Einführung in die Theoretische Informatik

Tobias Nipkow + Helmut Seidl + Javier Esparza +
Jan Křetínský

Fakultät für Informatik
TU München

Sommersemester 2018

© T. Nipkow / H. Seidl / J. Esparza / J. Kretinsky

Kapitel I Organisatorisches

Siehe https://www7.in.tum.de/um/courses/theo/ss2018

Termine

- Vorlesung:
  - Mo 10:10–11:40 im MW 0001
  - Do 14:20–15:50 im MW 0001
- Sprechstunde: Do 16:00–16:45 im MI HS1
- Klausur: Fr 03.08.2018
- Wiederholungsklausur: Mi 10.10.2018
- Übungen

Übungskonzept

- Erfahrung aus letzten Jahren:
  - Sehr unterschiedliche Wissenstände unter den Studierenden
  - Zu wenig Zeit, um Aufgaben ausreichend zu bearbeiten

Übungskonzept

- Erfahrung aus letzten Jahren:
  - Sehr unterschiedliche Wissenstände unter den Studierenden
  - Zu wenig Zeit, um Aufgaben ausreichend zu bearbeiten
- Jeder Student kann abhängig von seinem Wissensstand üben:

	Unterstützung		Übung	
Aufgabentyp	Tools	Hilfegr.	Übungsgr.	Vertiefungsgr.
A. Wissen	✓	✓		
B. Verständnis	✓	✓		
C. Anwenden	✓	✓	✓	✓
D. Analysieren			✓	✓
E. Beweisen			✓	✓

- Aufgaben werden entsprechend gekennzeichnet
- Tools und Hilfegruppen sind Unterstützung beim Selbststudium und reichen alleine nicht aus, um die Klausur zu bestehen

Übungskonzept

- Erfahrung aus letzten Jahren:
  - Sehr unterschiedliche Wissenstände unter den Studierenden
  - Zu wenig Zeit, um Aufgaben ausreichend zu bearbeiten
- Jeder Student kann abhängig von seinem Wissensstand üben:

	Unterstützung		Übung	
Aufgabentyp	Tools	Hilfegr.	Übungsgr.	Vertiefungsgr.
A. Wissen	✓	✓		
B. Verständnis	✓	✓		
C. Anwenden	✓	✓	✓	✓
D. Analysieren			✓	✓
E. Beweisen			✓	✓

- Aufgaben werden entsprechend gekennzeichnet
- Tools und Hilfegruppen sind Unterstützung beim Selbststudium und reichen alleine nicht aus, um die Klausur zu bestehen
- Gehen Sie vorbereitet in die Übungs- und Vertiefungsgruppen!

Übungsanmeldung

- Keine Anmeldung zu den Hilfegruppen

Zeit	Ort	Tutor	Nachname von-bis
MO 12:00-14:00	00.08.038	Philipps, Claudia	A-I
MO 12:00-14:00	00.08.059	Seidel, Ina	K-R
MO 12:00-14:00	02.07.014	Schulz, Jakob	S-Z
MO 16:00-18:00	00.13.036	Weininger, Maxi	A-N
MO 16:00-18:00	03.09.014	Jaax, Stefan	O-Z

Übungsanmeldung

- Keine Anmeldung zu den Hilfegruppen

Zeit	Ort	Tutor	Nachname von-bis
MO 12:00-14:00	00.08.038	Philipps, Claudia	A-I
MO 12:00-14:00	00.08.059	Seidel, Ina	K-R
MO 12:00-14:00	02.07.014	Schulz, Jakob	S-Z
MO 16:00-18:00	00.13.036	Weininger, Maxi	A-N
MO 16:00-18:00	03.09.014	Jaax, Stefan	O-Z

- Konzentration der Tutorübungen auf die erfahrungsgemäß beliebten Tage: Montag, Mittwoch und Donnerstag (aufgeteilt auf zwei LV in TUMonline)
- Anmeldung über TUMonline
  - Beginn: Do 12.04. 19:00/19:30
  - Ende: Sa 14.04. 23:59
  - FCFS-Verfahren
- Anmeldung für ihre bevorzugte Gruppenstufe

Hausaufgaben

- Wöchentliche Hausaufgaben
- Abwechselnd Programmier- und handschriftliche Hausaufgaben
- Notenbonus

Hausaufgaben - Abgabe

- Abgabe der handschriftlichen Hausaufgaben in Zweier-Teams
  - Festlegung der Teams in der ersten Übungswoche
  - Sie dürfen gruppenübergreifend Teams bilden
  - Bei Einzelabgaben kein Anspruch auf Korrektur
  - Jedes Teammitglied muss 3 der 6 handschriftlichen Abgaben selbst geschrieben haben
  - Formale Kriterien (Deckblatt, etc.) auf der Website
- Abgabe der Programmierhausaufgaben **alleine**
  - Details zur Abgabe der Programmierhausaufgaben auf dem ersten Hausaufgabenblatt

Hausaufgaben

- Plagiieren Sie nicht! (Programmierhausaufgaben von *jedem* Studenten selber verfasst)
- Bei Krankheit kein Anspruch auf Nachprüfung

Notenbonus

- Nur auf bestandene Klausuren anrechenbar
- Nicht besser als 1,0
- Für beide Klausuren im SoSe18
- Maximal 60 Punkte in den Hausaufgaben (30 handschriftlich, 30 programmieren)

Bonus	Von	Bis
0,0	0	39,5
0,3	40	46,5
0,6	47	53,5
1,0	54	60

Klausur

- Schriftliche Klausur
- Keine zusätzlichen Hilfsmittel
- Anmeldung über TUMonline

Fragen, Wünsche, Beschwerden und Feedback an
Tutoren, Hilfegruppen, Sprechstunde oder
theo-uebungsleitung@in.tum.de
(in dieser Priorisierung)

Inhalt und Gliederung der Vorlesung

- Endliche Automaten und reguläre Ausdrücke
  - Grundlagen der Textanalyse, der lexikalischen Analyse von Programmiersprachen, und der Spezifikation und Analyse von Kommunikationsprotokollen.
- Kontextfrei Grammatiken
  - Grundlagen der syntaktische Analyse von Programmiersprachen, Parsing, Compilerbau.
- Theorie der Berechenbarkeit
  - Untersuchung der Grenzen, was Rechner prinzipiell können.
- Komplexitätstheorie
  - Untersuchung der Grenzen, was Rechner mit begrenzten Ressourcen können.

Geschichte:

- 1936 Berechenbarkeitstheorie: Church & Turing
- 1956 Automaten und reguläre Ausdrücke: Kleene
- 1956 Grammatiken: Chomsky
- 1971 Komplexitätstheorie: Cook

Vorkenntnisse und weiterführende Vorlesungen

- Vorkenntnisse:
  - Einführung in die Informatik 1
  - Diskrete Strukturen
- Weiterführende Vorlesungen:
  - Automata und Formal Languages
  - Complexity Theory
  - Logic
  - Model Checking
  - Quantitative Verification
  - Compiler Construction
  - ...

Worum geht es in Theo?

Der Zweck der Einführung in die theoretische Informatik ist, dass Sie lernen:

- Grundkonzepte der Informatik formal zu beschreiben und anzuwenden
- Strukturierte Problemlösungen zu entwerfen
- die Korrektheit von Aussagen zu beweisen
- das Gelernte auf neue Probleme anzuwenden

Es geht nicht darum Wissen anzuhäufen, sondern darum Kompetenzen im

- Beschreiben,
- Erklärungen,
- Analysieren,
- Argumentieren,
- Beweisen und
- Anwenden von Wissen zu vertiefen.

Dies ist nicht immer einfach und erfordert viel Üben!

Beispiele von konkreten Zielen

Abstraktion von *irrelevanten* Details:

Beispiele von konkreten Zielen

Abstraktion von *irrelevanten* Details:
zB nicht x86 sondern Turingmaschine.

Beispiele von konkreten Zielen

Abstraktion von *irrelevanten* Details:
zB nicht x86 sondern Turingmaschine.

Formalisierung durch mathematische Objekte (Mengen, Funktionen, Relationen)

Beispiele von konkreten Zielen

Abstraktion von *irrelevanten* Details:
zB nicht x86 sondern Turingmaschine.

Formalisierung durch mathematische Objekte (Mengen, Funktionen, Relationen)

Simulation eines Formalismus durch einen anderen:

Beispiele von konkreten Zielen

Abstraktion von *irrelevanten* Details:
zB nicht x86 sondern Turingmaschine.

Formalisierung durch mathematische Objekte (Mengen, Funktionen, Relationen)

Simulation eines Formalismus durch einen anderen:
zB deterministische durch nichtdeterministische Maschinen

Beispiele von konkreten Zielen

Abstraktion von *irrelevanten* Details:
zB nicht x86 sondern Turingmaschine.

Formalisierung durch mathematische Objekte (Mengen, Funktionen, Relationen)

Simulation eines Formalismus durch einen anderen:
zB deterministische durch nichtdeterministische Maschinen

Äquivalenz von Formalismen:

Beispiele von konkreten Zielen

Abstraktion von *irrelevanten* Details:
zB nicht x86 sondern Turingmaschine.

Formalisierung durch mathematische Objekte (Mengen, Funktionen, Relationen)

Simulation eines Formalismus durch einen anderen:
zB deterministische durch nichtdeterministische Maschinen

Äquivalenz von Formalismen:
zB endliche Automaten und reguläre Ausdrücke.

Beispiele von konkreten Zielen

Abstraktion von *irrelevanten* Details:
zB nicht x86 sondern Turingmaschine.

Formalisierung durch mathematische Objekte (Mengen, Funktionen, Relationen)

Simulation eines Formalismus durch einen anderen:
zB deterministische durch nichtdeterministische Maschinen

Äquivalenz von Formalismen:
zB endliche Automaten und reguläre Ausdrücke.

Reduktion von einem Problem auf ein anderes:

Beispiele von konkreten Zielen

Abstraktion von *irrelevanten* Details:
zB nicht x86 sondern Turingmaschine.

Formalisierung durch mathematische Objekte (Mengen, Funktionen, Relationen)

Simulation eines Formalismus durch einen anderen:
zB deterministische durch nichtdeterministische Maschinen

Äquivalenz von Formalismen:
zB endliche Automaten und reguläre Ausdrücke.

Reduktion von einem Problem auf ein anderes:
zB Formeln auf Automaten, ...

Beispiele von konkreten Zielen

Abstraktion von *irrelevanten* Details:
zB nicht x86 sondern Turingmaschine.

Formalisierung durch mathematische Objekte (Mengen, Funktionen, Relationen)

Simulation eines Formalismus durch einen anderen:
zB deterministische durch nichtdeterministische Maschinen

Äquivalenz von Formalismen:
zB endliche Automaten und reguläre Ausdrücke.

Reduktion von einem Problem auf ein anderes:
zB Formeln auf Automaten, ...

Endliche Beschreibungen unendlicher Mengen

Literatur

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

Dexter Kozen.
Automata and Computability.

Michael Sipser.
Introduction to the Theory of Computation.

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

Uwe Schöning.
Theoretische Informatik — kurzgefasst.

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

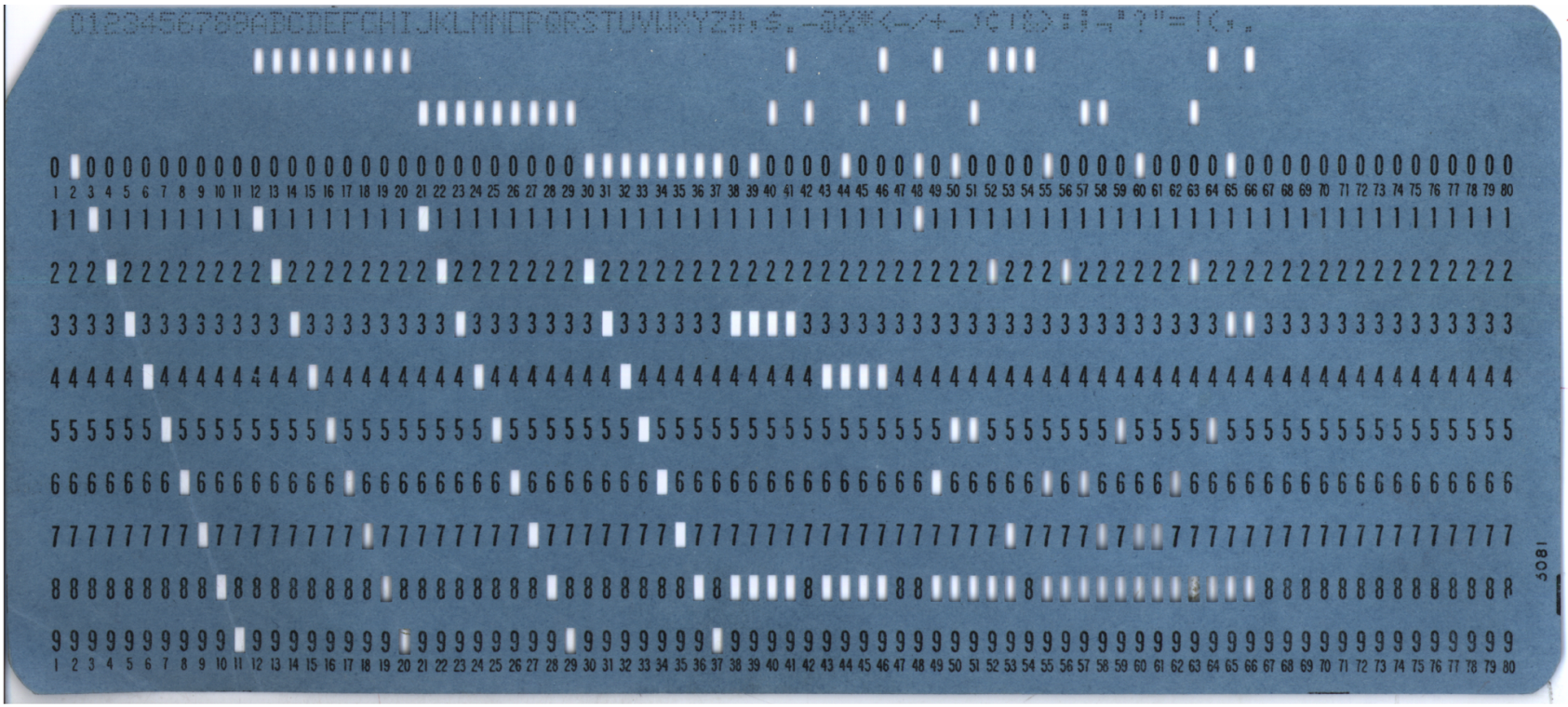


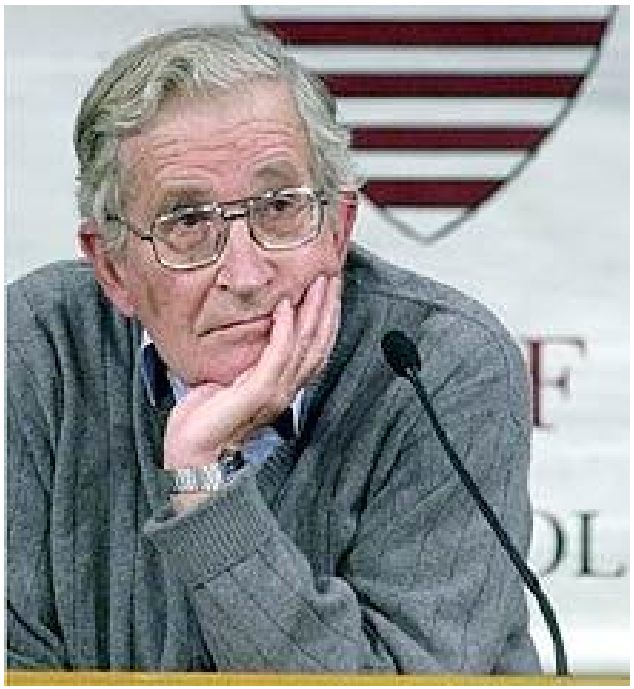
Abbildung: Programmieren mit Lochkarten in den 1960er Jahre

- In den 1950er setzt sich die Idee durch,

  Programmiersprachen

  für die Kommunikation mit Maschinen zu entwickeln: Fortran (FORmula TRANslator, 1954), Lisp (LISt Processor, 1959), COBOL (COmmon Business Oriented Language, 1959).
- 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 Professor für Linguistik am MIT und ein bedeutender Sprachwissenschaftler.
Neben seiner linguistischen Arbeit gilt Chomsky als einer der bedeutendsten Intellektuellen Nordamerikas und ist als scharfer Kritiker der US-amerikanischen Außenpolitik bekannt.
[Quelle: Wikipedia]

- 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 ist keine Sprache

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: $\{ab, b\}\{a, bb\} = \{aba, abbb,$

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: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \cdots w_n \mid w_1, \ldots, w_n \in A\} = \underbrace{A \cdots A}_{n}$ 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: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \cdots w_n \mid w_1, \ldots, w_n \in A\} = \underbrace{A \cdots A}_{n}$ Bsp:

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

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: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \cdots w_n \mid w_1, \ldots, w_n \in A\} = \underbrace{A \cdots A}_{n}$ Bsp:

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

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: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \cdots w_n \mid w_1, \ldots, w_n \in A\} = \underbrace{A \cdots A}_{n}$ Bsp:

  $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \cdots w_n \mid n \geq 0 \wedge w_1, \ldots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \ldots\} \neq \{0, 1\}^*$
- $A^+ = AA^* = \bigcup_{n \geq 1} A^n$
  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: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \cdots w_n \mid w_1, \ldots, w_n \in A\} = \underbrace{A \cdots A}_{n}$ Bsp:

  $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \cdots w_n \mid n \geq 0 \wedge w_1, \ldots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \ldots\} \neq \{0, 1\}^*$
- $A^+ = AA^* = \bigcup_{n \geq 1} A^n$
  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: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \cdots w_n \mid w_1, \ldots, w_n \in A\} = \underbrace{A \cdots A}_{n}$ Bsp:

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

- $V = \{\langle \text{Expr} \rangle, \langle \text{Term} \rangle, \langle \text{Faktor} \rangle\}$
- $\Sigma = \{a, b, c, +, *, (,)\}$
- $S = \langle \text{Expr} \rangle$
- 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 $a * (b + c)$:

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

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

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 $a * (b + c)$:

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

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)

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

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

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

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

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 $a * (b + c)$:

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

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

Eine Grammatik für $\{a^n b^n c^n \mid n \geq 0\}$

Welche ist richtig?

$$\begin{array}{rcl} S & \rightarrow & abcS \mid \epsilon \\ ba & \rightarrow & ab \\ cb & \rightarrow & bc \end{array}$$

Beispiel 2.11

Eine Grammatik für $\{a^n b^n c^n \mid n \geq 0\}$

Welche ist richtig?

$$\begin{array}{rcl} S & \to & abcS \mid \epsilon \\ ba & \to & ab \\ cb & \to & bc \end{array}$$

$$\begin{array}{rcl} S & \to & aBSc \mid \epsilon \\ Ba & \to & aB \\ Bb & \to & bb \\ Bc & \to & bc \end{array}$$

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 $|\alpha| \leq |\beta|$
für jede Produktion $\alpha \to \beta$ unterschiedlich von
$S \to \epsilon$.

2.2 Die Chomsky-Hierarchie

Eine Grammatik G ist vom

Typ 0 immer.

Typ 1 falls $|\alpha| \leq |\beta|$
für jede Produktion $\alpha \to \beta$ unterschiedlich von
$S \to \epsilon$.

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

2.2 Die Chomsky-Hierarchie

Eine Grammatik G ist vom

Typ 0 immer.

Typ 1 falls $|\alpha| \leq |\beta|$
für jede Produktion $\alpha \to \beta$ unterschiedlich von
$S \to \epsilon$.

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

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

2.2 Die Chomsky-Hierarchie

Eine Grammatik G ist vom

- Typ 0 immer.
- Typ 1 falls $|\alpha| \leq |\beta|$
  für jede Produktion $\alpha \to \beta$ unterschiedlich von $S \to \epsilon$.
- Typ 2 falls G vom Typ 1 ist und $\alpha \in V$ gilt
  für jede Produktion $\alpha \to \beta$.
- Typ 3 falls G vom Typ 2 ist und $\beta \in \Sigma \cup \Sigma V$ gilt
  für jede Produktion $\alpha \to \beta$ unterschiedlich von $S \to \epsilon$.

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
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

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

Das Erkennungsproblem wird durch das Wortproblem formalisiert:

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

Das Erkennungsproblem wird durch das Wortproblem formalisiert:

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

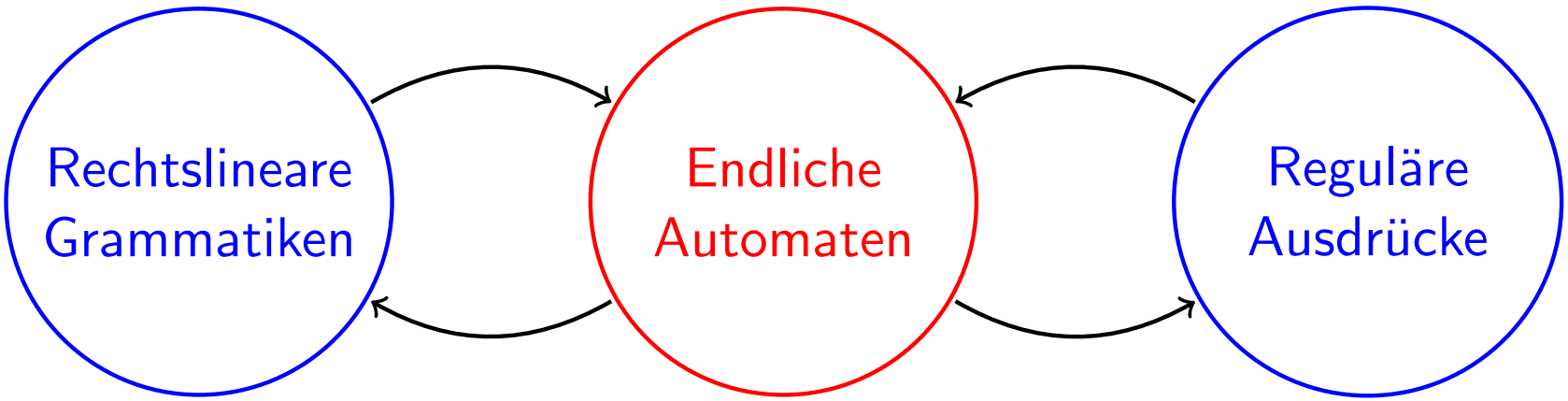
In den kommenden Wochen untersuchen wir das Wortproblem für Grammatiken von Typ 3 und Typ 2.

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

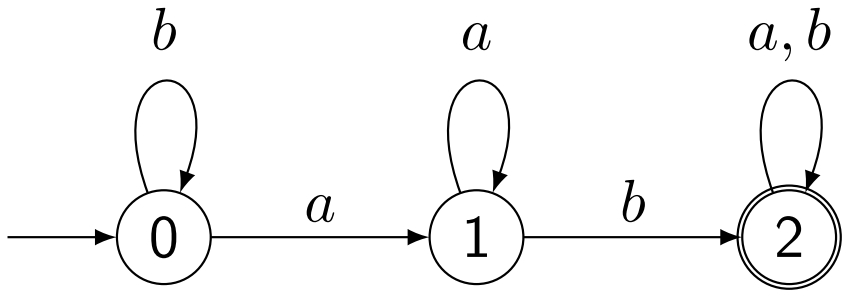
A_G ist eine abstrakte Beschreibung eines Programms zur Lösung des Erkennungsproblems.
(abstrakt = unabhänging von jeder Programmiersprache)

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.

DFA $\rightarrow$ Recognizer, der 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.)

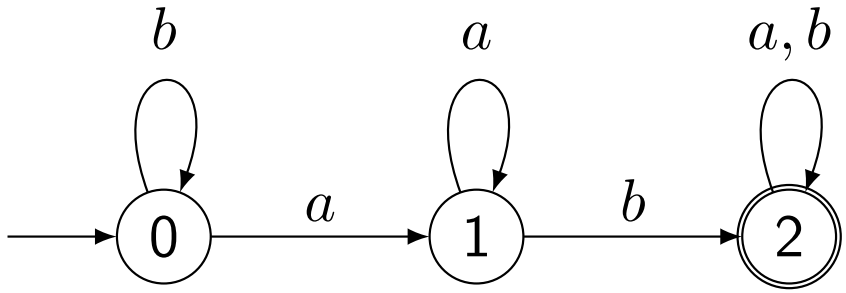
Eigenschaften von $\hat{\delta}$:

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

Graphische Darstellung

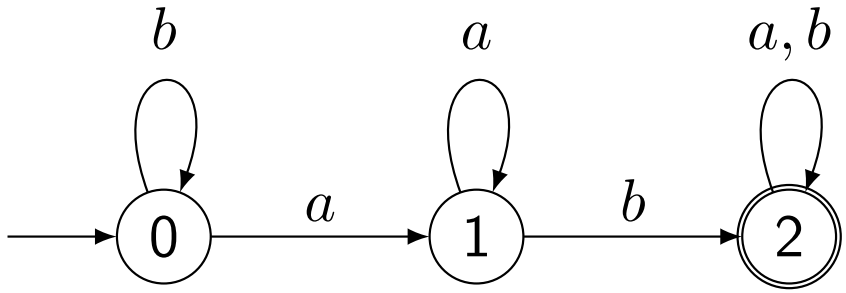
- Endliche Automaten können durch gerichtete und markierte Zustandsgraphen veranschaulicht werden:
  - Knoten $\widehat{=}$ Zuständen
  - Kanten $\widehat{=}$ Übergängen: $p \xrightarrow{a} q \;\widehat{=}\; \delta(p, a) = q$
- Der Anfangszustand wird durch einen Pfeil, Endzustände werden durch doppelte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

Beispiel 3.3



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

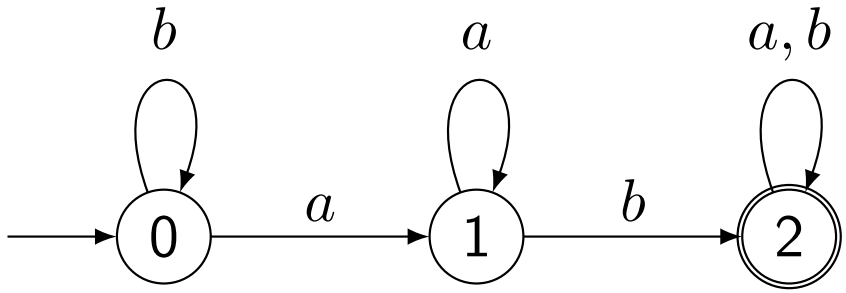
Beispiel 3.3



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).
  Ummitelbar 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).
  Ummitelbar davor wird a gelesen und unmittelbar danach b
- Jedes Wort, das ab enthält, wird akzeptiert.
  Für alle $q \in \{0, 1, 2\}$ gilt: $\hat{\delta}(q, ab) = 2$.
  Für alle $\alpha \in \{a, b\}^*$ gilt: $\hat{\delta}(2, \alpha) = 2$.

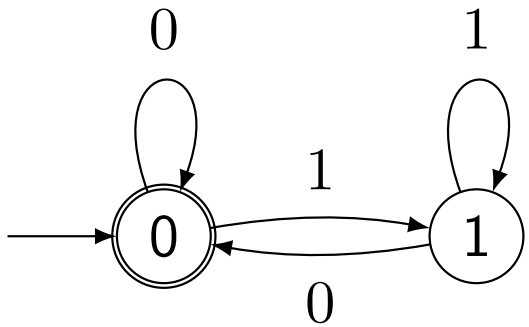
Beispiel 3.4

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

Der DFA A akzeptiert genau die Wörter $w \in \{0,1\}^*$ mit: $\#w$ ist gerade.

Beispiel 3.4

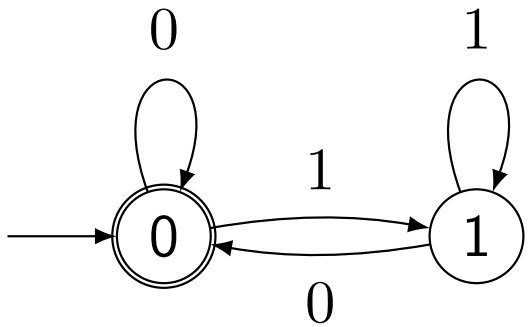
Für $w \in \{0,1\}^*$ sei $\#w$ die von w binär repräsentierte Zahl, zB $\#100 = 4$.



Der DFA A akzeptiert genau die Wörter $w \in \{0,1\}^*$ mit: $\#w$ ist gerade.

Beispiel 3.4

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



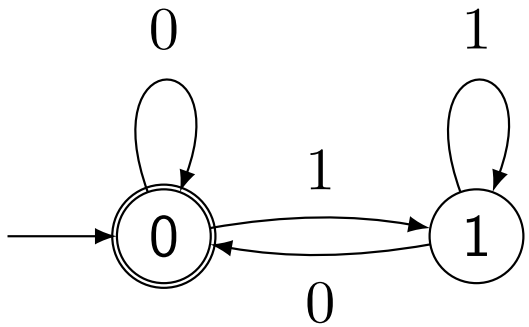
Der DFA A akzeptiert genau die Wörter $w \in \{0,1\}^*$ mit: $\#w$ ist gerade.

Beweis:

Für alle $w \neq \epsilon$, $\#w$ ist gerade gdw. w endet mit 0.
1. $\hat{\delta}(0, \alpha 0) =$

Beispiel 3.4

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



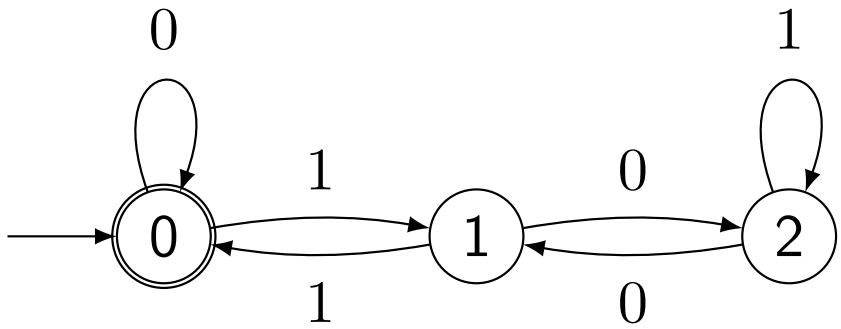
Der DFA A akzeptiert genau die Wörter $w \in \{0,1\}^*$ mit: $\#w$ ist gerade.

Beweis:

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

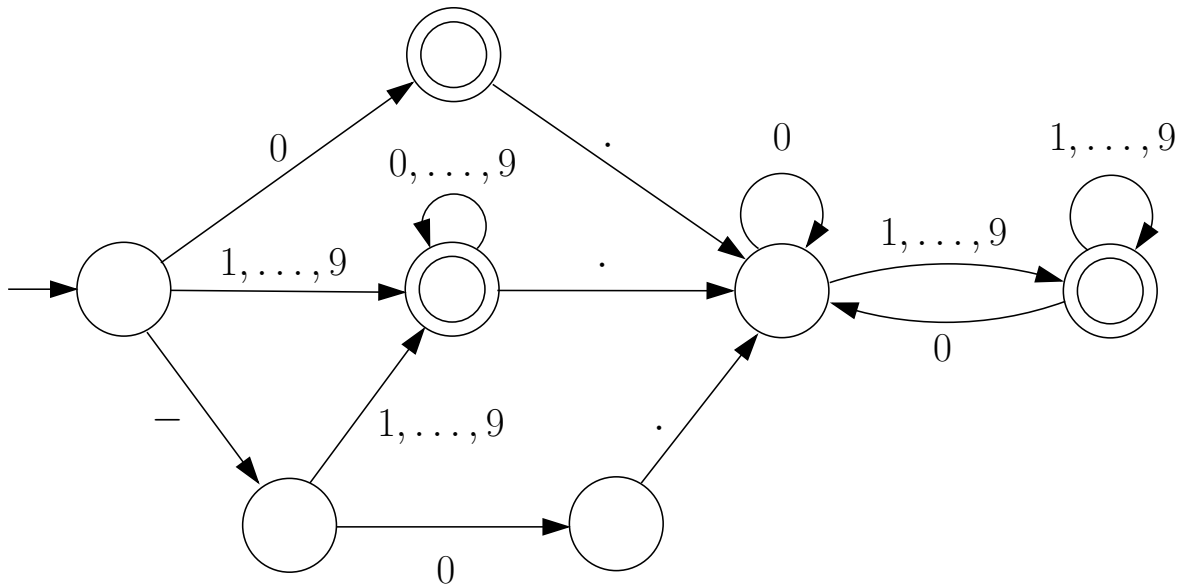
Für $w = \epsilon$, $\hat{\delta}(0, w) = 0$, daher $\epsilon \in L(A)$ und $\#\epsilon = 0$. □

Beispiel 3.5



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

Beispiel 3.6



Ein DFA für die Dezimalzahlen

3.2 Von rechtslinearen Grammatiken zu DFA (und zurück)

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)

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 $L(G) = L(M)$.

Satz 3.7

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 $L(M) = L(G)$:

Satz 3.7

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 $L(M) = L(G)$:
Es gilt (Induktion über n)

$$\hat{\delta}(q_0, a_1 \dots a_n) \in F$$

gdw

$$q_0 \to_G a_1 q_1 \to_G a_1 a_2 q_2 \to_G \cdots \to_G a_1 \cdots a_n q_n \to_G a_1 \cdots a_n$$

ist eine Ableitung von G.

Damit gilt $L(G) = L(A)$. □

3.3 Von rechtslinearen Grammatiken zu DFA (und zurück)

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 $L(G) = L(M)$.

3.3 Von rechtslinearen Grammatiken zu DFA (und zurück)

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 $L(G) = L(M)$.

Beispiel

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

3.3 Von rechtslinearen Grammatiken zu DFA (und zurück)

Wir zeigen:

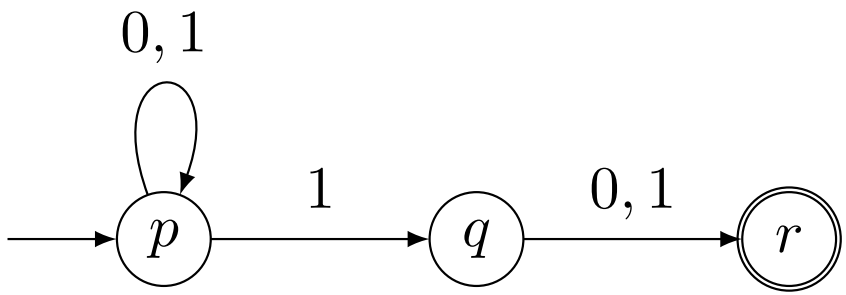
- Für jede rechtslineare Grammatik G gibt es einen DFA M mit $L(M) = L(G)$.
- Für jeden DFA M gibt es eine rechtslineare Grammatik G mit $L(G) = L(M)$.

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 $L(M) = L(N)$.

NFA sind eine Verallgemeinerung von DFA: Aus einem Zustand eines NFAs können $0, 1, 2,$ oder mehr Transitionen mit derselben Beschriftung ausgehen.

Beispiel 3.8



Ein Wort wird akzeptiert gdw es einen Weg zum Endzustand gibt.
Intuitive Vorstellung: Der Automat "rät" den richtigen Weg.

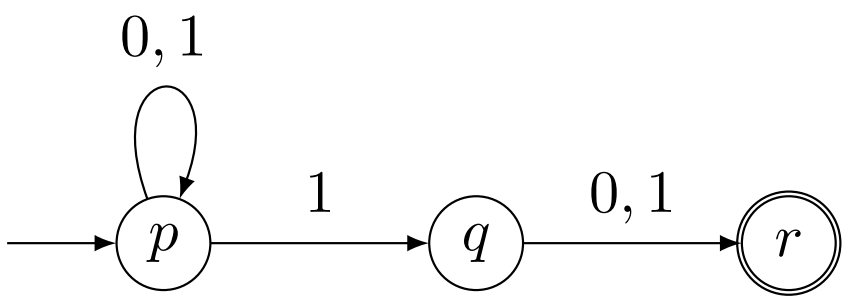
Nicht nutzlich als Recognizer, aber doch als Zwischenschritt (oder als Datenstruktur).

Definition 3.9

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

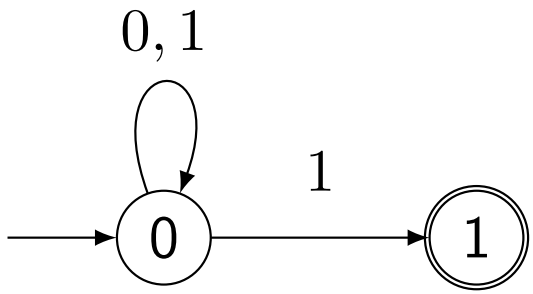
- Q, Σ, q_0 und F sind wie bei einem DFA
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$
  $\mathcal{P}(Q)$ = Menge aller Teilmengen von $Q = 2^Q$.
  Alternative: Relation $\delta \subseteq Q \times \Sigma \times Q$.

Beispiel



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

Beispiel 3.10



Der DFA A akzeptiert genau die Wörter $w \in \{0,1\}^*$ mit: $\#w$ ist ungerade.

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

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

Intuition:

- $\bar{\delta}(S, a)$ ist die Menge aller Zustände, die sich aus mindestens einem Zustand in S mit dem Alphabetzeichen a erreichen lassen.
- $\hat{\bar{\delta}}(S, w)$ ist die Menge aller Zustände, die sich aus mindestens einem Zustand in S mit dem Wort w erreichen lassen.

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

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

Intuition:

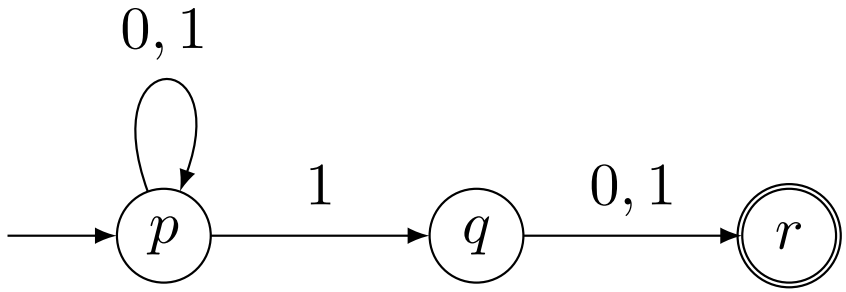
- $\bar{\delta}(S, a)$ ist die Menge aller Zustände, die sich aus mindestens einem Zustand in S mit dem Alphabetzeichen a erreichen lassen.
- $\hat{\bar{\delta}}(S, w)$ ist die Menge aller Zustände, die sich aus mindestens einem Zustand in S mit dem Wort w erreichen lassen.

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

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

Um Tod durch Notation zu vermeiden schreiben wir oft nur δ statt $\bar{\delta}$ und $\hat{\delta}$ statt $\hat{\bar{\delta}}$.

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 3.11

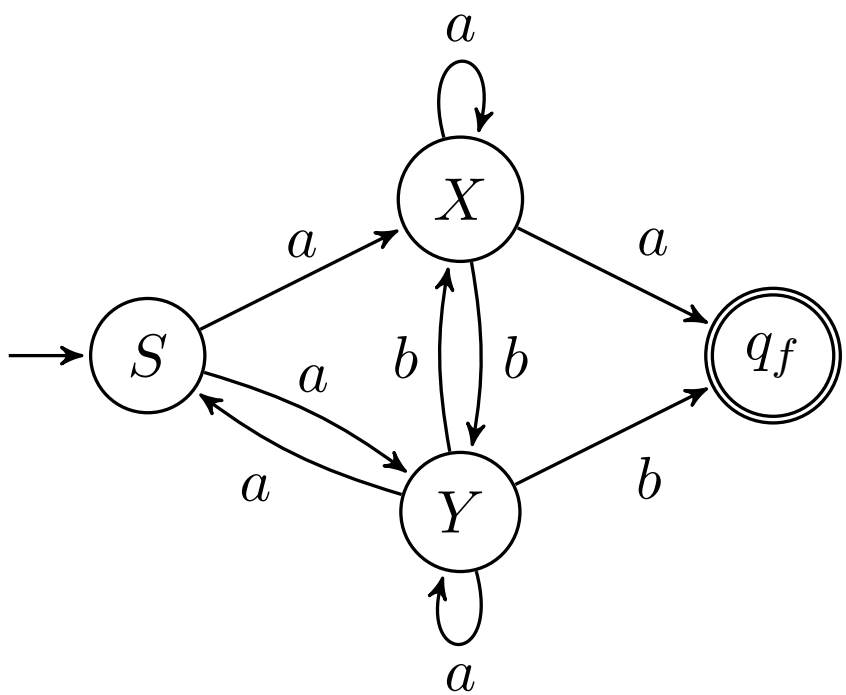
Für jede rechtslineare Grammatik G gibt es einen NFA M mit $L(G) = L(M)$.

Satz 3.11

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

Beispiel

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

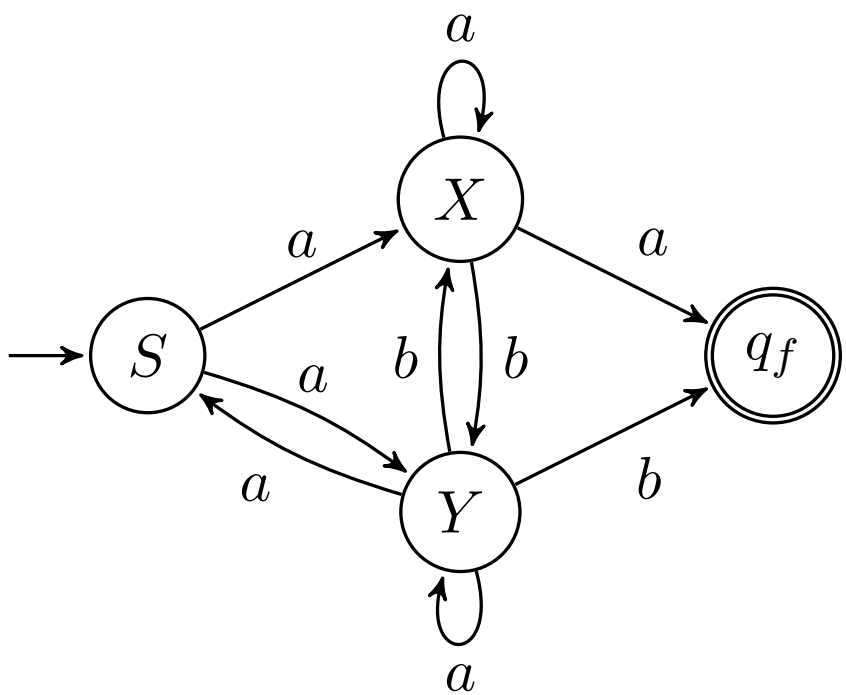


Satz 3.11

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

Beispiel

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



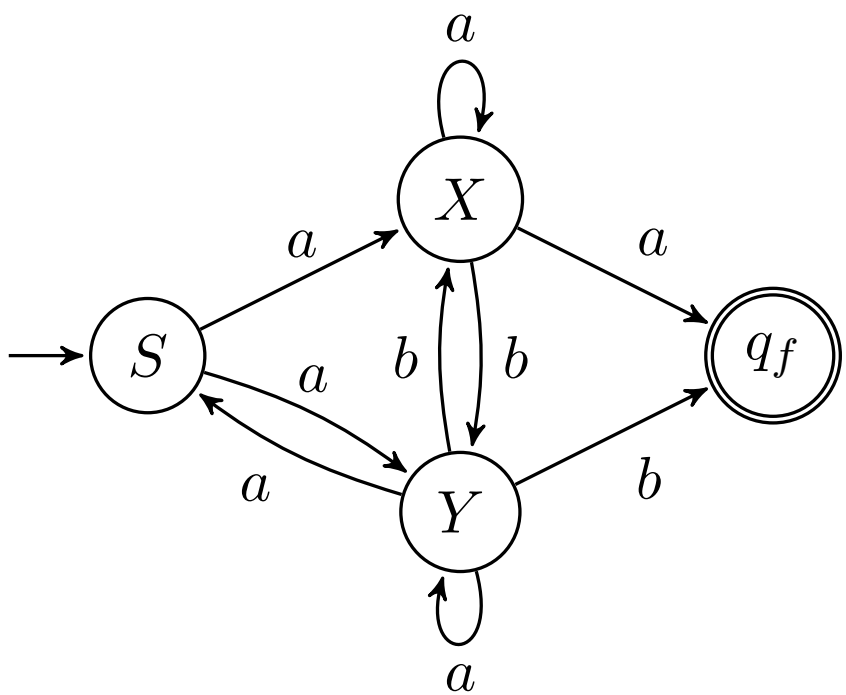
Aufgabe: Und wenn $S \to aX \mid aY \mid \epsilon$?

Satz 3.11

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

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 $(X \to a) \in P$
- $q_0 = S$
- $F = \{q_f\}$

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)

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.

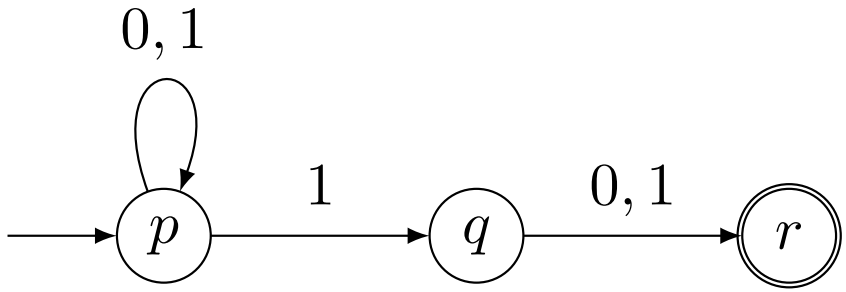
Satz 3.12

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

Satz 3.12

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:

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

□

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:

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

□

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.

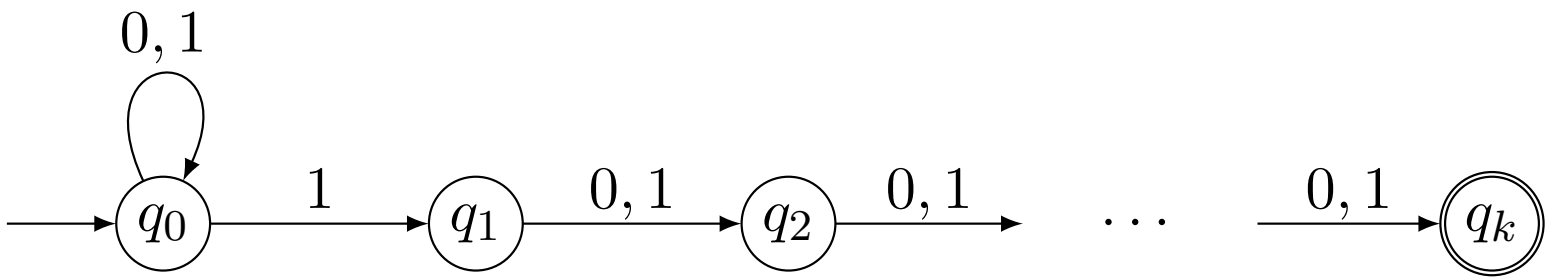
Beispiel 3.13

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

Beispiel 3.13

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

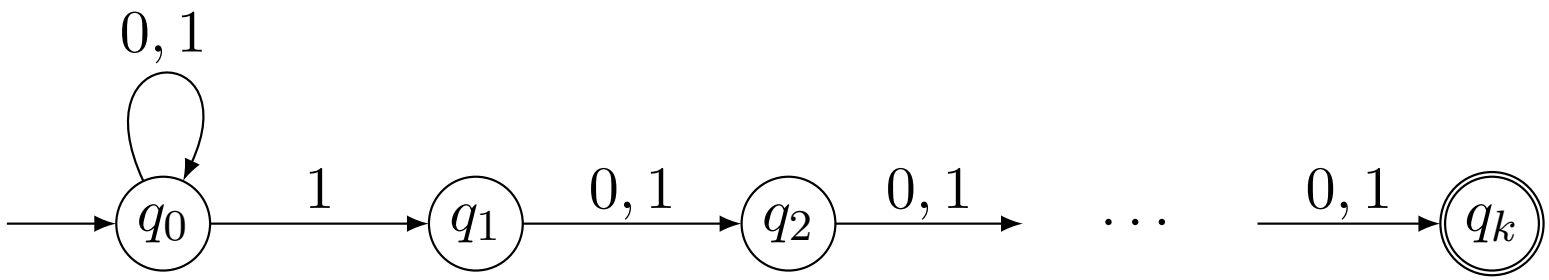
Ein NFA für diese Sprache ist gegeben durch:



Beispiel 3.13

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

Ein NFA für diese Sprache ist gegeben durch:

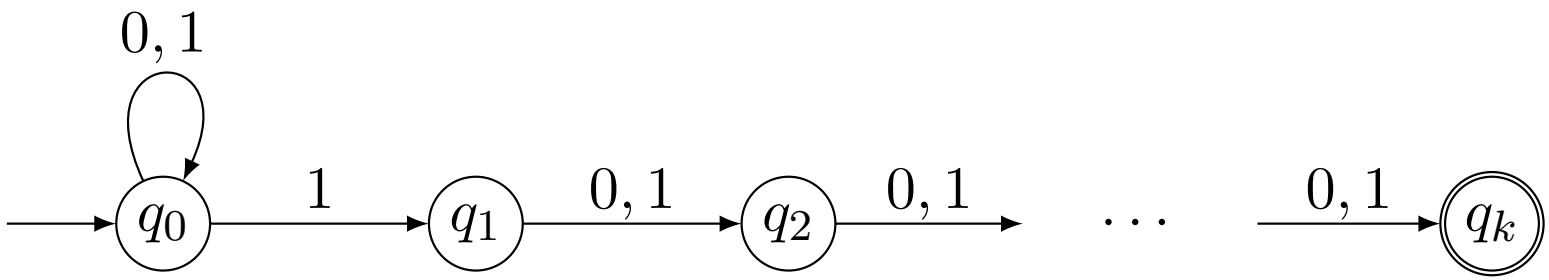


Die Konstruktion liefert einen DFA für L_k mit 2^k Zuständen. Geht es kompakter?

Beispiel 3.13

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

Ein NFA für diese Sprache ist gegeben durch:



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

Lemma 3.14

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

Im *schlimmsten* Fall ist ein exponentieller Sprung unvemeidlich.

Beweis:

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

Beweis:

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:

- Es folgt: M hat mindestens 2^k Zustände.

Beweis:

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.

□

3.4 NFAs mit ϵ-Übergängen

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

Beispiel 3.15

Ein kleines Fragment der lexikalischen Grammatik von Java:

```
BasicType:
    byte
    short
    char
    int
    long
    float
    double
    boolean
```

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

Definition 3.16

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

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

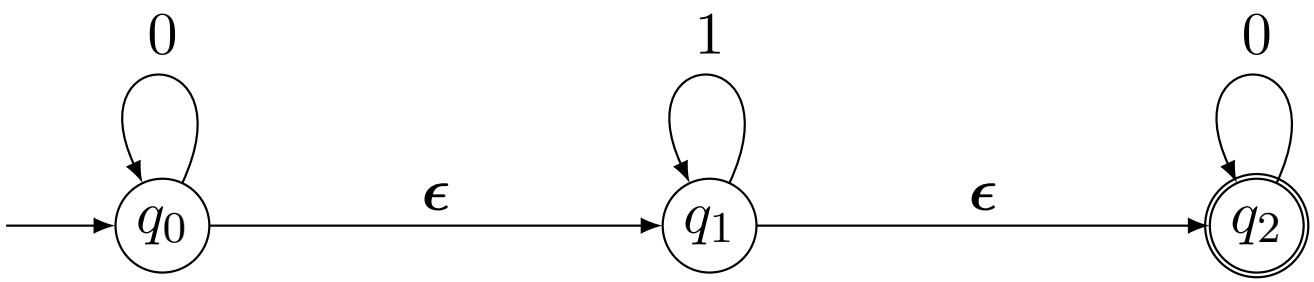
Ein ϵ-Übergang darf ausgeführt werden, ohne dass ein Eingabezeichen gelesen wird.

Definition 3.16

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.



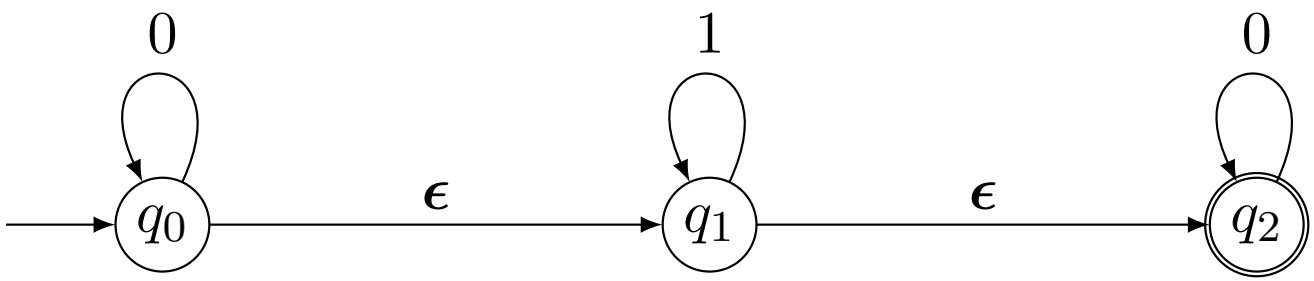
Akzeptiert: ϵ, 00, 11, ... Nicht akzeptiert: 101, ...

Definition 3.16

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 3.17

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

Lemma 3.17

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 3.17

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':

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

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

- Falls N das leere Wort ϵ akzeptiert, also falls

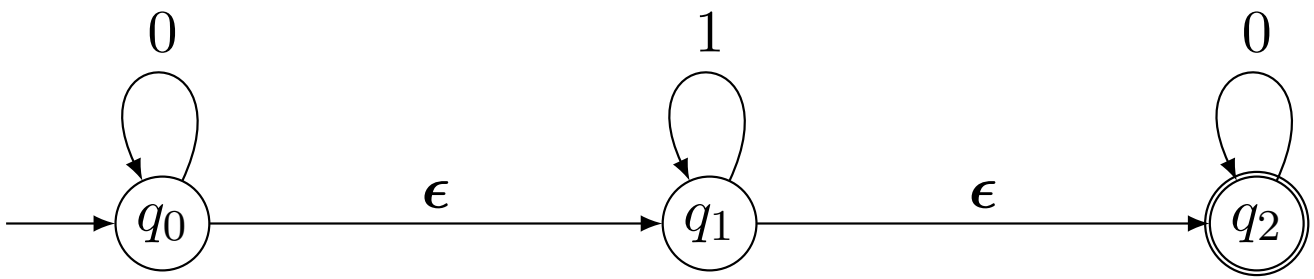
$$\exists i \geq 0. \ \hat{\delta}(\{q_0\}, \boldsymbol{\epsilon}^i) \cap F \neq \emptyset$$

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

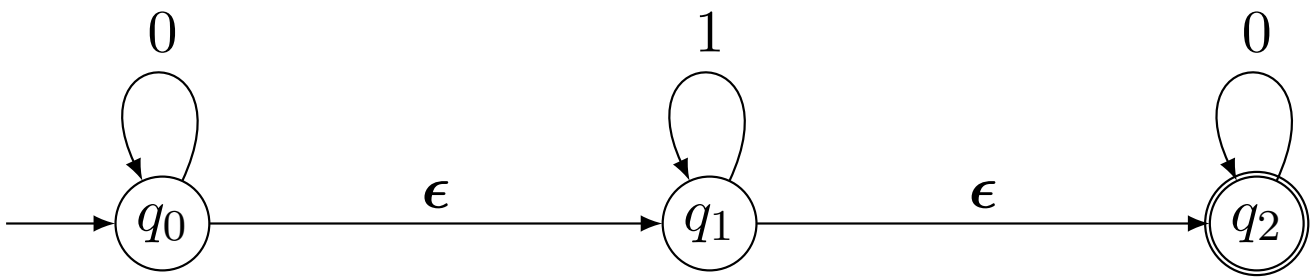
□

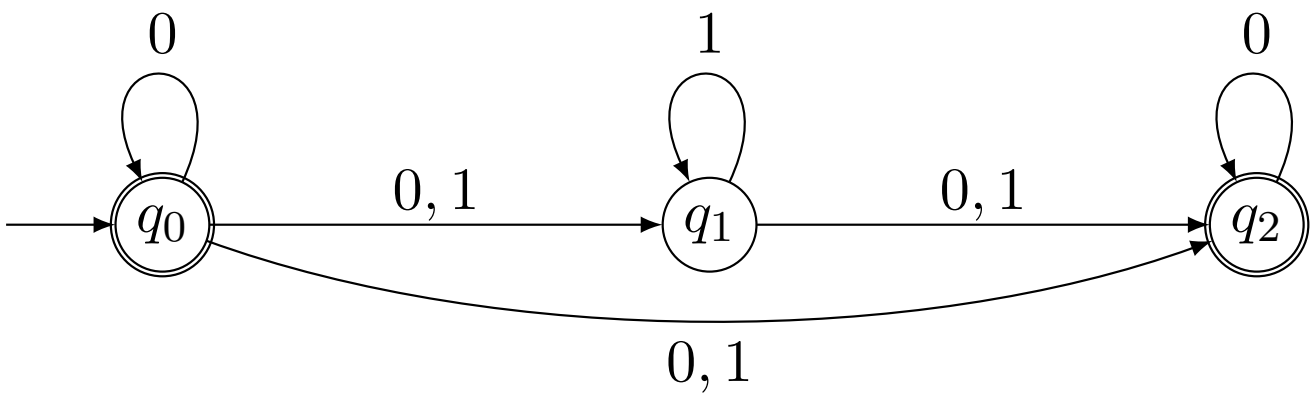
Ab jetzt: "ϵ-Übergang" und "ϵ-NFA".

Beispiel 3.18



Beispiel 3.18





Fazit: Die Automatentypen

- DFA
- NFA
- ϵ-NFA

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

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

3.5 Reguläre Ausdrücke

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

Werden oft in Kombination mit Grammatiken verwendet.

3.5 Reguläre Ausdrücke

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

Werden oft in Kombination mit Grammatiken verwendet.

Beispiel 3.19

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 3.20

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 3.20

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 3.21

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 3.22

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

- Alle Wörter, die mit 00 enden:

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

Beispiel 3.22

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 3.22

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 3.22

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 3.22

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 3.23

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

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

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

Erweiterte reguläre Ausdrücke in UNIX:

Erweiterte reguläre Ausdrücke in UNIX:

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

Strukturelle Induktion

Da die regulären Ausdrücke induktiv definiert sind, gilt für sie das Prinzip der strukturellen Induktion:

Um zu beweisen, dass Eigenschaft P für alle regulären Ausdrücke γ gilt, also $P(\gamma)$, beweise

- $P(\emptyset)$
- $P(\boldsymbol{\epsilon})$
- $P(a)$ für alle $a \in \Sigma$
- $P(\alpha) \wedge P(\beta) \Rightarrow P(\alpha\beta)$
- $P(\alpha) \wedge P(\beta) \Rightarrow P(\alpha \mid \beta)$
- $P(\alpha) \Rightarrow P(\alpha^*)$

Satz 3.24 (Kleene 1956)

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

Satz 3.24 (Kleene 1956)

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

Beweis:

"$\Longrightarrow$":

Sei $L = L(\gamma)$.

Wir konstruieren einen ϵ-NFA N mit $L = L(N)$ mit Hilfe struktureller Induktion über γ.

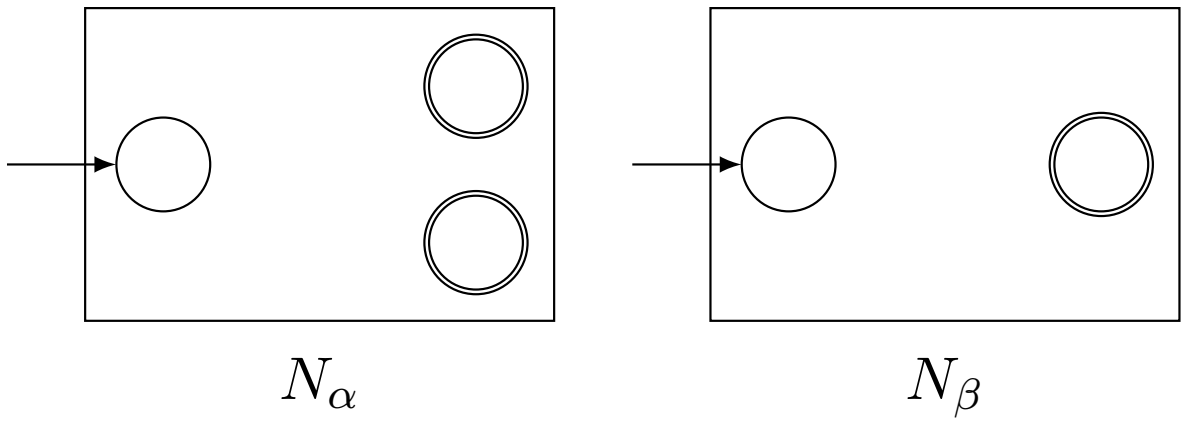
Die Basisfälle $\gamma = \emptyset$, $\gamma = \epsilon$, und $\gamma = a \in \Sigma$ sind offensichtlich.

Fall $\gamma = \alpha\beta$:

Nach Induktionsannahme können wir ϵ-NFAs N_α und N_β konstruieren mit $L(N_\alpha) = L(\alpha)$ und $L(N_\beta) = L(\beta)$.

Fall $\gamma = \alpha\beta$:

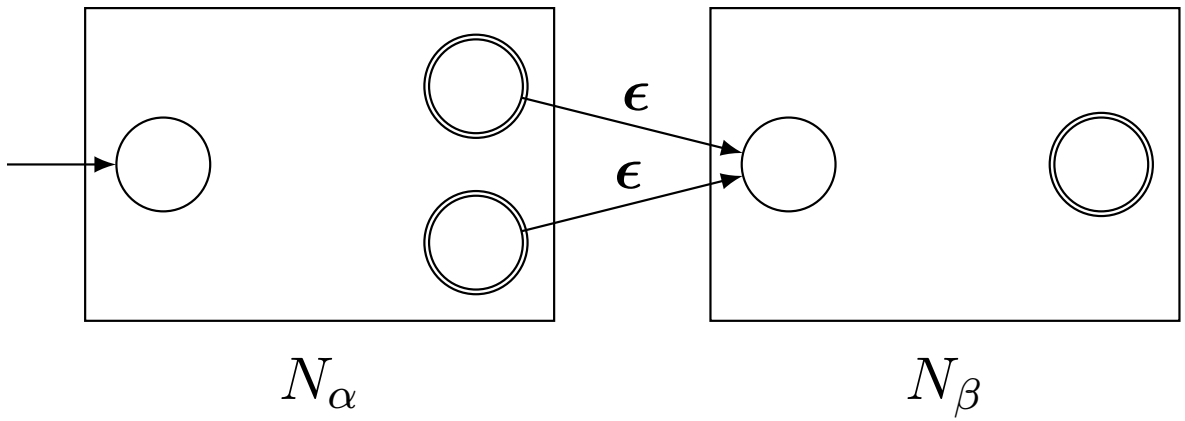
Nach Induktionsannahme können wir ϵ-NFAs N_α und N_β konstruieren mit $L(N_\alpha) = L(\alpha)$ und $L(N_\beta) = L(\beta)$.



N_α N_β

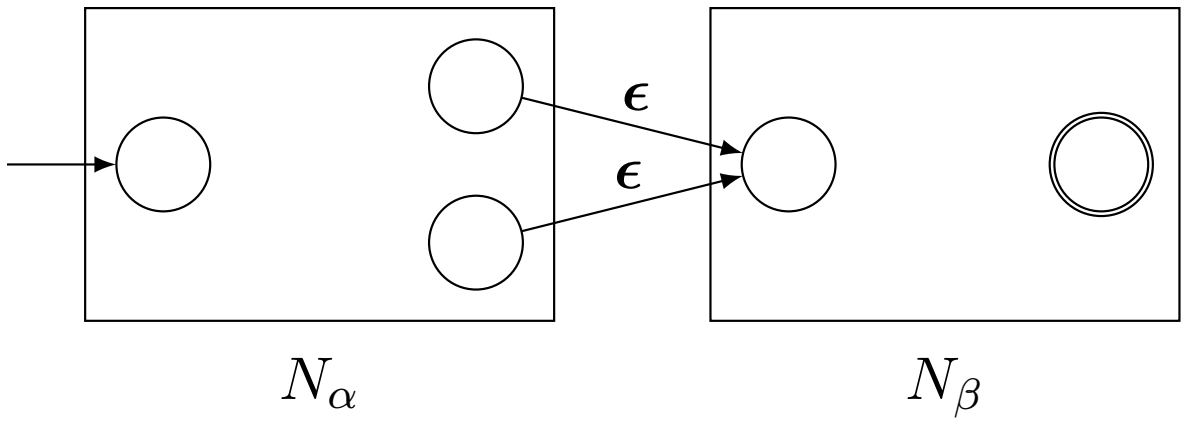
Fall $\gamma = \alpha\beta$:

Nach Induktionsannahme können wir ϵ-NFAs N_α und N_β konstruieren mit $L(N_\alpha) = L(\alpha)$ und $L(N_\beta) = L(\beta)$.



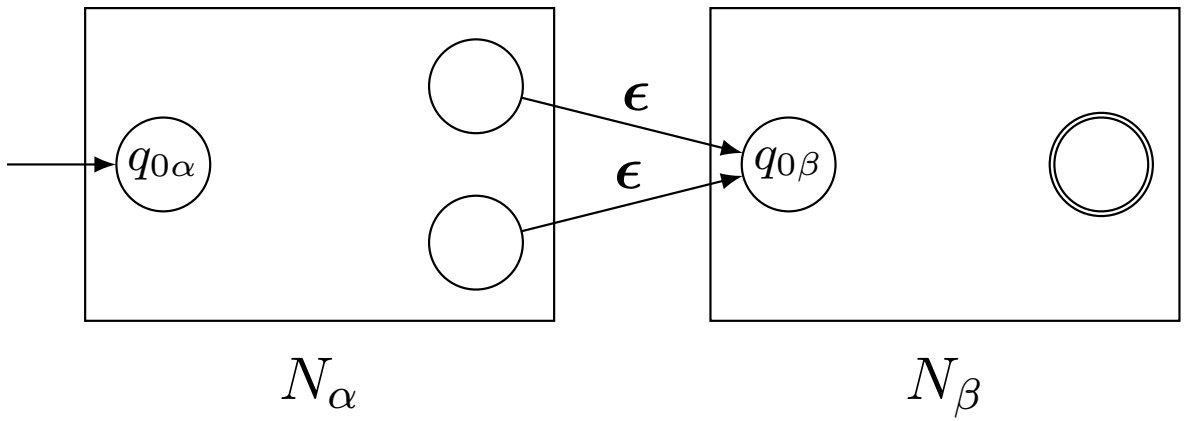
Fall $\gamma = \alpha\beta$:

Nach Induktionsannahme können wir ϵ-NFAs N_α und N_β konstruieren mit $L(N_\alpha) = L(\alpha)$ und $L(N_\beta) = L(\beta)$.



Fall $\gamma = \alpha\beta$:

Nach Induktionsannahme können wir ϵ-NFAs N_α und N_β konstruieren mit $L(N_\alpha) = L(\alpha)$ und $L(N_\beta) = L(\beta)$.

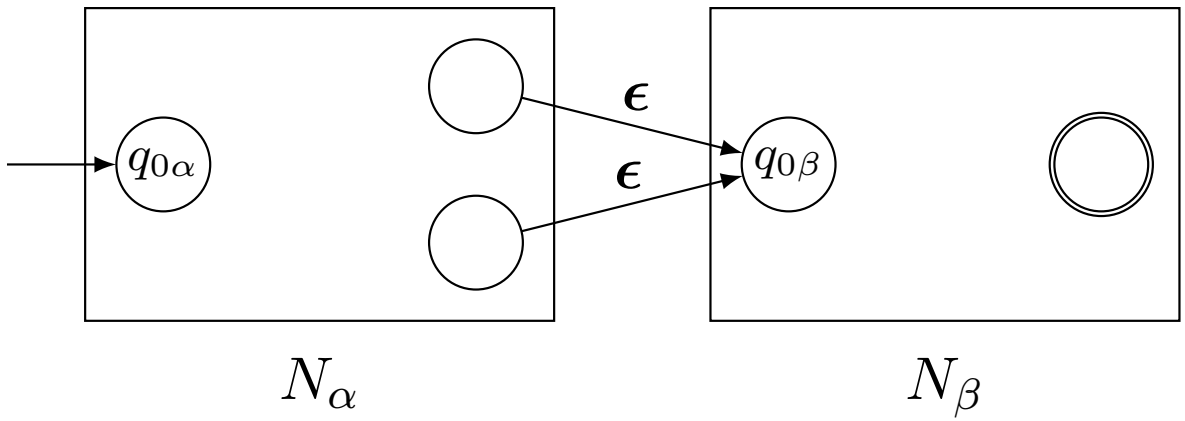


Formal:

$$\begin{aligned} N_\alpha &= (Q_\alpha, \Sigma, \delta_\alpha, q_{0\alpha}, F_\alpha) \\ N_\beta &= (Q_\beta, \Sigma, \delta_\beta, q_{0\beta}, F_\beta) \end{aligned}$$

Fall $\gamma = \alpha\beta$:

Nach Induktionsannahme können wir ϵ-NFAs N_α und N_β konstruieren mit $L(N_\alpha) = L(\alpha)$ und $L(N_\beta) = L(\beta)$.

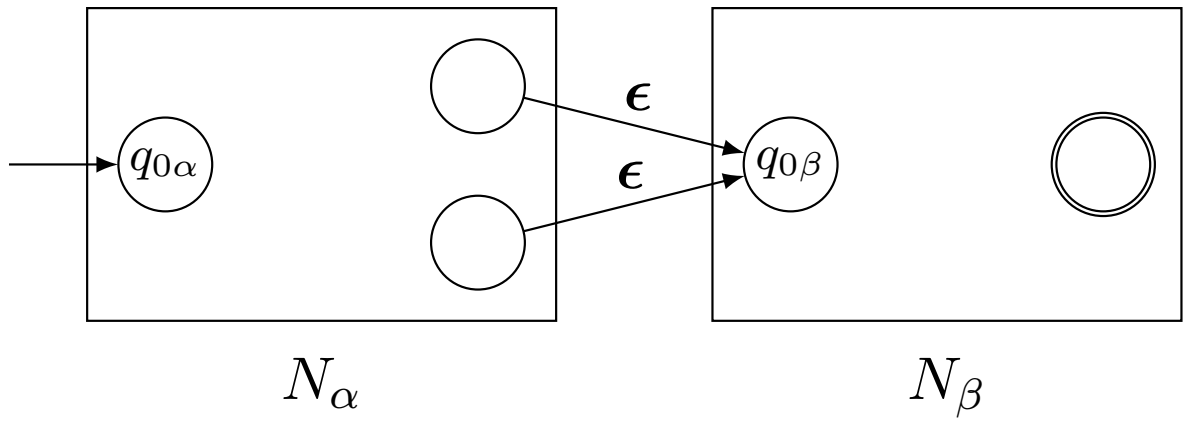


Formal:

$$\begin{aligned} N_\alpha &= (Q_\alpha, \Sigma, \delta_\alpha, q_{0\alpha}, F_\alpha) \\ N_\beta &= (Q_\beta, \Sigma, \delta_\beta, q_{0\beta}, F_\beta) \qquad \text{(mit } Q_\alpha \cap Q_\beta = \emptyset\text{)} \end{aligned}$$

Fall $\gamma = \alpha\beta$:

Nach Induktionsannahme können wir ϵ-NFAs N_α und N_β konstruieren mit $L(N_\alpha) = L(\alpha)$ und $L(N_\beta) = L(\beta)$.

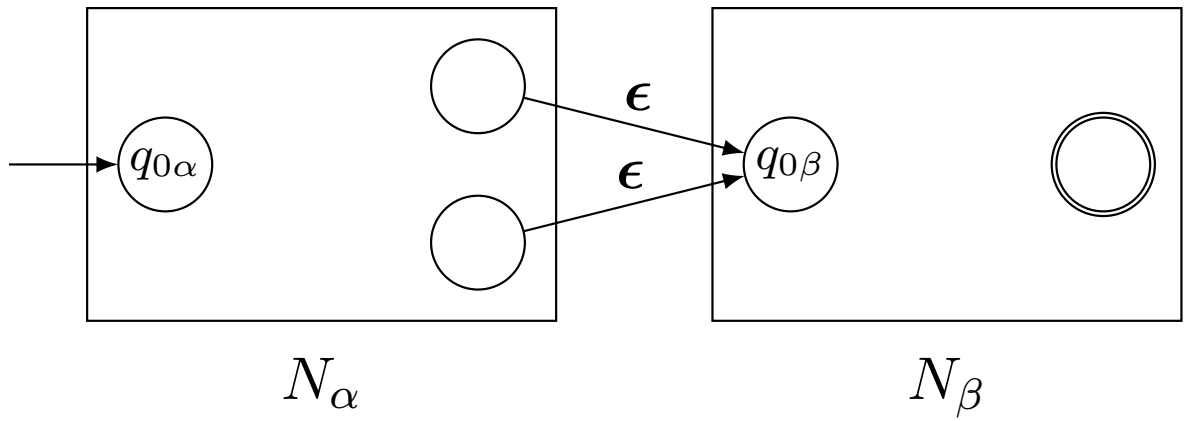


Formal:

$$
\begin{aligned}
N_\alpha &= (Q_\alpha, \Sigma, \delta_\alpha, q_{0\alpha}, F_\alpha) \\
N_\beta &= (Q_\beta, \Sigma, \delta_\beta, q_{0\beta}, F_\beta) \qquad \text{(mit } Q_\alpha \cap Q_\beta = \emptyset\text{)} \\
N_{\alpha\beta} &:= (Q_\alpha \cup Q_\beta, \Sigma, \delta, q_{0\alpha}, F_\beta)
\end{aligned}
$$

Fall $\gamma = \alpha\beta$:

Nach Induktionsannahme können wir ϵ-NFAs N_α und N_β konstruieren mit $L(N_\alpha) = L(\alpha)$ und $L(N_\beta) = L(\beta)$.



Formal:

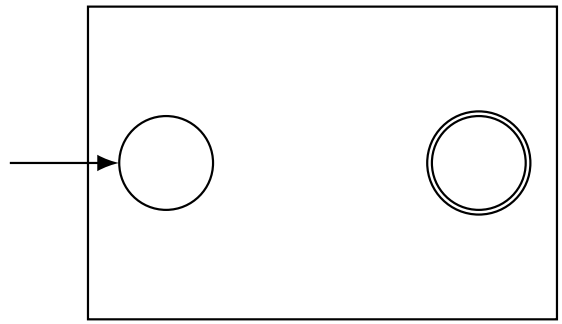
$$
\begin{aligned}
N_\alpha &= (Q_\alpha, \Sigma, \delta_\alpha, q_{0\alpha}, F_\alpha) \\
N_\beta &= (Q_\beta, \Sigma, \delta_\beta, q_{0\beta}, F_\beta) \qquad \text{(mit } Q_\alpha \cap Q_\beta = \emptyset\text{)} \\
N_{\alpha\beta} &:= (Q_\alpha \cup Q_\beta, \Sigma, \delta, q_{0\alpha}, F_\beta) \\
\delta &:= \delta_\alpha \cup \delta_\beta \cup \{(f, \boldsymbol{\epsilon}) \mapsto \{q_{0\beta}\} \mid f \in F_\alpha\}
\end{aligned}
$$

Fall $\gamma = \alpha \mid \beta$:

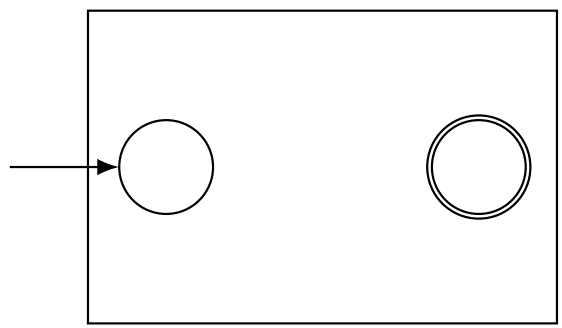
Nach Induktionsannahme können wir ϵ-NFAs N_α und N_β konstruieren mit $L(N_\alpha) = L(\alpha)$ und $L(N_\beta) = L(\beta)$.

Fall $\gamma = \alpha \mid \beta$:

Nach Induktionsannahme können wir ϵ-NFAs N_α und N_β konstruieren mit $L(N_\alpha) = L(\alpha)$ und $L(N_\beta) = L(\beta)$.



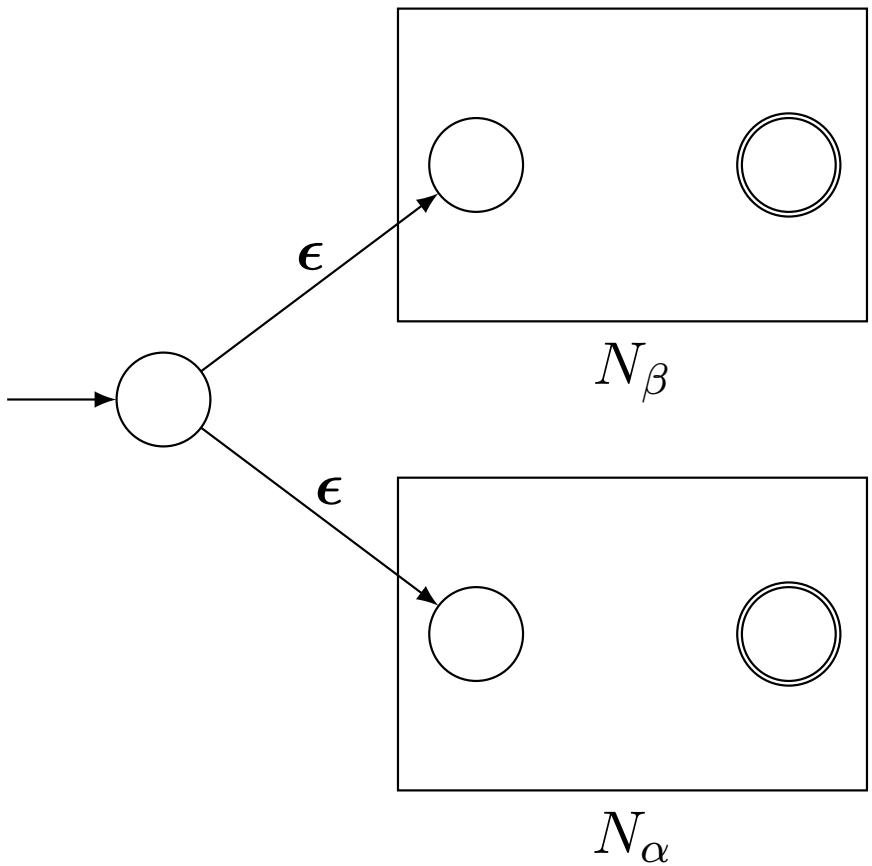
N_β



N_α

Fall $\gamma = \alpha \mid \beta$:

Nach Induktionsannahme können wir ϵ-NFAs N_α und N_β konstruieren mit $L(N_\alpha) = L(\alpha)$ und $L(N_\beta) = L(\beta)$.



Fall $\gamma = \alpha^*$:

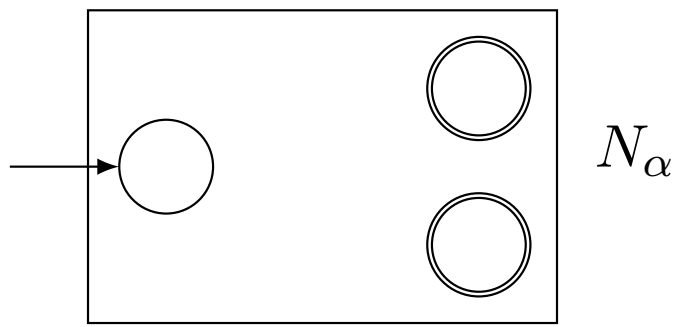
Nach Induktionsannahme können wir einen ϵ-NFA N_α konstruieren mit

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

Fall $\gamma = \alpha^*$:

Nach Induktionsannahme können wir einen ϵ-NFA N_α konstruieren mit

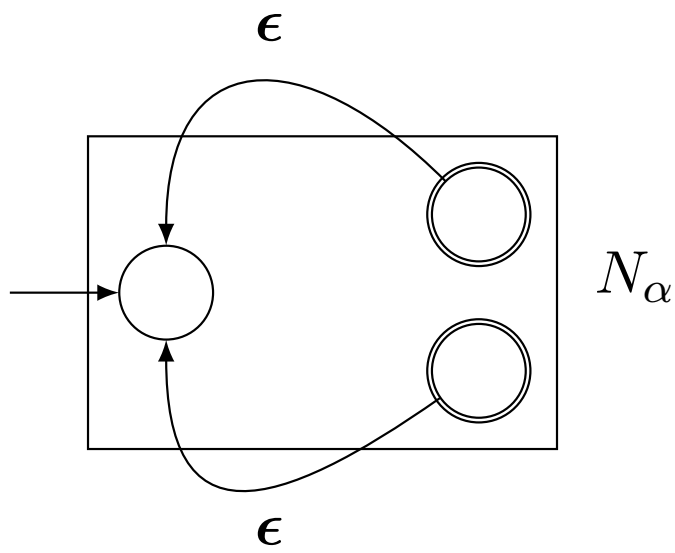
$$L(N_\alpha) = L(\alpha) \ .$$



Fall $\gamma = \alpha^*$:

Nach Induktionsannahme können wir einen ϵ-NFA N_α konstruieren mit

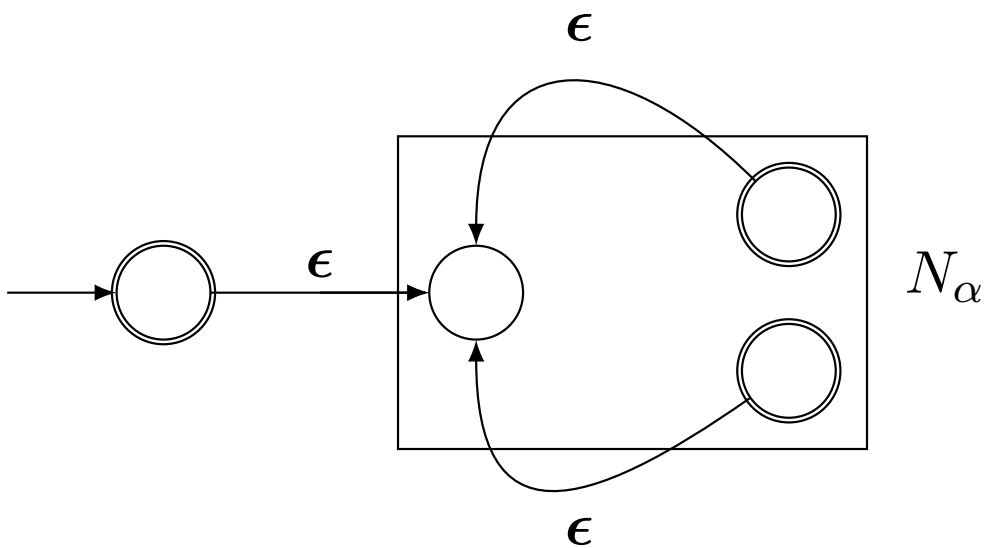
$$L(N_\alpha) = L(\alpha) \ .$$



Fall $\gamma = \alpha^*$:

Nach Induktionsannahme können wir einen ϵ-NFA N_α konstruieren mit

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



"$\Longleftarrow$":
Sei $M = (Q, \Sigma, \delta, q_1, F)$ ein DFA.
Wir konstruieren einen RE γ mit $L(M) = L(\gamma)$.

"$\Longleftarrow$":
Sei $M = (Q, \Sigma, \delta, q_1, F)$ ein DFA.
Wir konstruieren einen RE γ mit $L(M) = L(\gamma)$.

Sei $Q = \{q_1, \ldots, q_n\}$. Wir setzen

$$R_{ij}^k := \{w \in \Sigma^* \mid \text{die Eingabe } w \text{ führt von } q_i \text{ in } q_j, \text{ wobei alle Zwischenzustände (ohne ersten und letzten) einen Index } \leq k \text{ haben } \}$$

"$\Longleftarrow$":
Sei $M = (Q, \Sigma, \delta, q_1, F)$ ein DFA.
Wir konstruieren einen RE γ mit $L(M) = L(\gamma)$.

Sei $Q = \{q_1, \ldots, q_n\}$. Wir setzen

$$R_{ij}^k := \{w \in \Sigma^* \mid \text{die Eingabe } w \text{ führt von } q_i \text{ in } q_j, \text{ wobei alle Zwischenzustände (ohne ersten und letzten) einen Index } \leq k \text{ haben } \}$$

Behauptung: Für alle $i, j \in \{1, \ldots, n\}$ und $k \in \{0, \ldots, n\}$ können wir einen RE α_{ij}^k konstruieren mit $L(\alpha_{ij}^k) = R_{ij}^k$.

Bew.:

Induktion über k:
$k = 0$: Hier gilt

$$R_{ij}^0 = \begin{cases} \{a \in \Sigma \mid \delta(q_i, a) = q_j\}, & \text{falls } i \neq j \\ \{a \in \Sigma \mid \delta(q_i, a) = q_j\} \cup \{\epsilon\}, & \text{falls } i = j \end{cases}$$

Setze

$$\alpha_{ij}^0 := \begin{cases} a_1|\ldots|a_l & \text{falls } i \neq j \\ a_1|\ldots|a_l|\boldsymbol{\epsilon} & \text{falls } i = j \end{cases}$$

wobei $\{a_1 \ldots, a_l\} = \{a \in \Sigma \mid \delta(q_i, a) = q_j\}$.

Bew.:

Induktion über k:
$k \Rightarrow k+1$: Hier gilt

$$R_{ij}^{k+1} = R_{ij}^{k} \cup R_{i(k+1)}^{k} (R_{(k+1)(k+1)}^{k})^{*} R_{(k+1)j}^{k}$$

weil man jede Folge von Zahlen $\leq k+1$ schreiben kann als

$$F_1, k+1, F_2, k+1, \ldots, k+1, F_m$$

wobei jede F_l Folge von Zahlen $\leq k$ ist.

Bew.:

Induktion über k:
$k \Rightarrow k+1$: Hier gilt

$$R_{ij}^{k+1} = R_{ij}^{k} \cup R_{i(k+1)}^{k} (R_{(k+1)(k+1)}^{k})^* R_{(k+1)j}^{k}$$

weil man jede Folge von Zahlen $\leq k+1$ schreiben kann als

$$F_1, k+1, F_2, k+1, \ldots, k+1, F_m$$

wobei jede F_l Folge von Zahlen $\leq k$ ist.
Wir definieren rekursiv

$$\alpha_{ij}^{k+1} = \alpha_{ij}^{k} \mid \alpha_{i(k+1)}^{k} (\alpha_{(k+1)(k+1)}^{k})^* \alpha_{(k+1)j}^{k}$$

Bew.:

Induktion über k:
$k \Rightarrow k+1$: Hier gilt

$$R_{ij}^{k+1} = R_{ij}^{k} \cup R_{i(k+1)}^{k} (R_{(k+1)(k+1)}^{k})^* R_{(k+1)j}^{k}$$

weil man jede Folge von Zahlen $\leq k+1$ schreiben kann als

$$F_1, k+1, F_2, k+1, \ldots, k+1, F_m$$

wobei jede F_l Folge von Zahlen $\leq k$ ist.
Wir definieren rekursiv

$$\alpha_{ij}^{k+1} = \alpha_{ij}^{k} \mid \alpha_{i(k+1)}^{k} (\alpha_{(k+1)(k+1)}^{k})^* \alpha_{(k+1)j}^{k}$$

Somit gilt

$$L(M) = L(\alpha_{1i_1}^{n} \mid \cdots \mid \alpha_{1i_r}^{n})$$

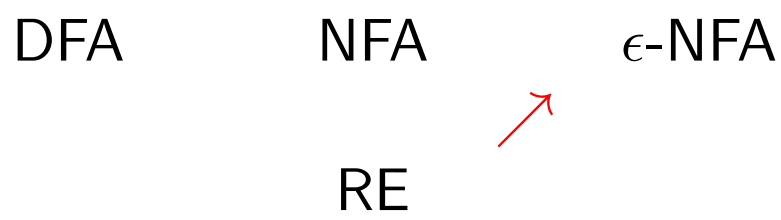
wobei $F = \{i_1, \ldots, i_r\}$. □

Unsere Konversionen auf einen Blick:

DFA NFA ϵ-NFA

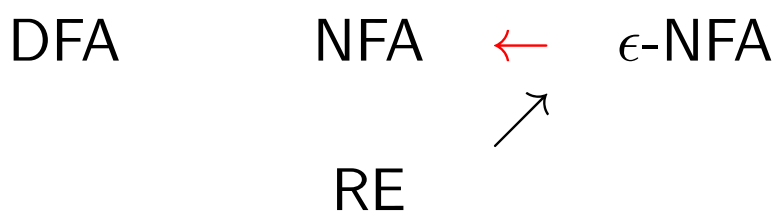
RE

Unsere Konversionen auf einen Blick:



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

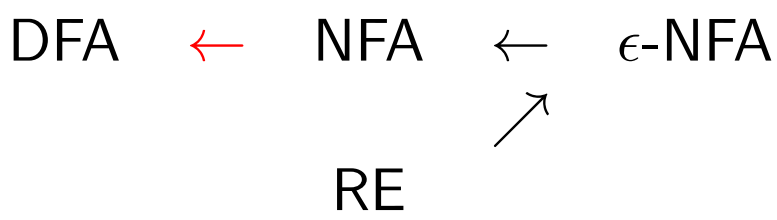
Unsere Konversionen auf einen Blick:



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

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

Unsere Konversionen auf einen Blick:

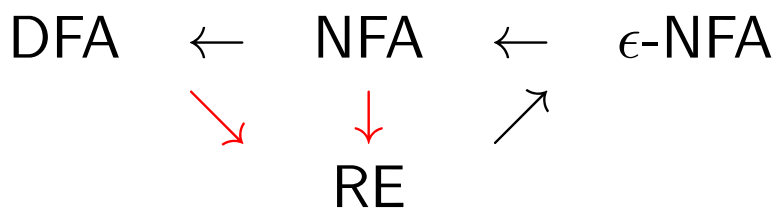


RE $\to$ ϵ-NFA: RE der Länge $n \rightsquigarrow O(n)$ Zustände

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

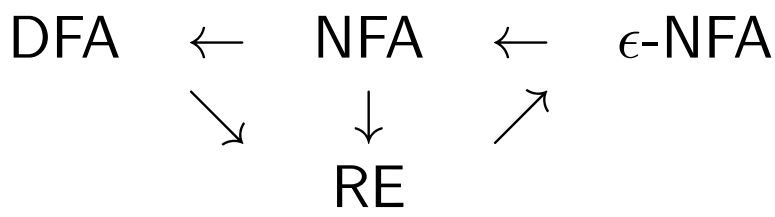
NFA $\to$ DFA: n Zustände $\rightsquigarrow O(2^n)$ Zustände

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
FA $\to$ RE:	n Zustände $\rightsquigarrow$ RE der Länge $O(n4^n)$

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
FA $\to$ RE:	n Zustände $\rightsquigarrow$ RE der Länge $O(n4^n)$

Beweis FA$\to$RE: $m_k :=$ maximale Länge von α_{ij}^k

- $m_0 = c_1|\Sigma|$
- $m_{k+1} = 4 * m_k + c_2$
- $m_k \in O(4^k)$
- Länge des Gesamtausdrucks: $O(n4^n)$

3.6 Abschlusseigenschaften regulärer Sprachen

Satz 3.25

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.

3.6 Abschlusseigenschaften regulärer Sprachen

Satz 3.25

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 3.26
Sind $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_i)$ *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 3.26
Sind $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_i)$ *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?

3.7 Rechnen mit regulären Ausdrücken

Definition 3.27

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

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

3.7 Rechnen mit regulären Ausdrücken

Definition 3.27
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 3.28

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

Lemma 3.29

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 3.30

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

Stern:

Lemma 3.30

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

Beispiel 3.31

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

Stern:

Lemma 3.30

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

Beispiel 3.31

Herleitung einer Äquivalenz aus obigen Lemmas:

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

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

Stern:

Lemma 3.30

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

Beispiel 3.31

Herleitung einer Äquivalenz aus obigen Lemmas:

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

Stern:

Lemma 3.30

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

Beispiel 3.31

Herleitung einer Äquivalenz aus obigen Lemmas:

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

Stern:

Lemma 3.30

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

Beispiel 3.31

Herleitung einer Äquivalenz aus obigen Lemmas:

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

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 3.32 (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 3.32 (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.

Für Typ 3 Sprachen können wir automatisch effiziente Recognizer synthetisieren.

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

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 von Beispiel 2.11 ist nicht rechtslinear.
- Das zeigt jedoch noch nicht das die Sprache der arithmethischen Ausdrücke nicht regulär ist: Es könnte eine andere rechtslineare Grammatik (vielleicht viel größer als die vom Beispiel 2.11), die die selbe Sprache erzeugt.

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?*

Satz 3.33 (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*
- $\forall i \geq 0.\ uv^i w \in R$.

- Die logische Struktur des Satzes:

  $\forall reg.\ R.$

- Die logische Struktur des Satzes:

  $\forall \mathit{reg}.\ R.\ \exists n > 0.$

- Die logische Struktur des Satzes:

  $\forall \textit{reg}.\ R.\ \exists n > 0.\ \forall z \in R.\ |z| \geq n \Rightarrow$

- Die logische Struktur des Satzes:

$$\forall \textit{reg}.\ R.\ \exists n > 0.\ \forall z \in R.\ |z| \geq n \Rightarrow\ \exists u\ v\ w.\ z = uvw \wedge \ldots$$

- Die logische Struktur des Satzes:

  $\forall \textit{reg}.\ R.\ \exists n > 0.\ \forall z \in R.\ |z| \geq n \Rightarrow\ \exists u\ v\ w.\ z = uvw \wedge \ldots$

- Das Pumping Lemma als Spiel:
  - Spieler I wählt eine reguläre Sprache R
  - Spieler II wählt ein $n > 0$
  - Spieler I wählt ein $z \in R$ mit $|z| \geq n$
  - Spieler II wählt eine Zerlegung uvw von z

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

- Die logische Struktur des Satzes:

  $\forall \textit{reg}.\ R.\ \exists n > 0.\ \forall z \in R.\ |z| \geq n \Rightarrow\ \exists u\ v\ w.\ z = uvw \wedge \ldots$

- Das Pumping Lemma als Spiel:
  - Spieler I wählt eine reguläre Sprache R
  - Spieler II wählt ein $n > 0$
  - Spieler I wählt ein $z \in R$ mit $|z| \geq n$
  - Spieler II wählt eine Zerlegung uvw von z

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

  Das Pumping Lemma sagt, dass Spieler II eine *Gewinnstrategie* hat: Wenn er richtig spielt, dann gewinnt er jede Partie.

- Die logische Struktur des Satzes:

  $\forall \textit{reg}.\ R.\ \exists n > 0.\ \forall z \in R.\ |z| \geq n \Rightarrow\ \exists u\ v\ w.\ z = uvw \wedge \ldots$

- Das Pumping Lemma als Spiel:
  - Spieler I wählt eine reguläre Sprache R
  - Spieler II wählt ein $n > 0$
  - Spieler I wählt ein $z \in R$ mit $|z| \geq n$
  - Spieler II wählt eine Zerlegung uvw von z

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

  Das Pumping Lemma sagt, dass Spieler II eine *Gewinnstrategie* hat: Wenn er richtig spielt, dann gewinnt er jede Partie.

- Sprechweise: n ist eine Pumping-Lemma-Zahl für R, falls alle $z \in R$ mit $|z| \geq n$ sich so wie im Pumping-Lemma zerlegen und aufpumpen lassen.

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

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

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

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

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

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

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

Wir teilen z wie folg 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

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

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

Wir teilen z wie folg 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

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

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

Wir teilen z wie folg 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?]

Fazit:

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

Fazit:

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

Pumping-Lemma-Zahl für $L(ab^*c)$?

Pumping-Lemma-Zahl für $\{aaaaaa\}$?

Fazit:

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

Pumping-Lemma-Zahl für $L(ab^*c)$?

Pumping-Lemma-Zahl für $\{aaaaaa\}$?

Ist $|Q_M| + 1$ auch eine Pumping-Lemma-Zahl für R?

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.34

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

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.34

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

Beweis:

Angenommen, L sei doch regulär.

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.34

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.

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.34

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 $z = a^n b^n \in L$.

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.34

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 $z = a^n b^n \in L$.
Dann ist z zerlegbar in uvw mit
$u, v \in \{a\}^*$ (weil $|uv| \leq n$) und $v \neq \epsilon$.

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.34

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 $z = a^n b^n \in L$.
Dann ist z zerlegbar in uvw mit
$u, v \in \{a\}^*$ (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"

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.34

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 $z = a^n b^n \in L$.
Dann ist z zerlegbar in uvw mit
$u, v \in \{a\}^*$ (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"

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.34

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 $z = a^n b^n \in L$.
Dann ist z zerlegbar in uvw mit
$u, v \in \{a\}^*$ (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"

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

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.35
$L = \{0^{m^2} \mid m \geq 0\}$ *ist nicht regulär.*

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.35

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

Beweis:

Angenommen, L sei doch regulär.

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.35

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

Beweis:

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

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.35

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

Beweis:

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

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

und $uv^l w \in L$ für alle $l \in \mathbb{N}$. D.h. insb. $|uv^2 w|$ ist Quadratzahl.

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.35
$L = \{0^{m^2} \mid m \geq 0\}$ *ist nicht regulär.*

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

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

und $uv^l w \in L$ für alle $l \in \mathbb{N}$. D.h. insb. $|uv^2w|$ ist Quadratzahl.
Dies führt zu einem Widerspruch:

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

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.35
$L = \{0^{m^2} \mid m \geq 0\}$ *ist nicht regulär.*

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

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

und $uv^l w \in L$ für alle $l \in \mathbb{N}$. D.h. insb. $|uv^2w|$ ist Quadratzahl.
Dies führt zu einem Widerspruch:

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

Denn zwischen n^2 und $(n+1)^2$ liegt keine Quadratzahl. □

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.36

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

Beweis:

Aufgabe. □

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.36

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

Beweis:

Aufgabe. □

Satz 3.37

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

Beispiel für die Anwendung des Pumping Lemmas:

Satz 3.36

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

Beweis:

Aufgabe. □

Satz 3.37

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 3.36. □

Bemerkung

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

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

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

3.9 Entscheidungsverfahren

Wir untersuchen Entscheidungsprobleme für reguläre Sprachen, d.h., Probleme der Gestalt

Eingabe: Ein oder mehrere Objekte, die reguläre Sprachen beschreiben (DFA, NFA, RE, Typ 3 Gram., ...)
Frage: Haben diese Objekte eine Eigenschaft X?

3.9 Entscheidungsverfahren

Wir untersuchen Entscheidungsprobleme für reguläre Sprachen, d.h., Probleme der Gestalt

Eingabe: Ein oder mehrere Objekte, die reguläre Sprachen beschreiben (DFA, NFA, RE, Typ 3 Gram., ...)
Frage: Haben diese Objekte eine Eigenschaft X?

Ein (Entscheidungs)Problem ist entscheidbar wenn es einen Algorithmus gibt, der bei jeder Eingabe in endlicher Zeit die richtige Antwort auf die Frage festellt.

3.9 Entscheidungsverfahren

Wir untersuchen Entscheidungsprobleme für reguläre Sprachen, d.h., Probleme der Gestalt

Eingabe: Ein oder mehrere Objekte, die reguläre Sprachen beschreiben (DFA, NFA, RE, Typ 3 Gram., ...)
Frage: Haben diese Objekte eine Eigenschaft X?

Ein (Entscheidungs)Problem ist entscheidbar wenn es einen Algorithmus gibt, der bei jeder Eingabe in endlicher Zeit die richtige Antwort auf die Frage festellt.

In der zweiten Hälfte der Vorlesung wird gezeit, dass nicht alle Entscheidungsprobleme für Grammatiken enstcheidbar sind!

3.9 Entscheidungsverfahren

Wir untersuchen Entscheidungsprobleme für reguläre Sprachen, d.h., Probleme der Gestalt

Eingabe: Ein oder mehrere Objekte, die reguläre Sprachen beschreiben (DFA, NFA, RE, Typ 3 Gram., ...)
Frage: Haben diese Objekte eine Eigenschaft X?

Ein (Entscheidungs)Problem ist entscheidbar wenn es einen Algorithmus gibt, der bei jeder Eingabe in endlicher Zeit die richtige Antwort auf die Frage festellt.

In der zweiten Hälfte der Vorlesung wird gezeit, dass nicht alle Entscheidungsprobleme für Grammatiken enstcheidbar sind!

Welche Entscheidungsprobleme sind für rechtslineare Grammatiken (oder DFA; NFA; RE ...) entscheidbar?

Wie hängt die Laufzeit des Algorithmus mit der Beschreibung zusammen?

Definition 3.38

Sei D ein DFA, NFA, RE, rechtslineare Grammatik ....

Definition 3.38

Sei D ein DFA, NFA, RE, rechtslineare Grammatik ....

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

Leerheitsproblem: Gegeben D, gilt $L(D) = \emptyset$?

Endlichkeitsproblem: Gegeben D, ist $L(D)$ endlich?

Äquivalenzproblem: Gegeben D_1, D_2, gilt $L(D_1) = L(D_2)$?

Lemma 3.39

Das Wortproblem ist für ein Wort w und DFA M in Zeit $O(|w| + |M|)$ entscheidbar.

Lemma 3.39

Das Wortproblem ist für ein Wort w und DFA M in Zeit $O(|w| + |M|)$ entscheidbar.

Lemma 3.40

Das Wortproblem ist für ein Wort w und NFA N in Zeit $O(|Q|^2|w| + |N|)$ entscheidbar.

Lemma 3.39

Das Wortproblem ist für ein Wort w und DFA M in Zeit $O(|w| + |M|)$ entscheidbar.

Lemma 3.40

Das Wortproblem ist für ein Wort w und NFA N in Zeit $O(|Q|^2|w| + |N|)$ entscheidbar.

Beweis:

Sei $Q = \{1, \ldots, s\}$, $q_0 = 1$ und $w = a_1 \ldots a_n$.

$S := \{1\}$
for $i := 1$ **to** n **do** $S := \bigcup_{j \in S} \delta(j, a_i)$
return $(S \cap F \neq \emptyset)$

□

Lemma 3.41

Das Leerheitsproblem ist für NFAs und DFAs entscheidbar (in Zeit $O(|Q|^2|\Sigma|)$ bzw $O(|Q||\Sigma|)$).

Lemma 3.41

Das Leerheitsproblem ist für NFAs und DFAs entscheidbar (in Zeit $O(|Q|^2|\Sigma|)$ bzw $O(|Q||\Sigma|)$).

Beweis:

$L(M) = \emptyset$ gdw kein Endzustand von q_0 erreichbar ist.
Dies ist eine einfache Suche in einem Graphen, die jede Kante maximal ein Mal benutzen muss.
Ein NFA hat $\leq |Q|^2|\Sigma|$ Kanten, ein DFA hat $\leq |Q||\Sigma|$ Kanten. □

Lemma 3.41

Das Leerheitsproblem ist für NFAs und DFAs entscheidbar (in Zeit $O(|Q|^2|\Sigma|)$ bzw $O(|Q||\Sigma|)$).

Beweis:

$L(M) = \emptyset$ gdw kein Endzustand von q_0 erreichbar ist.
Dies ist eine einfache Suche in einem Graphen, die jede Kante maximal ein Mal benutzen muss.
Ein NFA hat $\leq |Q|^2|\Sigma|$ Kanten, ein DFA hat $\leq |Q||\Sigma|$ Kanten. □

Ist Σ fix, z.B. ASCII, so wird daraus $O(|Q|^2)$ bzw $O(|Q|)$.

Lemma 3.42

Das Endlichkeitsproblem ist für DFAs oder NFAs entscheidbar.

Lemma 3.42

Das Endlichkeitsproblem ist für DFAs oder NFAs entscheidbar.

Beweis:

$|L(M)| = \infty$ gdw von q_0 aus eine nicht-leere Schleife erreichbar ist, von der aus F erreichbar ist.

Lemma 3.42

Das Endlichkeitsproblem ist für DFAs oder NFAs entscheidbar.

Beweis:

$|L(M)| = \infty$ gdw von q_0 aus eine nicht-leere Schleife erreichbar ist, von der aus F erreichbar ist.

$Reach(K) =$ $R := \emptyset$; $W := K$
while $W \neq \emptyset$ **do**
pick and remove some $p \in W$
if $p \notin R$ **then**
$R := R \cup \{p\}$; $W := W \cup \bigcup_{a \in \Sigma} \delta(p, a)$
return R

$Finite(Q, \Sigma, \delta, q_0, F) =$ $R := Reach(\{q_0\})$
$C := \{p \in R \mid p \in Reach(\bigcup_{a \in \Sigma} \delta(p, a))\}$
return $(Reach(C) \cap F = \emptyset)$

Lemma 3.43

Das Äquivalenzproblem ist für DFAs entscheidbar.

Beweis:

Folgt direkt aus

$$\begin{aligned} L_1 \subseteq L_2 &\Leftrightarrow L_1 \cap \overline{L_2} = \emptyset \\ L_1 = L_2 &\Leftrightarrow L_1 \subseteq L_2 \wedge L_2 \subseteq L_1 \end{aligned}$$

da für DFAs Komplement und Durchschnitt wieder endliche Automaten liefern und das Leerheitsproblem für endliche Automaten entscheidbar ist. □

Satz 3.44

Das Äquivalenzproblem für DFAs ist in Zeit $O(|Q_1||Q_2||\Sigma|)$ *entscheidbar.*

Satz 3.44

Das Äquivalenzproblem für DFAs ist in Zeit $O(|Q_1||Q_2||\Sigma|)$ entscheidbar.

Beweis:

Gegeben: DFAs M_1 mit m und M_2 mit n Zuständen.
Mit Hilfe der Produkt-Konstruktion für $\cap$ folgt:

	Anzahl der Zustände
$L(M_1) \cap \overline{L(M_2)}$	mn
$\overline{L(M_1)} \cap L(M_2)$	mn

□

Korollar 3.45

Das Äquivalenzproblem für NFAs ist in Zeit $O(2^{|Q_1|+|Q_2|})$ *entscheidbar (bei fixem* Σ*).*

Korollar 3.45

Das Äquivalenzproblem für NFAs ist in Zeit $O(2^{|Q_1|+|Q_2|})$ entscheidbar (bei fixem Σ).

Beweis:

2 NFAs mit m und n Zuständen $\rightsquigarrow$
2 DFAs mit 2^m und 2^n Zuständen $\rightsquigarrow$
Äquivalenztest in Zeit $O(2^m 2^n)$ □

Korollar 3.45

Das Äquivalenzproblem für NFAs ist in Zeit $O(2^{|Q_1|+|Q_2|})$ entscheidbar (bei fixem Σ).

Beweis:

2 NFAs mit m und n Zuständen $\rightsquigarrow$
2 DFAs mit 2^m und 2^n Zuständen $\rightsquigarrow$
Äquivalenztest in Zeit $O(2^m 2^n)$ □

Korollar 3.46

Das Äquivalenzproblem für reguläre Ausdrücke ist entscheidbar.

Fazit:

Die Kodierung der Eingabe (DFA, NFA, RE, ...) kann entscheidend für die Komplexität eines Problems sein.

3.10 Automaten und Gleichungssysteme

Nicht mehr *Beweisen* von Äquivalenzen

Gilt $XX^* \equiv X^*X$ für alle X?

3.10 Automaten und Gleichungssysteme

Nicht mehr *Beweisen* von Äquivalenzen

$$\text{Gilt } XX^* \equiv X^*X \text{ für alle } X?$$

sondern *Lösen*:

$$\text{Für welches } X \text{ gilt } X \equiv aX \mid b \text{ ?}$$

3.10 Automaten und Gleichungssysteme

Nicht mehr *Beweisen* von Äquivalenzen

$$\text{Gilt } XX^* \equiv X^*X \text{ für alle } X?$$

sondern *Lösen*:

$$\text{Für welches } X \text{ gilt } X \equiv aX \mid b \text{ ?}$$

Anwendung:

$$\text{Automat} \rightsquigarrow \text{Gleichungssystem} \rightsquigarrow \text{RE}$$

3.10 Automaten und Gleichungssysteme

Nicht mehr *Beweisen* von Äquivalenzen

$$\text{Gilt } XX^* \equiv X^*X \text{ für alle } X?$$

sondern *Lösen*:

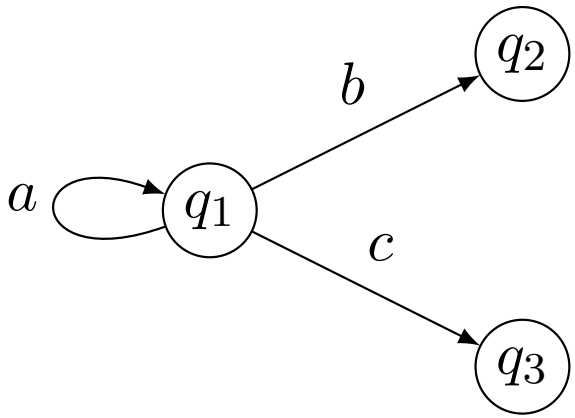
$$\text{Für welches } X \text{ gilt } X \equiv aX \mid b \text{ ?}$$

Anwendung:

$$\text{Automat} \rightsquigarrow \text{Gleichungssystem} \rightsquigarrow \text{RE}$$

Beispiel 3.47

Ein Automatenfragment:



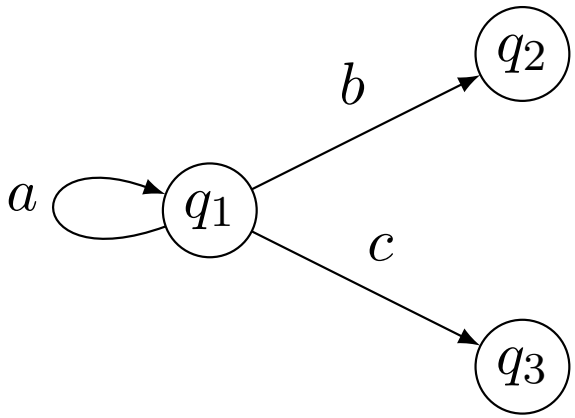
$$L_i := \{w \mid \hat{\delta}(q_i, w) \in F\}$$
$$\Longrightarrow$$
$$L_1 = \{a\}L_1 \cup \{b\}L_2 \cup \{c\}L_3$$

Da die L_i regulär sein müssen, arbeiten wir direkt mit REs:

$$X_1 \equiv aX_1 \mid bX_2 \mid cX_3$$

Beispiel 3.47

Ein Automatenfragment:



$$L_i := \{w \mid \hat{\delta}(q_i, w) \in F\}$$
$$\Longrightarrow$$
$$L_1 = \{a\}L_1 \cup \{b\}L_2 \cup \{c\}L_3$$

Da die L_i regulär sein müssen, arbeiten wir direkt mit REs:

$$X_1 \equiv aX_1 \mid bX_2 \mid cX_3$$

Lösung X_i ist RE für die von q_i aus akzeptierte Sprache.

Satz 3.48 (Ardens Lemma)

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

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

Satz 3.48 (Ardens Lemma)

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

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

Korollar 3.49

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

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

Satz 3.48 (Ardens Lemma)

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

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

Korollar 3.49

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

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

Bemerkungen

- $X = \{\epsilon\}X \cup B$ hat keine eindeutige Lösung: jede Sprache $X \supseteq B$ ist Lösung.
- $X \equiv aXb \mid \epsilon$ hat keine *reguläre* Lösung.

Satz 3.48 (Ardens Lemma)

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

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

Korollar 3.49

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

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

Bemerkungen

- $X = \{\epsilon\}X \cup B$ hat keine eindeutige Lösung: jede Sprache $X \supseteq B$ ist Lösung.
- $X \equiv aXb \mid \epsilon$ hat keine *reguläre* Lösung.

Satz 3.48 (Ardens Lemma)

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

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

Korollar 3.49

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

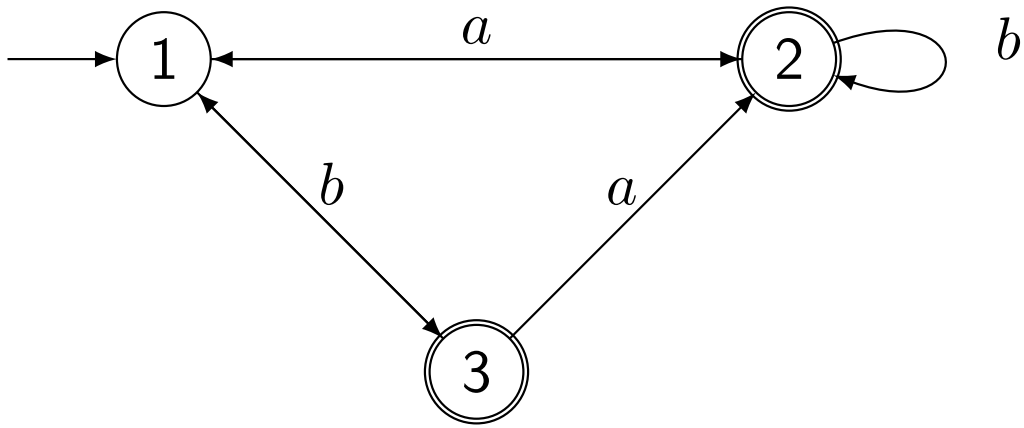
$$X \equiv \alpha X \mid \beta \quad \Longrightarrow \quad X \equiv \alpha^*\beta$$

Bemerkungen

- $X = \{\epsilon\}X \cup B$ hat keine eindeutige Lösung: jede Sprache $X \supseteq B$ ist Lösung.
- $X \equiv aXb \mid \boldsymbol{\epsilon}$ hat keine *reguläre* Lösung.

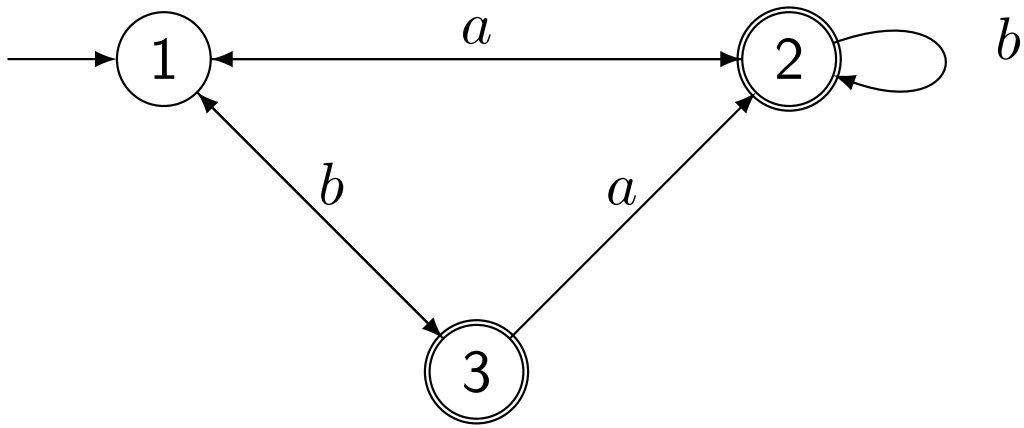
Umwandlung eines DFAs in einen äquivalenten regulären Ausdruck:

Beispiel 3.50



Umwandlung eines DFAs in einen äquivalenten regulären Ausdruck:

Beispiel 3.50



Äquivalentes Gleichungssystem:

X_i ist ein RE für die von q_i aus akzeptierte Sprache.

$$\begin{aligned} X_1 &\equiv aX_2 \mid bX_3 \\ X_2 &\equiv aX_1 \mid bX_2 \mid \boldsymbol{\epsilon} \\ X_3 &\equiv bX_1 \mid aX_2 \mid \boldsymbol{\epsilon} \end{aligned}$$

Lösen des Gleichungssystems:

$$\begin{aligned} X_1 &\equiv aX_2 \mid bX_3 \\ X_2 &\equiv aX_1 \mid bX_2 \mid \boldsymbol{\epsilon} \\ X_3 &\equiv bX_1 \mid aX_2 \mid \boldsymbol{\epsilon} \end{aligned}$$

Lösen des Gleichungssystems:

$$\begin{aligned} X_1 &\equiv aX_2 \mid bX_3 \\ X_2 &\equiv aX_1 \mid bX_2 \mid \boldsymbol{\epsilon} \\ X_3 &\equiv bX_1 \mid aX_2 \mid \boldsymbol{\epsilon} \end{aligned}$$

Löse $X_2 \equiv bX_2 \mid (aX_1 \mid \boldsymbol{\epsilon})$ nach X_2 auf:

$$X_2 \equiv b^*(aX_1 \mid \boldsymbol{\epsilon})$$

Lösen des Gleichungssystems:

$$\begin{aligned} X_1 &\equiv aX_2 \mid bX_3 \\ X_2 &\equiv aX_1 \mid bX_2 \mid \boldsymbol{\epsilon} \\ X_3 &\equiv bX_1 \mid aX_2 \mid \boldsymbol{\epsilon} \end{aligned}$$

Löse $X_2 \equiv bX_2 \mid (aX_1 \mid \boldsymbol{\epsilon})$ nach X_2 auf:

$$X_2 \equiv b^*(aX_1 \mid \boldsymbol{\epsilon})$$

Zurück einsetzen:

$$\begin{aligned} X_1 &\equiv a(b^*(aX_1 \mid \boldsymbol{\epsilon})) \mid bX_3 \\ X_3 &\equiv bX_1 \mid a(b^*(aX_1 \mid \boldsymbol{\epsilon})) \mid \boldsymbol{\epsilon} \end{aligned}$$

Lösen des Gleichungssystems:

$$\begin{aligned} X_1 &\equiv aX_2 \mid bX_3 \\ X_2 &\equiv aX_1 \mid bX_2 \mid \boldsymbol{\epsilon} \\ X_3 &\equiv bX_1 \mid aX_2 \mid \boldsymbol{\epsilon} \end{aligned}$$

Löse $X_2 \equiv bX_2 \mid (aX_1 \mid \boldsymbol{\epsilon})$ nach X_2 auf:

$$X_2 \equiv b^*(aX_1 \mid \boldsymbol{\epsilon})$$

Zurück einsetzen:

$$\begin{aligned} X_1 &\equiv a(b^*(aX_1 \mid \boldsymbol{\epsilon})) \mid bX_3 \\ X_3 &\equiv bX_1 \mid a(b^*(aX_1 \mid \boldsymbol{\epsilon})) \mid \boldsymbol{\epsilon} \end{aligned}$$

Ausmultiplizieren und X_i ausklammern:

$$\begin{aligned} X_1 &\equiv ab^*aX_1 \mid ab^* \mid bX_3 \\ X_3 &\equiv (b \mid ab^*a)X_1 \mid ab^* \mid \boldsymbol{\epsilon} \end{aligned}$$

$$
\begin{aligned}
X_1 &\equiv ab^*aX_1 \mid bX_3 \mid ab^* \\
X_3 &\equiv (b \mid ab^*a)X_1 \mid ab^* \mid \boldsymbol{\epsilon}
\end{aligned}
$$

$$
\begin{aligned}
X_1 &\equiv ab^*aX_1 \mid bX_3 \mid ab^* \\
X_3 &\equiv (b \mid ab^*a)X_1 \mid ab^* \mid \boldsymbol{\epsilon}
\end{aligned}
$$

X_3 ist gelöst, in 1. Gleichung einsetzen:

$$
X_1 \equiv ab^*aX_1 \mid b((b \mid ab^*a)X_1 \mid ab^* \mid \boldsymbol{\epsilon}) \mid ab^*
$$

$$
\begin{aligned}
X_1 &\equiv ab^*aX_1 \mid bX_3 \mid ab^* \\
X_3 &\equiv (b \mid ab^*a)X_1 \mid ab^* \mid \boldsymbol{\epsilon}
\end{aligned}
$$

X_3 ist gelöst, in 1. Gleichung einsetzen:

$$X_1 \equiv ab^*aX_1 \mid b((b \mid ab^*a)X_1 \mid ab^* \mid \boldsymbol{\epsilon}) \mid ab^*$$

Ausmultiplizieren und X_1 ausklammern:

$$X_1 \equiv (ab^*a \mid bb \mid bab^*a)X_1 \mid bab^* \mid b \mid ab^*$$

$$
\begin{aligned}
X_1 &\equiv ab^*aX_1 \mid bX_3 \mid ab^* \\
X_3 &\equiv (b \mid ab^*a)X_1 \mid ab^* \mid \boldsymbol{\epsilon}
\end{aligned}
$$

X_3 ist gelöst, in 1. Gleichung einsetzen:

$$X_1 \equiv ab^*aX_1 \mid b((b \mid ab^*a)X_1 \mid ab^* \mid \boldsymbol{\epsilon}) \mid ab^*$$

Ausmultiplizieren und X_1 ausklammern:

$$X_1 \equiv (ab^*a \mid bb \mid bab^*a)X_1 \mid bab^* \mid b \mid ab^*$$

Nach X_1 auflösen:

$$X_1 \equiv (ab^*a \mid bb \mid bab^*a)^*(bab^* \mid b \mid ab^*)$$

Umwandlung eines FAs in einen äquivalenten regulären Ausdruck:

Umwandlung eines FAs in einen äquivalenten regulären Ausdruck:

- Wandle FA mit n Zuständen in ein System von n Gleichungen mit n Variablen um:

$$X_i \equiv a_{i1}X_1 \mid \cdots \mid a_{in}X_n \mid b_i$$

Umwandlung eines FAs in einen äquivalenten regulären Ausdruck:

- Wandle FA mit n Zuständen in ein System von n Gleichungen mit n Variablen um:

$$X_i \equiv a_{i1}X_1 \mid \cdots \mid a_{in}X_n \mid b_i$$

$$a_{ij} := c_1 \mid \cdots \mid c_k \quad \text{falls } \{c_1, \ldots, c_k\} = \{c \in \Sigma \mid q_i \xrightarrow{c} q_j\}$$

wobei $a_{ij} := \emptyset$ falls $q_i \xrightarrow{c} q_j$ für kein $c \in \Sigma$

Umwandlung eines FAs in einen äquivalenten regulären Ausdruck:

- Wandle FA mit n Zuständen in ein System von n Gleichungen mit n Variablen um:

$$X_i \equiv a_{i1}X_1 \mid \cdots \mid a_{in}X_n \mid b_i$$

$$a_{ij} := c_1 \mid \cdots \mid c_k \quad \text{falls } \{c_1, \ldots, c_k\} = \{c \in \Sigma \mid q_i \xrightarrow{c} q_j\}$$

wobei $a_{ij} := \emptyset$ falls $q_i \xrightarrow{c} q_j$ für kein $c \in \Sigma$

$$b_i := \begin{cases} \epsilon & \text{falls } q_i \in F \\ \emptyset & \text{sonst} \end{cases}$$

Umwandlung eines FAs in einen äquivalenten regulären Ausdruck:

- Wandle FA mit n Zuständen in ein System von n Gleichungen mit n Variablen um:

$$X_i \equiv a_{i1}X_1 \mid \cdots \mid a_{in}X_n \mid b_i$$

$$a_{ij} := c_1 \mid \cdots \mid c_k \quad \text{falls } \{c_1, \ldots, c_k\} = \{c \in \Sigma \mid q_i \xrightarrow{c} q_j\}$$
$$\text{wobei } a_{ij} := \emptyset \text{ falls } q_i \xrightarrow{c} q_j \text{für kein } c \in \Sigma$$

$$b_i := \begin{cases} \epsilon & \text{falls } q_i \in F \\ \emptyset & \text{sonst} \end{cases}$$

- Löse das System durch schrittweise Elimination von Variablen mit Hilfe von Ardens Lemma für REs (Korollar 3.49).

Umwandlung eines FAs in einen äquivalenten regulären Ausdruck:

- Wandle FA mit n Zuständen in ein System von n Gleichungen mit n Variablen um:

$$X_i \equiv a_{i1}X_1 \mid \cdots \mid a_{in}X_n \mid b_i$$

$$a_{ij} := c_1 \mid \cdots \mid c_k \quad \text{falls } \{c_1, \ldots, c_k\} = \{c \in \Sigma \mid q_i \xrightarrow{c} q_j\}$$
$$\text{wobei } a_{ij} := \emptyset \text{ falls } q_i \xrightarrow{c} q_j \text{für kein } c \in \Sigma$$

$$b_i := \begin{cases} \epsilon & \text{falls } q_i \in F \\ \emptyset & \text{sonst} \end{cases}$$

- Löse das System durch schrittweise Elimination von Variablen mit Hilfe von Ardens Lemma für REs (Korollar 3.49).
- Ist k der Startzustand, so beschreibt X_k die vom Automaten akzeptierte Sprache.

Beweis von Ardens Lemma:
Wir nehmen an $X = AX \cup B$.

$$X \quad = \quad A(AX \cup B) \cup B = A^2X \cup AB \cup B$$

Beweis von Ardens Lemma:
Wir nehmen an $X = AX \cup B$.

$$\begin{aligned} X &= A(AX \cup B) \cup B = A^2X \cup AB \cup B \\ &= A^2(AX \cup B) \cup AB \cup B = A^3X \cup A^2B \cup AB \cup B = \ldots \end{aligned}$$

Beweis von Ardens Lemma:
Wir nehmen an $X = AX \cup B$.

$$\begin{aligned} X &= A(AX \cup B) \cup B = A^2X \cup AB \cup B \\ &= A^2(AX \cup B) \cup AB \cup B = A^3X \cup A^2B \cup AB \cup B = \ldots \end{aligned}$$

Mit Induktion zeigt man für alle $n \in \mathbb{N}$:

$$X = A^{n+1}X \cup \bigcup_{i \leq n} A^i B \qquad (1)$$

Beweis von Ardens Lemma:
Wir nehmen an $X = AX \cup B$.

$$\begin{aligned} X &= A(AX \cup B) \cup B = A^2X \cup AB \cup B \\ &= A^2(AX \cup B) \cup AB \cup B = A^3X \cup A^2B \cup AB \cup B = \ldots \end{aligned}$$

Mit Induktion zeigt man für alle $n \in \mathbb{N}$:

$$X = A^{n+1}X \cup \bigcup_{i \leq n} A^iB \qquad (1)$$

0: Behauptung wird zu $X = AX \cup B$, der Annahme.

Beweis von Ardens Lemma:
Wir nehmen an $X = AX \cup B$.

$$\begin{aligned} X &= A(AX \cup B) \cup B = A^2X \cup AB \cup B \\ &= A^2(AX \cup B) \cup AB \cup B = A^3X \cup A^2B \cup AB \cup B = \ldots \end{aligned}$$

Mit Induktion zeigt man für alle $n \in \mathbb{N}$:

$$X = A^{n+1}X \cup \bigcup_{i \leq n} A^i B \qquad (1)$$

0: Behauptung wird zu $X = AX \cup B$, der Annahme.

$n+1$:
$$\begin{aligned} X &= A^{n+1}X \cup \textstyle\bigcup_{i \leq n} A^i B \\ &= A^{n+1}(AX \cup B) \cup \textstyle\bigcup_{i \leq n} A^{i+1} B \\ &= A^{n+2}X \cup A^{n+1}B \cup \textstyle\bigcup_{i \leq n} A^i B \\ &= A^{n+2}X \cup \textstyle\bigcup_{i \leq n+1} A^i B \\ &= A^{n+1}X \cup \textstyle\bigcup_{i \leq n} A^i B \end{aligned}$$

Wir zeigen nun $X = A^*B$.

Wir zeigen nun $X = A^*B$.

$A^*B \subseteq X$: $\quad w \in A^*B = \bigcup_{i \in \mathbb{N}} A^i B$

$$\implies \exists n.\ w \in A^n B$$

$$\implies w \in X$$

Wir zeigen nun $X = A^*B$.

$A^*B \subseteq X$: $\quad w \in A^*B = \bigcup_{i \in \mathbb{N}} A^i B$

$$\implies \exists n.\ w \in A^n B$$

$$\implies w \in X$$

$X \subseteq A^*B$:

Wir zeigen nun $X = A^*B$.

$A^*B \subseteq X$: $\quad w \in A^*B = \bigcup_{i \in \mathbb{N}} A^i B$

$$\Longrightarrow \exists n.\ w \in A^n B$$

$$\Longrightarrow w \in X$$

$X \subseteq A^*B$: Sei $w \in X$ und $n := |w|$.

$$\epsilon \notin A$$

$$\Longrightarrow \forall u \in A^{n+1}.\ |u| \geq n+1$$

$$\Longrightarrow w \notin A^{n+1}X$$

$$\Longrightarrow w \in \bigcup_{i \leq n} A^i B \qquad \text{(wegen (1))}$$

$$\Longrightarrow w \in \bigcup_{i \in \mathbb{N}} A^i B = A^*B$$

□

3.11 Minimierung endlicher Automaten

Wir zeigen, dass jede reguläre Sprache einen einzigen minimalen Recognizer hat und geben Algorithmen an, die diesen Recognizer konstruieren.

1. Beispiele
2. Algorithmen
3. Minimalitätsbeweis

Algorithmus zur Minimierung eines DFA

1. Entferne alle von q_0 aus nicht erreichbaren Zustände.
2. Berechne die äquivalenten Zustände des Automaten.
3. Kollabiere den Automaten durch Zusammenfassung aller äquivalenten Zustände.

Algorithmus zur Minimierung eines DFA

1. Entferne alle von q_0 aus nicht erreichbaren Zustände.
2. Berechne die äquivalenten Zustände des Automaten.
3. Kollabiere den Automaten durch Zusammenfassung aller äquivalenten Zustände.

Zustände p und q sind unterscheidbar wenn es $w \in \Sigma^*$ gibt mit $\hat{\delta}(p, w) \in F$ und $\hat{\delta}(q, w) \notin F$ oder umgekehrt.
Zustände sind äquivalent wenn sie nicht unterscheidbar sind, d.h. wenn für alle $w \in \Sigma^*$ gilt:

$$\hat{\delta}(p, w) \in F \quad \Leftrightarrow \quad \hat{\delta}(q, w) \in F$$

Algorithmus zur Minimierung eines DFA

1. Entferne alle von q_0 aus nicht erreichbaren Zustände.
2. Berechne die äquivalenten Zustände des Automaten.
3. Kollabiere den Automaten durch Zusammenfassung aller äquivalenten Zustände.

Zustände p und q sind unterscheidbar wenn es $w \in \Sigma^*$ gibt mit $\hat{\delta}(p, w) \in F$ und $\hat{\delta}(q, w) \notin F$ oder umgekehrt.
Zustände sind äquivalent wenn sie nicht unterscheidbar sind, d.h. wenn für alle $w \in \Sigma^*$ gilt:

$$\hat{\delta}(p, w) \in F \quad \Leftrightarrow \quad \hat{\delta}(q, w) \in F$$

- Gilt $p \in F$ und $q \notin F$, dann sind p und q unterscheidbar.
- Sind $\delta(p, a)$ und $\delta(q, a)$ unterscheidbar, dann auch p und q.

  $\Rightarrow$ Unterscheidbarkeit pflanzt sich rückwärts fort.

Berechnung äquivalenter Zustände eines DFA

Berechnung äquivalenter Zustände eines DFA

Eingabe: DFA $A = (Q, \Sigma, \delta, q_0, F)$

Ausgabe: Äquivalenzrelation auf Q.

Berechnung äquivalenter Zustände eines DFA

Eingabe: DFA $A = (Q, \Sigma, \delta, q_0, F)$

Ausgabe: Äquivalenzrelation auf Q.

Datenstruktur: Eine Menge U *ungeordneter* Paare $\{p, q\} \subseteq Q$.

Berechnung äquivalenter Zustände eines DFA

Eingabe: DFA $A = (Q, \Sigma, \delta, q_0, F)$

Ausgabe: Äquivalenzrelation auf Q.

Datenstruktur: Eine Menge U *ungeordneter* Paare $\{p, q\} \subseteq Q$.

Algorithmus U:

1. $U := \{\{p, q\} \mid p \in F \wedge q \notin F\}$
2. **while** $\exists \{p, q\} \notin U.\ \exists a \in \Sigma.\ \{\delta(p, a), \delta(q, a)\} \in U$
   do $U := U \cup \{\{p, q\}\}$

Berechnung äquivalenter Zustände eines DFA

Eingabe: DFA $A = (Q, \Sigma, \delta, q_0, F)$

Ausgabe: Äquivalenzrelation auf Q.

Datenstruktur: Eine Menge U *ungeordneter* Paare $\{p, q\} \subseteq Q$.

Algorithmus U:

1. $U := \{\{p, q\} \mid p \in F \wedge q \notin F\}$
2. **while** $\exists \{p, q\} \notin U.\ \exists a \in \Sigma.\ \{\delta(p, a), \delta(q, a)\} \in U$ **do** $U := U \cup \{\{p, q\}\}$

Invariante: $\{p, q\} \in U \implies p$ und q unterscheidbar

Berechnung äquivalenter Zustände eines DFA

Eingabe: DFA $A = (Q, \Sigma, \delta, q_0, F)$

Ausgabe: Äquivalenzrelation auf Q.

Datenstruktur: Eine Menge U *ungeordneter* Paare $\{p, q\} \subseteq Q$.

Algorithmus U:

1. $U := \{\{p, q\} \mid p \in F \wedge q \notin F\}$
2. **while** $\exists \{p, q\} \notin U.\ \exists a \in \Sigma.\ \{\delta(p, a), \delta(q, a)\} \in U$
   do $U := U \cup \{\{p, q\}\}$

Invariante: $\{p, q\} \in U \implies p$ und q unterscheidbar

Lemma 3.51

Am Ende gilt: U ist Menge aller unterscheidbaren Zustandspaare.

Berechnung äquivalenter Zustände eines DFA

Eingabe: DFA $A = (Q, \Sigma, \delta, q_0, F)$

Ausgabe: Äquivalenzrelation auf Q.

Datenstruktur: Eine Menge U *ungeordneter* Paare $\{p, q\} \subseteq Q$.

Algorithmus U:

1. $U := \{\{p, q\} \mid p \in F \wedge q \notin F\}$
2. **while** $\exists \{p, q\} \notin U.\ \exists a \in \Sigma.\ \{\delta(p, a), \delta(q, a)\} \in U$
   do $U := U \cup \{\{p, q\}\}$

Invariante: $\{p, q\} \in U \implies p$ und q unterscheidbar

Lemma 3.51

Am Ende gilt: U ist Menge aller unterscheidbaren Zustandspaare.

Beweis:

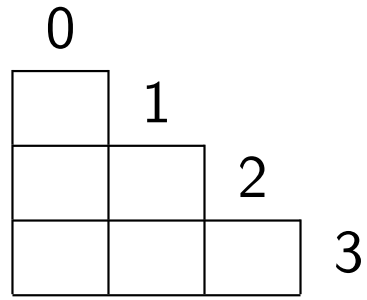
$\{p, q\} \in U \implies p$ und q unterscheidbar:
Invariante p und q unterscheidbar $\implies \{p, q\} \in U$:
Induktion über die Länge eines unterscheidenden Worts. □

Implementierung von U:

Tabelle von anfangs unmarkierten Paaren $\{p, q\}$, $p \neq q$.

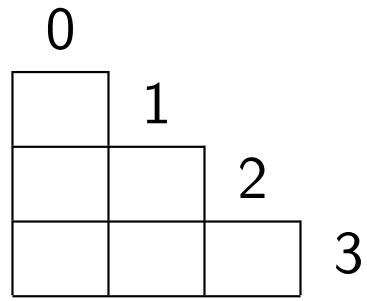
Implementierung von U:

Tabelle von anfangs unmarkierten Paaren $\{p, q\}$, $p \neq q$.



Implementierung von U:

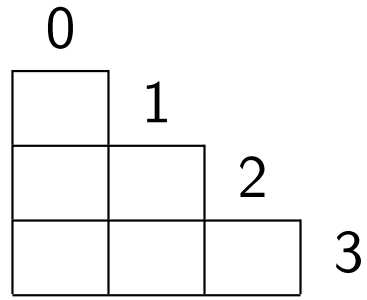
Tabelle von anfangs unmarkierten Paaren $\{p, q\}$, $p \neq q$.



for all $p \in F$, $q \in Q \setminus F$ **do** markiere $\{p, q\}$
while $\exists$ unmarkiertes $\{p, q\}$

Implementierung von U:

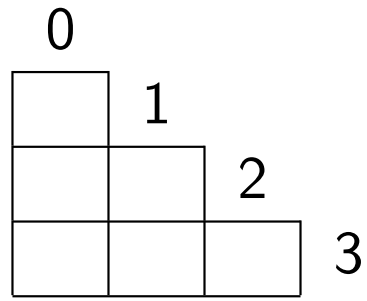
Tabelle von anfangs unmarkierten Paaren $\{p, q\}$, $p \neq q$.



for all $p \in F$, $q \in Q \setminus F$ **do** markiere $\{p, q\}$
while $\exists$ unmarkiertes $\{p, q\}$ $\exists a \in \Sigma$. $\{\delta(p, a), \delta(q, a)\}$ ist markiert
do markiere $\{p, q\}$

Implementierung von U:

Tabelle von anfangs unmarkierten Paaren $\{p, q\}$, $p \neq q$.



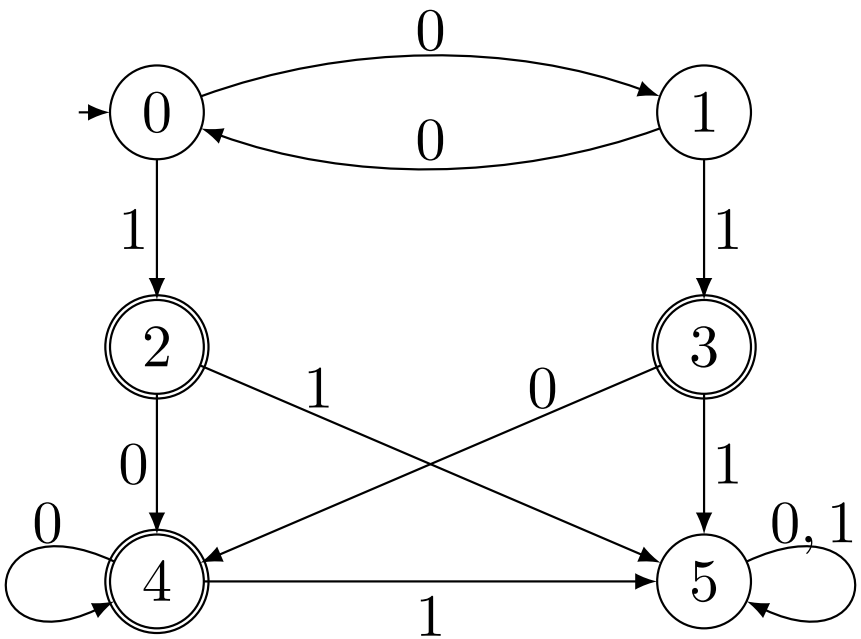
for all $p \in F$, $q \in Q \setminus F$ **do** markiere $\{p, q\}$
while $\exists$ unmarkiertes $\{p, q\}$ $\exists a \in \Sigma$. $\{\delta(p, a), \delta(q, a)\}$ ist markiert
do markiere $\{p, q\}$

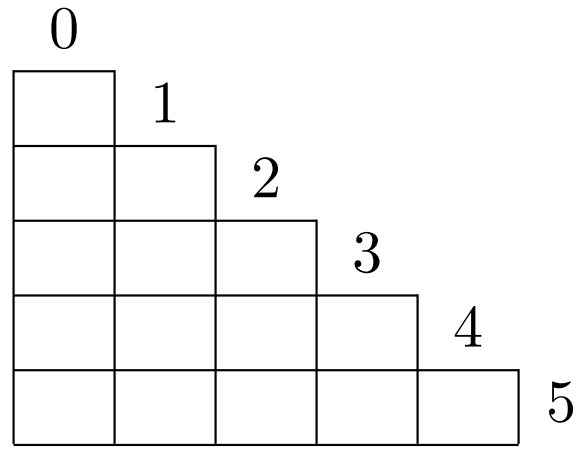
Komplexität:

$$O\left(\binom{n}{2} + \binom{n}{2}\binom{n}{2}|\Sigma|\right) = O\left(\frac{n(n-1)}{2} + \frac{n^2(n-1)^2}{4}|\Sigma|\right) = O(n^4)$$

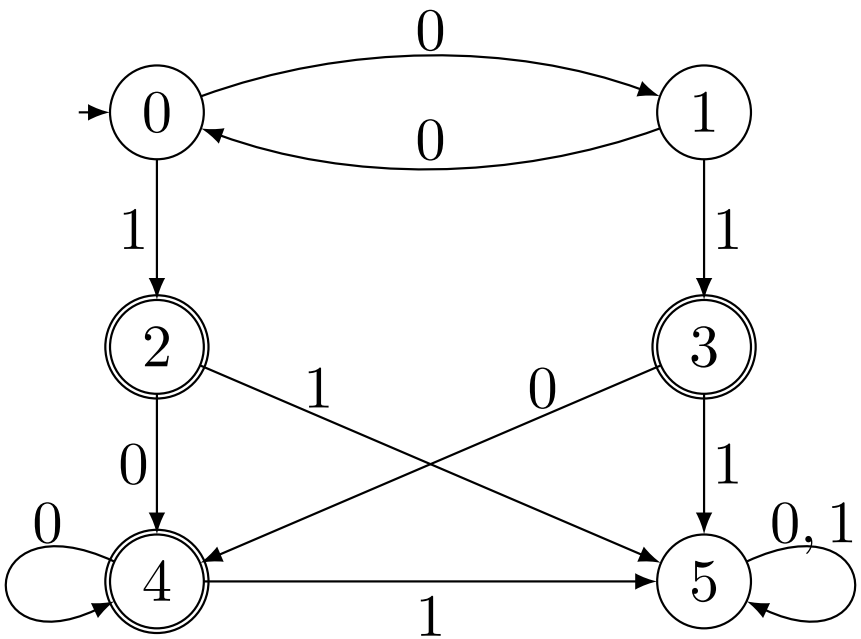
bei fixem Σ.

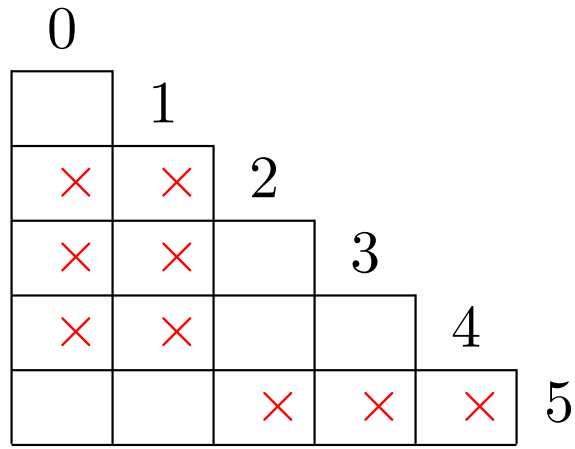
Beispiel 3.52



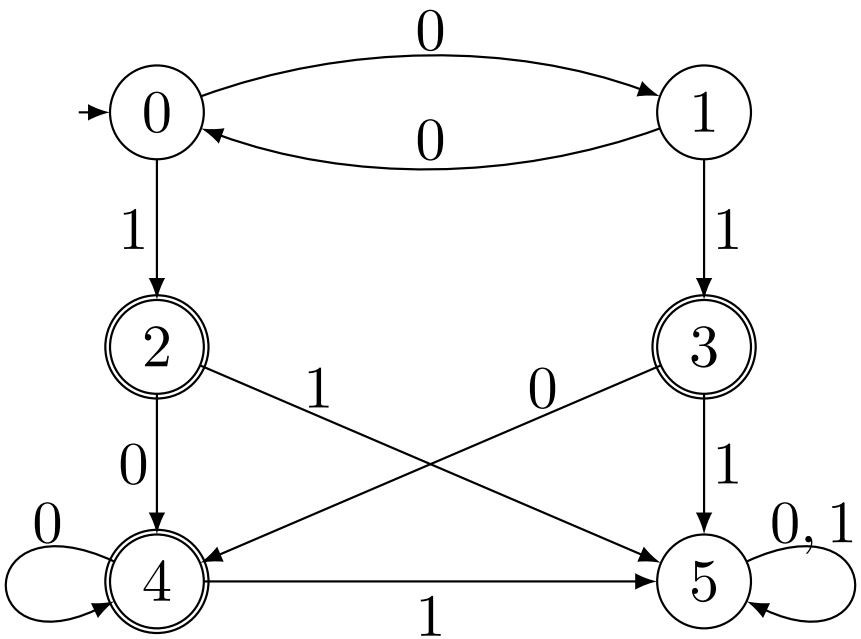


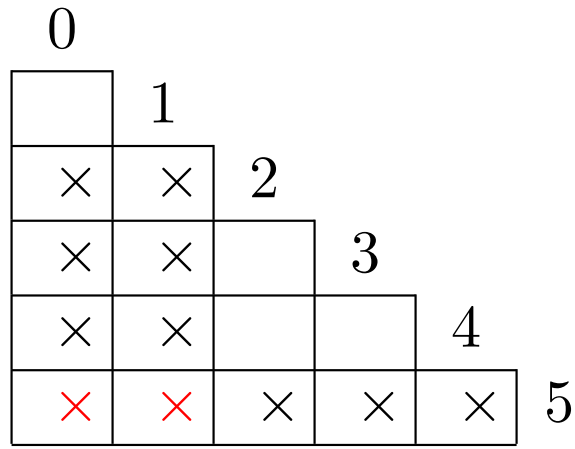
Beispiel 3.52



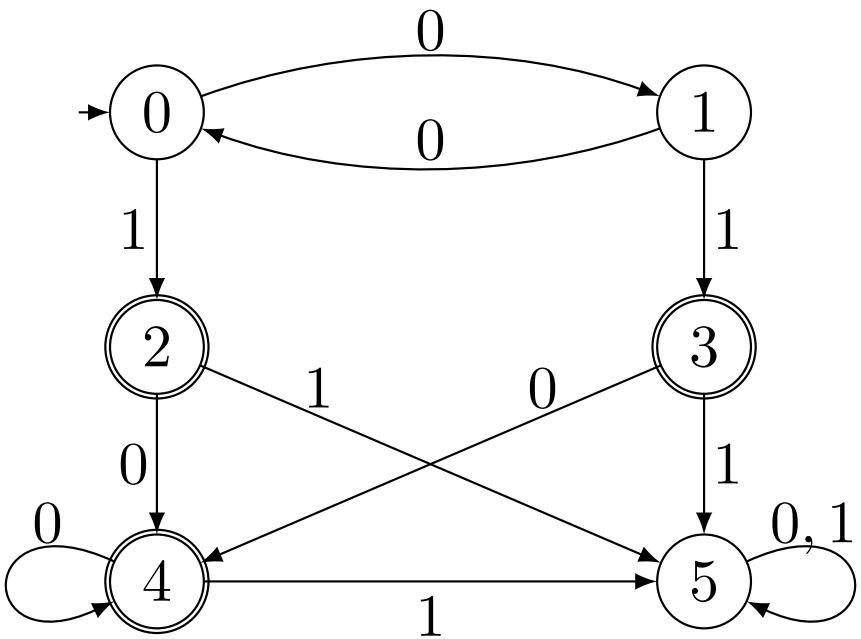


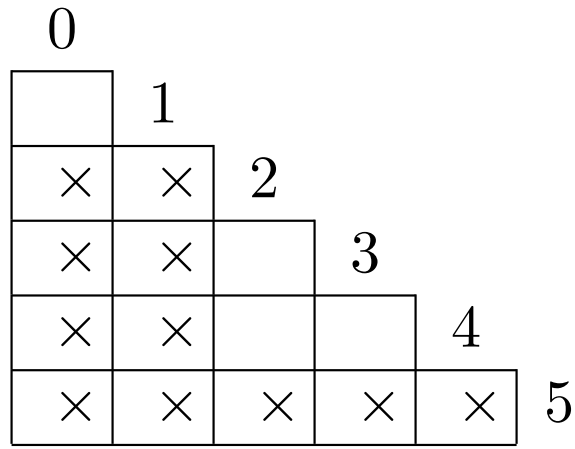
Beispiel 3.52

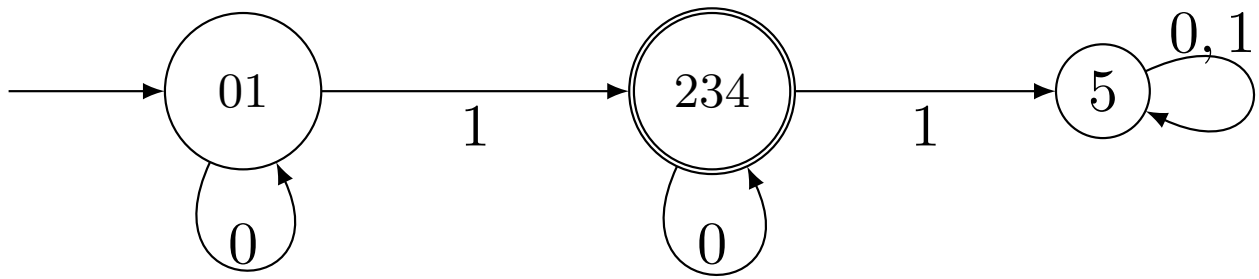




Beispiel 3.52







Von $O(n^4)$ zu $O(n^2)$ mit Abhängigkeitsanalyse:

Von $O(n^4)$ zu $O(n^2)$ mit Abhängigkeitsanalyse:

$\{p', q'\}$ unterscheidbar $\Longrightarrow$
$\{p, q\}$ unterscheidbar falls $\{p, q\} \xrightarrow{a} \{p', q'\}$

Von $O(n^4)$ zu $O(n^2)$ mit Abhängigkeitsanalyse:

$\{p', q'\}$ unterscheidbar $\Longrightarrow$
$\{p, q\}$ unterscheidbar falls $\{p, q\} \xrightarrow{a} \{p', q'\}$

$D[\{p', q'\}]$: Menge der Paare $\{p, q\}$ wie oben, anfangs leer

Von $O(n^4)$ zu $O(n^2)$ mit Abhängigkeitsanalyse:

$\{p', q'\}$ unterscheidbar $\Longrightarrow$
$\{p, q\}$ unterscheidbar falls $\{p, q\} \xrightarrow{a} \{p', q'\}$

$D[\{p', q'\}]$: Menge der Paare $\{p, q\}$ wie oben, anfangs leer

for all $\{p, q\} \subseteq Q$ mit $p \neq q$, $a \in \Sigma$ **do**
  $p' := \delta(p, a)$; $q' := \delta(q, a)$
  if $p' \neq q'$ **then** $D[\{p', q'\}] := D[\{p', q'\}] \cup \{\{p, q\}\}$
for all $p' \in F$, $q' \in Q \setminus F$ **do** $mark(\{p', q'\})$

$mark(\{p', q'\}) =$
  if $\{p', q'\}$ unmarkiert **then**
    markiere $\{p', q'\}$
    for all $\{p, q\} \in D[\{p', q'\}]$ **do** $mark(\{p, q\})$

Von $O(n^4)$ zu $O(n^2)$ mit Abhängigkeitsanalyse:

$$\{p', q'\} \text{ unterscheidbar } \Longrightarrow$$
$$\{p, q\} \text{ unterscheidbar falls } \{p, q\} \xrightarrow{a} \{p', q'\}$$

$D[\{p', q'\}]$: Menge der Paare $\{p, q\}$ wie oben, anfangs leer

for all $\{p, q\} \subseteq Q$ mit $p \neq q$, $a \in \Sigma$ **do**
  $p' := \delta(p, a)$; $q' := \delta(q, a)$
  if $p' \neq q'$ **then** $D[\{p', q'\}] := D[\{p', q'\}] \cup \{\{p, q\}\}$
for all $p' \in F$, $q' \in Q \setminus F$ **do** $mark(\{p', q'\})$

$mark(\{p', q'\}) =$
  if $\{p', q'\}$ unmarkiert **then**
    markiere $\{p', q'\}$
    for all $\{p, q\} \in D[\{p', q'\}]$ **do** $mark(\{p, q\})$

Komplexität: $O(n^2 + n^2) = O(n^2)$.

John Hopcroft.
An $n \log n$ Algorithm for Minimizing the States in a Finite Automaton. 1971.

Eine weitere Anwendung: Äquivalenztest von DFAs.

Eine weitere Anwendung: Äquivalenztest von DFAs.

1. Gegeben DFAs M_1 und M_2, bilde disjunkte Vereiningung. („Male M_1 und M_2 nebeneinander.“)
2. Berechne Menge der äquivalenten Zustände.
3. $L(M_1) = L(M_2)$ gdw die beiden Startzustände äquivalent sind.

Bisher: Der Minimierungsalgorithmus (zur Erinnerung).

1. Entferne alle von q_0 aus nicht erreichbaren Zustände.
2. Berechne die *äquivalenten* Zustände des Automaten.
3. Kollabiere den Automaten durch Zusammenfassung aller äquivalenten Zustände.

Formale Definition des “kollabierten Automaten”

Eine Relation $\approx \subseteq A \times A$ ist eine Äquivalenzrelation falls

- Reflexivität: $\forall a \in A.\ a \approx a$
- Symmetrie: $\forall a, b \in A.\ a \approx b \implies b \approx a$
- Transitivität: $\forall a, b, c \in A.\ a \approx b \wedge b \approx c \implies a \approx c$

Formale Definition des “kollabierten Automaten”

Eine Relation $\approx \subseteq A \times A$ ist eine Äquivalenzrelation falls

- Reflexivität: $\forall a \in A.\ a \approx a$
- Symmetrie: $\forall a, b \in A.\ a \approx b \implies b \approx a$
- Transitivität: $\forall a, b, c \in A.\ a \approx b \wedge b \approx c \implies a \approx c$

Äquivalenzklasse:

$$[a]_{\approx} := \{b \mid a \approx b\}$$

Es gilt:

$$[a]_{\approx} = [b]_{\approx} \quad \Leftrightarrow \quad a \approx b$$

Formale Definition des “kollabierten Automaten”

Eine Relation $\approx \subseteq A \times A$ ist eine Äquivalenzrelation falls

- Reflexivität: $\forall a \in A.\ a \approx a$
- Symmetrie: $\forall a, b \in A.\ a \approx b \implies b \approx a$
- Transitivität: $\forall a, b, c \in A.\ a \approx b \wedge b \approx c \implies a \approx c$

Äquivalenzklasse:

$$[a]_{\approx} := \{b \mid a \approx b\}$$

Es gilt:

$$[a]_{\approx} = [b]_{\approx} \quad \Leftrightarrow \quad a \approx b$$

Quotientenmenge:

$$A/{\approx} := \{[a]_{\approx} \mid a \in A\}$$

Im Folgenden sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DFA ohne unerreichbare Zustände.

Definition 3.53 (Äquivalenz von Zuständen)

$$p \equiv_M q \quad \Leftrightarrow \quad (\forall w \in \Sigma^*.\ \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F)$$

(Intuition: *Zwei Zustände sind äquivalent wenn sie dieselbe Sprache erkennen.*)

Fakt: $\equiv_M$ ist eine Äquivalenzrelation.

Wir schreiben $\equiv$ statt $\equiv_M$ wenn M klar ist.

Im Folgenden sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DFA ohne unerreichbare Zustände.

Definition 3.53 (Äquivalenz von Zuständen)

$$p \equiv_M q \quad \Leftrightarrow \quad (\forall w \in \Sigma^*.\ \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F)$$

(Intuition: *Zwei Zustände sind äquivalent wenn sie dieselbe Sprache erkennen.*)

Fakt: $\equiv_M$ ist eine Äquivalenzrelation.

Wir schreiben $\equiv$ statt $\equiv_M$ wenn M klar ist.

Erinnerung:

- $p \equiv_M q \implies \delta(p, a) \equiv_M \delta(q, a)$
- Algorithmus U liefert die unterscheidbaren Zustände, also $\not\equiv$.

In der weiteren Analyse beziehen wir uns direkt auf $\equiv$, nicht mehr auf den Algorithmus.

Die „Kollabierung“ von M bzgl. $\equiv$ ist der *Quotientenautomat*:

Definition 3.54 (Quotientenautomat)

$$\begin{aligned} M/\equiv &:= (Q/\equiv,\ \Sigma, \delta',\ [q_0]_\equiv,\ F/\equiv) \\ \delta'([p]_\equiv, a) &:= [\delta(p,a)]_\equiv \end{aligned}$$

Die „Kollabierung“ von M bzgl. $\equiv$ ist der *Quotientenautomat*:

Definition 3.54 (Quotientenautomat)

$$\begin{aligned} M/{\equiv} &:= (Q/{\equiv},\ \Sigma, \delta',\ [q_0]_\equiv,\ F/{\equiv}) \\ \delta'([p]_\equiv, a) &:= [\delta(p,a)]_\equiv \end{aligned}$$

Die Definition von δ' ist wohlgeformt da unabhängig von der Wahl des Repräsentanten p:

$$\begin{aligned} [p]_\equiv = [p']_\equiv &\implies p \equiv p' \implies \delta(p,a) \equiv \delta(p',a) \\ &\implies [\delta(p,a)]_\equiv = [\delta(p',a)]_\equiv \end{aligned}$$

Die „Kollabierung“ von M bzgl. $\equiv$ ist der *Quotientenautomat*:

Definition 3.54 (Quotientenautomat)

$$\begin{aligned} M/{\equiv} &:= (Q/{\equiv},\ \Sigma, \delta',\ [q_0]_\equiv,\ F/{\equiv}) \\ \delta'([p]_\equiv, a) &:= [\delta(p,a)]_\equiv \end{aligned}$$

Die Definition von δ' ist wohlgeformt da unabhängig von der Wahl des Repräsentanten p:

$$\begin{aligned} [p]_\equiv = [p']_\equiv &\implies p \equiv p' \implies \delta(p,a) \equiv \delta(p',a) \\ &\implies [\delta(p,a)]_\equiv = [\delta(p',a)]_\equiv \end{aligned}$$

Lemma 3.55

M und $M/{\equiv}$ erkennen dieselbe Sprache: $L(M/{\equiv}) = L(M)$.

Die „Kollabierung“ von M bzgl. $\equiv$ ist der *Quotientenautomat*:

Definition 3.54 (Quotientenautomat)

$$\begin{aligned} M/\equiv &:= (Q/\equiv,\ \Sigma, \delta',\ [q_0]_\equiv,\ F/\equiv) \\ \delta'([p]_\equiv, a) &:= [\delta(p,a)]_\equiv \end{aligned}$$

Die Definition von δ' ist wohlgeformt da unabhängig von der Wahl des Repräsentanten p:

$$\begin{aligned} [p]_\equiv = [p']_\equiv \quad &\Longrightarrow \quad p \equiv p' \quad \Longrightarrow \quad \delta(p,a) \equiv \delta(p',a) \\ &\Longrightarrow \quad [\delta(p,a)]_\equiv = [\delta(p',a)]_\equiv \end{aligned}$$

Lemma 3.55

M und $M/\equiv$ erkennen dieselbe Sprache: $L(M/\equiv) = L(M)$.

Beweis: Übung.

Minimalität des Quotientenautomaten

Eine Beobachtung: Für $p := \hat{\delta}(q_0, u)$ und $q := \hat{\delta}(q_0, v)$ gilt

$$\begin{aligned} p \equiv_M q \quad &\Leftrightarrow \quad \forall w \in \Sigma^*.\ \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F \\ &\Leftrightarrow \quad \forall w \in \Sigma^*.\ \hat{\delta}(q_0, uw) \in F \Leftrightarrow \hat{\delta}(q_0, vw) \in F \\ &\Leftrightarrow \quad \forall w \in \Sigma^*.\ uw \in L(M) \Leftrightarrow vw \in L(M) \end{aligned}$$

Minimalität des Quotientenautomaten

Eine Beobachtung: Für $p := \hat{\delta}(q_0, u)$ und $q := \hat{\delta}(q_0, v)$ gilt

$$\begin{aligned} p \equiv_M q &\Leftrightarrow \forall w \in \Sigma^*.\ \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F \\ &\Leftrightarrow \forall w \in \Sigma^*.\ \hat{\delta}(q_0, uw) \in F \Leftrightarrow \hat{\delta}(q_0, vw) \in F \\ &\Leftrightarrow \forall w \in \Sigma^*.\ uw \in L(M) \Leftrightarrow vw \in L(M) \end{aligned}$$

Definition 3.56

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

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

Minimalität des Quotientenautomaten

Eine Beobachtung: Für $p := \hat{\delta}(q_0, u)$ und $q := \hat{\delta}(q_0, v)$ gilt

$$\begin{aligned} p \equiv_M q &\Leftrightarrow \forall w \in \Sigma^*.\ \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F \\ &\Leftrightarrow \forall w \in \Sigma^*.\ \hat{\delta}(q_0, uw) \in F \Leftrightarrow \hat{\delta}(q_0, vw) \in F \\ &\Leftrightarrow \forall w \in \Sigma^*.\ uw \in L(M) \Leftrightarrow vw \in L(M) \end{aligned}$$

Definition 3.56

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

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

Obige Beobachtung lässt sich nun schreiben als

$$\hat{\delta}(q_0, u) \equiv_M \hat{\delta}(q_0, v) \quad \Leftrightarrow \quad u \equiv_{L(M)} v$$

(Zwei Wörter sind äquivalent gdw sie zu äquivalenten Zuständen führen)

Achtung

$p \equiv_M q$ ist eine Relation auf Zuständen von M

$u \equiv_L v$ ist eine Relation auf Wörtern

$$u \equiv_{L(M)} v \quad \Leftrightarrow \quad \hat{\delta}(q_0, u) \equiv_M \hat{\delta}(q_0, v)$$

Da alle Zustände von q_0 erreichbar sind, gilt sogar: Die Abbildung

$$[u]_{\equiv_{L(M)}} \quad \mapsto \quad [\hat{\delta}(q_0, u)]_{\equiv_M}$$

ist eine Bijektion zwischen den $\equiv_{L(M)}$ und $\equiv_M$ Äquivalenzklassen.

$$u \equiv_{L(M)} v \quad \Leftrightarrow \quad \hat{\delta}(q_0, u) \equiv_M \hat{\delta}(q_0, v)$$

Da alle Zustände von q_0 erreichbar sind, gilt sogar: Die Abbildung

$$[u]_{\equiv_{L(M)}} \quad \mapsto \quad [\hat{\delta}(q_0, u)]_{\equiv_M}$$

ist eine Bijektion zwischen den $\equiv_{L(M)}$ und $\equiv_M$ Äquivalenzklassen.

Satz 3.57

Ist M ein DFA ohne unerreichbare Zustände, so ist der von Algorithmus U *berechnete Quotientenautomat $M/\equiv$ ein minimaler DFA für $L(M)$.*

$$u \equiv_{L(M)} v \quad \Leftrightarrow \quad \hat{\delta}(q_0, u) \equiv_M \hat{\delta}(q_0, v)$$

Da alle Zustände von q_0 erreichbar sind, gilt sogar: Die Abbildung

$$[u]_{\equiv_{L(M)}} \quad \mapsto \quad [\hat{\delta}(q_0, u)]_{\equiv_M}$$

ist eine Bijektion zwischen den $\equiv_{L(M)}$ und $\equiv_M$ Äquivalenzklassen.

Satz 3.57

Ist M ein DFA ohne unerreichbare Zustände, so ist der von Algorithmus U *berechnete Quotientenautomat $M/\equiv$ ein minimaler DFA für $L(M)$.*

Beweis:

Sei $L := L(M)$ und M' ein DFA mit $L(M') = L$. Dann gilt:

$$|Q'| \geq |Q'/\equiv_{M'}| = |\Sigma^*/\equiv_L| = |Q/\equiv_M|$$

□

Es gilt sogar:

Fakt 3.58

Alle Quotientenautomaten $M/\equiv_M$ für die gleiche Sprache $L(M)$ haben die gleiche Struktur, d.h. sie unterscheiden sich nur durch eine Umbenennung der Zustände.

Es gilt sogar:

Fakt 3.58

Alle Quotientenautomaten $M/\equiv_M$ für die gleiche Sprache $L(M)$ haben die gleiche Struktur, d.h. sie unterscheiden sich nur durch eine Umbenennung der Zustände.

Daher beschriften wir die Zustände des kanonischen Minimalautomaten für eine Sprache L mit $\equiv_L$ Äquivalenzklassen.

Es gilt sogar:

Fakt 3.58

Alle Quotientenautomaten $M/\equiv_M$ für die gleiche Sprache $L(M)$ haben die gleiche Struktur, d.h. sie unterscheiden sich nur durch eine Umbenennung der Zustände.

Daher beschriften wir die Zustände des kanonischen Minimalautomaten für eine Sprache L mit $\equiv_L$ Äquivalenzklassen.

Beispiel 3.59

Sei $L := \{w \in \{0,1\}^* \mid w \text{ endet mit } 00\}$.

Es gilt sogar:

Fakt 3.58

Alle Quotientenautomaten $M/\equiv_M$ für die gleiche Sprache $L(M)$ haben die gleiche Struktur, d.h. sie unterscheiden sich nur durch eine Umbenennung der Zustände.

Daher beschriften wir die Zustände des kanonischen Minimalautomaten für eine Sprache L mit $\equiv_L$ Äquivalenzklassen.

Beispiel 3.59

Sei $L := \{w \in \{0,1\}^* \mid w \text{ endet mit } 00\}$.
Die einzigen drei $\equiv_L$ Äquivalenzklassen sind:

$$[\epsilon]_{\equiv_L} \quad = \quad \{w \mid w \text{ endet nicht mit } 0\}$$

Es gilt sogar:

Fakt 3.58

Alle Quotientenautomaten $M/\equiv_M$ für die gleiche Sprache $L(M)$ haben die gleiche Struktur, d.h. sie unterscheiden sich nur durch eine Umbenennung der Zustände.

Daher beschriften wir die Zustände des kanonischen Minimalautomaten für eine Sprache L mit $\equiv_L$ Äquivalenzklassen.

Beispiel 3.59

Sei $L := \{w \in \{0,1\}^* \mid w \text{ endet mit } 00\}$.
Die einzigen drei $\equiv_L$ Äquivalenzklassen sind:

$$
\begin{aligned}
[\epsilon]_{\equiv_L} &= \{w \mid w \text{ endet nicht mit } 0\} \\
[0]_{\equiv_L} &= \{w \mid w \text{ endet mit 0, aber nicht mit } 00\}
\end{aligned}
$$

Es gilt sogar:

Fakt 3.58

Alle Quotientenautomaten $M/\equiv_M$ für die gleiche Sprache $L(M)$ haben die gleiche Struktur, d.h. sie unterscheiden sich nur durch eine Umbenennung der Zustände.

Daher beschriften wir die Zustände des kanonischen Minimalautomaten für eine Sprache L mit $\equiv_L$ Äquivalenzklassen.

Beispiel 3.59

Sei $L := \{w \in \{0,1\}^* \mid w \text{ endet mit } 00\}$.
Die einzigen drei $\equiv_L$ Äquivalenzklassen sind:

$$
\begin{aligned}
[\epsilon]_{\equiv_L} &= \{w \mid w \text{ endet nicht mit } 0\} \\
[0]_{\equiv_L} &= \{w \mid w \text{ endet mit 0, aber nicht mit } 00\} \\
[00]_{\equiv_L} &= \{w \mid w \text{ endet mit } 00\}
\end{aligned}
$$

Definition 3.60 (Kanonischer Minimalautomat)

$$M_L := (\Sigma^*/\equiv_L,\ \Sigma,\ \delta_L,\ [\epsilon]_{\equiv_L},\ F_L)$$

mit $\delta_L([w]_{\equiv_L}, a) := [wa]_{\equiv_L}$ und $F_L := \{[w]_{\equiv_L} \mid w \in L\}$.

Definition 3.60 (Kanonischer Minimalautomat)

$$M_L := (\Sigma^*/\equiv_L,\ \Sigma,\ \delta_L,\ [\epsilon]_{\equiv_L},\ F_L)$$

mit $\delta_L([w]_{\equiv_L}, a) := [wa]_{\equiv_L}$ und $F_L := \{[w]_{\equiv_L} \mid w \in L\}$.

Man sieht: δ_L ist wohldefiniert und $\hat{\delta}_L([\epsilon]_{\equiv_L}, w) = [w]_{\equiv_L}$.
Dann gilt offensichtlich $L(M_L) = L$.

Definition 3.60 (Kanonischer Minimalautomat)

$$M_L := (\Sigma^*/\equiv_L, \Sigma, \delta_L, [\epsilon]_{\equiv_L}, F_L)$$

mit $\delta_L([w]_{\equiv_L}, a) := [wa]_{\equiv_L}$ und $F_L := \{[w]_{\equiv_L} \mid w \in L\}$.

Man sieht: δ_L ist wohldefiniert und $\hat{\delta}_L([\epsilon]_{\equiv_L}, w) = [w]_{\equiv_L}$.
Dann gilt offensichtlich $L(M_L) = L$.

Satz 3.61 (Myhill-Nerode)

Eine Sprache $L \subseteq \Sigma^*$ ist genau dann regulär, wenn $\equiv_L$ endlich viele Äquivalenzklassen hat.

Definition 3.60 (Kanonischer Minimalautomat)

$$M_L := (\Sigma^*/\equiv_L, \Sigma, \delta_L, [\epsilon]_{\equiv_L}, F_L)$$

mit $\delta_L([w]_{\equiv_L}, a) := [wa]_{\equiv_L}$ und $F_L := \{[w]_{\equiv_L} \mid w \in L\}$.

Man sieht: δ_L ist wohldefiniert und $\hat{\delta}_L([\epsilon]_{\equiv_L}, w) = [w]_{\equiv_L}$.
Dann gilt offensichtlich $L(M_L) = L$.

Satz 3.61 (Myhill-Nerode)

Eine Sprache $L \subseteq \Sigma^*$ ist genau dann regulär, wenn $\equiv_L$ endlich viele Äquivalenzklassen hat.

Beweis:

„$\Longrightarrow$“: Ist L regulär, so wird L von einem DFA M akzeptiert.
Daher hat $\equiv_L$ so viele Äquivalenzklassen wie $M/\equiv_M$ Zustände.

Definition 3.60 (Kanonischer Minimalautomat)

$$M_L := (\Sigma^*/\equiv_L, \Sigma, \delta_L, [\epsilon]_{\equiv_L}, F_L)$$

mit $\delta_L([w]_{\equiv_L}, a) := [wa]_{\equiv_L}$ und $F_L := \{[w]_{\equiv_L} \mid w \in L\}$.

Man sieht: δ_L ist wohldefiniert und $\hat{\delta}_L([\epsilon]_{\equiv_L}, w) = [w]_{\equiv_L}$.
Dann gilt offensichtlich $L(M_L) = L$.

Satz 3.61 (Myhill-Nerode)

Eine Sprache $L \subseteq \Sigma^*$ ist genau dann regulär, wenn $\equiv_L$ endlich viele Äquivalenzklassen hat.

Beweis:

„$\Longrightarrow$“: Ist L regulär, so wird L von einem DFA M akzeptiert.
Daher hat $\equiv_L$ so viele Äquivalenzklassen wie $M/\equiv_M$ Zustände.
„$\Longleftarrow$“: Hat $\equiv_L$ endlich viele Äquivalenzklasse,
so ist M_L ein DFA mit $L(M_L) = L$. □

Wie sieht M_L aus wenn L *nicht* regulär ist?

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$\rightarrow [\epsilon]$

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$\rightarrow [\epsilon] \quad \stackrel{0}{\rightarrow} \quad [0]$$

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$\to [\epsilon] \quad \xrightarrow{0} \quad [0] \quad \xrightarrow{0} \quad [0^2]$$

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$\to [\epsilon] \quad \overset{0}{\to} \quad [0] \quad \overset{0}{\to} \quad [0^2] \quad \overset{0}{\to} \cdots \overset{0}{\to} \quad [0^i]$$

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$\to [\epsilon] \xrightarrow{0} [0] \xrightarrow{0} [0^2] \xrightarrow{0} \cdots \xrightarrow{0} [0^i] \xrightarrow{0} \cdots$$

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$\to [\epsilon] \xrightarrow{0} [0] \xrightarrow{0} [0^2] \xrightarrow{0} \cdots \xrightarrow{0} [0^i] \xrightarrow{0} \cdots$$

$$\downarrow 1$$

$$[01]$$

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$\begin{array}{ccccccccc} \to [\epsilon] & \xrightarrow{0} & [0] & \xrightarrow{0} & [0^2] & \xrightarrow{0} \cdots \xrightarrow{0} & [0^i] & \xrightarrow{0} & \cdots \\ & & \downarrow 1 & & \downarrow 1 & & & & \\ & & [01] & & [0^2 1] & & & & \end{array}$$

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$\begin{array}{ccccccccccc} \to [\epsilon] & \xrightarrow{0} & [0] & \xrightarrow{0} & [0^2] & \xrightarrow{0} \cdots \xrightarrow{0} & [0^i] & \xrightarrow{0} & \cdots \\ & & \downarrow 1 & & \downarrow 1 & & \downarrow 1 & & \\ & & [01] & & [0^2 1] & & [0^i 1] & & \end{array}$$

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$
\begin{array}{ccccccccc}
\to [\epsilon] & \xrightarrow{0} & [0] & \xrightarrow{0} & [0^2] & \xrightarrow{0} \cdots \xrightarrow{0} & [0^i] & \xrightarrow{0} & \cdots \\
 & & \downarrow 1 & & \downarrow 1 & & \downarrow 1 & & \\
 & & [01] & \xleftarrow{1} & [0^2 1] & & [0^i 1] & &
\end{array}
$$

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$
\begin{array}{ccccccccc}
\to [\epsilon] & \xrightarrow{0} & [0] & \xrightarrow{0} & [0^2] & \xrightarrow{0} \cdots \xrightarrow{0} & [0^i] & \xrightarrow{0} & \cdots \\
 & & \downarrow 1 & & \downarrow 1 & & \downarrow 1 & & \\
 & & [01] & \xleftarrow{1} & [0^2 1] & \xleftarrow{1} \cdots \xleftarrow{1} & [0^i 1] & &
\end{array}
$$

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$
\begin{array}{ccccccccc}
\to [\epsilon] & \xrightarrow{0} & [0] & \xrightarrow{0} & [0^2] & \xrightarrow{0} \cdots \xrightarrow{0} & [0^i] & \xrightarrow{0} & \cdots \\
 & & \downarrow 1 & & \downarrow 1 & & \downarrow 1 & & \\
 & & [01] & \xleftarrow{1} & [0^2 1] & \xleftarrow{1} \cdots \xleftarrow{1} & [0^i 1] & \xleftarrow{1} & \cdots
\end{array}
$$

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$
\begin{array}{ccccccccc}
\to [\epsilon] & \xrightarrow{0} & [0] & \xrightarrow{0} & [0^2] & \xrightarrow{0} \cdots \xrightarrow{0} & [0^i] & \xrightarrow{0} & \cdots \\
 & & \downarrow 1 & & \downarrow 1 & & \downarrow 1 & & \\
 & & [01] & \xleftarrow{1} & [0^2 1] & \xleftarrow{1} \cdots \xleftarrow{1} & [0^i 1] & \xleftarrow{1} & \cdots
\end{array}
$$

Warum $0^i \not\equiv_L 0^j$ falls $i \neq j$?
Was fehlt noch?

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$
\begin{array}{ccccccccc}
\to [\epsilon] & \xrightarrow{0} & [0] & \xrightarrow{0} & [0^2] & \xrightarrow{0} \cdots \xrightarrow{0} & [0^i] & \xrightarrow{0} & \cdots \\
 & & \downarrow 1 & & \downarrow 1 & & \downarrow 1 & & \\
 & & [01] & \xleftarrow{1} & [0^2 1] & \xleftarrow{1} \cdots \xleftarrow{1} & [0^i 1] & \xleftarrow{1} & \cdots
\end{array}
$$

Warum $0^i \not\equiv_L 0^j$ falls $i \neq j$?
Was fehlt noch?

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$\begin{array}{ccccccccc}
\to [\epsilon] & \xrightarrow{0} & [0] & \xrightarrow{0} & [0^2] & \xrightarrow{0} \cdots \xrightarrow{0} & [0^i] & \xrightarrow{0} & \cdots \\
 & & \downarrow 1 & & \downarrow 1 & & \downarrow 1 & & \\
 & & [01] & \xleftarrow{1} & [0^2 1] & \xleftarrow{1} \cdots \xleftarrow{1} & [0^i 1] & \xleftarrow{1} & \cdots
\end{array}$$

Warum $0^i \not\equiv_L 0^j$ falls $i \neq j$?
Was fehlt noch?

Vollständige Methode um Nichtregularität von L zu zeigen:

Wie sieht M_L aus wenn L *nicht* regulär ist?

Beispiel 3.62

$L = \{0^i 1^i \mid i \in \mathbb{N}\}$

$$\begin{array}{ccccccccc}
\to [\epsilon] & \xrightarrow{0} & [0] & \xrightarrow{0} & [0^2] & \xrightarrow{0} \cdots \xrightarrow{0} & [0^i] & \xrightarrow{0} & \cdots \\
 & & \downarrow 1 & & \downarrow 1 & & \downarrow 1 & & \\
 & & [01] & \xleftarrow{1} & [0^2 1] & \xleftarrow{1} \cdots \xleftarrow{1} & [0^i 1] & \xleftarrow{1} & \cdots
\end{array}$$

Warum $0^i \not\equiv_L 0^j$ falls $i \neq j$?
Was fehlt noch?

Vollständige Methode um Nichtregularität von L zu zeigen:

Gib unendliche Menge $w_1, w_2, \ldots$ an mit $w_i \not\equiv_L w_j$ falls $i \neq j$.

Bemerkung

Eindeutigkeit des minimalen Automaten (modulo Umbenennung der Zustände) gilt nur bei DFAs, nicht bei NFAs!

Bemerkung

Eindeutigkeit des minimalen Automaten (modulo Umbenennung der Zustände) gilt nur bei DFAs, nicht bei NFAs!

Aufgabe: Finde Gegenbeispiel für NFAs

Rückblick reguläre Sprachen

- DFA, NFA, ϵ-NFA und RE definieren die gleiche Sprachklasse.
  - Potenzmengenkonstruktion (exponentiell)
  - Strukturelle Übersetzung von RE nach ϵ-NFA
  - Übersetzung NFA nach RE, zB über Gleichungssysteme (exponentiell)
- Regularitätserhaltende Konstruktionen auf Sprachen bzw Automaten:
  - $\cup$, $\cap$, $\ldots$, ^{R}, $\ldots$
  - Für welche Darstellung wie teuer? (Komplement, Produkt, $\ldots$)
  - NFA kompakt aber oft teuer; DFA oft billiger
- Entscheidungsprobleme auf Automaten und REs:
  - Direkt entscheidbar? Auf Automat? (Oft mit Erreichbarkeit) Auf RE?
  - Reduzierbar auf anderes Problem? ZB $L(A) \subseteq L(B)$ auf $L(A) \cap \overline{L(B)} = \emptyset$
  - Für welche Darstellung wie teuer?

- Pumping-Lemma
- Äquivalenzen $\equiv$ zw REs:
  - Standardregeln: Kommutativität etc, Beweis mit $L(.)$
  - $\alpha \equiv \beta$ entscheidbar da $L(\alpha) = L(\beta)$ über Automaten entscheidbar
  - Auch für REs mit Variablen entscheidbar: betrachte Variablen als Konstanten
  - Gleichungssysteme lösbar
    durch Variablenelimination mit Ardens Lemma
- Minimierung
  - Algorithmen zur Kollabierung
  - Relationen $\equiv_M$ und $\equiv_L$
  - Quotientenautomat $M/\equiv_M$ minimal,
    da gleichgroß wie kanonischer minimaler Automat $M_{L(M)}$
  - L genau dann regulär wenn $\equiv_L$ endlich viele Klassen hat,
    dh wenn M_L endlich ist (Myhill-Nerode).

4. Kontextfreie Sprachen

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

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

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\text{<Expr>} & \to & \text{<Term>} \\
\text{<Expr>} & \to & \text{<Expr>} + \text{<Term>} \\
\text{<Term>} & \to & \text{<Factor>} \\
\text{<Term>} & \to & \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} & \to & a \\
\text{<Factor>} & \to & (\text{<Expr>})
\end{array}
$$

Eine (Links)Ableitung:

<Expr> $\to$

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

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\text{<Expr>} & \to & \text{<Term>} \\
\text{<Expr>} & \to & \text{<Expr>} + \text{<Term>} \\
\text{<Term>} & \to & \text{<Factor>} \\
\text{<Term>} & \to & \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} & \to & a \\
\text{<Factor>} & \to & (\text{<Expr>})
\end{array}
$$

Eine (Links)Ableitung:

<Expr> $\to$

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

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\text{<Expr>} & \to & \text{<Term>} \\
\text{<Expr>} & \to & \text{<Expr>} + \text{<Term>} \\
\text{<Term>} & \to & \text{<Factor>} \\
\text{<Term>} & \to & \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} & \to & a \\
\text{<Factor>} & \to & (\text{<Expr>})
\end{array}
$$

Eine (Links)Ableitung:

<Expr> $\to$ <Term>

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

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\text{<Expr>} & \to & \text{<Term>} \\
\text{<Expr>} & \to & \text{<Expr>} + \text{<Term>} \\
\text{<Term>} & \to & \text{<Factor>} \\
\text{<Term>} & \to & \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} & \to & a \\
\text{<Factor>} & \to & (\text{<Expr>})
\end{array}
$$

Eine (Links)Ableitung:

$\text{<Expr>} \to \text{<Term>} \to \text{<Term>} * \text{<Factor>}$

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

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\text{<Expr>} & \to & \text{<Term>} \\
\text{<Expr>} & \to & \text{<Expr>} + \text{<Term>} \\
\text{<Term>} & \to & \text{<Factor>} \\
\text{<Term>} & \to & \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} & \to & a \\
\text{<Factor>} & \to & (\text{<Expr>})
\end{array}
$$

Eine (Links)Ableitung:

$\text{<Expr>} \to \text{<Term>} \to \text{<Term>} * \text{<Factor>}$

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

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\text{<Expr>} & \to & \text{<Term>} \\
\text{<Expr>} & \to & \text{<Expr>} + \text{<Term>} \\
\text{<Term>} & \to & \text{<Factor>} \\
\text{<Term>} & \to & \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} & \to & a \\
\text{<Factor>} & \to & (\text{<Expr>})
\end{array}
$$

Eine (Links)Ableitung:

<Expr> $\to$ <Term> $\to$ <Term> $*$ <Factor>
$\to$ <Factor> $*$ <Factor>

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

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

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

Eine (Links)Ableitung:

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

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

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{aligned}
\text{<Expr>} &\to \text{<Term>} \\
\text{<Expr>} &\to \text{<Expr>} + \text{<Term>} \\
\text{<Term>} &\to \text{<Factor>} \\
\text{<Term>} &\to \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} &\to a \\
\text{<Factor>} &\to (\text{<Expr>})
\end{aligned}
$$

Eine (Links)Ableitung:

$$
\begin{aligned}
&\text{<Expr>} \to \text{<Term>} \to \text{<Term>} * \text{<Factor>} \\
&\to \text{<Factor>} * \text{<Factor>} \to a * \text{<Factor>} \\
&\to a * (a + a)
\end{aligned}
$$

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\text{<Expr>} & \to & \text{<Term>} \\
\text{<Expr>} & \to & \text{<Expr>} + \text{<Term>} \\
\text{<Term>} & \to & \text{<Factor>} \\
\text{<Term>} & \to & \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} & \to & a \\
\text{<Factor>} & \to & (\text{<Expr>})
\end{array}
$$

Eine (Links)Ableitung:

<Expr> $\to$ <Term> $\to$ <Term> $*$ <Factor>
$\to$ <Factor> $*$ <Factor> $\to a *$ <Factor>
$\to a * (\text{<Expr>})$

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

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{aligned}
\text{<Expr>} &\to \text{<Term>} \\
\text{<Expr>} &\to \text{<Expr>} + \text{<Term>} \\
\text{<Term>} &\to \text{<Factor>} \\
\text{<Term>} &\to \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} &\to a \\
\text{<Factor>} &\to (\text{<Expr>})
\end{aligned}
$$

Eine (Links)Ableitung:

$$
\begin{aligned}
&\text{<Expr>} \to \text{<Term>} \to \text{<Term>} * \text{<Factor>} \\
&\to \text{<Factor>} * \text{<Factor>} \to a * \text{<Factor>} \\
&\to a * (\text{<Expr>}) \to a * (\text{<Expr>} + \text{<Term>}) \\
&\to a * (a + a)
\end{aligned}
$$

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{aligned}
\text{<Expr>} &\to \text{<Term>} \\
\text{<Expr>} &\to \text{<Expr>} + \text{<Term>} \\
\text{<Term>} &\to \text{<Factor>} \\
\text{<Term>} &\to \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} &\to a \\
\text{<Factor>} &\to (\text{<Expr>})
\end{aligned}
$$

Eine (Links)Ableitung:

$$
\begin{aligned}
&\text{<Expr>} \to \text{<Term>} \to \text{<Term>} * \text{<Factor>} \\
&\to \text{<Factor>} * \text{<Factor>} \to a * \text{<Factor>} \\
&\to a * (\text{<Expr>}) \to a * (\text{<Expr>} + \text{<Term>}) \\
&\to a * (a + a)
\end{aligned}
$$

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\text{<Expr>} & \to & \text{<Term>} \\
\text{<Expr>} & \to & \text{<Expr>} + \text{<Term>} \\
\text{<Term>} & \to & \text{<Factor>} \\
\text{<Term>} & \to & \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} & \to & a \\
\text{<Factor>} & \to & (\text{<Expr>})
\end{array}
$$

Eine (Links)Ableitung:

$$
\begin{aligned}
&\text{<Expr>} \to \text{<Term>} \to \text{<Term>} * \text{<Factor>} \\
&\to \text{<Factor>} * \text{<Factor>} \to a * \text{<Factor>} \\
&\to a * (\text{<Expr>}) \to a * (\text{<Expr>} + \text{<Term>}) \\
&\to a * (\text{<Term>} + \text{<Term>}) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \to a * (a + a)
\end{aligned}
$$

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{aligned}
\text{<Expr>} &\to \text{<Term>} \\
\text{<Expr>} &\to \text{<Expr>} + \text{<Term>} \\
\text{<Term>} &\to \text{<Factor>} \\
\text{<Term>} &\to \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} &\to a \\
\text{<Factor>} &\to (\text{<Expr>})
\end{aligned}
$$

Eine (Links)Ableitung:

$$
\begin{aligned}
&\text{<Expr>} \to \text{<Term>} \to \text{<Term>} * \text{<Factor>} \\
&\to \text{<Factor>} * \text{<Factor>} \to a * \text{<Factor>} \\
&\to a * (\text{<Expr>}) \to a * (\text{<Expr>} + \text{<Term>}) \\
&\to a * (\text{<Term>} + \text{<Term>}) \\
&\to a * (a + a)
\end{aligned}
$$

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\text{<Expr>} & \to & \text{<Term>} \\
\text{<Expr>} & \to & \text{<Expr>} + \text{<Term>} \\
\text{<Term>} & \to & \text{<Factor>} \\
\text{<Term>} & \to & \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} & \to & a \\
\text{<Factor>} & \to & (\text{<Expr>})
\end{array}
$$

Eine (Links)Ableitung:

$$
\begin{aligned}
&\text{<Expr>} \to \text{<Term>} \to \text{<Term>} * \text{<Factor>} \\
&\to \text{<Factor>} * \text{<Factor>} \to a * \text{<Factor>} \\
&\to a * (\text{<Expr>}) \to a * (\text{<Expr>} + \text{<Term>}) \\
&\to a * (\text{<Term>} + \text{<Term>}) \to a * (\text{<Factor>} + \text{<Term>}) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \to a * (a + a)
\end{aligned}
$$

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{aligned}
\text{<Expr>} &\to \text{<Term>} \\
\text{<Expr>} &\to \text{<Expr>} + \text{<Term>} \\
\text{<Term>} &\to \text{<Factor>} \\
\text{<Term>} &\to \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} &\to a \\
\text{<Factor>} &\to (\text{<Expr>})
\end{aligned}
$$

Eine (Links)Ableitung:

$$
\begin{aligned}
&\text{<Expr>} \to \text{<Term>} \to \text{<Term>} * \text{<Factor>} \\
&\to \text{<Factor>} * \text{<Factor>} \to a * \text{<Factor>} \\
&\to a * (\text{<Expr>}) \to a * (\text{<Expr>} + \text{<Term>}) \\
&\to a * (\text{<Term>} + \text{<Term>}) \to a * (\text{<Factor>} + \text{<Term>}) \\
&\to a * (a + a)
\end{aligned}
$$

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\text{<Expr>} & \to & \text{<Term>} \\
\text{<Expr>} & \to & \text{<Expr>} + \text{<Term>} \\
\text{<Term>} & \to & \text{<Factor>} \\
\text{<Term>} & \to & \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} & \to & a \\
\text{<Factor>} & \to & (\text{<Expr>})
\end{array}
$$

Eine (Links)Ableitung:

$$
\begin{array}{l}
\text{<Expr>} \to \text{<Term>} \to \text{<Term>} * \text{<Factor>} \\
\to \text{<Factor>} * \text{<Factor>} \to a * \text{<Factor>} \\
\to a * (\text{<Expr>}) \to a * (\text{<Expr>} + \text{<Term>}) \\
\to a * (\text{<Term>} + \text{<Term>}) \to a * (\text{<Factor>} + \text{<Term>}) \\
\to a * (a + \text{<Term>}) \qquad\qquad\qquad\qquad \to a * (a + a)
\end{array}
$$

4. Kontextfreie Sprachen

4.1 Kontextfreie Grammatiken

Beispiel 4.1 (Arithmetische Ausdrücke)

$$
\begin{array}{rcl}
\text{<Expr>} & \to & \text{<Term>} \\
\text{<Expr>} & \to & \text{<Expr>} + \text{<Term>} \\
\text{<Term>} & \to & \text{<Factor>} \\
\text{<Term>} & \to & \text{<Term>} * \text{<Factor>} \\
\text{<Factor>} & \to & a \\
\text{<Factor>} & \to & (\text{<Expr>})
\end{array}
$$

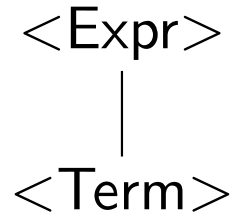
Eine (Links)Ableitung:

$$
\begin{aligned}
&\text{<Expr>} \to \text{<Term>} \to \text{<Term>} * \text{<Factor>} \\
&\to \text{<Factor>} * \text{<Factor>} \to a * \text{<Factor>} \\
&\to a * (\text{<Expr>}) \to a * (\text{<Expr>} + \text{<Term>}) \\
&\to a * (\text{<Term>} + \text{<Term>}) \to a * (\text{<Factor>} + \text{<Term>}) \\
&\to a * (a + \text{<Term>}) \to a * (a + \text{<Factor>}) \to a * (a + a)
\end{aligned}
$$

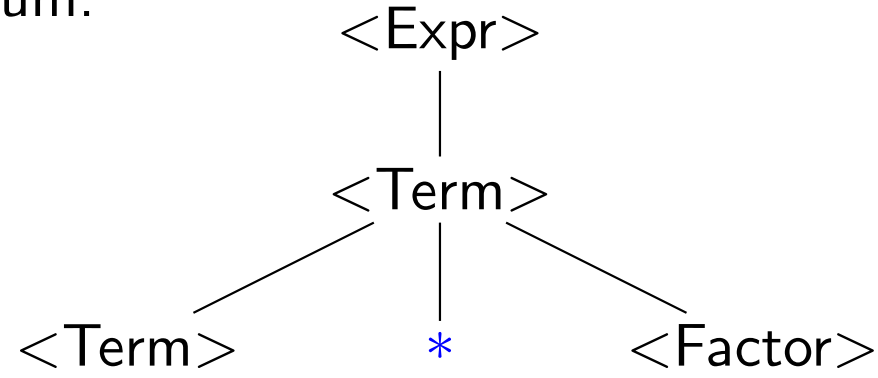
Der Syntaxbaum:

<Expr>

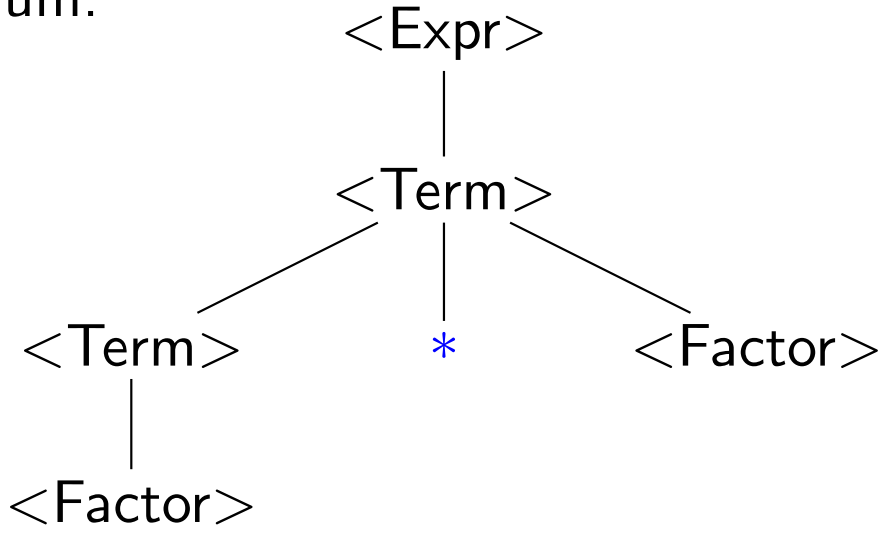
Der Syntaxbaum:



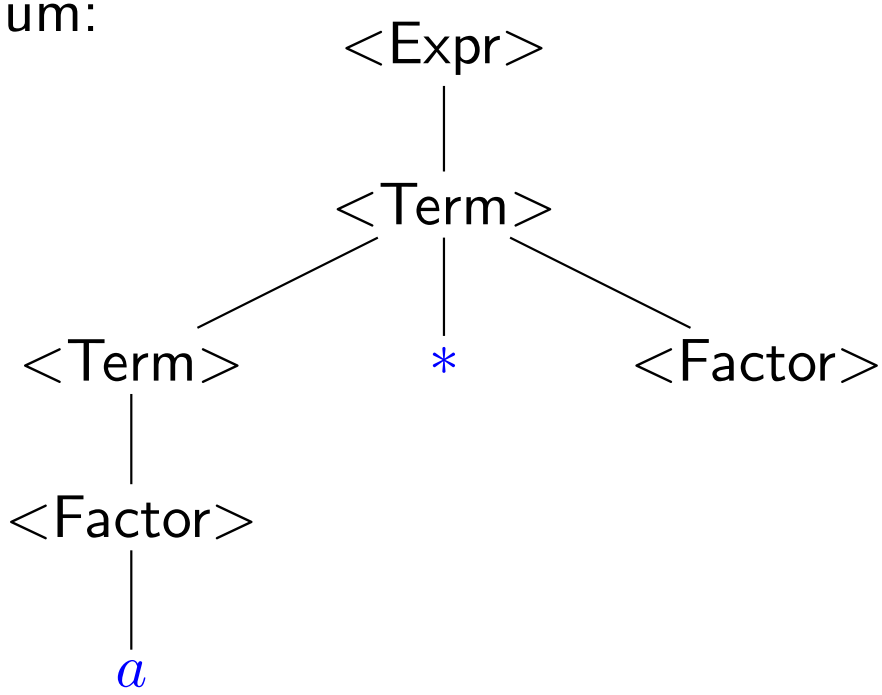
Der Syntaxbaum:



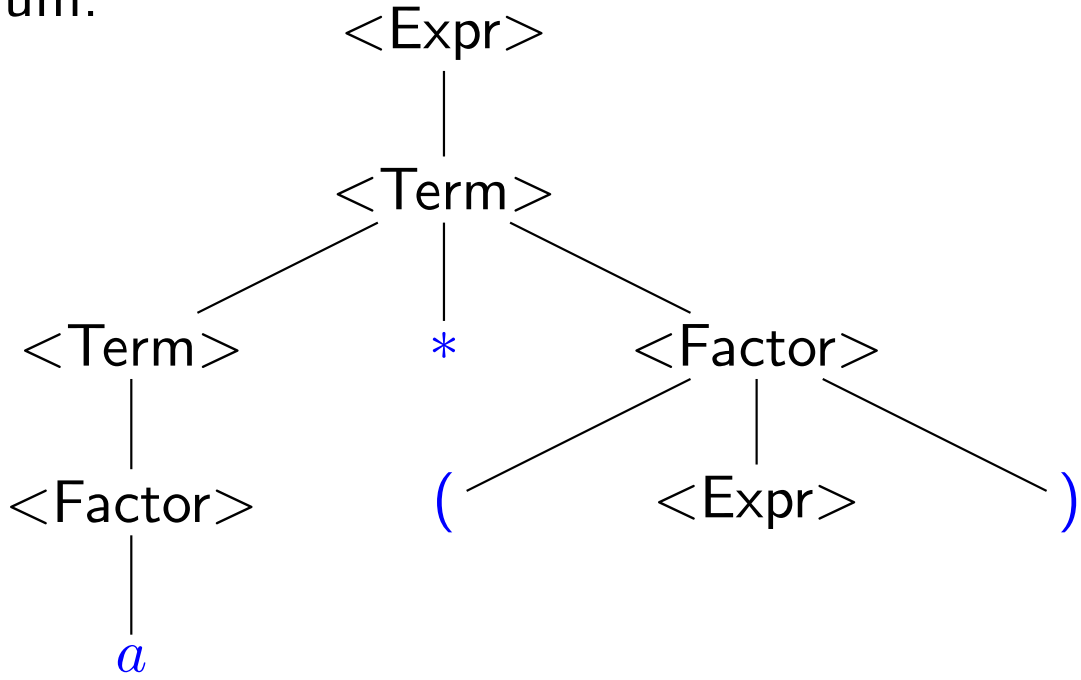
Der Syntaxbaum:



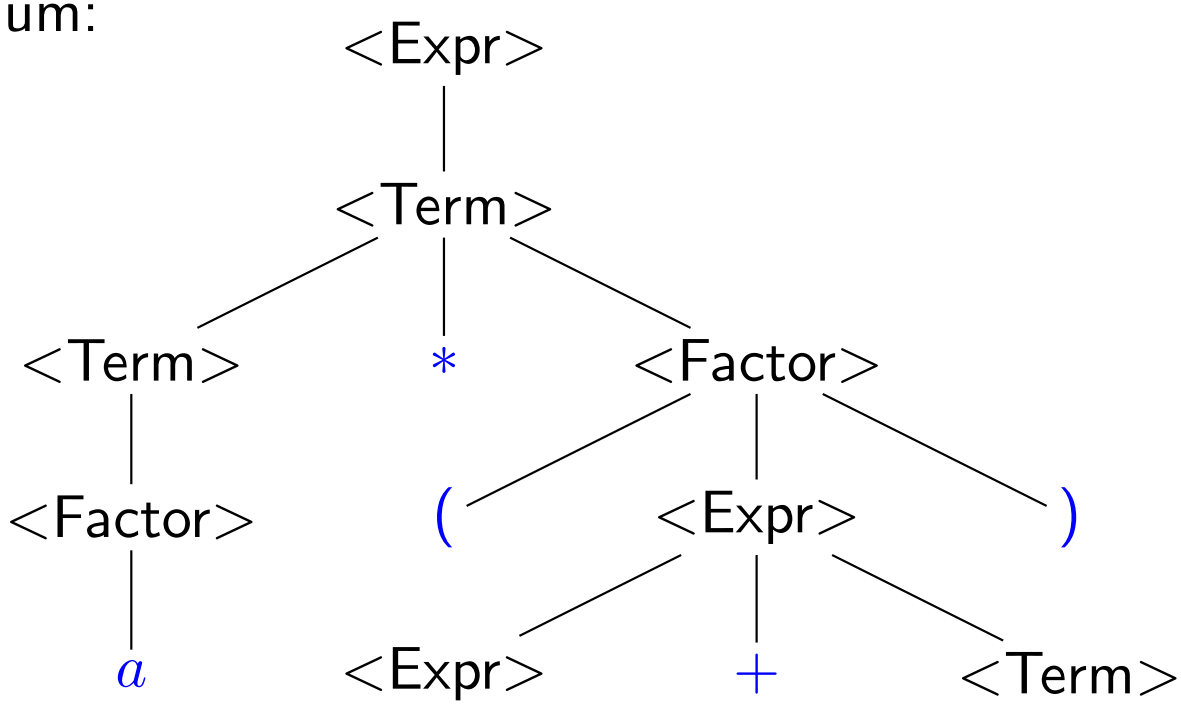
Der Syntaxbaum:



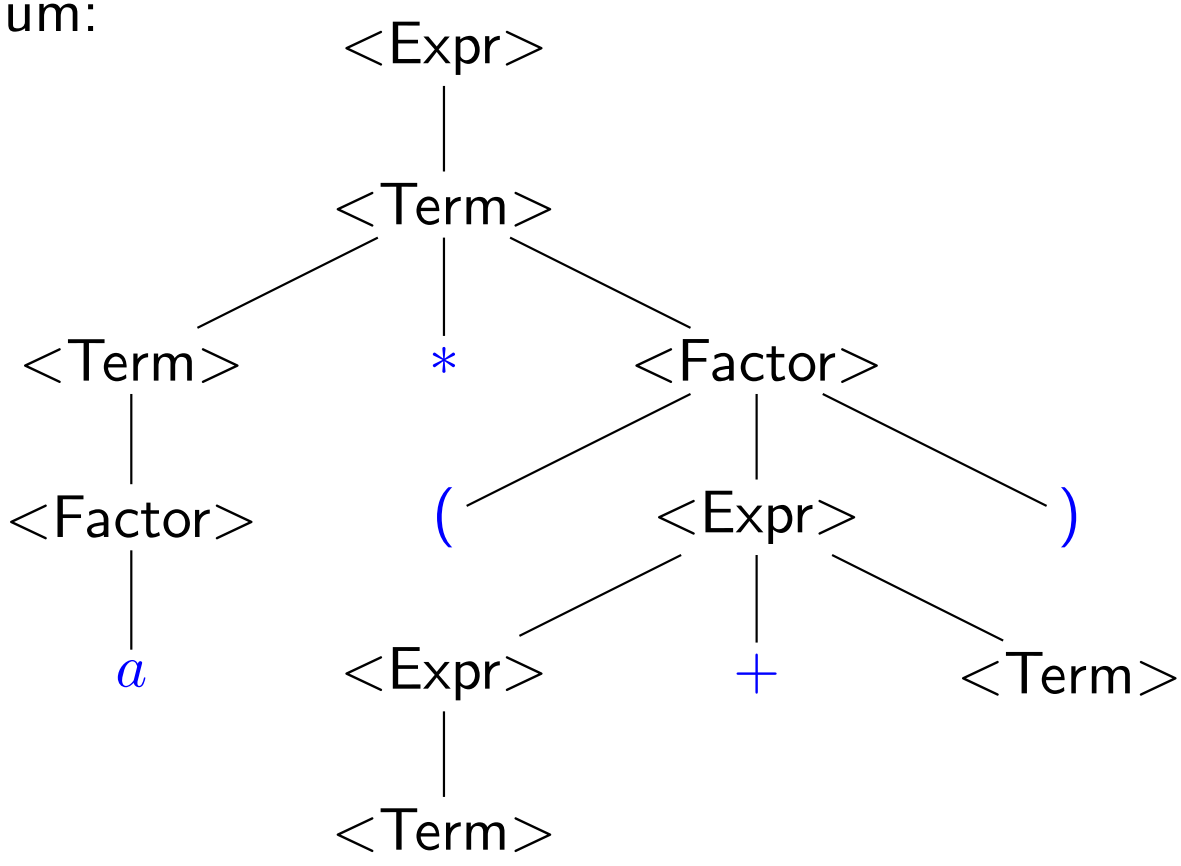
Der Syntaxbaum:



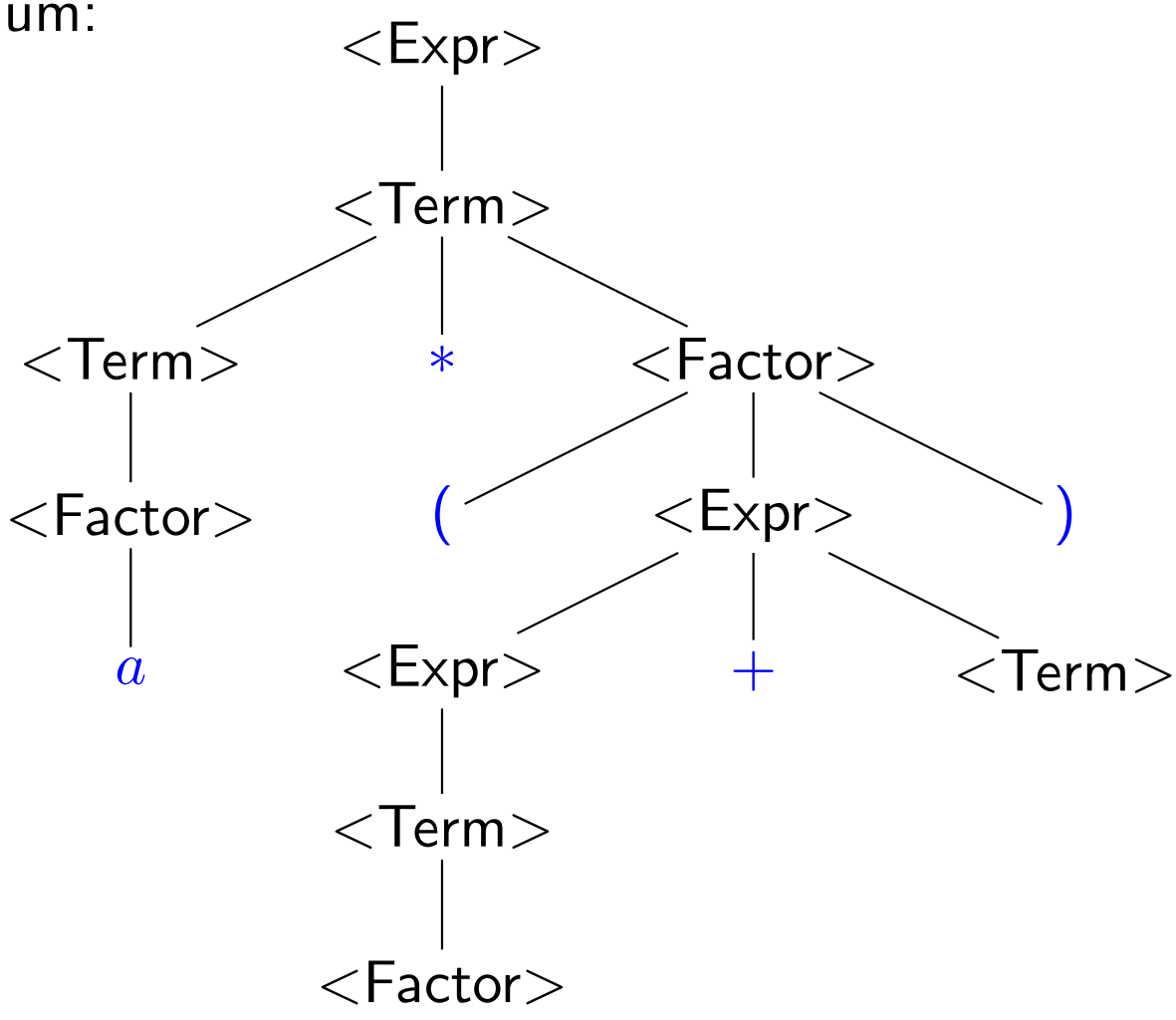
Der Syntaxbaum:



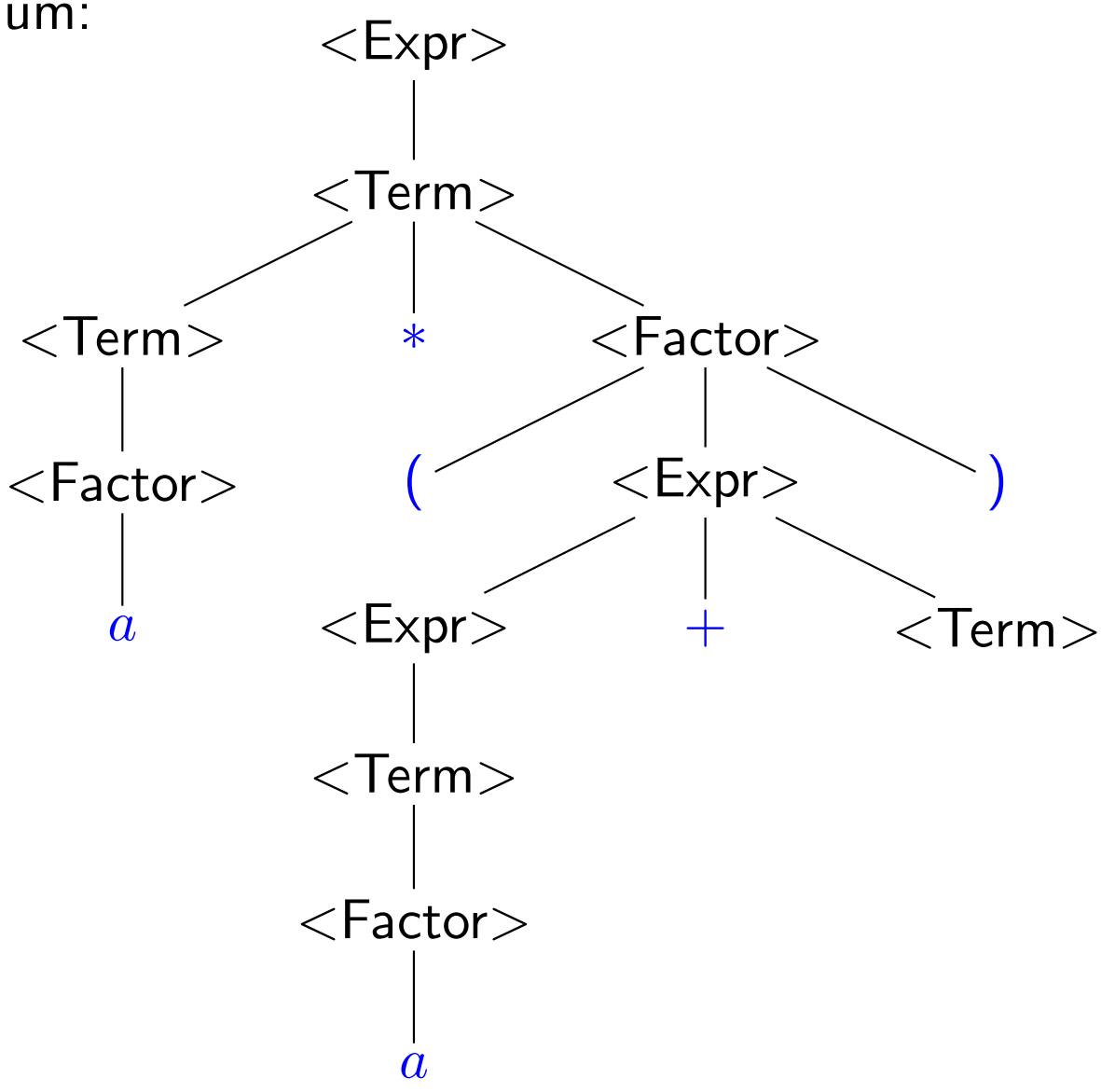
Der Syntaxbaum:



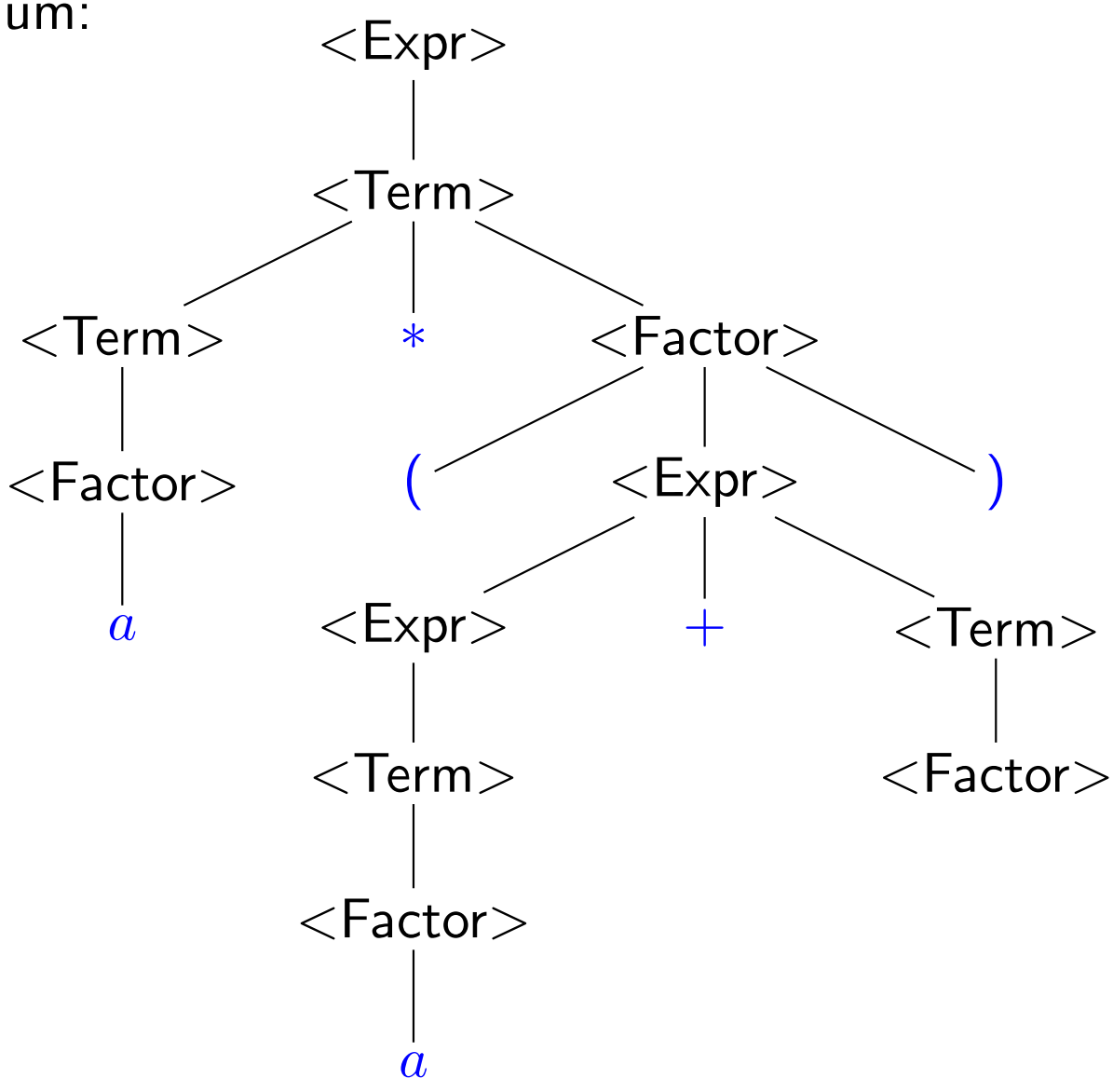
Der Syntaxbaum:



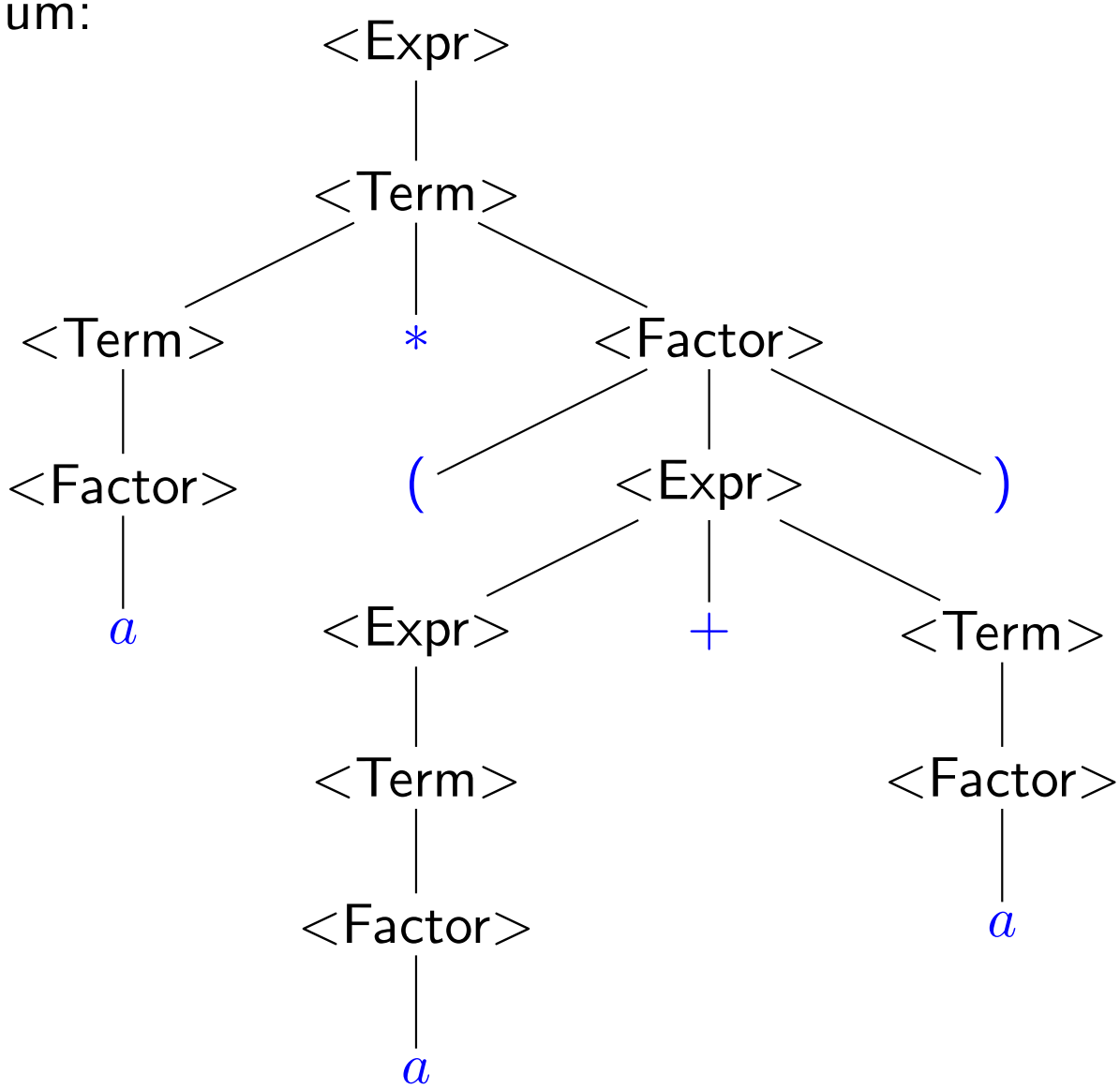
Der Syntaxbaum:



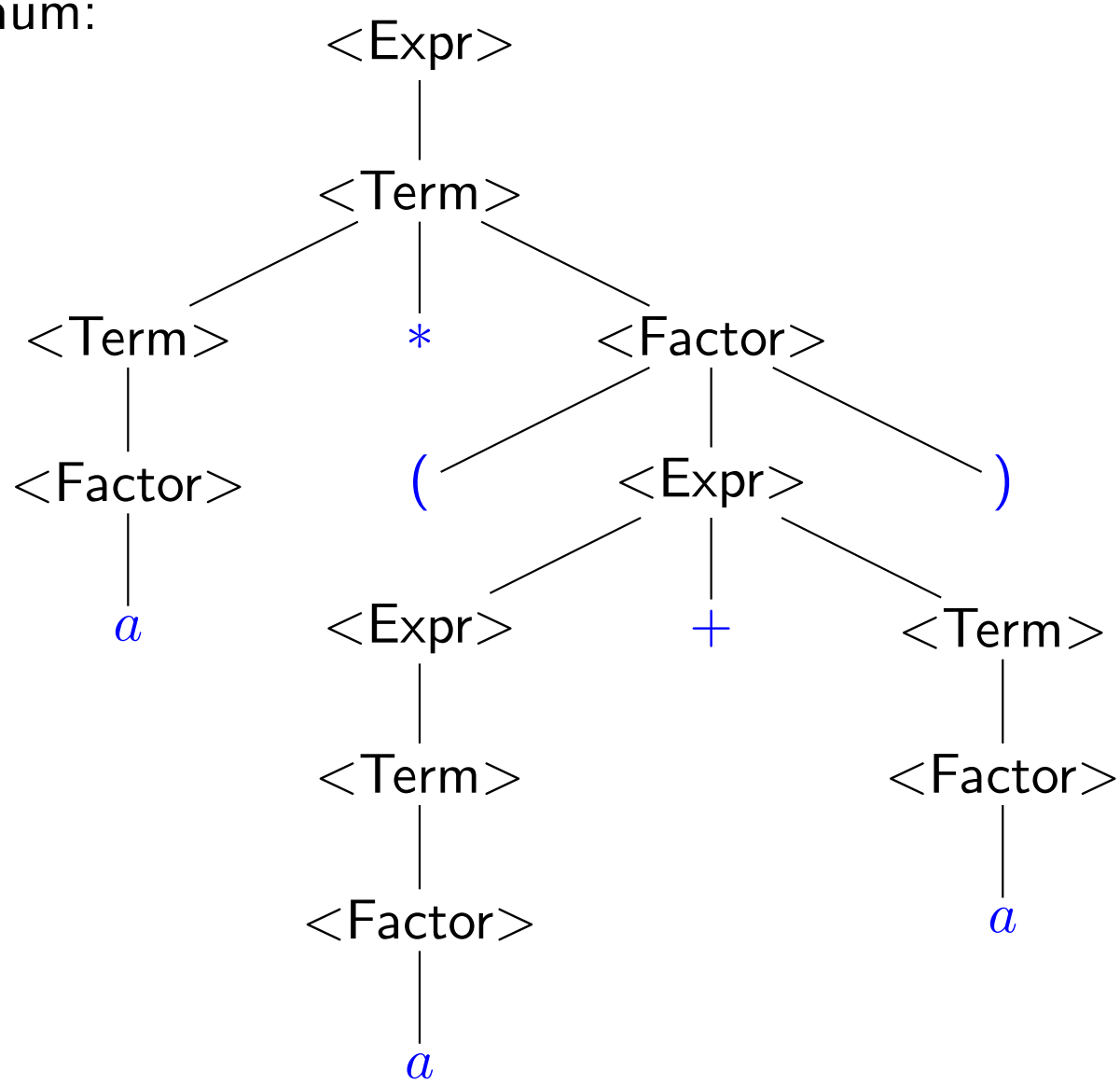
Der Syntaxbaum:



Der Syntaxbaum:



Der Syntaxbaum:



Die Blätter des Baums, von links nach rechts gelesen, ergeben das abgeleitete Wort

Bemerkungen

- Der vollständige Syntaxbaum enthält die gesamte Information über die Ableitung, bis auf die (irrelevante) Reihenfolge des Aufbaus.
- Kontextfreie Grammatiken dienen zur Spezifikation von Sprachen. Das Parsen ist:
  - Die Überprüfung, ob ein Wort von einer Grammatik abgeleitet werden kann, bzw
  - Die Erzeugung des Syntaxbaums (*parse tree*).

Parsen ist die Transformation eines Worts in einen Syntaxbaum.

Definition 4.2

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

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

Definition 4.2

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

- V ist eine endlichen Menge, die Nichtterminalzeichen (oder Variablen),
- Σ ist ein Alphabet, die Terminalzeichen, disjunkt von V,
- $P \subseteq V \times (V \cup \Sigma)^*$ eine endlichen Menge, die 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 $A \to \alpha$ statt $(A, \alpha) \in P$.
- Statt $A \to \alpha_1$, $A \to \alpha_2$, $A \to \alpha_3$ schreiben wir einfach

$$A \to \alpha_1 \mid \alpha_2 \mid \alpha_3$$

Beispiel 4.3 (Arithmetische Ausdrücke)

$$V = \{E, T, F\}$$

$$\Sigma = \{a, +, *, (,)\}$$

$$P = \left\{ \begin{array}{lcl} E & \rightarrow & T \mid E + T \\ T & \rightarrow & F \mid T * F \\ F & \rightarrow & a \mid (E) \end{array} \right\}$$

$$S = E$$

Definition 4.4

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

$$\alpha \rightarrow_G \beta$$

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

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

Definition 4.4

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

$$\alpha \rightarrow_G \beta$$

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

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

Beispiel:

$$a + T + a \quad \rightarrow_G \quad a + T * F + a$$

Definition 4.5 (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: $E \to_G^{11} a * (a + a)$ und daher $E \to_G^* a * (a + a)$.

Definition 4.5 (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: $E \to_G^{11} a * (a + a)$ und daher $E \to_G^* a * (a + a)$.

Wir nennen

$$\alpha_1 \to_G \alpha_2 \to_G \cdots \to_G \alpha_n$$

eine Linksableitung gdw in jedem Schritt das linkeste Nichtterminal in α_i ersetzt wird.

Definition 4.6 (Kontextfreie Sprache)

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

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

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

Definition 4.6 (Kontextfreie Sprache)

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

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

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

Abkürzungen:

CFG Kontextfreie Grammatik (*context-free grammar*)

CFL Kontextfreie Sprache (*context-free language*)

Definition 4.6 (Kontextfreie Sprache)

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

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

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

Abkürzungen:

CFG Kontextfreie Grammatik (*context-free grammar*)

CFL Kontextfreie Sprache (*context-free language*)

Konvention:

Ist G aus dem Kontext eindeutig ersichtlich, so schreibt man auch nur $\alpha \to \beta$ statt $\alpha \to_G \beta$.

Beispiel 4.7

Die nicht-reguläre Sprache $L = \{a^n b^n \mid n \in \mathbb{N}\}$ ist kontextfrei, da sie von der CFG

$$S \to aSb \mid \epsilon$$

erzeugt wird. Genauer: $L = L(G)$ wobei $G = (V, \Sigma, P, S)$ mit

$$\begin{aligned} V &= \{S\} \\ \Sigma &= \{a, b\} \\ P &= \{S \to aSb \mid \epsilon\} \end{aligned}$$

Beispiel 4.7

Die nicht-reguläre Sprache $L = \{a^n b^n \mid n \in \mathbb{N}\}$ ist kontextfrei, da sie von der CFG

$$S \to aSb \mid \epsilon$$

erzeugt wird. Genauer: $L = L(G)$ wobei $G = (V, \Sigma, P, S)$ mit

$$\begin{aligned} V &= \{S\} \\ \Sigma &= \{a, b\} \\ P &= \{S \to aSb \mid \epsilon\} \end{aligned}$$

Der unendliche Baum aller möglichen (Links-)Ableitungen:

$$\begin{array}{ccccccccc} S & \to & aSb & \to & a^2Sb^2 & \to & a^3Sb^3 & \to & \dots \\ & \searrow & & \searrow & & \searrow & & \searrow & \\ & & \epsilon & & ab & & a^2b^2 & & a^3b^3 \end{array}$$

Beispiel 4.8

Die nicht-reguläre Sprache der Palindrome ($w = w^R$) über $\{a, b\}$ wird von folgender CFG erzeugt:

$$S \to \epsilon \mid a \mid b \mid aSa \mid bSb$$

Beispiel 4.8

Die nicht-reguläre Sprache der Palindrome ($w = w^R$) über $\{a, b\}$ wird von folgender CFG erzeugt:

$$S \to \epsilon \mid a \mid b \mid aSa \mid bSb$$

Der Anfang des unendlichen Baums aller (Links-)Ableitungen (Achtung, dies ist *kein* Syntaxbaum!):

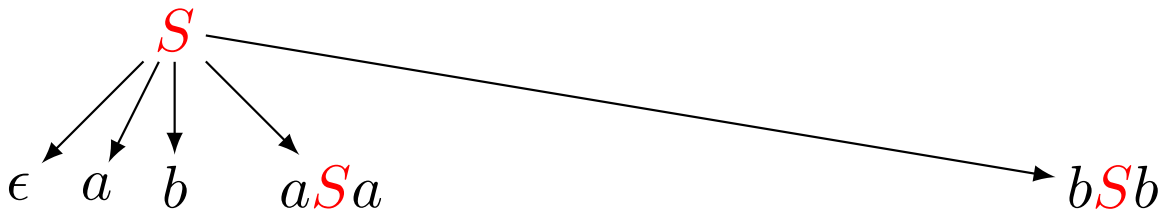
S

Beispiel 4.8

Die nicht-reguläre Sprache der Palindrome ($w = w^R$) über $\{a, b\}$ wird von folgender CFG erzeugt:

$$S \to \epsilon \mid a \mid b \mid aSa \mid bSb$$

Der Anfang des unendlichen Baums aller (Links-)Ableitungen (Achtung, dies ist *kein* Syntaxbaum!):

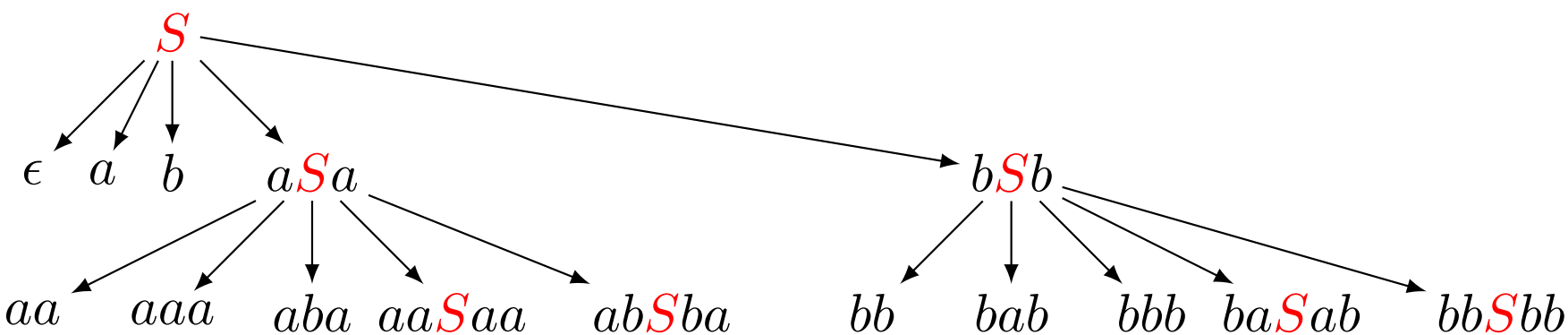


Beispiel 4.8

Die nicht-reguläre Sprache der Palindrome ($w = w^R$) über $\{a, b\}$ wird von folgender CFG erzeugt:

$$S \to \epsilon \mid a \mid b \mid aSa \mid bSb$$

Der Anfang des unendlichen Baums aller (Links-)Ableitungen (Achtung, dies ist *kein* Syntaxbaum!):



Lemma 4.9 (Dekompositionslemma)

$$\alpha_1\alpha_2 \to_G^n \beta$$
$$\Leftrightarrow$$
$$\exists \beta_1, \beta_2, n_1, n_2.\ \ \beta = \beta_1\beta_2 \ \wedge\ n = n_1 + n_2 \ \wedge\ \alpha_i \to_G^{n_i} \beta_i \ (i = 1, 2)$$

Beweis:

Übung! □

Definition 4.10

Eine CFG heißt rechts-linear gdw jede Produkion von der Form

$$A \to aB \quad \text{oder} \quad A \to \epsilon \quad \text{ist.}$$

Definition 4.10

Eine CFG heißt rechts-linear gdw jede Produkion von der Form

$$A \to aB \quad \text{oder} \quad A \to \epsilon \quad \text{ist.}$$

Eine CFG heißt links-linear gdw jede Produkion von der Form

$$A \to Ba \quad \text{oder} \quad A \to \epsilon \quad \text{ist.}$$

Definition 4.10

Eine CFG heißt rechts-linear gdw jede Produkion von der Form

$$A \to aB \quad \text{oder} \quad A \to \epsilon \quad \text{ist.}$$

Eine CFG heißt links-linear gdw jede Produkion von der Form

$$A \to Ba \quad \text{oder} \quad A \to \epsilon \quad \text{ist.}$$

Lemma 4.11

Die rechts-linearen und links-linearen Grammatiken erzeugen jeweils genau die regulären Sprachen.

Definition 4.10

Eine CFG heißt rechts-linear gdw jede Produkion von der Form

$$A \to aB \quad \text{oder} \quad A \to \epsilon \quad \text{ist.}$$

Eine CFG heißt links-linear gdw jede Produkion von der Form

$$A \to Ba \quad \text{oder} \quad A \to \epsilon \quad \text{ist.}$$

Lemma 4.11

Die rechts-linearen und links-linearen Grammatiken erzeugen jeweils genau die regulären Sprachen.

Beweis: Übung!

Korollar 4.12

Die regulären Sprachen sind eine echte Teilklasse der kontextfreien Sprachen.

4.2 Induktive Definitionen, Syntaxbäume und Ableitungen

1. Beispiel: Balancierte Klammern
2. Induktive Definitionen und Syntaxbäume allgemein
3. Äquivalenz von Ableitung, Syntaxbaum und induktiver Erzeugung

Beispiel 4.13

$$S \to \epsilon \mid [S] \mid SS$$

Um zu zeigen, dass diese Grammatik die Menge aller balancierten Klammerwörter in $\{[,]\}^*$ erzeugt, betrachten wir die Grammatik als induktive Definition einer Sprache $L_G(S)$:

Beispiel 4.13

$$S \to \epsilon \mid [S] \mid SS$$

Um zu zeigen, dass diese Grammatik die Menge aller balancierten Klammerwörter in $\{[,]\}^*$ erzeugt, betrachten wir die Grammatik als induktive Definition einer Sprache $L_G(S)$:

Beispiel 4.13

$$S \to \epsilon \mid [S] \mid SS$$

Um zu zeigen, dass diese Grammatik die Menge aller balancierten Klammerwörter in $\{[,]\}^*$ erzeugt, betrachten wir die Grammatik als induktive Definition einer Sprache $L_G(S)$:

$\epsilon \in L_G(S)$

Beispiel 4.13

$$S \to \epsilon \mid [S] \mid SS$$

Um zu zeigen, dass diese Grammatik die Menge aller balancierten Klammerwörter in $\{[,]\}^*$ erzeugt, betrachten wir die Grammatik als induktive Definition einer Sprache $L_G(S)$:

$$\epsilon \in L_G(S)$$

$$u \in L_G(S) \implies [u] \in L_G(S)$$

Beispiel 4.13

$$S \to \epsilon \mid [S] \mid SS$$

Um zu zeigen, dass diese Grammatik die Menge aller balancierten Klammerwörter in $\{[,]\}^*$ erzeugt, betrachten wir die Grammatik als induktive Definition einer Sprache $L_G(S)$:

$$\begin{array}{rcl} & & \epsilon \in L_G(S) \\ u \in L_G(S) & \Longrightarrow & [u] \in L_G(S) \\ u \in L_G(S) \wedge v \in L_G(S) & \Longrightarrow & uv \in L_G(S) \end{array}$$

Beispiel 4.13

$$S \to \epsilon \mid \texttt{[}S\texttt{]} \mid SS$$

Um zu zeigen, dass diese Grammatik die Menge aller balancierten Klammerwörter in $\{\texttt{[},\texttt{]}\}^*$ erzeugt, betrachten wir die Grammatik als induktive Definition einer Sprache $L_G(S)$:

$$\begin{array}{rcl} & & \epsilon \in L_G(S) \\ u \in L_G(S) & \Longrightarrow & \texttt{[}u\texttt{]} \in L_G(S) \\ u \in L_G(S) \wedge v \in L_G(S) & \Longrightarrow & uv \in L_G(S) \end{array}$$

Damit gilt zB:

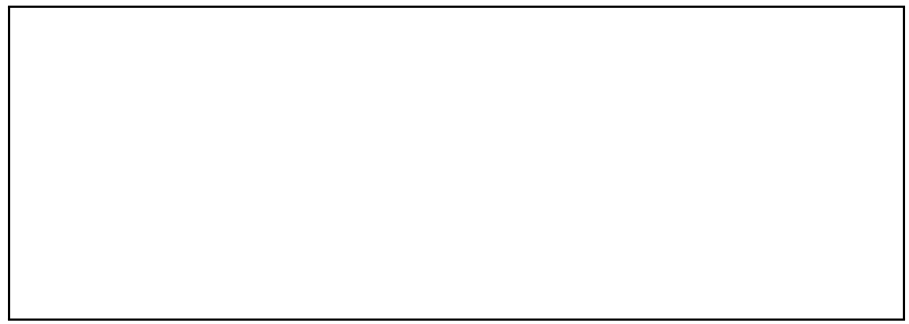
$\epsilon \in L_G(S) \implies \texttt{[]} \in L_G(S) \implies \texttt{[[]]} \in L_G(S)$
$\implies \texttt{[[]][]} \in L_G(S)$

Bemerkungen

- Die Produktionen ($\rightarrow$) erzeugen Wörter *top-down*, d.h. von einem Nichtterminal zu einem Wort hin.
- Die induktive Definition ($\Longrightarrow$) erzeugt Wörter *bottom-up*, d.h. sie setzt kleinere Wörter zu größeren zusammen.
- Die induktive Definition betrachtet nur Wörter aus Σ^*.

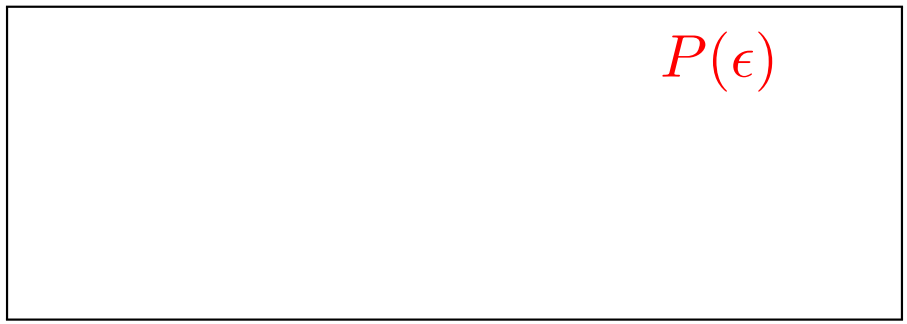
Zur induktiven Definition von $L_G(S)$ gehört ein Induktionsprinzip:

Um zu zeigen, dass für alle $u \in L_G(S)$ eine Eigenschaft $P(u)$ gilt, dh $\forall u \in L_G(S).\ P(u)$, zeige:



Zur induktiven Definition von $L_G(S)$ gehört ein Induktionsprinzip:

Um zu zeigen, dass für alle $u \in L_G(S)$ eine Eigenschaft $P(u)$ gilt, dh $\forall u \in L_G(S).\ P(u)$, zeige:



Zur induktiven Definition von $L_G(S)$ gehört ein Induktionsprinzip:

Um zu zeigen, dass für alle $u \in L_G(S)$ eine Eigenschaft $P(u)$ gilt, dh $\forall u \in L_G(S).\ P(u)$, zeige:

$$\begin{array}{rcl} & & P(\epsilon) \\ P(u) & \Longrightarrow & P(\texttt{[}u\texttt{]}) \end{array}$$

Zur induktiven Definition von $L_G(S)$ gehört ein Induktionsprinzip:

Um zu zeigen, dass für alle $u \in L_G(S)$ eine Eigenschaft $P(u)$ gilt, dh $\forall u \in L_G(S).\ P(u)$, zeige:

$$\begin{array}{rcl} & & P(\epsilon) \\ P(u) & \Longrightarrow & P(\texttt{[}u\texttt{]}) \\ P(u) \wedge P(v) & \Longrightarrow & P(uv) \end{array}$$

Zur induktiven Definition von $L_G(S)$ gehört ein Induktionsprinzip:

Um zu zeigen, dass für alle $u \in L_G(S)$ eine Eigenschaft $P(u)$ gilt, dh $\forall u \in L_G(S).\ P(u)$, zeige:

$$\begin{array}{rcl} & & P(\epsilon) \\ P(u) & \Longrightarrow & P(\texttt{[}u\texttt{]}) \\ P(u) \wedge P(v) & \Longrightarrow & P(uv) \end{array}$$

„Induktion über die Erzeugung von u“

Zur induktiven Definition von $L_G(S)$ gehört ein Induktionsprinzip:

Um zu zeigen, dass für alle $u \in L_G(S)$ eine Eigenschaft $P(u)$ gilt, dh $\forall u \in L_G(S).\ P(u)$, zeige:

$$\begin{array}{rcl} & & P(\epsilon) \\ P(u) & \Longrightarrow & P(\texttt{[}u\texttt{]}) \\ P(u) \wedge P(v) & \Longrightarrow & P(uv) \end{array}$$

„Induktion über die Erzeugung von u“

Lemma 4.14
Alle $u \in L_G(S)$ enthalten gleich viele [wie].

Zur induktiven Definition von $L_G(S)$ gehört ein Induktionsprinzip:

Um zu zeigen, dass für alle $u \in L_G(S)$ eine Eigenschaft $P(u)$ gilt, dh $\forall u \in L_G(S).\ P(u)$, zeige:

$$\begin{array}{rcl} & & P(\epsilon) \\ P(u) & \Longrightarrow & P(\texttt{[}u\texttt{]}) \\ P(u) \wedge P(v) & \Longrightarrow & P(uv) \end{array}$$

„Induktion über die Erzeugung von u“

Lemma 4.14

Alle $u \in L_G(S)$ enthalten gleich viele [wie].

Beweis:

Mit Induktion über die Erzeugung von u:

- ϵ enthält 0 [und].
- Enthält u gleich viele [wie], so auch $[u]$.
- Enthalten u und v gleich viele [wie], so auch uv.

□

Hinweis

Die Aussage

$$\forall x \in M.\ P(x)$$

ist eine Abkürzung für

$$\forall x.\ x \in M \implies P(x)$$

sind logisch äquivalent.

Der Allquantor wird dann oft weggelassen:

$$x \in M \implies P(x)$$

Definition 4.15

Präfix:

$$u \preceq w \quad :\Leftrightarrow \quad \exists v.\ uv = w$$

Anzahl der Vorkommen:

$$\#_a(w) := \text{Anzahl der Vorkommen von } a \text{ in } w$$

Für unser Beispiel führen wir zwei Abk. ein:

$$A(w) := \#_{[}(w) \qquad B(w) := \#_{]}(w)$$

Wir nennen $w \in \{[,]\}^*$ balanciert gdw

Definition 4.15

Präfix:

$$u \preceq w \quad :\Leftrightarrow \quad \exists v.\ uv = w$$

Anzahl der Vorkommen:

$$\#_a(w) := \text{Anzahl der Vorkommen von } a \text{ in } w$$

Für unser Beispiel führen wir zwei Abk. ein:

$$A(w) := \#_{[}(w) \qquad B(w) := \#_{]}(w)$$

Wir nennen $w \in \{[,]\}^*$ balanciert gdw

(1) $A(w) = B(w)$ und

Definition 4.15

Präfix:

$$u \preceq w \quad :\Leftrightarrow \quad \exists v.\ uv = w$$

Anzahl der Vorkommen:

$$\#_a(w) := \text{Anzahl der Vorkommen von } a \text{ in } w$$

Für unser Beispiel führen wir zwei Abk. ein:

$$A(w) := \#_{[}(w) \qquad B(w) := \#_{]}(w)$$

Wir nennen $w \in \{[,]\}^*$ balanciert gdw

(1) $A(w) = B(w)$ und

(2) für alle Präfixe u von w gilt $A(u) \geq B(u)$

Definition 4.15

Präfix:

$$u \preceq w \quad :\Leftrightarrow \quad \exists v.\ uv = w$$

Anzahl der Vorkommen:

$$\#_a(w) := \text{Anzahl der Vorkommen von } a \text{ in } w$$

Für unser Beispiel führen wir zwei Abk. ein:

$$A(w) := \#_{\texttt{[}}(w) \qquad B(w) := \#_{\texttt{]}}(w)$$

Wir nennen $w \in \{\texttt{[}, \texttt{]}\}^*$ balanciert gdw

(1) $A(w) = B(w)$ und

(2) für alle Präfixe u von w gilt $A(u) \geq B(u)$

- Balanciert: `[]`, `[[]]`, `[][]`, `[[[][]][]]`

Definition 4.15

Präfix:

$$u \preceq w \quad :\Leftrightarrow \quad \exists v.\ uv = w$$

Anzahl der Vorkommen:

$$\#_a(w) := \text{Anzahl der Vorkommen von } a \text{ in } w$$

Für unser Beispiel führen wir zwei Abk. ein:

$$A(w) := \#_{\texttt{[}}(w) \qquad B(w) := \#_{\texttt{]}}(w)$$

Wir nennen $w \in \{\texttt{[}, \texttt{]}\}^*$ balanciert gdw

(1) $A(w) = B(w)$ und

(2) für alle Präfixe u von w gilt $A(u) \geq B(u)$

- Balanciert: `[]`, `[[]]`, `[][]`, `[[[][]][]]`
- Nicht balanciert: `][`, `[]][[]`

Satz 4.16

Die Grammatik $S \to \epsilon \mid [S] \mid SS$ erzeugt genau die Menge der balancierten Wörter.

Satz 4.16

Die Grammatik $S \to \epsilon \mid [S] \mid SS$ erzeugt genau die Menge der balancierten Wörter.

Beweis:

$u \in L_G(S) \implies u$ balanciert:

Mit Ind. über die Erzeugung von u. (1) bewiesen, wir zeigen (2).

ϵ: $p \preceq \epsilon \implies p = \epsilon \implies A(p) = B(p)$

Satz 4.16

Die Grammatik $S \to \epsilon \mid [S] \mid SS$ erzeugt genau die Menge der balancierten Wörter.

Beweis:

$u \in L_G(S) \implies u$ balanciert:

Mit Ind. über die Erzeugung von u. (1) bewiesen, wir zeigen (2).

ϵ: $p \preceq \epsilon \implies p = \epsilon \implies A(p) = B(p)$

$[u]$: Sei $p \preceq [u]$

Fall $p = \epsilon$: $A(p) = B(p)$

Fall $p = [u]$: $A(p) = A(u) + 1 = B(u) + 1 = B(p)$

Fall $p = [q$ mit $q \preceq u$:

$A(p) = A(q) + 1 \geq B(q) + 1 > B(q) = B(p)$

Satz 4.16

Die Grammatik $S \to \epsilon \mid [S] \mid SS$ erzeugt genau die Menge der balancierten Wörter.

Beweis:

$u \in L_G(S) \implies u$ balanciert:

Mit Ind. über die Erzeugung von u. (1) bewiesen, wir zeigen (2).

ϵ: $p \preceq \epsilon \implies p = \epsilon \implies A(p) = B(p)$

$[u]$: Sei $p \preceq [u]$

Fall $p = \epsilon$: $A(p) = B(p)$

Fall $p = [u]$: $A(p) = A(u) + 1 = B(u) + 1 = B(p)$

Fall $p = [q$ mit $q \preceq u$:

$A(p) = A(q) + 1 \geq B(q) + 1 > B(q) = B(p)$

uv: Sei $p \preceq uv$.

Fall $p \preceq u$: $A(p) \geq B(p)$ mit IA für u

Fall $p = uq$ und $q \preceq v$:

$A(p) = A(u) + A(q) = B(u) + A(q) \geq B(u) + B(q) = B(uq) = B(p)$

Für $p = uv$, $A(p) = B(p)$

Beweis (Forts.)

u balanciert $\implies$ $u \in L_G(S)$:

Mit vollständiger Induktion über $n := |u|$. (dh $u = a_1 \dots a_n$)

IA: Jedes balancierte Wort $< n$ liegt in $L_G(S)$.

Zz: $u \in L_G(S)$.

Beweis (Forts.)

u balanciert $\implies$ $u \in L_G(S)$:

Mit vollständiger Induktion über $n := |u|$. (dh $u = a_1 \dots a_n$)

IA: Jedes balancierte Wort $< n$ liegt in $L_G(S)$.

Zz: $u \in L_G(S)$.

Falls $n = 0$, so $u = \epsilon \in L_G(S)$.

Beweis (Forts.)

u balanciert $\implies$ $u \in L_G(S)$:

Mit vollständiger Induktion über $n := |u|$. (dh $u = a_1 \dots a_n$)

IA: Jedes balancierte Wort $< n$ liegt in $L_G(S)$.

Zz: $u \in L_G(S)$.

Falls $n = 0$, so $u = \epsilon \in L_G(S)$.

Sei $n > 0$.

Betrachte Werteverlauf von $h(w) := A(w) - B(w)$:

$h(\epsilon)$, $h(a_1)$, $h(a_1a_2)$,. . . , $h(a_1 \dots a_n)$ (alle ≥ 0!).

Insb. gilt $a_1 =$ [und $a_n =$]. Wir betrachten zwei Fälle:

Beweis (Forts.)

u balanciert $\implies$ $u \in L_G(S)$:

Mit vollständiger Induktion über $n := |u|$. (dh $u = a_1 \dots a_n$)

IA: Jedes balancierte Wort $< n$ liegt in $L_G(S)$.

Zz: $u \in L_G(S)$.

Falls $n = 0$, so $u = \epsilon \in L_G(S)$.

Sei $n > 0$.

Betrachte Werteverlauf von $h(w) := A(w) - B(w)$:

$h(\epsilon)$, $h(a_1)$, $h(a_1 a_2)$,..., $h(a_1 \dots a_n)$ (alle ≥ 0!).

Insb. gilt $a_1 =$ [und $a_n =$]. Wir betrachten zwei Fälle:

- Es gibt nur die Nullstellen $h(\epsilon)$ und $h(a_1 \dots a_n)$.
  Dann ist $v := a_2 \dots a_{n-1}$ balanciert:
  (1) $A(v) = A(\,[v]\,) - 1 = B(\,[v]\,) - 1 = B(v)$
  (2) $p \preceq v$: $h(\,[p) = A(\,[p) - B(\,[p) > 0 \implies$
  $A(p) = A(\,[p) - 1 > B(\,[p) - 1 = B(p) - 1 \implies$
  $A(p) \geq B(p)$
  $\implies$ $v \in L_G(S)$ (nach IA) $\implies$ $u = [v] \in L_G(S)$

Beweis (Forts.):

- Es gibt noch eine Nullstelle $h(a_1 \dots a_k)$.
  Sei $u_1 := a_1 \dots a_k$, $u_2 := a_{k+1} \dots a_n$
  Dann sind u_1 und u_2 balanciert:
  (1) $A(u_1) = B(u_1)$ da $h(u_1) = 0$;
  $A(u_2) = A(u) - A(u_1) = B(u) - B(u_1) = B(u_2)$
  (2) $p \preceq u_1 \implies p \preceq u \implies A(p) \geq B(p)$
  $p \preceq u_2$: $A(p) = A(u_1 p) - A(u_1) \geq B(u_1 p) - B(u_1) = B(p)$
  $\implies u_1, u_2 \in L_G(S)$ (nach IA, da $|u_i| < n$)
  $\implies u = u_1 u_2 \in L_G(S)$

□

Moral:

- „$w \in L_G(S) \implies P(w)$“ beweist man immer schematisch mit Induktion über die Erzeugung von w.
- „$P(w) \implies w \in L_G(S)$“ beweist man oft mit Induktion über $|w|$. Erfordert meist Kreativität.

Wir übertragen die Idee der induktiven Erzeugung auf beliebige CFGs $G = (V, \Sigma, P, S)$. Sei $V = \{A_1, \ldots, A_k\}$. Damit ist jede Produktion von der Form

$$A_i \to w_0 A_{i_1} w_1 \ldots w_{n-1} A_{i_n} w_n$$

Wir übertragen die Idee der induktiven Erzeugung auf beliebige CFGs $G = (V, \Sigma, P, S)$. Sei $V = \{A_1, \ldots, A_k\}$. Damit ist jede Produktion von der Form

$$A_i \to w_0 A_{i_1} w_1 \ldots w_{n-1} A_{i_n} w_n$$

Man kann G als eine (simultane) induktive Definition von Sprachen $L_G(A_i)$ für all $A_i \in V$ sehen. Jede Produktion korrespondiert zu einer Erzeugungsregel

$$u_1 \in L_G(A_{i_1}) \wedge \cdots \wedge u_n \in L_G(A_{i_n}) \implies w_0 u_1 w_1 \ldots u_n w_n \in L_G(A_i)$$

Wir übertragen die Idee der induktiven Erzeugung auf beliebige CFGs $G = (V, \Sigma, P, S)$. Sei $V = \{A_1, \ldots, A_k\}$. Damit ist jede Produktion von der Form

$$A_i \to w_0 A_{i_1} w_1 \ldots w_{n-1} A_{i_n} w_n$$

Man kann G als eine (simultane) induktive Definition von Sprachen $L_G(A_i)$ für all $A_i \in V$ sehen. Jede Produktion korrespondiert zu einer Erzeugungsregel

$$u_1 \in L_G(A_{i_1}) \wedge \cdots \wedge u_n \in L_G(A_{i_n}) \implies w_0 u_1 w_1 \ldots u_n w_n \in L_G(A_i)$$

Aussagen der Form

$$(u \in L_G(A_1) \implies P_1(u)) \wedge \cdots \wedge (u \in L_G(A_k) \implies P_k(u))$$

werden durch simultane Induktion über die Erzeugung von u bewiesen, indem man für jede Produktion zeigt:

$$P_{i_1}(u_1) \wedge \cdots \wedge P_{i_n}(u_n) \implies P_i(w_0 u_1 w_1 \ldots w_{n-1} u_n w_n)$$

Beispiel 4.17

Grammatik:

$$A \to \epsilon \mid aB \qquad B \to Aa$$

Induktive Definition:

- $\epsilon \in L_G(A)$
- $w \in L_G(B) \implies aw \in L_G(A)$
- $w \in L_G(A) \implies wa \in L_G(B)$

Beispiel 4.17

Grammatik:

$$A \to \epsilon \mid aB \qquad B \to Aa$$

Induktive Definition:

- $\epsilon \in L_G(A)$
- $w \in L_G(B) \implies aw \in L_G(A)$
- $w \in L_G(A) \implies wa \in L_G(B)$

Beim induktiven Beweis von

$$\begin{aligned}&(w \in L_G(A) \implies \#_a(w) \text{ ist gerade}) \wedge \\ &(w \in L_G(B) \implies \#_a(w) \text{ ist ungerade})\end{aligned}$$

muss man zeigen

- $\#_a(\epsilon)$ ist gerade
- $\#_a(w)$ ist ungerade $\implies$ $\#_a(aw)$ ist gerade
- $\#_a(w)$ ist gerade $\implies$ $\#_a(wa)$ ist ungerade

Definition 4.18 (Syntaxbaum)

Ein Syntaxbaum für eine Ableitung mit einer Grammatik $G = (V, \Sigma, P, S)$ ist ein Baum, so dass

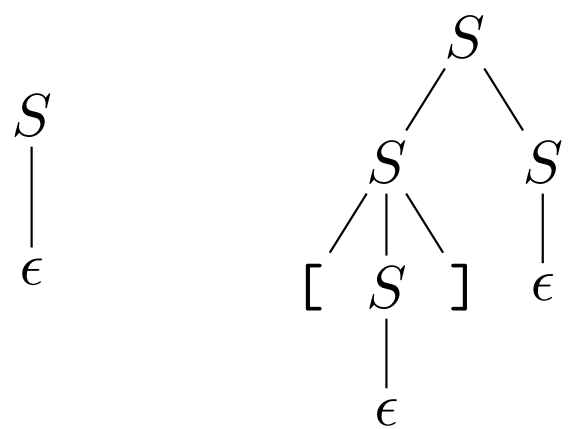
- jedes Blatt mit einem Zeichen aus $\Sigma \cup \{\epsilon\}$ beschriftet ist,
- jeder innere Knoten mit einem $A \in V$ beschriftet ist, und falls die Nachfolger (von links nach rechts) mit $X_1, \ldots, X_n \in V \cup \Sigma \cup \{\epsilon\}$ beschriftet sind, dann ist $A \to X_1 \ldots X_n$ eine Produktion in P,
- ein Blatt ϵ der einzige Nachfolger seines Vorgängers ist.

Definition 4.18 (Syntaxbaum)

Ein Syntaxbaum für eine Ableitung mit einer Grammatik $G = (V, \Sigma, P, S)$ ist ein Baum, so dass

- jedes Blatt mit einem Zeichen aus $\Sigma \cup \{\epsilon\}$ beschriftet ist,
- jeder innere Knoten mit einem $A \in V$ beschriftet ist, und falls die Nachfolger (von links nach rechts) mit $X_1, \ldots, X_n \in V \cup \Sigma \cup \{\epsilon\}$ beschriftet sind, dann ist $A \to X_1 \ldots X_n$ eine Produktion in P,
- ein Blatt ϵ der einzige Nachfolger seines Vorgängers ist.

Beispiel 4.19 (Syntaxbäume für $S \to \epsilon \mid [S] \mid SS$)

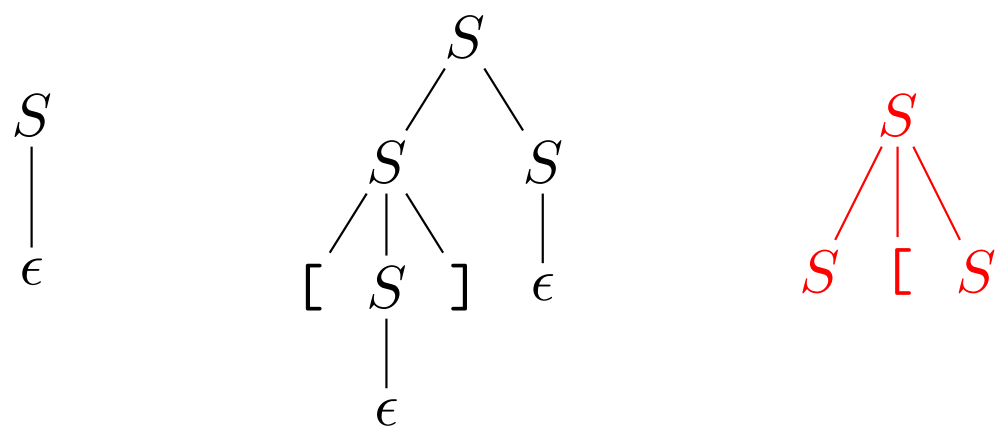


Definition 4.18 (Syntaxbaum)

Ein Syntaxbaum für eine Ableitung mit einer Grammatik $G = (V, \Sigma, P, S)$ ist ein Baum, so dass

- jedes Blatt mit einem Zeichen aus $\Sigma \cup \{\epsilon\}$ beschriftet ist,
- jeder innere Knoten mit einem $A \in V$ beschriftet ist, und falls die Nachfolger (von links nach rechts) mit $X_1, \ldots, X_n \in V \cup \Sigma \cup \{\epsilon\}$ beschriftet sind, dann ist $A \to X_1 \ldots X_n$ eine Produktion in P,
- ein Blatt ϵ der einzige Nachfolger seines Vorgängers ist.

Beispiel 4.19 (Syntaxbäume für $S \to \epsilon \mid [S] \mid SS$)



Satz 4.20

Für eine CFG und ein $w \in \Sigma^*$ sind folgende Bedingungen äquivalent:

1. $A \to_G^* w$
2. $w \in L_G(A)$ *(gemäß der induktiven Definition)*
3. *Es gibt einen Syntaxbaum mit Wurzel A, dessen Rand (Blätter von links nach rechts gelesen) das Wort w ist.*

Beweis:

Skizze (Details: [HMU]) $1 \Rightarrow 2$: Sei $A \to_G^n w$. Wir konstruieren dazu einen Beweis für $w \in L_G(A)$ mit Induktion über n. Die Ableitung muss von folgender Form sein:

$$A \to_G \; w_0 A_1 w_1 \ldots A_m w_m \; \to_G^{n-1} \; w$$

Nach dem Dekompositionslemma muss es u_i und $n_i \leq n-1$ geben mit $A_i \to_G^{n_i} u_i$ und $w = w_0 u_1 w_1 \ldots u_m w_m$. Nach IA (da $n_i < n$) gilt $u_i \in L_G(A_i)$ und daher (mit einem weiteren Erzeugungsschritt) auch $w \in L_G(A)$.

$2 \Rightarrow 3$: Parallel zur induktiven Erzeugung von $w \in L_G(A)$ kann man einen Syntaxbaum mit Rand w erzeugen. Formal: Induktion über die Erzeugung von w.
$3 \Rightarrow 1$: Jeder Baum lässt sich in eine Ableitung, sogar eine Linksableitung(!) umformen. Induktion über die Höhe des Baums: Transformiere die Unterbäume t_i mit Beschriftung A_i in Ableitungen $A_i \to_G^* u_i$ und füge diese mit dem Dekompositionslemma zu einer grossen Ableitung $A \to_G w_0 A_1 w_1 \dots A_m w_m \to_G^* w_0 u_1 w_1 \dots u_m w_m$ zusammen.

Definition 4.21

- Eine CFG G heißt mehrdeutig gdw es ein $w \in L(G)$ gibt, das zwei verschiedene Syntaxbäume hat, also zwei verschiedene Syntaxbäume mit Wurzel S und Rand w.
- Eine CFL L heißt inhärent mehrdeutig gdw jede CFG G mit $L(G) = L$ mehrdeutig ist.

Definition 4.21

- Eine CFG G heißt mehrdeutig gdw es ein $w \in L(G)$ gibt, das zwei verschiedene Syntaxbäume hat, also zwei verschiedene Syntaxbäume mit Wurzel S und Rand w.
- Eine CFL L heißt inhärent mehrdeutig gdw jede CFG G mit $L(G) = L$ mehrdeutig ist.

Beispiel 4.22

Die Grammatik $E \to E + E \mid E * E \mid (E) \mid a$ ist mehrdeutig — betrachte $a + a * a$. Die erzeugte Sprache ist aber nicht inhärent mehrdeutig (siehe Beispiel Arithmetische Ausdrücke).

Definition 4.21

- Eine CFG G heißt mehrdeutig gdw es ein $w \in L(G)$ gibt, das zwei verschiedene Syntaxbäume hat, also zwei verschiedene Syntaxbäume mit Wurzel S und Rand w.
- Eine CFL L heißt inhärent mehrdeutig gdw jede CFG G mit $L(G) = L$ mehrdeutig ist.

Beispiel 4.22

Die Grammatik $E \to E + E \mid E * E \mid (E) \mid a$ ist mehrdeutig — betrachte $a + a * a$. Die erzeugte Sprache ist aber nicht inhärent mehrdeutig (siehe Beispiel Arithmetische Ausdrücke).

Satz 4.23

Die Sprache $\{a^i b^j c^k \mid i = j \vee j = k\}$ *ist inhärent mehrdeutig.*

4.3 Die Chomsky-Normalform

Definition 4.24

Eine kontextfreie Grammatik G ist in Chomsky-Normalform gdw alle Produktionen eine der Formen

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

haben.

4.3 Die Chomsky-Normalform

Definition 4.24

Eine kontextfreie Grammatik G ist in Chomsky-Normalform gdw alle Produktionen eine der Formen

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

haben.

Wir konstruieren und beweisen nun schrittweise:

Satz 4.25

Zu jeder CFG G kann man eine CFG G' in Chomsky-Normalform konstruieren mit $L(G') = L(G) \setminus \{\epsilon\}$.

4.3 Die Chomsky-Normalform

Definition 4.24

Eine kontextfreie Grammatik G ist in Chomsky-Normalform gdw alle Produktionen eine der Formen

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

haben.

Wir konstruieren und beweisen nun schrittweise:

Satz 4.25

Zu jeder CFG G kann man eine CFG G' in Chomsky-Normalform konstruieren mit $L(G') = L(G) \setminus \{\epsilon\}$.

Wer auf $\epsilon \in L(G')$ nicht verzichten möchte:
Füge am Ende wieder $S \to \epsilon$ hinzu.

Wir nennen $A \to \epsilon$ eine ϵ-Produktion.

Wir nennen $A \to \epsilon$ eine ϵ-Produktion.

Lemma 4.26

Zu jeder CFG $G = (V, \Sigma, P, S)$ kann man eine CFG G' konstruieren, die keine ϵ-Produktionen enthält, so dass gilt $L(G') = L(G) \setminus \{\epsilon\}$

Wir nennen $A \to \epsilon$ eine ϵ-Produktion.

Lemma 4.26

Zu jeder CFG $G = (V, \Sigma, P, S)$ kann man eine CFG G' konstruieren, die keine ϵ-Produktionen enthält, so dass gilt $L(G') = L(G) \setminus \{\epsilon\}$

Beweis:

Wir erweitern P induktiv zu eine Obermenge $\hat{P}$:

1. Jede Produktion aus P ist in $\hat{P}$
2. Sind $B \to \epsilon$ und $A \to \alpha B \beta$ in $\hat{P}$, so füge auch $A \to \alpha\beta$ hinzu.

Offensichtlich gilt $L(\hat{G}) = L(G)$: Jede neue Produktion kann von 2 alten simuliert werden.

Wir nennen $A \to \epsilon$ eine ϵ-Produktion.

Lemma 4.26

Zu jeder CFG $G = (V, \Sigma, P, S)$ kann man eine CFG G' konstruieren, die keine ϵ-Produktionen enthält, so dass gilt $L(G') = L(G) \setminus \{\epsilon\}$

Beweis:

Wir erweitern P induktiv zu eine Obermenge $\hat{P}$:

1. Jede Produktion aus P ist in $\hat{P}$
2. Sind $B \to \epsilon$ und $A \to \alpha B \beta$ in $\hat{P}$, so füge auch $A \to \alpha\beta$ hinzu.

Offensichtlich gilt $L(\hat{G}) = L(G)$: Jede neue Produktion kann von 2 alten simuliert werden.

Wir definieren G' als $\hat{G}$ ohne die ϵ-Produktionen.
Denn diese sind nun überflüssig:

Beweis (Forts.):

Sei $S \to^*_{\hat{G}} w$, $w \neq \epsilon$, eine Ableitung minimaler Länge.
Käme darin $B \to \epsilon$ vor, also

$$S \to^*_{\hat{G}} \gamma B \delta \to_{\hat{G}} \gamma\delta \to^*_{\hat{G}} w$$

so wäre γ oder δ nichtleer, dh B muss durch eine Produktion $A \to \alpha B \beta$ eingeführt worden sein:

$$S \to^k_{\hat{G}} \alpha' A \beta' \to_{\hat{G}} \alpha'\alpha B \beta\beta' \to^m_{\hat{G}} \gamma B \delta \to_{\hat{G}} \gamma\delta \to^n_{\hat{G}} w$$

Beweis (Forts.):

Sei $S \to^*_{\hat{G}} w$, $w \neq \epsilon$, eine Ableitung minimaler Länge.
Käme darin $B \to \epsilon$ vor, also

$$S \to^*_{\hat{G}} \gamma B \delta \to_{\hat{G}} \gamma\delta \to^*_{\hat{G}} w$$

so wäre γ oder δ nichtleer, dh B muss durch eine Produktion $A \to \alpha B \beta$ eingeführt worden sein:

$$S \to^k_{\hat{G}} \alpha' A \beta' \to_{\hat{G}} \alpha'\alpha B \beta\beta' \to^m_{\hat{G}} \gamma B \delta \to_{\hat{G}} \gamma\delta \to^n_{\hat{G}} w$$

Dann wäre aber folgende Ableitung mit $(A \to \alpha\beta) \in \hat{P}$ kürzer:

$$S \to^k_{\hat{G}} \alpha' A \beta' \to_{\hat{G}} \alpha'\alpha\beta\beta' \to^m_{\hat{G}} \gamma\delta \to^n_{\hat{G}} w$$

Beweis (Forts.):

Sei $S \to^*_{\hat{G}} w$, $w \neq \epsilon$, eine Ableitung minimaler Länge.
Käme darin $B \to \epsilon$ vor, also

$$S \to^*_{\hat{G}} \gamma B \delta \to_{\hat{G}} \gamma\delta \to^*_{\hat{G}} w$$

so wäre γ oder δ nichtleer, dh B muss durch eine Produktion $A \to \alpha B \beta$ eingeführt worden sein:

$$S \to^k_{\hat{G}} \alpha' A \beta' \to_{\hat{G}} \alpha'\alpha B \beta\beta' \to^m_{\hat{G}} \gamma B \delta \to_{\hat{G}} \gamma\delta \to^n_{\hat{G}} w$$

Dann wäre aber folgende Ableitung mit $(A \to \alpha\beta) \in \hat{P}$ kürzer:

$$S \to^k_{\hat{G}} \alpha' A \beta' \to_{\hat{G}} \alpha'\alpha\beta\beta' \to^m_{\hat{G}} \gamma\delta \to^n_{\hat{G}} w$$

Also kommen in Ableitungen minimaler Länge keine ϵ-Produktionen vor. Letztere sind also überflüssig. □

Beispiel 4.27

$$S \to AB, \quad A \to aAA \mid B, \qquad B \to bBS \mid \epsilon$$

Wir nennen $A \to B$ eine Kettenproduktion.

Wir nennen $A \to B$ eine Kettenproduktion.

Lemma 4.28

Zu jeder CFG $G = (V, \Sigma, P, S)$ kann man eine CFG G' konstruieren, die keine Kettenproduktionen enthält, so dass gilt $L(G') = L(G)$.

Wir nennen $A \to B$ eine Kettenproduktion.

Lemma 4.28

Zu jeder CFG $G = (V, \Sigma, P, S)$ kann man eine CFG G' konstruieren, die keine Kettenproduktionen enthält, so dass gilt $L(G') = L(G)$.

Beweis:

Wir erweitern P induktiv zu eine Obermenge $\hat{P}$:

1. Jede Produktion aus P ist in $\hat{P}$
2. Sind $A \to B$ und $B \to \alpha$ in $\hat{P}$ mit $\alpha \neq A$, so füge auch $A \to \alpha$ hinzu.

Das Ergebnis G' ist $\hat{G}$ ohne die (nun überflüssigen) Kettenproduktionen.
Rest des Beweises analog zur Elimination von ϵ-Produktionen. □

Beispiel 4.29

$$A \to a \mid B, \qquad B \to b \mid C \mid aBB, \qquad C \to A \mid AB$$

Konstruktion einer Chomsky-Normalform

Eingabe: Eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$

Konstruktion einer Chomsky-Normalform

Eingabe: Eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$

1. Füge für jedes $a \in \Sigma$, das in einer rechten Seite der Länge ≥ 2 vorkommt, ein neues Nichtterminal A_a zu V hinzu, ersetze a in allen rechten Seiten der Länge ≥ 2 durch A_a, und füge $A_a \to a$ zu P hinzu.
2. Ersetze jede Produktion der Form

   $$A \to B_1 B_2 \cdots B_k \ (k \geq 3)$$

   durch

   $$A \to B_1 C_2,\ C_2 \to B_2 C_3,\ \ldots,\ C_{k-1} \to B_{k-1} B_k$$

   wobei $C_2, \ldots, C_{k-1}$ neue Nichtterminale sind.
3. Eliminiere alle ϵ-Produktionen.
4. Eliminiere alle Kettenproduktionen.

Aufgabe:

- Zeige: Mit diesem Algorithmus ist das Ergebnis höchstens quadratisch größer als G.
- Zeige: die Reihenfolge der Schritte ist wichtig!
  - Finde eine andere Reihenfolge, die inkorrekt ist: Das resultierende Verfahren ergibt nicht immer eine Grammatik in CNF.
  - Finde eine andere Reihenfolge, die zwar zu einem korrekten Verfahren führt, aber eine Grammatik in CNF produzieren kann, die exponentiell größer als G ist.

Definition 4.30

Eine CFG ist in Greibach-Normalform (benannt nach Sheila Greibach, UCLA), falls jede Produktion von der Form

$$A \to aA_1 \dots A_n$$

ist.

Definition 4.30

Eine CFG ist in Greibach-Normalform (benannt nach Sheila Greibach, UCLA), falls jede Produktion von der Form

$$A \to aA_1 \dots A_n$$

ist.

Satz 4.31

Zu jeder CFG G gibt es eine CFG G' in Greibach-Normalform mit $L(G') = L(G) \setminus \{\epsilon\}$.

4.4 Das Pumping-Lemma für kontextfreie Sprachen

Zur Erinnerung: Das Pumping-Lemma für reguläre Sprachen: Für jede reguläre Sprache L gibt es ein $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ zerlegen lässt in $z = uvw$ mit $|uv| \leq n$, $v \neq \epsilon$ und $uv^*w \subseteq L$.

4.4 Das Pumping-Lemma für kontextfreie Sprachen

Zur Erinnerung: Das Pumping-Lemma für reguläre Sprachen: Für jede reguläre Sprache L gibt es ein $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ zerlegen lässt in $z = uvw$ mit $|uv| \leq n$, $v \neq \epsilon$ und $uv^*w \subseteq L$.

Zum Beweis haben wir $n = |Q|$ gewählt, wobei Q die Zustandsmenge eines L erkennenden DFA war. Das Argument war dann, dass beim Erkennen von z (mindestens) ein Zustand zweimal besucht werden muss und damit der dazwischen liegende Weg im Automaten beliebig oft wiederholt werden kann.

4.4 Das Pumping-Lemma für kontextfreie Sprachen

Zur Erinnerung: Das Pumping-Lemma für reguläre Sprachen: Für jede reguläre Sprache L gibt es ein $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ zerlegen lässt in $z = uvw$ mit $|uv| \leq n$, $v \neq \epsilon$ und $uv^*w \subseteq L$.

Zum Beweis haben wir $n = |Q|$ gewählt, wobei Q die Zustandsmenge eines L erkennenden DFA war. Das Argument war dann, dass beim Erkennen von z (mindestens) ein Zustand zweimal besucht werden muss und damit der dazwischen liegende Weg im Automaten beliebig oft wiederholt werden kann.

Ein ähnliches Argument kann auch auf kontextfreie Grammatiken angewendet werden.

Satz 4.32 (Pumping-Lemma)

Für jede kontextfreie Sprache L gibt es ein $n \geq 1$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ zerlegen lässt in

$$z = uvwxy,$$

mit

Satz 4.32 (Pumping-Lemma)

Für jede kontextfreie Sprache L gibt es ein $n \geq 1$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ zerlegen lässt in

$$z = uvwxy,$$

mit

- $vx \neq \epsilon$,

Satz 4.32 (Pumping-Lemma)

Für jede kontextfreie Sprache L gibt es ein $n \geq 1$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ zerlegen lässt in

$$z = uvwxy,$$

mit

- $vx \neq \epsilon$,
- $|vwx| \leq n$, *und*

Satz 4.32 (Pumping-Lemma)

Für jede kontextfreie Sprache L *gibt es ein* $n \geq 1$*, so dass sich jedes Wort* $z \in L$ *mit* $|z| \geq n$ *zerlegen lässt in*

$$z = uvwxy,$$

mit

- $vx \neq \epsilon$,
- $|vwx| \leq n$, *und*
- $\forall i \in \mathbb{N}.\ uv^i wx^i y \in L.$

Satz 4.32 (Pumping-Lemma)

Für jede kontextfreie Sprache L gibt es ein $n \geq 1$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ zerlegen lässt in

$$z = uvwxy,$$

mit

- $vx \neq \epsilon$,
- $|vwx| \leq n$, *und*
- $\forall i \in \mathbb{N}.\ uv^i wx^i y \in L$.

Beweis:

Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-Normalform mit $L(G) = L \setminus \{\epsilon\}$. Wähle $n = 2^{|V|}$. Sei $z \in L(G)$ mit $|z| \geq n$.

Beweis:

Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-Normalform mit $L(G) = L \setminus \{\epsilon\}$. Wähle $n = 2^{|V|}$. Sei $z \in L(G)$ mit $|z| \geq n$.

Jeder Binärbaum mit $\geq 2^k$ Blättern hat die Höhe $\geq k$

Beweis:

Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-Normalform mit $L(G) = L \setminus \{\epsilon\}$. Wähle $n = 2^{|V|}$. Sei $z \in L(G)$ mit $|z| \geq n$.

Jeder Binärbaum mit $\geq 2^k$ Blättern hat die Höhe $\geq k$

Dann hat der Syntaxbaum für z (ohne die letzte Stufe für die Terminale) mindestens einen Pfad der Länge $\geq |V|$, da er wegen der Chomsky-Normalform ein Binärbaum ist: Wir betrachten einen solchen Pfad maximaler Länge.

Beweis:

Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-Normalform mit $L(G) = L \setminus \{\epsilon\}$. Wähle $n = 2^{|V|}$. Sei $z \in L(G)$ mit $|z| \geq n$.

Jeder Binärbaum mit $\geq 2^k$ Blättern hat die Höhe $\geq k$

Dann hat der Syntaxbaum für z (ohne die letzte Stufe für die Terminale) mindestens einen Pfad der Länge $\geq |V|$, da er wegen der Chomsky-Normalform ein Binärbaum ist: Wir betrachten einen solchen Pfad maximaler Länge.

Auf diesem Pfad der Länge $\geq |V|$ kommen $\geq |V| + 1$ Nichtterminale vor, also mindestens eins doppelt. Nennen wir es A.

Beweis:

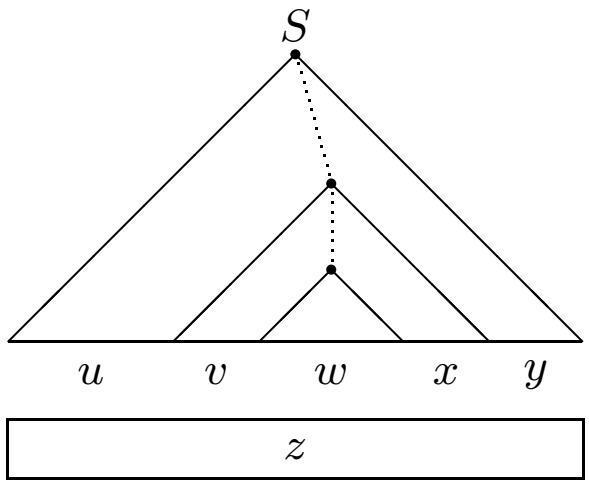
Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-Normalform mit $L(G) = L \setminus \{\epsilon\}$. Wähle $n = 2^{|V|}$. Sei $z \in L(G)$ mit $|z| \geq n$.

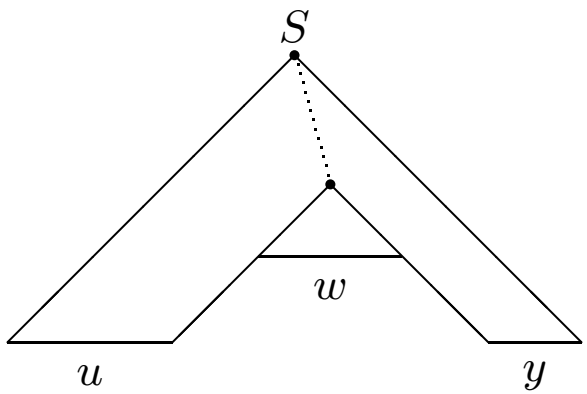
Jeder Binärbaum mit $\geq 2^k$ Blättern hat die Höhe $\geq k$

Dann hat der Syntaxbaum für z (ohne die letzte Stufe für die Terminale) mindestens einen Pfad der Länge $\geq |V|$, da er wegen der Chomsky-Normalform ein Binärbaum ist: Wir betrachten einen solchen Pfad maximaler Länge.

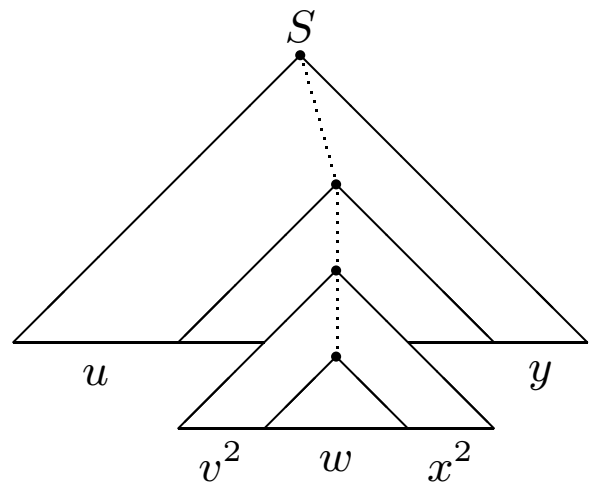
Auf diesem Pfad der Länge $\geq |V|$ kommen $\geq |V| + 1$ Nichtterminale vor, also mindestens eins doppelt. Nennen wir es A.

Die zwischen zwei Vorkommen von A liegende Teilableitung kann nun beliebig oft wiederholt werden.





Dieser Ableitungsbaum zeigt
$uwy \in L$



Dieser Ableitungsbaum zeigt
$uv^2wx^2y \in L$

Beweis (Forts.):

Die Bedingung $vx \neq \epsilon$ folgt, da das obere der beiden A's mit $A \to BC$ abgeleitet werden muss.

Beweis (Forts.):

Die Bedingung $vx \neq \epsilon$ folgt, da das obere der beiden A's mit $A \to BC$ abgeleitet werden muss.

Um $|vwx| \leq n$ zu erreichen, darf das obere A maximal $|V| + 1$ Schritte von jedem Blatt entfernt sein, weil $n = 2^{|V|}$. Da der ausgewählte Pfad maximale Länge hat, genügt es, dass das obere A vom Blatt dieses Pfads $|V| + 1$ Schritte entfernt ist. Da es nur $|V|$ Nichtterminale gibt, muss ein solches doppeltes A existieren. □

Beispiel 4.33

Wir zeigen, dass die Sprache

$$L := \{a^i b^i c^i \mid i \in \mathbb{N}\}$$

nicht kontextfrei ist.

Beispiel 4.33

Wir zeigen, dass die Sprache

$$L := \{a^i b^i c^i \mid i \in \mathbb{N}\}$$

nicht kontextfrei ist.

Wäre L kontextfrei, so gäbe es eine dazugehörige Pumping-Lemma Zahl n und wir könnten jedes $z \in L$ mit $|z| \geq n$, insb. $z = a^n b^n c^n$, zerlegen und aufpumpen, ohne aus der Sprache herauszufallen.

Beispiel 4.33

Wir zeigen, dass die Sprache

$$L := \{a^i b^i c^i \mid i \in \mathbb{N}\}$$

nicht kontextfrei ist.

Wäre L kontextfrei, so gäbe es eine dazugehörige Pumping-Lemma Zahl n und wir könnten jedes $z \in L$ mit $|z| \geq n$, insb. $z = a^n b^n c^n$, zerlegen und aufpumpen, ohne aus der Sprache herauszufallen.

Eine Zerlegung $z = uvwxy$ mit $|vwx| \leq n$ bedeutet aber, dass vwx nicht a's und c's enthalten kann. OE nehmen wir an, vwx enhält nur a's und b's. Da $vx \neq \epsilon$, muss vx mindestens ein a oder ein b enthalten.

Beispiel 4.33

Wir zeigen, dass die Sprache

$$L := \{a^i b^i c^i \mid i \in \mathbb{N}\}$$

nicht kontextfrei ist.

Wäre L kontextfrei, so gäbe es eine dazugehörige Pumping-Lemma Zahl n und wir könnten jedes $z \in L$ mit $|z| \geq n$, insb. $z = a^n b^n c^n$, zerlegen und aufpumpen, ohne aus der Sprache herauszufallen.

Eine Zerlegung $z = uvwxy$ mit $|vwx| \leq n$ bedeutet aber, dass vwx nicht a's und c's enthalten kann. OE nehmen wir an, vwx enhält nur a's und b's. Da $vx \neq \epsilon$, muss vx mindestens ein a oder ein b enthalten.

Damit gilt aber $\#_a(uv^2wx^2y) > \#_c(uv^2wx^2y)$ oder $\#_b(uv^2wx^2y) > \#_c(uv^2wx^2y)$. Also $uv^2wx^2y \notin L$. `

Beispiel 4.34

Mit dem Pumping-Lemma kann man zeigen, dass die Sprache $\{ww \mid w \in \{a, b\}^*\}$ nicht kontextfrei ist.

Beispiel 4.34

Mit dem Pumping-Lemma kann man zeigen, dass die Sprache $\{ww \mid w \in \{a, b\}^*\}$ nicht kontextfrei ist.

Dies zeigt, dass Kontextbedingungen in Programmiersprachen, etwa „*Deklaration vor Benutzung*“, nicht durch kontextfreie Grammatiken ausgedrückt werden können.

4.5 Algorithmen für kontextfreie Grammatiken

Wie üblich: $G = (V, \Sigma, P, S)$ ist eine CFG.

4.5 Algorithmen für kontextfreie Grammatiken

Wie üblich: $G = (V, \Sigma, P, S)$ ist eine CFG.

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

nützlich gdw es eine Ableitung $S \to_G^* w \in \Sigma^*$ gibt, in der X vorkommt.

erzeugend gdw es eine Ableitung $X \to_G^* w \in \Sigma^*$ gibt.

erreichbar gdw es eine Ableitung $S \to_G^* \alpha X \beta$ gibt.

4.5 Algorithmen für kontextfreie Grammatiken

Wie üblich: $G = (V, \Sigma, P, S)$ ist eine CFG.

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

nützlich gdw es eine Ableitung $S \to_G^* w \in \Sigma^*$ gibt, in der X vorkommt.

erzeugend gdw es eine Ableitung $X \to_G^* w \in \Sigma^*$ gibt.

erreichbar gdw es eine Ableitung $S \to_G^* \alpha X \beta$ gibt.

Fakt 4.35

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

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

4.5 Algorithmen für kontextfreie Grammatiken

Wie üblich: $G = (V, \Sigma, P, S)$ ist eine CFG.

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

nützlich gdw es eine Ableitung $S \to_G^* w \in \Sigma^*$ gibt, in der X vorkommt.

erzeugend gdw es eine Ableitung $X \to_G^* w \in \Sigma^*$ gibt.

erreichbar gdw es eine Ableitung $S \to_G^* \alpha X \beta$ gibt.

Fakt 4.35

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

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

Ziel: Elimination der unnützen Symbole und der Produktionen, in denen sie vorkommen.

Beispiel 4.36

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

Beispiel 4.36

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

1. Elimination nicht erzeugender Symbole:

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

2. Elimination unerreichbarer Symbole:

$$S \to a$$

Beispiel 4.36

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

1. Elimination nicht erzeugender Symbole:

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

2. Elimination unerreichbarer Symbole:

$$S \to a$$

Umgekehrte Reihenfolge:

1. Elimination unerreichbarer Symbole:

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

2. Elimination nicht erzeugender Symbole:

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

Satz 4.37

Eliminiert man aus einer Grammatik G

1. *alle nicht erzeugenden Symbole, mit Resultat G_1, und*
2. *aus G_1 alle unerreichbaren Symbole, mit Resultat G_2,*

dann enthält G_2 nur noch nützliche Symbole und $L(G_2) = L(G)$.

Satz 4.37

Eliminiert man aus einer Grammatik G

1. *alle nicht erzeugenden Symbole, mit Resultat G_1, und*
2. *aus G_1 alle unerreichbaren Symbole, mit Resultat G_2,*

dann enthält G_2 nur noch nützliche Symbole und $L(G_2) = L(G)$.

Beweis:

Wir zeigen zuerst $L(G_2) = L(G)$.

Da $P_2 \subseteq P$ gilt $L(G_2) \subseteq L(G)$.

Umgekehrt, sei $w \in L(G)$, dh $S \to_G^* w$.
Jedes Symbol in dieser Ableitung ist erreichbar und erzeugend.
Also gilt auch $S \to_{G_2}^* w$, dh $w \in L(G_2)$.

Beweis (Forts.): $[G_1$: erzeugend in G, G_2 : erreichbar in $G_1]$

Beweis (Forts.): $[G_1$: erzeugend in $G,\ G_2$: erreichbar in $G_1]$

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$.
Da alle Symbole in der Ableitung erreichbar sind: $S \to^*_{G_2} \alpha X \beta$.

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$.
Da alle Symbole in der Ableitung erreichbar sind: $S \to^*_{G_2} \alpha X \beta$.

Alle Symbole in $\alpha X \beta$ müssen in G erzeugend sein:

$$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma^*.\ Y \to^*_G u$$

Da alle Symbole in den Ableitungen $Y \to^*_G u$ erzeugend sind:

$$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma_1^*.\ Y \to^*_{G_1} u$$

Also gibt es $w \in \Sigma_1^*$ mit $\alpha X \beta \to^*_{G_1} w$.

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$.
Da alle Symbole in der Ableitung erreichbar sind: $S \to^*_{G_2} \alpha X \beta$.

Alle Symbole in $\alpha X \beta$ müssen in G erzeugend sein:

$$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma^*.\ Y \to^*_G u$$

Da alle Symbole in den Ableitungen $Y \to^*_G u$ erzeugend sind:

$$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma_1^*.\ Y \to^*_{G_1} u$$

Also gibt es $w \in \Sigma_1^*$ mit $\alpha X \beta \to^*_{G_1} w$.

Die Ableitung $\alpha X \beta \to^*_{G_1} w$ enthält nur Symbole, die in G_1 erreichbar sind. Daher auch $\alpha X \beta \to^*_{G_2} w$.

$$S \ \to^*_{G_2} \ \alpha X \beta \ \to^*_{G_2} \ w$$

□

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

- Alle Symbole in Σ sind erzeugend.
- Falls $(A \to \alpha) \in P$ und alle Symbole in α sind erzeugend, dann ist auch A erzeugend. □

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

- Alle Symbole in Σ sind erzeugend.
- Falls $(A \to \alpha) \in P$ und alle Symbole in α sind erzeugend, dann ist auch A erzeugend. □

Beispiel 4.39

$$S \to SAB, \quad A \to BC, \quad B \to C, \quad C \to c$$

Erzeugend: $\{c, C, B, A\}$.

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

- Alle Symbole in Σ sind erzeugend.
- Falls $(A \to \alpha) \in P$ und alle Symbole in α sind erzeugend, dann ist auch A erzeugend.

□

Beispiel 4.39

$$S \to SAB, \quad A \to BC, \quad B \to C, \quad C \to c$$

Erzeugend: $\{c, C, B, A\}$.

Korollar 4.40

Für eine kontextfreie Grammatik G ist entscheidbar, ob $L(G) = \emptyset$.

Beweis:

$L(G) = \emptyset \iff S$ ist nicht erzeugend.

□

Satz 4.41

Die Menge der erreichbaren Symbole einer CFG ist berechenbar.

Beweis:

Satz 4.41

Die Menge der erreichbaren Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erreichbaren Symbole induktiv:

- S ist erreichbar.
- Ist A erreichbar und $(A \to \alpha X \beta) \in P$,
  so ist auch X erreichbar.

$\square$

Satz 4.42

Das Wortproblem ($w \in L(G)$?) ist für eine CFG G entscheidbar.

Satz 4.42

Das Wortproblem ($w \in L(G)$?) ist für eine CFG G entscheidbar.

Beweis:

OE sei $w \neq \epsilon$. Wir eliminieren zuerst alle ϵ-Produktionen aus G (wie in Lemma 4.26).
Dann berechnen wir induktiv die Menge R aller von S ableitbaren Wörter $\in (V \cup \Sigma)^*$, die nicht länger als w sind:

- $S \in R$
- Wenn $\alpha B \gamma \in R$ und $(B \to \beta) \in P$ und $|\alpha\beta\gamma| \leq |w|$, dann auch $\alpha\beta\gamma \in R$.

Satz 4.42

Das Wortproblem ($w \in L(G)$?) ist für eine CFG G entscheidbar.

Beweis:

OE sei $w \neq \epsilon$. Wir eliminieren zuerst alle ϵ-Produktionen aus G (wie in Lemma 4.26).
Dann berechnen wir induktiv die Menge R aller von S ableitbaren Wörter $\in (V \cup \Sigma)^*$, die nicht länger als w sind:

- $S \in R$
- Wenn $\alpha B \gamma \in R$ und $(B \to \beta) \in P$ und $|\alpha\beta\gamma| \leq |w|$, dann auch $\alpha\beta\gamma \in R$.

Es gilt:

$$w \in L_V(G) \quad \Leftrightarrow \quad w \in R$$

wobei $L_V(G) := \{w \in (V \cup \Sigma)^* \mid S \to_G^* w\}$.
Da R endlich ist ($|R| \leq |V \cup \Sigma|^{|w|}$), ist $w \in R$ entscheidbar, und damit auch $w \in L_V(G)$, und damit auch $w \in L(G)$. □

4.6 Der Cocke-Younger-Kasami-Algorithmus

Der CYK-Algorithmus entscheidet das Wortproblem für kontextfreie Grammatiken in Chomsky-Normalform.

4.6 Der Cocke-Younger-Kasami-Algorithmus

Der CYK-Algorithmus entscheidet das Wortproblem für kontextfreie Grammatiken in Chomsky-Normalform.

Eingabe: Grammatik $G = (V, \Sigma, P, S)$ in Chomsky-Normalform, $w = a_1 \dots a_n \in \Sigma^*$.

4.6 Der Cocke-Younger-Kasami-Algorithmus

Der CYK-Algorithmus entscheidet das Wortproblem für kontextfreie Grammatiken in Chomsky-Normalform.

Eingabe: Grammatik $G = (V, \Sigma, P, S)$ in Chomsky-Normalform, $w = a_1 \dots a_n \in \Sigma^*$.

Definition 4.43

$$V_{ij} := \{A \in V \mid A \to_G^* a_i \dots a_j\} \quad \text{für } i \leq j$$

Damit gilt:

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

Der CYK-Algorithmus berechnet die V_{ij} rekursiv nach wachsendem $j - i$:

$$
\begin{aligned}
V_{ii} &= \{A \in V \mid (A \to a_i) \in P\} \\
V_{ij} &= \left\{ A \in V \mid \begin{array}{c} \exists i \leq k < j,\ B \in V_{ik},\ C \in V_{k+1,j} \cdot \\ (A \to BC) \in P \end{array} \right\} \quad \text{für } i < j
\end{aligned}
$$

Der CYK-Algorithmus berechnet die V_{ij} rekursiv nach wachsendem $j - i$:

$$
\begin{aligned}
V_{ii} &= \{A \in V \mid (A \to a_i) \in P\} \\
V_{ij} &= \left\{ A \in V \mid \begin{array}{c} \exists i \leq k < j,\ B \in V_{ik},\ C \in V_{k+1,j} \cdot \\ (A \to BC) \in P \end{array} \right\} \quad \text{für } i < j
\end{aligned}
$$

Korrektheitsbeweis: Induktion nach $j - i$.

Der CYK-Algorithmus berechnet die V_{ij} rekursiv nach wachsendem $j - i$:

$$
\begin{aligned}
V_{ii} &= \{A \in V \mid (A \to a_i) \in P\} \\
V_{ij} &= \left\{ A \in V \mid \begin{array}{c} \exists i \le k < j,\ B \in V_{ik},\ C \in V_{k+1,j} \cdot \\ (A \to BC) \in P \end{array} \right\} \quad \text{für } i < j
\end{aligned}
$$

Korrektheitsbeweis: Induktion nach $j - i$.

Die V_{ij} als Tabelle (mit ij statt V_{ij}):

14			
13	24		
12	23	34	
11	22	33	44
a_1	a_2	a_3	a_4

Beispiel 4.44

$$
\begin{array}{lcl}
S & \to & AB \mid BC \\
A & \to & BA \mid a \\
B & \to & CC \mid b \\
C & \to & AB \mid a
\end{array}
$$

15				
14	25			
13	24	35		
12	23	34	45	
11	22	33	44	55
b	a	a	b	a

Satz 4.45

Der CYK-Algorithmus entscheidet das Wortproblem $w \in L(G)$ für eine fixe CFG G in Chomsky-Normalform in Zeit $O(|w|^3)$.

Beweis:

Sei $n := |w|$. Es werden $\frac{n(n-1)}{2} \in O(n^2)$ Mengen V_{ij} berechnet.

$$V_{ij} \quad = \quad \left\{ A \in V \mid \begin{array}{c} \exists i \leq k < j,\ B \in V_{ik},\ C \in V_{k+1,j} \cdot \\ (A \to BC) \in P \end{array} \right\} \quad (i < j)$$

Satz 4.45

Der CYK-Algorithmus entscheidet das Wortproblem $w \in L(G)$ für eine fixe CFG G in Chomsky-Normalform in Zeit $O(|w|^3)$.

Beweis:

Sei $n := |w|$. Es werden $\frac{n(n-1)}{2} \in O(n^2)$ Mengen V_{ij} berechnet.

$$V_{ij} \quad = \quad \left\{ A \in V \mid \begin{array}{c} \exists i \leq k < j,\ B \in V_{ik},\ C \in V_{k+1,j} \cdot \\ (A \to BC) \in P \end{array} \right\} \quad (i < j)$$

Für jede dieser Mengen werden

- $j - i < n$ Werte für k betrachtet,
- für jedes k wird für alle Produktionen $A \to BC$ untersucht, ob $B \in V_{ik}$ und $C \in V_{k+1,j}$, wobei $|V_{ik}|, |V_{k+1,j}| \leq |V|$.

Satz 4.45

Der CYK-Algorithmus entscheidet das Wortproblem $w \in L(G)$ für eine fixe CFG G in Chomsky-Normalform in Zeit $O(|w|^3)$.

Beweis:

Sei $n := |w|$. Es werden $\frac{n(n-1)}{2} \in O(n^2)$ Mengen V_{ij} berechnet.

$$V_{ij} = \left\{ A \in V \mid \begin{array}{c} \exists i \leq k < j,\ B \in V_{ik},\ C \in V_{k+1,j}. \\ (A \to BC) \in P \end{array} \right\} \quad (i < j)$$

Für jede dieser Mengen werden

- $j - i < n$ Werte für k betrachtet,
- für jedes k wird für alle Produktionen $A \to BC$ untersucht, ob $B \in V_{ik}$ und $C \in V_{k+1,j}$, wobei $|V_{ik}|, |V_{k+1,j}| \leq |V|$.

Gesamtzeit: $O(n^3)$, denn $|P|$ und $|V|$ sind Konstanten unabhängig von n. [Konstruktion jeder Menge V_{ii}: $O(1)$. Für alle V_{ij} also $O(n)$.] □

Erweiterung Der CYK-Algorithmus kann so erweitert werden, dass er nicht nur das Wortproblem entscheidet, sondern auch die Menge der Syntaxbäume für die Eingabe berechnet.

Erweiterung Der CYK-Algorithmus kann so erweitert werden, dass er nicht nur das Wortproblem entscheidet, sondern auch die Menge der Syntaxbäume für die Eingabe berechnet.

Realisierung:

- V_{ij} ist die Menge der Syntaxbäume mit Rand $a_i \dots a_j$.
- Statt A enthält V_{ij} die Syntaxbäume, dessen Wurzel mit A beschriftet ist.

Vorschau

Für CFGs sind folgende Probleme nicht entscheidbar:

- Äquivalenz: $L(G_1) = L(G_2)$?
- Schnittproblem: $L(G_1) \cap L(G_2) = \emptyset$?
- Regularität: $L(G)$ regulär?
- Mehrdeutigkeit: Ist G mehrdeutig?

4.7 Abschlusseigenschaften

Satz 4.46

Seien kontextfreie Grammatiken $G_1 = (V_1, \Sigma_1, P_1, S_1)$ *und* $G_2 = (V_2, \Sigma_2, P_2, S_2)$ *gegeben. Dann kann man in linearer Zeit kontextfreie Grammatiken für*

1. $L(G_1) \cup L(G_2)$
2. $L(G_1)L(G_2)$
3. $(L(G_1))^*$
4. $(L(G_1))^R$

konstruieren.

Die Klasse der kontextfreien Sprachen ist also unter Vereinigung, Konkatenation, Stern und Spiegelung abgeschlossen.

Beweis:

OE nehmen wir an, dass $V_1 \cap V_2 = \emptyset$.

1. $V = V_1 \cup V_2 \cup \{S\}$; S neu
   $P = P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}$
2. $V = V_1 \cup V_2 \cup \{S\}$; S neu
   $P = P_1 \cup P_2 \cup \{S \to S_1 S_2\}$
3. $V = V_1 \cup \{S\}$; S neu
   $P = P_1 \cup \{S \to \epsilon \mid S_1 S\}$
4. $P = \{A \to \alpha^R \mid (A \to \alpha) \in P_1\}$

□

Beweis:

OE nehmen wir an, dass $V_1 \cap V_2 = \emptyset$.

1. $V = V_1 \cup V_2 \cup \{S\}$; S neu
   $P = P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}$
2. $V = V_1 \cup V_2 \cup \{S\}$; S neu
   $P = P_1 \cup P_2 \cup \{S \to S_1 S_2\}$
3. $V = V_1 \cup \{S\}$; S neu
   $P = P_1 \cup \{S \to \epsilon \mid S_1 S\}$
4. $P = \{A \to \alpha^R \mid (A \to \alpha) \in P_1\}$

□

Es folgt: Verallgemeinerte kontextfreie Grammatiken mit Produktionen der Gestalt $X \to r$, wobei r ein regulärer Ausdruck über $(V \cup T)$ ist, erzeugen kontextfreie Sprachen.

Satz 4.47

Die Menge der kontextfreien Sprachen ist nicht abgeschlossen unter Durchschnitt oder Komplement.

Satz 4.47

Die Menge der kontextfreien Sprachen ist nicht abgeschlossen unter Durchschnitt oder Komplement.

Beweis:

$$L_1 := \{a^i b^i c^j \mid i, j \in \mathbb{N}\} \text{ ist kontextfrei}$$

$$L_2 := \{a^i b^j c^j \mid i, j \in \mathbb{N}\} \text{ ist kontextfrei}$$

$$L_1 \cap L_2 = \{a^i b^i c^i \mid i \in \mathbb{N}\} \text{ ist nicht kontextfrei}$$

Satz 4.47

Die Menge der kontextfreien Sprachen ist nicht abgeschlossen unter Durchschnitt oder Komplement.

Beweis:

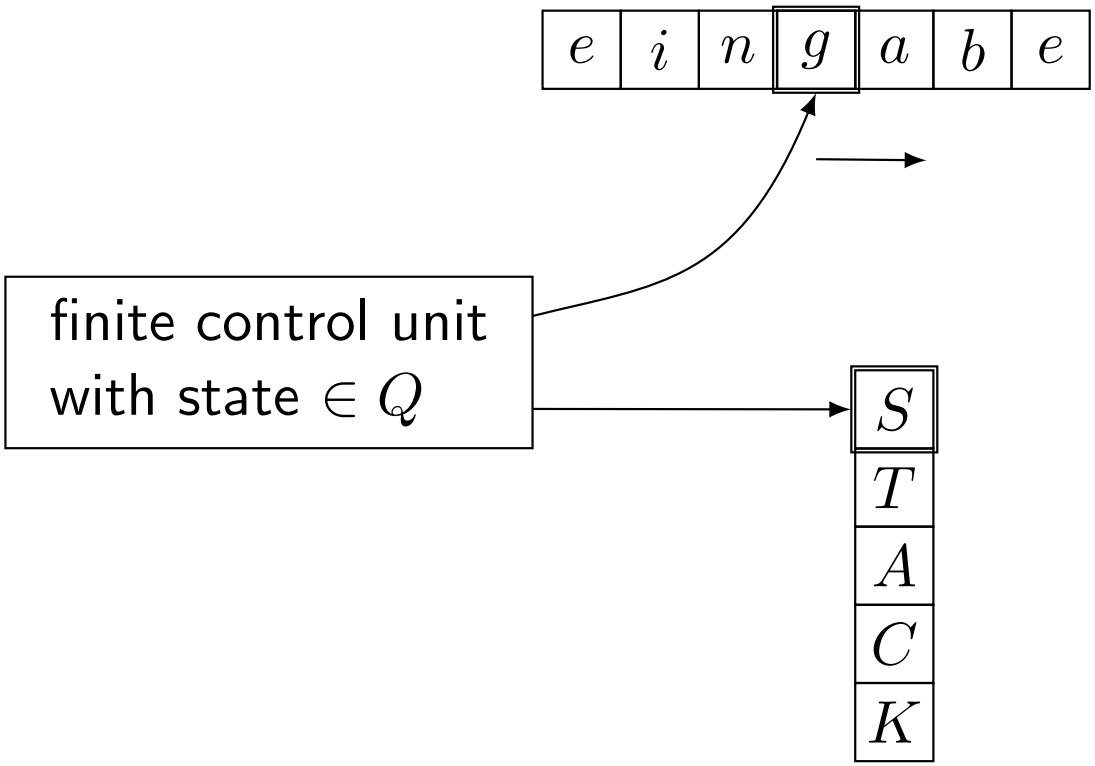
$$L_1 := \{a^i b^i c^j \mid i, j \in \mathbb{N}\} \text{ ist kontextfrei}$$

$$L_2 := \{a^i b^j c^j \mid i, j \in \mathbb{N}\} \text{ ist kontextfrei}$$

$$L_1 \cap L_2 = \{a^i b^i c^i \mid i \in \mathbb{N}\} \text{ ist nicht kontextfrei}$$

Wegen de Morgan (1806–1871) können die CFLs daher auch nicht unter Komplement abgeschlossen sein. □

4.8 Kellerautomaten



Anwendungsgebiete von Kellerautomaten:

- Syntaxanalyse von Programmiersprachen
- Analyse von Programmen mit Rekursion

Definition 4.48

Ein (nichtdeterministischer) Kellerautomat $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,
- einem endlichen Eingabealphabet Σ,
- einem endlichen Kelleralphabet Γ,
- einem Anfangszustand q_0,
- einem untersten Kellerzeichen Z_0,
- einer Übergangsfunktion $\delta\colon Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}_e(Q \times \Gamma^*)$, (Hierbei bedeutet $\mathcal{P}_e$ die Menge aller endlichen Teilmengen)
- einer Menge $F \subseteq Q$ von Endzuständen.

Definition 4.48

Ein (nichtdeterministischer) Kellerautomat $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,
- einem endlichen Eingabealphabet Σ,
- einem endlichen Kelleralphabet Γ,
- einem Anfangszustand q_0,
- einem untersten Kellerzeichen Z_0,
- einer Übergangsfunktion $\delta\colon Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}_e(Q \times \Gamma^*)$, (Hierbei bedeutet $\mathcal{P}_e$ die Menge aller endlichen Teilmengen)
- einer Menge $F \subseteq Q$ von Endzuständen.

Intuitive Bedeutung von $(q', \alpha) \in \delta(q, a, Z)$:

> *Wenn sich M in Zustand q befindet, das Eingabezeichen a liest, und Z das oberste Kellerzeichen ist, so kann M im nächsten Schritt in q' übergehen und Z durch α ersetzen.*

Intuitive Bedeutung von $(q', \alpha) \in \delta(q, a, Z)$:

Wenn sich M in Zustand q befindet, das Eingabezeichen a liest, und Z das oberste Kellerzeichen ist, so kann M im nächsten Schritt in q' übergehen und Z durch α ersetzen.

Intuitive Bedeutung von $(q', \alpha) \in \delta(q, a, Z)$:

> *Wenn sich M in Zustand q befindet, das Eingabezeichen a liest, und Z das oberste Kellerzeichen ist, so kann M im nächsten Schritt in q' übergehen und Z durch α ersetzen.*

Spezialfälle:

POP-Operation:	$\alpha = \epsilon$.
	Das oberste kellerzeichen Z wird entfernt.
PUSH-Operation:	$\alpha = Z'Z$.
	Z' wird als neues oberstes Kellerzeichen gePUSHt.
ϵ-Übergang	$a = \epsilon$.
	Ohne Lesen eines Eingabezeichens.

Intuitive Bedeutung von $(q', \alpha) \in \delta(q, a, Z)$:

> *Wenn sich M in Zustand q befindet, das Eingabezeichen a liest, und Z das oberste Kellerzeichen ist, so kann M im nächsten Schritt in q' übergehen und Z durch α ersetzen.*

Spezialfälle:

POP-Operation:	$\alpha = \epsilon$.
	Das oberste kellerzeichen Z wird entfernt.
PUSH-Operation:	$\alpha = Z'Z$.
	Z' wird als neues oberstes Kellerzeichen gePUSHt.
ϵ-Übergang	$a = \epsilon$.
	Ohne Lesen eines Eingabezeichens.

Aufgabe: Jeder Schritt kann durch eine Sequenz von diesen drei Operationen simuliert werden. (Hilfszustände verwenden.)

Definition 4.49

Eine Konfiguration eines Kellerautomaten M ist ein Tripel (q, w, α) mit $q \in Q$, $w \in \Sigma^*$ und $\alpha \in \Gamma^*$.
Die Anfangskonfiguration von M für die Eingabe $w \in \Sigma^*$ ist (q_0, w, Z_0).

Definition 4.49

Eine Konfiguration eines Kellerautomaten M ist ein Tripel (q, w, α) mit $q \in Q$, $w \in \Sigma^*$ und $\alpha \in \Gamma^*$.
Die Anfangskonfiguration von M für die Eingabe $w \in \Sigma^*$ ist (q_0, w, Z_0).

Intuitiv stellt eine Konfiguration (q, w, α) eine "Momentaufnahme" des Kellerautomaten dar:

- Der momentane Zustand ist q.
- Der noch zu lesende Teil der Eingabe ist w.
- Der aktuelle Kellerinhalt ist α (das oberste Kellerzeichen ganz links stehend).

Definition 4.50

Auf der Menge aller Konfigurationen definieren wir eine binäre Relation $\rightarrow_M$ wie folgt:

$$(q, aw, Z\alpha) \rightarrow_M \begin{cases} (q', w, \beta\alpha) & \text{falls } (q', \beta) \in \delta(q, a, Z) \\ (q', aw, \beta\alpha) & \text{falls } (q', \beta) \in \delta(q, \epsilon, Z) \end{cases}$$

Die reflexive und transitive Hülle von $\rightarrow_M$ wird mit $\rightarrow_M^*$ bezeichnet.

Definition 4.50

Auf der Menge aller Konfigurationen definieren wir eine binäre Relation $\to_M$ wie folgt:

$$(q, aw, Z\alpha) \to_M \begin{cases} (q', w, \beta\alpha) & \text{falls } (q', \beta) \in \delta(q, a, Z) \\ (q', aw, \beta\alpha) & \text{falls } (q', \beta) \in \delta(q, \epsilon, Z) \end{cases}$$

Die reflexive und transitive Hülle von $\to_M$ wird mit $\to_M^*$ bezeichnet.

Intuitive Bedeutung von $(q, w, \alpha) \to_M (q', w', \alpha')$:

> *Wenn M sich in der Konfiguration (q, w, α) befindet, dann kann er in einen Schritt in die Nachfolgerkonfiguration (q', w', α') übergehen.*

Achtung: eine Konfiguration kann mehrere Nachfolger haben (Nichtdeterminismus!)

Definition 4.51

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

$$(q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma) \text{ für ein } f \in F,\ \gamma \in \Gamma^*.$$

$$L_F(M) := \{w \mid \exists f \in F, \gamma \in \Gamma^*.\ (q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma)\}$$

Definition 4.51

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

$$(q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma) \text{ für ein } f \in F,\ \gamma \in \Gamma^*.$$

$$L_F(M) := \{w \mid \exists f \in F, \gamma \in \Gamma^*.\ (q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma)\}$$

- Ein PDA M akzeptiert $w \in \Sigma^*$ mit leeren Keller gdw

$$(q_0, w, Z_0) \to_M^* (q, \epsilon, \epsilon) \text{ für ein } q \in Q.$$

$$L_\epsilon(M) := \{w \mid \exists q \in Q.\ (q_0, w, Z_0) \to_M^* (q, \epsilon, \epsilon)\}$$

Definition 4.51

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

$$(q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma) \text{ für ein } f \in F,\ \gamma \in \Gamma^*.$$

$$L_F(M) := \{w \mid \exists f \in F, \gamma \in \Gamma^*.\ (q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma)\}$$

- Ein PDA M akzeptiert $w \in \Sigma^*$ mit leeren Keller gdw

$$(q_0, w, Z_0) \to_M^* (q, \epsilon, \epsilon) \text{ für ein } q \in Q.$$

$$L_\epsilon(M) := \{w \mid \exists q \in Q.\ (q_0, w, Z_0) \to_M^* (q, \epsilon, \epsilon)\}$$

Konvention: Wir blenden die F-Komponente von M aus, wenn wir nur an $L_\epsilon(M)$ interessiert sind.

Beispiel 4.52

Die Sprache $L = \{ww^R \mid w \in \{0,1\}^*\}$ wird vom PDA

$$M = (\{p, q, r\}, \{0, 1\}, \{0, 1, Z_0\}, p, Z_0, \delta, \{r\})$$

$$\begin{array}{lcll}
\delta(p, a, Z) & = & \{(p, aZ)\} & \text{für } a \in \{0,1\},\ Z \in \{0, 1, Z_0\} \\
\delta(p, \epsilon, Z) & = & \{(q, Z)\} & \text{für } Z \in \{0, 1, Z_0\} \\
\delta(q, a, a) & = & \{(q, \epsilon)\} & \text{für } a \in \{0, 1\} \\
\delta(q, \epsilon, Z_0) & = & \{(r, \epsilon)\} &
\end{array}$$

sowohl mit Endzustand als auch mit leerem Keller akzeptiert.

Bsp: $(p, 110011, Z_0) \to_M^* (r, \epsilon, \epsilon)$.

Beispiel 4.52

Die Sprache $L = \{ww^R \mid w \in \{0,1\}^*\}$ wird vom PDA

$$M = (\{p,q,r\},\ \{0,1\},\{0,1,Z_0\},p,\ Z_0,\ \delta,\ \{r\})$$

$$\begin{array}{lcll}
\delta(p,a,Z) & = & \{(p,\ aZ)\} & \text{für } a \in \{0,1\},\ Z \in \{0,1,Z_0\} \\
\delta(p,\epsilon,Z) & = & \{(q,\ Z)\} & \text{für } Z \in \{0,1,Z_0\} \\
\delta(q,a,a) & = & \{(q,\ \epsilon\ )\} & \text{für } a \in \{0,1\} \\
\delta(q,\epsilon,Z_0) & = & \{(r,\ \epsilon)\} &
\end{array}$$

sowohl mit Endzustand als auch mit leerem Keller akzeptiert.

Bsp: $(p,110011,Z_0) \to_M^* (r,\epsilon,\epsilon)$.

Definiert man $\delta(q,\epsilon,Z_0)$ als $\{(q,\ \epsilon)\}$ statt $\{(r,\ \epsilon)\}$ so gilt:

$$L_F(M') = \emptyset \text{ aber } L_\epsilon(M') = L.$$

Bemerkungen: PDAs und das Wortproblem

- Mit einem NFA A kann man $w \in L(A)$ durch parallele Verfolgung aller Berechnungspfade entscheiden, da sie alle endlich sind.

Bemerkungen: PDAs und das Wortproblem

- Mit einem NFA A kann man $w \in L(A)$ durch parallele Verfolgung aller Berechnungspfade entscheiden, da sie alle endlich sind.
- Bei einem PDA kann es wegen ϵ-Übergängen auch unendliche Berechnungen $\to_M$ geben, zB $\delta(q, \epsilon, Z) = (q, ZZ)$:

$$(q, w, Z) \to_M (q, w, ZZ) \to_M (q, w, ZZZ) \to_M \ldots$$

  Diese sind wegen des möglicherweise wachsenden oder pulsierenden Kellers nicht einfach zu eliminieren.

Bemerkungen: PDAs und das Wortproblem

- Mit einem NFA A kann man $w \in L(A)$ durch parallele Verfolgung aller Berechnungspfade entscheiden, da sie alle endlich sind.
- Bei einem PDA kann es wegen ϵ-Übergängen auch unendliche Berechnungen $\to_M$ geben, zB $\delta(q, \epsilon, Z) = (q, ZZ)$:

$$(q, w, Z) \to_M (q, w, ZZ) \to_M (q, w, ZZZ) \to_M \ldots$$

  Diese sind wegen des möglicherweise wachsenden oder pulsierenden Kellers nicht einfach zu eliminieren.
- Daher ist es a priori unklar, wie man mit einem PDA das Wortproblem entscheidet.

Ziel:

Akzeptanz durch Endzustände und leeren Keller gleich mächtig.

Ziel:

Akzeptanz durch Endzustände und leeren Keller gleich mächtig.

Satz 4.53 (Endzustand $\rightarrow$ leerer Keller)

Zu jedem PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ *kann man in linearer Zeit einen PDA* $M' = (Q', \Sigma, \Gamma', q_0', Z_0', \delta')$ *konstruieren mit* $L_F(M) = L_\epsilon(M')$.

Ziel:

Akzeptanz durch Endzustände und leeren Keller gleich mächtig.

Satz 4.53 (Endzustand $\rightarrow$ leerer Keller)

Zu jedem PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ *kann man in linearer Zeit einen PDA* $M' = (Q', \Sigma, \Gamma', q_0', Z_0', \delta')$ *konstruieren mit* $L_F(M) = L_\epsilon(M')$.

Ziel:

$$(q_0, w, Z_0) \rightarrow_M^* (f, \epsilon, \gamma) \quad \Leftrightarrow \quad (q_0', w, Z_0') \rightarrow_{M'}^* (q, \epsilon, \epsilon)$$

Beweisskizze:

M' (mit leerem Keller) simuliert M (mit Endzustand). Zusätzlich:

- Sobald M einen Endzustand erreicht, darf M' den Keller leeren (im neuen Zustand $\bar{q}$).
- Um zu verhindern, dass der Keller von M' leer wird, ohne dass M in einem Endzustand ist, führen wir ein neues Kellersymbol Z_0' ein.

Beweisskizze:

M' (mit leerem Keller) simuliert M (mit Endzustand). Zusätzlich:

- Sobald M einen Endzustand erreicht, darf M' den Keller leeren (im neuen Zustand $\bar{q}$).
- Um zu verhindern, dass der Keller von M' leer wird, ohne dass M in einem Endzustand ist, führen wir ein neues Kellersymbol Z_0' ein.

$$\begin{aligned} Q' &:= Q \uplus \{\bar{q}, q_0'\} \\ \Gamma' &:= \Gamma \uplus \{Z_0'\} \end{aligned}$$

Wir erweitern δ zu δ':

$$\begin{aligned} \delta'(q_0', \epsilon, Z_0') &= \{(q_0, Z_0 Z_0')\} \\ \delta'(q, b, Z) &= \delta(q, b, Z) && \text{für } q \in Q \setminus F,\ b \in \Sigma \cup \{\epsilon\},\ Z \in \Gamma \\ \delta'(f, a, Z) &= \delta(f, a, Z) && \text{für } f \in F,\ a \in \Sigma,\ Z \in \Gamma \\ \delta'(f, \epsilon, Z) &= \delta(f, \epsilon, Z) \cup \{(\bar{q}, \epsilon)\} && \text{für } f \in F,\ Z \in \Gamma' \\ \delta'(\bar{q}, \epsilon, Z) &= \{(\bar{q}, \epsilon)\} && \text{für } Z \in \Gamma' \end{aligned}$$

□

Satz 4.54 (Leerer Keller $\rightarrow$ Endzustand)

Zu jedem PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$ *kann man in linearer Zeit einen PDA* $M' = (Q', \Sigma, \Gamma', q_0', Z_0', \delta', F)$ *konstruieren mit* $L_\epsilon(M) = L_F(M')$.

Satz 4.54 (Leerer Keller $\rightarrow$ Endzustand)

Zu jedem PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$ *kann man in linearer Zeit einen PDA* $M' = (Q', \Sigma, \Gamma', q_0', Z_0', \delta', F)$ *konstruieren mit* $L_\epsilon(M) = L_F(M')$.

Beweisskizze:

M' (mit Endzustand) simuliert M (mit leerem Keller). Zusätzlich:

- M' schreibt am Anfang ein neues Zeichen Z_0' auf den Keller.
- Sobald M auf dem Keller Z_0' findet, ist der Keller eigentlich leer, und M' geht in den (neuen) Endzustand f.

Satz 4.54 (Leerer Keller $\rightarrow$ Endzustand)

Zu jedem PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$ *kann man in linearer Zeit einen PDA* $M' = (Q', \Sigma, \Gamma', q_0', Z_0', \delta', F)$ *konstruieren mit* $L_\epsilon(M) = L_F(M')$.

Beweisskizze:

M' (mit Endzustand) simuliert M (mit leerem Keller). Zusätzlich:

- M' schreibt am Anfang ein neues Zeichen Z_0' auf den Keller.
- Sobald M auf dem Keller Z_0' findet, ist der Keller eigentlich leer, und M' geht in den (neuen) Endzustand f.

$$
\begin{array}{lcl}
Q' & := & Q \uplus \{q_0', f\} \\
\Gamma' & := & \Gamma \uplus \{Z_0'\} \\
F & := & \{f\}
\end{array}
$$

$$
\begin{array}{lcll}
\delta'(q_0', \epsilon, Z_0') & = & \{(q_0, Z_0 Z_0')\} & \\
\delta'(q, b, Z) & = & \delta(q, b, Z) & \text{für } q \in Q,\ b \in \Sigma \cup \{\epsilon\},\ Z \in \Gamma \\
\delta'(q, \epsilon, Z_0') & = & \{(f, Z_0')\} & \text{für } q \in Q
\end{array}
$$

□

Die folgenden Sätze erleichtern Beweise über Berechnungen von PDAs.

Die folgenden Sätze erleichtern Beweise über Berechnungen von PDAs.

Lemma 4.55 (Erweiterungslemma)

$$(q, u, \alpha) \to_M^n (q', u', \alpha') \implies (q, uv, \alpha\beta) \to_M^n (q', u'v, \alpha'\beta)$$

Die folgenden Sätze erleichtern Beweise über Berechnungen von PDAs.

Lemma 4.55 (Erweiterungslemma)

$$(q, u, \alpha) \to_M^n (q', u', \alpha') \implies (q, uv, \alpha\beta) \to_M^n (q', u'v, \alpha'\beta)$$

Beweis:

Nach Definition von $\to_M$ gilt

$$\begin{aligned}&(q_i, u_i, \alpha_i) \to_M (q_{i+1}, u_{i+1}, \alpha_{i+1}) \implies \\ &(q_i, u_i v, \alpha_i \beta) \to_M (q_{i+1}, u_{i+1} v, \alpha_{i+1} \beta)\end{aligned}$$

Mit Induktion über n folgt die Behauptung. □

Die folgenden Sätze erleichtern Beweise über Berechnungen von PDAs.

Lemma 4.55 (Erweiterungslemma)

$$(q, u, \alpha) \to_M^n (q', u', \alpha') \implies (q, uv, \alpha\beta) \to_M^n (q', u'v, \alpha'\beta)$$

Beweis:

Nach Definition von $\to_M$ gilt

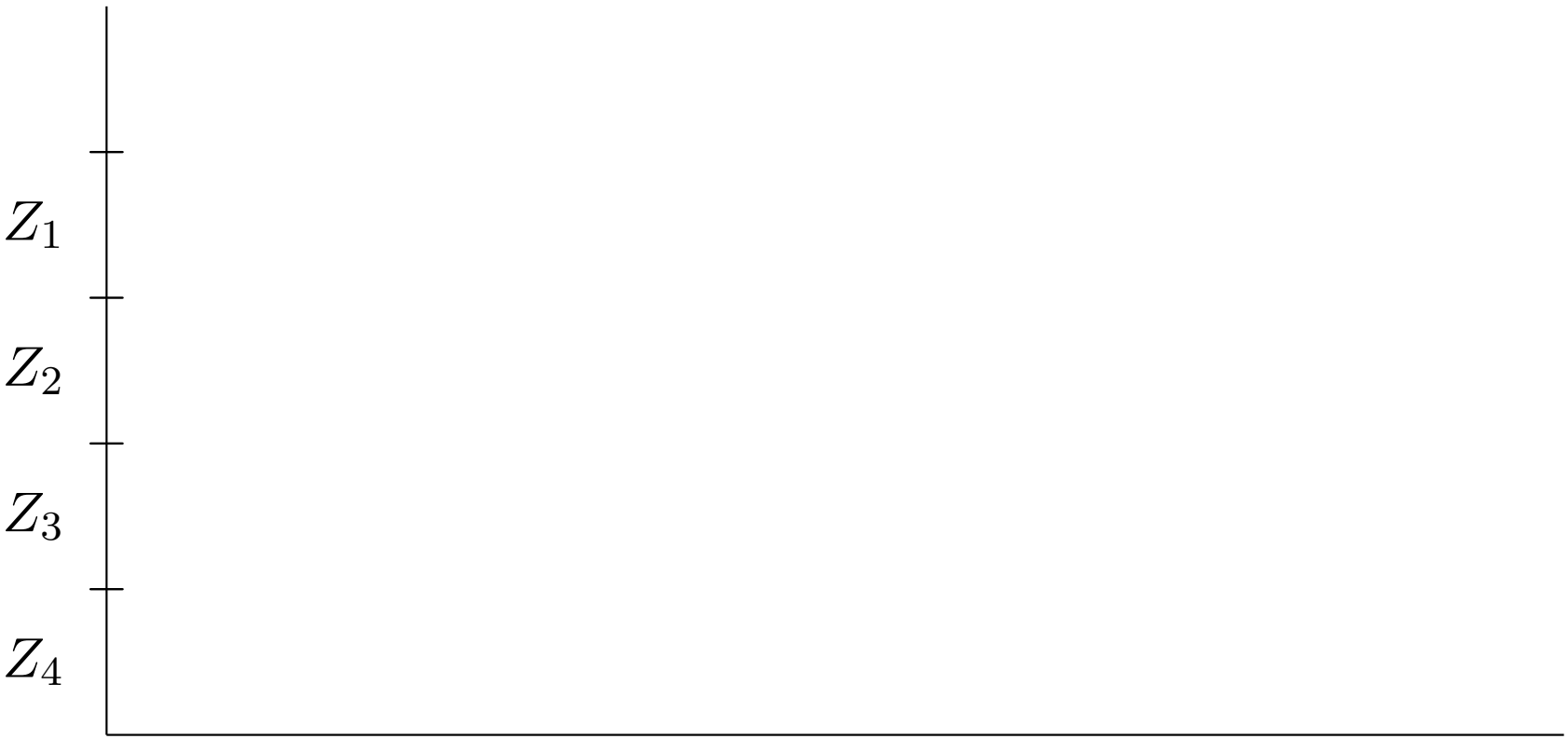
$$(q_i, u_i, \alpha_i) \to_M (q_{i+1}, u_{i+1}, \alpha_{i+1}) \implies$$
$$(q_i, u_i v, \alpha_i \beta) \to_M (q_{i+1}, u_{i+1} v, \alpha_{i+1} \beta)$$

Mit Induktion über n folgt die Behauptung. □

Gilt die Umkehrung, zB

$$(q_1, aa, XZ) \to_M^* (q_3, \epsilon, YZ) \implies (q_1, aa, X) \to_M^* (q_3, \epsilon, Y) \ ?$$

Keller



Schritte

Satz 4.56 (Zerlegungssatz)

Wenn $(q, w, Z_{1\ldots k}) \to_M^n (q', \epsilon, \epsilon)$
dann gibt es u_i, p_i, n_i, *so dass*

$$(p_{i-1}, u_i, Z_i) \to_M^{n_i} (p_i, \epsilon, \epsilon) \qquad (i = 1, \ldots, k)$$

und $w = u_1 \ldots u_k$, $p_0 = q$, $p_k = q'$, $\sum n_i = n$.

Satz 4.56 (Zerlegungssatz)

Wenn $(q, w, Z_{1\ldots k}) \to_M^n (q', \epsilon, \epsilon)$
dann gibt es u_i, p_i, n_i, *so dass*

$$(p_{i-1}, u_i, Z_i) \to_M^{n_i} (p_i, \epsilon, \epsilon) \qquad (i = 1, \ldots, k)$$

und $w = u_1 \ldots u_k$, $p_0 = q$, $p_k = q'$, $\sum n_i = n$.

Beweis: Mit Induktion über n. Basis trivial.

Satz 4.56 (Zerlegungssatz)

Wenn $(q, w, Z_{1\ldots k}) \to_M^n (q', \epsilon, \epsilon)$
dann gibt es u_i, p_i, n_i, *so dass*

$$(p_{i-1}, u_i, Z_i) \to_M^{n_i} (p_i, \epsilon, \epsilon) \qquad (i = 1, \ldots, k)$$

und $w = u_1 \ldots u_k$, $p_0 = q$, $p_k = q'$, $\sum n_i = n$.

Beweis: Mit Induktion über n. Basis trivial.

Schritt: Eine Berechnung der Länge $n+1$ hat die Form

$$(q, bw', Z_{1\ldots k}) \to_M (p, w', Y_{1\ldots l} Z_{2\ldots k}) \to_M^n (q', \epsilon, \epsilon)$$

Satz 4.56 (Zerlegungssatz)

Wenn $(q, w, Z_{1\ldots k}) \to_M^n (q', \epsilon, \epsilon)$
dann gibt es u_i, p_i, n_i, *so dass*

$$(p_{i-1}, u_i, Z_i) \to_M^{n_i} (p_i, \epsilon, \epsilon) \qquad (i = 1, \ldots, k)$$

und $w = u_1 \ldots u_k$, $p_0 = q$, $p_k = q'$, $\sum n_i = n$.

Beweis: Mit Induktion über n. Basis trivial.

Schritt: Eine Berechnung der Länge $n+1$ hat die Form

$$(q, bw', Z_{1\ldots k}) \to_M (p, w', Y_{1\ldots l} Z_{2\ldots k}) \to_M^n (q', \epsilon, \epsilon)$$

IA $\Rightarrow \exists v_j, r_j, m_j$ $(j = 1, \ldots, l)$, $\exists u_i, p_i, n_i$ $(i = 2, \ldots, k)$ mit

$$(r_{j-1}, v_j, Y_j) \to_M^{m_j} (r_j, \epsilon, \epsilon) \qquad (p_{i-1}, u_i, Z_i) \to_M^{n_i} (p_i, \epsilon, \epsilon)$$

und $w' = v_{1\ldots l} u_{2\ldots k}$, $r_0 = p$, $r_l = p_1$, $p_k = q'$, $\sum m_j + \sum n_i = n$.

Satz 4.56 (Zerlegungssatz)

Wenn $(q, w, Z_{1\ldots k}) \to_M^n (q', \epsilon, \epsilon)$
dann gibt es u_i, p_i, n_i, *so dass*

$$(p_{i-1}, u_i, Z_i) \to_M^{n_i} (p_i, \epsilon, \epsilon) \qquad (i = 1, \ldots, k)$$

und $w = u_1 \ldots u_k$, $p_0 = q$, $p_k = q'$, $\sum n_i = n$.

Beweis: Mit Induktion über n. Basis trivial.
Schritt: Eine Berechnung der Länge $n+1$ hat die Form

$$(q, bw', Z_{1\ldots k}) \to_M (p, w', Y_{1\ldots l} Z_{2\ldots k}) \to_M^n (q', \epsilon, \epsilon)$$

IA $\Rightarrow \exists v_j, r_j, m_j$ $(j = 1, \ldots, l)$, $\exists u_i, p_i, n_i$ $(i = 2, \ldots, k)$ mit

$$(r_{j-1}, v_j, Y_j) \to_M^{m_j} (r_j, \epsilon, \epsilon) \qquad (p_{i-1}, u_i, Z_i) \to_M^{n_i} (p_i, \epsilon, \epsilon)$$

und $w' = v_{1\ldots l} u_{2\ldots k}$, $r_0 = p$, $r_l = p_1$, $p_k = q'$, $\sum m_j + \sum n_i = n$.
Mit Erweiterungslemma: $(r_{j-1}, v_{j\ldots l}, Y_{j\ldots l}) \to_M^{m_j} (r_j, v_{j+1\ldots l}, Y_{j+1\ldots l})$
$\Rightarrow (r_0, v_{1\ldots l}, Y_{1\ldots l}) \to_M^{n_1 - 1} (r_l, \epsilon, \epsilon)$ wobei $n_1 := 1 + \sum m_j$.

Satz 4.56 (Zerlegungssatz)

Wenn $(q, w, Z_{1\ldots k}) \to_M^n (q', \epsilon, \epsilon)$
dann gibt es u_i, p_i, n_i, *so dass*

$$(p_{i-1}, u_i, Z_i) \to_M^{n_i} (p_i, \epsilon, \epsilon) \qquad (i = 1, \ldots, k)$$

und $w = u_1 \ldots u_k$, $p_0 = q$, $p_k = q'$, $\sum n_i = n$.

Beweis: Mit Induktion über n. Basis trivial.
Schritt: Eine Berechnung der Länge $n+1$ hat die Form

$$(q, bw', Z_{1\ldots k}) \to_M (p, w', Y_{1\ldots l} Z_{2\ldots k}) \to_M^n (q', \epsilon, \epsilon)$$

IA $\Rightarrow \exists v_j, r_j, m_j$ $(j = 1, \ldots, l)$, $\exists u_i, p_i, n_i$ $(i = 2, \ldots, k)$ mit

$$(r_{j-1}, v_j, Y_j) \to_M^{m_j} (r_j, \epsilon, \epsilon) \qquad (p_{i-1}, u_i, Z_i) \to_M^{n_i} (p_i, \epsilon, \epsilon)$$

und $w' = v_{1\ldots l} u_{2\ldots k}$, $r_0 = p$, $r_l = p_1$, $p_k = q'$, $\sum m_j + \sum n_i = n$.
Mit Erweiterungslemma: $(r_{j-1}, v_{j\ldots l}, Y_{j\ldots l}) \to_M^{m_j} (r_j, v_{j+1\ldots l}, Y_{j+1\ldots l})$
$\Rightarrow (r_0, v_{1\ldots l}, Y_{1\ldots l}) \to_M^{n_1 - 1} (r_l, \epsilon, \epsilon)$ wobei $n_1 := 1 + \sum m_j$.
Aus $(q, bv_{1\ldots l}, Z_1) \to_M (p, v_{1\ldots l}, Y_{1\ldots l})$ folgt $(p_0, u_1, Z_1) \to_M^{n_1} (p_1, \epsilon, \epsilon)$
wobei $p_0 := q$, $u_1 := bv_{1\ldots l}$, $p_1 := r_l$.

Satz 4.56 (Zerlegungssatz)

Wenn $(q, w, Z_{1\ldots k}) \to_M^n (q', \epsilon, \epsilon)$
dann gibt es u_i, p_i, n_i, *so dass*

$$(p_{i-1}, u_i, Z_i) \to_M^{n_i} (p_i, \epsilon, \epsilon) \qquad (i = 1, \ldots, k)$$

und $w = u_1 \ldots u_k$, $p_0 = q$, $p_k = q'$, $\sum n_i = n$.

Beweis: Mit Induktion über n. Basis trivial.

Schritt: Eine Berechnung der Länge $n+1$ hat die Form

$(q, bw', Z_{1\ldots k}) \to_M (p, w', Y_{1\ldots l} Z_{2\ldots k}) \to_M^n (q', \epsilon, \epsilon)$

IA $\Rightarrow \exists v_j, r_j, m_j$ $(j = 1, \ldots, l)$, $\exists u_i, p_i, n_i$ $(i = 2, \ldots, k)$ mit

$(r_{j-1}, v_j, Y_j) \to_M^{m_j} (r_j, \epsilon, \epsilon) \qquad (p_{i-1}, u_i, Z_i) \to_M^{n_i} (p_i, \epsilon, \epsilon)$

und $w' = v_{1\ldots l} u_{2\ldots k}$, $r_0 = p$, $r_l = p_1$, $p_k = q'$, $\sum m_j + \sum n_i = n$.

Mit Erweiterungslemma: $(r_{j-1}, v_{j\ldots l}, Y_{j\ldots l}) \to_M^{m_j} (r_j, v_{j+1\ldots l}, Y_{j+1\ldots l})$

$\Rightarrow (r_0, v_{1\ldots l}, Y_{1\ldots l}) \to_M^{n_1 - 1} (r_l, \epsilon, \epsilon)$ wobei $n_1 := 1 + \sum m_j$.

Aus $(q, bv_{1\ldots l}, Z_1) \to_M (p, v_{1\ldots l}, Y_{1\ldots l})$ folgt $(p_0, u_1, Z_1) \to_M^{n_1} (p_1, \epsilon, \epsilon)$
wobei $p_0 := q$, $u_1 := bv_{1\ldots l}$, $p_1 := r_l$.

Damit auch $w = bw' = bv_{1\ldots l} u_{2\ldots k} = u_{1\ldots k}$ und $\sum n_i = n + 1$. □

4.9 Äquivalenz von PDAs und CFGs

Satz 4.57 (CFG $\rightarrow$ PDA)

Zu jeder CFG G kann man einen PDA M konstruieren, der mit leerem Stack akzeptiert, so dass $L_\epsilon(M) = L(G)$.

4.9 Äquivalenz von PDAs und CFGs

Satz 4.57 (CFG $\rightarrow$ PDA)

Zu jeder CFG G kann man einen PDA M konstruieren, der mit leerem Stack akzeptiert, so dass $L_\epsilon(M) = L(G)$.

Konstruktion:

Zuerst bringen wir alle Produktionen von G in die Form

$$A \rightarrow bB_1 \dots B_k$$

wobei $b \in \Sigma \cup \{\epsilon\}$.

4.9 Äquivalenz von PDAs und CFGs

Satz 4.57 (CFG $\rightarrow$ PDA)

Zu jeder CFG G kann man einen PDA M konstruieren, der mit leerem Stack akzeptiert, so dass $L_\epsilon(M) = L(G)$.

Konstruktion:

Zuerst bringen wir alle Produktionen von G in die Form

$$A \rightarrow bB_1 \dots B_k$$

wobei $b \in \Sigma \cup \{\epsilon\}$.

Methode: Für jedes $a \in \Sigma$

1. füge ein neues A_a zu V hinzu,
2. ersetze a rechts in P durch A_a (außer am Kopfende),
3. füge eine neue Produkion $A_a \rightarrow a$ hinzu.

Alle Produktionen in $G = (V, \Sigma, P, S)$ haben jetzt die Form

$$A \to bB_1 \dots B_k$$

Der PDA wird wie folgt definiert:

$$M \quad := \quad (\{q\},\, \Sigma,\, V,\, q, S,\, \delta)$$

wobei

$$(A \to b\beta) \in P \implies \delta(q, b, A) \ni (q, \beta)$$

also für alle $b \in \Sigma \cup \{\epsilon\}$ und $A \in V$:

$$\delta(q, b, A) := \{(q, \beta) \mid (A \to b\beta) \in P\}$$

Lemma 4.58

Für alle $u, v \in \Sigma^*$ *und* $\gamma \in V^*$ *und* $A \in V$ *gilt:*

$$A \to_G^n u\gamma \text{ mit Linksableitung gdw } (q, uv, A) \to_M^n (q, v, \gamma)$$

Lemma 4.58

Für alle $u, v \in \Sigma^*$ und $\gamma \in V^*$ und $A \in V$ gilt:

$$A \to_G^n u\gamma \text{ mit Linksableitung gdw } (q, uv, A) \to_M^n (q, v, \gamma)$$

Beweis:

Mit Induktion über n. Basis $n = 0$ trivial.

Lemma 4.58

Für alle $u, v \in \Sigma^*$ *und* $\gamma \in V^*$ *und* $A \in V$ *gilt:*

$$A \to_G^n u\gamma \text{ mit Linksableitung gdw } (q, uv, A) \to_M^n (q, v, \gamma)$$

Beweis:

Mit Induktion über n. Basis $n = 0$ trivial. Schritt:

$$A \to_G^{n+1} u\gamma$$
$$\Leftrightarrow$$
$$\exists (B \to b\beta) \in P, w, \alpha.\ A \to_G^n wB\alpha \to_G wb\beta\alpha = u\gamma$$
$$\text{(dh } wb = u, \beta\alpha = \gamma)$$
$$\Leftrightarrow$$
$$(q, wbv, A) \to_M^n (q, bv, B\alpha) \wedge (q, bv, B\alpha) \to_M (q, v, \beta\alpha)$$
$$\Leftrightarrow$$
$$(q, uv, A) \to_M^{n+1} (q, v, \gamma)$$

□

$A \to_G^n u\gamma$ mit Linksableitung gdw $(q, uv, A) \to_M^n (q, v, \gamma)$

Satz 4.59

$L(G) = L_\epsilon(M)$

$A \to_G^n u\gamma$ mit Linksableitung gdw $(q, uv, A) \to_M^n (q, v, \gamma)$

Satz 4.59
$L(G) = L_\epsilon(M)$

Beweis:

$u \in L(G)$

$\Leftrightarrow S \to_G^* u$ mit Linksableitung

$\Leftrightarrow (q, u, S) \to_M^* (q, \epsilon, \epsilon)$

$\Leftrightarrow u \in L_\epsilon(M)$ □

Satz 4.60 (PDA $\rightarrow$ CFG)

Zu jedem PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$, der mit leerem Keller akzeptiert, kann man eine CFG G konstruieren mit $L(G) = L_\epsilon(M)$.

Satz 4.60 (PDA $\rightarrow$ CFG)

Zu jedem PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$, der mit leerem Keller akzeptiert, kann man eine CFG G konstruieren mit $L(G) = L_\epsilon(M)$.

Konstruktion: $G := (V, \Sigma, P, S)$ mit

$V := Q \times \Gamma \times Q \cup \{S\}$ wobei wir die Tripel mit $[.,.,.]$ notieren

und P die folgenden Produktionen enthält:

Satz 4.60 (PDA $\rightarrow$ CFG)

Zu jedem PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$, der mit leerem Keller akzeptiert, kann man eine CFG G konstruieren mit $L(G) = L_\epsilon(M)$.

Konstruktion: $G := (V, \Sigma, P, S)$ mit

$V := Q \times \Gamma \times Q \cup \{S\}$ wobei wir die Tripel mit $[.,.,.]$ notieren

und P die folgenden Produktionen enthält:

- $S \rightarrow [q_0, Z_0, q]$ für alle $q \in Q$
- Für alle $\delta(q, b, Z) \ni (r_0, Z_1 \dots Z_k)$ und für alle $r_1, \dots, r_k \in Q$:

$$[q, Z, r_k] \rightarrow b[r_0, Z_1, r_1][r_1, Z_2, r_2] \dots [r_{k-1}, Z_k, r_k]$$

Satz 4.60 (PDA $\to$ CFG)

Zu jedem PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$, der mit leerem Keller akzeptiert, kann man eine CFG G konstruieren mit $L(G) = L_\epsilon(M)$.

Konstruktion: $G := (V, \Sigma, P, S)$ mit

$V := Q \times \Gamma \times Q \cup \{S\}$ wobei wir die Tripel mit $[.,.,.]$ notieren

und P die folgenden Produktionen enthält:

- $S \to [q_0, Z_0, q]$ für alle $q \in Q$
- Für alle $\delta(q, b, Z) \ni (r_0, Z_1 \dots Z_k)$ und für alle $r_1, \dots, r_k \in Q$:

$$[q, Z, r_k] \to b[r_0, Z_1, r_1][r_1, Z_2, r_2] \dots [r_{k-1}, Z_k, r_k]$$

Idee: $[q, Z, r_k] \to_G^* w$ gdw $(q, w, Z) \to_M^* (r_k, \epsilon, \epsilon)$

Die $r_1, \dots, r_k$ sind potenzielle Zwischenzustände beim Akzeptieren der Teilwörter von $bu_1 \dots u_k = w$, die zu $Z_1, \dots, Z_k$ gehören. (Zerlegungssatz!)

Lemma 4.61

$[q, Z, p] \rightarrow_G^n w$ *gdw* $(q, w, Z) \rightarrow_M^n (p, \epsilon, \epsilon)$

Lemma 4.61

$[q, Z, p] \to_G^n w$ *gdw* $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$

Beweis:

($\Rightarrow$): Wenn $[q, Z, p] \to_G^n w$ dann $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$.
Mit Induktion über n.
IA: Beh. gilt für alle Werte $< n$. Zz: Beh. gilt für n.

Lemma 4.61

$[q, Z, p] \to_G^n w$ *gdw* $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$

Beweis:

($\Rightarrow$): Wenn $[q, Z, p] \to_G^n w$ dann $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$.
Mit Induktion über n.
IA: Beh. gilt für alle Werte $< n$. Zz: Beh. gilt für n.
Die Ableitung $[q, Z, p] \to_G^n w$ muss von folgender Form sein:

$$\begin{array}{lll} [q, Z, r_k] & \to_G & b[r_0, Z_1, r_1] \dots [r_{k-1}, Z_k, r_k] \\ & \to_G^{n-1} & bu_1 \dots u_k = w \end{array}$$

mit $r_k = p$, $(r_0, Z_1 \dots Z_k) \in \delta(q, b, Z)$,
$[r_{i-1}, Z_i, r_i] \to_G^{n_i} u_i$ $(i = 1, \dots, k)$ und $\sum n_i = n - 1$.

Lemma 4.61

$[q, Z, p] \to_G^n w$ *gdw* $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$

Beweis:

($\Rightarrow$): Wenn $[q, Z, p] \to_G^n w$ dann $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$.
Mit Induktion über n.
IA: Beh. gilt für alle Werte $< n$. Zz: Beh. gilt für n.
Die Ableitung $[q, Z, p] \to_G^n w$ muss von folgender Form sein:

$$\begin{array}{lll} [q, Z, r_k] & \to_G & b[r_0, Z_1, r_1] \dots [r_{k-1}, Z_k, r_k] \\ & \to_G^{n-1} & bu_1 \dots u_k = w \end{array}$$

mit $r_k = p$, $(r_0, Z_1 \dots Z_k) \in \delta(q, b, Z)$,
$[r_{i-1}, Z_i, r_i] \to_G^{n_i} u_i$ $(i = 1, \dots, k)$ und $\sum n_i = n - 1$.
IA $\Rightarrow (r_{i-1}, u_i, Z_i) \to_M^{n_i} (r_i, \epsilon, \epsilon)$

Lemma 4.61

$[q, Z, p] \to_G^n w$ *gdw* $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$

Beweis:

($\Rightarrow$): Wenn $[q, Z, p] \to_G^n w$ dann $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$.
Mit Induktion über n.
IA: Beh. gilt für alle Werte $< n$. Zz: Beh. gilt für n.
Die Ableitung $[q, Z, p] \to_G^n w$ muss von folgender Form sein:

$$\begin{array}{lll} [q, Z, r_k] & \to_G & b[r_0, Z_1, r_1] \dots [r_{k-1}, Z_k, r_k] \\ & \to_G^{n-1} & bu_1 \dots u_k = w \end{array}$$

mit $r_k = p$, $(r_0, Z_1 \dots Z_k) \in \delta(q, b, Z)$,
$[r_{i-1}, Z_i, r_i] \to_G^{n_i} u_i$ $(i = 1, \dots, k)$ und $\sum n_i = n - 1$.
IA $\Rightarrow (r_{i-1}, u_i, Z_i) \to_M^{n_i} (r_i, \epsilon, \epsilon)$
Erweiterungslemma
$\Rightarrow (r_{i-1}, u_i \dots u_k, Z_i \dots Z_k) \to_M^{n_i} (r_i, u_{i+1} \dots u_k, Z_{i+1} \dots Z_k)$
$\Rightarrow (q, w, Z) \to_M (r_0, u_1 \dots u_k, Z_1 \dots Z_k) \to_M^{n-1} (r_k, \epsilon, \epsilon)$ □

Beweis (Forts.):

($\Leftarrow$): Wenn $(q, w, Z) \rightarrow_M^n (p, \epsilon, \epsilon)$ dann $[q, Z, p] \rightarrow_G^n w$.
Mit Induktion über n.

Beweis (Forts.):

($\Leftarrow$): Wenn $(q, w, Z) \rightarrow_M^n (p, \epsilon, \epsilon)$ dann $[q, Z, p] \rightarrow_G^n w$.
Mit Induktion über n.
IA: Beh. gilt für alle Werte $< n$. Zz: Beh. gilt für n.

Beweis (Forts.):

($\Leftarrow$): Wenn $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$ dann $[q, Z, p] \to_G^n w$.
Mit Induktion über n.
IA: Beh. gilt für alle Werte $< n$. Zz: Beh. gilt für n.
Die Berechnung $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$ muss von dieser Form sein:

$$(q, w, Z) \to_M (r_0, w', Z_1 \dots Z_k) \to_M^{n-1} (p, \epsilon, \epsilon)$$

mit $w = bw'$, $(r_0, Z_1 \dots Z_k) \in \delta(q, b, Z)$.

Beweis (Forts.):

($\Leftarrow$): Wenn $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$ dann $[q, Z, p] \to_G^n w$.
Mit Induktion über n.
IA: Beh. gilt für alle Werte $< n$. Zz: Beh. gilt für n.
Die Berechnung $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$ muss von dieser Form sein:

$$(q, w, Z) \to_M (r_0, w', Z_1 \dots Z_k) \to_M^{n-1} (p, \epsilon, \epsilon)$$

mit $w = bw'$, $(r_0, Z_1 \dots Z_k) \in \delta(q, b, Z)$.
Nach dem Zerlegungssatz muss es r_i, u_i und n_i geben mit

$$(r_{i-1}, u_i, Z_i) \to_M^{n_i} (r_i, \epsilon, \epsilon) \qquad (i = 1, \dots, k)$$

und $w' = u_1 \dots u_k$, $\sum n_i = n - 1$.

Beweis (Forts.):

($\Leftarrow$): Wenn $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$ dann $[q, Z, p] \to_G^n w$.
Mit Induktion über n.
IA: Beh. gilt für alle Werte $< n$. Zz: Beh. gilt für n.
Die Berechnung $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$ muss von dieser Form sein:

$$(q, w, Z) \to_M (r_0, w', Z_1 \dots Z_k) \to_M^{n-1} (p, \epsilon, \epsilon)$$

mit $w = bw'$, $(r_0, Z_1 \dots Z_k) \in \delta(q, b, Z)$.
Nach dem Zerlegungssatz muss es r_i, u_i und n_i geben mit

$$(r_{i-1}, u_i, Z_i) \to_M^{n_i} (r_i, \epsilon, \epsilon) \qquad (i = 1, \dots, k)$$

und $w' = u_1 \dots u_k$, $\sum n_i = n - 1$.
IA $\Rightarrow$ $[r_{i-1}, Z_i, r_i] \to_G^{n_i} u_i$

Beweis (Forts.):

($\Leftarrow$): Wenn $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$ dann $[q, Z, p] \to_G^n w$.
Mit Induktion über n.
IA: Beh. gilt für alle Werte $< n$. Zz: Beh. gilt für n.
Die Berechnung $(q, w, Z) \to_M^n (p, \epsilon, \epsilon)$ muss von dieser Form sein:

$$(q, w, Z) \to_M (r_0, w', Z_1 \dots Z_k) \to_M^{n-1} (p, \epsilon, \epsilon)$$

mit $w = bw'$, $(r_0, Z_1 \dots Z_k) \in \delta(q, b, Z)$.
Nach dem Zerlegungssatz muss es r_i, u_i und n_i geben mit

$$(r_{i-1}, u_i, Z_i) \to_M^{n_i} (r_i, \epsilon, \epsilon) \qquad (i = 1, \dots, k)$$

und $w' = u_1 \dots u_k$, $\sum n_i = n - 1$.

$$\begin{array}{lcll} \text{IA} & \Rightarrow & [r_{i-1}, Z_i, r_i] \to_G^{n_i} u_i & \\ & \Rightarrow & [q, Z, p] & \to_G \quad b[r_0, Z_1, r_1] \dots [r_{k-1}, Z_k, r_k] \\ & & & \to_G^{n-1} \quad bu_1 \dots u_k = w \end{array}$$

□

Damit haben wir bewiesen:

Satz 4.62

Eine Sprache ist kontextfrei gdw sie von einem Kellerautomaten akzeptiert wird.

4.10 Deterministische Kellerautomaten

Definition 4.63

Ein PDA heißt deterministisch (DPDA) gdw für alle $q \in Q$, $a \in \Sigma$ und $Z \in \Gamma$

$$|\delta(q, a, Z)| + |\delta(q, \epsilon, Z)| \leq 1$$

4.10 Deterministische Kellerautomaten

Definition 4.63

Ein PDA heißt deterministisch (DPDA) gdw für alle $q \in Q$, $a \in \Sigma$ und $Z \in \Gamma$

$$|\delta(q, a, Z)| + |\delta(q, \epsilon, Z)| \leq 1$$

Beispiel 4.64

Der folgende DPDA mit $\Sigma = \{0, 1, \$\}$, $\Gamma = \{Z_0, 0, 1\}$ akzeptiert die Sprache $\{w\$w^R \mid w \in \{0, 1\}^*\}$.

Definition 4.65

Eine CFL heißt deterministisch (DCFL) gdw sie von einem DPDA akzeptiert wird.

Man kann zeigen: Die CFL $\{ww^R \mid w \in \{0,1\}^*\}$ ist nicht deterministisch.

Definition 4.65

Eine CFL heißt deterministisch (DCFL) gdw sie von einem DPDA akzeptiert wird.

Man kann zeigen: Die CFL $\{ww^R \mid w \in \{0,1\}^*\}$ ist nicht deterministisch.

Da man jeden DFA leicht mit einem DPDA simulieren kann:

Fakt 4.66

Jede reguläre Sprache ist eine DCFL.

Eine Sprache erfüllt die Präfix Bedingung gdw wenn sie keine zwei Wörter enthält, so dass das eine ein *echtes* Präfix des anderen ist.

Beispiel 4.67

- Die Sprache a^* erfüllt die Präfix Bedingung nicht.
- Die Sprache $\{w\$w^R \mid w \in \{0,1\}^*\}$ erfüllt die Präfix Bedingung.

Eine Sprache erfüllt die Präfix Bedingung gdw wenn sie keine zwei Wörter enthält, so dass das eine ein *echtes* Präfix des anderen ist.

Beispiel 4.67

- Die Sprache a^* erfüllt die Präfix Bedingung nicht.
- Die Sprache $\{w\$w^R \mid w \in \{0,1\}^*\}$ erfüllt die Präfix Bedingung.

Lemma 4.68

$\exists$*DPDA* M. $L = L_\epsilon(M) \quad \Leftrightarrow$
$\exists$*DPDA* M. $L = L_F(M)$ *und* L *erfüllt die Präfix Bedingung*

Beweis: Übung!

Eine Sprache erfüllt die Präfix Bedingung gdw wenn sie keine zwei Wörter enthält, so dass das eine ein *echtes* Präfix des anderen ist.

Beispiel 4.67

- Die Sprache a^* erfüllt die Präfix Bedingung nicht.
- Die Sprache $\{w\$w^R \mid w \in \{0,1\}^*\}$ erfüllt die Präfix Bedingung.

Lemma 4.68

$\exists$*DPDA* M. $L = L_\epsilon(M) \quad \Leftrightarrow$
$\exists$*DPDA* M. $L = L_F(M)$ *und* L *erfüllt die Präfix Bedingung*

Beweis: Übung!

Es gibt also einen DPDA M mit $L_F(M) = L(a^*)$ aber keinen DPDA mit $L_\epsilon(M) = L(a^*)$.
Im Folgenden: Akzeptanz durch Endzustand.

Satz 4.69

Die Klasse der DCFLs ist unter Komplement abgeschlossen.

Beweis: Komplex. Siehe zB Erk&Priese oder Kozen.

Da die CFLs *nicht* unter Komplement abgeschlossen sind:

Korollar 4.70

Die DCFLs sind eine echte Teilklasse der CFLs.

Satz 4.69

Die Klasse der DCFLs ist unter Komplement abgeschlossen.

Beweis: Komplex. Siehe zB Erk&Priese oder Kozen.

Da die CFLs *nicht* unter Komplement abgeschlossen sind:

Korollar 4.70

Die DCFLs sind eine echte Teilklasse der CFLs.

Hier ist eine konkrete Sprache in CFL\DCFL:
Sei $L_{ww} := \{ww \mid w \in \{a, b\}^*\}$.

Satz 4.69

Die Klasse der DCFLs ist unter Komplement abgeschlossen.

Beweis: Komplex. Siehe zB Erk&Priese oder Kozen.

Da die CFLs *nicht* unter Komplement abgeschlossen sind:

Korollar 4.70

Die DCFLs sind eine echte Teilklasse der CFLs.

Hier ist eine konkrete Sprache in CFL\DCFL:
Sei $L_{ww} := \{ww \mid w \in \{a,b\}^*\}$.
Dann ist $L := \{a,b\}^* \setminus L_{ww}$ eine CFL aber keine DCFL.
L wird von folgender CFG erzeugt (ohne Beweis):

$$S \to AB \mid BA \mid A \mid B$$

$$A \to CAC \mid a \qquad B \to CBC \mid b \qquad C \to a \mid b$$

Wäre L eine DCFL, dann auch $\overline{L} = L_{ww}$ (wegen Satz 4.69).
Aber mit dem Pumping Lemma kann man zeigen, dass L_{ww} keine CFL ist, und daher erst recht keine DCFL.

Lemma 4.71

Die Klasse der DCFLs ist weder unter Schnitt noch unter Vereinigung abgeschlossen.

Lemma 4.71

Die Klasse der DCFLs ist weder unter Schnitt noch unter Vereinigung abgeschlossen.

Beweis:

Erinnerung: CFLs nicht unter Schnitt abgeschlossen:

$$\begin{aligned} L_1 &:= \{a^i b^i c^j \mid i,j \in \mathbb{N}\} \text{ ist kontextfrei} \\ L_2 &:= \{a^i b^j c^j \mid i,j \in \mathbb{N}\} \text{ ist kontextfrei} \end{aligned}$$

Lemma 4.71

Die Klasse der DCFLs ist weder unter Schnitt noch unter Vereinigung abgeschlossen.

Beweis:

Erinnerung: CFLs nicht unter Schnitt abgeschlossen:

$$\begin{aligned} L_1 &:= \{a^i b^i c^j \mid i,j \in \mathbb{N}\} \text{ ist kontextfrei} \\ L_2 &:= \{a^i b^j c^j \mid i,j \in \mathbb{N}\} \text{ ist kontextfrei} \\ L_1 \cap L_2 &= \{a^i b^i c^i \mid i \in \mathbb{N}\} \text{ ist nicht kontextfrei} \end{aligned}$$

Aber L_1 und L_2 sind sogar DCFLs.

Lemma 4.71

Die Klasse der DCFLs ist weder unter Schnitt noch unter Vereinigung abgeschlossen.

Beweis:

Erinnerung: CFLs nicht unter Schnitt abgeschlossen:

$$\begin{aligned} L_1 &:= \{a^i b^i c^j \mid i,j \in \mathbb{N}\} \text{ ist kontextfrei} \\ L_2 &:= \{a^i b^j c^j \mid i,j \in \mathbb{N}\} \text{ ist kontextfrei} \\ L_1 \cap L_2 &= \{a^i b^i c^i \mid i \in \mathbb{N}\} \text{ ist nicht kontextfrei} \end{aligned}$$

Aber L_1 und L_2 sind sogar DCFLs.

Da $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ können die DCFLs auch nicht unter Vereinigung abgeschlossen sein. □

Lemma 4.72

Jede DCFL ist nicht inhärent mehrdeutig, dh sie wird von einer nicht-mehrdeutigen Grammatik erzeugt.

Beweisidee: Die Konversion PDA $\rightarrow$ CFG erzeugt aus einem DPDA eine nicht-mehrdeutige CFG. [HMU]

Lemma 4.72

Jede DCFL ist nicht inhärent mehrdeutig, dh sie wird von einer nicht-mehrdeutigen Grammatik erzeugt.

Beweisidee: Die Konversion PDA $\rightarrow$ CFG erzeugt aus einem DPDA eine nicht-mehrdeutige CFG. [HMU]

Satz 4.73

Das Wortproblem für DCFLs ist in linearer Zeit lösbar.

Lemma 4.72

Jede DCFL ist nicht inhärent mehrdeutig, dh sie wird von einer nicht-mehrdeutigen Grammatik erzeugt.

Beweisidee: Die Konversion PDA $\rightarrow$ CFG erzeugt aus einem DPDA eine nicht-mehrdeutige CFG. [HMU]

Satz 4.73

Das Wortproblem für DCFLs ist in linearer Zeit lösbar.

Mehr Information: Vorlesungen zum Übersetzerbau

4.11 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern

4.11 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Regulär	ja	ja	ja	ja	ja

4.11 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Regulär	ja	ja	ja	ja	ja
DCFL	nein	nein	ja	nein	nein

4.11 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Regulär	ja	ja	ja	ja	ja
DCFL	nein	nein	ja	nein	nein
CFL	nein	ja	nein	ja	ja

4.11 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Regulär	ja	ja	ja	ja	ja
DCFL	nein	nein	ja	nein	nein
CFL	nein	ja	nein	ja	ja

Entscheidbarkeit

	Wortproblem			
DFA				
DPDA				
CFG				

4.11 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Regulär	ja	ja	ja	ja	ja
DCFL	nein	nein	ja	nein	nein
CFL	nein	ja	nein	ja	ja

Entscheidbarkeit

	Wortproblem			
DFA	$O(n)$			
DPDA	$O(n)$			
CFG	$O(n^3)$			

4.11 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Regulär	ja	ja	ja	ja	ja
DCFL	nein	nein	ja	nein	nein
CFL	nein	ja	nein	ja	ja

Entscheidbarkeit

	Wortproblem	Leerheit		
DFA	$O(n)$			
DPDA	$O(n)$			
CFG	$O(n^3)$			

4.11 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Regulär	ja	ja	ja	ja	ja
DCFL	nein	nein	ja	nein	nein
CFL	nein	ja	nein	ja	ja

Entscheidbarkeit

	Wortproblem	Leerheit		
DFA	$O(n)$	ja		
DPDA	$O(n)$	ja		
CFG	$O(n^3)$	ja		

4.11 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Regulär	ja	ja	ja	ja	ja
DCFL	nein	nein	ja	nein	nein
CFL	nein	ja	nein	ja	ja

Entscheidbarkeit

	Wortproblem	Leerheit	Äquivalenz	
DFA	$O(n)$	ja		
DPDA	$O(n)$	ja		
CFG	$O(n^3)$	ja		

4.11 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Regulär	ja	ja	ja	ja	ja
DCFL	nein	nein	ja	nein	nein
CFL	nein	ja	nein	ja	ja

Entscheidbarkeit

	Wortproblem	Leerheit	Äquivalenz	
DFA	$O(n)$	ja	ja	
DPDA	$O(n)$	ja	ja	
CFG	$O(n^3)$	ja	nein(*)	

Sénizergues (1997), Stirling (2001)

(*) Vorschau

4.11 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Regulär	ja	ja	ja	ja	ja
DCFL	nein	nein	ja	nein	nein
CFL	nein	ja	nein	ja	ja

Entscheidbarkeit

	Wortproblem	Leerheit	Äquivalenz	Schnittproblem
DFA	$O(n)$	ja	ja	
DPDA	$O(n)$	ja	ja	
CFG	$O(n^3)$	ja	nein(*)	

Sénizergues (1997), Stirling (2001)

(*) Vorschau

4.11 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Regulär	ja	ja	ja	ja	ja
DCFL	nein	nein	ja	nein	nein
CFL	nein	ja	nein	ja	ja

Entscheidbarkeit

	Wortproblem	Leerheit	Äquivalenz	Schnittproblem
DFA	$O(n)$	ja	ja	ja
DPDA	$O(n)$	ja	ja	nein(*)
CFG	$O(n^3)$	ja	nein(*)	nein(*)

Sénizergues (1997), Stirling (2001)

(*) Vorschau