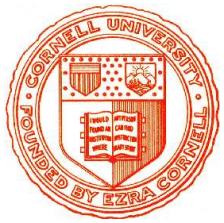
Theoretische Informatik I

Einheit 5



Rückblick

Theoretische Informatik I

Themen der Theoretischen Informatik I & II

- **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
  - Allgemeine und kontextsensitive Sprachen

- **Theorie der Berechenbarkeit** TI-2
  - Berechenbarkeitsmodelle
  - Aufzählbarkeit, Entscheidbarkeit, Unlösbare Probleme
- **Komplexitätstheorie** TI-2
  - Komplexitätsmaße und -klassen für Algorithmen und Probleme
  - Nicht handhabbare Probleme ($\mathcal{NP}$-Vollständigkeit)
  - Effiziente Alternativen zu konventionellen Verfahren

Reguläre Sprachen

- **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/ohne ϵ-Übergänge
  - Umwandlung in deterministische Variante über Teilmengenkonstruktion
- **Reguläre Ausdrücke**
  - Algebraische Notation für Sprachen: ϵ, $\emptyset$, Symbole von Σ, $+$, $\circ$, *
  - Umwandelbar in ϵ-NEAs (iterative Konstruktion)
  - DEAs umwandelbar durch Zustandselemination im VNEAs
- **Grammatiken**
  - Beschreibung des Aufbaus von Sprachen durch Produktionsregeln
  - Erzeugte Sprache: schrittweise Ableitung endet in Terminalworten
  - Typ-3 (rechtsslineare) Grammatiken sind äquivalent zu ϵ-NEAs
    Direkte Umwandlung zwischen Produktionen und Überführungsfunktion

Eigenschaften regulärer Sprachen

- **Abschlußeigenschaften**
  - Operationen $\cup$, $\cap$, $\overline{\ }$, -, ^{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

Kontextfreie Sprachen

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 ϵ-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)

Eigenschaften kontextfreier Sprachen

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

Jenseits von kontextfreien Sprachen

- **Turingmaschinen als Maschinenmodell**
  - Deterministischer endlicher Automat mit unendlichem Speicherband
  - Äquivalent zu realen Computern
  - Viele gleichmächtige Varianten
  - Simulation nichtdeterministischer Maschine ist exponentiell
- **Typ-0 Sprachen**
  - Keine Einschränkung an Produktionsregeln
  - Auch Terminalsymbole und ganze Wörter dürfen ersetzt werden
  - Typ-0 Grammatiken sind äquivalent zu Turingmaschinen
    - Umwandlung von Konfigurationsübergängen in Regeln und umgekehrt
- **Entscheidbare Sprachen**
  - Sprachen von Turingmaschinen, die immer terminieren
- **Typ-1 Sprachen**
  - Produktionsregeln der Grammatiken dürfen Wörter nicht verkleinern
  - Sprachen von Turingmaschinen mit linear beschränktem Band

FRAGEN ?