Grundlagen Theoretischer Informatik 2

WiSe 2009/10 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Grundlagen Theoretischer Informatik 2 Gesamtübersicht

- Organisatorisches; Einführung
- Ersetzungsverfahren: Grammatiken und Automaten
- Rekursionstheorie-Einführung
- Komplexitätstheorie-Einführung

Organisatorisches

Vorlesungen DI 8.15-9.45 im H 11

Übungsbetrieb in Form von einer Übungsgruppe
BEGINN: vorlesungsartig in der ersten Semesterwoche
FR 10-12, HZ 204

Dozentensprechstunde DO 13-14 in meinem Büro H 410 (4. Stock)

Mitarbeitersprechstunde (Stefan Gulan) FR 12-13 H 413

Tutorensprechstunde MO 13.30-14.30 & MI 14-15 H 407/H412

Benotung

Die Note ergibt sich ausschließlich aus der gezeigten Leistung bei der Abschlussprüfung.

Die Abschlussprüfung erfolgt entweder schriftlich am 23.2.2010 oder mündlich (n.V.).
Wir werden die schriftliche Form bevorzugen.
Rückmeldungen Ihrerseits sind erwünscht.

Zulassungskriterien

Um an der Abschlussprüfung teilnehmen zu dürfen, wird vorausgesetzt:

(1) Mindestens zweimaliges erfolgreiches Vorrechnen von Hausaufgaben sowie
(2) 40% der Hausaufgabenpunkte (Zweier-/ Dreiergruppenabgabe möglich) sowie
(3) Bestehen der Zwischenklausur.

Abgabe von Hausaufgaben

Diese sollte in Gruppen zu 2-3 Personen erfolgen.

Abgabeschluss ist MI 14 Uhr, im mit GTI 2 beschrifteten Kasten im 4. Stock vor dem gemeinsamen Sekretariat von Prof. Näher.
Verspätete Abgaben gelten als nicht abgegeben und werden dementsprechend mit 0 Punkten bewertet

In der darauffolgenden Übung (am Freitag) werden die korrigierten Lösungen zurückgegeben. Wir nutzen die Zeit (i.d.R. zwei Tage) auch dazu, uns die Namen derjenigen herauszusuchen, die gewisse Aufgaben dann vorzurechnen haben.
Wer die Aufgaben auf Aufforderung **nicht erläuternd vorrechnen** kann, bekommt die durch Hausaufgabenabgabe erzielten Punkte für das gesamte Aufgabenblatt abgezogen.
Selbstverständlich gilt das Vorrechnen dann nicht als “erfolgreich” im Sinne der vorigen Folie.
Sollte ein Vorrechnen nicht erfolgreich sein, wird i.d.R. ein weiterer Studierender Gelegenheit zum Vorrechnen bekommen.

Die Lösungen sind handschriftlich anzufertigen; weder Schreibmaschinen- noch Computerausdrucke werden akzeptiert, erst recht keine Kopien.

Probleme ? Fragen ?

Klären Sie bitte Schwierigkeiten mit Vorlesungen oder Übungen möglichst umgehend in den zur Verfügung gestellten Sprechzeiten.

In der Tutorensprechstunde stehen Ihnen (alternierend) Studenten höherer Semester zu Rückfragen bereit.

Wir helfen Ihnen gerne!
. . . wir sind aber keine Hellseher, die Ihnen Ihre Schwierigkeiten an der Nasenspitze ansehen. . .

Aufgabe Gegeben sei $G = (\{S, B, C\}, \{a, b, c\}, P, S)$ mit

$$\begin{aligned} P \quad = \{ \quad & S \to aSBC, S \to aBC, CB \to BC, \\ & aB \to ab, bB \to bb, bC \to bc, cC \to cc\} \end{aligned}$$

Wir wollen zeigen, dass gilt:

$$L(G) = \{a^n b^n c^n | n > 0\}$$

(Achtung: Hier gibt es keine Syntaxbäume, da G nicht kontextfrei ist!)

Wie beweist man so eine Behauptung $L(G) = L$?

In der Regel ist es sinnvoll, die Behauptung in zwei Schritten sich zu überlegen:
(1) $L \subseteq L(G)$ und
(2) $L(G) \subseteq L$.

Der erste Schritt ist oft der einfachere.
Beim zweiten Schritt hat man mit möglichen bösen Ableitungen zu kämpfen.

Der Nachweis jedes der Schritte erfordert in der Regel mehrere (kleinere) Induktionsbeweise.
Manchmal (siehe Beispiel aus der vorigen Vorlesung) lassen sich die beiden Schritte "zusammenlegen."
Dann überlegt man sich zumeist, wie die Menge der in n Schritten ableitbaren Wörter (über dem Gesamtalphabet) aussieht.

$L(G) = \{a^n b^n c^n | n > 0\} =: L$! 1. Schritt: $L \subseteq L(G)$.

- $(n-1)$-faches Anwenden der Regel $S \to aSBC$:

$$S \Rightarrow^* a^{n-1} S (BC)^{n-1}$$

- einmaliges Anwenden von $S \to aBC$:

$$a^{n-1} S (BC)^{n-1} \Rightarrow a^n (BC)^n$$

- mehrfaches Vertauschen $CB \to BC$:

$$a^n (BC)^n \Rightarrow^* a^n B^n C^n$$

- Zusammengesetzt $S \Rightarrow^* a^n B^n C^n$
- einmal $aB \to ab$, $(n-1)$-mal $bB \to bb$:$S \Rightarrow^* a^n b^n C^n$
- einmal $bC \to bc$, $(n-1)$-mal $cC \to cc$:$S \Rightarrow^* a^n b^n c^n$

Ein Induktionsbeweis en detail

Behauptung: Für alle $n \geq 1$ gilt: $S \Rightarrow^* a^{n-1}S(BC)^{n-1}$.
Präzisierung: Für alle $n \geq 1$ gilt: $S \Rightarrow^{n-1} a^{n-1}S(BC)^{n-1}$.
Induktionsanfang (IA) für $n = 1$: $S \Rightarrow^0$. OK!
(Unnötiger Test für $n = 2$: $S \Rightarrow aSBC$. Stimmt auch!)
Induktionshypothese (Induktionsvoraussetzung IV): Angenommen, die Aussage(form) gelte für $n = m$.
Formaler IV: $S \Rightarrow^{m-1} a^{m-1}S(BC)^{m-1}$
Wir müssen daraus die Gültigkeit der Behauptung für $n = m + 1$ ableiten.
Formale IB: $S \Rightarrow^m a^m S(BC)^m$.
Eine $(m+1)$-Schritt-Ableitung lässt sich schreiben als eine m-Schritt-Ableitung, gefolgt von einem weiteren Ableitungsschritt. Nach IV und nach einer (weiteren) Anwendung der Regel $S \to aSBC$ gilt: $S \Rightarrow^{m-1} a^{m-1}S(BC)^{m-1} \Rightarrow a^m S(BC)^m$.
Also gilt die Induktionsbehauptung.
Nach dem Prinzip der vollständigen Induktion folgt die “Präzisierung” und damit die ursprüngliche Behauptung.

Umkehrung: 2. Schritt: $L(G) \subseteq L$.

- Balancebedingung per Induktion:
  $S \Rightarrow^* w$ impliziert $\#_a w = \#_b w + \#_B w = \#_c w + \#_C w$
- Zudem per Induktion: $S \rightarrow^* w$ impliziert

  $$w = a^i S v \text{ oder } w = a^i b^j c^k v$$

  für geeignete $i, j, k \in \mathbb{N}$, $i > 0$, und $v \in \{B, C\}^*$.
- Zusammengesetzt: $S \Rightarrow^* w$ und $w \in T^+$ impliziert: $w = a^i b^j c^k v$ mit $i = j = k > 0$ und $v = \lambda$.

Chomsky-Hierarchie Eine Chomsky-Grammatik $G = (N, T, P, S)$ heißt

- *Typ-3-Grammatik* oder *rechtslinear*,
  wenn alle Produktionen von einer der Formen $A \to w$ oder $A \to wB$ sind ($w \in T^*, A, B \in N$).
- *Typ-2-Grammatik* oder *kontextfrei*,
  wenn alle Produktionen von der Form $A \to \gamma$ sind (mit $\gamma \in (N \cup T)^*, A \in N$).
- *Typ-1-Grammatik* oder *kontextsensitiv*,
  wenn alle Produktionen von der Form $\alpha A \beta \to \alpha \gamma \beta$ sind (mit $A \in N$, $\alpha, \beta, \gamma \in (N \cup T)^*, \gamma \neq \lambda$).
  $S \to \lambda$ nur dann gelten, falls S auf keiner rechten Seite einer Regel auftritt.
- *Typ-0-Grammatik* oder *allgemeine Chomsky-Grammatik*,
  wenn die Form der Produktionen nicht weiter eingeschränkt ist.

Chomsky-Hierarchie

Eine Sprache heißt *Typ-i-Sprache* ($i = 0, 1, 2, 3$), wenn es eine Typ-i-Grammatik gibt, die diese Sprache erzeugt.

Die Menge aller Typ-i-Sprachen bezeichnen wir mit $\mathcal{L}_i$.

- Typ-3-Grammatik $\Longrightarrow$ Typ-2-Grammatik
- Typ-3-Sprachen $\subseteq$ Typ-2-Sprachen
- Wegen der Bedingung $\gamma \neq \lambda$ offen:
  Wie verhalten sich Typ-2- zu Typ-1-Sprachen?

Warum gilt $\mathcal{L}_2 \subseteq \mathcal{L}_1$? Elimination von λ-Regeln!

Satz: Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik. Es gibt eine kontextfreie Grammatik $G' = (N, T, P', S)$ mit $L(G') = L(G)\backslash\{\lambda\}$, die keine Produktion der Form $A \to \lambda$ mit $A \in N$ enthält.

Anmerkung: G' ist damit nicht nur kontextfrei, sondern sogar vom Typ-1 !

Beweis I: Bestimme $N_1 \subseteq N$ mit $A \Rightarrow_G^* \lambda \Leftrightarrow A \in N_1$.
Rekursive Konstruktion von N_1:

1. $N_1^{(0)} := \{A \in N \mid A \to \lambda \in P\}$.
2. A wird in $N_1^{(i)}$ aufgenommen, wenn $A \in N_1^{(i-1)}$ oder es eine Regel $A \to A_1 \dots A_k$ gibt mit $\{A_1, \dots, A_k\} \subseteq N_1^{(i-1)}$

Wiederhole Teil 2) der Konstruktion, bis $N_1^{(i-1)} = N_1^{(i)}$.
Da $N_1^0 \subseteq N_1^1 \subseteq \cdots \rightsquigarrow$ Konstruktion ist nach endlicher Zeit abgeschlossen!
Dann ist $N_1 = N_1^{(i)}$!

Beweis II: Erweitere Regelmenge P iterativ:

1. $P^{(0)} := P$

2. $P^{(i)} := P^{(i-1)} \cup \{A \to xy \mid A \to xBy \in P^{(i-1)}, B \in N_1\}$.

Konstruktion liefert nach endlicher Zeit $P^{(i)} = P^{(i-1)} =: \overline{P}$

Beweis III: Entferne alle Regeln der Form $A \to \lambda$ mit $A \in N$ aus $\overline{P}$,
P' sei dabei entstehende Menge von Regeln.

Mit vollständiger Induktion kann man zeigen: $L(G') = L(G) \setminus \{\lambda\}$.

Folgerung: Zu jedem kontextfreien G gibt es kontextfreies G'' so dass:

- $L(G) = L(G'')$
- G'' ist nicht nur kontextfrei, sondern sogar kontextsensitiv!

Beweis: Falls $\lambda \notin L(G)$:

- Wähle $G'' = G'$ aus obigem Satz.

Falls $\lambda \in L(G)$:

- Bestimme P' wie oben,
- Wähle neues Symbol $S' \notin N \cup T$ und
- setze $G'' := (N \cup \{S'\}, T, P' \cup \{S' \to S, S' \to \lambda\}, S')$.

Grammatiken heißen *(schwach) äquivalent*, wenn sie die gleiche Sprache erzeugen.

Damit gilt:

- Zu jeder kontextfreien Grammatik gibt es eine äquivalente kontextsensitive Grammatik.

- Die Chomsky-Klassen $\mathcal{L}_0$, $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$ bilden eine durch Inklusion geordnete *Hierarchie*:

- Später werden wir sogar die Echtheit aller Inklusionen zeigen.

Eine Grammatik $G = (N, T, P, S)$ ist *in rechtslinearer Normalform*, wenn alle Produktionen von einer der Formen $A \to \lambda$, $A \to aB$ (mit $a \in T$, $A, B \in N$) sind.

Satz: Eine Sprache ist genau dann vom Typ 3 oder rechtslinear, wenn sie von einer Grammatik in rechtslinearer Normalform erzeugt wird.

Zu zeigen: Beweis durch Konstruktion einer rechtslinearen Normalform-Grammatik aus beliebiger rechtslinearer Grammatik

Schritt 1: Eliminination aller Regeln der Form $A \to w$ mit $w \in T^+$, $A \in N$.Dazu:

- neues Nichtterminalsymbol E
- neue Regel $E \to \lambda$
- ersetze alle Regeln $A \to w$ (mit $w \in T^+$, $A \in N$) durch $A \to wE$.

Neue Grammatik ist rechtslinear und erzeugt die gleiche Sprache.

Nur noch Regeln der Formen $A \to \lambda$, $A \to wB$ mit $w \in T^*$, $A, B \in N$.

Schritt 2: Elimination aller Regeln der Form $A \to wB$ mit $|w| > 1$. Dazu:

- für jede Regel der Form $A \to avB$ mit $a \in T$, $v \in T^+$, $A, B \in N$, neues Nichtterminal C,
- ersetze $A \to avB$ durch $A \to aC$ und $C \to vB$,
- iteriere die Ersetzung so lange wie möglich.

Neue Grammatik ist immer noch rechtslinear und erzeugt immer noch die gleiche Sprache.

Nur noch Regeln der Formen $A \to \lambda, A \to aB, A \to B$ mit $a \in T$, $A, B \in N$.

Schritt 3: Elimination aller Regeln der Form $A \to B$ (*Kettenregeln*). Dazu:

- Für jedes Regelpaar der Form $A \to B$, $B \to w$ füge Regel $A \to w$ hinzu
- iteriere das Hinzufügen so lange, wie neue Regeln entstehen.
- Entferne alle Regeln $A \to B$.

Neue Grammatik ist in rechtslinearer Normalform und erzeugt immer noch die gleiche Sprache.

Beispiel: Wir betrachten die rechtslineare Grammatik $G = (\{S, T\}, \{a, b\}, P, S)$ mit den folgenden Produktionen: $P = \{S \to aT, S \to T, S \to \lambda, T \to bbbT, T \to a\}$.
Wir konstruieren nach dem Verfahren aus dem Beweis des Satzes eine Grammatik G' in rechtslinearer Normalform, die die gleiche Sprache erzeugt.
Schritt 1: Wir erhalten neues Nichtterminalzeichen E und die Regeln

$$\{S \to aT|T|\lambda, T \to bbbT|aE, E \to \lambda\}$$

Schritt 2: Ersetze (a) Regel $T \to bbbT$ durch zwei neue Regeln $T \to bU$ und $U \to bbT$ (mit einem neuen Nichtterminalsymbol U) und dann (b) Regel $U \to bbT$ durch zwei neue Regeln $U \to bV$ und $V \to bT$.

$$\rightsquigarrow \{S \to aT|T|\lambda, T \to bU|aE, E \to \lambda, U \to bV, V \to bT\}$$

Schritt 3: Eliminiere Kettenregel $S \to T$ durch $S \to bU$ und $S \to aE$.

$$P' = \{S \to aT|bU|aE|\lambda, T \to bU|aE, E \to \lambda, U \to bV, V \to bT\}$$

Die Grammatik $G' = (\{S, T, U, V, E\}, \{a, b\}, P', S)$ ist in rechtslinearer Normalform und erzeugt die gleiche Sprache wie die Grammatik G.

Endliche Automaten als Ersetzungssysteme

Ersetzungssysteme sind allgemeiner als Grammatiken.
So kann man endliche Automaten wie folgt beschreiben:
Ein Ersetzungssystem $E = (\Sigma, P)$ heißt *endlicher Automat*, falls
$\Sigma = Z \cup T$, $Z \cap T = \emptyset$, $P \subseteq ZT \times Z$.
Z: Zustände; T: Eingabezeichen
Sei $z_0 \in Z$ Anfangszustand und $Z_f \subseteq Z$ Endzustände, so *akzeptiert* (E, z_0, Z_f) die Sprache:

$$L_a(E, z_0, Z_f) = \{w \in T^* \mid \exists z_f \in Z_f : z_0 w \Rightarrow^* z_f\}$$

Aufgaben: 1. Geben Sie in diesem Formalismus einen endlichen Automaten an, der die Sprache $a^*bb^*c^*$ beschreibt.
2. Wie kann man in diesem Formalismus einen deterministischen endlichen Automaten beschreiben?

Typ-3: Endliche Automaten und rechtslineare Grammatiken.

Satz: Die Typ-3-Sprachen sind genau die von endlichen Automaten akzeptierten Sprachen.

Hinweis: Der Ableitung

$$S \Rightarrow_G a_1A_1 \Rightarrow_G a_1a_2A_2 \overset{*}{\Rightarrow}_G a_1 \cdots a_nA_n \Rightarrow_G a_1 \cdots a_n$$

einer rechtslinearen Grammatik G in rechtslinearer Normalform entspricht die (akzeptierende) Ableitung

$$Sa_1a_2 \cdots a_n \Rightarrow_A A_1a_2 \cdots a_n \overset{*}{\Rightarrow}_A A_n.$$

S ist also Startzustand, A_n Endzustand, falls $A_n \to \lambda$ Grammatik-Regel.

Der Regel $S \to_G a_1A_1$ (z.B.) entspricht die Automaten-Ersetzung $Sa_1 \to_A A_1$.

Ganz entsprechend sieht man die andere Simulation ein.

Zur Echtheit der Chomsky-Hierarchie

Wir geben im Folgenden Beispiele an, weshalb $\mathcal{L}_3 \subsetneq \mathcal{L}_2$ gilt.

Elementares Hilfsmittel: Das Pumping-Lemma (auch Iterationslemma genannt).

Notation: Die *Länge* eines Wortes werde $\ell(w)$ oder $|w|$ geschrieben.
Rekursive Definition für $w \in \Sigma^*$: $\ell(w) = 0$, falls $w = \lambda$, und $\ell(w) = 1 + \ell(v)$, falls $\exists a \in \Sigma : w = av$.

Das Pumping-Lemma

Satz: Zu jeder regulären Sprache L gibt es eine Zahl $n > 0$, sodass jedes Wort $w \in L$ mit $\ell(w) \geq n$ als Konkatenation $w = xyz$ dargestellt werden kann mit geeigneten x, y, z mit folgenden Eigenschaften:

1. $\ell(y) > 0$;
2. $\ell(xy) \leq n$;
3. $\forall i \geq 0 : xy^i z \in L$.

Hinweis: Die Umkehrung gilt nicht !

Zur Anwendung des Pumping-Lemmas (schematisch)

1. Wir vermuten, eine vorgegebene Sprache L ist nicht regulär.
2. Im Widerspruch zu unserer Annahme nehmen wir an, L wäre doch regulär. Dann gibt es die im Pumping-Lemma genannte Pumping-Konstante n.
3. Wir wählen ein geeignetes, hinreichend langes Wort $w \in L$ (d.h., $\ell(w) \geq n$). Dies ist der Schritt, wo man leicht "gut" oder "schlecht" wählt ! Bemerkung: Da wir ja vermuten, L ist nicht regulär, ist L insbesondere unendlich, d.h., zu jedem n finden wir ein $w \in L$ mit $\ell(w) \geq n$.
4. Wir diskutieren alle möglichen Zerlegungen $w = xyz$ mit $\ell(y) > 0$ und $\ell(xy) \leq n$ und zeigen für jede solche Zerlegung, dass es ein $i \geq 0$ gibt, sodass $xy^iz \notin L$ gilt.

Zur Anwendung des Pumping-Lemmas (ein geschickter Einsatz)

Betrachte $L = \{a^k b^k \mid k \in \mathbb{N}\}$.

Wäre L regulär, so gäbe es Pumping-Konstante n.

Betrachte $a^n b^n \in L$; denn: $\ell(a^n b^n) = 2n \geq n$ und Präfix der Länge n ist a^n (sehr schön).

Diskutiere $a^n b^n = xyz$ mit $\ell(xy) \leq n$ und $\ell(y) > 0$:
Offenbar ist $xy \in \{a\}^+$ und damit $y = a^m$ für ein $m > 0$.
$\rightsquigarrow$ Nullpumpen liefert $a^{n-m} b^n \notin L$, $\lightning$ zur Annahme, L wäre regulär.

Zur Anwendung des Pumping-Lemmas (ein ungeschickter Einsatz)

Betrachte $L = \{a^k b^k \mid k \in \mathbb{N}\}$.

Wäre L regulär, so gäbe es Pumping-Konstante n. Da L nur Wörter gerader Länge enthält, können wir annehmen, n ist gerade.

Betrachte $w = a^{n/2} b^{n/2} \in L$ mit $\ell(w) = n$.

Diskutiere $w = xyz$ mit $\ell(xy) \leq n$ und $\ell(y) > 0$:
Fall 1: $xy \in \{a\}^+$ (Nullpumpen ähnlich wie letzte Folie)
Fall 2: $xy = a^{n/2} b^m$ mit $m > 0$.
Fall 2a: $y \in \{b\}^+$ (Nullpumpen ähnlich wie letzte Folie)
Fall 2b: $y = a^r b^m$ mit $r, m > 0$. Dann liegt auch $xy^2z = a^{n/2} b^m a^r b^{n/2} \in L$ ↯ zur Struktur von L.

. . . und noch eine . . .

Betrachte $L = \{a^{k^2} \mid k \in \mathbb{N}\}$.

Wäre L regulär, so gäbe es Pumping-Konstante n für L.

Betrachte $w = a^{(n+1)^2} \in L$ mit $\ell(w) \geq n$.

Widerspruch ergibt sich durch Nullpumpen:
Genauer haben wir, dass für ein $0 < i \leq n$ stimmen muss, dass $a^{(n+1)^2-i} \in L$ gilt, im Widerspruch zu folgender Abschätzung, die $a^{(n+1)^2-i} = a^{r^2}$ annimmt:

$$r^2 \leq n^2 < n^2 + n + (n - i) + 1 = n^2 + 2n + 1 - i = (n + 1)^2 - i.$$

Der Beweis des Pumping-Lemmas

Ist L endlich, so ist die Aussage trivial mit $n := \max\{\ell(w) \mid w \in L\}$.

Ist L unendlich aber regulär, so wird L von einem endlichen Automaten A mit n Zuständen akzeptiert mit Anfangszustand z_0.

Betrachte ein Terminalwort $w = a_1 \dots a_m \in L$, $a_i \in \Sigma$, $m \geq n$.
Es gibt also Ableitung

$$z_0 w \Rightarrow z_1 a_2 \dots a_m \Rightarrow \cdots \Rightarrow z_n a_{n+1} \cdots a_m \overset{*}{\Rightarrow} z \in Z_e$$

Unter den Zuständen $z_0, \dots, z_n$ muss nach dem Schubfachprinzip ein Zustand zweimal erreicht werden, d.h. $\exists 0 \leq r < s \leq n : z_r = z_s$.
Also gilt: $z_r a_{r+1} \dots a_s \overset{*}{\Rightarrow} z_s = z_r$.
Wegen $r < s$ ist $a_{r+1} \dots a_s \neq \lambda$.
Mit $w = a_1 \cdots a_r a_{r+1} \dots a_s a_{s+1} \dots a_m$ werden daher alle Wörter

$$a_1 \cdots a_r (a_{r+1} \dots a_s)^i a_{s+1} \dots a_m$$

(mit $i \in \mathbb{N}$ beliebig) akzeptiert.
Mit $x = a_1 \dots a_r$ und $z = a_{s+1} \dots a_m$ folgt die Behauptung.