Diskrete Mathematik

Martin Avanzini Arne Dür Christoph Kollreider Georg Moser

Fakultät für Mathematik, Informatik und Physik @ UIBK
Sommersemester 2010

Zeittafel

Woche 1	8. März, 10. März	Woche 9	17. Mai, 19. Mai
Woche 2	15. März, 17. März	Woche 10	26. Mai
Woche 3	22. März, 24. März	Woche 11	31. Mai, 2. Juni
Woche 4	12. April, 14. April	Woche 12	7. Juni, 9. Juni
Woche 5	19. April, 21. April	Woche 13	14. Juni, 16. Juni
Woche 6	26. April, 28. April	Woche 14	21. Juni, 23. Juni
Woche 7	3. Mai, 5. Mai	Woche 15	28. Juni, 30. Juni
Woche 8	10. Mai, 12. Mai		

Proseminartest

1. Klausur

Zeit und Ort

- Montag, 10:15–11:45, HS A
- Mittwoch, 14:15–15:00, HS A

Vorlesungsmaterial

Literatur & Online Material

Skriptum zu Diskrekte Mathematik 2,
3te Auflage


Universität Innsbruck Sommersemester 2010

Skriptum

Diskrete Mathematik 2

Ein Skriptum zur Vorlesung im Sommersemester 2010

Georg Moser

Sommersemester 2010

- Skript
- Folien und Hausaufgaben sind auf der LV Homepage abrufbar
- Ausgewählte Lösung sind ebenfalls online verfügbar, nachdem sie in den Übungsgruppen besprochen wurden

Prüfungs- und Übungsmodus

Prüfung

- die erste Vorlesungsprüfung findet am 28. Juni statt

Proseminar

- 50% der Aufgaben müssen angekreuzt werden
- Proseminartest (45 min) am 12. Mai zum Zeitpunkt der Vorlesung
- In der Vorlesung am 30. Juni wird die Klausur nachbesprochen
- Im Proseminar am 2. Juli werden allfällig übrige Aufgaben besprochen

Übersicht

Automaten, reguläre Sprachen und Grammatiken, (nicht)-deterministische endliche Automaten, Teilmengenkonstruktion, Automaten mit ϵ-Übergängen, Umwandlung endlicher Automaten in reguläre Ausdrücke, Minimierung

Einführung in die Berechenbarkeitstheorie, Turing Maschinen, Äquivalente Formulierungen, Entscheidungsprobleme, Universelle Maschinen und Diagonalisierung,

Einführung in die Komplexitätstheorie, Laufzeitkomplexität, die Klassen P und NP, logarithmisch platzbeschränkte Reduktionen, Speicherplatzkomplexität

Automaten und Formale Sprachen

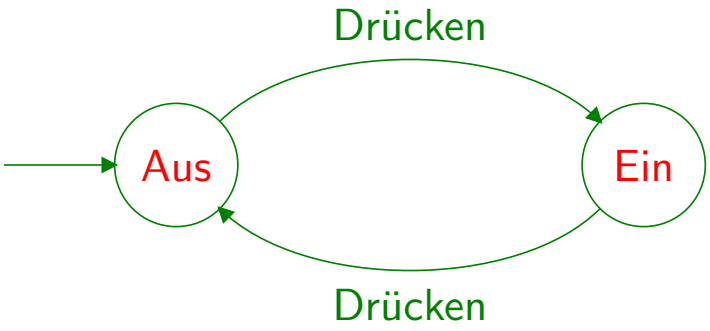
- *Die Automatentheorie ist ein Teilgebiet der Theoretischen Informatik, das sich mit dem Studium von Automaten (Modellrechnern) und mit den von diesen Automaten lösbaren Problemen beschäftigt.*
  `http://de.wikipedia.org/` 2010

- *Eine formale Sprache ist eine Menge von Wörtern, welche aus einem gegebenen Alphabet gebildet werden können (einschließlich des leeren Wortes). Sie ist definiert als eine Teilmenge der Kleeneschen Hülle über dem gegebenen Alphabet. Eine Teilmenge der formalen Sprachen kann mit Hilfe einer formalen Grammatik beschrieben werden, welche die Wörter der Sprache herzuleiten gestattet.*
  `http://de.wikipedia.org/` 2010

	Rechenmodelle	Digitalrechner	
	Turing Maschinen, Berechenbarkeit-keitstheorie, Alan Turing	—	1930er
	Endliche Automaten, Automatentheorie	`Zuse Z3`, `ENIAC`	1940er
	Grammatiken, Grundlagen des Compilerbaus, Noam Chomsky	`UNIVAC`, Transistoren statt Röhren	1950er
	P vs. NP Kom-plexitätstheorie, Stephen Cook	Minicomputer, inte-grierte Schaltkreise	1960er

Endliche Automaten

Ein-Aus Schalter

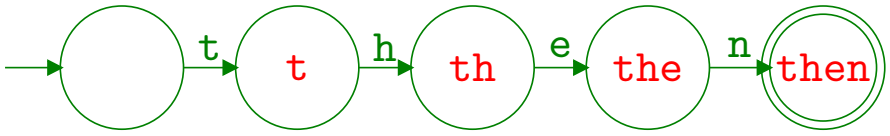


Anwendung

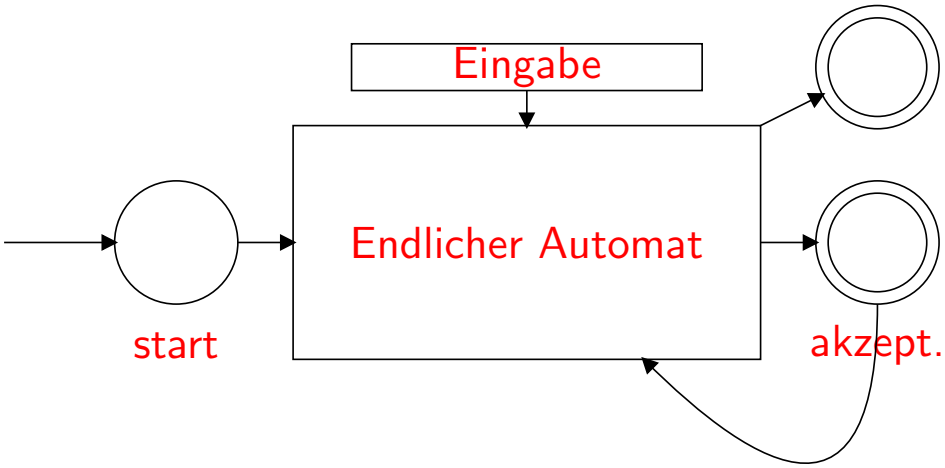
- Softwarebasiertes Entwickeln und Testen von Schaltkreisen
- Compilerbau: *Lexical analyzers*
- Textsuche; Pattern Matching
- Softwareverifikation von Protokollen
- Spielengine von Computerspiele
- ...

Beispiel: *Lexical Analyzer*

Erkenne `then`



Automaten und Sprachen



Beschreibungen von Automaten

Endliche Automaten beschreiben

1. formale Sprachen

Formale Sprachen werden durch

2. Grammatiken oder
3. reguläre Ausdrücke beschrieben

- formale Sprachen sind Mengen von Buchstabenfolgen oder Wörtern
  Wörter sind Zeichenketten über einem Alphabet, siehe DM1
- Grammatiken definieren Regeln um Wörter oder Sätze zu bilden
  - Grammatiken sind besonders geeignet zur Bearbeitung rekursiver Strukturen. Stichwort: Parsen
- Reguläre Ausdrücke beschreiben Buchstabenfolgen, also Wörter
  - Reguläre Ausdrücke werden zur Textsuche oder in Formularen verwendet

Beispiel einer "Grammatik"

$$
\begin{aligned}
S &\rightarrow \text{Pronomen Nomen Verb Adjektiv} \\
\text{Nomen} &\rightarrow \text{Lehrveranstaltungsleiter} \\
\text{Nomen} &\rightarrow \text{Vortragende} \\
\text{Pronomen} &\rightarrow \text{Unsere} \mid \text{Meine} \\
\text{Verb} &\rightarrow \text{sind} \\
\text{Adjektiv} &\rightarrow \text{lästig} \mid \text{nett} \mid \text{streng} \mid \text{zu schnell} \mid \text{anspruchsvoll}
\end{aligned}
$$

In der "Grammatik" ist der Satz

Unsere Lehrveranstaltungsleiter sind anspruchsvoll

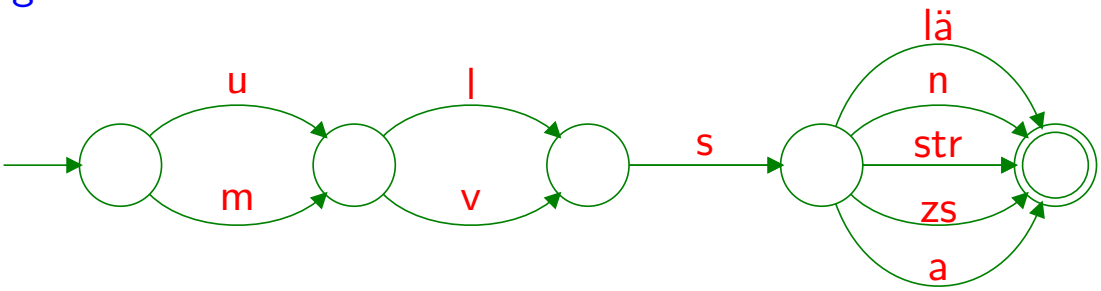
ableitbar.

Komprimierte Darstellung

$$
\begin{aligned}
S &\rightarrow PHVA \\
H &\rightarrow l \mid v \\
P &\rightarrow u \mid m \\
V &\rightarrow s \\
A &\rightarrow lä \mid n \mid str \mid zs \mid a
\end{aligned}
$$

Es gilt, dass ulsa ableitbar ist.

Darstellung als Automat



Reguläre Ausdrücke

DOS `dir␣a*.exe`
Unix `^[0-9]+\.[0-9]*(E[+-]?[0-9]+)?$`

Formale Definition

Basis

1. $\varnothing$ ist ein regulärer Ausdruck (kurz: RA)
2. ϵ ist ein RA
3. Für jedes Symbol a ist **a** ein RA

Schritt

1. Für jeden RA E ist auch E^* ein RA
2. Für reguläre Ausdrücke E und F ist auch EF ein RA
3. Für reguläre Ausdrücke E und F ist auch $E + F$ ein RA
4. Wenn E ein RA ist, dann ist auch (E) ein RA

Beispiel

$$\text{ulsa} \in \mathrm{L}((\text{u} + \text{m})(\text{l} + \text{v})(\text{s})(\text{lä} + \text{n} + \text{str} + \text{zs} + \text{a}))$$

für einen RA E, bezeichnet $\mathrm{L}(E)$ die Sprache von E

Beispiel

reguläre Ausdrücke in Unix

`.`	bezeichnet jedes Zeichen
`\s`	steht für das Sonderzeichen s
`^`, `$`	bezeichnen Zeilenanfang, bzw. Zeilenende
$[a_1 a_2 \cdots a_k]$	bezeichnet $(a_1 + a_2 \cdots + a_k)$
$[\hat{}\, a_1 a_2 \cdots a_k]$	alle Zeichen außer $a_1, a_2, \ldots, a_k$
$[x - y]$	alle ASCII Zeichen zwischen x und y. Beispiel: $[A - Z]$ beschreibt alle Großbuchstaben.
`?`	heißt "keines oder eines"
`+`	bezeichnet "eines oder mehrere"
`*`	bezeichnet Kleene Stern
$R\{n\}$	"genau n Kopien" von R

Ausdrucksstärke regulärer Sprachen

Beispiel — Programmiersprachen

```
<expression> ::= <unconditional expression>

  | if <condition> then <expression> fi

  | <expression> <expression>
```

Frage

Durch regulären Ausdruck beschreibbar?

Antwort

Nein

Deterministischer endlicher Automat

Definition — DEA

ein deterministischer endlicher Automat besteht aus

1. einer endliche Menge Q, deren Elemente Zustände heißen
2. einer endliche Menge Σ, die Eingabealphabet heißt und deren Elemente Eingabezeichen genannt werden
3. einer Abbildung
$$\delta\colon Q \times \Sigma \to Q$$
die Übergangsfunktion
4. einem ausgezeichneten Zustand q_0; der Startzustand
5. einer Teilmenge $F \subseteq Q$; die akzeptierenden Zustände

die kompakteste Repräsentation eines DEA ist das Quintupel:

$$A = (Q, \Sigma, \delta, q_0, F)$$

Alternative Darstellungen

Zustandstabelle

momentaner Zustand	Eingabe $a \in \Sigma$
$\vdots$ $Q \ni p$ $\vdots$	$\delta(p, a)$

Zustandsgraph

1. Die Ecken sind die Zustände.
2. Für Zustände p, $q \in Q$ werden die Übergänge durch markierte Kanten dargestellt:

$$(p, a, q) \quad \text{mit} \quad a \in \Sigma \quad \text{und} \quad \delta(p, a) = q$$

3. Startzustand mit Pfeil markiert; Endzustand doppelt eingekreist

Erweiterte Übergangsfunktion

Sei δ eine Übergangsfunktion, dann definiere $\widehat{\delta}\colon Q \times \Sigma^* \to Q$ induktiv (über Strings).

Definition $\widehat{\delta}$

1. Basis:

$$\widehat{\delta}(p, \epsilon) := p$$

2. Schritt:

$$\widehat{\delta}(p, xa) := \delta(\widehat{\delta}(p, x), a)$$

Definition $L(A)$

sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DEA; die Sprache $L(A)$ von A:

$$L(A) := \{x \mid \widehat{\delta}(q_0, x) \in F\}$$

Beispiel (1)

definiere DEA A, der alle aus 0en und 1en bestehenden Zeichenketten akzeptiert, die die Folge 01 enthalten

$$L = \{x01y \mid x, y \text{ sind beliebige Zeichenketten aus 0en und 1en}\}$$

L enthält etwa die Strings

$$01 \quad 11010 \quad 100011$$

q_0	A hat Sequenz 01 noch nicht gefunden A wechselt in Zustand q_1, sobald 0 gelesen wird sonst verharrt A in Zustand q_0
q_1	A hat Sequenz 0 gelesen A wechselt in Zustand q_2, sobald 1 gelesen wird sonst verharrt A in Zustand q_1
q_2	A hat Sequenz 01 gelesen A akzeptiert jede weitere Eingabe

Beispiel (2)

definiere Übergangsfunktion δ

$$\delta(q_0, 1) = q_0$$

dies entspricht dem Verharren in q_0, setze

$$\delta(q_0, 0) = q_1$$

das heißt A geht bei 0 in den Zustand q_1 und

$$\delta(q_1, 0) = q_1 \qquad \delta(q_1, 1) = q_2$$

mit

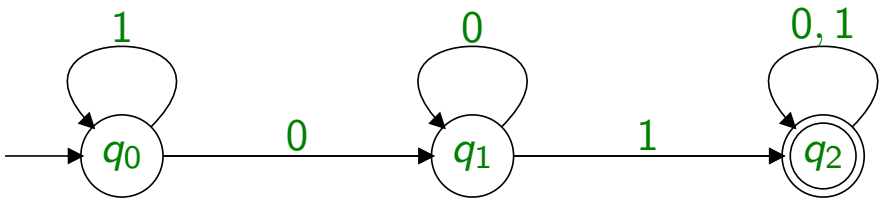
$$\delta(q_2, 0) = q_2 \qquad \delta(q_2, 1) = q_2$$

wir erhalten

$$A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$$

Beispiel (3)

Automat A kann durch seinen Zustandsgraphen dargestellt werden:



Automat A kann durch seinen Zustandstafel dargestellt werden:

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
$*q_2$	q_2	q_2