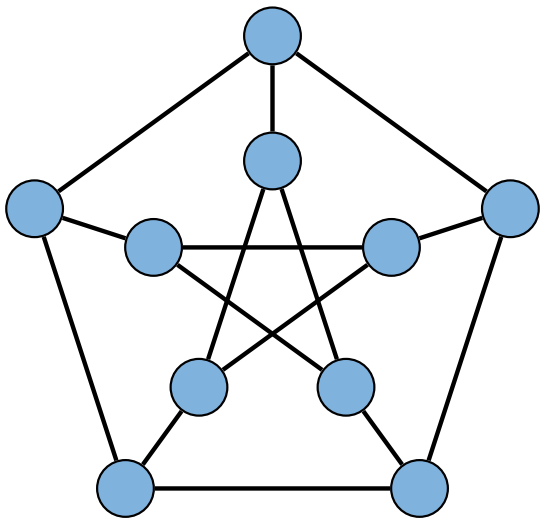
Gegeben eine aussagenlogische Formel F.
Ist F erfüllbar, also gibt es eine Belegung der Variablen in F, sodass F gilt?

Satz (Cook 1971)

SAT *ist NP-vollständig.*

Definition (3COL)

Gegeben ein Graph $G = (V, E)$.
Gibt es eine Färbung der Knoten V mit 3 Farben, so dass keine zwei benachbarten Knoten die gleiche Farbe haben?

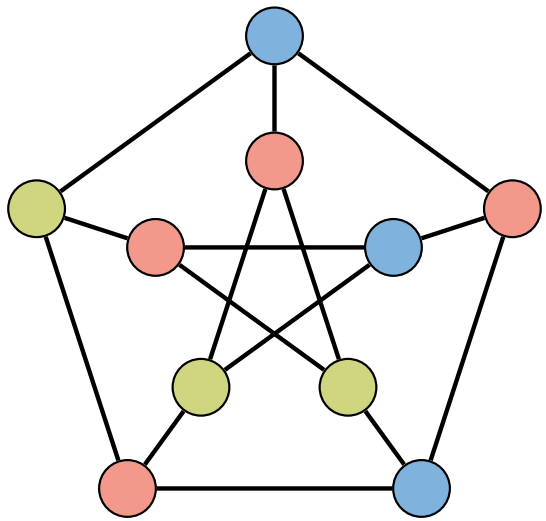


Satz

Es ist **3COL** $\leq_P$ **SAT** *und* **SAT** $\leq_P$ **3SAT** $\leq_P$ **3COL**.

Definition (3COL)

Gegeben ein Graph $G = (V, E)$.
Gibt es eine Färbung der Knoten V mit 3 Farben, so dass keine zwei benachbarten Knoten die gleiche Farbe haben?



Satz

Es ist **3COL** $\leq_P$ **SAT** *und* **SAT** $\leq_P$ **3SAT** $\leq_P$ **3COL**.