Informatik III

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Arne Vater

Wintersemester 2006/07

28. Vorlesung

09.02.2007

Die Chomsky-Klassifizierung

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Chomsky-Hierachien**
  - 3: Reguläre Grammatiken
  - 2: Kontextfreie Grammatiken
  - 1: Wachsende kontextsensitive Grammatiken
  - 0: Allgemeine Grammatiken
- **Alternative Beschreibung**
  - Reguläre Sprachen und endliche Automaten
  - Kontextfreie Sprachen und Kellerautomaten
  - Wachsende kontextsensitive Sprachen und linearer Platz
  - Chomsky-0-Sprachen und Rekursiv aufzählbare Sprachen

Übersicht Chomsky-Charakterisierung

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Stufe	Sprache	Regeln	Maschinenmodell
Typ-0	Rekursiv Aufzählbar	keine Einschränkung	Turing-Maschine
Typ-1	(Wachsend) Kontextsensitive Sprachen	$\alpha A\beta \rightarrow \alpha\gamma\beta$ $A \in V$, $\lvert\gamma\rvert \geq 1$ $\alpha,\beta,\gamma \in (\Sigma \cup V)^*$	Linear-Platz-NTM
Typ-2	Kontextfreie Sprachen	$A \rightarrow \gamma$ $A \in V$ $\gamma \in (\Sigma \cup V)^*$	Nichtdet. Kellerautomat (PDA)
Typ-3	Reguläre Sprachen	$A \rightarrow aB$ $A \rightarrow a$ $A,B \in V$ $a \in \Sigma$	Endlicher Automat (NFA/DFA)

Chomsky-3: Reguläre Sprachen und endliche Automaten

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Definition**
  - Eine (rechts)-reguläre Grammatik ist ein Vierer-Tupel $G = (V,\Sigma,R,S)$, wobei
    - V ist die endliche Menge der Variablen (Nichtterminale)
    - Σ ist die endliche Menge der Terminale (Terminalsymbole)
    - R ist eine endliche Menge an Ersetzungsregeln (Regeln/Produktionen)
      - Jede Regel bildet eine Variable auf ein Terminal
        - $A \to a$, mit $A \in V$ und $a \in \Sigma$
      - oder auf ein Wort aus einem Terminal und einer Variable ab
        - $A \to aB$ mit $A,B \in V$ und $a \in \Sigma$
    - $S \in V$ ist die Startvariable
      - oder die Startvariable wird auf das leere Wort abgebildet
        - $S \to \varepsilon$
- **Beispiel**
  - $(\{S,T\},\{0,1\}, \{S \to 0S, S \to 0, S \to 1T, T \to 1S, , T \to 0T\}, S)$
- **Theorem**
  - Die Menge der regulären Sprachen werden durch die rechts-regulären Grammatiken beschrieben

Chomsky-3: Reguläre Sprachen und endliche Automaten

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem**
  - Die Menge der regulären Sprachen werden durch die rechts-regulären Grammatiken beschrieben.
- **Beweis: Rückrichtung**
  - Rechts-reguläre Grammatiken können durch nicht-deterministische endliche Automaten beschrieben werden
    - Zustandsmenge: $Q = V \cup \{Q_{akz}\}$
    - Startzustand: S
    - Akzeptierende Zustandsmenge: $\{Q_{akz}\}$
    - Bei Regel $A \to aB$ wird B zur Menge von $\delta(A,a)$ hinzugefügt, so dass
      - $B \in \delta(A,a) \Leftrightarrow (A \to aB) \in R$
    - Bei Regel $A \to a$ wird Q_{akz} zur Menge von $\delta(A,a)$ hinzugefügt, so dass
      - $Q_{akz} \in \delta(A,a) \Leftrightarrow (A \to a) \in R$
    - Bei Regel $S \to \varepsilon$ wird Q_{akz} zur Menge von $\delta(S, \varepsilon)$ hinzugefügt, so dass
      - $Q_{akz} \in \delta(S, \varepsilon) \Leftrightarrow (S \to \varepsilon) \in R$

- **Theorem**
  - Die Menge der regulären Sprachen werden durch die rechts-regulären Grammatiken beschrieben.
- **Beweis: Hinrichtung**
  - Deterministische endliche Automaten können durch rechts-reguläre Grammatiken beschrieben werden
    - Variablen: V = Q
    - Startsymbol ist der Startzustand
    - Terminale sind das Alphabet der Sprache
    - Falls $\delta(A,a) = B$ füge Regel $A \to aB$ hinzu
      - $\delta(A,a) = B \Leftrightarrow (A \to aB) \in R$
    - Falls $\delta(A,a) = B$ und B ist akzeptierender Zustand füge Regel $A \to a$ hinzu
      - $\delta(A,a) = B$ und B ist akzeptierend $\Leftrightarrow (A \to a) \in R$

Beispiel

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Bei Regel A → aB wird B zur Menge von δ(A,a) hinzugefügt, so dass**
  - $B \in \delta(A,a) \Leftrightarrow (A \rightarrow aB) \in R$
- **Bei Regel A → a wird Q_{akz} zur Menge von δ(A,a) hinzugefügt, so dass**
  - $Q_{akz} \in \delta(A,a) \Leftrightarrow (A \rightarrow a) \in R$
- **Bei Regel S → ε wird Q_{akz} zur Menge von δ(S, ε) hinzugefügt, so dass**
  - $Q_{akz} \in \delta(S, \varepsilon) \Leftrightarrow (S \rightarrow \varepsilon) \in R$

- **({S,T},{0,1}, {S → 0S, S → 0, S → 1T, T → 1S, , T → 0T}, S}**

- **Falls δ(A,a) = B füge Regel A → aB hinzu**
  - $\delta(A,a) = B \Leftrightarrow (A \rightarrow aB) \in R$
- **Falls δ(A,a) = B und B ist akzeptierender Zustand füge Regel A → a hinzu**
  - $\delta(A,a) = B$ und B ist akzeptierend $\Leftrightarrow (A \rightarrow a) \in R$

Chomsky-2: Kontextfreie Sprachen und PDA

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Definition**
  - Eine kontextfreie Grammatik ist ein Vierer-Tupel $G = (V,\Sigma,R,S)$, wobei
    - V ist die endliche Menge der Variablen (Nichtterminale)
    - Σ ist die endliche Menge der Terminale (Terminalsymbole)
      - V und Σ sind disjunkte Mengen
    - R ist eine endliche Menge an Ersetzungsregeln (Regeln/Produktionen)
      - Jede Regel besteht aus einer Variable und einer Zeichenkette aus Variablen und Terminalen,
        - $A \rightarrow w$, mit $A \in V$, $w \in (V \cup \Sigma)^*$
    - $S \in V$ ist die Startvariable
- **Ableitung**
  - Falls die Regel $A \rightarrow w$ in R ist, dann ist $uAv \Rightarrow uwv$, d.h.
    - uAv kann zu uwv in einem Schritt abgeleitet werden
  - Wir sagen das u zu v abgeleitet werden kann oder $u \Rightarrow^* v$, wenn es ein $k \geq 0$ gibt mit
    - $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$
- **Sprache der Grammatik G**
  - $L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* v\}$
- **Die kontextfreien Sprachen werden durch kontextfreie Grammatiken erzeugt.**

Wiederholung: Keller-Automat

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Definition**
  - Ein Kellerautomat (pushdown automaton - PDA) wird durch $(Q, \Sigma, \Gamma, \delta, q_0, F)$ beschrieben, wobei
  1. Q ist die Menge der Zustände
  2. Σ ist das Eingabealphabet
  3. Γ ist das Kelleralphabet
  4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathbf{P}(Q \times \Gamma_\varepsilon)$ ist die Übergangsfunktion
  5. q_0 ist der Startzustand
  6. $F \subseteq Q$ ist die Menge der akzeptierenden Zustände

- **Ein PDA akzeptiert w,**
  - wenn es $w = w_1 w_2 ... w_m$ mit $w_i \in \Sigma_\varepsilon$ gibt
  - und $q = r_0 r_1 ... r_m$ mit $s_i \in Q$ gibt
  - und $s_0, s_1, ..., s_m \in \Gamma_\varepsilon^*$ gibt, so dass

1. $r_0 = q_0$ und $s_0 = \varepsilon$
  - Startzustand mit leeren Keller

2. Für i = 0, ..., m-1 gilt:
  - $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, wobei
  - $s_i = at$ und $s_{i+1} = bt$ für passende $a,b \in \Gamma_\varepsilon$, $t \in \Gamma_\varepsilon^*$

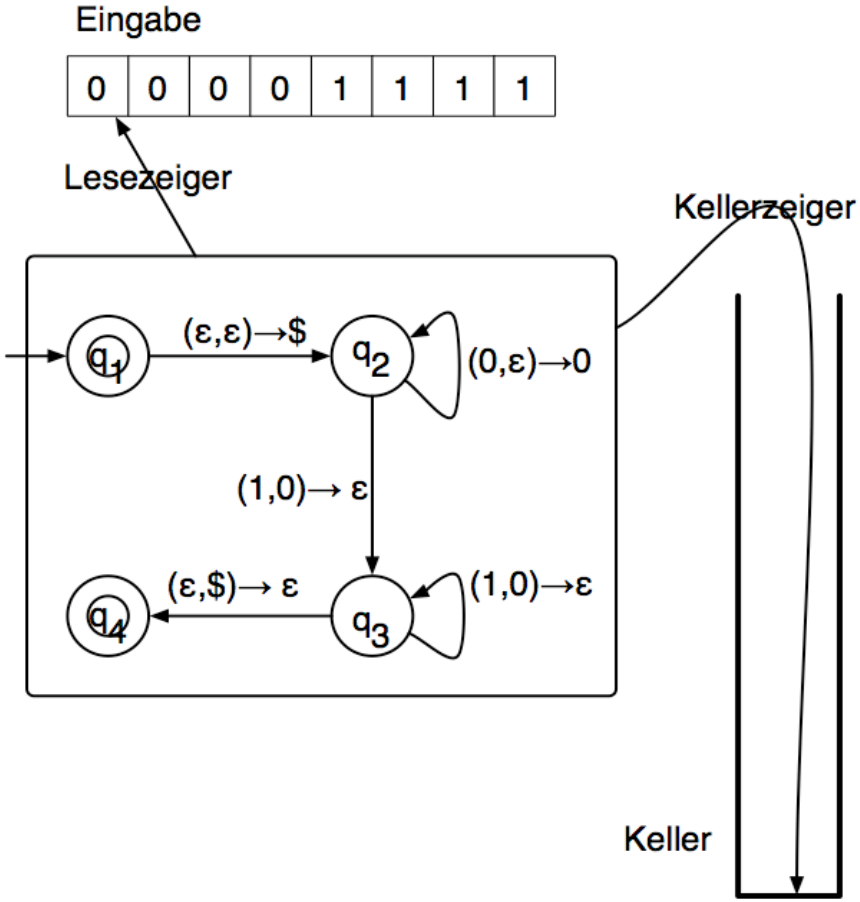
3. $r_m \in F$

Wiederholung: Keller-Automat

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Definition**
  - Ein Kellerautomat (pushdown automaton - PDA) wird durch $(Q, \Sigma, \Gamma, \delta, q_0, F)$, wobei

1. Q ist die Menge der Zustände
2. Σ ist das Eingabealphabet
3. Γ ist das Kelleralphabet
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow \mathbf{P}(Q \times \Gamma_\varepsilon)$ ist die Übergangsfunktion
5. q_0 ist der Startzustand
6. $F \subseteq Q$ ist die Menge der akzeptierenden Zustände



Keller-Automaten beschreiben genau die kontextfreien Sprachen

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem**
  - Eine Sprache ist genau dann kontextfrei, wenn ein Kellerautomat sie erkennt
- **Beweisideen**
  - Hinrichtung
    - durch Bau eines Kellerautomaten der die Regeln im Keller rät
    - und dann beim Einlesen der Eingabe verifiziert
  - Rückrichtung
    - durch spezielle Variablen A[pq] die Zustände bei gleicher Höhe des Kellers beschreiben
    - Dadurch ergeben sich die Ableitungen
      - relativ aufwändige Konstruktion

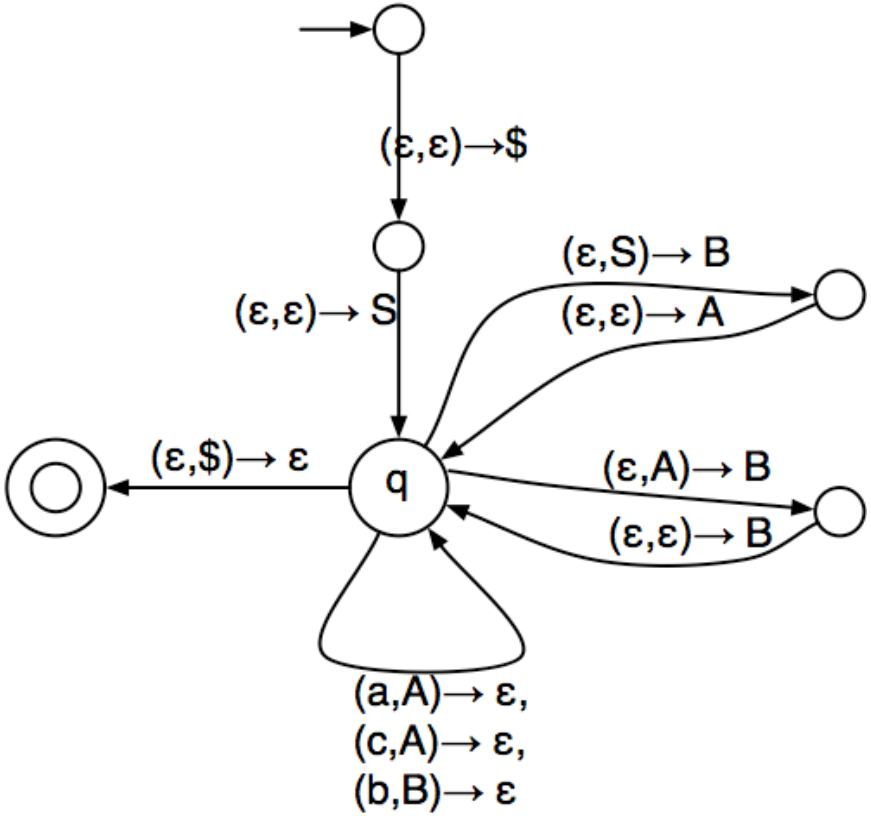
Beispielkonstruktion für Hinrichtung

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Grammatik: G = (V,Σ,R,S) mit**
  - V = {S,A,B}
  - Σ = {a,b,c}

- **R = { S → AB,**
  A → BB,
  A → a,
  A → c,
  B → b }

- **Beispiel:**
  - S Keller: $S Eingabe: cb
    ⇒ AB Keller: $BA Eingabe: cb
    ⇒ cB Keller: $B Eingabe: b
    ⇒ cb Keller: $ Eingabe: -



Chomsky-1: (Wachsend) Kontext-sensitive Sprachen und NSPACE(n)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Definition (Wachsend) Kontextsensitive Grammatik**
  - Eine wachsend kontext-sensitive Grammatik ist ein Vierer-Tupel $G = (V,\Sigma,R,S)$, mit
    - Variablen V (Nichtterminale)
    - Terminalen Σ und
    - einer Regelmenge R
      - Jede Regel besteht aus einer Variable und einem Kontext α, β, wobei die Variable auf ein Wort mindestens der Länge 1 abgebildet wird:
      - $\alpha A \beta \rightarrow \alpha\gamma\beta$ mit $A \in V$, $|\gamma| \geq 1$, $\alpha,\beta,\gamma \in (\Sigma \cup V)^*$
      - oder $S \rightarrow \varepsilon$
    - $S \in V$ ist die Startvariable
  - Die kontextsensitive Grammatiken produzieren die kontextsensitiven Sprachen CSL
- **Theorem**
  - CSL = NSPACE(n)

Beispiel einer kontext-sensitiven Grammatik

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **G=(V,Σ,R,S) mit**
  - Variablen V = {S,B,C,D}
  - Terminalen Σ = {a,b,c}
  - Regeln R:
    - S → aSBC
    - S → aBC
    - CB → DB
    - DB → DC
    - DC →BC
    - aB → ab
    - bB → bb
    - bC → bc
    - cC → cc
  - Startsymbol: S

- **Beispiel:**

S ⇒ **aSBC**
⇒ a**aSBC**BC
⇒ aa**aBC**BCBC
⇒ aaaB**D**BCBC
⇒ aaaBD**C**CBC
⇒ aaaB**B**CCBC
…
⇒ aaaBBC**BC**C
…
⇒ aaaBB**BC**CC
⇒ aaa**b**BBCCC
⇒ aaab**b**BCCC
⇒ aaabb**b**CCC
⇒ aaabbb**c**CC
⇒ aaabbbc**c**C
⇒ aaabbbcc**c**

- **$L(G) = \{ a^n b^n c^n \mid n \geq 1 \}$**

Chomsky-1: (Wachsend) Kontext-sensitive Sprachen und NSPACE(n)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem**
  - CSL = NSPACE(n)
- **Beweisidee**
  - CSL ⊆ NSPACE(n)
    - NTM M rät die Ableitungen bis die Eingabewortläge erreicht wird
    - M akzeptiert, falls die Eingabe abgeleitet worden ist
    - Problem: nichthaltende wiederholende Berechnungspfade
    - Lösung:
      - M führt einen Zähler, der mit jeder geratenen Produktion um eins erhöht wird.
      - Überschreitet der Zähler $2^{c\,n}$, für geeignete Konstante c, dann verwirft M
    - Diese Lösung funktioniert, da höchstens 2^{cn} mögliche Zwischenschritte in der minimalen Ableitung einer kontextsensitiven Sprache vorkommen

Chomsky-1: (Wachsend) Kontext-sensitive Sprachen und NSPACE(n)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem**
  - CSL = NSPACE(n)
- **Beweisidee**
  - CSL ⊇ NSPACE(n)
    - Betrachte 1-Band-NTM mit linearem Platz, die genau auf der Eingabe rechnet und keine weiteren Symbole des Bandes bearbeitet
    - Die Berechnung der 1-Band-NTM wird rückwärts dargestellt
    - Für jedes Eingabesymbol a wird durch eine Variable A und ein Terminal a dargestellt
    - Am Anfang wird eine akzeptierende Konfiguration erzeugt durch entsprechende Regeln: Von Startsymbol S → "Q_{akz} _ _ _ _ _ .... _ "
    - Für jeden Übergang (q,a) → (q',r,b) der NTM führe zwei Variablen
      - $X_{[(q,a) \to (q',r,b)]}$ und $Y_{[(q,a) \to (q',r,b)]}$ ein
    - Nun werden für (q,a) → (q',r,b) die folgenden Produktionen eingfügt
      - B Q' → $X_{[(q,a) \to (q',r,b)]}$Q'
      - $X_{[(q,a) \to (q',r,b)]}$Q' → $X_{[(q,a) \to (q',r,b)]}$ $Y_{[(q,a) \to (q',r,b)]}$
      - $X_{[(q,a) \to (q',r,b)]}$ $Y_{[(q,a) \to (q',r,b)]}$ → Q $Y_{[(q,a) \to (q',r,b)]}$
      - Q $Y_{[(q,a) \to (q',r,b)]}$ → Q A
    - Übergänge (q,a) → (q',l,b) werden analog bearbeitet
    - Damit wird die Berechnung rückwärts dargestellt
    - Am Ende werden die Terminale A → a eingesetzt: Die "Rückwärtsrechnung hält"

Chomsky-0: Allgemeine Grammatiken und Rekursiv Aufzählbar

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Definition Allgemeine (Chomsky-0) Grammatik**
  - Eine allgemeine (Chomsky-0) Grammatik ist ein Vierer-Tupel $G = (V,\Sigma,R,S)$, mit
    - Variablen V (Nichtterminale)
    - Terminalen Σ und
    - einer Regelmenge R
      - Jede Regel besteht aus einem Wort $\alpha \in (\Sigma \cup V)^*$ auf der linken Seite und einem Wort $\beta \in (\Sigma \cup V)^*$ auf der rechten
      - Jede Regel $\alpha \rightarrow \beta$ mit $\alpha,\beta \in (\Sigma \cup V)^*$ ist erlaubt
    - $S \in V$ ist die Startvariable
- **Theorem**
  - Die allgemeinen Grammatiken beschreiben genau die Menge der rekursiv aufzählbaren Sprachen

Chomsky-0: Allgemeine Grammatiken und Rekursiv Aufzählbar

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem**
  - Die allgemeinen Grammatiken beschreiben genau die Menge der rekursiv aufzählbaren Sprachen
- **Beweisidee**
  - Hinrichtung:
    - NTM rät eine Ableitung und versucht dadurch aus dem Startsymbol das Eingabe-Wort abzuleiten
    - Falls keine Ableitung möglich, hält die NTM nie
  - Rückrichtung
    - Berechnung einer NTM wird umgekehrt
    - Ausgehend von der (eindeutigen) Endkonfiguration mit leerem Band (als Startsymbol) wird jeder Konfigurationsübergang durch lokale kontextsensitive Ableitungsregeln beschrieben
      - wie bei der Konstruktion für kontextsensitive Sprachen
    - Blank-Symbole am rechten Speicherrand können durch verkürzende kontextsensitive Regeln erzeugt und wieder freigegeben werden

Zusammenfassung der Vorlesung

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Formale Sprachen**
  - Chomsky-0 = Rekursiv Aufzählbar
  - Chomsky-1 = Kontextsensitiv = NSpace(n)
  - Chomsky-2 = Kontextfrei = PDA
    - Pumping-Lemma
  - Chomsky-3 = Regulär = NFA = DFA = Space(0)
    - Pumping-Lemma, Myhill-Nerode
- **Berechenbarkeitstheorie**
  - Abzählbar
    - Diagonalisierung
  - Rekursiv Aufzählbar = Akzeptor
    - Rekursiv Ko-Aufzählbar
    - Diagonalisierung, Abbildungsreduktion
  - Rekursiv = Entscheidbar
    - Diagonalisierung, Abbildungsreduktion, Turing-Reduktion
  - Beschreibungskomplexität
    - Unkomprimierbarkeit von Worten
- **Komplexitätstheorie**
  - DTM, NTM
    - Bänder, Konfigurationen
  - TIME(T(n)), NTIME(T(n))
  - P,NP
    - Abbildungsreduktion
    - NP-schwierig
    - NP-vollständig
    - SAT, 3-SAT, Clique, Subset-Sum, Hamiltonian
    - Approximationsalgorithmen
  - NSPACE(S(n))
    - $\subseteq$ SPACE($S(n)^2$) Satz von Savitch
  - PSPACE
    - PSPACE-schwierig
    - PSPACE-vollständig
    - QBF

Ende der 28. Vorlesung

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Arne Vater

Wintersemester 2006/07

28. Vorlesung

09.02.2007