

Theoretische Grundlagen der Informatik

Vorlesung am 21.10.2014

INSTITUT FÜR THEORETISCHE INFORMATIK



- Team
- Vorlesungsaufzeichnungen
- Homepage
- Übungsblätter
- Tutorien
- Klausur

Team

Vorlesung:

Prof. Dorothea Wagner

Übung:

Benjamin Niedermann (Benjamin.Niedermann@kit.edu)

Franziska Wegner (Franziska.Wegner@kit.edu)

Tutorium:

Stefan Altmayer, Julien Duman, Peter Eisenmann, Robin Freyler
Michael Fürst, Stefan Hanßen, Harald Herrlich, Jan Jacob,
Moritz Klammler, Fabian Klute, Stefan Rettenmayr, Tobias Ribizel,
Michael Schrempp, Tobias Stolzmann, Simon Sudrich, Oliver Thal,
Jérôme Urhausen, Kai Wallisch

- Ton und Folien der Vorlesungen und Übungen werden aufgezeichnet.
- Aufnahmen sind abrufbar über YouTube Kanal und iTunes U Kanal:
 - Aufnahmen werden nach den Vorlesungen online gestellt (mit gewisser Verzögerung).
 - Für Links siehe Vorlesungshomepage.

- <http://i11www.iti.kit.edu/>
- Aktuelle Informationen / Termine
- Alte Klausuren
- Skript
- Folien
- Übungsblätter
- Forum
 - Für Fragen an die Übungsleiter
 - Für Austausch untereinander
- Links zu Vorlesungsaufzeichnungen
- Literaturempfehlungen

- Ingo Wegener
Theoretische Informatik
B.G. Teubner Verlag Stuttgart, 1993
- Uwe Schöning
Theoretische Informatik - kurzgefasst
Hochschultaschenbuch, Spektrum Akademischer Verlag, 1997

- R. Garey und D. S. Johnson
Computers and Intractability: A Guide to the Theory of NP-Completeness
W. H. Freeman, New York, 1979
- T. Cormen, C. Leiserson, R. Rivest
Introduction to Algorithms
The MIT press, 1997, 2001.
- A. Asteroth, C. Baier
Theoretische Informatik: eine Einführung in Berechenbarkeit, Komplexität und formale Sprachen mit 101 Beispielen
Pearson Studium, 2002

In der Regel:

- Ausgabe etwa jeden zweiten Dienstag
- Bearbeitungszeit: 1-2 Wochen (siehe Abgabedatum auf Übungsblatt!)
- Abgabe: Kasten im Untergeschoss des Informatik-Hauptgebäudes (50.34)
- Abgabe bis Dienstag, 11.00 Uhr im Kasten
- Rückgabe der korrigierten Blätter in den Tutorien
- Lösungen in der Übung
- Ausgabe des ersten Übungsblatts:
Dienstag 21.10.2014

- Doppelabgabe erlaubt, aber nur, wenn beide für das gleiche Tutorium eingeteilt sind
- Kopf des Blattes:
 - Name(n)
 - Vorname(n)
 - Matrikelnummer(n)
 - Tutoriumsnummer
 - Name des Tutors
- Keine losen Blätter, bitte zusammenheften
- Übungsblätter müssen handschriftlich sein, also nicht gedruckt oder kopiert
- Bei Abschreiben: Keine Punkte
- Ab 50% der erreichbaren Punkte gibt es einen Klausurbonus
- Der Klausurbonus wird nur auf bestandene Klausuren angerechnet

- Rückgabe der Übungsblätter
- Zusätzliche Aufgaben und Beispiele zum Stoff der Vorlesung
- Start: Nächste Woche (ab 27.10.2014)
- Einteilung über webinscribe, Homepage:
<http://webinscribe.informatik.kit.edu/>
- Dort verlinkt: Merkblatt und Termine der Tutorien
- Anmeldebeginn: Dienstag, 21.10., 18:00 Uhr
- Anmeldeschluss: Donnerstag, 23.10., 18:00 Uhr!

- Klausur: 20.02.2015 um 8:00 Uhr.
- Dauer: 2 Stunden
- Orientierung: Klausuren auf der Homepage
 - TGI-Klausuren aus den Semestern WS10/11 und WS11/12.
 - *Informatik III*-Klausuren, **aber**: leicht veränderte Stoffwahl und *Informatik III*-Klausuren waren für eine Stunde ausgelegt.
- Klausurbonus ab 50% der erreichbaren Übungsblattpunkte
- Klausurbonus wird nur auf bestandene Klausuren angerechnet
- Termin für Nachklausur wird noch bekannt gegeben

Welche Fragestellungen werden in TGI behandelt?

- Theoretische Grundlagen zu Algorithmen- und Programmentwurf: Wie kann man allgemeingültige (rechner- und programmiersprachenunabhängige) Aussagen zu gegebenen Problemstellungen machen?
- Gibt es Probleme die nicht von Computern gelöst werden können?
- Gibt es Probleme, für die Ausprobieren die beste Lösungsstrategie ist?
- Wie kann man konkrete Computer abstrakt betrachten oder modellieren?
- Wie kann man konkrete Programmiersprachen abstrakt betrachten oder modellieren?

- Vertiefter Einblick in die Grundlagen der Theoretischen Informatik
- Beherrschen der Berechnungsmodelle und Beweistechniken der TI
- Verständnis für Grenzen und Möglichkeiten der Informatik in Bezug auf die Lösung von definierbaren aber nur bedingt berechenbaren Problemen
- Abstraktionsvermögen für grundlegende Aspekte der Informatik von konkreten Gegebenheiten wie konkreten Rechnern oder Programmiersprachen
- Anwendung der erlernten Beweistechniken bei der Spezifikation von Systemen der Informatik und für den systematischen Entwurf von Programmen und Algorithmen

- Wörter
- Formale Sprachen
- Reguläre Ausdrücke
- Endliche Automaten
- Kontextfreie Grammatiken

- Ein *Alphabet* Σ ist eine endliche Menge von Zeichen.
Beispiel: $\Sigma = \{0, 1\}$.
- Ein *Wort* w über einem Alphabet Σ ist eine (möglicherweise leere) Folge von Zeichen aus Σ .
Beispiel: $\Sigma = \{0, 1\}$ und $w = 0010010$.
- Das *leere Wort* wird mit ε symbolisiert.
- Die *Menge aller Wörter* über einem Alphabet Σ wird mit Σ^* abgekürzt.
Beispiel:
 $\Sigma = \{0, 1\}$ und $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$

- Die Menge aller Wörter mit Länge n über einem Alphabet Σ wird mit Σ^n abgekürzt. Beispiel $\Sigma = \{0, 1\}$:

$$\Sigma^0 = \{\varepsilon\}$$

$$\Sigma^1 = \{0, 1\}$$

$$\Sigma^2 = \{00, 01, 10, 11\}$$

- Die *Konkatenation* (Aneinanderhängen) zweier Wörter w_1 und w_2 wird mit $w_1 \cdot w_2$ oder $w_1 w_2$ abgekürzt.
Beispiel: $w_1 = 001$ und $w_2 = 110$ dann $w_1 \cdot w_2 = 001110$.

- Die *iterierte Konkatination* eines Wortes w ist $w^k := \underbrace{w \cdot \dots \cdot w}_{k\text{-mal}}$

Beispiel: $w = 110$.

$$\begin{aligned}w^0 &= \epsilon \\w^1 &= 110 \\w^2 &= 110110 \\w^3 &= 110110110\end{aligned}$$

- Eine *formale Sprache* L über einem Alphabet Σ ist eine Teilmenge $L \subseteq \Sigma^*$.

Beispiel: Sprache L' aller Wörter deren vorletztes Zeichen 0 ist

$$L' = \{w0z \mid w \in \Sigma^*, z \in \Sigma\}$$

- Das Produkt $L_1 \cdot L_2$ der Sprachen L_1 und L_2 ist definiert als

$$L_1 \cdot L_2 := \{w_1 w_2 \mid w_1 \in L_1 \text{ und } w_2 \in L_2\}$$

Beispiel:

$$L' =$$

- Eine *formale Sprache* L über einem Alphabet Σ ist eine Teilmenge $L \subseteq \Sigma^*$.

Beispiel: Sprache L' aller Wörter deren vorletztes Zeichen 0 ist

$$L' = \{w0z \mid w \in \Sigma^*, z \in \Sigma\}$$

- Das Produkt $L_1 \cdot L_2$ der Sprachen L_1 und L_2 ist definiert als

$$L_1 \cdot L_2 := \{w_1 w_2 \mid w_1 \in L_1 \text{ und } w_2 \in L_2\}$$

Beispiel:

$$L' = \Sigma^* \cdot \{00, 01\}$$

- Produkte einer Sprache L mit sich selbst werden abgekürzt als

$$L^k := \underbrace{L \cdot \dots \cdot L}_{k\text{-mal}}$$

Beispiel $L = \{00, 1\}$:

$$L^0 = \{\varepsilon\}$$

$$L^1 = \{00, 1\}$$

$$L^2 = \{00, 1\} \cdot \{00, 1\} = \{0000, 001, 100, 11\}$$

$$\begin{aligned} L^3 &= \{0000, 001, 100, 11\} \cdot \{00, 1\} \\ &= \{000000, 00100, 10000, 1100, 00001, 0011, 1001, 111\} \end{aligned}$$

Läßt sich ein Wort w schreiben als $w = u \cdot v \cdot x$, wobei u, v, x beliebige Wörter sind, so heißt:

$$\left. \begin{array}{l} u \quad \text{Präfix} \\ v \quad \text{Teilwort} \\ x \quad \text{Suffix} \end{array} \right\} \text{ von } w$$

Beispiel $w = \text{TAL}$

$$\begin{array}{ll} \text{Präfixe} & P = \{ \varepsilon, T, TA, TAL \} \\ \text{Suffixe} & S = \{ \varepsilon, L, AL, TAL \} \\ \text{Teilworte} & \{ A \} \cup P \cup S \end{array}$$

Seien $L, L_1, L_2 \subseteq \Sigma^*$ Sprachen.

Produktsprache $L_1 \cdot L_2 := \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$

k -faches Produkt $L^k := \{w_1 \cdot w_2 \cdot \dots \cdot w_k \mid w_i \in L \text{ für } 1 \leq i \leq k\}$
 $L^0 := \{\epsilon\}$

Quotientensprache $L_1 / L_2 := \{w \in \Sigma^* \mid \exists z \in L_2 \text{ mit } w \cdot z \in L_1\}$

Kleene'scher Abschluss $L^* := \bigcup_{i \geq 0} L^i$

Positiver Abschluss $L^+ := \bigcup_{i \geq 1} L^i$

Komplementsprache $L^c := \Sigma^* \setminus L$

Eine Sprache $L \subseteq \Sigma^*$ heißt *regulär*, wenn für sie einer der folgenden Punkte gilt: (induktive Definition)

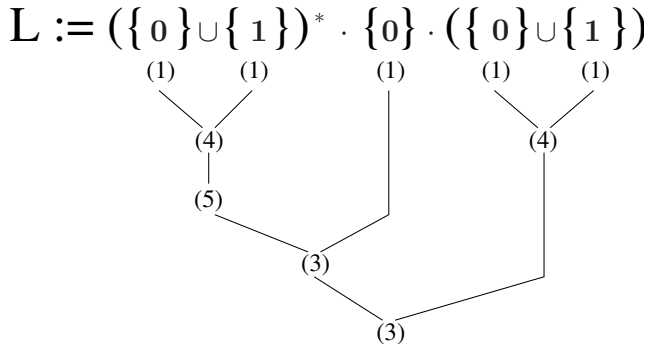
1 Verankerung:

- 1 $L = \{a\}$ mit $a \in \Sigma$ oder
- 2 $L = \emptyset$

2 Induktion: Seien L_1, L_2 reguläre Sprachen

- 1 $L = L_1 \cdot L_2$ oder
- 2 $L = L_1 \cup L_2$ oder
- 3 $L = L_1^*$

Beispiel: Die Sprache aller Wörter über $\{0, 1\}$, die als vorletztes Zeichen eine 0 haben



- Wir benutzen eine leicht andere Schreibweise für reguläre Ausdrücke als in GBI.
- Sei Σ eine Alphabet. Eine reguläre Sprache über Σ kann durch einen *regulären Ausdruck* beschrieben werden.

Dabei bezeichnet

- \emptyset den regulären Ausdruck, der die leere Menge beschreibt.
- ε den regulären Ausdruck, der die Menge $\{\varepsilon\}$ beschreibt.
- a den regulären Ausdruck, der die Menge $\{a\}$ beschreibt.

Wenn α, β reguläre Ausdrücke sind, die die Sprachen $L(\alpha), L(\beta)$ beschreiben, so schreiben wir

- $(\alpha) \cup (\beta)$ für $L(\alpha) \cup L(\beta)$
- $(\alpha) \cdot (\beta)$ für $L(\alpha) \cdot L(\beta)$
- $(\alpha)^+$ für $L(\alpha)^+$
- $(\alpha)^*$ für $L(\alpha)^*$

Notation

- Wir schreiben auch α statt $L(\alpha)$ und $w \in \alpha$ statt $w \in L(\alpha)$.
- * bindet stärker als \cdot und \cdot stärker als \cup
- Das heißt zum Beispiel:

$$a \cup b \cdot c = a \cup (b \cdot c)$$

- Wir lassen unnötige Klammern weg.

- L ist der reguläre Ausdruck für die Sprache aller Wörter über $\{0, 1\}$, die als vorletztes Zeichen eine 0 haben

$$L = (0 \cup 1)^* 0 (0 \cup 1)$$

- $L := \{w \in \{0, 1\}^* \mid w \text{ enthält } 10 \text{ als Teilwort}\}$

$$L = (0 \cup 1)^* 10 (0 \cup 1)^*$$

- $L := \{w \in \{0, 1\}^* \mid w \text{ enthält } 101 \text{ als Teilwort}\}$

$$L = (0 \cup 1)^* 101 (0 \cup 1)^*$$

- $L := \{w \in \{0, 1\}^* \mid w \text{ enthält } 10 \text{ nicht als Teilwort}\} = 0^*1^*$,

denn

- $w \in 0^*1^* \Rightarrow w \in L$; also ist $0^*1^* \subseteq L$.
- Sei $w \in L$. Dann kommen nach der ersten Eins keine Nullen mehr vor, d.h. $w = w'1 \dots 1$ wobei w' keine 1 enthält. Also ist $w \in 0^*1^*$.

- In GBI wurden Mealy- und Moore-Automaten behandelt.
- In TGI werden nur endliche Akzeptoren benötigt.

Ein (deterministischer) endlicher Automat DEA $(Q, \Sigma, \delta, s, F)$ besteht aus:

- Q , einer endlichen Menge von *Zuständen*
- Σ , einer endlichen Menge von *Eingabesymbolen*
- $\delta: Q \times \Sigma \rightarrow Q$, einer *Übergangsfunktion*
- $s \in Q$, einem *Startzustand*;
- $F \subseteq Q$, einer Menge von *Endzuständen*.

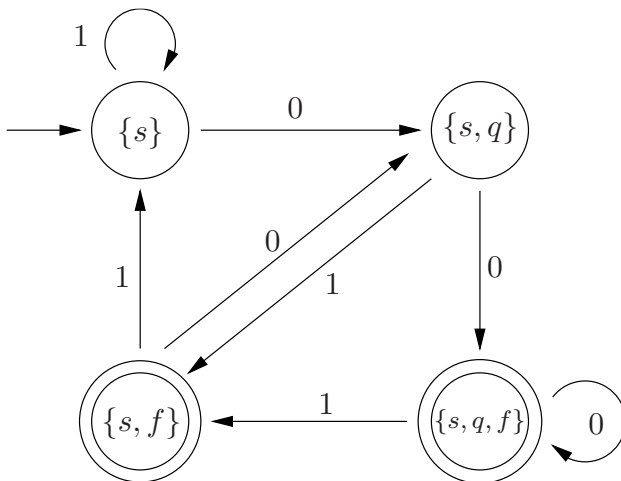
Der Automat heißt

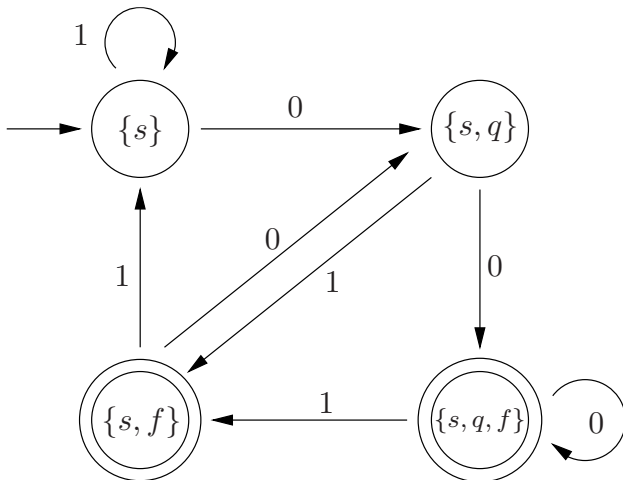
- endlich, da die Zustandsmenge (vgl. mit Speicher, Gedächtnis) endlich ist.
- deterministisch, da δ eine Funktion ist und der Automat somit in jedem Schritt eindeutig arbeitet. Es gibt keine Zufälligkeiten oder Wahlmöglichkeiten.

Was kann ein endlicher Automat?

- Gegeben ist eine Eingabe als endliche Folge von Eingabesymbolen. Der Automat entscheidet, ob die Eingabe zulässig ist oder nicht, indem er in einem Endzustand endet oder nicht.

- Ein endlicher Automat **erkennt** oder **akzeptiert** eine Sprache L , d.h. eine Menge von Wörtern über dem Alphabet des Automaten, wenn er nach Abarbeitung eines Wortes w genau dann in einem Endzustand ist, wenn das Wort w in der Sprache L ist ($w \in L$).
- Eine formale Sprache heißt *endliche Automatensprache*, wenn es einen endlichen Automaten gibt, der sie erkennt.





Sprache aller Wörter, deren vorletztes Symbol 0 ist: $L = (0 \cup 1)^* 0 (0 \cup 1)$

Eine kontextfreie Grammatik $G = (\Sigma, V, S, R)$ ist gegeben durch

- ein endliches Alphabet Σ (auch Terminalalphabet genannt)
- eine endlichen Menge V mit $V \cap \Sigma = \emptyset$ von Variablen (auch Nichtterminale genannt)
- ein Startsymbol $S \in V$
- eine endlichen Menge von Ableitungsregeln R ,
d.h. durch eine Menge von Tupeln $(l, r) \in V \times (\Sigma \cup V)^*$
(auch Produktionen genannt)

Wir schreiben Produktionen auch in der Form $l \rightarrow r$.

Gegeben ist eine Regel $l \rightarrow r$.

- Wenn in einem Wort w das Zeichen l vorhanden ist, darf l in w durch r ersetzt werden.
- Wir schreiben $w \rightarrow z$, wenn w durch Anwendung *einer* Ableitungsregel in z verwandelt wird.
- Wir schreiben $w \xrightarrow{*} z$, wenn w durch Anwendung *mehrerer* Ableitungsregel in z verwandelt wird.

Die von einer Grammatik $G = (\Sigma, V, S, R)$ erzeugte Sprache $L(G)$ ist die Menge aller Wörter $z \in \Sigma^*$ für die gilt $S \xrightarrow{*} z$.

Beispiel Kontextfreie Grammatiken

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit

$$\Sigma := \{0, 1\}$$

$$V := \{S\}$$

$$R := \{S \rightarrow 01, S \rightarrow 0S1\} \text{ wir schreiben dafür kurz } \{S \rightarrow 01|0S1\}$$

Welche Wörter erzeugt G , d.h. was ist $L(G)$?

$$\begin{aligned} S &\rightarrow 01|0S1 \rightarrow 01|0011|00S11 \\ &\rightarrow 01|0011|000111|000S111 \rightarrow \dots \end{aligned}$$

Es ist $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$.

Beispiel Kontextfreie Grammatiken

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit

$$\Sigma := \{0, 1\}$$

$$V := \{S\}$$

$$R := \{S \rightarrow 01, S \rightarrow 0S1\} \text{ wir schreiben dafür kurz } \{S \rightarrow 01|0S1\}$$

Welche Wörter erzeugt G , d.h. was ist $L(G)$?

$$\begin{aligned} S &\rightarrow 01|0S1 \rightarrow 01|0011|00S11 \\ &\rightarrow 01|0011|000111|000S111 \rightarrow \dots \end{aligned}$$

Es ist $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$.

Beispiel Kontextfreie Grammatiken

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit

$$\Sigma := \{0, 1\}$$

$$V := \{S\}$$

$$R := \{S \rightarrow 01, S \rightarrow 0S1\} \text{ wir schreiben dafür kurz } \{S \rightarrow 01|0S1\}$$

Welche Wörter erzeugt G , d.h. was ist $L(G)$?

$$\begin{aligned} S &\rightarrow 01|0S1 \rightarrow 01|0011|00S11 \\ &\rightarrow 01|0011|000111|000S111 \rightarrow \dots \end{aligned}$$

Es ist $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$.

Beispiel Kontextfreie Grammatiken

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit

$$\Sigma := \{0, 1\}$$

$$V := \{S\}$$

$$R := \{S \rightarrow 01, S \rightarrow 0S1\} \text{ wir schreiben dafür kurz } \{S \rightarrow 01|0S1\}$$

Welche Wörter erzeugt G , d.h. was ist $L(G)$?

$$\begin{aligned} S &\rightarrow 01|0S1 \rightarrow 01|0011|00S11 \\ &\rightarrow 01|0011|000111|000S111 \rightarrow \dots \end{aligned}$$

Es ist $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$.

Beispiel Kontextfreie Grammatiken

Welche Grammatik erzeugt die Sprache über dem Alphabet $\Sigma = \{0, 1\}$ deren vorletztes Zeichen 0 ist?

Beispiel Kontextfreie Grammatiken

Welche Grammatik erzeugt die Sprache über dem Alphabet $\Sigma = \{0, 1\}$ deren vorletztes Zeichen 0 ist?

Grammatik $G = (\Sigma, V, S, R)$ mit

$$\Sigma := \{0, 1\}$$

$$V := \{S, A, B\}$$

$$R := \{S \rightarrow A0|A1, \quad A \rightarrow B0, \quad B \rightarrow \epsilon|B0|B1\}$$

Ableitung:

$$S \rightarrow A0|A1 \rightarrow B00|B01 \rightarrow \dots$$

In TGI werden wir Grammatiken kennenlernen, deren Produktionen *mächtiger* sind, als die von kontextfreien Grammatiken.