

Übersichtsfolien zur Übung

Theoretische Informatik Sommersemester 2013

Markus Kaiser

July 30, 2013

- 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

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

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**

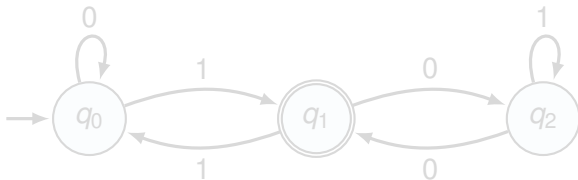
Definition (Operationen auf Sprachen)

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

Definition (Deterministischer endlicher Automat)

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

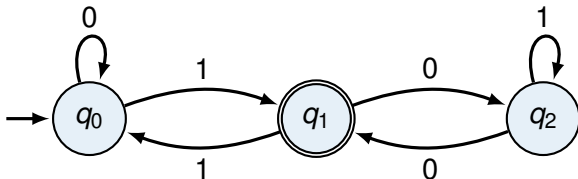
- endlichen Menge von **Zuständen** Q
- endlichen **Eingabealphabet** Σ
- totalen **Übergangsfunktion** $\delta : Q \times \Sigma \rightarrow Q$
- **Startzustand** $q_0 \in Q$
- Menge von **Endzuständen** $F \subseteq Q$



Definition (Deterministischer endlicher Automat)

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

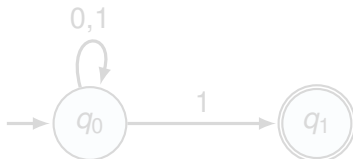
- endlichen Menge von **Zuständen** Q
- endlichen **Eingabealphabet** Σ
- totalen **Übergangsfunktion** $\delta : Q \times \Sigma \rightarrow Q$
- **Startzustand** $q_0 \in Q$
- Menge von **Endzuständen** $F \subseteq Q$



Definition (Nicht-Deterministischer endlicher Automat)

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

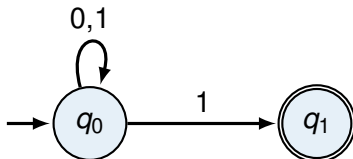
- Q, Σ, q_0, F wie ein DFA
- **Übergangsfunktion** $\delta : Q \times \Sigma \rightarrow P(Q)$



Definition (Nicht-Deterministischer endlicher Automat)

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

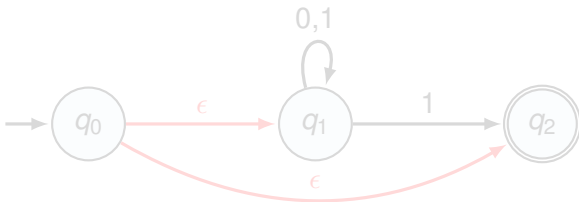
- Q, Σ, q_0, F wie ein DFA
- **Übergangsfunktion** $\delta : Q \times \Sigma \rightarrow P(Q)$



Definition (NFA mit ϵ -Übergängen)

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

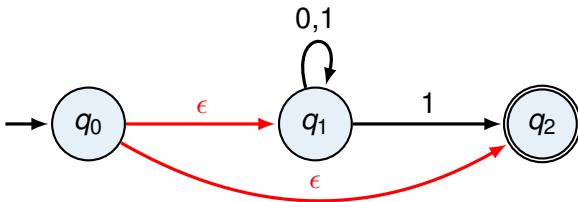
- Q, Σ, q_0, F wie ein DFA
- **Übergangsfunktion** $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow P(Q)$



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** $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow P(Q)$



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

$$\alpha = (0|1)^*00$$

$$L(\alpha) = \{x \mid x \text{ Binärzahl, } x \bmod 4 = 0\}$$

Idee (Kleene)

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

$$\gamma = \emptyset$$



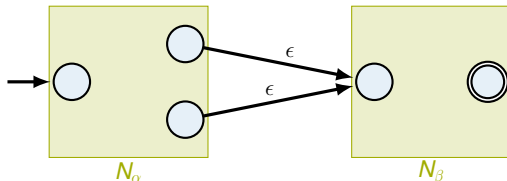
$$\gamma = \epsilon$$



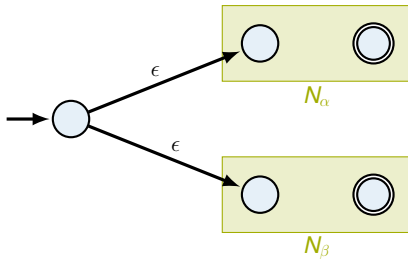
$$\gamma = a \in \Sigma$$



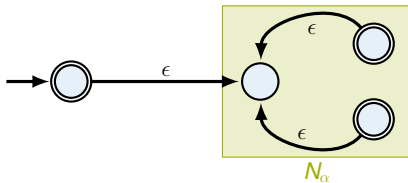
$$\gamma = \alpha\beta$$



$$\gamma = \alpha \mid \beta$$



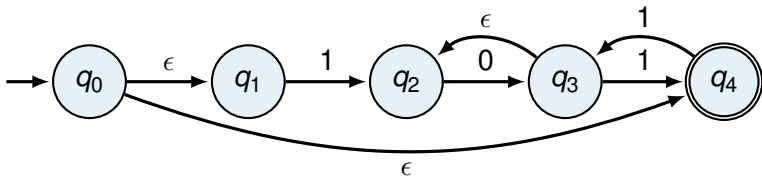
$$\gamma = \alpha^*$$



Idee

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

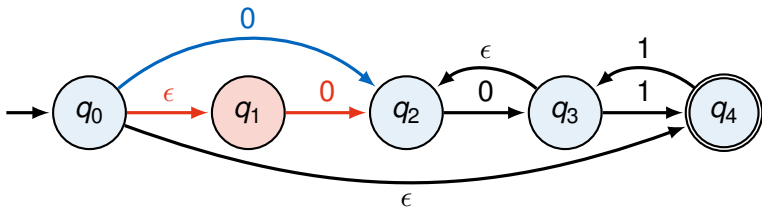
- 1 Entferne **unnötige Knoten**.
- 2 Für jeden **Pfad** der Form $\epsilon \dots \epsilon a \epsilon \dots \epsilon$ verbinde Anfangs- und Endknoten mit einer **a -Kante**.
- 3 Entferne alle **ϵ -Kanten** und unerreichbare Knoten.
- 4 Wurde das leere Wort akzeptiert mache den **Anfangszustand** zum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



Idee

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

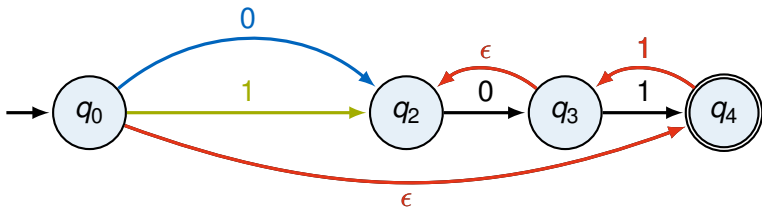
- 1 Entferne **unnötige Knoten**.
- 2 Für jeden **Pfad** der Form $\epsilon \dots \epsilon a \epsilon \dots \epsilon$ verbinde Anfangs- und Endknoten mit einer **a -Kante**.
- 3 Entferne alle **ϵ -Kanten** und unerreichbare Knoten.
- 4 Wurde das leere Wort akzeptiert mache den **Anfangszustand** zum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



Idee

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

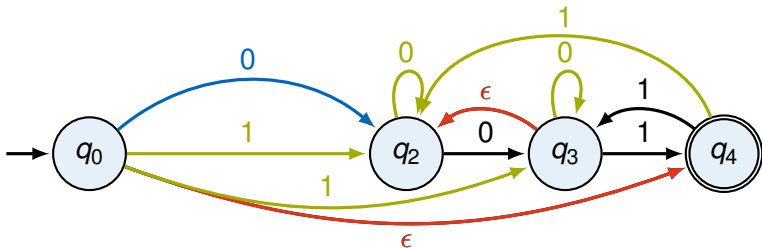
- 1 Entferne **unnötige Knoten**.
- 2 Für jeden **Pfad** der Form $\epsilon \dots \epsilon a \epsilon \dots \epsilon$ verbinde Anfangs- und Endknoten mit einer **a -Kante**.
- 3 Entferne alle ϵ -Kanten und unerreichbare Knoten.
- 4 Wurde das leere Wort akzeptiert mache den **Anfangszustand** zum Endzustand.



Idee

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

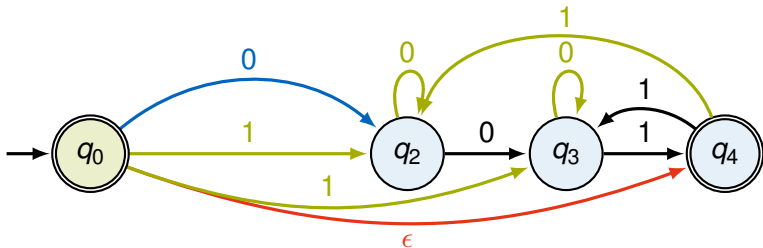
- 1 Entferne **unnötige Knoten**.
- 2 Für jeden **Pfad** der Form $\epsilon \dots \epsilon a \epsilon \dots \epsilon$ verbinde Anfangs- und Endknoten mit einer **a -Kante**.
- 3 Entferne alle **ϵ -Kanten** und unerreichbare Knoten.
- 4 Wurde das leere Wort akzeptiert mache den **Anfangszustand** zum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



Idee

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

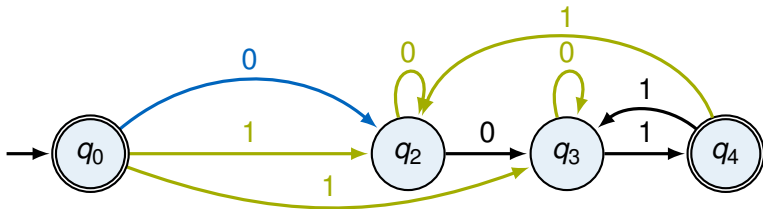
- 1 Entferne **unnötige Knoten**.
- 2 Für jeden **Pfad** der Form $\epsilon \dots \epsilon a \epsilon \dots \epsilon$ verbinde Anfangs- und Endknoten mit einer **a -Kante**.
- 3 Entferne alle **ϵ -Kanten** und unerreichbare Knoten.
- 4 Wurde das leere Wort akzeptiert mache den **Anfangszustand** zum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



Idee

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

- 1 Entferne **unnötige Knoten**.
- 2 Für jeden **Pfad** der Form $\epsilon \dots \epsilon a \epsilon \dots \epsilon$ 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.



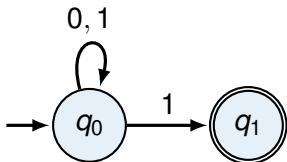
Idee (Potenzmengenkonstruktion)

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

- $\bar{\delta} : P(Q) \times \Sigma \rightarrow P(Q)$

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

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



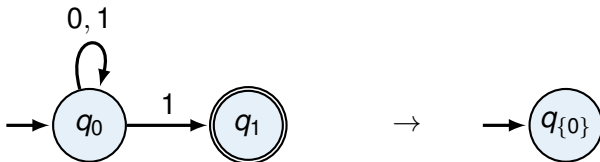
Idee (Potenzmengenkonstruktion)

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

- $\bar{\delta} : P(Q) \times \Sigma \rightarrow P(Q)$

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

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



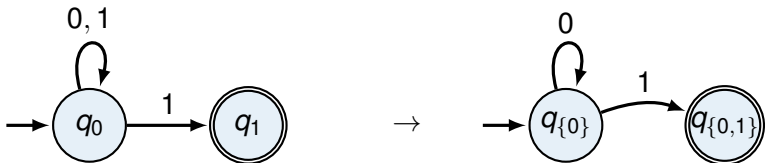
Idee (Potenzmengenkonstruktion)

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

- $\bar{\delta} : P(Q) \times \Sigma \rightarrow P(Q)$

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

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



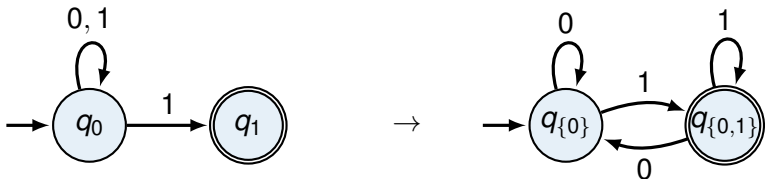
Idee (Potenzmengenkonstruktion)

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

- $\bar{\delta} : P(Q) \times \Sigma \rightarrow P(Q)$

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

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



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 $|$ einen **Halbring** $\langle \mathfrak{R}, |, \circ, \emptyset, \epsilon \rangle$.

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

Beispiel

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

Satz (Ardens Lemma)

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

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

Speziell gilt für reguläre Ausdrücke

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

Beispiel

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

Idee

Erzeuge ein Gleichungssystem aus allen Zuständen.

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

$$X_0 \equiv 1X_0 \mid 0X_1$$

$$\equiv$$

$$\equiv 1(1X_0 \mid 0X_1) \mid 0(1X_0 \mid 0X_1)$$

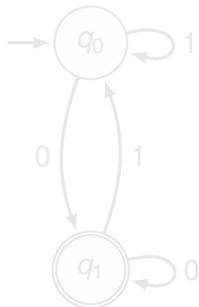
$$\equiv$$

$$\equiv 1(1X_0 \mid 0X_1) \mid 0(1X_0 \mid 0X_1)$$

$$X_1 \equiv 1X_0 \mid 0X_1 \mid \epsilon$$

$$\equiv 0X_1 \mid (\epsilon \mid 1X_0)$$

$$\equiv 0^*(\epsilon \mid 1X_0)$$



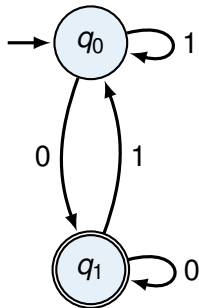
Idee

Erzeuge ein Gleichungssystem aus allen Zuständen.

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

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

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



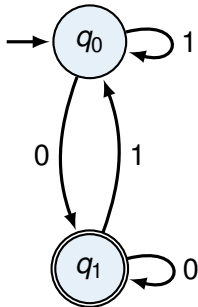
Idee

Erzeuge ein Gleichungssystem aus allen Zuständen.

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

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

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



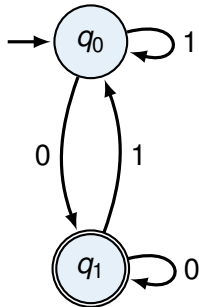
Idee

Erzeuge ein Gleichungssystem aus allen Zuständen.

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

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

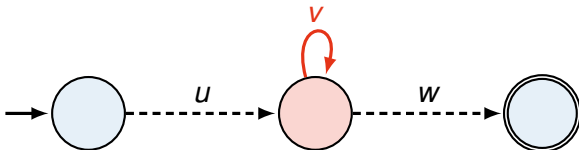
$$\begin{aligned}
 X_1 &\equiv 1X_0 \mid 0X_1 \mid \epsilon \\
 &\equiv 0X_1 \mid (\epsilon \mid 1X_0) \\
 &\equiv 0^*(\epsilon \mid 1X_0)
 \end{aligned}$$



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

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



Idee

Gegenbeispiel fürs Pumpinglemma suchen.

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

Beispiel

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

Idee

Gegenbeispiel fürs Pumpinglemma suchen.

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

$\equiv_L \subseteq \Sigma^* \times \Sigma^*$:

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

Definition (Äquivalente Zustände)

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

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

Definition (Äquivalente Worte)

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

$\equiv_L \subseteq \Sigma^* \times \Sigma^*$:

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

Definition (Äquivalente Zustände)

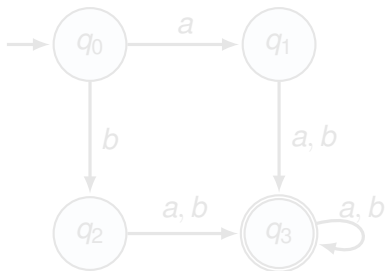
Zwei Zustände im DFA A sind **äquivalent** wenn sie die selbe Sprache akzeptieren.

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

Idee

Erzeuge den **Quotientenautomaten**.

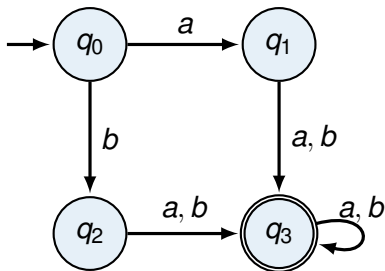
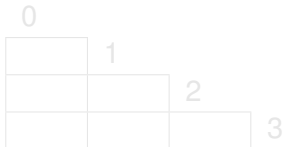
- 1 Entferne alle von q_0 **nicht erreichbaren** Zustände
- 2 Berechne die **unterscheidbaren** Zustände
- 3 **Kollabiere** die äquivalenten Zustände



Idee

Erzeuge den **Quotientenautomaten**.

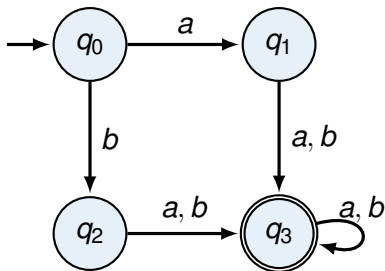
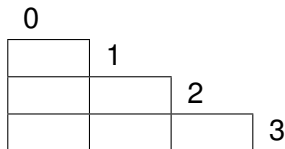
- 1 Entferne alle von q_0 **nicht erreichbaren** Zustände
- 2 Berechne die **unterscheidbaren** Zustände
- 3 **Kollabiere** die äquivalenten Zustände



Idee

Erzeuge den **Quotientenautomaten**.

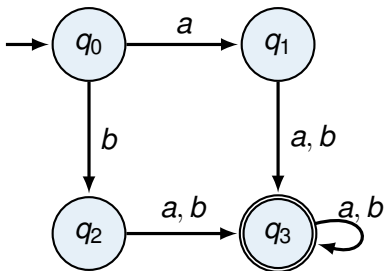
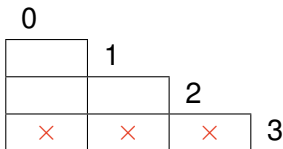
- 1 Entferne alle von q_0 **nicht erreichbaren** Zustände
- 2 Berechne die **unterscheidbaren** Zustände
- 3 **Kollabiere** die äquivalenten Zustände



Idee

Erzeuge den **Quotientenautomaten**.

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

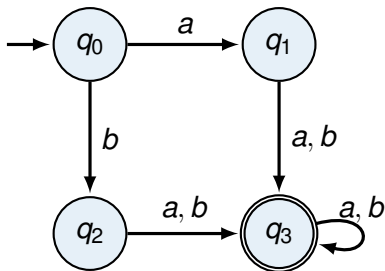


Idee

Erzeuge den **Quotientenautomaten**.

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

0			
1/a	1		
1/a		2	
×	×	×	3



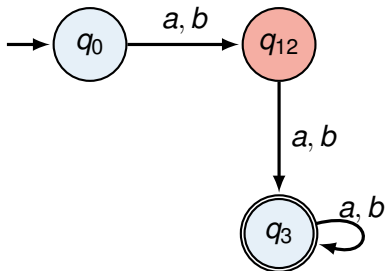
Idee

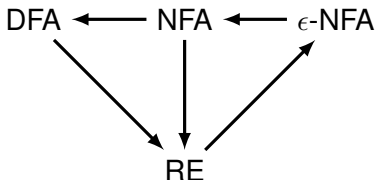
Erzeuge den **Quotientenautomaten**.

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

0

1/a	1		
1/a		2	
×	×	×	3





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.

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

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.

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

Definition (Ableitungsrelation)

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

$$\alpha \rightarrow_G \beta$$

gdw es eine Regel $A \rightarrow \gamma$ 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 $S \rightarrow 0S1 \mid S11 \mid 1$:

$$\begin{aligned} S &\rightarrow_G 0S1 \rightarrow_G 00S11 \rightarrow_G 00S1111 \rightarrow_G 0011111 \\ \Rightarrow S &\rightarrow_G^* 0011111 \end{aligned}$$

Definition (Kontextfreie Sprache)

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

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

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

Induktive Sprachdefinition

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

Beispiel (Vorbereitung 3)

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

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

Also z.B.:

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

Definition (Chomsky-Normalform)

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

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

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

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

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

wird zu:

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

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

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

wird zu:

$$A \rightarrow a \mid bS$$

$$S \rightarrow a \mid bS$$

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.

$$S \rightarrow aa \mid Bb \mid b, \quad B \rightarrow \dots$$

wird zu:

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

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 $A \rightarrow B_1 B_2 \dots B_k$ durch neue Nichtterminale mit Produktionen der Länge 2.

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

wird zu:

$$\begin{aligned} S &\rightarrow X_a T_1 \\ T_1 &\rightarrow X_b T_2, \quad T_2 \rightarrow B X_a \end{aligned}$$

Definition

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

Ein Symbol $X \in V \cup \Sigma$ ist

nützlich es gibt $S \rightarrow_G^* w \in \Sigma^*$ in der X **vorkommt**

erzeugend es gibt $X \rightarrow_G^* w \in \Sigma^*$

erreichbar es gibt $S \rightarrow_G^* \alpha X \beta$

Satz

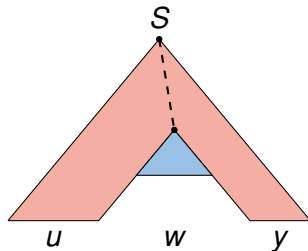
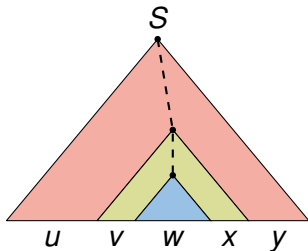
*Nützliche Symbole **sind** erzeugend und erreichbar. Aber **nicht** notwendigerweise umgekehrt.*

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

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

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



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

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

ist

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

$$V_{ij} = \{A \in V \mid (A \rightarrow a_i) \in P\}$$

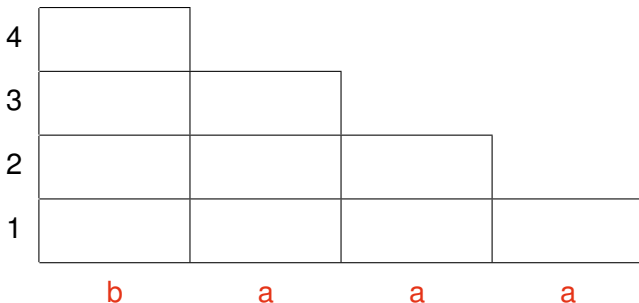
$$V_{ij} = \{A \in V \mid \exists k, B \in V_{ik}, C \in V_{k+1,j} \cdot (A \rightarrow BC) \in P\}$$

Idee

Kombiniere **Teilwörter** zum ganzen Wort, wenn möglich.

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

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



Idee

Kombiniere **Teilwörter** zum ganzen Wort, wenn möglich.

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

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

4				
3				
2				
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_{ij} .
- 2 Befülle die Tabelle von unten nach oben.

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

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_{ij} .
- 2 Befülle die Tabelle von unten nach oben.

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

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_{ij} .
- 2 Befülle die Tabelle von unten nach oben.

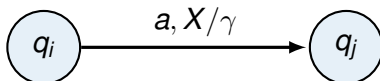
$S \rightarrow AB \mid BC, \quad A \rightarrow BA \mid a, \quad B \rightarrow CC \mid b, \quad C \rightarrow AB \mid a$

4	S, ...			
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-Automat) 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** $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow P(Q \times \Gamma^*)$
- **Startzustand** $q_0 \in Q$
- **Kellerinitialisierung** $Z_0 \in \Gamma$
- Menge von **Endzuständen** $F \subseteq Q$



Definition (Kellerautomat)

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

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

Definition (Akzeptanz)

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

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

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

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

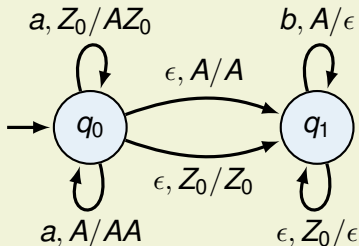
Definition (Kellerautomat)

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

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

Beispiel

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



CFG \longleftrightarrow CNF \longleftrightarrow PDA_ε \longleftrightarrow PDA_F

■ 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)

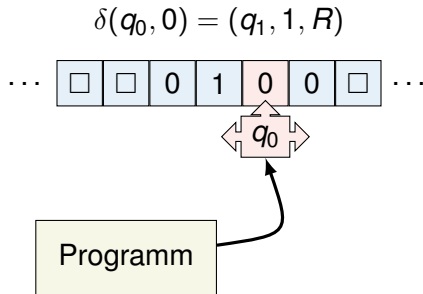
Eine deterministische **Turingmaschine (TM)** ist ein Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ aus einer/einem

- endlichen Menge von **Zuständen** Q
- endlichen **Eingabealphabet** Σ
- endlichen **Bandalphabet** Γ mit $\Sigma \subset \Gamma$
- **Übergangsfunktion** $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$
- **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

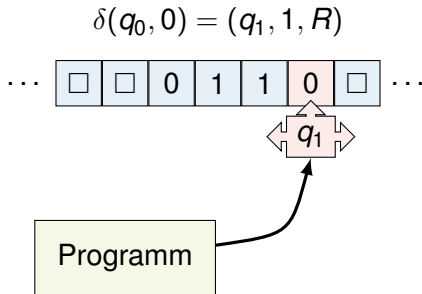
- **Übergangsfunktion** $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$



Definition (Turingmaschine)

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

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



Definition (Nichtdeterministische Turingmaschine)

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

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

Satz

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

Berechenbare Funktionen

Typ 0 - Rekursiv aufzählbar
Turingmaschinen, λ -Kalkül

Typ 1 - Kontextsensitiv
CSG

Typ 2 - Kontextfrei
PDA, CFG

Typ 3 - Regulär
DFA, RE

Definition (LOOP-Programm)

Syntax von **LOOP-Programmen**.

Es ist $X \in \{x_0, x_1, \dots\}$ und $C \in \mathbb{N}$.

```
P → X := X + C
   | X := X - C
   | P; P
   | LOOP X DO P END
   | IF X = 0 DO P ELSE Q END
```

- Ausgabe steht in x_0 , Eingaben in x_1, \dots, 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** $\pi_i^k : \mathbb{N}^k \rightarrow \mathbb{N}, i \in [k]$

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

Definition (Komposition)

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

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

Basisfunktionen und Komposition

Schon **PR** sind:

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

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. . .

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

Definition (Erweitertes PR-Schema)

Das **erweiterte Schema der primitiven Rekursion** erlaubt

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

wobei

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

Satz

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

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 $f : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ eine Funktion.

Der μ -Operator definiert eine neue Funktion $\mu f : \mathbb{N}^k \rightarrow \mathbb{N}$:

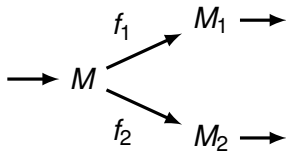
$$(\mu f)(\bar{x}) := \begin{cases} \min \{n \in \mathbb{N} \mid f(n, \bar{x}) = 0\} & \text{falls } n \text{ existent}^* \\ \perp & \text{sonst} \end{cases}$$

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

$$\mu f(\bar{x}) = \text{find}(0, \bar{x})$$

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

Sind f_1 und f_2 Endzustände von M , so bezeichnet



eine **Fallunterscheidung**.

Beispiel (Band=0?)

$$\delta(q_0, 0) = (q_0, 0, R)$$

$$\delta(q_0, \square) = (ja, \square, L)$$

$$\delta(q_0, a) = (nein, a, N) \quad \text{für } a \neq 0, \square$$

Definition (WHILE-Programm)

Syntax von **WHILE-Programmen**.

Es ist $X \in \{x_0, x_1, \dots\}$ und $C \in \mathbb{N}$.

```
P → X := X + C
   | X := X - C
   | P; P
   | WHILE X ≠ 0 DO P END
   | LOOP X DO P END
   | IF X = 0 DO P ELSE Q END
```

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

Definition (GOTO-Programm)

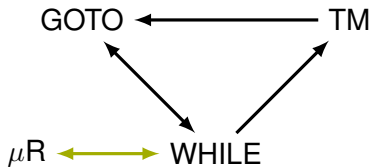
Syntax von **GOTO-Programmen**.

Es ist $X \in \{x_0, x_1, \dots\}$ und $C \in \mathbb{N}$.

Alle Anweisungen haben eine Markierung $M_1 : A_1; M_2 : A_2$.

```
P → X := X + C
   | X := X - C
   | P; P
   | GOTO Mi
   | IF X = 0 GOTO Mi
   | HALT
```

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



Satz

Sei A formale Sprache, dann ist äquivalent:

- A ist *Typ 0 Sprache*
- 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

■ Abschlusseigenschaften

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

■ Entscheidbarkeit

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

Formale Sprache - Σ^*

Typ 0 - Semi-Entscheidbar

Entscheidbar

LOOP, PR

NP

P

Typ 2 - Kontextfrei

Typ 3 - Regulär

Endlich

Definition (Intuitive Berechenbarkeit)

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

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

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

berechenbar ist.

Definition (Reduzierbarkeit)

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

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, \bar{K} nicht.

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 $f : \mathbb{N} \rightarrow A$ gibt, so dass

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

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 (Rice)

Sei F eine Menge berechenbarer Funktionen.
Sei weder $F = \emptyset$ noch $F = \text{alle ber. Funktionen}$ (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 (Postisches Korrespondenzproblem)

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

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

x_i
y_i

001
00

10
11

1
01

1

2

3

Satz

Das PCP ist **unentscheidbar**, aber semi-entscheidbar.

Idee

Mögliche Lösungen aufzählen, richtige Lösungen identifizieren

$$\frac{x_i}{y_i}$$

$$\frac{001}{00}$$

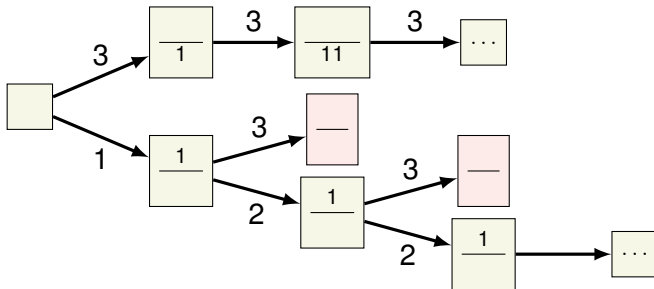
$$\frac{01}{10}$$

$$\frac{1}{11}$$

1

2

3



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

Idee

Mögliche Lösungen aufzählen, richtige Lösungen identifizieren

$$\frac{x_i}{y_i}$$

$$\frac{001}{00}$$

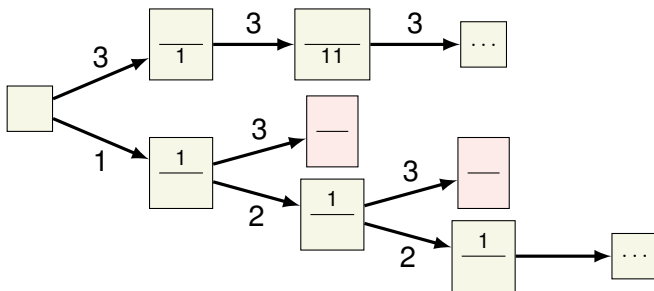
$$\frac{01}{10}$$

$$\frac{1}{11}$$

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

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

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 $ntime_M(w) \in \mathbb{N}$.

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

Dann ist

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

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.

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

Definition

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

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

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 $f : \Sigma^* \rightarrow \Gamma^*$ 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**.

Formeln $F \rightarrow \neg F \mid (F \wedge F) \mid (F \vee F) \mid X$

Variablen $X \rightarrow x \mid y \mid z \mid \dots$

Definition (SAT)

Gegeben eine **aussagenlogische Formel** F .

Ist F **erfüllbar**, also gibt es eine Belegung der Variablen in F , sodass F gilt?

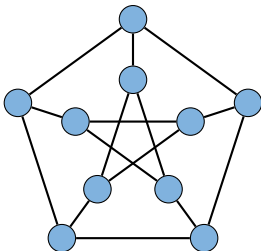
Satz (Cook 1971)

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?



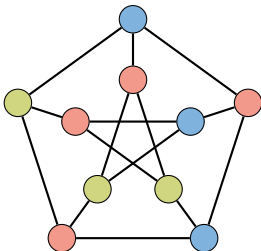
Satz

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 $SAT \leq_P 3SAT \leq_P 3COL$.

- RE \rightarrow ϵ -NFA \rightarrow NFA \rightarrow DFA
- (Produktautomat)
- Quotientenautomat, Minimale DFAs
- Reguläres Pumpinglemma
- CNF-Synthese
- Nützliche Symbole, CYK
- (Kellerautomaten)
- Kontextfreies Pumpinglemma
- Turingmaschinen
- LOOP und PR
- WHILE und μ -Rekursion
- Berechenbarkeit, Entscheidbarkeit, Satz von Rice
- PCP
- (P und NP, Verifikatoren)
- Reduktionen von und auf NP-Probleme