Script generated by TTT

Title: Seidl: Theoretische_Informatik (16.04.2012)

Date: Mon Apr 16 10:17:42 CEST 2012

Duration: 90:19 min

Pages: 84

Einführung in die Theoretische Informatik

Tobias Nipkow + Helmut Seidl

Fakultät für Informatik
TU München

http://theo.in.tum.de

Sommersemester 2012

Kapitel 0 Organisatorisches

Siehe
http://www2.in.tum.de/hp/Main?nid=194

- Vorkenntnisse:
  - Einführung in die Informatik 1/2
  - Diskrete Strukturen

- Vorkenntnisse:
  - Einführung in die Informatik 1/2
  - Diskrete Strukturen
- Weiterführende Vorlesungen:
  - Automaten, Formale Sprachen, Berechenbarkeit und Entscheidbarkeit
  - Komplexitätstheorie
  - Logik
  - Compilerbau
  - ...

1. Themen und Ziel der Vorlesung

Themengebiete:

- Berechenbarkeitstheorie

1. Themen und Ziel der Vorlesung

Themengebiete:

- Berechenbarkeitstheorie
  - Untersuchung der Grenzen, was Rechner prinzipiell können

1. Themen und Ziel der Vorlesung

Themengebiete:

- Berechenbarkeitstheorie
  - Untersuchung der Grenzen, was Rechner prinzipiell können
- Komplexitätstheorie

1. Themen und Ziel der Vorlesung

Themengebiete:

- Berechenbarkeitstheorie
  - Untersuchung der Grenzen, was Rechner prinzipiell können
- Komplexitätstheorie
  - Untersuchung der Grenzen, was Rechner mit begrenzten Ressourcen können

1. Themen und Ziel der Vorlesung

Themengebiete:

- Berechenbarkeitstheorie
  - Untersuchung der Grenzen, was Rechner prinzipiell können
- Komplexitätstheorie
  - Untersuchung der Grenzen, was Rechner mit begrenzten Ressourcen können
- Automatentheorie

1. Themen und Ziel der Vorlesung

Themengebiete:

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

1. Themen und Ziel der Vorlesung

Themengebiete:

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

1. Themen und Ziel der Vorlesung

Themengebiete:

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

Geschichte:

1936 Berechenbarkeitstheorie: Church & Turing

Geschichte:

1936 Berechenbarkeitstheorie: Church & Turing

1956 Automaten und reguläre Ausdrücke: Kleene

Geschichte:

1936 Berechenbarkeitstheorie: Church & Turing

1956 Automaten und reguläre Ausdrücke: Kleene

1956 Grammatiken: Chomsky

Geschichte:

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

Gliederung (1. & 2. Semesterhälfte):

1. Automaten und formale Sprachen
   - Endliche Automaten und reguläre Ausdrücke (`rm *.pdf`)

Gliederung (1. & 2. Semesterhälfte):

1. Automaten und formale Sprachen
   - Endliche Automaten und reguläre Ausdrücke (`rm *.pdf`)

Gliederung (1. & 2. Semesterhälfte):

1. Automaten und formale Sprachen
   - Endliche Automaten und reguläre Ausdrücke (`rm *.pdf`)
   - Kontextfreie Grammatiken (`<A> ::= <A> + <A> | <A> * <A> | ...`)

Gliederung (1. & 2. Semesterhälfte):

1. Automaten und formale Sprachen
   - Endliche Automaten und reguläre Ausdrücke (`rm *.pdf`)
   - Kontextfreie Grammatiken (`<A> ::= <A> + <A> | <A> * <A> | ...`)
2. Berechenbarkeit

Gliederung (1. & 2. Semesterhälfte):

1. Automaten und formale Sprachen
   - Endliche Automaten und reguläre Ausdrücke (`rm *.pdf`)
   - Kontextfreie Grammatiken (`<A> ::= <A> + <A> | <A> * <A> | ...`)
2. Berechenbarkeit
   - Turing Maschinen, Berechenbarkeit, Entscheidbarkeit, Aufzählbarkeit

Gliederung (1. & 2. Semesterhälfte):

1. Automaten und formale Sprachen
   - Endliche Automaten und reguläre Ausdrücke (`rm *.pdf`)
   - Kontextfreie Grammatiken (`<A> ::= <A> + <A> | <A> * <A> | ...`)
2. Berechenbarkeit
   - Turing Maschinen, Berechenbarkeit, Entscheidbarkeit, Aufzählbarkeit
   - Komplexitätsklassen (P und NP)

Ziele:

Abstraktion von *irrelevanten* Details:

Ziele:

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

Ziele:

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

Formalisierung durch mathematische Objekte (Mengen, Funktionen, Relationen)

Ziele:

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

Formalisierung durch mathematische Objekte (Mengen, Funktionen, Relationen)

Simulation eines Formalismus durch einen anderen:

Ziele:

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

Ziele:

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:

Ziele:

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.

Ziele:

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:

Ziele:

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, ...

Ziele:

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

Ziele:

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

**Informatik ist keine (schwarze) Kunst
sondern eine exakte Wissenschaft.**

2. Literatur

- John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman.
  Introduction to Automata Theory, Languages, and Computation.
- Dexter Kozen.
  Automata and Computability.
- Katrin Erk, Lutz Priese.
  Theoretische Informatik: Eine umfassende Einführung.
- Uwe Schöning.
  Theoretische Informatik — kurzgefasst.

Kapitel I Formale Sprachen und Automaten

1. Grundbegriffe

Definition 1.1

- Ein Alphabet Σ ist eine endliche Menge.

Kapitel I Formale Sprachen und Automaten

1. Grundbegriffe

Definition 1.1

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

Kapitel I Formale Sprachen und Automaten

1. Grundbegriffe

Definition 1.1

- 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.

Kapitel I Formale Sprachen und Automaten

1. Grundbegriffe

Definition 1.1

- 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.
- Das leere Wort wird mit ϵ bezeichnet.

Kapitel I Formale Sprachen und Automaten

1. Grundbegriffe

Definition 1.1

- 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.
- Das leere Wort wird mit ϵ bezeichnet.
- Sind u und v Wörter, so ist uv ihre Konkatenation.

Kapitel I Formale Sprachen und Automaten

1. Grundbegriffe

Definition 1.1

- 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.
- Das leere Wort 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 $w^{n+1} = uw^n$.

Kapitel I Formale Sprachen und Automaten

1. Grundbegriffe

Definition 1.1

- 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.
- Das leere Wort 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: $(ab)^3 = ababab$.

Kapitel I Formale Sprachen und Automaten

1. Grundbegriffe

Definition 1.1

- 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.
- Das leere Wort 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: $(ab)^3 = ababab$.

- $|w|$ ist die Länge eines Wortes w.

Kapitel I Formale Sprachen und Automaten

1. Grundbegriffe

Definition 1.1

- 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.
- Das leere Wort 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: $(ab)^3 = ababab$.
- $|w|$ ist die Länge eines Wortes w.

- Σ^* ist die Menge aller Wörter über Σ.

Kapitel I Formale Sprachen und Automaten

1. Grundbegriffe

Definition 1.1

- 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.
- Das leere Wort 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: $(ab)^3 = ababab$.
- $|w|$ ist die Länge eines Wortes w.
- Σ^* ist die Menge aller Wörter über Σ.

Kapitel I Formale Sprachen und Automaten

1. Grundbegriffe

Definition 1.1

- 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.
- Das leere Wort 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: $(ab)^3 = ababab$.
- $|w|$ ist die Länge eines Wortes w.
- Σ^* ist die Menge aller Wörter über Σ.
- Eine Teilmenge $L \subseteq \Sigma^*$ ist eine (formale) Sprache.

Beispiel 1.2 (Formale Sprachen)

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

Beispiel 1.2 (Formale Sprachen)

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

Beispiel 1.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die „Menge“ der deutschen Sätze ist keine formale Sprache
- Die Menge aller legalen Java 1.5 Programme ist keine formale Sprache

Beispiel 1.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die „Menge“ der deutschen Sätze ist keine formale Sprache
- Die Menge aller legalen Java 1.5 Programme ist keine formale Sprache
- $L_1 = \{\epsilon, ab, abab, ababab, \dots\} = \{(ab)^n \mid n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a, b\})$

Beispiel 1.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die „Menge“ der deutschen Sätze ist keine formale Sprache
- Die Menge aller legalen Java 1.5 Programme ist keine formale Sprache
- $L_1 = \{\epsilon, ab, abab, ababab, \dots\} = \{(ab)^n \mid n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a, b\})$
- $L_2 = \{\epsilon, ab, aabb, aaabbb, \dots\} = \{ \quad \mid n \in \mathbb{N}\}$
  $(\Sigma_2 = \{a, b\})$

Beispiel 1.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die „Menge“ der deutschen Sätze ist keine formale Sprache
- Die Menge aller legalen Java 1.5 Programme ist keine formale Sprache
- $L_1 = \{\epsilon, ab, abab, ababab, \dots\} = \{(ab)^n \mid n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a, b\})$
- $L_2 = \{\epsilon, ab, aabb, aaabbb, \dots\} = \{a^n b^n \mid n \in \mathbb{N}\}$
  $(\Sigma_2 = \{a, b\})$
- $L_3 = \{\epsilon, 1, 100, 1001, 10000, \dots\} =$
  $\{w \in \{0, 1\}^* \mid \qquad \}$
  $(\Sigma_3 = \{0, 1\})$

Beispiel 1.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die „Menge“ der deutschen Sätze ist keine formale Sprache
- Die Menge aller legalen Java 1.5 Programme 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 1.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die „Menge“ der deutschen Sätze ist keine formale Sprache
- Die Menge aller legalen Java 1.5 Programme 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 1.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die „Menge“ der deutschen Sätze ist keine formale Sprache
- Die Menge aller legalen Java 1.5 Programme 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 1.2 (Formale Sprachen)

- Die Menge aller Wörter im Duden (24. Aufl.)
- Die „Menge“ der deutschen Sätze ist keine formale Sprache
- Die Menge aller legalen Java 1.5 Programme 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 1.3 (Operationen auf Sprachen)

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

Definition 1.3 (Operationen auf Sprachen)

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

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$

Definition 1.3 (Operationen auf Sprachen)

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

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} =$

Definition 1.3 (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 1.3 (Operationen auf Sprachen)

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

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

Definition 1.3 (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 1.3 (Operationen auf Sprachen)

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

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

- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\}$

Definition 1.3 (Operationen auf Sprachen)

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

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

  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$

Definition 1.3 (Operationen auf Sprachen)

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

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = A \cdots A$
  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\}$

Definition 1.3 (Operationen auf Sprachen)

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

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = A \cdots A$
  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$

Definition 1.3 (Operationen auf Sprachen)

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

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = A \cdots A$
  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \dots\}$

Definition 1.3 (Operationen auf Sprachen)

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

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = A \cdots A$
  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \dots\} \neq \{0, 1\}^*$

Definition 1.3 (Operationen auf Sprachen)

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

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = A \cdots A$
  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \dots\} \neq \{0, 1\}^*$

- $A^+ = AA^*$

Definition 1.3 (Operationen auf Sprachen)

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

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = A \cdots A$
  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \dots\} \neq \{0, 1\}^*$
- $A^+ = AA^* = \bigcup_{n \geq 1} A^n$

Definition 1.3 (Operationen auf Sprachen)

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

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = A \cdots A$
  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \dots\} \neq \{0, 1\}^*$
- $A^+ = AA^* = \bigcup_{n \geq 1} A^n$
  Bsp: $\Sigma^+ =$ Menge aller nicht-leeren Wörter über Σ

Definition 1.3 (Operationen auf Sprachen)

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

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = A \cdots A$
  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \dots\} \neq \{0, 1\}^*$
- $A^+ = AA^* = \bigcup_{n \geq 1} A^n$
  Bsp: $\Sigma^+ =$ Menge aller nicht-leeren Wörter über Σ

Achtung:

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

Definition 1.3 (Operationen auf Sprachen)

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

- Konkatenation: $AB = \{uv \mid u \in A \wedge v \in B\}$
  Bsp: $\{ab, b\}\{a, bb\} = \{aba, abbb, ba, bbb\}$
  NB: $\{ab, b\} \times \{a, bb\} = \{(ab, a), (ab, bb), (b, a), (b, bb)\}$
- $A^n = \{w_1 \dots w_n \mid w_1, \dots, w_n \in A\} = A \cdots A$
  Bsp: $\{ab, ba\}^2 = \{abab, abba, baab, baba\}$
  Rekursiv: $A^0 = \{\epsilon\}$ und $A^{n+1} = AA^n$
- $A^* = \{w_1 \dots w_n \mid n \geq 0 \wedge w_1, \dots, w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
  Bsp: $\{01\}^* = \{\epsilon, 01, 0101, 010101, \dots\} \neq \{0, 1\}^*$
- $A^+ = AA^* = \bigcup_{n \geq 1} A^n$
  Bsp: $\Sigma^+ =$ Menge aller nicht-leeren Wörter über Σ

Achtung:

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

Einige Rechenregeln:

Lemma 1.4

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

$\{\varepsilon u \mid u \in A\} = \{u \mid u \in A\} = A$

Einige Rechenregeln:

Lemma 1.4

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

Lemma 1.5

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

Einige Rechenregeln:

Lemma 1.4

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

Lemma 1.5

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

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

$A = \{a\}^* \quad B = \{\varepsilon\} \quad C = \{a\}$

$A(B \cap C) = A\emptyset = \emptyset$

$AB \cap AC = A \cap a^+ = a^+$

Einige Rechenregeln:

Lemma 1.4

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

Lemma 1.5

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

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

Lemma 1.6
$A^*A^* = A^*$

$A^* \subseteq A^* A^*$

$\varepsilon \in A^*$

$A^* = \{\varepsilon\} A^* \subseteq A^* A^*$

Erinnerung: Eine Menge M ist abzählbar falls sie gleich mächtig wie eine Teilmenge von $\mathbb{N}$ ist,

Erinnerung: Eine Menge M ist abzählbar falls sie gleich mächtig wie eine Teilmenge von $\mathbb{N}$ ist,

- d.h., es gibt eine Bijektion zw. M und einer Teilmenge von $\mathbb{N}$,
- d.h., es gibt eine Nummerierung der Elemente von M so dass $M = \{m_0, m_1, \dots\}$.

Erinnerung: Eine Menge M ist abzählbar falls sie gleich mächtig wie eine Teilmenge von $\mathbb{N}$ ist,

- d.h., es gibt eine Bijektion zw. M und einer Teilmenge von $\mathbb{N}$,
- d.h., es gibt eine Nummerierung der Elemente von M so dass $M = \{m_0, m_1, \dots\}$.

Eine Menge ist überabzählbar wenn sie nicht abzählbar ist.

Erinnerung: Eine Menge M ist abzählbar falls sie gleich mächtig wie eine Teilmenge von $\mathbb{N}$ ist,

- d.h., es gibt eine Bijektion zw. M und einer Teilmenge von $\mathbb{N}$,
- d.h., es gibt eine Nummerierung der Elemente von M so dass $M = \{m_0, m_1, \dots\}$.

Eine Menge ist überabzählbar wenn sie nicht abzählbar ist.

Lemma 1.7
Σ^* ist abzählbar (falls Σ endlich).

Erinnerung: Eine Menge M ist abzählbar falls sie gleich mächtig wie eine Teilmenge von $\mathbb{N}$ ist,

- d.h., es gibt eine Bijektion zw. M und einer Teilmenge von $\mathbb{N}$,
- d.h., es gibt eine Nummerierung der Elemente von M so dass $M = \{m_0, m_1, \dots\}$.

Eine Menge ist überabzählbar wenn sie nicht abzählbar ist.

Lemma 1.7
Σ^* ist abzählbar (falls Σ endlich).

Beweis:
Durch Beispiel:

$$\{0,1\}^* = \{\epsilon,\ 0,\ 1,\ 00,\ 01,\ 10,\ 11,\ 000, \dots\}$$

Erinnerung: Eine Menge M ist abzählbar falls sie gleich mächtig wie eine Teilmenge von $\mathbb{N}$ ist,

- d.h., es gibt eine Bijektion zw. M und einer Teilmenge von $\mathbb{N}$,
- d.h., es gibt eine Nummerierung der Elemente von M so dass $M = \{m_0, m_1, \dots\}$.

Eine Menge ist überabzählbar wenn sie nicht abzählbar ist.

Lemma 1.7
Σ^* ist abzählbar (falls Σ endlich).

Beweis:
Durch Beispiel:

$$\begin{aligned} \{0,1\}^* &= \{\epsilon,\ 0,\ 1,\ 00,\ 01,\ 10,\ 11,\ 000, \dots\} \\ \mathbb{N} &= \{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7, \dots\} \end{aligned}$$

□

Satz 1.8

Die Menge der Sprachen über einem nicht-leeren Alphabet ist überabzählbar.