



Informatik III Grundlagen der theoretischen Informatik

Peter Sanders

Übungen:

Thomas Käußl

Roman Dementiev und Johannes Singler

Institut für theoretische Informatik, Algorithmik II



Organisatorisches

Vorlesungen: Dienstags und Donnerstags 11.30–13.00 Uhr

Saalübung: Freitags 9.45–11.15 Uhr, HSaF (erstmalig 3.11.)

Tutorübungen: Passwortausgabe RZ (Geb. 20.21) Raum -109:

Di. 24.10.06 17:00–18:00 Uhr

Mi. 25.10.06 10:00–14:00 Uhr

Do. 26.10.06 10:00–14:00 Uhr

Eintragung: Webinscribe webinscribe.ira.uka.de/

Di. 24.10.06 17:00–Do. 26.10.06 18:00



Organisatorisches

Übungsblätter:

- Ausgabe: Veröffentlichung auf den Webseiten der Vorlesung – Donnerstags ab 14 Uhr (erstmalig 26.10.)
- Abgabe: Eine Woche später – Freitags, bis 9.45 Uhr
- Ort: Einwurfskasten beim Raum –118, Informatikgebäude
- Eine bestimmter Anteil der Punkte (1/3–1/2)
 - ↪ Schein
 - ↪ Zulassung Vordiplom (Info)/Prüfungszulassung (InfWi)
- Kein** Einfluß auf die Klausurnote



Organisatorisches

Sprechstunde:

- Peter Sanders, Dienstag 15–16 Uhr, Raum 217
- Thomas Käußl, Montag 11–12 Uhr, Raum 207

Klausuren:

Mo. 26.2.2007, Beginn 9 Uhr

Di. 10.4.2007, Beginn 14 Uhr

Web: <http://algo2.iti.uka.de/info3/>

Alle Angaben auch auf den Webseiten der Vorlesung.



Materialien

Folien, Übungsblätter

Bücher: hauptsächlich Schöning

Skript als Ergänzung

Wikipedia!



Schöning: Theoretische Informatik

— kurzgefaßt

- 4. Auflage, Spektrum Verlag, 2001, 20 Euro
- Vorlesung: ähnliche Inhalte.
Andere Reihenfolge
- Kurz aber verständlich
- Beweise manchmal nicht so ausführlich



Bücher

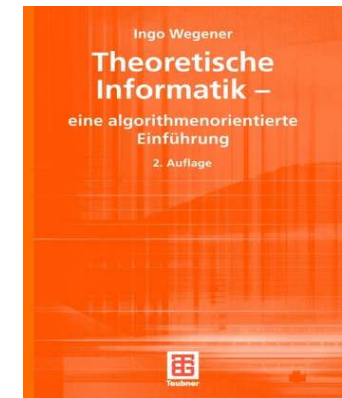
- Wie Sand am Meer
- Vergleichbare Inhalte
- Gut zum Nachlesen und für zusätzliche Motivationen



Wegener: Theoretische Informatik

— eine algorithmenorientierte Einführung

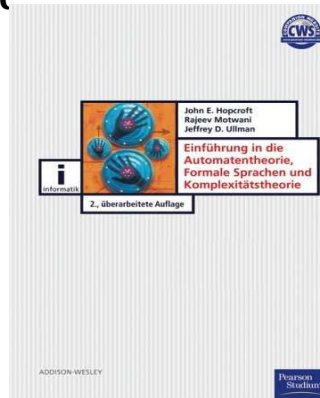
- 2. Auflage, Teubner, 1999, 22 Euro
- Gute Anwendungsmotivationen
- Zusätzliche, informellere Darstellung:
Kompendium Theoretische Informatik



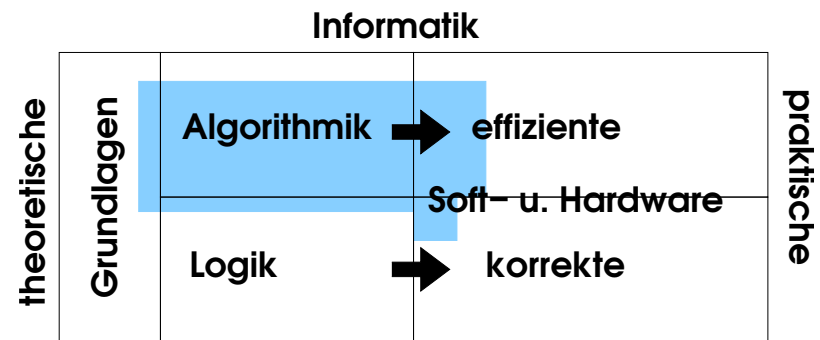


Hopcroft, Motwani, Ullman Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie

- 2. Auflage, Addison-Wesley, 2002, 40 Euro
 - Gute Anwendungsmotivationen
 - Wenig Stoff sehr ausführlich erklärt
 - Geeignet zum Selbststudium?
- ≠ altes Hopcroft Ullman Buch



Informatik



Einführung

Grundlagen theoretischer Informatik

Formale Sprachen: Wie kommunizieren Computer?

Automatentheorie: Formale Modelle von Computern

Berechenbarkeit: Was können Computer (nicht)

Komplexitätstheorie: Was kann man (nicht) effizient berechnen



Warum das Ganze?

- (in)direkte **Anwendungen**
(Algorithmen, Konzepte, Methoden)
in Programmiersprachenentwurf, Compilerbau, Textverarbeitung,
(Verarbeitung natürlicher Sprache), Hardwareentwurf,
Softwaremodellierung, . . .
- Unmöglichkeitsresultate** ersparen uns blutige Nasen
- Übung mit informatikrelevanten **Beweistechniken**
↪ Algorithmentheorie, Logik, . . .
- Fester Boden auf dem Grund **philosophischer** Diskussionssümpfe:
Leib-Seele-Problem, KI, Turing-Test, . . .



Formale Sprachen

Notation

Alphabet: **endliche** Menge (Σ)

Wort (über Σ): aneinandergehängte Zeichen aus Σ (w)

formale Sprache: eine Menge von Wörtern (L)

leeres Wort: ε ($\{\varepsilon\} \neq \emptyset$!)

Konkatenation von Zeichenketten: \cdot assoziativ.

Beispiel: $ac \cdot bab = acbab$.

Produktsprache: $L_1 \cdot L_2 := \{w_1 \cdot w_2 : w_1 \in L_1 \wedge w_2 \in L_2\}$.

Beispiel: $\{ab, ba\} \cdot \{aa, bb\} = \{abaa, abbb, baaa, babb\}$



Notation

k -faches Produkt (Potenzierung): $w^0 := \varepsilon$, $w^n := w \cdot w^{n-1}$ für $n \geq 1$.

$L^0 := \{\varepsilon\}$, $L^n := L \cdot L^{n-1}$ für $n \geq 1$.

Beispiele $a^3 = aaa$, $\{a, bb\}^2 = \{aa, abb, bba, bbbb\}$

Kleensche Hülle: $L^* := \bigcup_{i=0}^{\infty} L^i$

(jedes einzelne Wort ist immer noch endlich!)

Beispiel: $\{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$

Σ^* : Menge der Wörter über dem Alphabet Σ .

$(\Sigma^*, \cdot, \varepsilon)$ ist ein **Monoid**.

positive Hülle: $L^+ := \bigcup_{i=1}^{\infty} L^i$

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

$|w|$: Anzahl Buchstaben in w



Notation

Sei $w = u \cdot v \cdot x$

Präfix: u

Teilwort: v

Suffix: x

Spiegelung: $(c_1 c_2 \dots c_k)^R = c_k \dots c_2 c_1$



Beispiele für formale Sprachen z.B. $\Sigma = \{0, 1\}$

$L^= := \{0^n 1^n : n \geq 1\} = \{01, 0011, 000111, \dots\}$

$\{a^n b^n c^n : n \in \mathbb{N}\} = \{abc, aabbcc, aaabbbccc, \dots\}$

$\{ww : w \in \Sigma^*\} = \{\varepsilon, 00, 11, 0000, 0101, 1010, 1111, \dots\}$

$\{ww^R : w \in \Sigma^*\} = \{\varepsilon, 00, 11, 0000, 0110, 1001, 1111, \dots\}$

$L_P := \{w : w = w^R\} = \{\varepsilon, 0, 1, 00, 11, 000, 010, 101, 111, \dots\}$

Palindrome, z.B. anna, otto, aua, gnudung, hangnah, kajak, lagerregal, reliefpfeiler, Saippukauppias (finnisch Seifenhändler), Sator Arepo Tenet Opera Rotas (Lateinisch; Übersetzung: Der Sähmann Arepo dreht mit Mühe die Räder), tragart „ein Neger mit Gazelle zagt im Regen nie“, „eine Hure bei Liebe ruhe nie“



Beispiele für formale Sprachen

Wohlgeformte Klammerausdrücke $L_{()}:$

- $\varepsilon \in L_{()},$
- $uv \in L_{()} \text{ falls } u \in L_{()}, v \in L_{()},$
- $(u) \in L_{()} \text{ falls } u \in L_{()}.$



Typische Fragestellungen

Wortproblem: $w \in L?$

Äquivalenz: $L_1 = L_2?$

Spezifikation: mathematische Definition (siehe oben), **Grammatiken**,

Akzeptoren=Automaten (Maschinen), die Wörter w lesen und
'sagen' ob $w \in L$

Umwandlung zwischen verschiedenen Spezifikationen



Warum formale Sprachen?

- Programmiersprachen**, **Dateiformate**,... sind formale Sprachen.
- Eingaben** im weitesten Sinne (z.B. Impulsfolgen an Pins von Chips) lassen sich als formale Sprachen **auffassen**
- Einfache** klar definierte Fragestellungen
- Die meisten algorithmischen Fragestellung lassen sich auf Wortprobleme zurückführen

„wenn wir formale Sprachen verstehen, verstehen wir die Informatik“



Beispiel Rucksackproblem



- n Gegenstände mit **Gewicht** $w_i \in \mathbb{N}$ und **profit** p_i
- Wähle eine Teilmenge **x** von Gegenständen
- so dass $\sum_{i \in x} w_i \leq W$ und
- maximiere den Profit** $\sum_{i \in x} p_i$



Beispiel Rucksackproblem

definiere $L \subseteq \{0, 1, '\cdot\}^*$:

$w \in L$ falls w eine kommaseparierte Liste von $2n + 2$ Binärzahlen

$P, W, w_1, p_1, \dots, w_n, p_n$ ist, so dass

$\exists \mathbf{x} \subseteq \{1, \dots, n\} : \sum_{i \in \mathbf{x}} p_i \geq P$ und $\sum_{i \in \mathbf{x}} w_i \leq W$

Wortproblem für $L \rightsquigarrow$ Rucksackproblem:

- rate optimales P durch **binäre Suche**
- finde \mathbf{x} durch Weglassen einzelner Elemente

Insgesamt $\leq n + \log \sum_i p_i$ Wortprobleme lösen

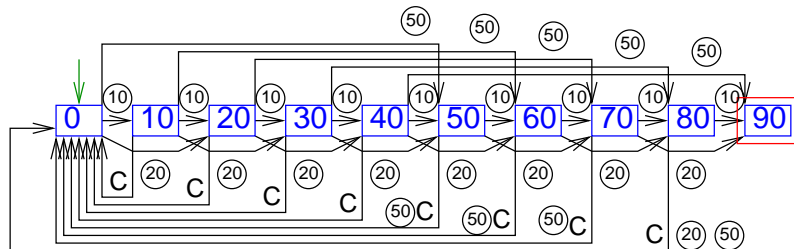
„wenig“



Beispiel: einfacher Fahrkartenautomat

- Einheitsfahrpreis **90 Cents**
- 10, 20** und **50** Centstücke werden akzeptiert
- Abbruch bei Taste C oder zu viel Geld
- Genau 90 Cents eingeworfen \rightsquigarrow fertig

$(\{10, 20, 50, C\}, \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90\}, \delta, 0, \{90\})$

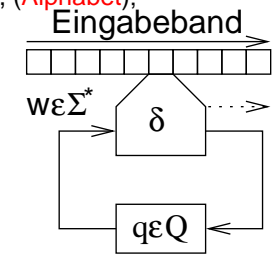


Automatentheorie

Endliche Automaten

Ein deterministischer endlicher Automat (oder Akzeptor) besteht aus:

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



Mächtiger Sprachen und Maschinen

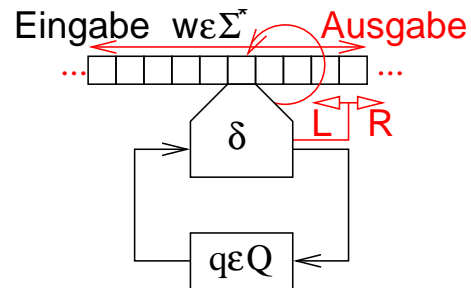
Chomsky-Hierarchie: Grammatiken, Automatenmodelle, und passende Familien von formalen Sprachen.

Insbesondere: kontextfreie Sprachen. Zum Beispiel zwecks Spezifikation des Syntax von Programmiersprachen.

- Kontextfreie Sprachen effizient erkennen
- Welche Sprachen können wir in **linearer** Zeit erkennen?



Noch Mächtigere Maschinen: Turingmaschinen und **Berechenbarkeit**



- Turingmaschinen können nicht mehr und nicht weniger als alle anderen **hinreichend mächtigen Maschinenmodelle**
- Es gibt (wichtige) **nichtberechenbare Funktionen**. Insbesondere können wir kaum etwas mit beliebigen Turingmaschinen „machen“



Komplexitätstheorie: Was können wir **effizient** berechnen?

- Wieder sind Turingmaschinen (fast) ObdA. Wir vernachlässigen polynomiell große Faktoren in der Laufzeit (als Funktion der Eingabegröße).

$$n \approx n^2 \approx \dots n^{42} \approx \dots \ll 2^{0.134n}$$

- Wir wissen wenig über **untere Laufzeitschranken**
- Aber es gibt große Klassen von wichtigen algorithmischen Problemen, von denen **alle oder keines** effizient lösbar sind