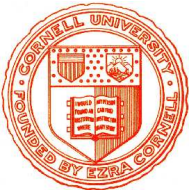


# Theoretische Informatik I



## Einheit 4

### Rückblick Theoretische Informatik I



1. Mathematische Methoden
2. Reguläre Sprachen
3. Kontextfreie Sprachen

- Mathematische **Methodik** in der Informatik

TI-1

- Mathematische **Methodik** in der Informatik TI-1
  - **Automatentheorie** und **Formale Sprachen** TI-1
    - Endliche Automaten und Reguläre Sprachen
      - Lexikalische Analyse
    - Kontextfreie Sprachen und Pushdown Automaten
      - Syntaxanalyse und Semantik
    - Die Chomsky Hierarchie
-

- Mathematische **Methodik** in der Informatik

TI-1

- **Automatentheorie** und **Formale Sprachen**

TI-1

- Endliche Automaten und Reguläre Sprachen
  - Lexikalische Analyse
- Kontextfreie Sprachen und Pushdown Automaten
  - Syntaxanalyse und Semantik
- Die Chomsky Hierarchie

---

- Theorie der **Berechenbarkeit**

TI-2

- Berechenbarkeitsmodelle
- Aufzählbarkeit und Entscheidbarkeit
- Unlösbare Probleme (Unentscheidbarkeit)

- Mathematische **Methodik** in der Informatik TI-1

- **Automatentheorie** und **Formale Sprachen** TI-1

- Endliche Automaten und Reguläre Sprachen
  - Lexikalische Analyse
- Kontextfreie Sprachen und Pushdown Automaten
  - Syntaxanalyse und Semantik
- Die Chomsky Hierarchie

---

- Theorie der **Berechenbarkeit** TI-2

- Berechenbarkeitsmodelle
- Aufzählbarkeit und Entscheidbarkeit
- Unlösbare Probleme (Unentscheidbarkeit)

- **Komplexitätstheorie** TI-2

- Komplexitätsmaße und -klassen für Algorithmen und Probleme
- Nicht handhabbare Probleme ( $\mathcal{NP}$ -Vollständigkeit)

## ● Methodik des Problemlösens

- Klärung der Voraussetzungen
- Lösungsweg konkretisieren: Einzelschritte und Begründungen
- Ergebnis kurz und prägnant zusammenfassen

## ● Methodik des Problemlösens

- Klärung der Voraussetzungen
- Lösungsweg konkretisieren: Einzelschritte und Begründungen
- Ergebnis kurz und prägnant zusammenfassen

## ● Viele Arten von Beweisen

- Deduktive Beweise: Logische Beweisschritte von Annahme zur Konklusion  
ggf. Definitionen auflösen, Mengenäquivalenz punktweise zeigen
- Widerlegungsbeweise: Widerspruch, Gegenbeispiel, Diagonalisierung  
auch genutzt für indirekte Beweisführung
- Induktionsbeweise: Zahlen-, vollständige, strukturelle, gegenseitige Induktion

## ● Methodik des Problemlösens

- Klärung der Voraussetzungen
- Lösungsweg konkretisieren: Einzelschritte und Begründungen
- Ergebnis kurz und prägnant zusammenfassen

## ● Viele Arten von Beweisen

- Deduktive Beweise: Logische Beweisschritte von Annahme zur Konklusion  
ggf. Definitionen auflösen, Mengenäquivalenz punktweise zeigen
- Widerlegungsbeweise: Widerspruch, Gegenbeispiel, Diagonalisierung  
auch genutzt für indirekte Beweisführung
- Induktionsbeweise: Zahlen-, vollständige, strukturelle, gegenseitige Induktion

**Präzise Argumente sind essentiell in der Informatik**



## ● Endliche Automaten

- Endliche Menge von Zuständen und Eingabesymbolen
- Verarbeitung von Eingabesymbolen ändert internen Zustand
- **Erkannte Sprache**: Abarbeitung endet in akzeptierendem Zustand
- Varianten: Deterministisch, nichtdeterministisch, mit  $\epsilon$ -Übergängen
- Umwandlung in deterministische Variante über Teilmengenkonstruktion

## ● Endliche Automaten

- Endliche Menge von Zuständen und Eingabesymbolen
- Verarbeitung von Eingabesymbolen ändert internen Zustand
- **Erkannte Sprache**: Abarbeitung endet in akzeptierendem Zustand
- Varianten: Deterministisch, nichtdeterministisch, mit  $\epsilon$ -Übergängen
- Umwandlung in deterministische Variante über Teilmengenkonstruktion

## ● Reguläre Ausdrücke

- **Algebraische Notation** für Sprachen:  $\epsilon$ ,  $\emptyset$ , Symbole von  $\Sigma$ ,  $+$ ,  $\circ$ ,  $*$
- Umwandelbar in  $\epsilon$ -NEAs (iterative Konstruktion)
- DEAs umwandelbar in reguläre Ausdrücke für Verarbeitungspfade oder durch Zustandselemination im RA Automaten

## ● Endliche Automaten

- Endliche Menge von Zuständen und Eingabesymbolen
- Verarbeitung von Eingabesymbolen ändert internen Zustand
- **Erkannte Sprache**: Abarbeitung endet in akzeptierendem Zustand
- Varianten: Deterministisch, nichtdeterministisch, mit  $\epsilon$ -Übergängen
- Umwandlung in deterministische Variante über Teilmengenkonstruktion

## ● Reguläre Ausdrücke

- **Algebraische Notation** für Sprachen:  $\epsilon$ ,  $\emptyset$ , Symbole von  $\Sigma$ ,  $+$ ,  $\circ$ ,  $*$
- Umwandelbar in  $\epsilon$ -NEAs (iterative Konstruktion)
- DEAs umwandelbar in reguläre Ausdrücke für Verarbeitungspfade oder durch Zustandselemination im RA Automaten

## ● Grammatiken

- Beschreibung des **Aufbaus von Sprachen** durch **Produktionsregeln**
- **Erzeugte Sprache**: schrittweise Ableitung endet in Terminalworten
- Typ-3 (rechtsslineare) Grammatiken sind äquivalent zu  $\epsilon$ -NEAs  
Direkte Umwandlung zwischen Produktionen und Überföhrungsfunktion

- **Abschlußeigenschaften**

- Operationen  $\cup$ ,  $\cap$ ,  $\overline{\phantom{x}}$ ,  $-$ ,  $^R$ ,  $\circ$ ,  $*$ ,  $h$ ,  $h^{-1}$  erhalten Regularität von Sprachen
- Verwendbar zum Nachweis von Regularität oder zur Widerlegung

## ● Abschlußeigenschaften

- Operationen  $\cup$ ,  $\cap$ ,  $\bar{\phantom{x}}$ ,  $-$ ,  $^R$ ,  $\circ$ ,  $*$ ,  $h$ ,  $h^{-1}$  erhalten Regularität von Sprachen
- Verwendbar zum Nachweis von Regularität oder zur Widerlegung

## ● Automatische Prüfungen

- Man kann testen ob eine reguläre Sprache leer ist
- Man kann testen ob ein Wort zu einer regulären Sprache gehört
- Man kann testen ob zwei reguläre Sprachen gleich sind

## ● Abschlußeigenschaften

- Operationen  $\cup$ ,  $\cap$ ,  $^c$ ,  $-$ ,  $^R$ ,  $\circ$ ,  $*$ ,  $h$ ,  $h^{-1}$  erhalten Regularität von Sprachen
- Verwendbar zum Nachweis von Regularität oder zur Widerlegung

## ● Automatische Prüfungen

- Man kann testen ob eine reguläre Sprache leer ist
- Man kann testen ob ein Wort zu einer regulären Sprache gehört
- Man kann testen ob zwei reguläre Sprachen gleich sind

## ● Minimierung von Automaten

- Ein Automat kann minimiert werden indem man äquivalente Zustände zusammenlegt und unerreichbare Zustände entfernt

## ● Abschlußeigenschaften

- Operationen  $\cup$ ,  $\cap$ ,  $\bar{\phantom{x}}$ ,  $-$ ,  $^R$ ,  $\circ$ ,  $*$ ,  $h$ ,  $h^{-1}$  erhalten Regularität von Sprachen
- Verwendbar zum Nachweis von Regularität oder zur Widerlegung

## ● Automatische Prüfungen

- Man kann testen ob eine reguläre Sprache leer ist
- Man kann testen ob ein Wort zu einer regulären Sprache gehört
- Man kann testen ob zwei reguläre Sprachen gleich sind

## ● Minimierung von Automaten

- Ein Automat kann minimiert werden indem man äquivalente Zustände zusammenlegt und unerreichbare Zustände entfernt

## ● Pumping Lemma

- Wiederholt man einen bestimmten Teil ausreichend großer Worte einer regulären Sprache beliebig oft, so erhält man immer ein Wort der Sprache
- Verwendbar zur Widerlegung von Regularität

## Kompliziertere Struktur als reguläre Sprachen

### ● Kontextfreie Grammatiken

- Produktionsregeln ersetzen **einzelne Variablen** durch beliebige Worte
- Ableitungsbäume beschreiben Struktur von Terminalworten (**Compiler!**)
- Ableitungsbäume entsprechen **Links- (oder Rechts-)ableitungen**
- Programmiersprachen brauchen **eindeutig** bestimmbare Ableitungsbäume



## Kompliziertere Struktur als reguläre Sprachen

### ● Kontextfreie Grammatiken

- Produktionsregeln ersetzen **einzelne Variablen** durch beliebige Worte
- Ableitungsbäume beschreiben Struktur von Terminalworten (**Compiler!**)
- Ableitungsbäume entsprechen **Links- (oder Rechts-)ableitungen**
- Programmiersprachen brauchen **eindeutig** bestimmbare Ableitungsbäume

### ● Pushdown-Automaten

- Nichtdeterministischer **endlicher Automat** mit **Stack** und  $\epsilon$ -Übergängen
- Erkennung von Worten durch **Endzustand** oder **leeren Stack**
- Analyse durch Betrachtung von **Konfigurationsübergängen**
- Nichtdeterministische PDAs äquivalent zu kontextfreien Grammatiken
  - Umwandlung von **Konfigurationsübergängen in Regeln** und umgekehrt
- Deterministische PDAs weniger mächtig (nur eindeutige Typ-2 Sprachen)

## ● Abschlußeigenschaften

- Operationen  $\cup$ ,  $^R$ ,  $\circ$ ,  $*$ ,  $\sigma$ ,  $h^{-1}$  erhalten Kontextfreiheit von Sprachen
- Keine Abgeschlossenheit unter  $\cap$ ,  $\overline{\phantom{x}}$ ,  $-$

## ● Abschlußeigenschaften

- Operationen  $\cup$ ,  $^R$ ,  $\circ$ ,  $*$ ,  $\sigma$ ,  $h^{-1}$  erhalten Kontextfreiheit von Sprachen
- Keine Abgeschlossenheit unter  $\cap$ ,  $\overline{\phantom{x}}$ ,  $-$

## ● Automatische Prüfungen

- Man kann testen ob eine kontextfreie Sprache leer ist
- Man kann testen ob ein Wort zu einer kontextfreien Sprache gehört
- Man kann nicht testen ob zwei kontextfreie Sprachen gleich sind

Viele wichtige Fragen sind nicht automatisch prüfbar

## ● Abschlußeigenschaften

- Operationen  $\cup$ ,  $^R$ ,  $\circ$ ,  $*$ ,  $\sigma$ ,  $h^{-1}$  erhalten Kontextfreiheit von Sprachen
- Keine Abgeschlossenheit unter  $\cap$ ,  $\overline{\phantom{x}}$ ,  $-$

## ● Automatische Prüfungen

- Man kann testen ob eine kontextfreie Sprache leer ist
- Man kann testen ob ein Wort zu einer kontextfreien Sprache gehört
- Man kann nicht testen ob zwei kontextfreie Sprachen gleich sind

Viele wichtige Fragen sind nicht automatisch prüfbar

## ● Pumping Lemma

- Wiederholt man bestimmte Teile ausreichend großer Worte einer kontextfreien Sprache beliebig oft, so erhält man immer ein Wort der Sprache
- Viele einfache Sprachen sind nicht kontextfrei

# FRAGEN ?