

Grundlagen der Theoretischen Informatik

Wintersemester 2024/25

Prof. Dr. Heribert Vollmer

Institut für Theoretische Informatik
Leibniz Universität Hannover

Link für Kurzfragen:



URL: <https://pingo.coactum.de/events/739935>

Bitte scannen!



URL: <https://pingo.coactum.de/events/739935>

Organisatorisches

- ▶ **Vorlesung** Mo 10.15h hier in E001, Aufzeichnung in Stud.IP (aus WS20/21) werden eine Woche vor Vorlesung bereitgestellt
- ▶ **Materialien** in Stud.IP: Skript, Folien
- ▶ **Übungen** in Kleingruppen mit max. 25 TN, Beginn: **heute!**
- ▶ **Hausübungen** werden dreimal im Semester verteilt;
vermutlich Mitte Nov., Mitte Dez., Ende Jan.;
Bearbeitungszeit 2 Wochen;
elektr. Abgabe (PDF) in Gruppen (1-4 TN)
- ▶ **Studienleistung:** 60% der Punkte der Hausübungen
bitte E-Mail an Vivian Holzapfel
- ▶ **Prüfungsleistung:** Klausur, 120 min,
geplanter Termin: 24.02.24, 14h;
für Lehramtsstudiengänge: mündl. Prüfung, Termin n. V.

Übungskonzept

- ▶ Tutorien **in Gruppen** (max. 25 TN)
- ▶ Vorbereitung: Vorlesung (ev. Aufzeichnung), Skript
- ▶ Keine Wiederholung des Vorlesungsinhaltes in der Übung!
- ▶ Aufgaben werden **in der Übung gerechnet**
- ▶ **Kleingruppen erarbeiten Lösungen**
- ▶ Lösungen werden **in Gesamtgruppe besprochen**

Inhalt

Sprachen und Grammatiken

Die Chomsky-Hierarchie

Reguläre (Typ-3-) Sprachen

Endliche Automaten

Nichtdeterministische endliche
Automaten

Endliche Automaten und
Typ-3-Grammatiken

Das Pumping Lemma für
reguläre Sprachen

Kontextfreie (Typ-2-) Sprachen

Kellerautomaten

Das Pumping-Lemma für
kontextfreie Sprachen

Typ-1- und Typ-0-Sprachen

Der intuitive Berechenbarkeitsbegriff

Berechenbarkeit durch Maschinen

Turing-Berechenbarkeit

Mehrband-Maschinen

Berechenbarkeit in
Programmiersprachen

Die Programmiersprache LOOP

Die Programmiersprache WHILE

Die Church'sche These

Entscheidbarkeit und Aufzählbarkeit

Unentscheidbare Probleme

Das Halteproblem

Der Satz von Rice

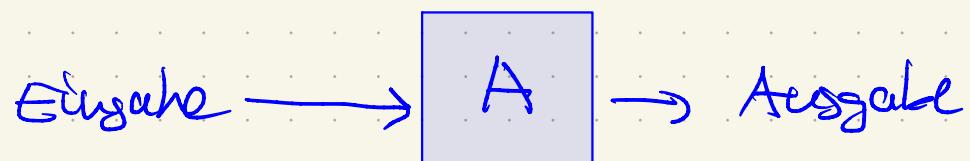
Algorithmisches Problem / Aufgabe :

Eingabedaten \rightarrow Aufgabe
(z.B. Sortieren) \rightarrow Ausgabedaten

mathematisch:

Eingabe \rightarrow f
(Vorlage) (Funktionswert)
Bild

Definitionsbereich M $f: M \rightarrow N$ Wertebereich N
(Menge der Eingaben)



Formale Sprachen: math. Präzisierung der Darstellung
von Ein-/Ausgabe

Sprachen und Grammatiken

Alphabete, Zeichen und Symbole

Ein **Alphabet** ist eine endliche, nichtleere Menge. Die Elemente eines Alphabets heißen auch **Zeichen** oder **Symbole**.

Wie üblich: Ist M eine Menge, so bezeichnet $|M|$ die Anzahl der Elemente von M .

Beispiele:

$$\Sigma_{kkl} = \{a, b, c, \dots, A, B, C, \dots, \ddot{a}, \ddot{t}, \dots, \beta, *, /, \gamma, \dots\}$$

$$\Sigma_{bin} = \{0, 1\}$$

$$|\Sigma_{bin}| = 2$$

Wörter und Sprachen

Sei Σ ein Alphabet.

Ein **Wort über Σ** ist eine Folge von Symbolen aus Σ .

Ein Wort entsteht also durch **Hintereinanderschreiben** (**Konkatenation**) von Symbolen aus Σ .

Mit ϵ wird das leere Wort bezeichnet.

Wörter über

$$-\Sigma_{\text{bin}} : \quad 10010$$

$$-\Sigma_{\text{abcd}} : \quad abba \quad (3+1) \times 4$$

Wörter und Sprachen

Die Menge aller Wörter über dem Alphabet Σ bezeichnen wir mit Σ^* . Eine Sprache über Σ ist eine Menge von Wörtern über Σ , also eine Teilmenge von Σ^* .

Σ_{bin}^* = Menge der Binärwörter

= $\{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \dots\}$
„quasitextolographische Ordnung“

Menge der Binärwörter, die geraden Zahlen entsprechen

$L_{\text{ger}} = \{0, 10, 100, 110, \dots\} \subseteq \Sigma_{\text{bin}}^*$

„Wörter Σ_{bin}^* “

$L_{\text{Java}} = \text{synt-korrekte Java-Programme}$

Konkatenation

- ▶ Operation auf Wörtern: Konkatenation bzw. Hintereinanderschreiben
- ▶ Schreibweise: $u \circ v$ oder kurz uv für Konkatenation der Wörter u und v
- ▶ Für ein Wort w und $n \in \mathbb{N}$ ist w^n die Konkatenation
$$w^n = \underbrace{w \circ w \circ \cdots \circ w}_{n-\text{mal}}$$
- ▶ Wir definieren: $w^0 = \varepsilon$.

$$\text{abba} \circ \text{baab} = \text{abbaaab}$$

Länge

- Die **Länge** eines Wortes w ist die Anzahl der Symbole in w .
Schreibweise: $|w|$
- $|\varepsilon| = 0$.
- Es ist $|w^n| = n|w|$.

$$|w|=k$$

Schreibweise: $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$

$$|\mathcal{M}|$$

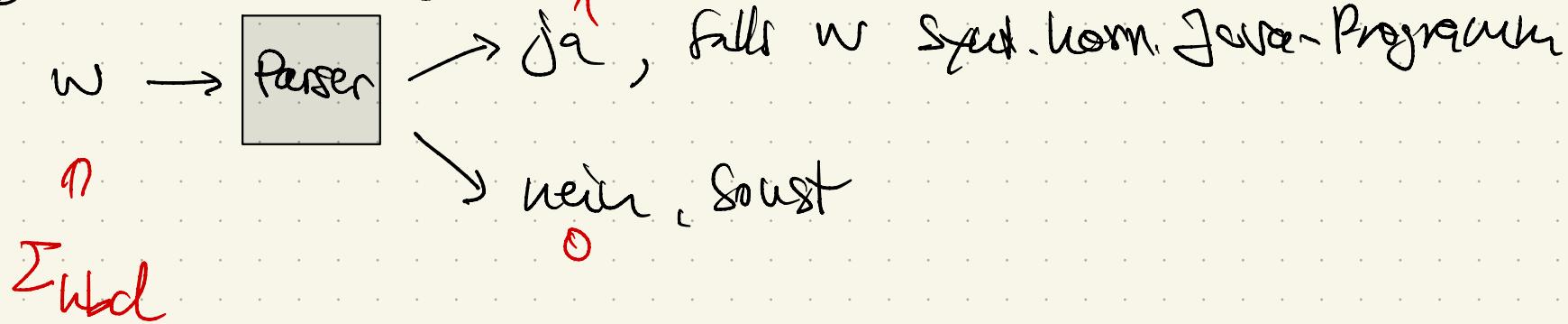
$$|w|$$

Algorithmisches Problem:

$$f: \Sigma^* \rightarrow \Delta^*$$

(Σ, Δ : Alphabet)

Programmiersprache Java:



$$f: \Sigma_{\text{Wld}}^* \rightarrow \{0, 1\}$$

$$f(w) = \begin{cases} 1, & \text{falls } w \in L_{\text{Java}} \\ 0, & \text{sonst} \end{cases}$$

„Entscheidungsproblem“

$|\Sigma^*| = \infty$ f.a. Σ .

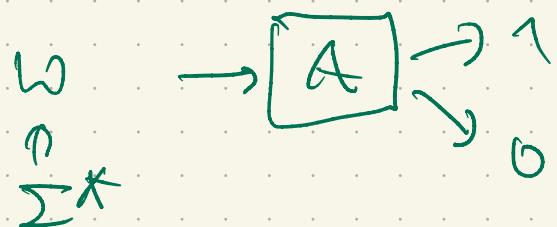
$L \subseteq \Sigma^*$ ist o.A. unendlich.

Problem: Wie kann L auf endliche Weise beschrieben werden?

Bsp.: endl. Beschreibung von L über
durch Text oder Syntaxdiagramme

Möglichkeit:

- Parser / Entscheidungsalgorithmus für $L \subseteq \Sigma^*$



- Grammatiken

Syntax der Aussagenlogik: Beispiel für EBNF

$\phi ::= p \mid 0 \mid 1 \mid \neg\phi \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\phi \leftrightarrow \phi),$

wobei p eine aussagenlogische Variable ist, also
 $p \in \{p_1, p_2, p_3, \dots\}.$

Bsp. für ein Grammatik:

$$\begin{cases} G = (V, \Sigma, R, S) \text{ mit } \Sigma = \Sigma_{\text{Wdl}} \\ V = \{\langle \text{EXP} \rangle, \langle \text{DIG} \rangle, \langle \text{OP} \rangle, \langle \text{NAT} \rangle\}, S = \langle \text{EXP} \rangle \end{cases}$$

L = arithm. Ausdrücke
mit Operatoren +, -, *, /
 η
 Σ^* über nat. Zahlen
 Σ_{Wdl}

$$\begin{array}{c} (3+1)*4 \in L \\ \hline (3+1)* \notin L \\ (3+1)*4 \notin L \end{array} \quad 47$$

R:

$$\begin{array}{ll} \langle \text{EXP} \rangle \rightarrow \langle \text{EXP} \rangle \langle \text{OP} \rangle \langle \text{EXP} \rangle & (1) \\ \langle \text{EXP} \rangle \rightarrow (\langle \text{EXP} \rangle) & (2) \\ \langle \text{EXP} \rangle \rightarrow \langle \text{NAT} \rangle & (3) \\ \langle \text{NAT} \rangle \rightarrow \langle \text{DIG} \rangle & (4) \\ \langle \text{NAT} \rangle \rightarrow \langle \text{DIG} \rangle \langle \text{NAT} \rangle & (5) \end{array}$$

$$\begin{array}{ll} \langle \text{OP} \rangle \rightarrow + & (6) \\ \langle \text{OP} \rangle \rightarrow - & (7) \\ \langle \text{OP} \rangle \rightarrow * & (8) \\ \langle \text{OP} \rangle \rightarrow / & (9) \end{array}$$

$$\langle \text{DIG} \rangle \rightarrow 0 \quad (10)$$

$$\langle \text{DIG} \rangle \rightarrow 1 \quad (11)$$

$$\langle \text{DIG} \rangle \rightarrow 2 \quad (12)$$

$$\begin{array}{l} \vdots \\ \langle \text{DIG} \rangle \rightarrow 3 \quad (13) \end{array}$$

Bsp. für Ableitung:

$$\begin{array}{l} \langle \text{EXP} \rangle \xrightarrow{(1)} \langle \text{EXP} \rangle \langle \text{OP} \rangle \langle \text{EXP} \rangle \\ \xrightarrow{(8)} X \langle \text{EXP} \rangle * Y \langle \text{EXP} \rangle \\ \xrightarrow{(3)} X \langle \text{EXP} \rangle * \langle \text{NAT} \rangle Y \\ \xrightarrow{(4)} X \langle \text{EXP} \rangle * \langle \text{DIG} \rangle Y \\ \xrightarrow{(14)} \langle \text{EXP} \rangle * 4 \xrightarrow{(2)} (\langle \text{EXP} \rangle) * 4 \xrightarrow{*} (3+1)*4 \end{array}$$

$u \Rightarrow_v v$
 $s \Rightarrow^* w$
 $\eta \Sigma^*$

Definition

Eine **Grammatik** ist ein **4-Tupel** $G = (V, \Sigma, P, S)$, wobei:

- ▶ V ist eine endliche Menge, die so genannte **Variablen**
- ▶ Σ ist ein Alphabet, das so genannte **Terminalalphabet**, mit $V \cap \Sigma = \emptyset$
- ▶ P ist die endliche Menge der **Produktionen**,
 $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$
- ▶ $S \in V$ ist die so genannte **Startvariable**

Definition

Sei $G = (V, \Sigma, P, S)$ eine Grammatik und seien $u, v \in (V \cup \Sigma)^*$.

Wir definieren eine Relation \Rightarrow_G wie folgt:

- ▶ $u \Rightarrow_G v$, falls u, v zerlegt werden können in Teilwörter $u = xyz$ und $v = xy'z$ mit $x, z \in (V \cup \Sigma)^*$ und $y \rightarrow y'$ ist Regel in P .
„ u geht unter (Anwendung einer Regel in) G unmittelbar über in v “
- ▶ $u \Rightarrow_G^* v$, falls $u = v$ oder es Wörter $w_1, \dots, w_k \in (V \cup \Sigma)^*$ gibt mit $u = w_1, w_i \Rightarrow_G w_{i+1}$ für $i = 1, 2, \dots, k-1$ und $v = w_k$.

Wir lassen den Index G weg, falls dieser eindeutig ist.

Die von G erzeugte Sprache ist $L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$.

Eine Ableitung von $w \in L(G)$ in k Schritten ist eine Folge (w_0, w_1, \dots, w_k) mit $w_0 = S$, $w_k = w$ und $w_i \Rightarrow_G w_{i+1}$ für $i = 0, 1, \dots, k-1$.