

Diskrete Mathematik

Arne Dür Kurt Girstmair Simon Legner
 Georg Moser Harald Zankl

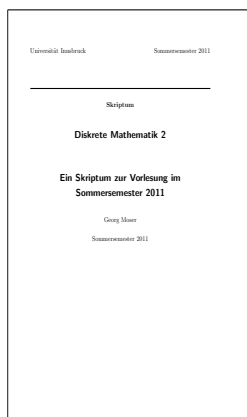
Fakultät für Mathematik, Informatik und Physik © UIBK
 Sommersemester 2011



Vorlesungsmaterial

Literatur & Online Material

Skriptum zu Diskrete Mathematik 2,
 4te Auflage



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

Zeitplan

Woche 1	7. März, 9. März	Woche 9	16. Mai, 18. Mai
Woche 2	14. März, 16. März	Woche 10	23. Mai, 25. Mai
Woche 3	21. März, 23. März	Woche 11	30. Mai, 1. Juni
Woche 4	28. März, 30. März	Woche 12	6. Juni, 8. Juni
Woche 5	4. April, 6. April	Woche 13	15. Juni
Woche 6	11. April, 13. April	Woche 14	20. Juni, 22. Juni
Woche 7	2. Mai , 4. Mai	Woche 15	27. Juni, 29. Juni
Woche 8	9. Mai, 11. Mai		1. Klausur

Proseminartest

Zeit und Ort

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

Prüfungs- und Proseminarmodus

Prüfung

- die erste Vorlesungsprüfung findet am **27. Juni** statt

Proseminar

- 50% der Aufgaben müssen angekreuzt werden
- Proseminartest (45 min) am **11. Mai** zum Zeitpunkt der Vorlesung
- In der Vorlesung am 29. Juni wird die Klausur nachbesprochen
- Im Proseminar am 1. Juli werden allfällig übrige Aufgaben besprochen

Übersicht

Endliche Automaten

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

Berechenbarkeitstheorie

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

Komplexitätstheorie

Einführung in die Komplexitätstheorie, 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/> 2011

- Formale Sprachen eignen sich zur **mathematisch** präzisen Beschreibung im Umgang mit Zeichenketten. So können zum Beispiel Datenformate oder ganze Programmiersprachen spezifiziert werden. Zusammen mit einer formalen Semantik erhalten die definierten Zeichenketten eine **mathematische** Bedeutung. Bei einer Programmiersprache kann damit einer Programmieranweisung (als Teil der formalen Sprache) ein eindeutiges Maschinenverhalten (als Teil der Semantik) zugeordnet werden.

<http://de.wikipedia.org/> 2011

Rechenmodelle

Digitalrechner



Turing Maschinen, **Berechenbarkeitstheorie**, Alan Turing

—

1930er

Endliche Automaten, **Automatentheorie**

Zuse Z3, ENIAC

1940er



Grammatiken, **Grundlagen des Compilerbaus**, Noam Chomsky

UNIVAC, Transistoren statt Röhren

1950er



P vs. NP **Komplexitätstheorie**, Stephen Cook

Minicomputer, integrierte Schaltkreise

1960er

Entschlüsselung der ENIGMA

- “Enigma” ist griechisch für Rätsel
- deutsche Kodiermaschine eingesetzt im 2. Weltkrieg
- galt als unentzifferbar, Entschlüsselung benötigte etwa 8 Jahre

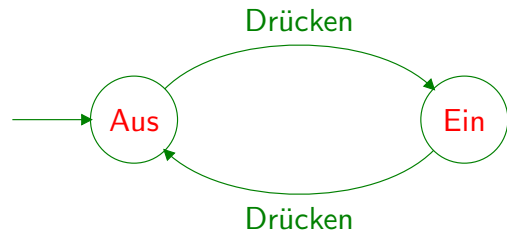


- manuelle Entschlüsselung erwies sich als unmöglich
- Code wurde maschinell entschlüsselt
- wesentliche Werkzeuge: **mathematische Analyse** und **Automatisierung**



Endliche Automaten

Beispiel



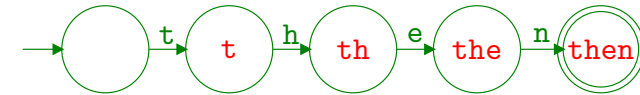
Anwendung

- Softwarebasiertes Entwickeln und Testen von **Schaltkreisen**
- Compilerbau: **lexikalische Analyse**
- Textsuche; **Pattern Matching**
- Softwareverifikation von **Protokollen**
- **Spielengine** in Computerspielen
- ...

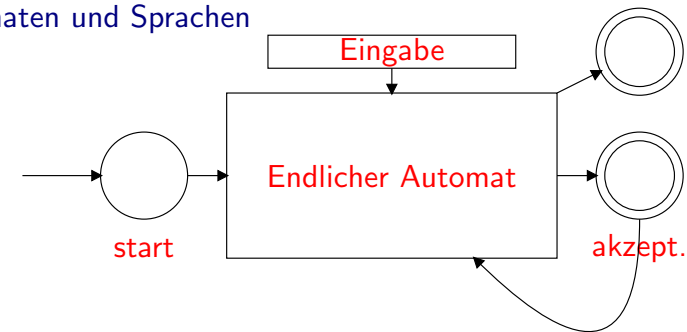
Lexikalische Analyse

Beispiel

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**, in der **Kommandozeile** oder in **Formularen** verwendet

Beispiel einer "Grammatik"

- S** → Pronomen Nomen Verb Adjektiv
- Nomen** → Lehrveranstaltungsleiter
- Nomen** → Vortragende
- Pronomen** → Unsere | Meine
- Verb** → sind
- Adjektiv** → lästig | nett | streng | zu schnell | anspruchsvoll

in der "Grammatik" ist der Satz

Unsere Lehrveranstaltungsleiter sind anspruchsvoll

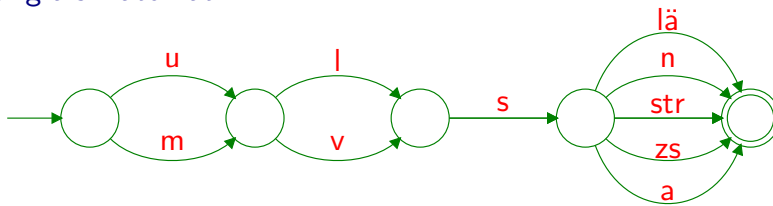
ableitbar

Komprimierte Darstellung

$S \rightarrow PNVA$
 $N \rightarrow l \mid v$
 $P \rightarrow u \mid m$
 $V \rightarrow s$
 $A \rightarrow lä \mid n \mid str \mid zs \mid a$

Es gilt, dass **ulsa** ableitbar ist.

Darstellung als Automat



Reguläre Ausdrücke

DOS `dir_a*.exe`

Unix `^[0-9]+\.[0-9]*(E[+-]?[0-9]+)?$`

Definition

Basis

- 1 \emptyset ist ein regulärer Ausdruck (kurz: RA)
- 2 ϵ ist ein RA
- 3 Für jedes Symbol a ist a ein RA

Sprache von \emptyset

$$L(\emptyset) := \emptyset$$

$$L(\epsilon) := \{\epsilon\}$$

$$L(a) := \{a\}$$

Schritt

- 1 Für jeden RA E ist E^* ein RA
- 2 Für RAs E und F ist EF ein RA
- 3 Für RAs E und F ist $E + F$ ein RA
- 4 Wenn E ein RA ist, dann ist (E) ein RA

$$L(E^*) := L(E)^*$$

$$L(EF) := L(E) \cdot L(F)$$

$$L(E + F) := L(E) \cup L(F)$$

$$L((E)) := L(E)$$

Definition

für einen RA E , bezeichnet $L(E)$ die Sprache von E

Beispiel

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

Reguläre Ausdrücke in Unix (Auswahl)

\cdot bezeichnet jedes Zeichen
 $\wedge, \$$ bezeichnen Zeilenanfang, bzw. Zeilenende
 $|$ bezeichnet "oder" (also +)
 $[a_1 a_2 \cdots a_k]$ bezeichnet $(a_1 + a_2 + \cdots + a_k)$
 $[\wedge a_1 a_2 \cdots a_k]$ alle Zeichen **außer** a_1, a_2, \dots, a_k
 $[x - y]$ alle ASCII Zeichen zwischen x und y
 $?$ heißt "keines oder eines"
 $+$ bezeichnet "eines oder mehrere"
 $*$ bezeichnet Kleene Stern
 $R\{n\}$ "genau n Kopien" von R

Erinnerung: 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, in der Kommandozeile oder in Formularen verwendet

Ausdrucksstärke regulärer Ausdrücke

Beispiel

betrachte Sprache L , die aus allen Strings von 1ern besteht, deren Anzahl eine Primzahl ist

Beispiel

```
<expression> ::= <condition>
  | if <condition> then <expression> fi
  | <expression> <expression>
```

Frage

Durch regulären Ausdrücke beschreibbar?

Antwort

Nein

Deterministischer endlicher Automat

Definition

ein **deterministischer endlicher Automat** (DEA) 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

$$\delta: Q \times \Sigma \rightarrow Q$$

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:

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

Zustandstabelle

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

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 $\hat{\delta}: Q \times \Sigma^* \rightarrow Q$ induktiv (über Strings).

Definition

- 1 **Basis:**

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

- 2 **Schritt:**

$$\hat{\delta}(p, xa) := \delta(\hat{\delta}(p, x), a) \quad (x \in \Sigma^* \ a \in \Sigma)$$

Definition

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

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

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 11010 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

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

mit

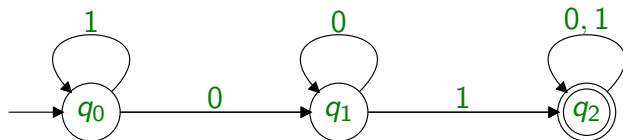
$$\delta(q_2, 0) = q_2 \quad \delta(q_2, 1) = q_2$$

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