Einführung in die Theoretische Informatik

Javier Esparza

CIT School
TU München

Sommersemester 2024

Version vom 21. April 2024

Kapitel I Organisatorisches

Siehe https://www.cs.cit.tum.de/tcs/lehre/sommersemester-2024/theo/

Vorkenntnisse und weiterführende Vorlesungen

- Vorkenntnisse:
  - IN0001 Einführung in die Informatik
  - IN0015 Diskrete Strukturen
- Weiterführende Vorlesungen:
  - Automaten und formale Sprachen
  - Komplexitätstheorie
  - Logik
  - Model Checking
  - Compilerbau
  - ...

Inhalt und Gliederung der Vorlesung

Formale Sprachen

- Endliche Automaten und reguläre Ausdrücke
  - Grundlagen der Textanalyse, der lexikalischen Analyse von Programmiersprachen, und der Spezifikation und Analyse von Kommunikationsprotokollen.
- Kontextfreie Grammatiken
  - Grundlagen der syntaktischen Analyse von Programmiersprachen, Parsing, Compilerbau.

Berechenbarkeit und Komplexität

- Theorie der Berechenbarkeit
  - Untersuchung der Grenzen, was Rechner prinzipiell können.
- Komplexitätstheorie
  - Untersuchung der Grenzen, was Rechner mit begrenzten Ressourcen können.

Literatur


John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman.
Introduction to Automata Theory, Languages, and Computation.


Michael Sipser.
Introduction to the Theory of Computation.


Dexter Kozen.
Automata and Computability.


Katrin Erk, Lutz Priese.
Theoretische Informatik: Eine umfassende Einführung.


Uwe Schöning.
Theoretische Informatik — kurzgefasst.


Javier Esparza and Michael Blondin.
Automata Theory – An Algorithmic Approach.

Kapitel II Formale Sprachen

1. Einleitung: Kommunikation mit Rechnern

Kapitel II Formale Sprachen

1. Einleitung: Kommunikation mit Rechnern

Abbildung: Rechenmaschinen für arithmetische Operationen, XVII-XX Jh.
(Addition, Substraktion, Multiplikation, Division)
(By Ezrdr (Own work), via Wikipedia)

Abbildung: Gezeitenrechenmaschine im Deutschen Museum, gebaut 1935-39. (Quelle: Deutsches Museum)

Für die Berechnung von Funktionen der Gestalt $f(t) = \sum_{i=1}^{34} a_i \cos w_i t$

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

6. *The universal computing machine.*

It is possible to invent a single machine which can be used to compute any computable sequence. If this machine $\mathfrak{U}$ is supplied with a tape on

Abbildung: Alan Turings Artikel (1936), in dem er eine (abstrakte) universelle Rechenmaschine beschreibt

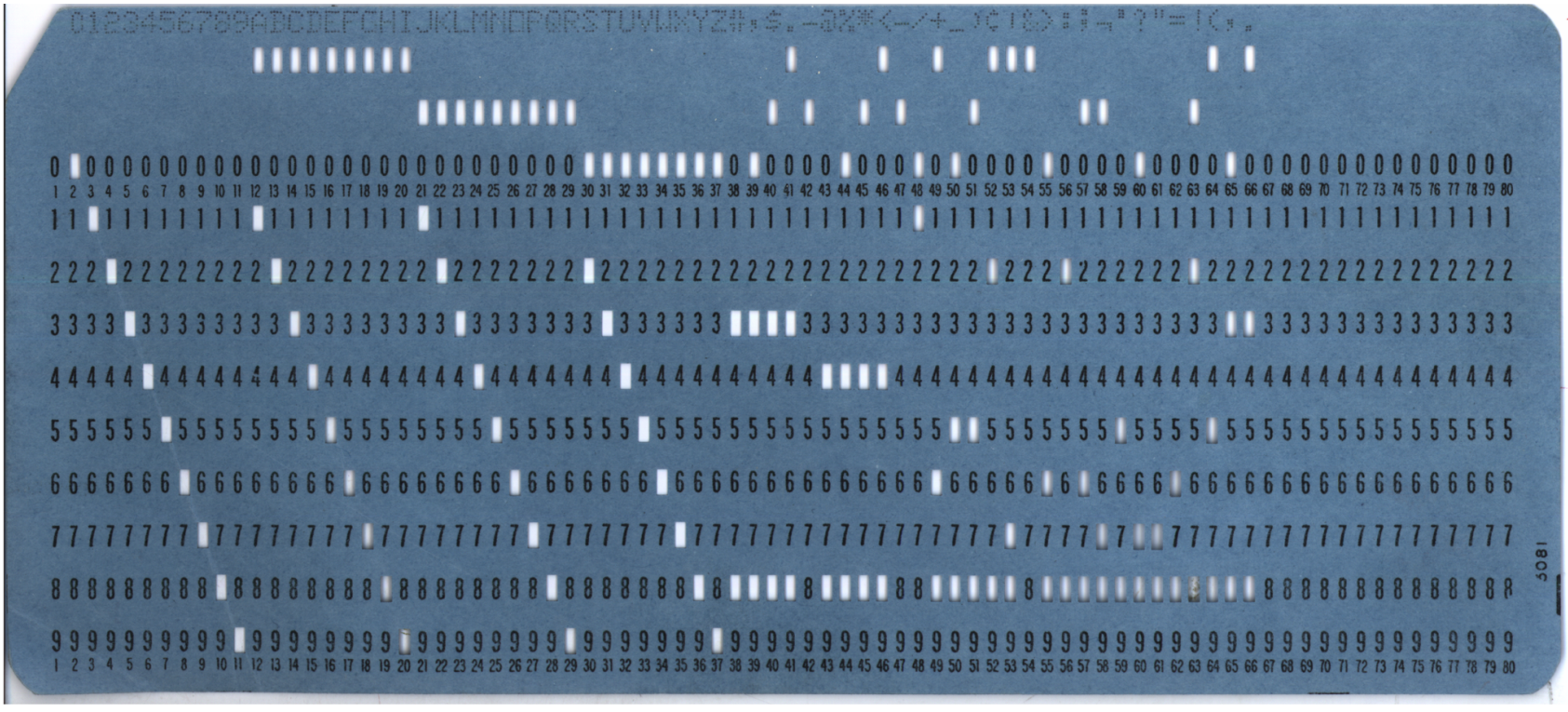


Abbildung: Programmieren mit Lochkarten in den 1960er Jahre

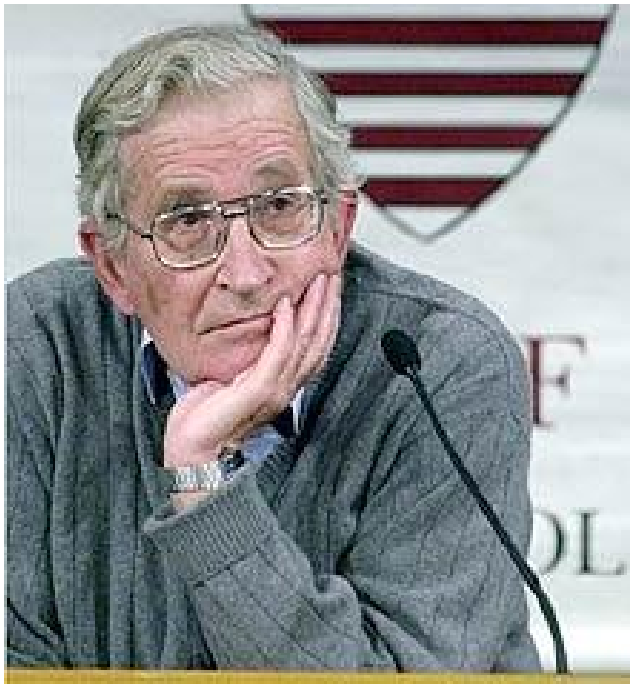
- In den 1950er setzt sich die Idee durch,

  Programmiersprachen

  für die Kommunikation mit Maschinen zu entwickeln.
- Gleichzeitig entwickelt der Linguist Noam Chomsky eine neue Theorie (Transformationgrammatik) der natürlichen Sprachen, mit dem Ziel, die Mechanismen zu erklären, die Menschen verwenden, um Sätze zu bauen.

  Noam Chomsky.
  Three Models for the Description of Language.
  Transactions on Information Theory, 113-124, 1956.
- Sein Begriff der formalen Grammatiken wird 1960 von dem ALGOL 60 Kommittee übernommen
  (ALGOL = ALGOrithmic Language)
- Seitdem wird die Syntax von Programmiersprachen durch formale Grammatiken beschrieben.



Noam Chomsky (1928 -) ist emeritierter Professor für Linguistik am Massachusetts Institute of Technology (MIT) und gehört zu den bekanntesten Linguisten der Gegenwart.

Er übte durch die Verbindung der Wissenschaftsdisziplinen Linguistik, Kognitionswissenschaften und Informatik besonders in der zweiten Hälfte des 20. Jahrhunderts starken Einfluss auf deren Entwicklung aus.

Chomsky war außerdem ab den 1960er- und 1970er-Jahren oft im politischen und wissenschaftlichen Diskurs in den Massenmedien präsent.

[Quelle: Wikipedia, 14.04.2024]

- Beispiele von Regeln für den Bau von Sätzen:

$$
\begin{aligned}
\text{Satz} &\rightarrow \text{Nominalphrase Verbalphrase} \\
\text{Verbalphrase} &\rightarrow \text{Verb Nominalphrase} \\
\text{Nominalphrase} &\rightarrow \text{Artikel Nomen} \\
\text{Satz} &\rightarrow \text{Präpositionalphrase Verbalphrase}
\end{aligned}
$$

- Beispiele von Regeln für den Bau von Sätzen:

$$
\begin{aligned}
\text{Satz} &\rightarrow \text{Nominalphrase Verbalphrase} \\
\text{Verbalphrase} &\rightarrow \text{Verb Nominalphrase} \\
\text{Nominalphrase} &\rightarrow \text{Artikel Nomen} \\
\text{Satz} &\rightarrow \text{Präpositionalphrase Verbalphrase}
\end{aligned}
$$

- Beispiele von Regeln für den Bau von Programmen:

$$
\begin{aligned}
\text{keyword\_item} &\rightarrow \text{identifier “=” expression} \\
\text{identifier} &\rightarrow \text{letter identifier2} \\
\text{identifier2} &\rightarrow \text{letter identifier2} \\
\text{identifier2} &\rightarrow \text{“ ”}
\end{aligned}
$$

- Das Erkennungsproblem: Ist eine gegebene Zeichenkette ein Programm, d.h. kann sie von den Regeln erzeugt werden?
- Recognizer: Programm, welches das Erkennungsproblem für eine gegebene Grammatik löst.
  Recognizer $\rightarrow$ Parser $\rightarrow$ Compiler
- Hauptfragen:
  - Für welche Grammatiken gibt es effiziente Recognizer?
  - Gegeben eine Grammatik, wie kann ein Recognizer automatisch konstruiert werden?

2. Grundbegriffe

Definition 2.1

- Ein Alphabet Σ ist eine endliche Menge.
  Bsp: $\{0, 1\}$, ASCII, Unicode.

2. Grundbegriffe

Definition 2.1

- Ein Alphabet Σ ist eine endliche Menge.
  Bsp: $\{0, 1\}$, ASCII, Unicode.
- Ein Wort/String über Σ is eine endliche Folge von Zeichen aus Σ, zB 010.

2. Grundbegriffe

Definition 2.1

- Ein Alphabet Σ ist eine endliche Menge.
  Bsp: $\{0, 1\}$, ASCII, Unicode.
- Ein Wort/String über Σ is eine endliche Folge von Zeichen aus Σ, zB 010.
- $|w|$ bezeichnet die Länge eines Wortes w.

2. Grundbegriffe

Definition 2.1

- Ein Alphabet Σ ist eine endliche Menge.
  Bsp: $\{0, 1\}$, ASCII, Unicode.
- Ein Wort/String über Σ is eine endliche Folge von Zeichen aus Σ, zB 010.
- $|w|$ bezeichnet die Länge eines Wortes w.
- Das leere Wort (das einzige Wort der Länge 0) wird mit ϵ bezeichnet.

2. Grundbegriffe

Definition 2.1

- Ein Alphabet Σ ist eine endliche Menge.
  Bsp: $\{0, 1\}$, ASCII, Unicode.
- Ein Wort/String über Σ is eine endliche Folge von Zeichen aus Σ, zB 010.
- $|w|$ bezeichnet die Länge eines Wortes w.
- Das leere Wort (das einzige Wort der Länge 0) wird mit ϵ bezeichnet.
- Sind u und v Wörter, so ist uv ihre Konkatenation.

2. Grundbegriffe

Definition 2.1

- Ein Alphabet Σ ist eine endliche Menge.
  Bsp: $\{0, 1\}$, ASCII, Unicode.
- Ein Wort/String über Σ is eine endliche Folge von Zeichen aus Σ, zB 010.
- $|w|$ bezeichnet die Länge eines Wortes w.
- Das leere Wort (das einzige Wort der Länge 0) wird mit ϵ bezeichnet.
- Sind u und v Wörter, so ist uv ihre Konkatenation.
- Ist w ein Wort, so ist w^n definiert durch $w^0 = \epsilon$ und $w^{n+1} = ww^n$. Bsp: $(ab)^3 = ababab$.

2. Grundbegriffe

Definition 2.1

- Ein Alphabet Σ ist eine endliche Menge.
  Bsp: $\{0, 1\}$, ASCII, Unicode.
- Ein Wort/String über Σ is eine endliche Folge von Zeichen aus Σ, zB 010.
- $|w|$ bezeichnet die Länge eines Wortes w.
- Das leere Wort (das einzige Wort der Länge 0) wird mit ϵ bezeichnet.
- Sind u und v Wörter, so ist uv ihre Konkatenation.
- Ist w ein Wort, so ist w^n definiert durch $w^0 = \epsilon$ und $w^{n+1} = ww^n$. Bsp: $(ab)^3 = ababab$.
- Σ^* ist die Menge aller Wörter über Σ.

2. Grundbegriffe

Definition 2.1

- Ein Alphabet Σ ist eine endliche Menge.
  Bsp: $\{0, 1\}$, ASCII, Unicode.
- Ein Wort/String über Σ is eine endliche Folge von Zeichen aus Σ, zB 010.
- $|w|$ bezeichnet die Länge eines Wortes w.
- Das leere Wort (das einzige Wort der Länge 0) wird mit ϵ bezeichnet.
- Sind u und v Wörter, so ist uv ihre Konkatenation.
- Ist w ein Wort, so ist w^n definiert durch $w^0 = \epsilon$ und $w^{n+1} = ww^n$. Bsp: $(ab)^3 = ababab$.
- Σ^* ist die Menge aller Wörter über Σ.
- Eine Teilmenge $L \subseteq \Sigma^*$ ist eine (formale) Sprache.

Beispiel 2.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)

Beispiel 2.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die Menge der deutschen Sätze ist keine formale Sprache

Beispiel 2.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die Menge der deutschen Sätze ist keine formale Sprache
- $L_1 = \{\epsilon, ab, abab, ababab, \ldots\} = \{(ab)^n \mid n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a, b\})$

Beispiel 2.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die Menge der deutschen Sätze ist keine formale Sprache
- $L_1 = \{\epsilon, ab, abab, ababab, \ldots\} = \{(ab)^n \mid n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a, b\})$
- $L_2 = \{\epsilon, ab, aabb, aaabbb, \ldots\} = \{a^n b^n \mid n \in \mathbb{N}\}$
  $(\Sigma_2 = \{a, b\})$

Beispiel 2.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die Menge der deutschen Sätze ist keine formale Sprache
- $L_1 = \{\epsilon, ab, abab, ababab, \ldots\} = \{(ab)^n \mid n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a, b\})$
- $L_2 = \{\epsilon, ab, aabb, aaabbb, \ldots\} = \{a^n b^n \mid n \in \mathbb{N}\}$
  $(\Sigma_2 = \{a, b\})$
- $L_3 = \{\epsilon, 1, 100, 1001, 10000, \ldots\} =$
  $\{w \in \{0,1\}^* \mid w$ ist eine binär kodierte Quadratzahl$\}$
  $(\Sigma_3 = \{0, 1\})$

Beispiel 2.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die Menge der deutschen Sätze ist keine formale Sprache
- $L_1 = \{\epsilon, ab, abab, ababab, \ldots\} = \{(ab)^n \mid n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a, b\})$
- $L_2 = \{\epsilon, ab, aabb, aaabbb, \ldots\} = \{a^n b^n \mid n \in \mathbb{N}\}$
  $(\Sigma_2 = \{a, b\})$
- $L_3 = \{\epsilon, 1, 100, 1001, 10000, \ldots\} =$
  $\{w \in \{0,1\}^* \mid w$ ist eine binär kodierte Quadratzahl$\}$
  $(\Sigma_3 = \{0, 1\})$
- $\emptyset$

Beispiel 2.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die Menge der deutschen Sätze ist keine formale Sprache
- $L_1 = \{\epsilon, ab, abab, ababab, \ldots\} = \{(ab)^n \mid n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a, b\})$
- $L_2 = \{\epsilon, ab, aabb, aaabbb, \ldots\} = \{a^n b^n \mid n \in \mathbb{N}\}$
  $(\Sigma_2 = \{a, b\})$
- $L_3 = \{\epsilon, 1, 100, 1001, 10000, \ldots\} =$
  $\{w \in \{0,1\}^* \mid w$ ist eine binär kodierte Quadratzahl$\}$
  $(\Sigma_3 = \{0, 1\})$
- $\emptyset$
- $\{\epsilon\}$

Beispiel 2.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die Menge der deutschen Sätze ist keine formale Sprache
- $L_1 = \{\epsilon, ab, abab, ababab, \ldots\} = \{(ab)^n \mid n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a, b\})$
- $L_2 = \{\epsilon, ab, aabb, aaabbb, \ldots\} = \{a^n b^n \mid n \in \mathbb{N}\}$
  $(\Sigma_2 = \{a, b\})$
- $L_3 = \{\epsilon, 1, 100, 1001, 10000, \ldots\} =$
  $\{w \in \{0,1\}^* \mid w$ ist eine binär kodierte Quadratzahl$\}$
  $(\Sigma_3 = \{0, 1\})$
- $\emptyset$
- $\{\epsilon\}$
- ϵ oder ab sind keine Sprachen

Definition 2.3 (Operationen auf Sprachen)

Seien $A, B \subseteq \Sigma^*$.

Definition 2.3 (Operationen auf Sprachen)

Seien $A, B \subseteq \Sigma^*$.

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$

Definition 2.3 (Operationen auf Sprachen)

Seien $A, B \subseteq \Sigma^*$.

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = \underbrace{A \cdots A}_{n}$

  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$

Definition 2.3 (Operationen auf Sprachen)

Seien $A, B \subseteq \Sigma^*$.

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = \underbrace{A \cdots A}_{n}$

  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \dots\}$

Definition 2.3 (Operationen auf Sprachen)

Seien $A, B \subseteq \Sigma^*$.

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = \underbrace{A \cdots A}_{n}$

  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \dots\} \neq \{0, 1\}^*$

Definition 2.3 (Operationen auf Sprachen)

Seien $A, B \subseteq \Sigma^*$.

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = \underbrace{A \cdots A}_{n}$

  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \dots\} \neq \{0, 1\}^*$
- $A^+ = AA^* = \bigcup_{n \geq 1} A^n$
  Bsp: $\Sigma^+ =$ Menge aller nicht-leeren Wörter über Σ

Definition 2.3 (Operationen auf Sprachen)

Seien $A, B \subseteq \Sigma^*$.

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = \underbrace{A \cdots A}_{n}$

  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \dots\} \neq \{0, 1\}^*$
- $A^+ = AA^* = \bigcup_{n \geq 1} A^n$
  Bsp: $\Sigma^+ =$ Menge aller nicht-leeren Wörter über Σ

Achtung:

- Für alle A: $\epsilon \in A^*$
- $\emptyset^* =$

Definition 2.3 (Operationen auf Sprachen)

Seien $A, B \subseteq \Sigma^*$.

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = \underbrace{A \cdots A}_{n}$

  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \dots\} \neq \{0, 1\}^*$
- $A^+ = AA^* = \bigcup_{n \geq 1} A^n$
  Bsp: $\Sigma^+ =$ Menge aller nicht-leeren Wörter über Σ

Achtung:

- Für alle A: $\epsilon \in A^*$
- $\emptyset^* = \{\epsilon\}$

Einige Rechenregeln:

Lemma 2.4

- $\emptyset A = \emptyset$
- $\{\epsilon\} A = A$

Einige Rechenregeln:

Lemma 2.4

- $\emptyset A = \emptyset$
- $\{\epsilon\} A = A$

Lemma 2.5

- $A(B \cup C) = AB \cup AC$
- $(A \cup B)C = AC \cup BC$

Einige Rechenregeln:

Lemma 2.4

- $\emptyset A = \emptyset$
- $\{\epsilon\} A = A$

Lemma 2.5

- $A(B \cup C) = AB \cup AC$
- $(A \cup B)C = AC \cup BC$

Achtung: i.A. gilt $A(B \cap C) = AB \cap AC$ *nicht*.

Einige Rechenregeln:

Lemma 2.4

- $\emptyset A = \emptyset$
- $\{\epsilon\} A = A$

Lemma 2.5

- $A(B \cup C) = AB \cup AC$
- $(A \cup B)C = AC \cup BC$

Achtung: i.A. gilt $A(B \cap C) = AB \cap AC$ *nicht*.

Lemma 2.6

$A^* A^* = A^*$

2.1 Grammatiken

2.1 Grammatiken

Definition 2.7

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

- V ist eine endliche Menge von Nichtterminalzeichen (oder Nichtterminale, oder Variablen),
- Σ ist eine endliche Menge von Terminalzeichen (oder Terminale), disjunkt von V, auch genannt ein Alphabet,
- $P \subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ ist eine Menge von Produktionen, und
- $S \in V$ ist das Startsymbol.

2.1 Grammatiken

Definition 2.7

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

- V ist eine endliche Menge von Nichtterminalzeichen (oder Nichtterminale, oder Variablen),
- Σ ist eine endliche Menge von Terminalzeichen (oder Terminale), disjunkt von V, auch genannt ein Alphabet,
- $P \subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ ist eine Menge von Produktionen, und
- $S \in V$ ist das Startsymbol.

Konventionen:

- $A, B, C, \ldots$ sind Nichtterminale,
- $a, b, c, \ldots$ (und Sonderzeichen wie $+, *, \ldots$) sind Terminale,
- $\alpha, \beta, \gamma, \ldots \in (V \cup \Sigma)^*$
- Produktionen schreiben wir $\alpha \to \beta$ statt $(\alpha, \beta) \in P$.
- Statt $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$ schreiben wir $\alpha \to \beta_1 \mid \cdots \mid \beta_n$

Beispiel 2.8 (Arithmetische Ausdrücke)

- $V = \{\langle \text{Expr} \rangle, \langle \text{Term} \rangle, \langle \text{Factor} \rangle\}$
- $\Sigma = \{a, b, c, +, *, (,)\}$
- $S = \langle \text{Expr} \rangle$
- Die Menge P enthält folgende Produktionen:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\rightarrow \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle &\rightarrow \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle &\rightarrow \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle &\rightarrow \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle &\rightarrow a \mid b \mid c \\
\langle \text{Factor} \rangle &\rightarrow (\langle \text{Expr} \rangle)
\end{aligned}
$$

Definition 2.9

Eine Grammatik $G = (V, \Sigma, P, S)$ induziert eine Ableitungsrelation $\rightarrow_G$ auf Wörtern über $V \cup \Sigma$:

$$\alpha \rightarrow_G \alpha'$$

gdw es eine Regel $\beta \rightarrow \beta'$ in P und Wörter α_1, α_2 gibt, so dass

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

Beispiel : Für die Grammatik der arithmetischen Ausdrücken gilt

$$a + \langle \text{Term} \rangle + b \quad \rightarrow_G \quad a + \langle \text{Term} \rangle * \langle \text{Factor} \rangle + b$$

Definition 2.9

Eine Grammatik $G = (V, \Sigma, P, S)$ induziert eine Ableitungsrelation $\rightarrow_G$ auf Wörtern über $V \cup \Sigma$:

$$\alpha \rightarrow_G \alpha'$$

gdw es eine Regel $\beta \rightarrow \beta'$ in P und Wörter α_1, α_2 gibt, so dass

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

Beispiel : Für die Grammatik der arithmetischen Ausdrücken gilt

$$a + \langle \text{Term} \rangle + b \quad \rightarrow_G \quad a + \langle \text{Term} \rangle * \langle \text{Factor} \rangle + b$$

Eine Sequenz $\alpha_1 \rightarrow_G \alpha_2 \rightarrow_G \cdots \rightarrow_G \alpha_n$ ist eine Ableitung von α_n aus α_1.

Wenn $\alpha_1 = S$ und $\alpha_n \in \Sigma^*$, dann erzeugt G das Wort α_n.

Die Sprache von G ist die Menge aller Wörter, die von G erzeugt werden. Sie wird mit $L(G)$ bezeichnet.

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$\langle \text{Expr} \rangle \to$

$\to a * (b + c)$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$\langle \text{Expr} \rangle \to$

$\to a * (b + c)$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$\langle \text{Expr} \rangle \to$

$\to a * (b + c)$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$\langle \text{Expr} \rangle \to \langle \text{Term} \rangle$

$\to a * (b + c)$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$\langle \text{Expr} \rangle \to \langle \text{Term} \rangle$

$\to a * (b + c)$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$\langle \text{Expr} \rangle \to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle$

$\to a * (b + c)$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$\langle \text{Expr} \rangle \to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle$

$\to a * (b + c)$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle)
\end{aligned}
$$

$$
\to a * (b + c)
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle)
\end{aligned}
$$

$$
\to a * (b + c)
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle) \to a * (\langle \text{Expr} \rangle + \langle \text{Term} \rangle) \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle) \to a * (\langle \text{Expr} \rangle + \langle \text{Term} \rangle) \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle) \to a * (\langle \text{Expr} \rangle + \langle \text{Term} \rangle) \\
&\to a * (\langle \text{Term} \rangle + \langle \text{Term} \rangle) \\
&\\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle) \to a * (\langle \text{Expr} \rangle + \langle \text{Term} \rangle) \\
&\to a * (\langle \text{Term} \rangle + \langle \text{Term} \rangle) \\
&\\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle) \to a * (\langle \text{Expr} \rangle + \langle \text{Term} \rangle) \\
&\to a * (\langle \text{Term} \rangle + \langle \text{Term} \rangle) \to a * (\langle \text{Factor} \rangle + \langle \text{Term} \rangle) \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle) \to a * (\langle \text{Expr} \rangle + \langle \text{Term} \rangle) \\
&\to a * (\langle \text{Term} \rangle + \langle \text{Term} \rangle) \to a * (\langle \text{Factor} \rangle + \langle \text{Term} \rangle) \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle &\to \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle &\to \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle &\to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle &\to a \mid b \mid c \\
\langle \text{Factor} \rangle &\to (\langle \text{Expr} \rangle)
\end{aligned}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle) \to a * (\langle \text{Expr} \rangle + \langle \text{Term} \rangle) \\
&\to a * (\langle \text{Term} \rangle + \langle \text{Term} \rangle) \to a * (\langle \text{Factor} \rangle + \langle \text{Term} \rangle) \\
&\to a * (b + \langle \text{Term} \rangle) \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle &\to \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle &\to \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle &\to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle &\to a \mid b \mid c \\
\langle \text{Factor} \rangle &\to (\langle \text{Expr} \rangle)
\end{aligned}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle) \to a * (\langle \text{Expr} \rangle + \langle \text{Term} \rangle) \\
&\to a * (\langle \text{Term} \rangle + \langle \text{Term} \rangle) \to a * (\langle \text{Factor} \rangle + \langle \text{Term} \rangle) \\
&\to a * (b + \langle \text{Term} \rangle) \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle) \to a * (\langle \text{Expr} \rangle + \langle \text{Term} \rangle) \\
&\to a * (\langle \text{Term} \rangle + \langle \text{Term} \rangle) \to a * (\langle \text{Factor} \rangle + \langle \text{Term} \rangle) \\
&\to a * (b + \langle \text{Term} \rangle) \to a * (b + \langle \text{Factor} \rangle) \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle) \to a * (\langle \text{Expr} \rangle + \langle \text{Term} \rangle) \\
&\to a * (\langle \text{Term} \rangle + \langle \text{Term} \rangle) \to a * (\langle \text{Factor} \rangle + \langle \text{Term} \rangle) \\
&\to a * (b + \langle \text{Term} \rangle) \to a * (b + \langle \text{Factor} \rangle) \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.10 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\langle \text{Expr} \rangle & \to & \langle \text{Term} \rangle \\
\langle \text{Expr} \rangle & \to & \langle \text{Expr} \rangle + \langle \text{Term} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Factor} \rangle \\
\langle \text{Term} \rangle & \to & \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
\langle \text{Factor} \rangle & \to & a \mid b \mid c \\
\langle \text{Factor} \rangle & \to & (\langle \text{Expr} \rangle)
\end{array}
$$

Das Startsymbol ist $\langle \text{Expr} \rangle$.

Eine Ableitung von $a * (b + c)$:

$$
\begin{aligned}
\langle \text{Expr} \rangle &\to \langle \text{Term} \rangle \to \langle \text{Term} \rangle * \langle \text{Factor} \rangle \\
&\to \langle \text{Factor} \rangle * \langle \text{Factor} \rangle \to a * \langle \text{Factor} \rangle \\
&\to a * (\langle \text{Expr} \rangle) \to a * (\langle \text{Expr} \rangle + \langle \text{Term} \rangle) \\
&\to a * (\langle \text{Term} \rangle + \langle \text{Term} \rangle) \to a * (\langle \text{Factor} \rangle + \langle \text{Term} \rangle) \\
&\to a * (b + \langle \text{Term} \rangle) \to a * (b + \langle \text{Factor} \rangle) \\
&\to a * (b + c)
\end{aligned}
$$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{lrcl}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{lrcl}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{lrcl}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{lrcl}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS \to abcabc$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS \to abcabc \to abacbc$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS \to abcabc \to abacbc \to aabcbc$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS \to abcabc \to abacbc \to aabcbc \to aabbcc$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS \to abcabc \to abacbc \to aabcbc \to aabbcc$

Es gilt: $L(G_2) = M$. Ableitung von $aabbcc$ aus S:

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS \to abcabc \to abacbc \to aabcbc \to aabbcc$

Es gilt: $L(G_2) = M$. Ableitung von $aabbcc$ aus S:

$S \to aBSc$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS \to abcabc \to abacbc \to aabcbc \to aabbcc$

Es gilt: $L(G_2) = M$. Ableitung von $aabbcc$ aus S:

$S \to aBSc \to aBaBScc$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS \to abcabc \to abacbc \to aabcbc \to aabbcc$

Es gilt: $L(G_2) = M$. Ableitung von $aabbcc$ aus S:

$S \to aBSc \to aBaBScc \to aBaBcc$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS \to abcabc \to abacbc \to aabcbc \to aabbcc$

Es gilt: $L(G_2) = M$. Ableitung von $aabbcc$ aus S:

$S \to aBSc \to aBaBScc \to aBaBcc \to aaBBcc$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS \to abcabc \to abacbc \to aabcbc \to aabbcc$

Es gilt: $L(G_2) = M$. Ableitung von $aabbcc$ aus S:

$S \to aBSc \to aBaBScc \to aBaBcc \to aaBBcc \to aaBbcc$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS \to abcabc \to abacbc \to aabcbc \to aabbcc$

Es gilt: $L(G_2) = M$. Ableitung von $aabbcc$ aus S:

$S \to aBSc \to aBaBScc \to aBaBcc \to aaBBcc \to aaBbcc \to aabBcc$

Beispiel 2.11

Welche Grammatik erzeugt $L = \{a^n b^n c^n \mid n \geq 0\}$?

$$
\begin{array}{llll}
G_1: & S & \to & abcS \mid \epsilon \\
 & ba & \to & ab \\
 & cb & \to & bc \\
 & ca & \to & ac
\end{array}
\qquad\qquad
\begin{array}{llll}
G_2: & S & \to & aBSc \mid \epsilon \\
 & Ba & \to & aB \\
 & Bb & \to & bB \\
 & Bc & \to & bc
\end{array}
$$

Es gilt $L(G_1) \neq L$ weil $S \to abcS \to abcabcS \to abcabc$

Aber $L \subseteq L(G_1)$. Ableitung von $aabbcc$ aus S:

$S \to abcS \to abcabcS \to abcabc \to abacbc \to aabcbc \to aabbcc$

Es gilt: $L(G_2) = M$. Ableitung von $aabbcc$ aus S:

$S \to aBSc \to aBaBScc \to aBaBcc \to aaBBcc \to aaBbcc \to$
$aabBcc \to aabbcc$

Definition 2.12 (Reflexive transitive Hülle)

$$\begin{aligned}
\alpha &\to_G^0 \alpha \\
\alpha \to_G^{n+1} \gamma &\quad :\Leftrightarrow \quad \exists \beta.\ \alpha \to_G^n \beta \to_G \gamma \\
\alpha \to_G^* \beta &\quad :\Leftrightarrow \quad \exists n.\ \alpha \to_G^n \beta \\
\alpha \to_G^+ \beta &\quad :\Leftrightarrow \quad \exists n > 0.\ \alpha \to_G^n \beta
\end{aligned}$$

Beispiel: $\langle \text{Expr} \rangle \to_G^{11} a * (b + c)$ und so $\langle \text{Expr} \rangle \to_G^* a * (b + c)$.

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

2.2 Die Chomsky-Hierarchie

Eine Grammatik G ist vom

2.2 Die Chomsky-Hierarchie

Eine Grammatik G ist vom

Typ 0 immer.

2.2 Die Chomsky-Hierarchie

Eine Grammatik G ist vom

Typ 0 immer.

Typ 1 falls jede Produktion $\alpha \to \beta$ außer $S \to \epsilon$
$|\alpha| \leq |\beta|$ gilt
und, wenn $S \to \epsilon$ eine Produktion ist,
dann kommt S in β nicht vor

2.2 Die Chomsky-Hierarchie

Eine Grammatik G ist vom

Typ 0 immer.

Typ 1 falls jede Produktion $\alpha \to \beta$ außer $S \to \epsilon$
$|\alpha| \leq |\beta|$ gilt
und, wenn $S \to \epsilon$ eine Produktion ist,
dann kommt S in β nicht vor

Typ 2 falls G vom Typ 1 ist und
für jede Produktion $\alpha \to \beta$ gilt $\alpha \in V$.

2.2 Die Chomsky-Hierarchie

Eine Grammatik G ist vom

Typ 0 immer.

Typ 1 falls jede Produktion $\alpha \to \beta$ außer $S \to \epsilon$ $|\alpha| \leq |\beta|$ gilt
und, wenn $S \to \epsilon$ eine Produktion ist,
dann kommt S in β nicht vor

Typ 2 falls G vom Typ 1 ist und
für jede Produktion $\alpha \to \beta$ gilt $\alpha \in V$.

Typ 3 falls G vom Typ 2 ist und
für jede Produktion $\alpha \to \beta$ außer $S \to \epsilon$
gilt $\beta \in \Sigma \cup \Sigma V$.

2.2 Die Chomsky-Hierarchie

Eine Grammatik G ist vom

Typ 0 immer.

Typ 1 falls jede Produktion $\alpha \to \beta$ außer $S \to \epsilon$ $|\alpha| \leq |\beta|$ gilt
und, wenn $S \to \epsilon$ eine Produktion ist,
dann kommt S in β nicht vor

Typ 2 falls G vom Typ 1 ist und
für jede Produktion $\alpha \to \beta$ gilt $\alpha \in V$.

Typ 3 falls G vom Typ 2 ist und
für jede Produktion $\alpha \to \beta$ außer $S \to \epsilon$
gilt $\beta \in \Sigma \cup \Sigma V$.

Offensichtlich gilt:

$$\text{Typ 3} \subset \text{Typ 2} \subset \text{Typ 1} \subset \text{Typ 0}$$

Grammatiken und Sprachklassen:

Typ 3	Rechtslineare Grammatik	Reguläre Sprachen

Grammatiken und Sprachklassen:

Typ 3	Rechtslineare Grammatik	Reguläre Sprachen
Typ 2	Kontextfreie Grammatik	Kontextfreie Sprachen

Grammatiken und Sprachklassen:

Typ 3	Rechtslineare Grammatik	Reguläre Sprachen
Typ 2	Kontextfreie Grammatik	Kontextfreie Sprachen
Typ 1	Kontextsensitive Grammatik	Kontextsens. Sprachen

Grammatiken und Sprachklassen:

Typ 3	Rechtslineare Grammatik	Reguläre Sprachen
Typ 2	Kontextfreie Grammatik	Kontextfreie Sprachen
Typ 1	Kontextsensitive Grammatik	Kontextsens. Sprachen
Typ 0	Phrasenstrukturgrammatik	Rekursiv aufzählbare Sprachen

Grammatiken und Sprachklassen:

Typ 3	Rechtslineare Grammatik	Reguläre Sprachen
Typ 2	Kontextfreie Grammatik	Kontextfreie Sprachen
Typ 1	Kontextsensitive Grammatik	Kontextsens. Sprachen
Typ 0	Phrasenstrukturgrammatik	Rekursiv aufzählbare Sprachen

Satz 2.13

$L(\text{Typ 3}) \subset L(\text{Typ 2}) \subset L(\text{Typ 1}) \subset L(\text{Typ 0})$

Grammatiken und Sprachklassen:

Typ 3	Rechtslineare Grammatik	Reguläre Sprachen
Typ 2	Kontextfreie Grammatik	Kontextfreie Sprachen
Typ 1	Kontextsensitive Grammatik	Kontextsens. Sprachen
Typ 0	Phrasenstrukturgrammatik	Rekursiv aufzählbare Sprachen

Satz 2.13

$L(\text{Typ 3}) \subset L(\text{Typ 2}) \subset L(\text{Typ 1}) \subset L(\text{Typ 0})$

Bemerkung Wir benutzen später eine etwas liberalere Definition von kontextfreien Grammatiken, die aber die gleiche Klasse von kontextfreien Sprachen erzeugt.

Das Wortproblem:

Gegeben: eine Grammatik G, ein Wort $w \in \Sigma^*$

Frage: Gilt $w \in L(G)$?

Das Wortproblem:

Gegeben: eine Grammatik G, ein Wort $w \in \Sigma^*$
Frage: Gilt $w \in L(G)$?

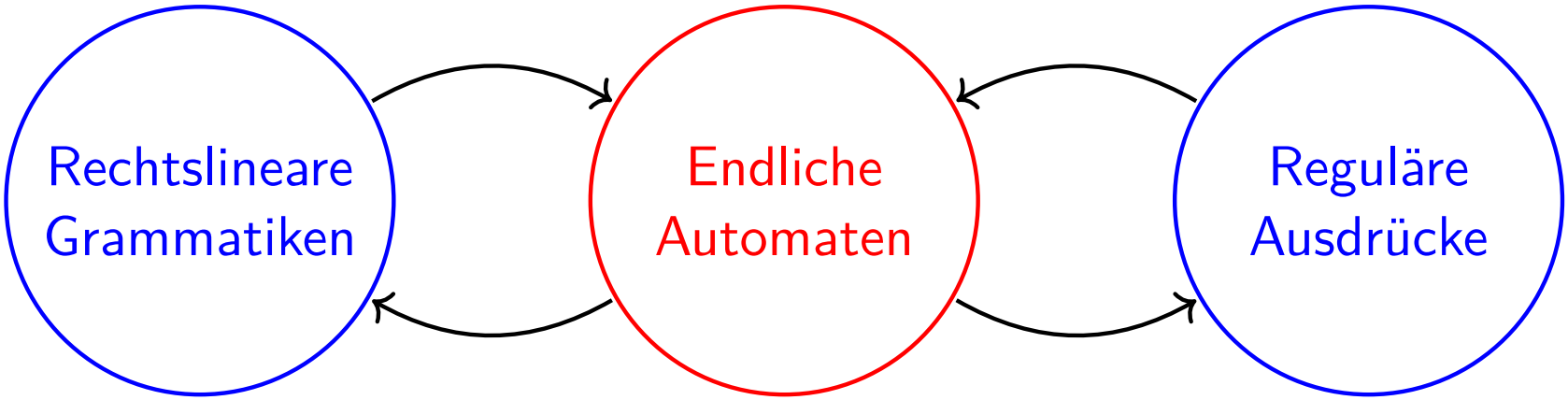
In den kommenden Wochen untersuchen wir das Wortproblem für Grammatiken vom Typ 3 und Typ 2.

Wir studieren Algorithmen, die für eine gegebene Grammatik G einen Automaten A_G konstruieren, und untersuchen ihre Laufzeit.

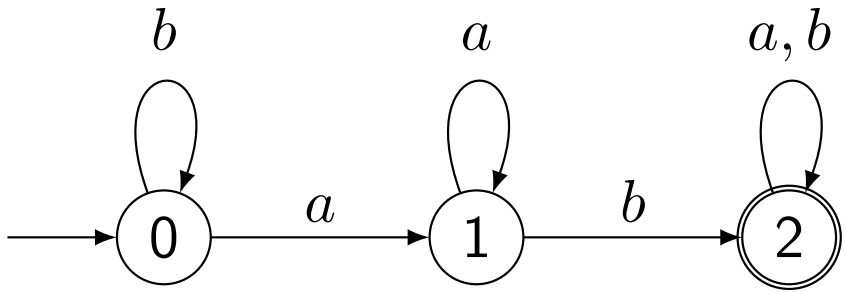
A_G ist eine abstrakte Beschreibung eines Programms zur Lösung des Wortproblems.

Der Algorithmus mit G als Eingabe und A_G als Ausgabe ist eine abstrakte Beschreibung eines Programms für die automatische Synthese von Recognizern (lex, yacc).

3. Reguläre Sprachen



3.1 Deterministische endliche Automaten



Eingabewort $baba \rightsquigarrow$ Zustandsfolge 0,0,1,2,2.

Erkannte Sprache: Menge der Wörter, die vom Startzustand in einen Endzustand führen.

Recognizer, die nur einmal das Wort durchläuft und in linearer Zeit es akzeptiert oder ablehnt.

Definition 3.1

Ein deterministischer endlicher Automat (*deterministic finite automaton*, DFA) $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,
- einem (endlichen) Eingabealphabet Σ,
- einer (totalen!) Übergangsfunktion $\delta : \ Q \times \Sigma \to Q$,
- einem Startzustand $q_0 \in Q$, und
- einer Menge $F \subseteq Q$ von Endzuständen (akzeptierenden Zust.)

Definition 3.2

Die von M akzeptierte Sprache ist

$$L(M) := \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\} \ ,$$

wobei $\hat{\delta} : Q \times \Sigma^* \to Q$ induktiv definiert ist durch

$$\begin{aligned} \hat{\delta}(q, \epsilon) &= q \\ \hat{\delta}(q, aw) &= \hat{\delta}(\delta(q, a), w) \quad \text{für } a \in \Sigma, w \in \Sigma^* \ . \end{aligned}$$

($\hat{\delta}(q, w)$ bezeichnet den Zustand, den man aus q mit w erreicht.)

Eine Sprache ist regulär gdw sie von einem DFA akzeptiert wird.

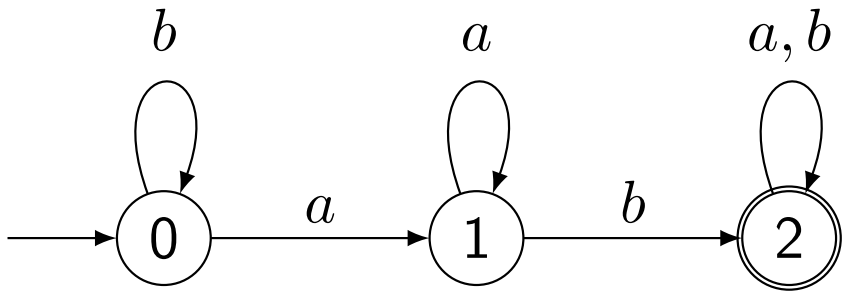
Eigenschaften von $\hat{\delta}$:

- $\hat{\delta}(q, a) = \delta(q, a)$.
- $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$.

Graphische Darstellung

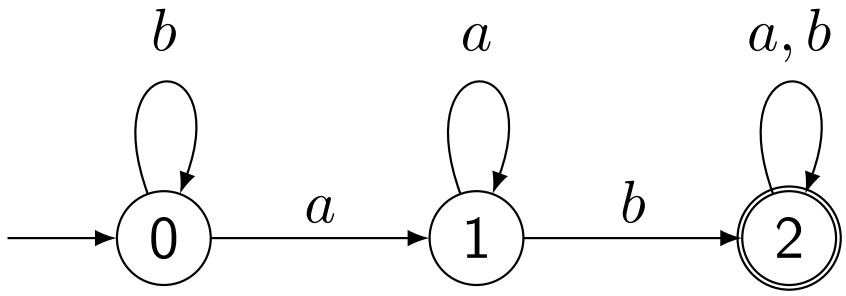
- Endliche Automaten können durch gerichtete und markierte Zustandsgraphen veranschaulicht werden:
  - Knoten $\widehat{=}$ Zuständen
  - Kanten $\widehat{=}$ Übergängen: $p \xrightarrow{a} q \;\widehat{=}\; \delta(p, a) = q$
- Der Anfangszustand wird durch einen Pfeil, Endzustände werden durch doppelte Kreise gekennzeichnet.

Beispiel 3.3



Die Sprache des DFAs ist die Menge aller Wörter über $\{a, b\}$, die ab enthalten.

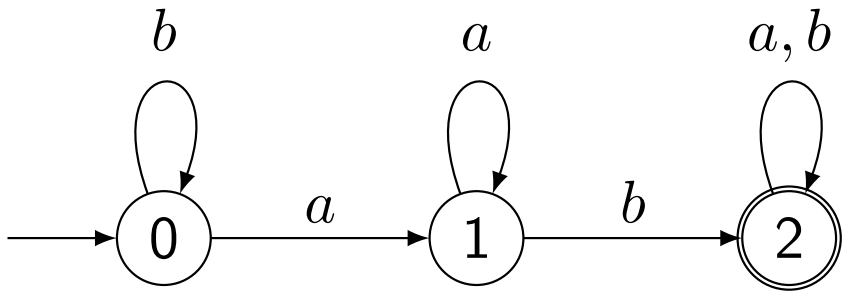
Beispiel 3.3



Die Sprache des DFAs ist die Menge aller Wörter über $\{a, b\}$, die ab enthalten.

- Jedes Wort, das akzeptiert wird, enthält ab.
  Sei w ein Wort, welches akzeptiert wird.
  Betrachte in der Zustandsfolge für w den Zeitpunkt, in dem Zustand 1 zum letzten Mal besucht wird (existiert weil die Folge in Zustand 2 endet).
  Unmittelbar davor wird a gelesen und unmittelbar danach b.

Beispiel 3.3



Die Sprache des DFAs ist die Menge aller Wörter über $\{a, b\}$, die ab enthalten.

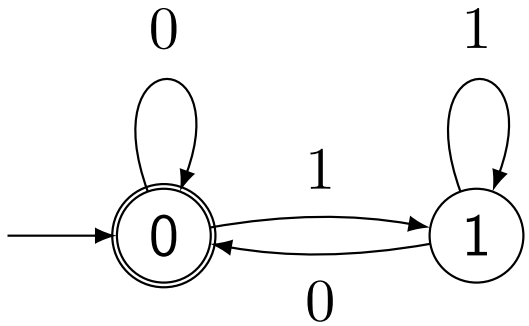
- Jedes Wort, das akzeptiert wird, enthält ab.
  Sei w ein Wort, welches akzeptiert wird.
  Betrachte in der Zustandsfolge für w den Zeitpunkt, in dem Zustand 1 zum letzten Mal besucht wird (existiert weil die Folge in Zustand 2 endet).
  Unmittelbar davor wird a gelesen und unmittelbar danach b.
- Jedes Wort, das ab enthält, wird akzeptiert.
  Für alle $q \in \{0, 1, 2\}$ gilt: $\hat{\delta}(q, ab) = 2$ und Zustand 2 kann nich verlassen werden.

Beispiel 3.4

Für $w \in \{0,1\}^*$ sei $\#w$ die von w binär repräsentierte Zahl, zB $\#100 = 4$.

Beispiel 3.4

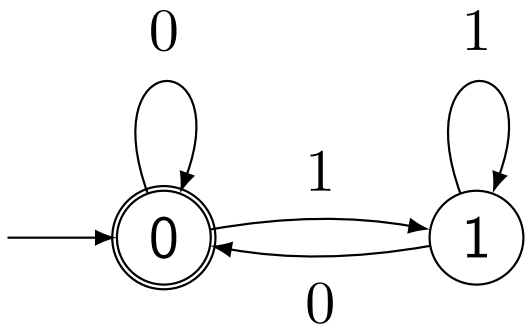
Für $w \in \{0,1\}^*$ sei $\#w$ die von w binär repräsentierte Zahl,
zB $\#100 = 4$.



Der DFA akzeptiert genau die $w \in \{0,1\}^*$ so dass $\#w$ gerade ist.

Beispiel 3.4

Für $w \in \{0,1\}^*$ sei $\#w$ die von w binär repräsentierte Zahl,
zB $\#100 = 4$.



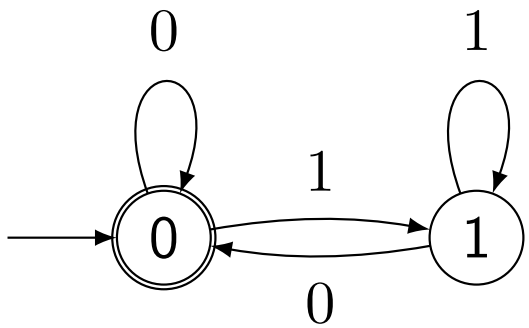
Der DFA akzeptiert genau die $w \in \{0,1\}^*$ so dass $\#w$ gerade ist.

Beweis:

Für alle $w \neq \epsilon$, $\#w$ ist gerade gdw w endet mit 0.

Beispiel 3.4

Für $w \in \{0,1\}^*$ sei $\#w$ die von w binär repräsentierte Zahl,
zB $\#100 = 4$.



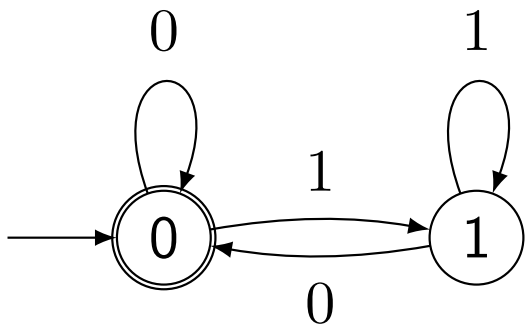
Der DFA akzeptiert genau die $w \in \{0,1\}^*$ so dass $\#w$ gerade ist.

Beweis:

Für alle $w \neq \epsilon$, $\#w$ ist gerade gdw w endet mit 0.
1. $\hat{\delta}(0, w0) = \delta(\hat{\delta}(0, w), 0) = 0 \in F$, daher $w0 \in L(A)$

Beispiel 3.4

Für $w \in \{0,1\}^*$ sei $\#w$ die von w binär repräsentierte Zahl,
zB $\#100 = 4$.



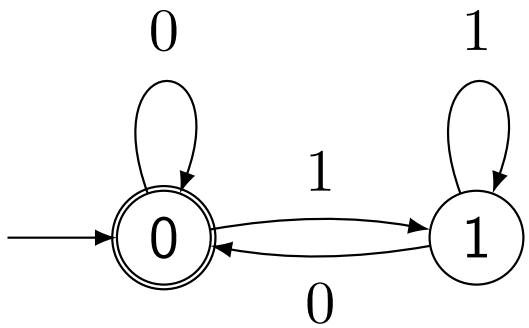
Der DFA akzeptiert genau die $w \in \{0,1\}^*$ so dass $\#w$ gerade ist.

Beweis:

Für alle $w \neq \epsilon$, $\#w$ ist gerade gdw w endet mit 0.
1. $\hat{\delta}(0, w0) = \delta(\hat{\delta}(0,w),0) = 0 \in F$, daher $w0 \in L(A)$
2. $\hat{\delta}(0, w1) = \delta(\hat{\delta}(0,w),1) = 1 \notin F$, daher $w1 \notin L(A)$

Beispiel 3.4

Für $w \in \{0,1\}^*$ sei $\#w$ die von w binär repräsentierte Zahl,
zB $\#100 = 4$.

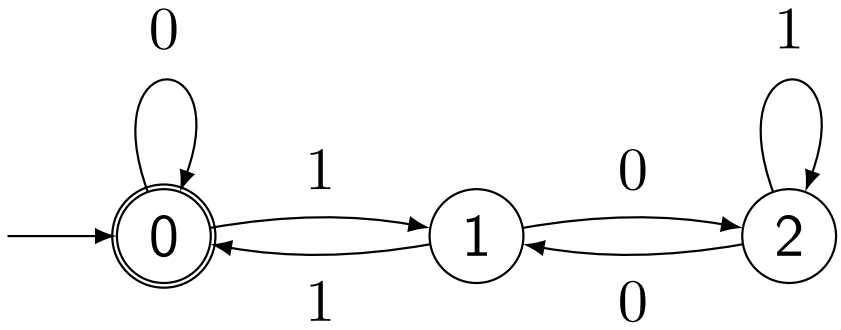


Der DFA akzeptiert genau die $w \in \{0,1\}^*$ so dass $\#w$ gerade ist.

Beweis:

Für alle $w \neq \epsilon$, $\#w$ ist gerade gdw w endet mit 0.
1. $\hat{\delta}(0, w0) = \delta(\hat{\delta}(0,w),0) = 0 \in F$, daher $w0 \in L(A)$
2. $\hat{\delta}(0, w1) = \delta(\hat{\delta}(0,w),1) = 1 \notin F$, daher $w1 \notin L(A)$
Für $w = \epsilon$, $\hat{\delta}(0,w) = 0$, daher $\epsilon \in L(A)$ und $\#\epsilon = 0$. □

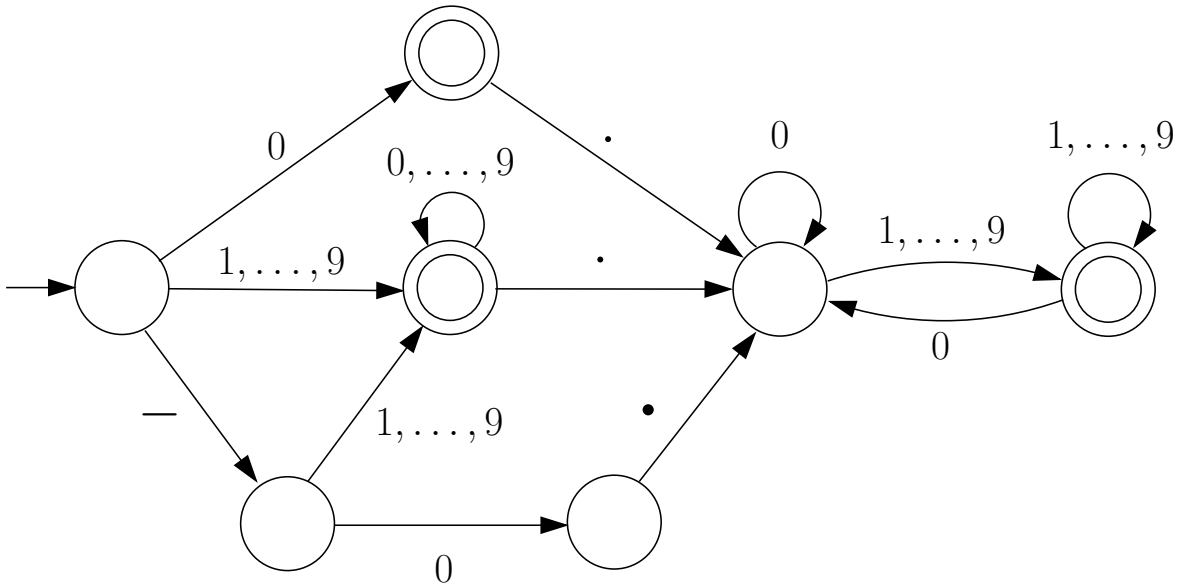
Beispiel 3.5



Für $w \in \{0,1\}^*$ sei $\#w$ die von w binär repräsentierte Zahl, zB $\#100 = 4$.

Der DFA akzeptiert genau die Wörter $w \in \{0,1\}^*$ mit: $\#w$ ist ein vielfaches von 3.

Beispiel 3.6



Ein DFA für die Gleitkommazahlen

3.2 Von rechtslinearen Grammatiken zu DFA (und zurück)

Rechtslineare Grammatik: Produktionen der Gestalt $A \to a$ und $A \to aB$

Wir zeigen:

- Für jede rechtslineare Grammatik G gibt es einen DFA M mit $L(M) = L(G)$.
- Für jeden DFA M gibt es eine rechtslineare Grammatik G mit $L(G) = L(M)$.

3.2 Von rechtslinearen Grammatiken zu DFA (und zurück)

Rechtslineare Grammatik: Produktionen der Gestalt $A \to a$ und $A \to aB$

Wir zeigen:

- Für jede rechtslineare Grammatik G gibt es einen DFA M mit $L(M) = L(G)$.
- Für jeden DFA M gibt es eine rechtslineare Grammatik G mit $L(G) = L(M)$.

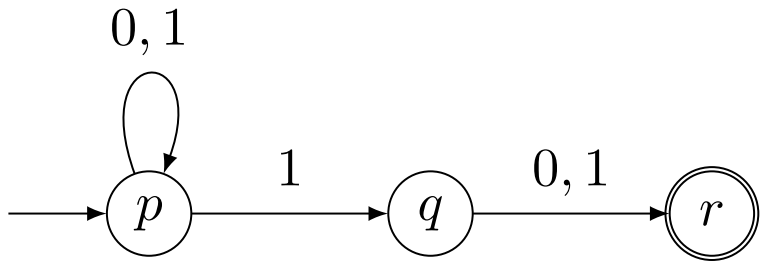
Wir führen nichtdeterministische endliche Automaten (NFA) als "Zwischenschritt" ein. Wir zeigen:

- Für jede rechtslineare Grammatik G gibt es einen NFA N mit $L(N) = L(G)$.
- Für jeden NFA N gibt es einen DFA M mit $L(M) = L(N)$.
- Für jeden DFA M gibt es eine rechtslineare Grammatik G mit $L(G) = L(M)$.

NFA sind eine Verallgemeinerung von DFA: Aus einem Zustand känn es $0, 1, 2, \ldots$ Übergänge mit derselben Beschriftung geben.

Beispiel 3.7

Erkennung der Binärzahlen, deren vorlezte Ziffer 1 ist:



Eingabewort $0111 \rightsquigarrow$ Mehrere Zustandsfolgen:

$$p, p, p, p, p \qquad p, p, p, p, q \qquad p, p, p, q, r$$

Wort wird akzeptiert gdw mindestens eine dieser Zustandfolgen zu einem Endzustand führt.
Intuitive Vorstellung: Der Automat "rät" den richtigen Weg.

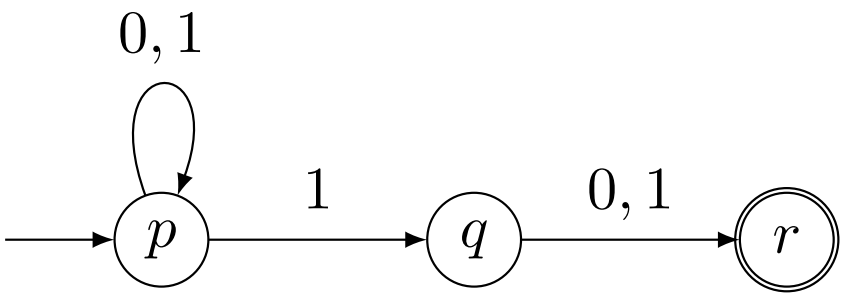
NFA nicht nutzlich als Recognizer, aber als Zwischenschritt oder Datenstruktur.

Definition 3.8

Ein nichtdeterministischer endlicher Automat (*nondeterministic finite automaton*, NFA) ist ein 5-Tupel $N = (Q, \Sigma, \delta, q_0, F)$, so dass

- Q, Σ, q_0 und F sind wie bei einem DFA
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$
  $\mathcal{P}(Q)$ = Menge aller Teilmengen von $Q = 2^Q$.
  Alternative: Relation $\delta \subseteq Q \times \Sigma \times Q$.

Beispiel



δ	0	1
p	$\{p\}$	$\{p, q\}$
q	$\{r\}$	$\{r\}$
r	$\emptyset$	$\emptyset$

Erweiterung von δ : $Q \times \Sigma \to \mathcal{P}(Q)$

auf $\bar{\delta} : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$ definiert durch

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

Es folgt: $\hat{\bar{\delta}} : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$

Erweiterung von δ : $Q \times \Sigma \to \mathcal{P}(Q)$

auf $\bar{\delta} : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$ definiert durch

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

Es folgt: $\hat{\bar{\delta}} : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$

Intuition: $\hat{\bar{\delta}}(S, w)$ ist Menge aller Zustände die sich von einem Zustand in S aus mit w erreichen lassen.

Erweiterung von δ : $Q \times \Sigma \to \mathcal{P}(Q)$

auf $\bar{\delta} : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$ definiert durch

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

Es folgt: $\hat{\bar{\delta}} : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$

Intuition: $\hat{\bar{\delta}}(S, w)$ ist Menge aller Zustände die sich von einem Zustand in S aus mit w erreichen lassen.

Die von $N = (Q, \Sigma, \delta, q_0, F)$ akzeptierte Sprache ist

$$L(N) := \{w \in \Sigma^* \mid \hat{\bar{\delta}}(\{q_0\}, w) \cap F \neq \emptyset\}$$

Erweiterung von δ : $Q \times \Sigma \to \mathcal{P}(Q)$

auf $\bar{\delta} : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$ definiert durch

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

Es folgt: $\hat{\bar{\delta}} : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$

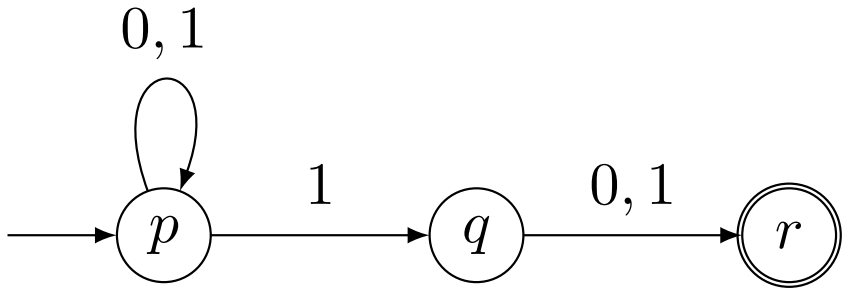
Intuition: $\hat{\bar{\delta}}(S, w)$ ist Menge aller Zustände die sich von einem Zustand in S aus mit w erreichen lassen.

Die von $N = (Q, \Sigma, \delta, q_0, F)$ akzeptierte Sprache ist

$$L(N) := \{w \in \Sigma^* \mid \hat{\bar{\delta}}(\{q_0\}, w) \cap F \neq \emptyset\}$$

Oft schreiben wir nur δ statt $\bar{\delta}$ und $\hat{\delta}$ statt $\hat{\bar{\delta}}$
(vgl. overloading in Programmiersprachen).

Beispiel



$$
\begin{aligned}
& \hat{\bar{\delta}}(\{p,q\},10) \\
= \quad & \hat{\bar{\delta}}(\bar{\delta}(\{p,q\},1),0) \\
= \quad & \hat{\bar{\delta}}(\delta(p,1) \cup \delta(q,1),0) \\
= \quad & \hat{\bar{\delta}}(\{p,q,r\},0) \\
= \quad & \bar{\delta}(\{p,q,r\},0) \\
= \quad & \delta(p,0) \cup \delta(q,0) \cup \delta(r,0) \\
= \quad & \{p\} \cup \{r\} \cup \emptyset \;=\; \{p,r\}
\end{aligned}
$$

Satz 3.9

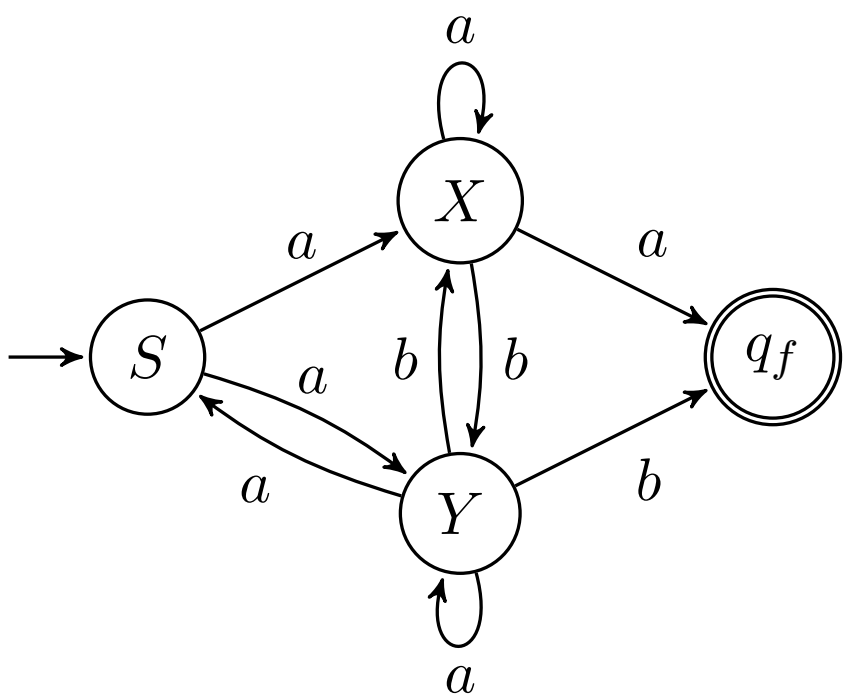
Für jede rechtslineare Grammatik G gibt es einen NFA M mit $L(G) = L(M)$.

Satz 3.9

Für jede rechtslineare Grammatik G gibt es einen NFA M mit $L(G) = L(M)$.

Beispiel

$S \to aX \mid aY \quad X \to aX \mid bY \mid a \quad Y \to aS \mid bX \mid aY \mid b$

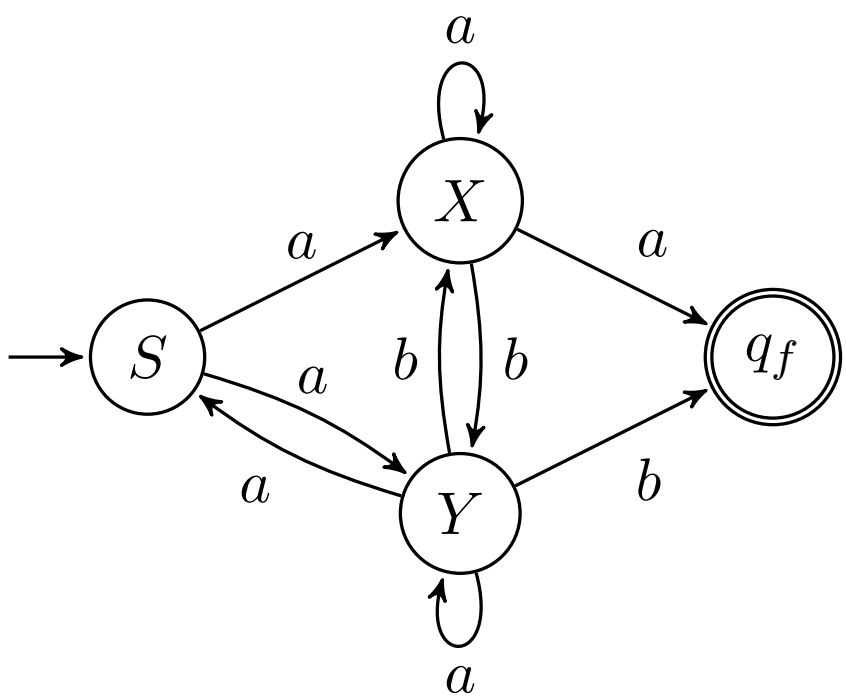


Satz 3.9

Für jede rechtslineare Grammatik G gibt es einen NFA M mit $L(G) = L(M)$.

Beispiel

$S \to aX \mid aY \quad X \to aX \mid bY \mid a \quad Y \to aS \mid bX \mid aY \mid b$



Aufgabe: Und wenn $S \to aX \mid aY \mid \epsilon$?

Beweis:

Sei $G = (V, \Sigma, P, S)$ eine rechtslineare Grammatik ohne die Produktion $S \to \epsilon$. Definiere den NFA $A = (Q, \Sigma, \delta, q_0, F)$ mit

- $Q = V \cup \{q_f\}$ (wobei $q_f \notin V$)
- $Y \in \delta(X, a)$ gdw $(X \to aY) \in P$
- $q_f \in \delta(X, a)$ gdw $(X \to a) \in P$
- $q_0 = S$
- $F = \{q_f\}$

Beweis:

Sei $G = (V, \Sigma, P, S)$ eine rechtslineare Grammatik ohne die Produktion $S \to \epsilon$. Definiere den NFA $A = (Q, \Sigma, \delta, q_0, F)$ mit

- $Q = V \cup \{q_f\}$ (wobei $q_f \notin V$)
- $Y \in \delta(X, a)$ gdw $(X \to aY) \in P$
- $q_f \in \delta(X, a)$ gdw $(X \to a) \in P$
- $q_0 = S$
- $F = \{q_f\}$

Es gilt (Induktion über n):

$$S \to a_1 X_1 \to_G a_1 a_2 X_2 \to_G \cdots \to_G a_1 \ldots a_{n-1} X_{n-1} \to_G a_1 \ldots a_n$$

ist eine Ableitung von G gdw

$$q_f \in \delta(S, a_1 \ldots a_n) \ .$$

Damit gilt $L(G) = L(A)$. (Fall mit $S \to \epsilon$: Aufgabe)

Lösung der Aufgabe: Wenn die Grammatik die Produktion $S \to \epsilon$ enthält, dann setze $F = \{S, q_f\}$.

Aufgabe: Sei $G = (V, T, P, S)$ eine rechtslineare Grammatik, die die Produktion $S \to \epsilon$ nicht enthält. Definiere die Grammatik $G' = (V, T, P \cup \{S \to \epsilon\}, S)$ (d.h., wir fügen $S \to \epsilon$ zu P hinzu).

- Gilt immer $L(G') = L(G) \cup \{\epsilon\}$?
- Geben Sie eine Grammatik G'' mit $L(G'') = L(G) \cup \{\epsilon\}$ an.

Satz 3.10

Für jeden NFA N gibt es einen DFA M mit $L(N) = L(M)$.

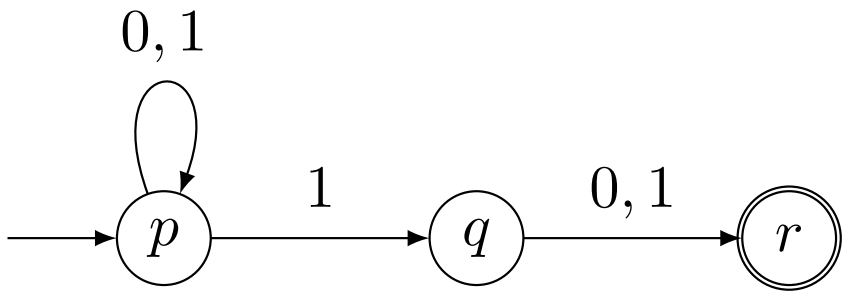
.

Satz 3.10

Für jeden NFA N gibt es einen DFA M mit $L(N) = L(M)$.

.

Beispiel



Beweis:

Sei $N = (Q, \Sigma, \delta, q_0, F)$ ein NFA.

Definiere den DFA $M = (\mathcal{P}(Q), \Sigma, \bar{\delta}, \{q_0\}, F_M)$:

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

Beweis:

Sei $N = (Q, \Sigma, \delta, q_0, F)$ ein NFA.

Definiere den DFA $M = (\mathcal{P}(Q), \Sigma, \bar{\delta}, \{q_0\}, F_M)$:

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

Dann gilt:

$$\begin{array}{lcll} w \in L(N) & \Leftrightarrow & \hat{\bar{\delta}}(\{q_0\}, w) \cap F \neq \emptyset & \text{Def.} \\ & \Leftrightarrow & \hat{\bar{\delta}}(\{q_0\}, w) \in F_M & \text{Def.} \\ & \Leftrightarrow & w \in L(M) & \text{Def.} \end{array}$$

□

Beweis:

Sei $N = (Q, \Sigma, \delta, q_0, F)$ ein NFA.

Definiere den DFA $M = (\mathcal{P}(Q), \Sigma, \bar{\delta}, \{q_0\}, F_M)$:

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

Dann gilt:

$$\begin{aligned} w \in L(N) &\Leftrightarrow \hat{\bar{\delta}}(\{q_0\}, w) \cap F \neq \emptyset && \text{Def.} \\ &\Leftrightarrow \hat{\bar{\delta}}(\{q_0\}, w) \in F_M && \text{Def.} \\ &\Leftrightarrow w \in L(M) && \text{Def.} \end{aligned}$$

□

Dies nennt man die Potenzmengen- oder Teilmengenkonstruktion.

In der Praxis werden nur die aus q_0 erreichbaren Zuständen konstruiert.

Trotzdem: Für einen NFA mit n Zuständen kann der entsprechende DFA bis zu 2^n Zustände haben.

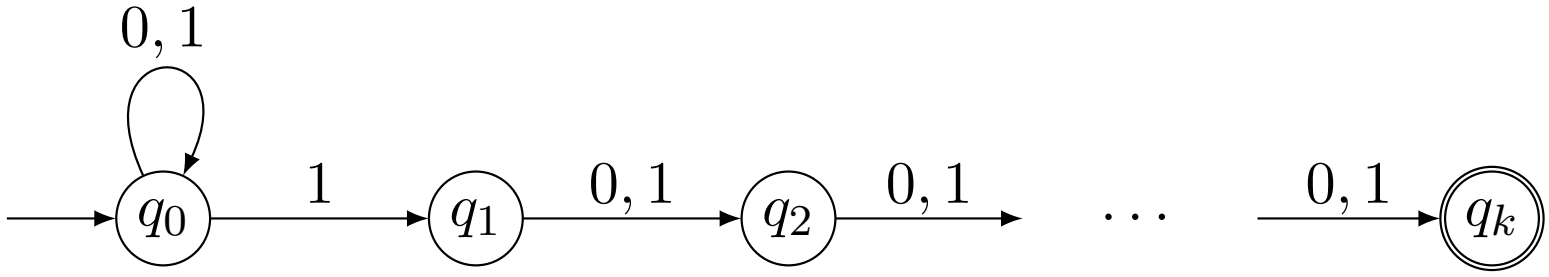
Beispiel 3.11

$$L_k := \{w \in \{0,1\}^* \mid \text{das } k\text{-letzte Bit von } w \text{ ist } 1\}$$

Beispiel 3.11

$$L_k := \{w \in \{0,1\}^* \mid \text{das } k\text{-letzte Bit von } w \text{ ist } 1\}$$

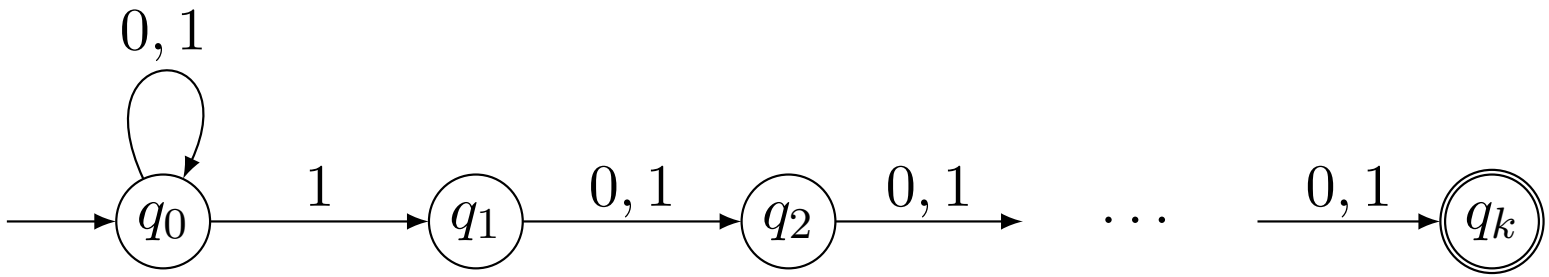
Ein NFA für diese Sprache:



Beispiel 3.11

$$L_k := \{w \in \{0,1\}^* \mid \text{das } k\text{-letzte Bit von } w \text{ ist } 1\}$$

Ein NFA für diese Sprache:

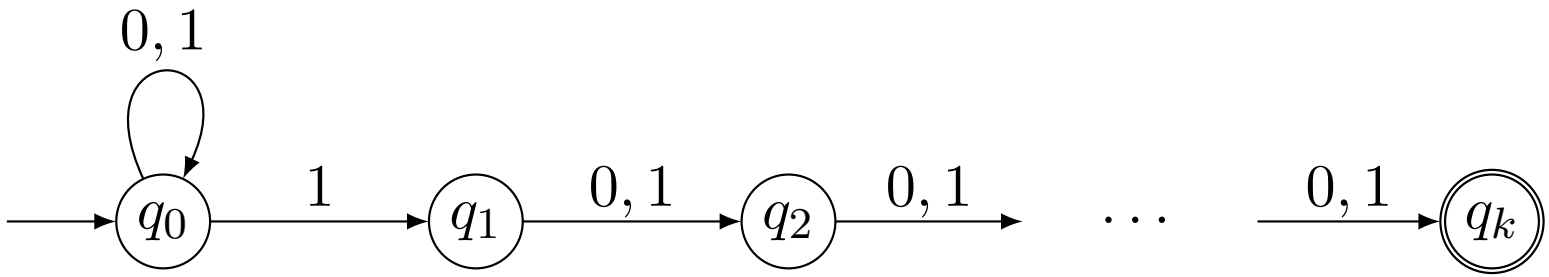


Die Potenzmengenkonstruktion liefert einen DFA für L_k mit 2^k Zuständen. Geht es kompakter?

Beispiel 3.11

$$L_k := \{w \in \{0,1\}^* \mid \text{das } k\text{-letzte Bit von } w \text{ ist } 1\}$$

Ein NFA für diese Sprache:



Die Potenzmengenkonstruktion liefert einen DFA für L_k mit 2^k Zuständen. Geht es kompakter?

Lemma 3.12

Jeder DFA M mit $L(M) = L_k$ hat mindestens 2^k Zuständen.

Im *schlimmsten* Fall ist ein exponentieller Sprung unvermeidlich.

Beweis:

Indirekt. Sei M ein DFA mit $< 2^k$ Zuständen, so dass $L(M) = L_k$.

Beweis:

Indirekt. Sei M ein DFA mit $< 2^k$ Zuständen, so dass $L(M) = L_k$.

- Zuerst zeigen wir für alle $w_1, w_2 \in \{0,1\}^k$:
  wenn $w_1 \neq w_2$ dann $\hat{\delta}(q_0, w_1) \neq \hat{\delta}(q_0, w_2)$.
  Annahme: Es gibt $w_1, w_2 \in \{0,1\}^k$ mit $w_1 \neq w_2$ aber
  $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$. Wir leiten einen Widerspruch ab:

Beweis:

Indirekt. Sei M ein DFA mit $< 2^k$ Zuständen, so dass $L(M) = L_k$.

- Zuerst zeigen wir für alle $w_1, w_2 \in \{0,1\}^k$:
  wenn $w_1 \neq w_2$ dann $\hat{\delta}(q_0, w_1) \neq \hat{\delta}(q_0, w_2)$.
  Annahme: Es gibt $w_1, w_2 \in \{0,1\}^k$ mit $w_1 \neq w_2$ aber $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$. Wir leiten einen Widerspruch ab:
- Sei $w_1 = wa_i \dots a_k$ und $w_2 = wb_i \dots b_k$ mit $a_i \neq b_i$

  OE sei $a_i = 1$, $b_i = 0$.

  Es gilt einerseits: $w_1 0^{i-1} = wa_i \dots a_k 0^{i-1} \in L_k$
  $w_2 0^{i-1} = wb_i \dots b_k 0^{i-1} \notin L_k$

  Aber es gilt auch: $\hat{\delta}(q_0, w_1 0^{i-1}) = \hat{\delta}(\hat{\delta}(q_0, w_1), 0^{i-1}) =$
  $\hat{\delta}(\hat{\delta}(q_0, w_2), 0^{i-1}) = \hat{\delta}(q_0, w_2 0^{i-1})$ ↯

Beweis:

Indirekt. Sei M ein DFA mit $< 2^k$ Zuständen, so dass $L(M) = L_k$.

- Zuerst zeigen wir für alle $w_1, w_2 \in \{0,1\}^k$:
  wenn $w_1 \neq w_2$ dann $\hat{\delta}(q_0, w_1) \neq \hat{\delta}(q_0, w_2)$.
  Annahme: Es gibt $w_1, w_2 \in \{0,1\}^k$ mit $w_1 \neq w_2$ aber $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$. Wir leiten einen Widerspruch ab:
- Sei $w_1 = wa_i \dots a_k$ und $w_2 = wb_i \dots b_k$ mit $a_i \neq b_i$

  OE sei $a_i = 1$, $b_i = 0$.

  Es gilt einerseits: $w_1 0^{i-1} = wa_i \dots a_k 0^{i-1} \in L_k$
  $w_2 0^{i-1} = wb_i \dots b_k 0^{i-1} \notin L_k$

  Aber es gilt auch: $\hat{\delta}(q_0, w_1 0^{i-1}) = \hat{\delta}(\hat{\delta}(q_0, w_1), 0^{i-1}) =$
  $\hat{\delta}(\hat{\delta}(q_0, w_2), 0^{i-1}) = \hat{\delta}(q_0, w_2 0^{i-1})$ ↯
- Es folgt: M hat mindestens 2^k Zustände.

□

Satz 3.13

Für jeden DFA M gibt es eine rechtslineare Grammatik G mit $L(M) = L(G)$.

Beweis:

Sei $M = (Q, \Sigma, \delta, q_0, F)$. Die Grammatik $G = (V, T, P, S)$ mit

- $V = Q$, $T = \Sigma$, $S = q_0$,
- $(q_1 \to aq_2) \in P$ gdw $\delta(q_1, a) = q_2$,
- $(q_1 \to a) \in P$ gdw $\delta(q_1, a) \in F$, und
- $(q_0 \to \epsilon) \in P$ gdw $q_0 \in F$,

ist von Typ 3 und erfüllt $L(M) = L(G)$. □

3.3 NFAs mit ϵ-Übergängen

Grammatiken von Programmiersprachen enthalten viele Produktionen der Gestalt $A \to B$.

Beispiel 3.14

Ein kleines Fragment der Grammatik von Java:

```
Literal:
    IntegerLiteral
    FloatingPointLiteral
    CharacterLiteral
    StringLiteral
    BooleanLiteral
    NullLiteral
```

Wir suchen Recognizer für Typ 3 Grammatiken, die auch solche Produktionen enthalten.

Definition 3.15

Ein NFA mit ϵ-Übergängen (auch ϵ-NFA) ist ein NFA mit einem speziellen Symbol $\epsilon \notin \Sigma$ und mit

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q) \ .$$

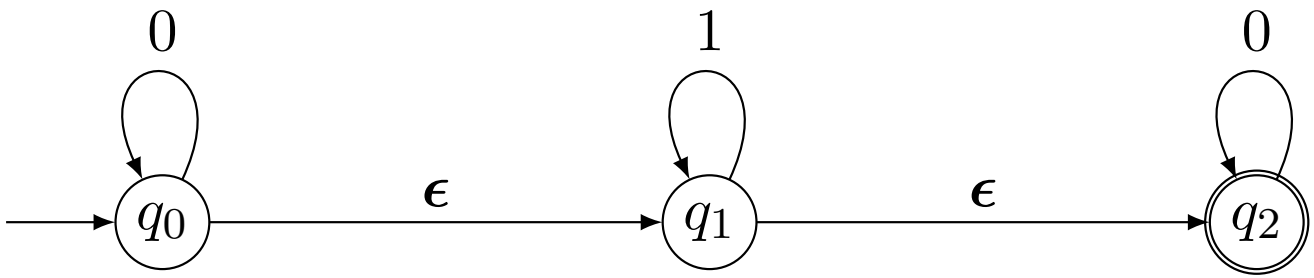
Ein ϵ-Übergang darf ausgeführt werden, ohne dass ein Eingabezeichen gelesen wird.

Definition 3.15

Ein NFA mit ϵ-Übergängen (auch ϵ-NFA) ist ein NFA mit einem speziellen Symbol $\epsilon \notin \Sigma$ und mit

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q) \ .$$

Ein ϵ-Übergang darf ausgeführt werden, ohne dass ein Eingabezeichen gelesen wird.



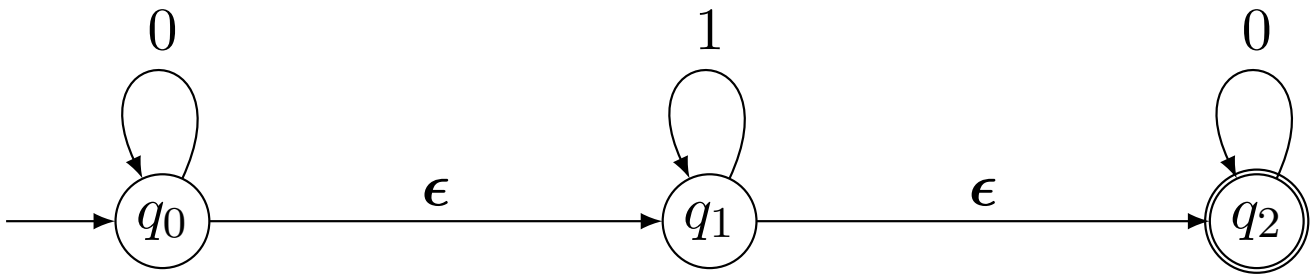
Akzeptiert: ϵ, 00, 11, ... Nicht akzeptiert: 101, ...

Definition 3.15

Ein NFA mit ϵ-Übergängen (auch ϵ-NFA) ist ein NFA mit einem speziellen Symbol $\epsilon \notin \Sigma$ und mit

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q) \ .$$

Ein ϵ-Übergang darf ausgeführt werden, ohne dass ein Eingabezeichen gelesen wird.



Akzeptiert: ϵ, 00, 11, ... Nicht akzeptiert: 101, ...

Bemerkung: $\epsilon \neq \epsilon$; ϵ ist ein einzelnes Symbol, ϵ das leere Wort.

Lemma 3.16

Für jeden ϵ-NFA N gibt es einen NFA N' mit $L(N) = L(N')$.

Lemma 3.16

Für jeden ϵ-NFA N gibt es einen NFA N' mit $L(N) = L(N')$.

Beweis:

Seie $N = (Q, \Sigma, \delta, q_0, F)$ ein ϵ-NFA. Wir definieren den NFA $N' = (Q, \Sigma, \delta', q_0, F')$ mit folgenden Definitionen für δ' und F':

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

$$\delta'(q, a) := \bigcup_{i \geq 0, j \geq 0} \hat{\delta}(\{q\}, \epsilon^i a \epsilon^j) \; .$$

- Falls N das leere Wort ϵ akzeptiert, also falls

$$\exists i \geq 0. \; \hat{\delta}(\{q_0\}, \epsilon^i) \cap F \neq \emptyset$$

dann setze $F' := F \cup \{q_0\}$, sonst setze $F' := F$.

□

Lemma 3.16

Für jeden ϵ-NFA N gibt es einen NFA N' mit $L(N) = L(N')$.

Beweis:

Seie $N = (Q, \Sigma, \delta, q_0, F)$ ein ϵ-NFA. Wir definieren den NFA $N' = (Q, \Sigma, \delta', q_0, F')$ mit folgenden Definitionen für δ' und F':

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

$$\delta'(q, a) := \bigcup_{i \geq 0, j \geq 0} \hat{\delta}(\{q\}, \epsilon^i a \epsilon^j) \; .$$

- Falls N das leere Wort ϵ akzeptiert, also falls

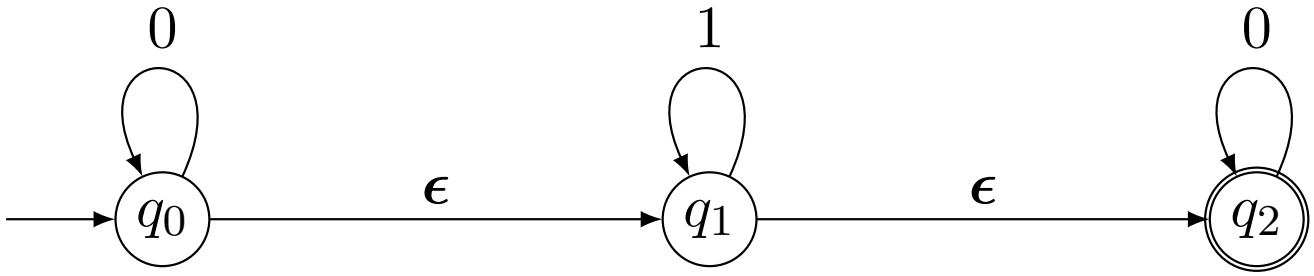
$$\exists i \geq 0. \; \hat{\delta}(\{q_0\}, \epsilon^i) \cap F \neq \emptyset$$

dann setze $F' := F \cup \{q_0\}$, sonst setze $F' := F$.

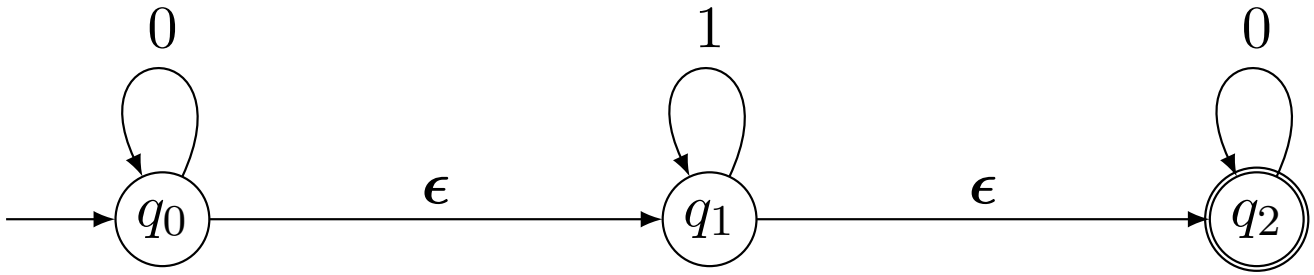
□

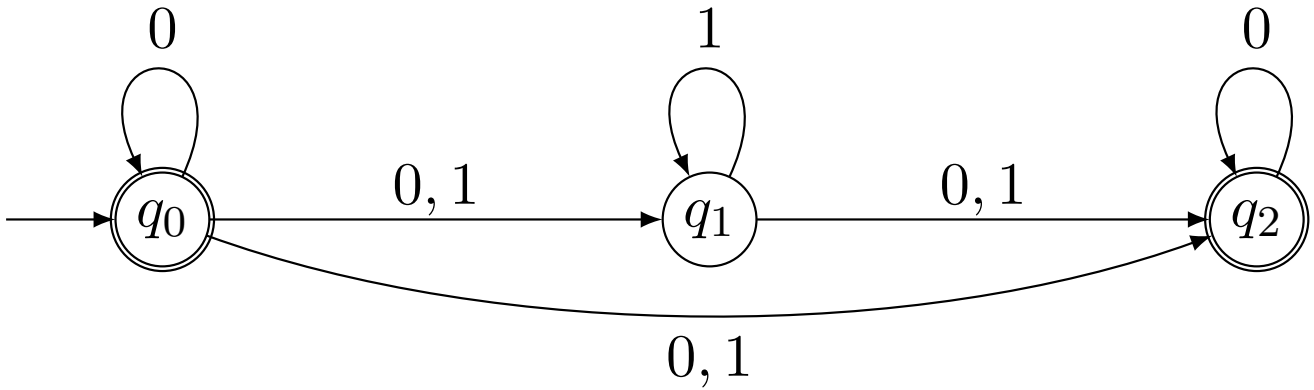
Ab jetzt: "ϵ-Übergang" und "ϵ-NFA".

Beispiel 3.17



Beispiel 3.17





Fazit: Die Automatentypen

- DFA
- NFA
- ϵ-NFA

sind gleich mächtig und Erkennen die Sprachen der Grammatiken mit Produktionen folgender Gestalt:

- $X \to aY$
- $X \to a$
- $X \to Y$
- $X \to \epsilon$

3.4 Reguläre Ausdrücke

Reguläre Ausdrücke sind eine alternative Notation für die Definition von formalen Sprachen.

Werden oft in Kombination mit Grammatiken verwendet.

3.4 Reguläre Ausdrücke

Reguläre Ausdrücke sind eine alternative Notation für die Definition von formalen Sprachen.

Werden oft in Kombination mit Grammatiken verwendet.

Beispiel 3.18

Ein Fragment der Grammatik für Python (`https://docs.python.org/2/reference/grammar.html`)

```
simple_stmt: small_stmt (';' small_stmt)* [';'] NEWLINE
```

Steht für eine unendliche Menge von Produktionen:

```
simple_stmt → small_stmt NEWLINE
simple_stmt → small_stmt ';' NEWLINE
simple_stmt → small_stmt ';' small_stmt NEWLINE
simple_stmt → small_stmt ';' small_stmt ';' NEWLINE
...
```

Definition 3.19

Reguläre Ausdrücke (*regular expressions*, REs) sind induktiv definiert:

- $\emptyset$ ist ein regulärer Ausdruck.
- ϵ ist ein regulärer Ausdruck.
- Für jedes $a \in \Sigma$ ist a ist ein regulärer Ausdruck.
- Wenn α und β reguläre Ausdrücke sind, dann auch
  - $\alpha\beta$
  - $\alpha \mid \beta$ (oft $\alpha + \beta$ geschrieben)
  - α^*
- Nichts sonst ist ein regulärer Ausdruck.

Definition 3.19

Reguläre Ausdrücke (*regular expressions*, REs) sind induktiv definiert:

- $\emptyset$ ist ein regulärer Ausdruck.
- ϵ ist ein regulärer Ausdruck.
- Für jedes $a \in \Sigma$ ist a ist ein regulärer Ausdruck.
- Wenn α und β reguläre Ausdrücke sind, dann auch
  - $\alpha\beta$
  - $\alpha \mid \beta$ (oft $\alpha + \beta$ geschrieben)
  - α^*
- Nichts sonst ist ein regulärer Ausdruck.

$*$ ist die Kleene'sche Iteration (Kleene iteration, Kleene star).

Bindungsstärke: * bindet stärker als Konkatenation stärker als $\mid$

- $ab^* = a(b^*) \neq (ab)^*$
- $ab \mid c = (ab) \mid c \neq a(b \mid c)$

Definition 3.20

Zu einem regulären Ausdruck γ ist die zugehörige Sprache $L(\gamma)$ rekursiv definiert:

- $L(\boldsymbol{\emptyset}) = \emptyset$
- $L(\boldsymbol{\epsilon}) = \{\epsilon\}$
- $L(a) = \{a\}$
- $L(\alpha\beta) = L(\alpha)L(\beta)$
- $L(\alpha \mid \beta) = L(\alpha) \cup L(\beta)$
- $L(\alpha^*) = L(\alpha)^*$

Beispiel 3.21

Sei das zugrunde liegende Alphabet $\Sigma = \{0, 1\}$.

- Alle Wörter, die mit 00 enden:

$$(0|1)^*00$$

Beispiel 3.21

Sei das zugrunde liegende Alphabet $\Sigma = \{0, 1\}$.

- Alle Wörter, die mit 00 enden:

$$(0|1)^*00$$

- Alle Wörter gerader Länge, in denen 0 und 1 alternieren:

$$(01)^* \mid (10)^*$$

Beispiel 3.21

Sei das zugrunde liegende Alphabet $\Sigma = \{0, 1\}$.

- Alle Wörter, die mit 00 enden:

$$(0|1)^*00$$

- Alle Wörter gerader Länge, in denen 0 und 1 alternieren:

$$(01)^* \mid (10)^*$$

- Alle Wörter, die eine gerade Anzahl von 1'en enthalten:

$$(0^*10^*1)^*0^*$$

Beispiel 3.21

Sei das zugrunde liegende Alphabet $\Sigma = \{0, 1\}$.

- Alle Wörter, die mit 00 enden:

$$(0|1)^*00$$

- Alle Wörter gerader Länge, in denen 0 und 1 alternieren:

$$(01)^* \mid (10)^*$$

- Alle Wörter, die eine gerade Anzahl von 1'en enthalten:

$$(0^*10^*1)^*0^*$$

- Alle Wörter, die die Binärdarstellung einer durch 3 teilbaren Zahl darstellen, also

$$0, 11, 110, 1001, 1100, 1111, 10010, \ldots$$

Beispiel 3.21

Sei das zugrunde liegende Alphabet $\Sigma = \{0, 1\}$.

- Alle Wörter, die mit 00 enden:

$$(0|1)^*00$$

- Alle Wörter gerader Länge, in denen 0 und 1 alternieren:

$$(01)^* \mid (10)^*$$

- Alle Wörter, die eine gerade Anzahl von 1'en enthalten:

$$(0^*10^*1)^*0^*$$

- Alle Wörter, die die Binärdarstellung einer durch 3 teilbaren Zahl darstellen, also

$$0, 11, 110, 1001, 1100, 1111, 10010, \ldots$$

  Hausaufgabe!

Beispiel 3.22

Gleitkommazahlen: $\Sigma = \{., -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$$(- \mid \boldsymbol{\epsilon})(DD^* \mid DD^*.D^* \mid D^*.DD^*)$$

wobei $D = (0|1|2|3|4|5|6|7|8|9)$

Beispiel 3.22

Gleitkommazahlen: $\Sigma = \{., -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$$(- \mid \boldsymbol{\epsilon})(DD^* \mid DD^*.D^* \mid D^*.DD^*)$$

wobei $D = (0|1|2|3|4|5|6|7|8|9)$

Version von Beispiel 3.6:

$$0 \mid (- \mid \boldsymbol{\epsilon})\big(ED^* \mid (0 \mid ED^*).D^*E\big)$$

wobei $D = (0|1|2|3|4|5|6|7|8|9)$ und $E = (1|2|3|4|5|6|7|8|9)$

Erweiterte reguläre Ausdrücke in UNIX:

$$
\begin{aligned}
\texttt{.} &= a_1|\ldots|a_n \text{ wobei } \Sigma = \{a_1, \ldots a_n\} \\
\texttt{[}a_1 \ldots a_n\texttt{]} &= a_1|\ldots|a_n \\
\texttt{[\^{}}a_1 \ldots a_n\texttt{]} &= b_1|\ldots|b_m \text{ wobei } \{b_1, \ldots, b_m\} = \Sigma \setminus \{a_1, \ldots a_n\} \\
\alpha\texttt{?} &= \epsilon|\alpha \\
\alpha\texttt{+} &= \alpha\alpha^* \\
\alpha\texttt{\{}n\texttt{\}} &= \alpha \ldots \alpha \ (n \text{ copies}) \\
&\ldots
\end{aligned}
$$

Für die Gleitkommazahlen (erste Version) erhalten wir z. B.

```
-?([0-9]+|[0-9]+\.[0-9]*|[0-9]*\.[0-9]+)
```

Satz 3.23 (Kleene 1956)

Eine Sprache $L \subseteq \Sigma^*$ ist genau dann durch einen regulären Ausdruck darstellbar, wenn sie regulär ist.

Für den Beweis definieren wir:

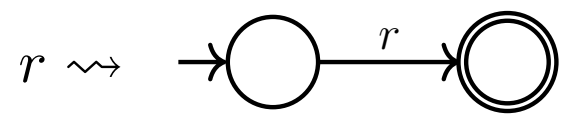
Definition 3.24

Ein NFA mit RE-Übergängen ist ein NFA M, dessen Übergänge mit regulären Ausdrücken über demselben Alphabet Σ beschriftet sind. Ein Wort $w \in \Sigma^*$ wird von M akzeptiert gdw M einen Pfad

$$q_0 \xrightarrow{\gamma_1} q_1 \xrightarrow{\gamma_2} q_2 \ \cdots \ q_{n-1} \xrightarrow{\gamma_n} q_n$$

mit folgende Eigenschaften enthält: q_0 ist der Anfangszustand, q_n ist Endzustand und $w \in L(\gamma_1\gamma_2 \cdots \gamma_n)$.

NFA mit ϵ-Übergängen sind ein Spezialfall. Reguläre Ausdrücke auch:



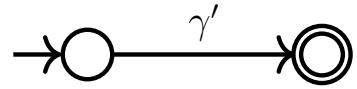
Beweis:

"$\Longrightarrow$": Sei γ ein regulärer Ausdruck. Wir konstruieren einen ϵ-NFA N mit $L(\gamma) = L(N)$ in zwei Schritten:

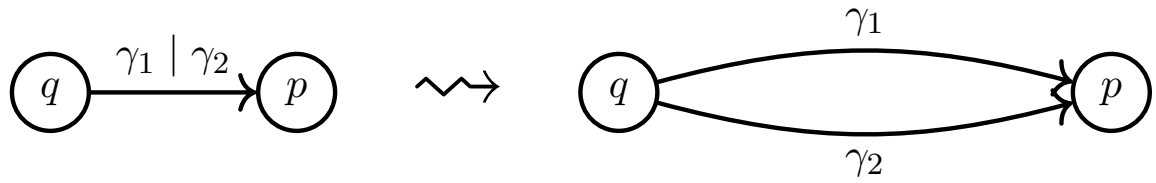
Schritt 1 (preprocessing): Wir wenden folgende Ersetzungsregeln so lange wie möglich auf γ an:

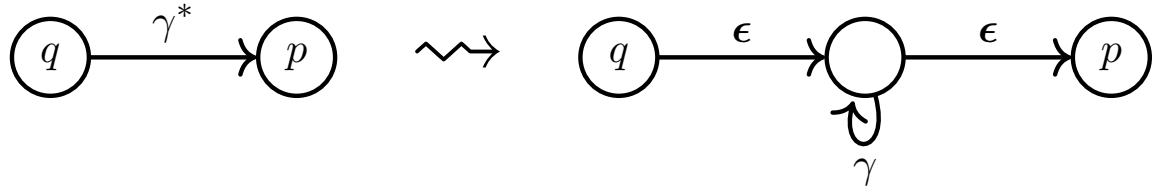
$$\gamma\,\emptyset \rightsquigarrow \emptyset \qquad \gamma \mid \emptyset \rightsquigarrow \gamma \qquad \emptyset^* \rightsquigarrow \epsilon$$
$$\emptyset\,\gamma \rightsquigarrow \emptyset \qquad \emptyset \mid \gamma \rightsquigarrow \gamma$$

Die Regeln erhalten die Sprache des regulären Ausdrucks.
Das Endergebnis γ' ist entweder $\emptyset$ (z.B. wenn $\gamma = \emptyset\emptyset$) oder ein regulärer Ausdruck ohne Vorkommnisse von $\emptyset$ (z.B. wenn $\gamma = a|\emptyset$).
Im ersten Fall setzen wir N als der NFA mit nur einen Zustand, keine Endzustände und keine Übergänge. Im zweiten Fall machen wir mit Schritt 2 weiter.

Schritt 2: Wir wenden folgende Transformationsregeln so lange wie möglich auf den Automaten  an.

Konkatenation: 

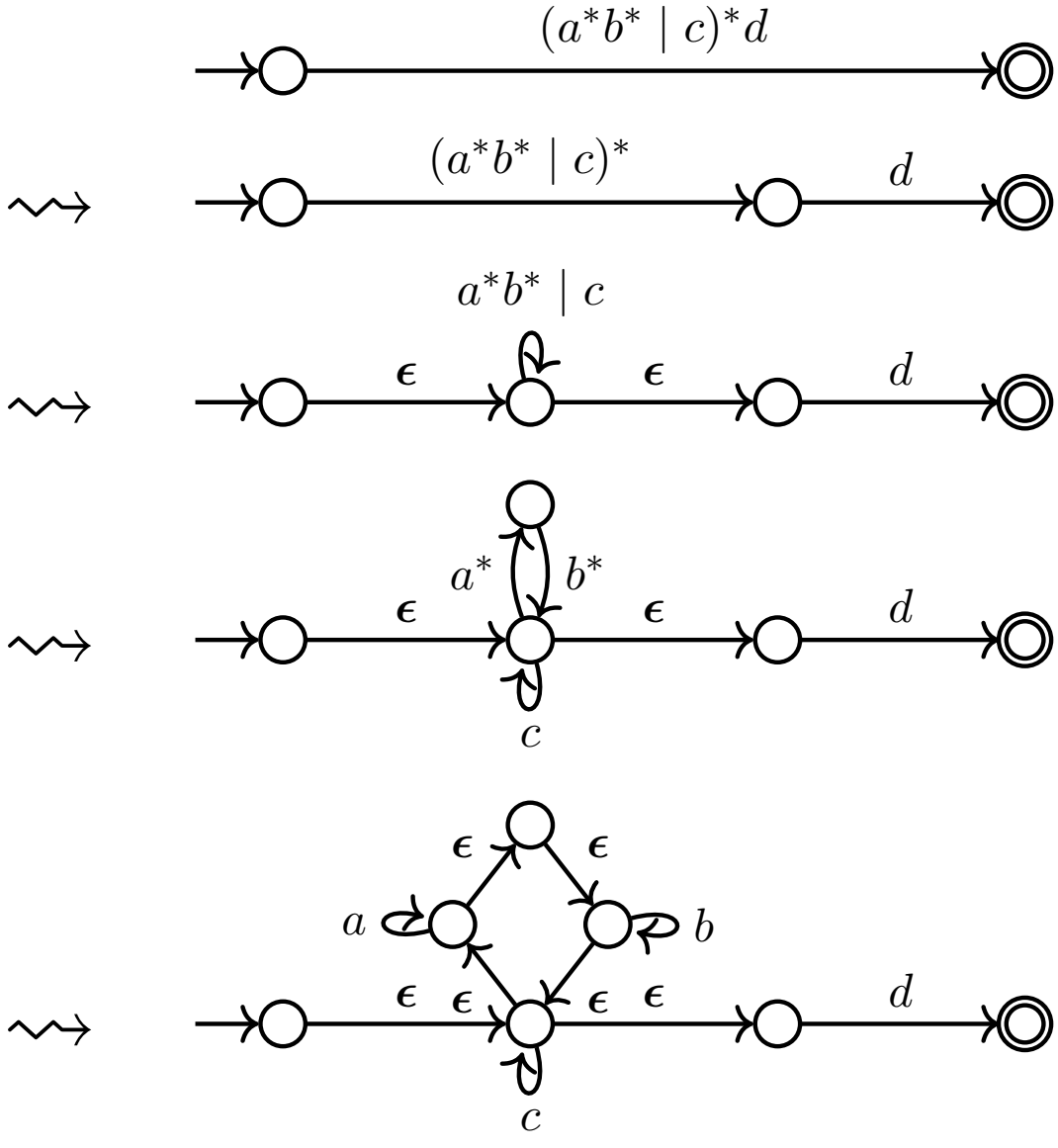
Auswahl: 

Iteration: 

Beachte: $q = p$ ist möglich und sowohl q wie auch p können Anfang oder Endzustände sein!.
Die Regeln erhalten die Sprache. (Übung!).
Das Endergebnis ist ein NFA mit ϵ-Übergängen.

Beispiel 3.25

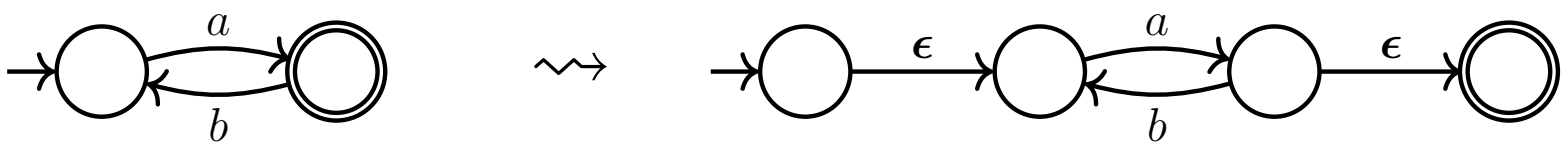
Betrachte den regulären Ausdruck $(a^*b^* \mid c)^*d$.



"$\Longleftarrow$": Sei $M = (Q, \Sigma, \delta, q_1, F)$ ein NFA mit ϵ-Übergänge.
Wir konstruieren einen RE γ mit $L(M) = L(\gamma)$ in zwei Schritten:

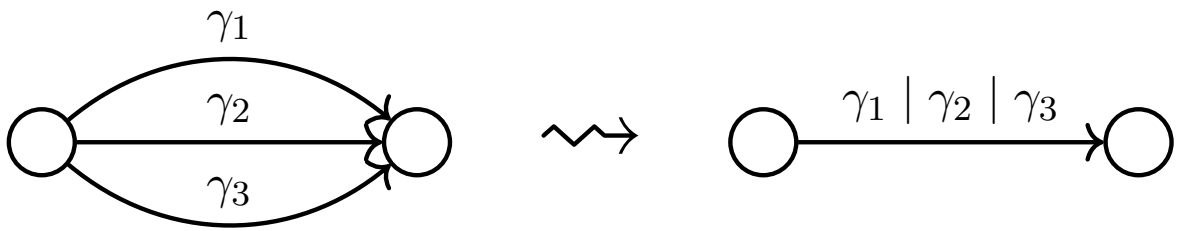
Schritt 1 (preprocessing): Wir transformieren M in einen äquivalenten Automaten a) mit höchstens einem Endzustand, b) ohne in den Anfangszustand einkommende Übergänge und c) ohne aus dem Endzustand ausgehende Übergänge:

- Hat q_1 Eingangsübergänge, dann fügen wir einen neuen Zustand q_0 hinzu sowie einen ϵ-Übergäng von q_0 nach q_1 und setzen q_0 als neuer Anfangsszustand.
- Enthält F mehr als einen Zustand oder einen Zustand mit Ausgangsübergangen, dann fügen wir einen neuen Zustand q_e sowie ϵ-Übergänge vom jedem Zustand von F zu q_e und setzen q_e als (einizigen) neuen Endzustand.

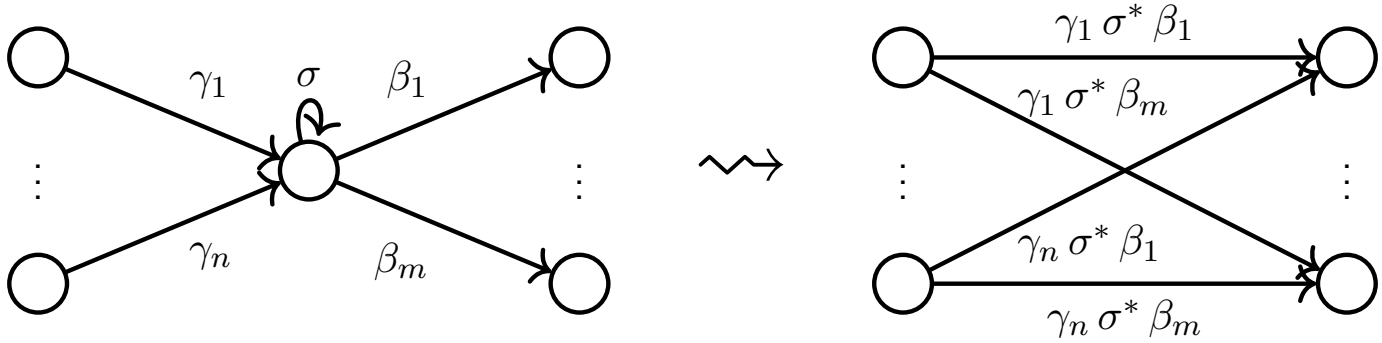


Schritt 2: Wiederhole so lang wie möglich:

- Solange es Übergänge (q, γ_1, p) and (q, γ_2, p) gibt, ersetze sie durch einen einzigen Übergang $(q, \gamma_1 \mid \gamma_2, q')$.

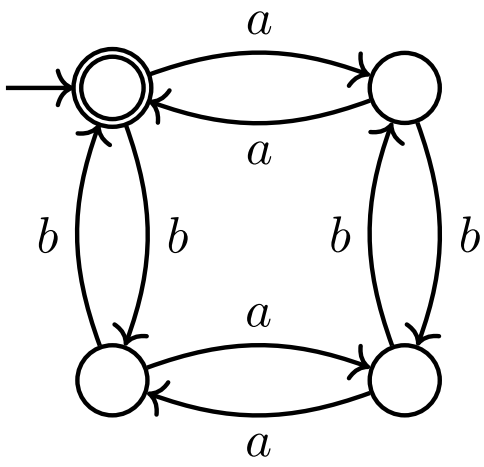


- Wähle Zustand q der weder Anfang- noch Endzustand ist.
- Hat q eine Schleife (q, σ, q), dann ersetze jedes Paar (p, γ, q), (q, β, p') von Übergängen mit $p \neq q \neq p'$ (aber möglicherweise $p = p'$) durch $(p, \gamma\sigma^*\beta, p')$, sonst durch $(p, \gamma\beta, p')$.
- Entferne den Zustand q zusammen mit all seinen eingehenden und ausgehenden Übergängen.



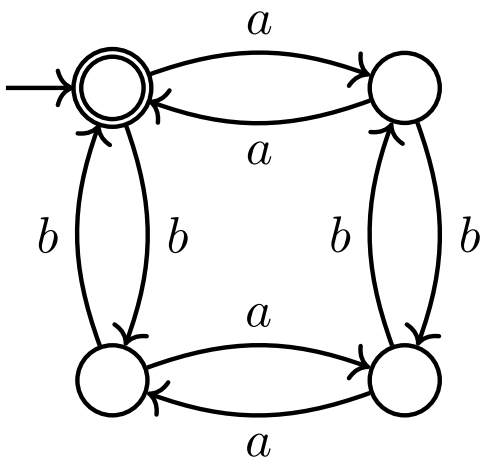
Beispiel 3.26

(a)

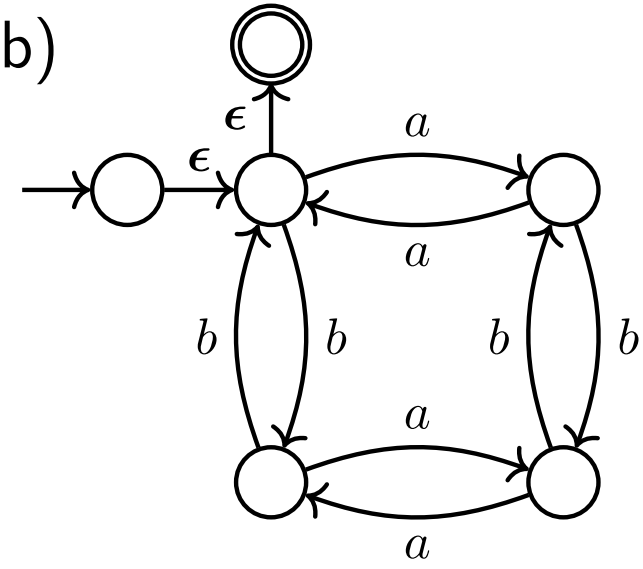


Beispiel 3.26

(a)

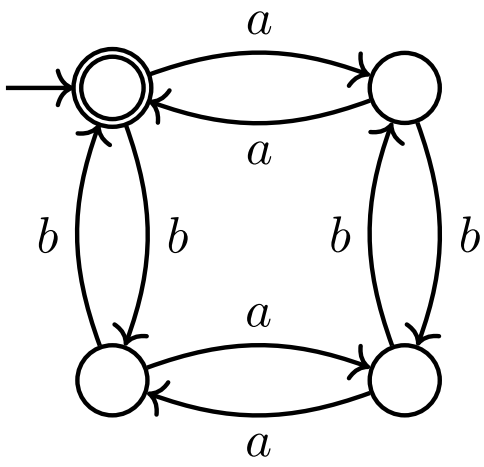


(b)

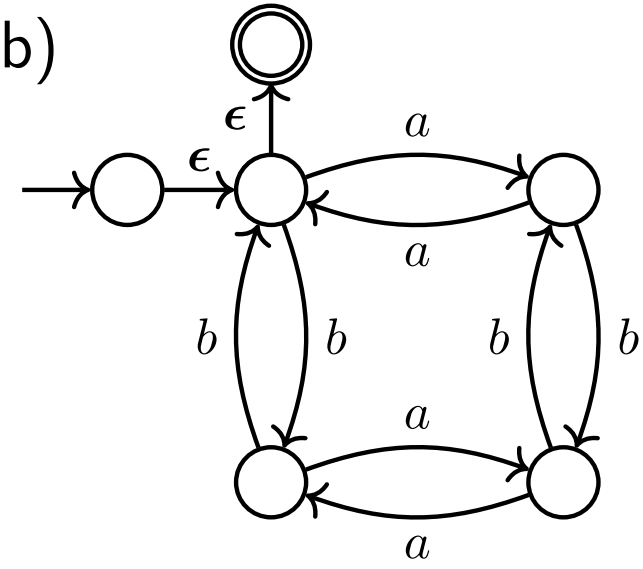


Beispiel 3.26

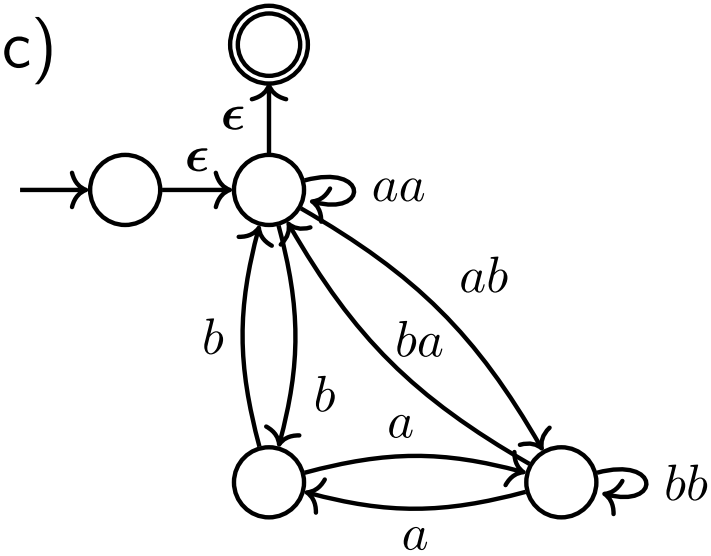
(a)



(b)

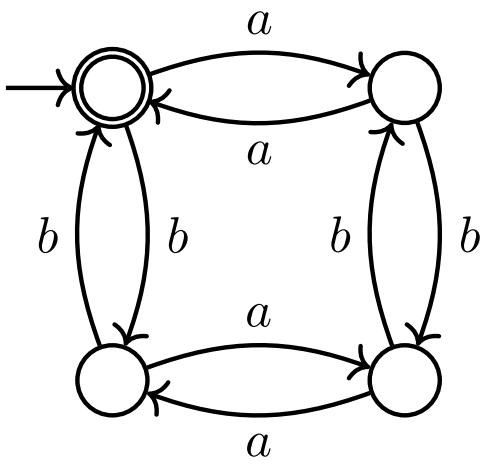


(c)

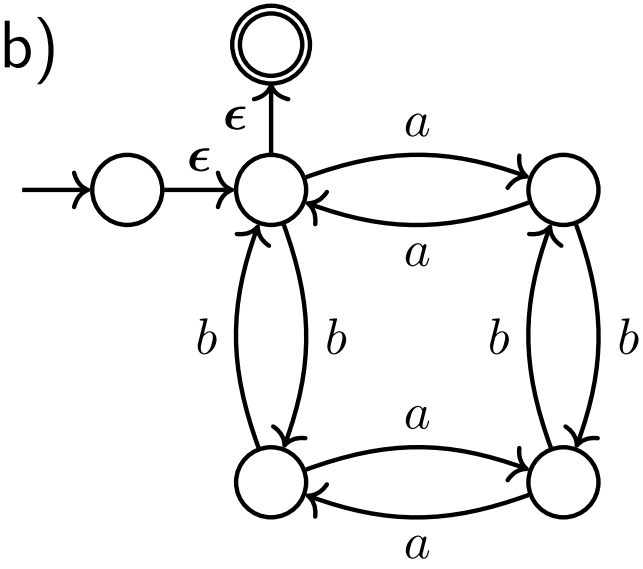


Beispiel 3.26

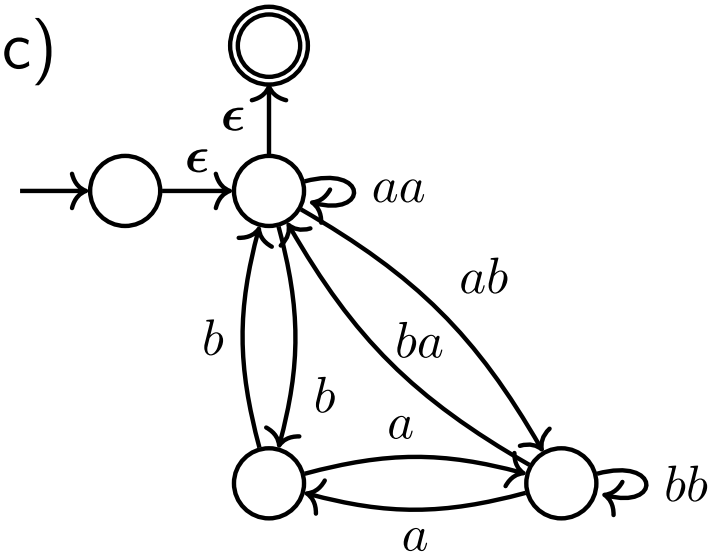
(a)



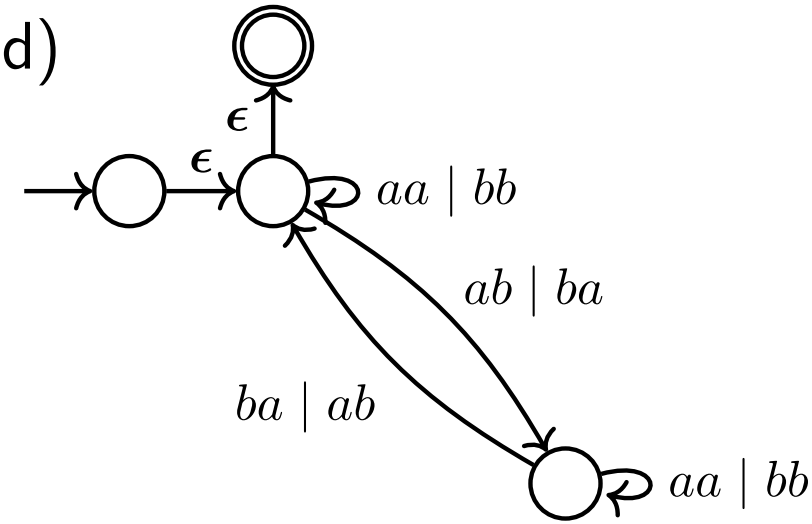
(b)



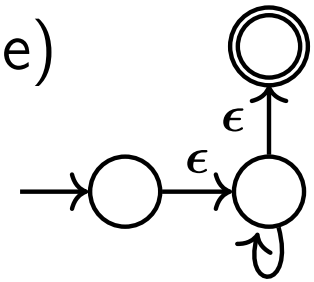
(c)



(d)

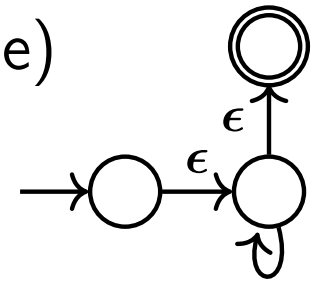


(e)



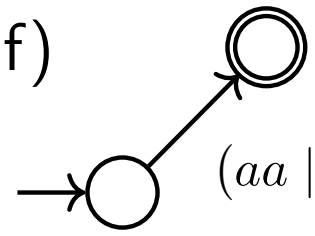
$$aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ba \mid ab)$$

(e)



$$aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ba \mid ab)$$

(f)



$$(aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ba \mid ab))^*$$

3.5 Von NFA zu RE als Lösung eines Gleichungssystems

Gleichungen, deren Variablen für unbekannte reguläre Ausdrücke und das Gleichheitszeichen für Sprachäquivalenz stehen.

Beispiel 3.27

$X \equiv aX \mid b$ Lösungen: alle RE γ mit $\gamma \equiv a^*b$.

$X \equiv X \mid aX$ Lösungen: ?

$X \equiv aX$ Lösungen: ?

Satz 3.28 (Ardens Lemma)

Sind α, β und X reguläre Ausdrücke mit $\epsilon \notin L(\alpha)$, so gilt

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

Beweis von Ardens Lemma

Wir nehmen an $X \equiv \alpha X \mid \beta$.

$$X \quad \equiv \quad \alpha(\alpha X \mid \beta) \mid \beta \equiv \alpha^2 X \mid \alpha\beta \mid \beta$$

Beweis von Ardens Lemma

Wir nehmen an $X \equiv \alpha X \mid \beta$.

$$\begin{aligned} X &\equiv \alpha(\alpha X \mid \beta) \mid \beta \equiv \alpha^2 X \mid \alpha\beta \mid \beta \\ &\equiv \alpha^2(\alpha X \mid \beta) \mid \alpha\beta \mid \beta \equiv \alpha^3 X \mid \alpha^2\beta \mid \alpha\beta \mid \beta \equiv \ldots \end{aligned}$$

Beweis von Ardens Lemma

Wir nehmen an $X \equiv \alpha X \mid \beta$.

$$\begin{aligned} X &\equiv \alpha(\alpha X \mid \beta) \mid \beta \equiv \alpha^2 X \mid \alpha\beta \mid \beta \\ &\equiv \alpha^2(\alpha X \mid \beta) \mid \alpha\beta \mid \beta \equiv \alpha^3 X \mid \alpha^2\beta \mid \alpha\beta \mid \beta \equiv \ldots \end{aligned}$$

Wir zeigen mit Induktion für alle $n \in \mathbb{N}$:

$$X \equiv \alpha^{n+1} X \mid (\alpha^n \mid \alpha^{n-1} \mid \cdots \mid \alpha \mid \boldsymbol{\epsilon})\beta \qquad (1)$$

Beweis von Ardens Lemma

Wir nehmen an $X \equiv \alpha X \mid \beta$.

$$\begin{aligned} X &\equiv \alpha(\alpha X \mid \beta) \mid \beta \equiv \alpha^2 X \mid \alpha\beta \mid \beta \\ &\equiv \alpha^2(\alpha X \mid \beta) \mid \alpha\beta \mid \beta \equiv \alpha^3 X \mid \alpha^2\beta \mid \alpha\beta \mid \beta \equiv \ldots \end{aligned}$$

Wir zeigen mit Induktion für alle $n \in \mathbb{N}$:

$$X \equiv \alpha^{n+1} X \mid (\alpha^n \mid \alpha^{n-1} \mid \cdots \mid \alpha \mid \boldsymbol{\epsilon})\beta \qquad (1)$$

$n = 0$: Behauptung wird zu $X \equiv \alpha X \mid \beta$, die Annahme.

Beweis von Ardens Lemma

Wir nehmen an $X \equiv \alpha X \mid \beta$.

$$\begin{aligned} X &\equiv \alpha(\alpha X \mid \beta) \mid \beta \equiv \alpha^2 X \mid \alpha\beta \mid \beta \\ &\equiv \alpha^2(\alpha X \mid \beta) \mid \alpha\beta \mid \beta \equiv \alpha^3 X \mid \alpha^2\beta \mid \alpha\beta \mid \beta \equiv \ldots \end{aligned}$$

Wir zeigen mit Induktion für alle $n \in \mathbb{N}$:

$$X \equiv \alpha^{n+1} X \mid (\alpha^n \mid \alpha^{n-1} \mid \cdots \mid \alpha \mid \boldsymbol{\epsilon})\beta \qquad (1)$$

$n = 0$: Behauptung wird zu $X \equiv \alpha X \mid \beta$, die Annahme.

$n \to n+1$:
$$\begin{aligned} X &\equiv \alpha^{n+1} X \mid (\alpha^n \mid \cdots \mid \alpha \mid \boldsymbol{\epsilon})\beta \\ &\equiv \alpha^{n+1}(\alpha X \mid \beta) \mid (\alpha^n \mid \cdots \mid \alpha \mid \boldsymbol{\epsilon})\beta \\ &\equiv \alpha^{n+2} X \mid \alpha^{n+1}\beta \mid (\alpha^n \mid \cdots \mid \alpha \mid \boldsymbol{\epsilon})\beta \\ &\equiv \alpha^{n+2} X \mid (\alpha^{n+1} \mid \alpha^n \mid \cdots \mid \alpha \mid \boldsymbol{\epsilon})\beta \end{aligned}$$

Wir zeigen nun $X \equiv \alpha^*\beta$, d.h. $L(X) = L(\alpha^*\beta)$.

Wir zeigen nun $X \equiv \alpha^*\beta$, d.h. $L(X) = L(\alpha^*\beta)$.

$L(\alpha^*\beta) \subseteq L(X)$:

$$
\begin{aligned}
& w \in L(\alpha^*\beta) = \bigcup_{n \geq 0} L(\alpha^n\beta) \\
\Longrightarrow\ & \exists n.\ w \in L(\alpha^n\beta) \\
\Longrightarrow\ & w \in L(X) \qquad \text{(wegen (1))}
\end{aligned}
$$

Wir zeigen nun $X \equiv \alpha^*\beta$, d.h. $L(X) = L(\alpha^*\beta)$.

$L(\alpha^*\beta) \subseteq L(X)$:

$$
\begin{aligned}
& w \in L(\alpha^*\beta) = \bigcup_{n \geq 0} L(\alpha^n\beta) \\
\Longrightarrow\ & \exists n.\ w \in L(\alpha^n\beta) \\
\Longrightarrow\ & w \in L(X) \qquad \text{(wegen (1))}
\end{aligned}
$$

Wir zeigen nun $X \equiv \alpha^*\beta$, d.h. $L(X) = L(\alpha^*\beta)$.

$L(\alpha^*\beta) \subseteq L(X)$:

$$
\begin{aligned}
& w \in L(\alpha^*\beta) = \bigcup_{n \geq 0} L(\alpha^n\beta) \\
\Longrightarrow\ & \exists n.\ w \in L(\alpha^n\beta) \\
\Longrightarrow\ & w \in L(X) && \text{(wegen (1))}
\end{aligned}
$$

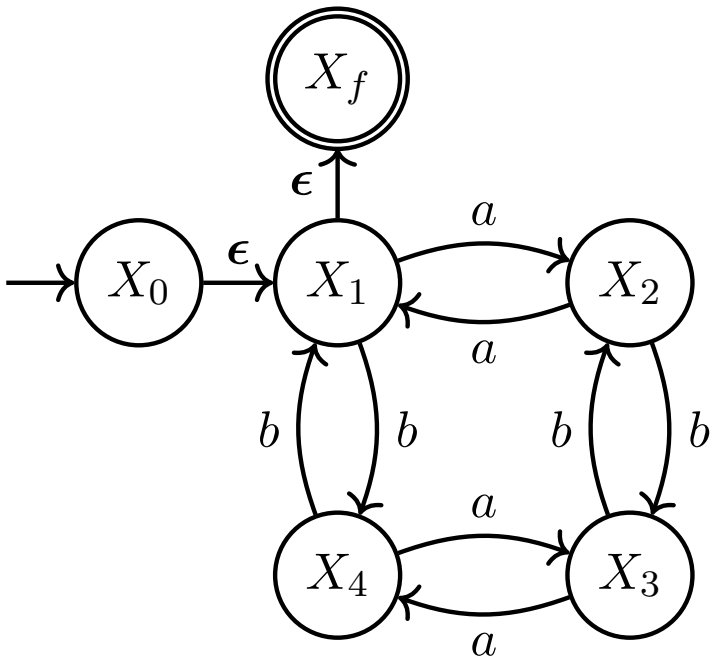
$L(X) \subseteq L(\alpha^*\beta)$: Sei $w \in L(X)$ und $n := |w|$.

$$
\begin{aligned}
& \epsilon \notin L(\alpha) \\
\Longrightarrow\ & \forall u \in L(\alpha^{n+1}).\ |u| \geq n+1 \\
\Longrightarrow\ & w \notin L(\alpha^{n+1}X) \\
\Longrightarrow\ & w \in L((\alpha^n \mid \cdots \mid \alpha \mid \boldsymbol{\epsilon})\beta) && \text{(wegen (1))} \\
\Longrightarrow\ & w \in L(\alpha^*\beta)
\end{aligned}
$$

□

Von NFA nach RE.

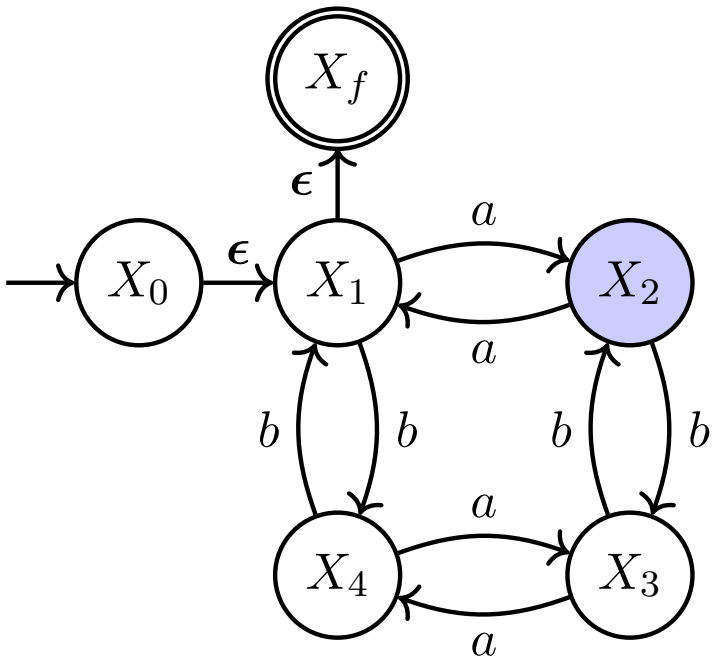
Erster Schritt: NFA als Gleichungssytem darstellen:



$$X_0 \equiv X_1$$
$$X_1 \equiv aX_2 \mid bX_4 \mid X_f$$
$$X_2 \equiv aX_1 \mid bX_3$$
$$X_3 \equiv aX_4 \mid bX_2$$
$$X_4 \equiv aX_3 \mid bX_1$$
$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

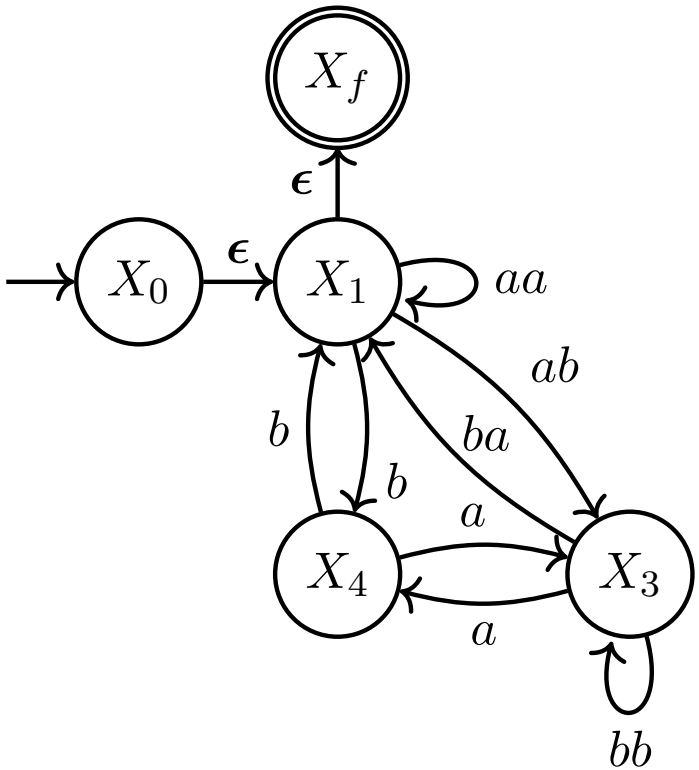
1. Zustand X_2 eliminieren $\rightsquigarrow$
   Variable X_2 eliminieren durch Einsetzen



$$X_0 \equiv X_1$$
$$X_1 \equiv aX_2 \mid bX_4 \mid X_f$$
$$X_2 \equiv aX_1 \mid bX_3$$
$$X_3 \equiv aX_4 \mid bX_2$$
$$X_4 \equiv aX_3 \mid bX_1$$
$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

1. Zustand X_2 eliminieren $\rightsquigarrow$
   Variable X_2 eliminieren durch Einsetzen

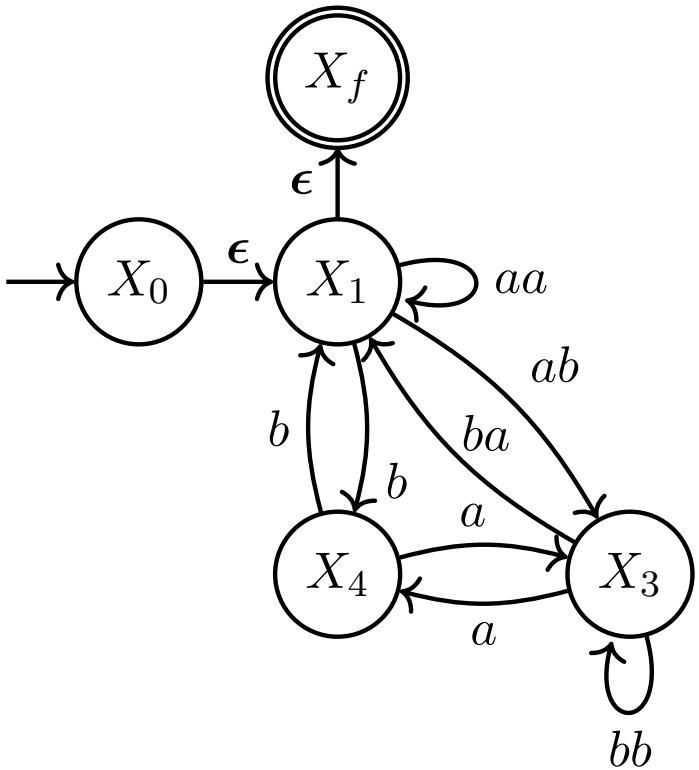


$$X_0 \equiv X_1$$
$$X_1 \equiv a(aX_1 \mid bX_3) \mid bX_4 \mid X_f$$

$$X_3 \equiv aX_4 \mid b(aX_1 \mid bX_3)$$
$$X_4 \equiv aX_3 \mid bX_1$$
$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

1. Zustand X_2 eliminieren $\rightsquigarrow$
   Variable X_2 eliminieren durch Einsetzen

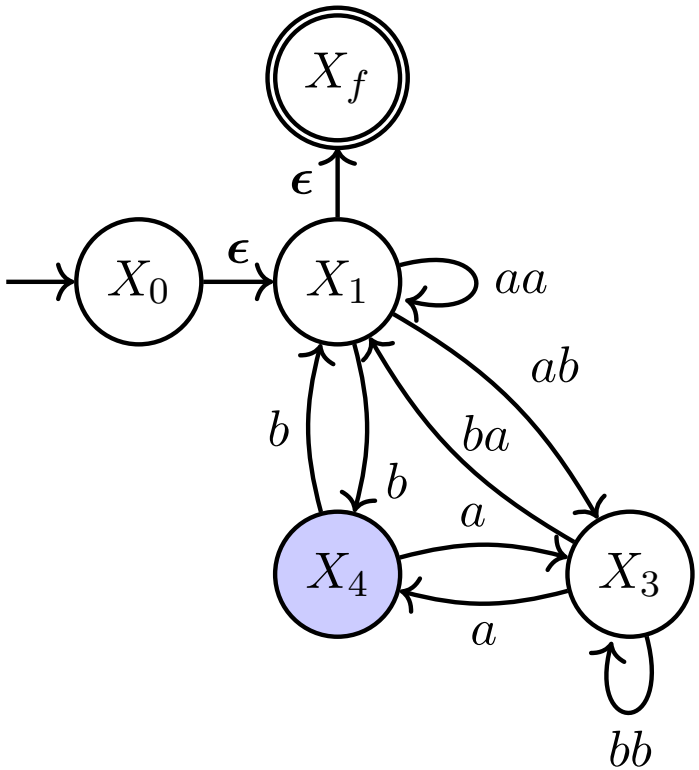


$$X_0 \equiv X_1$$
$$X_1 \equiv aaX_1 \mid abX_3 \mid bX_4 \mid X_f$$

$$X_3 \equiv baX_1 \mid aX_4 \mid bbX_3$$
$$X_4 \equiv aX_3 \mid bX_1$$
$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

2. Zustand X_4 eliminieren $\rightsquigarrow$
   Variable X_4 eliminieren durch Einsetzen

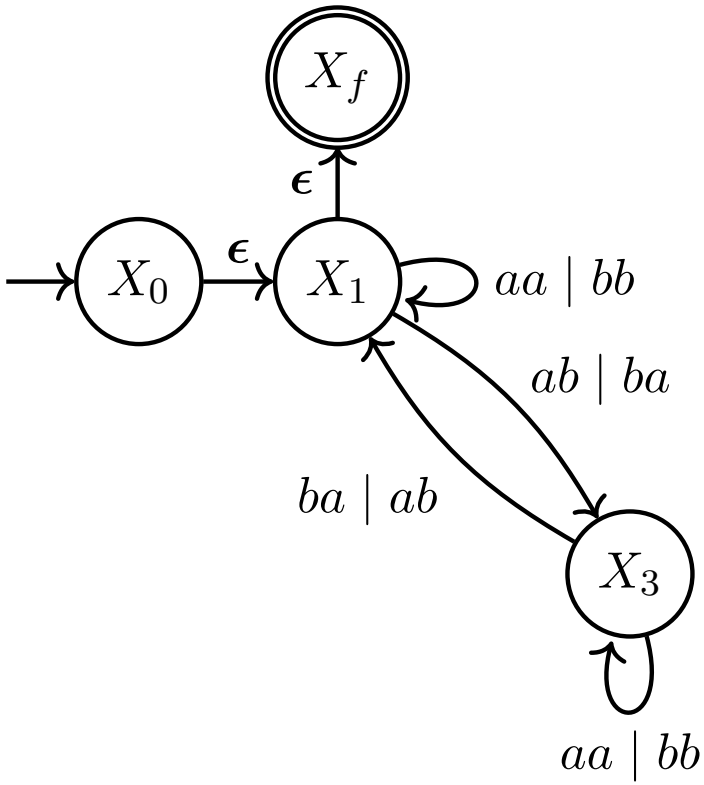


$$X_0 \equiv X_1$$
$$X_1 \equiv aaX_1 \mid abX_3 \mid bX_4 \mid X_f$$

$$X_3 \equiv baX_1 \mid aX_4 \mid bbX_3$$
$$X_4 \equiv aX_3 \mid bX_1$$
$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

2. Zustand X_4 eliminieren $\rightsquigarrow$
   Variable X_4 eliminieren durch Einsetzen

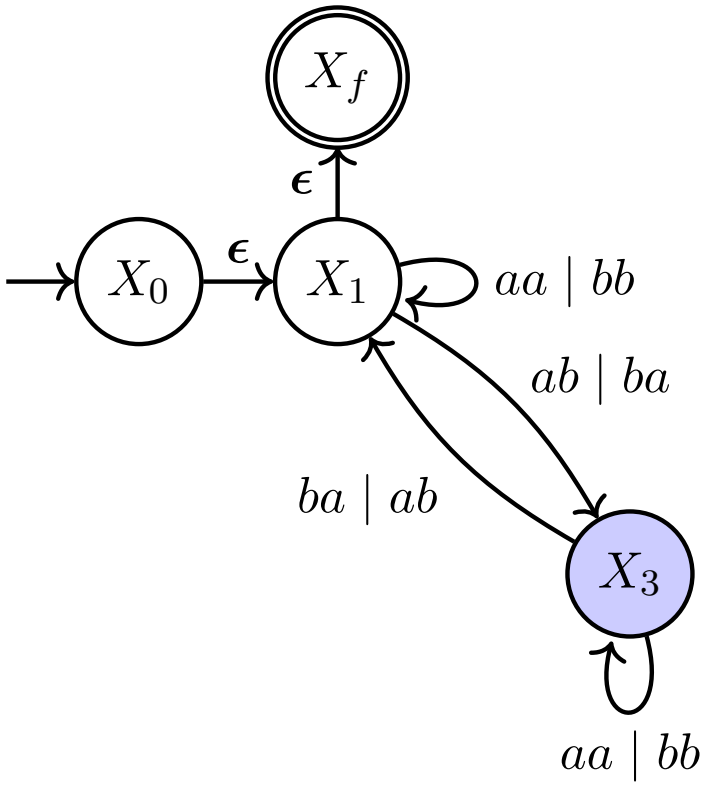


$$X_0 \equiv X_1$$
$$X_1 \equiv (aa \mid bb)X_1 \mid (ab \mid ba)X_3 \mid X_f$$
$$X_3 \equiv (ba \mid ab)X_1 \mid (aa \mid bb)X_3$$

$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

3. Zustand X_3 eliminieren $\rightsquigarrow$
   Variable X_3 eliminieren mit Hilfe von Ardens Lemma

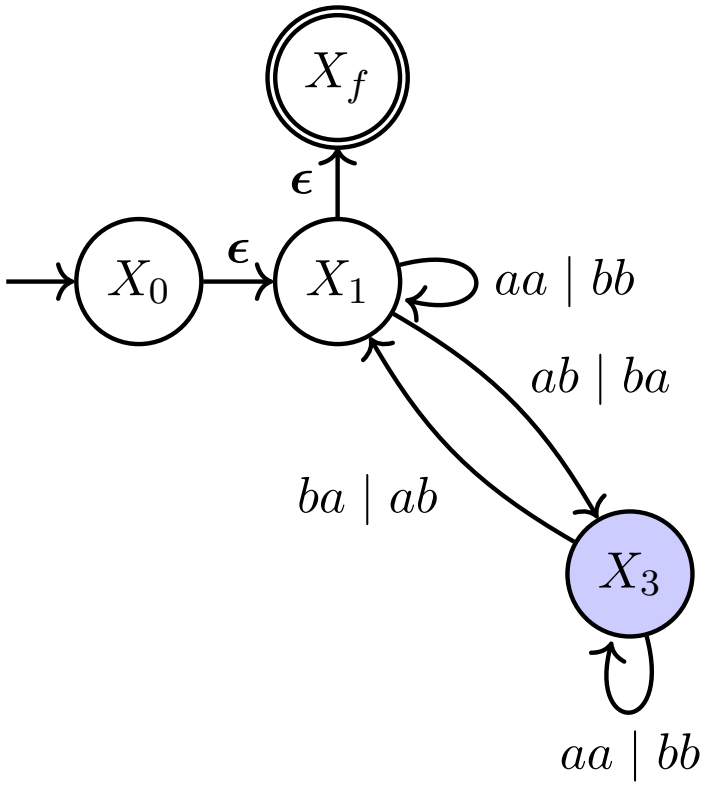


$$X_0 \equiv X_1$$
$$X_1 \equiv (aa \mid bb)X_1 \mid (ab \mid ba)X_3 \mid X_f$$
$$X_3 \equiv (ba \mid ab)X_1 \mid (aa \mid bb)X_3$$

$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

3. Zustand X_3 eliminieren $\rightsquigarrow$
   Variable X_3 eliminieren mit Hilfe von Ardens Lemma



$$X_0 \equiv X_1$$

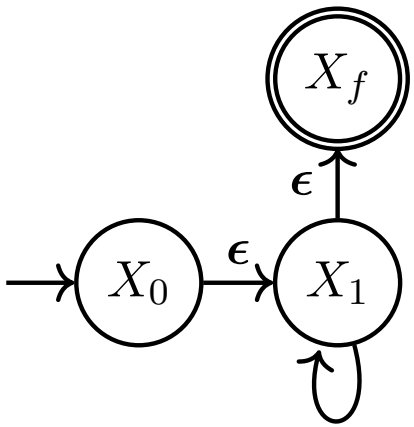
$$X_1 \equiv (aa \mid bb)X_1 \mid (ab \mid ba)X_3 \mid X_f$$

$$X_3 \equiv (aa \mid bb)^*(ba \mid ab)X_1$$

$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

3. Zustand X_3 eliminieren $\rightsquigarrow$
   Variable X_3 eliminieren mit Hilfe von Ardens Lemma

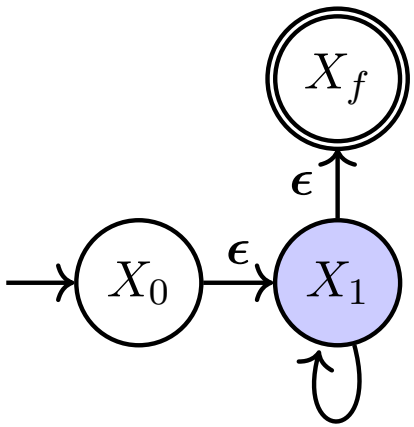


$aa \mid bb \mid$
$(ab \mid ba)(aa \mid bb)^*(ba \mid ab)$

$$X_0 \equiv X_1$$
$$X_1 \equiv \big(aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ba \mid ab)\big)X_1 \mid X_f$$
$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

4. Zustand X_1 eliminieren $\rightsquigarrow$
   Variable X_1 eliminieren mit Hilfe von Ardens Lemma

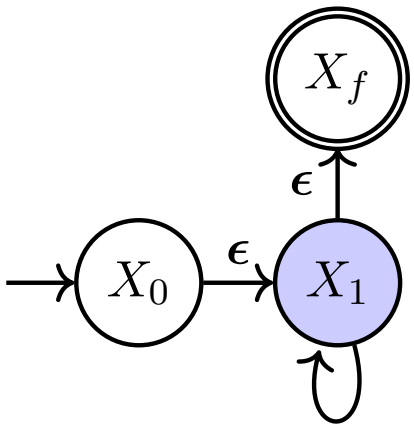


$aa \mid bb \mid$
$(ab \mid ba)(aa \mid bb)^*(ba \mid ab)$

$$X_0 \equiv X_1$$
$$X_1 \equiv (aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ba \mid ab))X_1 \mid X_f$$
$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

4. Zustand X_1 eliminieren $\rightsquigarrow$
   Variable X_1 eliminieren mit Hilfe von Ardens Lemma



$$aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ba \mid ab)$$

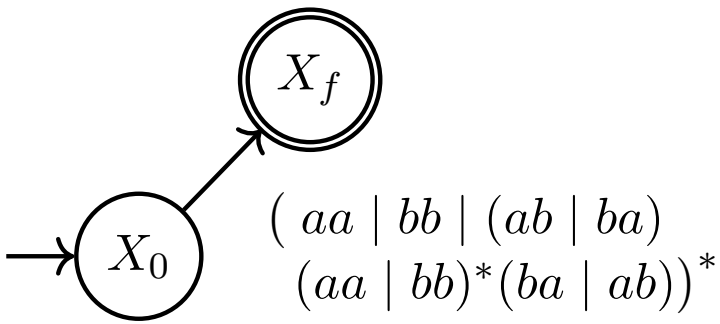
$$X_0 \equiv X_1$$

$$X_1 \equiv (aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ba \mid ab))^* X_f$$

$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

4. Zustand X_1 eliminieren $\rightsquigarrow$
   Variable X_1 eliminieren mit Hilfe von Ardens Lemma

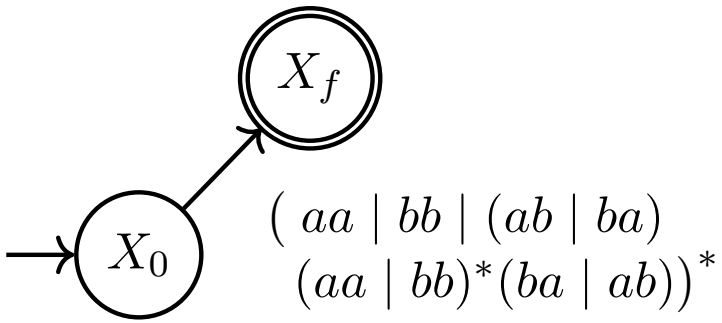


$$X_0 \equiv\ (aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ba \mid ab))^* X_f$$

$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

5. Variable X_f eliminieren durch Einsetzen

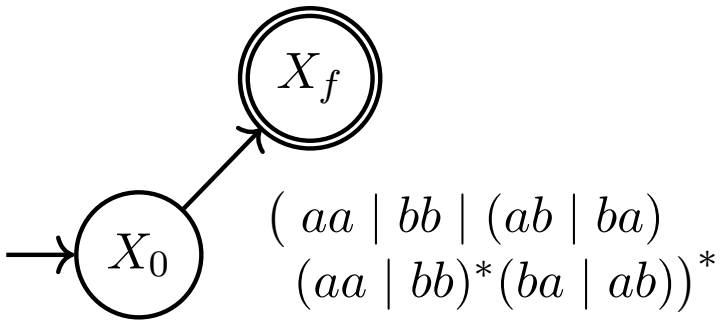


$$X_0 \equiv \ (aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ba \mid ab))^* X_f$$

$$X_f \equiv \epsilon$$

Zweiter Schritt: Gleichungssystem lösen:

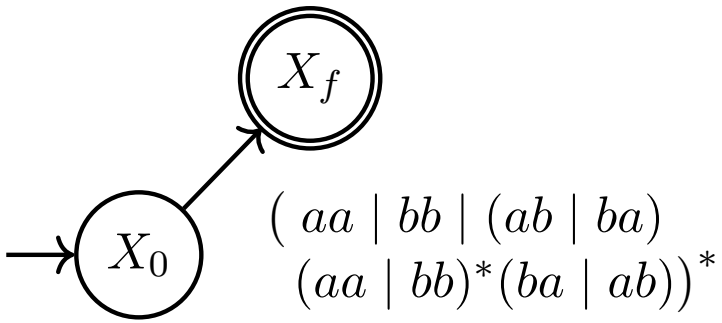
5. Variable X_f eliminieren durch Einsetzen



$$X_0 \equiv\ (aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ba \mid ab))^* \boldsymbol{\epsilon}$$

Zweiter Schritt: Gleichungssystem lösen:

5. Variable X_f eliminieren durch Einsetzen



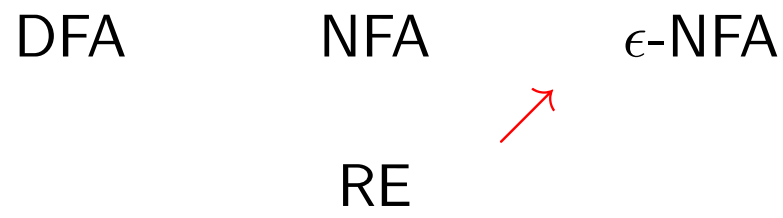
$$X_0 \equiv\ (aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ba \mid ab))^*$$

Unsere Konversionen auf einen Blick:

DFA NFA ϵ-NFA

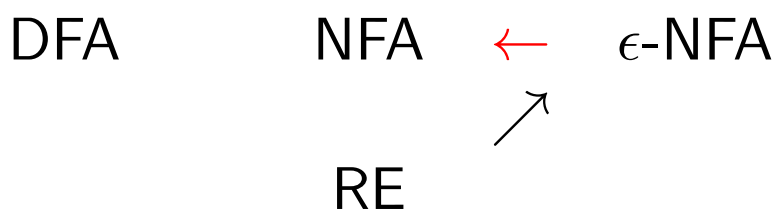
RE

Unsere Konversionen auf einen Blick:



RE $\to$ ϵ-NFA: RE der Länge $n \rightsquigarrow O(n)$ Zustände

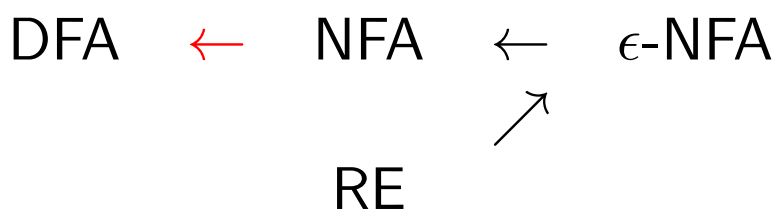
Unsere Konversionen auf einen Blick:



RE $\to$ ϵ-NFA: RE der Länge $n \rightsquigarrow O(n)$ Zustände

ϵ-NFA $\to$ NFA: $Q \rightsquigarrow Q$

Unsere Konversionen auf einen Blick:

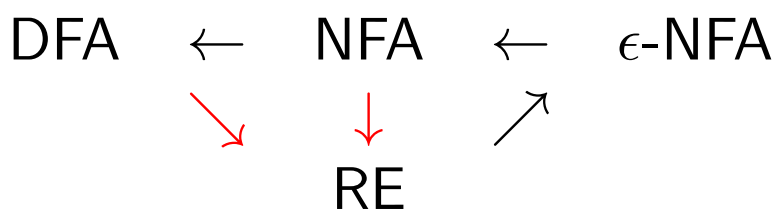


RE $\to$ ϵ-NFA: RE der Länge $n \rightsquigarrow O(n)$ Zustände

ϵ-NFA $\to$ NFA: $Q \rightsquigarrow Q$

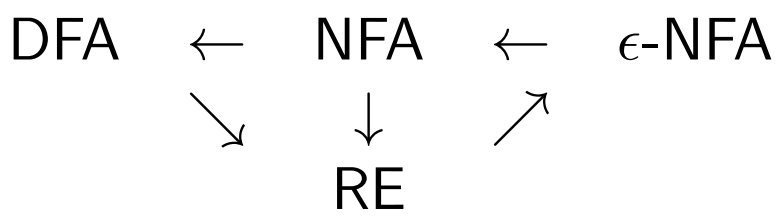
NFA $\to$ DFA: n Zustände $\rightsquigarrow O(2^n)$ Zustände

Unsere Konversionen auf einen Blick:



RE $\to$ ϵ-NFA:	RE der Länge $n \rightsquigarrow O(n)$ Zustände
ϵ-NFA $\to$ NFA:	$Q \rightsquigarrow Q$
NFA $\to$ DFA:	n Zustände $\rightsquigarrow O(2^n)$ Zustände
NFA $\to$ RE:	n Zustände $\rightsquigarrow$ RE der Länge $O(3^n)$

Unsere Konversionen auf einen Blick:



RE $\to$ ϵ-NFA:	RE der Länge $n \rightsquigarrow O(n)$ Zustände
ϵ-NFA $\to$ NFA:	$Q \rightsquigarrow Q$
NFA $\to$ DFA:	n Zustände $\rightsquigarrow O(2^n)$ Zustände
NFA $\to$ RE:	n Zustände $\rightsquigarrow$ RE der Länge $O(3^n)$

Beweis NFA$\to$RE: $m_k :=$ maximale Länge der RE nach Entfernung von k Zuständen

- $m_0 = 1$
- $m_{k+1} \leq 3m_k + 1$
- Länge des Gesamtausdrucks: $O(3^n)$

3.6 Rechnen mit regulären Ausdrücken

Definition 3.29
Zwei reguläre Ausdrücke sind äquivalent gdw sie die gleiche Sprache darstellen:

$$\alpha \equiv \beta \;\; :\Leftrightarrow \;\; L(\alpha) = L(\beta)$$

3.6 Rechnen mit regulären Ausdrücken

Definition 3.29
Zwei reguläre Ausdrücke sind äquivalent gdw sie die gleiche Sprache darstellen:

$$\alpha \equiv \beta \;\; :\Leftrightarrow \;\; L(\alpha) = L(\beta)$$

Beispiel zum Unterschied von $=$ (syntaktische Identität) und $\equiv$ (Bedeutungsäquivalenz):
$(\alpha \mid \beta) \equiv (\beta \mid \alpha)$ aber $(\alpha \mid \beta) \neq (\beta \mid \alpha)$.

Null und Eins:

Lemma 3.30

- $\emptyset \mid \alpha \equiv \alpha \mid \emptyset \equiv \alpha$
- $\emptyset\alpha \equiv \alpha\emptyset \equiv \emptyset$
- $\epsilon\alpha \equiv \alpha\epsilon \equiv \alpha$
- $\emptyset^* \equiv \epsilon$
- $\epsilon^* \equiv \epsilon$

Lemma 3.31

Assoziativität:

- $(\alpha \mid \beta) \mid \gamma \equiv \alpha \mid (\beta \mid \gamma)$
- $(\alpha\beta)\gamma \equiv \alpha(\beta\gamma)$

Kommutativität:

- $\alpha \mid \beta \equiv \beta \mid \alpha$

Distributivität:

- $\alpha(\beta \mid \gamma) \equiv \alpha\beta \mid \alpha\gamma$
- $(\alpha \mid \beta)\gamma \equiv \alpha\gamma \mid \beta\gamma$

Idempotenz:

- $\alpha \mid \alpha \equiv \alpha$

Stern:

Lemma 3.32

- $\epsilon \mid \alpha\alpha^* \equiv \alpha^*$
- $\alpha^*\alpha \equiv \alpha\alpha^*$
- $(\alpha^*)^* \equiv \alpha^*$

Stern:

Lemma 3.32

- $\epsilon \mid \alpha\alpha^* \equiv \alpha^*$
- $\alpha^*\alpha \equiv \alpha\alpha^*$
- $(\alpha^*)^* \equiv \alpha^*$

Beispiel 3.33

Herleitung einer Äquivalenz aus obigen Lemmas:
$\epsilon \mid \alpha^*$

Stern:

Lemma 3.32

- $\boldsymbol{\epsilon} \mid \alpha\alpha^* \equiv \alpha^*$
- $\alpha^*\alpha \equiv \alpha\alpha^*$
- $(\alpha^*)^* \equiv \alpha^*$

Beispiel 3.33

Herleitung einer Äquivalenz aus obigen Lemmas:

$\boldsymbol{\epsilon} \mid \alpha^*$

$\equiv \boldsymbol{\epsilon} \mid (\boldsymbol{\epsilon} \mid \alpha\alpha^*)$ Stern Lemma

Stern:

Lemma 3.32

- $\boldsymbol{\epsilon} \mid \alpha\alpha^* \equiv \alpha^*$
- $\alpha^*\alpha \equiv \alpha\alpha^*$
- $(\alpha^*)^* \equiv \alpha^*$

Beispiel 3.33

Herleitung einer Äquivalenz aus obigen Lemmas:

$\boldsymbol{\epsilon} \mid \alpha^*$
$\equiv \boldsymbol{\epsilon} \mid (\boldsymbol{\epsilon} \mid \alpha\alpha^*)$ Stern Lemma
$\equiv (\boldsymbol{\epsilon} \mid \boldsymbol{\epsilon}) \mid \alpha\alpha^*$ Assoziativität

Stern:

Lemma 3.32

- $\boldsymbol{\epsilon} \mid \alpha\alpha^* \equiv \alpha^*$
- $\alpha^*\alpha \equiv \alpha\alpha^*$
- $(\alpha^*)^* \equiv \alpha^*$

Beispiel 3.33

Herleitung einer Äquivalenz aus obigen Lemmas:

$$
\begin{aligned}
& \boldsymbol{\epsilon} \mid \alpha^* \\
\equiv\ & \boldsymbol{\epsilon} \mid (\boldsymbol{\epsilon} \mid \alpha\alpha^*) && \text{Stern Lemma} \\
\equiv\ & (\boldsymbol{\epsilon} \mid \boldsymbol{\epsilon}) \mid \alpha\alpha^* && \text{Assoziativität} \\
\equiv\ & \boldsymbol{\epsilon} \mid \alpha\alpha^* && \text{Idempotenz}
\end{aligned}
$$

Stern:

Lemma 3.32

- $\boldsymbol{\epsilon} \mid \alpha\alpha^* \equiv \alpha^*$
- $\alpha^*\alpha \equiv \alpha\alpha^*$
- $(\alpha^*)^* \equiv \alpha^*$

Beispiel 3.33

Herleitung einer Äquivalenz aus obigen Lemmas:

$$
\begin{aligned}
& \boldsymbol{\epsilon} \mid \alpha^* \\
\equiv\ & \boldsymbol{\epsilon} \mid (\boldsymbol{\epsilon} \mid \alpha\alpha^*) && \text{Stern Lemma} \\
\equiv\ & (\boldsymbol{\epsilon} \mid \boldsymbol{\epsilon}) \mid \alpha\alpha^* && \text{Assoziativität} \\
\equiv\ & \boldsymbol{\epsilon} \mid \alpha\alpha^* && \text{Idempotenz} \\
\equiv\ & \alpha^* && \text{Stern Lemma}
\end{aligned}
$$

Lässt sich jede gültige Äquivalenz $\alpha \equiv \beta$ aus den obigen Lemmas für $\equiv$ herleiten?

Lässt sich jede gültige Äquivalenz $\alpha \equiv \beta$ aus den obigen Lemmas für $\equiv$ herleiten?

Satz 3.34 (Redko 1964)

Es gibt keine endliche Menge von gültigen Äquivalenzen aus denen sich alle gültigen Äquivalenzen herleiten lassen.

Lässt sich jede gültige Äquivalenz $\alpha \equiv \beta$ aus den obigen Lemmas für $\equiv$ herleiten?

Satz 3.34 (Redko 1964)

Es gibt keine endliche Menge von gültigen Äquivalenzen aus denen sich alle gültigen Äquivalenzen herleiten lassen.

Wenn man mehr als nur Äquivalenzen zulässt:

Arto Salomaa.
Two Complete Axiom Systems for the Algebra of Regular Events. Journal of the ACM, 1966.

3.7 Abschlusseigenschaften regulärer Sprachen

Satz 3.35

Seien $R, R_1, R_2 \subseteq \Sigma^*$ reguläre Sprachen. Dann sind auch

$$R_1 R_2,\ R_1 \cup R_2,\ R^*,\ \overline{R}\ (:= \Sigma^* \setminus R),\ R_1 \cap R_2,\ R_1 \setminus R_2$$

reguläre Sprachen.

3.7 Abschlusseigenschaften regulärer Sprachen

Satz 3.35

Seien $R, R_1, R_2 \subseteq \Sigma^*$ reguläre Sprachen. Dann sind auch

$$R_1 R_2,\ R_1 \cup R_2,\ R^*,\ \overline{R}\ (:= \Sigma^* \setminus R),\ R_1 \cap R_2,\ R_1 \setminus R_2$$

reguläre Sprachen.

Beweis:

$R_1 R_2$, $R_1 \cup R_2$, und R^*: klar.

$\overline{R}$ Sei $R = L(A)$ für einen DFA $A = (Q, \Sigma, \delta, q_0, F)$.
Betrachte $A' = (Q, \Sigma, \delta, q_0, Q \setminus F)$.
Dann ist $L(A') = \overline{L(A)} = \overline{R}$

$R_1 \cap R_2 = \overline{\overline{R_1} \cup \overline{R_2}}$ (De Morgan)

$R_1 \setminus R_2 = R_1 \cap \overline{R_2}$

□

Bemerkung

Komplementierung ($\overline{R}$) durch Vertauschen von Endzuständen und Nicht-Endzuständen funktioniert nur bei DFAs, nicht bei NFAs!

Bemerkung
Komplementierung ($\overline{R}$) durch Vertauschen von Endzuständen und Nicht-Endzuständen funktioniert nur bei DFAs, nicht bei NFAs!

Bei NFAs:

Komplementierung erzwingt Determinierung.

Komplementierung ist (potenziell) teuer.

Die Produkt-Konstruktion: Durchschnitt direkt auf DFAs, ohne Umweg über de Morgan.

Die Produkt-Konstruktion: Durchschnitt direkt auf DFAs, ohne Umweg über de Morgan.

Beide DFAs laufen synchron parallel, Wort wird akzeptiert wenn *beide* akzeptieren.

Parallelismus = Kreuzprodukt der Zustandsräume

Die Produkt-Konstruktion: Durchschnitt direkt auf DFAs, ohne Umweg über de Morgan.

Beide DFAs laufen synchron parallel, Wort wird akzeptiert wenn *beide* akzeptieren.

Parallelismus = Kreuzprodukt der Zustandsräume

Satz 3.36
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.*

Die Produkt-Konstruktion: Durchschnitt direkt auf DFAs, ohne Umweg über de Morgan.

Beide DFAs laufen synchron parallel, Wort wird akzeptiert wenn *beide* akzeptieren.

Parallelismus = Kreuzprodukt der Zustandsräume

Satz 3.36
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.*

Erinnerung: $|Q_1 \times Q_2| = |Q_1||Q_2|$.

Beweis:

Durch Induktion über w läßt sich zeigen:

$$\hat{\delta}((q_1, q_2), w) = (\hat{\delta}_1(q_1, w), \hat{\delta}_2(q_2, w)).$$

Beweis:

Durch Induktion über w läßt sich zeigen:

$$\hat{\delta}((q_1, q_2), w) = (\hat{\delta}_1(q_1, w), \hat{\delta}_2(q_2, w)).$$

Damit gilt:

$$\begin{aligned}
& w \in L(M) \\
\Leftrightarrow \quad & (\hat{\delta}((s_1, s_2), w) \in F_1 \times F_2 \\
\Leftrightarrow \quad & (\hat{\delta}_1(s_1, w), \hat{\delta}_2(s_2, w)) \in F_1 \times F_2 \\
\Leftrightarrow \quad & \hat{\delta}_1(s_1, w) \in F_1 \wedge \hat{\delta}_2(s_2, w) \in F_2 \\
\Leftrightarrow \quad & w \in L(M_1) \wedge w \in L(M_2) \\
\Leftrightarrow \quad & w \in L(M_1) \cap L(M_2)
\end{aligned}$$

Beweis:

Durch Induktion über w läßt sich zeigen:

$$\hat{\delta}((q_1, q_2), w) = (\hat{\delta}_1(q_1, w), \hat{\delta}_2(q_2, w)).$$

Damit gilt:

$$\begin{aligned}
& w \in L(M) \\
\Leftrightarrow \quad & (\hat{\delta}((s_1, s_2), w) \in F_1 \times F_2 \\
\Leftrightarrow \quad & (\hat{\delta}_1(s_1, w), \hat{\delta}_2(s_2, w)) \in F_1 \times F_2 \\
\Leftrightarrow \quad & \hat{\delta}_1(s_1, w) \in F_1 \wedge \hat{\delta}_2(s_2, w) \in F_2 \\
\Leftrightarrow \quad & w \in L(M_1) \wedge w \in L(M_2) \\
\Leftrightarrow \quad & w \in L(M_1) \cap L(M_2)
\end{aligned}$$

□

Funktioniert Durchschnitt durch Produkt auch für NFAs?

3.8 Pumping Lemma

Oder: *Wie zeigt man, dass eine Sprache nicht regulär ist?*

3.8 Pumping Lemma

Oder: *Wie zeigt man, dass eine Sprache nicht regulär ist?*

Für Typ-3-Sprachen können wir automatisch effiziente Recognizer synthetisieren.

Sind Typ-3-Grammatiken mächtig genug für den Entwurf von Programmiersprachen?

3.8 Pumping Lemma

Oder: *Wie zeigt man, dass eine Sprache nicht regulär ist?*

Für Typ-3-Sprachen können wir automatisch effiziente Recognizer synthetisieren.

Sind Typ-3-Grammatiken mächtig genug für den Entwurf von Programmiersprachen?

- Jede Programmiersprache braucht arithmethische Ausdrücke.
- Die Grammatik für arithmethische Ausdrücke aus Beispiel 2.11 ist nicht rechtslinear.
- Das zeigt jedoch noch nicht, dass die Sprache der arithmethischen Ausdrücke nicht regulär ist. Es könnte eine andere rechtslineare Grammatik geben (vielleicht viel größer als die aus Beispiel 2.11), die dieselbe Sprache erzeugt.

Satz 3.37 (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*

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

Beweis:

Sei $R = L(A)$, $A = (Q, \Sigma, \delta, q_0, F)$.

Sei $n = |Q|$. Sei nun $z = a_1 \dots a_m \in R$ mit $m \geq n$.

Die beim Lesen von z durchlaufene Zustandsfolge sei

$$q_0 = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \dots \xrightarrow{a_m} p_m$$

Beweis:

Sei $R = L(A)$, $A = (Q, \Sigma, \delta, q_0, F)$.

Sei $n = |Q|$. Sei nun $z = a_1 \dots a_m \in R$ mit $m \geq n$.

Die beim Lesen von z durchlaufene Zustandsfolge sei

$$q_0 = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \dots \xrightarrow{a_m} p_m$$

Dann muss es $0 \leq i < j \leq n$ geben mit $p_i = p_j$.

Beweis:

Sei $R = L(A)$, $A = (Q, \Sigma, \delta, q_0, F)$.

Sei $n = |Q|$. Sei nun $z = a_1 \dots a_m \in R$ mit $m \geq n$.

Die beim Lesen von z durchlaufene Zustandsfolge sei

$$q_0 = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \dots \xrightarrow{a_m} p_m$$

Dann muss es $0 \leq i < j \leq n$ geben mit $p_i = p_j$.

Wir teilen z wie folgt auf: $\underbrace{a_1 \dots a_i}_{u} \underbrace{a_{i+1} \dots a_j}_{v} \underbrace{a_{j+1} \dots a_{|z|}}_{w}$

Beweis:

Sei $R = L(A)$, $A = (Q, \Sigma, \delta, q_0, F)$.

Sei $n = |Q|$. Sei nun $z = a_1 \dots a_m \in R$ mit $m \geq n$.

Die beim Lesen von z durchlaufene Zustandsfolge sei

$$q_0 = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \dots \xrightarrow{a_m} p_m$$

Dann muss es $0 \leq i < j \leq n$ geben mit $p_i = p_j$.

Wir teilen z wie folgt auf: $\underbrace{a_1 \dots a_i}_{u} \underbrace{a_{i+1} \dots a_j}_{v} \underbrace{a_{j+1} \dots a_{|z|}}_{w}$

Damit gilt:

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

□

Beweis:

Sei $R = L(A)$, $A = (Q, \Sigma, \delta, q_0, F)$.

Sei $n = |Q|$. Sei nun $z = a_1 \dots a_m \in R$ mit $m \geq n$.

Die beim Lesen von z durchlaufene Zustandsfolge sei

$$q_0 = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \dots \xrightarrow{a_m} p_m$$

Dann muss es $0 \leq i < j \leq n$ geben mit $p_i = p_j$.

Wir teilen z wie folgt auf: $\underbrace{a_1 \dots a_i}_{u} \underbrace{a_{i+1} \dots a_j}_{v} \underbrace{a_{j+1} \dots a_{|z|}}_{w}$

Damit gilt:

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

□

[Darf A NFA sein?]

- Logische Struktur: für jede reguläre Sprache R

  $\exists n > 0.$

- Logische Struktur: für jede reguläre Sprache R

  $\exists n > 0.\ \forall z \in R$ mit $|z| \geq n$.

- Logische Struktur: für jede reguläre Sprache R

  $\exists n > 0.\ \forall z \in R$ mit $|z| \geq n.\ \exists u, v, w.\ z = uvw \wedge \ldots$

- Logische Struktur: für jede reguläre Sprache R

  $\exists n > 0.\ \forall z \in R$ mit $|z| \geq n.\ \exists u, v, w.\ z = uvw \wedge \ldots$

- Spielbasierte Formulierung.
  Sei R eine Sprache (regulär oder nicht). Eine Partie des R-Spiels zwischen zwei Spielern läuft so:
  - Spieler I wählt eine Zahl $n > 0$
  - Spieler II wählt ein Wort $z \in R$ mit $|z| \geq n$
  - Spieler I wählt eine Zerlegung uvw von z.

  Spieler I gewinnt, wenn uvw die gewünschten Eigenschaften erfüllt, sonst gewinnt Spieler II.

- Logische Struktur: für jede reguläre Sprache R

  $\exists n > 0.\ \forall z \in R$ mit $|z| \geq n.\ \exists u, v, w.\ z = uvw \wedge \ldots$

- Spielbasierte Formulierung.
  Sei R eine Sprache (regulär oder nicht). Eine Partie des R-Spiels zwischen zwei Spielern läuft so:
  - Spieler I wählt eine Zahl $n > 0$
  - Spieler II wählt ein Wort $z \in R$ mit $|z| \geq n$
  - Spieler I wählt eine Zerlegung uvw von z.

  Spieler I gewinnt, wenn uvw die gewünschten Eigenschaften erfüllt, sonst gewinnt Spieler II.

- Das Pumping Lemma sagt: für jede reguläre Sprache R, Spieler I hat eine *Gewinnstrategie* im R-Spiel, d.h., wenn er richtig spielt, dann gewinnt er jede Partie.

- Logische Struktur: für jede reguläre Sprache R

  $\exists n > 0.\ \forall z \in R$ mit $|z| \geq n.\ \exists u, v, w.\ z = uvw \wedge \ldots$

- Spielbasierte Formulierung.
  Sei R eine Sprache (regulär oder nicht). Eine Partie des R-Spiels zwischen zwei Spielern läuft so:
  - Spieler I wählt eine Zahl $n > 0$
  - Spieler II wählt ein Wort $z \in R$ mit $|z| \geq n$
  - Spieler I wählt eine Zerlegung uvw von z.

  Spieler I gewinnt, wenn uvw die gewünschten Eigenschaften erfüllt, sonst gewinnt Spieler II.

- Das Pumping Lemma sagt: für jede reguläre Sprache R, Spieler I hat eine *Gewinnstrategie* im R-Spiel, d.h., wenn er richtig spielt, dann gewinnt er jede Partie.

- Gewinnstrategie für Spieler I: Regeln, um erst eine Pumping-Lemma-Zahl n, und dann eine gute Zerlegung uvw zu wählen (eine Zerlegung, die die Eigenschaften erfüllt).

Falls $L(M) = R$ so ist $|Q_M|$ eine Pumping-Lemma-Zahl für R.

Falls $L(M) = R$ so ist $|Q_M|$ eine Pumping-Lemma-Zahl für R.

Anwendung des Pumping Lemmas:

- Um zu beweisen, dass R nicht regulär ist, zeige, dass Spieler I keine Gewinnstrategie hat.
  - Durch Widerspruch
  - Äquivalent: Zeige, dass Spieler II eine Gewinnstrategie hat (Regeln, um z in Abhängigkeit von R und n zu wählen).

Satz 3.38

Die Sprache $\{a^i b^i \mid i \in \mathbb{N}\}$ *ist nicht regulär.*

Satz 3.38

Die Sprache $\{a^i b^i \mid i \in \mathbb{N}\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.

Satz 3.38

Die Sprache $\{a^i b^i \mid i \in \mathbb{N}\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.
Sei n eine Pumping-Lemma-Zahl für L.

Satz 3.38

Die Sprache $\{a^i b^i \mid i \in \mathbb{N}\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.
Sei n eine Pumping-Lemma-Zahl für L.
Wähle $z = a^n b^n \in L$.

Satz 3.38

Die Sprache $\{a^i b^i \mid i \in \mathbb{N}\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.
Sei n eine Pumping-Lemma-Zahl für L.
Wähle $z = a^n b^n \in L$.
Jede gute Zerlegung uvw von z erfüllt
$u, v \in \{a\}^*$ (weil $|uv| \leq n$) und $v \neq \epsilon$.

Satz 3.38

Die Sprache $\{a^i b^i \mid i \in \mathbb{N}\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.
Sei n eine Pumping-Lemma-Zahl für L.
Wähle $z = a^n b^n \in L$.
Jede gute Zerlegung uvw von z erfüllt
$u, v \in \{a\}^*$ (weil $|uv| \leq n$) und $v \neq \epsilon$.
Damit müsste gelten $a^{n-|v|} b^n = uw \in L$. ↯ □

"Endliche Automaten können nicht unbegrenzt zählen"

Satz 3.38

Die Sprache $\{a^i b^i \mid i \in \mathbb{N}\}$ ist nicht regulär.

Beweis:

Angenommen, L sei doch regulär.
Sei n eine Pumping-Lemma-Zahl für L.
Wähle $z = a^n b^n \in L$.
Jede gute Zerlegung uvw von z erfüllt
$u, v \in \{a\}^*$ (weil $|uv| \leq n$) und $v \neq \epsilon$.
Damit müsste gelten $a^{n-|v|} b^n = uw \in L$. $\lightning$ □

"Endliche Automaten können nicht unbegrenzt zählen"

Satz 3.38

Die Sprache $\{a^i b^i \mid i \in \mathbb{N}\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.
Sei n eine Pumping-Lemma-Zahl für L.
Wähle $z = a^n b^n \in L$.
Jede gute Zerlegung uvw von z erfüllt
$u, v \in \{a\}^*$ (weil $|uv| \leq n$) und $v \neq \epsilon$.
Damit müsste gelten $a^{n-|v|}b^n = uw \in L$. $\lightning$ □

"Endliche Automaten können nicht unbegrenzt zählen"

Ist die Sprache $\{a^n b^n \mid n \leq 10^6\}$ regulär?

Satz 3.39

$L = \{0^{m^2} \mid m \geq 0\}$ *ist nicht regulär.*

Satz 3.39

$L = \{0^{m^2} \mid m \geq 0\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.

Satz 3.39

$L = \{0^{m^2} \mid m \geq 0\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.
Sei n eine Pumping-Lemma-Zahl für L.

Satz 3.39

$L = \{0^{m^2} \mid m \geq 0\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.
Sei n eine Pumping-Lemma-Zahl für L.
Wähle $z = 0^{n^2} \in L$. Sei uvw eine Zerlegung von z mit

$$1 \leq |v| \leq |uv| \leq n$$

Wir zeigen $uv^2w \notin L$. D.h., wir beweisen, dass $|uv^2w|$ keine Quadratzahl ist.

Satz 3.39

$L = \{0^{m^2} \mid m \geq 0\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.
Sei n eine Pumping-Lemma-Zahl für L.
Wähle $z = 0^{n^2} \in L$. Sei uvw eine Zerlegung von z mit

$$1 \leq |v| \leq |uv| \leq n$$

Wir zeigen $uv^2w \notin L$. D.h., wir beweisen, dass $|uv^2w|$ keine Quadratzahl ist. Mit

$$n^2 = |z| = |uvw| < |uv^2w| \leq n^2 + n < n^2 + 2n + 1 = (n+1)^2$$

gilt $n^2 < |uv^2w| < (n+1)^2$. Zwischen n^2 und $(n+1)^2$ liegt keine Quadratzahl. □

Satz 3.40

Die Sprache der wohlgeklammerten Ausdrücke über dem Alphabet $\{(,)\}$ *ist nicht regulär.*

Beweis:

Aufgabe. □

Satz 3.40

Die Sprache der wohlgeklammerten Ausdrücke über dem Alphabet $\{(,)\}$ *ist nicht regulär.*

Beweis:

Aufgabe. □

Satz 3.41

Die Sprache $Arith$ *der arithmetischen Ausdrücken ist nicht regulär.*

Satz 3.40

Die Sprache der wohlgeklammerten Ausdrücke über dem Alphabet $\{(,)\}$ *ist nicht regulär.*

Beweis:

Aufgabe. □

Satz 3.41

Die Sprache $Arith$ *der arithmetischen Ausdrücken ist nicht regulär.*

Beweis:

Angenommen, $Arith$ sei doch regulär.
Dann gibt es einen DFA A mit $L(A) = Arith$.
Ersetze alle Transitionen von A, die nicht mit (oder) beschriftet sind durch ϵ-Transitionen.
Der resultierende ϵ-NFA erkennt die Sprache der wohlgeklammerten Ausdrücke. Widerspruch zu Satz 3.40. □

Bemerkung

Es gibt nicht-reguläre Sprachen, für die das Pumping-Lemma gilt!

$\Rightarrow$ Pumping-Lemma hinreichend aber nicht notwendig um Nicht-Regularität zu zeigen.

$$\text{regulär} \subset \text{Pumping-Lemma gilt} \subset \text{alle Sprachen}$$