Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik

Vorlesung am 25.10.2022

Torsten Ueckerdt | 25. Oktober 2022

Heute

Organisatorisches

- Team, Termine
- Vorlesungsaufzeichnungen
- Homepage, Literaturempfehlungen
- Übungsblätter, Klausur
- ILIAS und Forum

Wiederholung aus GBI

- Wörter, formale Sprachen
- Reguläre Sprachen, reguläre Ausdrücke
- Endliche Automaten
- Kontextfreie Grammatiken

Team

Übung:

Laura Merker
laura.merker2@kit.edu

Adrian Feilhauer
adrian.feilhauer@kit.edu

Vorlesung:

Torsten Ueckerdt

Termine (Änderungen vorbehalten)

Dienstags		Donnerstags	
25.10.	Vorlesung	27.10.	Vorlesung
01.11.	—	03.11.	Vorlesung
08.11.	Übung	10.11.	Vorlesung
15.11.	Vorlesung	17.11.	Übung
22.11.	Vorlesung	14.11.	Vorlesung
29.11.	Übung	01.12.	Vorlesung
06.12.	Vorlesung	08.12.	Übung
13.12.	Vorlesung	15.12.	Vorlesung
20.12.	Übung	22.12.	Vorlesung

Dienstags		Donnerstags	
10.01.	Vorlesung	12.01.	Vorlesung
17.01.	Übung	19.01.	Vorlesung
24.01.	Vorlesung	26.01.	Übung
31.01.	Vorlesung	02.02.	Vorlesung
07.02.	Übung	09.02.	Vorlesung
14.02.	Vorlesung	16.02.	Übung

Homepage und ILIAS

Homepage → `https://i11www.iti.kit.edu/teaching/winter2022/tgi/index`

- Aktuelle Informationen / Termine
- Alte Klausuren
- Übungsblätter und Musterlösungen
- Skripte und Literaturempfehlungen

ILIAS-Forum →
`https://ilias.studium.kit.edu/goto.php?target=crs_1938235&client_id=produktiv`

- Für Fragen
- Für Austausch untereinander
- Vorlesungsaufzeichnungen, Folien
- Übungsblätter

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 Intractibility: A Guide to the Theory of NP-Completeness
  W. H. Freeman, New York, 1979
- A. Asteroth, C. Baier:
  Theoretische Informatik: eine Einführung in Berechenbarkeit, Komplexität und formale Sprachen mit 101 Beispielen
  Pearson Studium, 2002
- J. Hromkovič: **Theoretical Computer Science**
  Springer Verlag, Berlin-Heidelberg, 2011

Übungsblätter

In der Regel:

- Ausgabe etwa alle zwei Wochen
- In derselben Übung: Besprechung von ähnlichen Aufgaben
- Bearbeitungszeit: 1-2 Wochen (siehe Abgabedatum auf Übungsblatt!)
- Abgabe fristgerecht im ILIAS
- Rückgabe und Besprechung der korrigierten Blätter in ILIAS / Tutorien
- Lösungen auf der Webseite
- Ausgabe des ersten Übungsblatts: Freitag, 28.10.2022

Übungsblätter

- Doppelabgabe erlaubt, wenn beide für das gleiche Tutorium eingeteilt sind
- Übungsblätter müssen handschriftlich sein
- kein Abschreiben!
- 25% der möglichen Gesamtpunktzahl $\rightsquigarrow$ 1 Bonuspunkt
  50% der möglichen Gesamtpunktzahl $\rightsquigarrow$ 2 Bonuspunkte
  75% der möglichen Gesamtpunktzahl $\rightsquigarrow$ 3 Bonuspunkte
- Der Klausurbonus wird nur auf bestandene Klausuren angerechnet

Klausur

- Hauptklausur: 14. März 2023 um 14:30 Uhr.
- Dauer: 2 Stunden
- Orientierung: Altklausuren auf der Homepage, Übungsblätter
- Termin für Nachklausur wird noch bekannt gegeben.

Tutorien

- Rückgabe und Besprechung der Übungsblätter
- Zusätzliche Aufgaben und Beispiele zum Stoff der Vorlesung
- Start: Nächste Woche (ab 31.10.2022)
- Einteilung über WebInScribe
  - `http://webinscribe.informatik.kit.edu/`
  - nur bis Donnerstag, 27.10.2022, 18:00 Uhr!
- Genauere Informationen auf der Homepage

Ziele der Vorlesung TGI

Welche Fragestellungen werden in TGI behandelt?

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

Ziele der Vorlesung TGI

Welche Kompetenzen sollen Sie erlernen?

- 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 Rechnern oder Programmiersprachen
- Anwendung der erlernten Beweistechniken

Wiederholung aus Grundbegriffe der Informatik

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

Wörter

- Ein *Alphabet* Σ ist eine endliche Menge von Zeichen.

  Beispiele: $\Sigma = \{0, 1\}$ oder $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \cup \{,\} \cup \{.\}$.

- Ein *Wort* w über einem Alphabet Σ ist eine (möglicherweise leere) Folge von Zeichen aus Σ.

  Beispiel: $\Sigma = \{0, 1\}$ und $w = 0010010$.

  $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \cup \{,\} \cup \{.\}$ und $w = 1.024,48$

- Das *leere Wort* wird mit ε symbolisiert.

- Die *Menge aller Wörter* über einem Alphabet Σ wird mit Σ^* abgekürzt.

  Beispiel: $\Sigma = \{0, 1\}$ und $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

  $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \cup \{,\} \cup \{.\}$

  $\rightsquigarrow$ ε und 1..0.4 und , , .7 etc. in Σ^*

Wörter

- Die *Menge aller Wörter* mit Länge n über einem Alphabet Σ wird mit Σ^n abgekürzt.

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

  $$\begin{aligned} \Sigma^0 &= \{\varepsilon\} \\ \Sigma^1 &= \{0, 1\} \\ \Sigma^2 &= \{00, 01, 10, 11\} \end{aligned}$$

- 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 $w_1 \cdot w_2 = 001110$.

  $w_1 = 001$ und $w_2 = \varepsilon$ dann $w_1 \cdot w_2 = 001$.

Wörter

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

Beispiel: $w = 110$.

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

Formale Sprachen

- 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:

$$L' = \Sigma^* \cdot \{0\} \cdot \Sigma$$

Formale Sprachen

- Produkte einer Sprache L mit sich selbst werden abgekürzt als

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

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

$$\begin{aligned} L^0 &= \{\varepsilon\} \\ L^1 &= \{00, 1\} \\ L^2 &= \{00, 1\} \cdot \{00, 1\} = \{0000, 001, 100, 11\} \\ L^3 &= \{0000, 001, 100, 11\} \cdot \{00, 1\} \\ &= \{000000, 00100, 10000, 1100, 00001, 0011, 1001, 111\} \end{aligned}$$

Formale Sprachen

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

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

Beispiel $w = \text{TAL}$

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

Formale Sprachen

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	$L^k := \{w_1 \cdot \ldots \cdot w_k \mid w_i \in L \text{ für } 1 \leq i \leq k\}$ $L^1 = L, \quad L^0 := \{\varepsilon\}$
Kleene'scher Abschluss	$L^* := \bigcup_{i \geq 0} L^i$
Positiver Abschluss	$L^+ := \bigcup_{i \geq 1} L^i$
Quotientensprache	$L_1/L_2 := \{w \in \Sigma^* \mid \exists z \in L_2 \text{ mit } w \cdot z \in L_1\}$
Komplementsprache	$L^c := \Sigma^* \setminus L$

Reguläre Sprachen

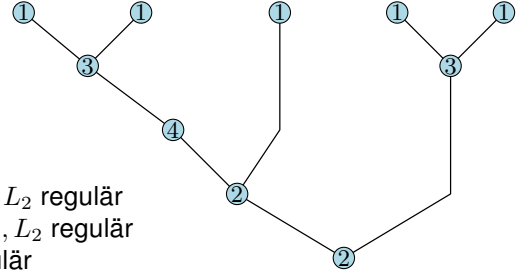
Eine Sprache $L \subseteq \Sigma^*$ heißt *regulär*, wenn für sie einer der folgenden Punkte gilt: (induktive Definition)

- Verankerung:
  - $L = \{a\}$ mit $a \in \Sigma$ oder
  - $L = \{\varepsilon\}$ oder
  - $L = \emptyset$
- Induktion: Es gibt reguläre Sprachen L_1, L_2 sodass
  - $L = L_1 \cdot L_2$ oder
  - $L = L_1 \cup L_2$ oder
  - $L = L_1^*$

Reguläre Sprachen

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

$$L' := (\{0\} \cup \{1\})^* \cdot \{0\} \cdot (\{0\} \cup \{1\})$$



Verankerung:

① $L = \{a\}$ mit $a \in \Sigma$

Induktion:

② $L = L_1 \cdot L_2$ mit L_1, L_2 regulär

③ $L = L_1 \cup L_2$ mit L_1, L_2 regulär

④ $L = L_1^*$ mit L_1 regulär

Reguläre Ausdrücke

- 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.
- Mit $L(\alpha)$ bezeichnen wir die Sprache die über den regulären Ausdruck α beschrieben ist.
- *Achtung:* Verschiedene reguläre Ausdrücke können die gleiche Sprache beschreiben.

$$L(\alpha) = L(\beta) \text{ trotz } \alpha \neq \beta \text{ ist möglich!}$$

Reguläre Ausdrücke

Dabei bezeichnet

- $\emptyset$ den regulären Ausdruck, der die leere Menge beschreibt.
  $$\rightsquigarrow L(\emptyset) = \{\}$$
- ε den regulären Ausdruck, der die Menge $\{\varepsilon\}$ beschreibt.
  $$\rightsquigarrow L(\varepsilon) = \{\varepsilon\}$$
- a den regulären Ausdruck, der die Menge $\{a\}$ beschreibt.
  $$\rightsquigarrow L(a) = \{a\}$$

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 $L(\alpha)^*$

Reguläre Ausdrücke

Notation

- Wir manchmal 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:

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

- Wir lassen Konkatenationspunkte und unnötige Klammern weg.

Reguläre Ausdrücke - Beispiele

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

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

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

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

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

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

Reguläre Ausdrücke - Beispiele

- $L := \{w \in \{0,1\}^* \mid w \text{ enthält 10 nicht als Teilwort}\} = 0^*1^*$, denn
  - Sei $w \in 0^*1^*$. Dann kommt in w nach keiner 1 eine 0.
    Also ist $w \in L$ und damit $0^*1^* \subseteq L$.
  - Sei $w \in L$. Dann kommen nach der ersten Eins keine Nullen mehr vor,
    d.h. $w = w'1 \ldots 1$ wobei w' keine 1 enthält. Also ist $w \in 0^*1^*$ und damit $L \subseteq 0^*1^*$.

Reguläre Ausdrücke - Beispiele

- $L := \{w \in \{0,1\}^* \mid w \text{ enthält 10 nicht als Teilwort}\} = 0^*1^*$, denn
  - Sei $w \in 0^*1^*$. Dann kommt in w nach keiner 1 eine 0.
    Also ist $w \in L$ und damit $0^*1^* \subseteq L$.
  - Sei $w \in L$. Dann kommen nach der ersten Eins keine Nullen mehr vor,
    d.h. $w = w'1 \ldots 1$ wobei w' keine 1 enthält. Also ist $w \in 0^*1^*$ und damit $L \subseteq 0^*1^*$.

Testen Sie sich:

Sei $\Sigma = \{0,1,2,3,4,5,6,7,8,9\} \cup \{,\} \cup \{.\}$.
Sei $L_{\mathbb{R}} \subseteq \Sigma^*$ die Sprache aller endlichen Dezimaldarstellungen von positiven reellen Zahlen.

Beispiele: $1.024,48 \in L_{\mathbb{R}}$ und $42 \in L_{\mathbb{R}}$
aber $1024,48 \notin L_{\mathbb{R}}$ und $042,00 \notin L_{\mathbb{R}}$ und $,,.123 \notin L_{\mathbb{R}}$

$\rightsquigarrow$ Finden Sie einen regulären Ausdruck für $L_{\mathbb{R}}$?

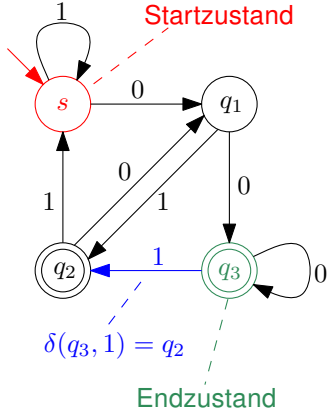
Endliche Automaten

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

Endliche Automaten

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

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



Endliche Automaten

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.

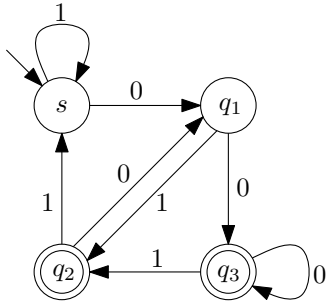
Endliche Automaten

- Ein endlicher Automat **erkennt** oder **akzeptiert** eine Sprache L, d.h. eine Menge von Wörtern über dem Alphabet Σ 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.

Endliche Automaten

Der Automat

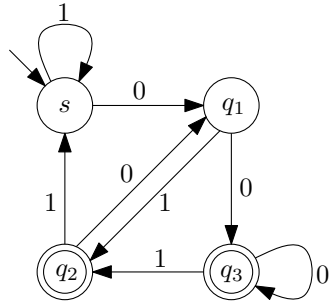
- akzeptiert $w = 001$.
- akzeptiert nicht $w = 110$.



Endliche Automaten

Der Automat

- akzeptiert $w = 001$.
- akzeptiert nicht $w = 110$.



Der Automat erkennt die Sprache L' aller Wörter,
deren vorletztes Zeichen 0 ist: $L' = (0 \cup 1)^*0(0 \cup 1)$

$\rightsquigarrow$ L' ist eine endliche Automatensprache.