

Grundlagen der Theoretischen Informatik

Sommersemester 2017

26.04.2017

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Organisatorisches

1. Teilklausur: 14.06.2017 oder 21.06.2017 (Doodle)
2. Teilklausur: 28.07.2017, 10:00-11:00, Raum D 028

Bis jetzt

1. Terminologie
2. Endliche Automaten und reguläre Sprachen
3. Kellerautomaten und kontextfreie Sprachen
4. Turingmaschinen und rekursiv aufzählbare Sprachen
5. Berechenbarkeit, (Un-)Entscheidbarkeit
6. Komplexitätsklassen P und NP

Bis jetzt

- **Alphabete, Wörter**

- Operationen auf Wörtern

- Konkatenation, i -te Potenz, Reverse

- **Sprache**

- Operationen auf Sprachen

- Konkatenation, i -te Potenz, Reverse, Kleene-Hülle

- **Reguläre Ausdrücke**

- Reguläre Ausdrücke als Suchmuster für `grep`

Grammatik

- Beschreibt eine Sprache
- Menge von Regeln, mit deren Hilfe man Wörter ableiten kann

Grammatik

Definition (Grammatik)

Eine **Grammatik** G über einem Alphabet Σ ist ein Tupel

$$G = (V, T, R, S)$$

Dabei ist

- V eine endliche Menge von **Variablen**
- $T \subseteq \Sigma$ eine endliche Menge von **Terminalen** mit $V \cap T = \emptyset$
- R eine endliche Menge von **Regeln**
- $S \in V$ das **Startsymbol**

Grammatik

Definition (Regel)

Eine Regel ist ein Element

$$(P, Q) \in ((V \cup T)^* V (V \cup T)^*) \times (V \cup T)^*$$

Das heißt:

- P und Q sind Wörter über $(V \cup T)$
- P muss mindestens eine Variable enthalten
- Q ist beliebig

Bezeichnung:

P : Prämisse

Q : Conclusio

Grammatik

Schreibweise für Regeln

- Schreibweise für Regel (P, Q) :

$$P \rightarrow_G Q \quad \text{bzw.} \quad P \rightarrow Q$$

- Abkürzung für mehrere Regeln mit derselben Prämisse:

$$P \rightarrow Q_1 \mid Q_2 \mid Q_3 \quad \text{für} \quad P \rightarrow Q_1, P \rightarrow Q_2, P \rightarrow Q_3$$

Konvention (meistens)

- **Variablen** als **Großbuchstaben**
- **Terminale** als **Kleinbuchstaben**

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

1. $aktuellWort := S$ (Startsymbol)
2. Wähle eine Regel $P \rightarrow Q$, so dass P in $aktuellWort$ vorkommt
3. Ersetze (ein) Vorkommen von P in $aktuellWort$ durch Q
4. Falls $aktuellWort$ noch Variablen enthält (nicht terminal), GOTO 2

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

1. $aktuellWort := S$ (Startsymbol)
2. Wähle eine Regel $P \rightarrow Q$, so dass P in $aktuellWort$ vorkommt
3. Ersetze (ein) Vorkommen von P in $aktuellWort$ durch Q
4. Falls $aktuellWort$ noch Variablen enthält (nicht terminal), GOTO 2

Ausgabe: Das terminale Wort $aktuellWort$

Beachte: Die Berechnung

- ist nicht deterministisch (Auswahl der Regel)
- kann mehr als ein Ergebnis liefern (oder auch keines)
- kann in Endlosschleifen geraten

Beispiel

Grammatik $G_{ab} = (\{S\}, \{a, b\}, \{R_1, R_2\}, S)$

$$R_1 = S \rightarrow aSb$$

$$R_2 = S \rightarrow \epsilon$$

Mögliche Ableitung:

$$S \Rightarrow_{R_1} aSb \Rightarrow_{R_1} aaSbb \Rightarrow_{R_1} aaaSbbb \Rightarrow_{R_2} aaabbb$$

Rechnung einer Grammatik

Gegeben:

- Grammatik $G = (V, T, R, S)$
- Wörter w, w' aus $(V \cup T)^*$

$$w \Longrightarrow_G^* w'$$

falls es Wörter $w_0, \dots, w_n \in (V \cup T)^*$ ($n \geq 0$) gibt mit

- $w = w_0$
- $w_n = w'$
- $w_i \Longrightarrow_G w_{i+1}$ für $0 \leq i < n$

Merke: $w \Longrightarrow_G^* w$ gilt stets ($n = 0$)

Die Folge w_0, \dots, w_n heißt **Ableitung** oder **Rechnung**

- von w_0 nach w_n
- in G
- der Länge n

Vorsicht: Indeterminismus

Beispiel

Wir betrachten die Grammatik $G = (\{S, B\}, \{a, b, c\}, \{R_0, R_1, R_2, R_3\}, S)$

$$R_0 = S \rightarrow aBBc$$

$$R_1 = B \rightarrow b$$

$$R_2 = B \rightarrow ba$$

$$R_3 = BB \rightarrow bBa$$

Vorsicht: Indeterminismus

Beispiel

Wir betrachten die Grammatik $G = (\{S, B\}, \{a, b, c\}, \{R_0, R_1, R_2, R_3\}, S)$

$$R_0 = S \rightarrow aBBc$$

$$R_1 = B \rightarrow b$$

$$R_2 = B \rightarrow ba$$

$$R_3 = BB \rightarrow bBa$$

Drei Möglichkeiten, das Wort *abbac* zu erzeugen:

$$S \xRightarrow{R_0} aBBc \xRightarrow{R_1} abBc \xRightarrow{R_2} abbac$$

$$S \xRightarrow{R_0} aBBc \xRightarrow{R_2} aBbac \xRightarrow{R_1} abbac$$

$$S \xRightarrow{R_0} aBBc \xRightarrow{R_3} abBac \xRightarrow{R_1} abbac$$

Vorsicht: Indeterminismus

Warum ist das ein Feature und kein Bug?

- Erlaubt einfachere Definition von Grammatiken
- Für manche Sprachen gibt es keine eindeutigen Grammatiken
- Eine Grammatik beschreibt die **Struktur** der Wörter.
Ein Wort kann mehrere mögliche Strukturen haben.
- Für **natürliche Sprachen** braucht man das unbedingt:
Manche Sätze sind mehrdeutig (in ihrer Grammatik),
also müssen auch die Grammatiken mehrdeutig sein!

Vorsicht: Indeterminismus

Beispiel: Mehrdeutige Grammatik natürlichsprachlicher Sätze

Time flies like an arrow.

Fruit flies like a banana.

Vorsicht: Indeterminismus

Beispiel: Mehrdeutige Grammatik natürlichsprachlicher Sätze

Time flies like an arrow.
Fruit flies like a banana.

- Beide Sätze haben zwei mögliche grammatische Strukturen.
- Erst unser semantisches Verständnis wählt eine aus.

Erzeugte Sprache, Äquivalenz

Definition (Erzeugte Sprache)

Gegeben: Eine Grammatik G

Die von G erzeugte Sprache $L(G)$ ist die Menge aller **terminalen** Wörter, die durch G vom Startsymbol S aus erzeugt werden können:

$$L(G) := \{w \in T^* \mid S \Longrightarrow_G^* w\}$$

Erzeugte Sprache, Äquivalenz

Definition (Erzeugte Sprache)

Gegeben: Eine Grammatik G

Die von G erzeugte Sprache $L(G)$ ist die Menge aller **terminalen** Wörter, die durch G vom Startsymbol S aus erzeugt werden können:

$$L(G) := \{w \in T^* \mid S \Longrightarrow_G^* w\}$$

Definition (Äquivalenz)

Zwei Grammatiken G_1, G_2 heißen **äquivalent** gdw

$$L(G_1) = L(G_2)$$

Beispiel

Grammatik $G_{ab} = (\{S\}, \{a, b\}, \{R_1, R_2\}, S)$

$$R_1 = S \rightarrow aSb$$

$$R_2 = S \rightarrow \epsilon$$

Mögliche Ableitung:

$$S \Rightarrow_{R_1} aSb \Rightarrow_{R_1} aaSbb \Rightarrow_{R_1} aaaSbbb \Rightarrow_{R_2} aaabbb$$

Also: $a^3b^3 \in L(G_{ab})$

$$L(G_{ab}) = \{a^n b^n \mid n \in \mathbb{N}\}$$

Beispiel

Grammatik $G_{ab} = (\{S\}, \{a, b\}, \{R_1, R_2\}, S)$

$$R_1 = S \rightarrow aSb$$

$$R_2 = S \rightarrow \epsilon$$

$$L(G_{ab}) = \{a^n b^n \mid n \in \mathbb{N}\}$$

Beweis:

Dass G_{ab} tatsächlich genau diese Sprache erzeugt, zeigen wir allgemein, indem wir alle möglichen Ableitungen von G_{ab} betrachten.

- \subseteq Zu zeigen: Jedes terminale Wort, das von G_{ab} erzeugt wird, hat die Form $a^n b^n$. Induktion über die Länge der Ableitung
- \supseteq Zu zeigen: Für alle n kann $a^n b^n$ von G_{ab} erzeugt werden.

Beweis

\subseteq : zu zeigen: Jedes terminale Wort, das von G_{ab} erzeugt wird, hat die Form $a^n b^n$.

Wir zeigen für alle $w \in (V \cup T)^*$: Falls $S \xRightarrow{*}_{G_{ab}} w$, dann gilt entweder $w = a^n S b^n$ oder $w = a^n b^n$ für ein $n \in \mathbb{N}$.

Dazu verwenden wir eine **Induktion über die Länge einer Ableitung** von S nach w .

Beweis

\subseteq : zu zeigen: Jedes terminale Wort, das von G_{ab} erzeugt wird, hat die Form $a^n b^n$.

Wir zeigen für alle $w \in (V \cup T)^*$: Falls $S \xRightarrow{*}_{G_{ab}} w$, dann gilt entweder $w = a^n S b^n$ oder $w = a^n b^n$ für ein $n \in \mathbb{N}$.

Dazu verwenden wir eine **Induktion über die Länge einer Ableitung** von S nach w .

Induktionsanfang: $w = S = a^0 S b^0$

Beweis

\subseteq : zu zeigen: Jedes terminale Wort, das von G_{ab} erzeugt wird, hat die Form $a^n b^n$.

Wir zeigen für alle $w \in (V \cup T)^*$: Falls $S \xRightarrow{*}_{G_{ab}} w$, dann gilt entweder $w = a^n S b^n$ oder $w = a^n b^n$ für ein $n \in \mathbb{N}$.

Dazu verwenden wir eine **Induktion über die Länge einer Ableitung** von S nach w .

Induktionsanfang: $w = S = a^0 S b^0$

Induktionsschritt: Es gelte $S \xRightarrow{*}_{G_{ab}} w \xRightarrow{G_{ab}} w'$, und für w gelte nach der Induktionsvoraussetzung bereits $w = a^n b^n$ oder $w = a^n S b^n$. Außerdem sei $w \xRightarrow{G_{ab}} w'$ eine Ableitung in einem Schritt. Nun ist zu zeigen: $w' = a^m b^m$ oder $w' = a^m S b^m$ für irgendein m .

Beweis

Fall 1: $w = a^n b^n$. Dann konnte keine Regel angewandt werden, da w schon terminal ist, also tritt dieser Fall nie auf.

Fall 2: $w = a^n S b^n$. Dann wurde von w nach w' entweder Regel R_1 oder R_2 angewandt.

Falls R_1 angewandt wurde, dann gilt $w = a^n S b^n \implies_{R_1} a^n a S b b^n = a^{n+1} S b^{n+1} = w'$.

Falls R_2 angewandt wurde, dann gilt $w = a^n S b^n \implies_{R_2} a^n \varepsilon b^n = w'$.
Dies Wort ist terminal und hat die geforderte Form $a^n b^n$.

Beweis Forts.

\supseteq : zu zeigen: Für alle n kann $a^n b^n$ von G_{ab} erzeugt werden: $S \xRightarrow{*}_{G_{ab}} a^n b^n$
 $\forall n \in \mathbb{N}$.

Um $a^n b^n$ zu erzeugen, wende man auf S n -mal die Regel R_1 und dann einmal die Regel R_2 an. \square

Beispiel: Dycksprache

Definition (Dycksprache)

Gegeben:

- $k \in \mathbb{N}$
- $\Sigma_k := \{x_1, \bar{x}_1, x_2, \dots, x_k, \bar{x}_k\}$ ein Alphabet mit $2k$ Symbolen

Beispiel: Dycksprache

Definition (Dycksprache)

Gegeben:

- $k \in \mathbb{N}$
- $\Sigma_k := \{x_1, \bar{x}_1, x_2, \dots, x_k, \bar{x}_k\}$ ein Alphabet mit $2k$ Symbolen

Die Dycksprache D_k ist die **kleinste Menge**, die folgende Bedingungen erfüllt:

1. $\epsilon \in D_k$,
2. Falls $w \in D_k$, so auch $x_i w \bar{x}_i$.
3. Falls $u, v \in D_k$, so auch uv .

Beispiel: Dycksprache

Definition (Dycksprache)

Gegeben:

- $k \in \mathbb{N}$
- $\Sigma_k := \{x_1, \bar{x}_1, x_2, \dots, x_k, \bar{x}_k\}$ ein Alphabet mit $2k$ Symbolen

Die Dycksprache D_k ist die kleinste Menge, die folgende Bedingungen erfüllt:

1. $\epsilon \in D_k$,
2. Falls $w \in D_k$, so auch $x_i w \bar{x}_i$.
3. Falls $u, v \in D_k$, so auch uv .

Interpretiert man die x_i als öffnende, die \bar{x}_i als zugehörige schließende Klammern, so kann man die Dycksprache als die **Menge aller korrekten Klammerausdrücke** sehen.

Beispiel: Dycksprache

$$k = 2$$

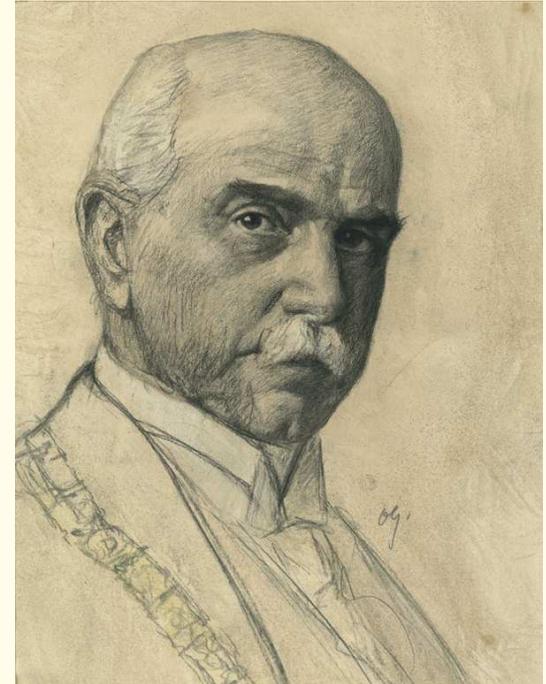
$$\Sigma_2 := \{ [,], (,) \}$$

- $[[()]]() \in D_2$
- $([]) \notin D_2$
- $)) \notin D_2$

Beispiel: Dycksprache

Walther von Dyck (1856-1934)

- Mathematiker
(Gruppentheorie, Funktionstheorie)
- Herausgeber der Werke von Johannes Kepler
- Hochschulpolitiker
- Erster Rektor der TU München
- Einer der Gründungsväter des Deutschen Museums



[Foto: Deutsches Museum]

Beispiel: Dycksprache

Definition (Dycksprache)

Gegeben:

- $k \in \mathbb{N}$, $\Sigma_k := \{x_1, \bar{x}_1, x_2, \dots, x_k, \bar{x}_k\}$ ein Alphabet mit $2k$ Symbolen

Die Dycksprache D_k ist die kleinste Menge, die folgende Bedingungen erfüllt:

1. $\epsilon \in D_k$,
2. Falls $w \in D_k$, so auch $x_i w \bar{x}_i$.
3. Falls $u, v \in D_k$, so auch uv .

Grammatik $G = (\{S\}, \{x_1, \bar{x}_1, x_2, \dots, x_k, \bar{x}_k\}, R, S)$ mit $L(G) = D_k$

$$R_1 : S \rightarrow \epsilon$$

$$R_2 : S \rightarrow x_1 S \bar{x}_1 \mid \dots \mid x_k S \bar{x}_k$$

$$R_3 : S \rightarrow SS$$

Beispiel: Dycksprache

Um zu zeigen, dass $L(G) = D_k$, zeigen wir, dass $L(G)$ die kleinste Menge ist, die folgende Bedingungen erfüllt:

1. $\epsilon \in L(G)$,
2. Falls $w \in L(G)$, so auch $x_i w \bar{x}_i$.
3. Falls $u, v \in L(G)$, so auch uv .

Beweis: an der Tafel.

Bis jetzt

- Reguläre Ausdrücke.
- Grammatik.
- Ableitung.
- die von einer Grammatik erzeugte Sprache.

Warum Sprachen?

Darstellung von Problemen

Fakt: So ziemlich alle Probleme können als Probleme über Sprachen formuliert werden.

Warum Sprachen?

Darstellung von Problemen

Fakt: So ziemlich alle Probleme können als Probleme über Sprachen formuliert werden.

Beispiel: Primzahlen

Alphabet $\Sigma_{\text{num}} := \{|\}$

Sprache $L_{\text{primes}} := \{ \underbrace{|| \dots |}_{p \text{ mal}} \mid p \text{ prim} \}$

Darstellung von Problemen

Eingabealphabet

$$\Sigma = \{0, 1, \dots, n - 1\}$$

erlaubt Darstellung einer Ganzzahl zur Basis n

Darstellung von Problemen

Eingabealphabet

$$\Sigma = \{0, 1, \dots, n - 1\}$$

erlaubt Darstellung einer Ganzzahl zur Basis n

Beispiel:

5 binär: 101

5 unär: ||||| (oder auch 11111)

Darstellung von Problemen

Speicheraufwand

n -äre Darstellung ($n > 1$) einer Zahl k führt zu einer Speicherersparnis:

$\log_n k$ (n -är) statt k (unär)

Nur der Schritt von unär auf binär ist wesentlich, denn

$$\log_n k = \frac{1}{\log_2 n} \cdot \log_2 k = c \cdot \log_2 k$$

(von binär auf n -är nur lineare Einsparung)

Darstellung von Problemen

Nächste Vorlesung:

- Darstellung des Erfüllbarkeitsproblems SAT
- Darstellung des Erreichbarkeitsproblems in Graphen