Automatentheorie und ihre Anwendungen
Einführung

Wintersemester 2017/18 Thomas Schneider

AG Theorie der künstlichen Intelligenz (TdKI)

http://tinyurl.com/ws1718-automaten

Einführung

1 Organisatorisches

2 Vorlesungsüberblick

Organisatorisches

Zeit und Ort

Mo. 10–12 MZH 1110
Mi. 8–10 MZH 1460

Vortragender

Thomas Schneider
Cartesium, Raum 2.56
Tel. (218) 64432
ts[ÄT]cs.uni-bremen.de

Position im Curriculum

Informatik: Master-Ergänzung,
Modul „Spezielle Themen der Theoretischen Informatik“

Mathematik: Ergänzungsfach

Organisatorisches

Form

K4 (in der Regel 3V, 1Ü)

Fragen und Diskussion **jederzeit erwünscht**.

Voraussetzungen

Grundkenntnisse aus Theoret. Informatik 1+2 hilfreich,
aber nicht zwingend erforderlich

Vorlesungsmaterial:

- Folien und Aufgabenblätter: `tinyurl.com/ws1718-automaten`
- Folien werden online gestellt, enthalten aber nicht alle Details. (Beweise, Beispiele etc. von der Tafel bitte mitschreiben.)
- Skript (englisch) für den Theorie-Teil der Vorlesung in Stud.IP
- Literatur: wird bei jedem Kapitel bekannt gegeben

Prüfungsmodalitäten

Übungsaufgaben & Fachgespräch:

- Übungsaufgaben ca. jede zweite Woche;
  voraussichtlich 6 Blätter, mit Zusatzaufgaben
- Werden in Gruppen (2–3 Personen) bearbeitet, abgegeben und korrigiert – jede_r muss mindestens einmal vorrechnen
- Aus der erreichten Gesamtpunktzahl aller Blätter ergibt sich die vorläufige Note für diesen Kurs
- Fachgespräche am Ende des Semesters
  (Prüfungsleistung, Änderung der Note möglich)
  Voraussetzung: insgesamt 50 % der Punkte in Übungsaufgaben

oder

Mündliche Prüfung

Wiederholungsregelungen auf der nächsten Folie ...

Prüfungsmodalitäten

Wiederholungsregelungen

- Fachgespräch nicht bestanden?

  $\rightsquigarrow$ 1 Wiederholungsversuch im selben Semester möglich

- Weitere Wiederholungsversuche (wenn nötig):

  mündliche Prüfung in den folgenden 4 Semestern
  (je 1 Versuch pro Semester)

Übungsbetrieb

Terminübersicht (geplant)

Blatt	Erscheinen (geplant)	Abgabe	Besprechung, Übungstermin
1	Mi. 18. 10.	So. 29. 10.	Mi. 1. 11.
2	Mo. 30. 10.	So. 12. 11.	Mi. 15. 11.
3	Sa. 18. 11.	**Fr.** 1. 12.	**Mo.** 4. 12.
4	Sa. 2. 12.	Fr. 15. 12.	Mo. 18. 12.
5	Sa. 16. 12.	Fr. 12. 1.	Mo. 15. 1.
6	Sa. 13. 1.	Fr. 26. 1.	Mo. 29. 1.

Einführung

1 Organisatorisches

2 Vorlesungsüberblick

Ursprünge der Automatentheorie

Automaten als Berechnungsmodelle, zur Definition formaler Sprachen

(3) (Nicht-)deterministische endliche Automaten (NEA/DEA)
[McCulloch & Pitts 1943; Kleene 1956]

(2) Kellerautomaten (pushdown automata, PDA)
[Newell, Shaw, Simon 1959]

(1) Linear beschränkte Automaten (LBA)
[Myhill 1960; Kuroda 1964]

(0) Turingmaschinen (TM)
[Turing 1936]

Ursprünge der Automatentheorie

Varianten endlicher Automaten

zum Lösen von Entscheidungsproblemen

- **Baumautomaten**
  = endliche Automaten auf Bäumen (statt auf Wörtern)

  ursprünglich für Schaltkreisverifikation
  [Church, 50er/60er]
- **Büchi-Automaten**
  = endliche Automaten auf unendlichen Wörtern

  ursprünglich zum Entscheiden logischer Theorien
  [Büchi 1962]
- **alternierende Automaten**
  (Alternierung = Verallgemeinerung des Nichtdeterminismus)

  [Chandra, Kozen, Stockmeyer 1981]
- und viele weitere . . .

Moderne Anwendungen der Automatentheorie

Automaten werden in der Informatik angewendet z. B. für

- Validierung semistrukturierter Daten (XML)
- Verifikation von Hard- und Software
- Komplexitätstheorie (Definition Komplexitätsklassen)
- Entscheidungsverfahren
  z. B. für Logiken (aus der KI, Verifikation und mehr)
- etc.

Es besteht eine enge Verbindung zwischen Automaten und Logik.

Automaten haben die Entwicklung der Informatik entscheidend mitbestimmt.

Fallbeispiel 1: XML

XML-Schema und Validierung von XML-Dokumenten
können als Automatenprobleme verstanden werden:

- XML-Dokument $\approx$ **Baum**
- XML-Schema beschreibt Menge der gültigen XML-Dokumente $\approx$ **formale Sprache** (Menge von Bäumen, i. d. R. unendlich)
- Formale Sprache kann man durch **endlichen Baumautomaten** beschreiben.

Dann entspricht . . .

- Validität eines XML-Dokuments $\hat{=}$ **Wortproblem**
- Konsistenz des XML-Schemas $\hat{=}$ **Leerheitsproblem**
- . . .

Fallbeispiel 2: Verifikation

Verifikation: nachweisen, dass ein Chip/Programm eine gewünschte Spezifikation erfüllt (z. B. keine Division durch 0, keine Deadlocks)

Manche Systeme sollen ∞ lange laufen (keine Terminierung): Betriebssysteme, Bankautomaten, Flugsicherungssysteme

Wichtige Technik: **Model checking** – oft automatenbasiert:

- Lauf des Systems = **unendliches Wort**
- System = **formale Sprache** L_1
  (Menge aller Läufe, i. d. R. unendlich)
- erlaubtes Verhalten = **formale Sprache** L_2
  (Menge aller erlaubten Läufe, i. d. R. unendlich)
- Beschreiben L_1 und L_2 durch **Büchi-Automaten**
  (endliche Automaten auf unendlichen Wörtern)

$\rightsquigarrow$ Model checking $\hat{=}$ „$L_1 \subseteq L_2$?“ $\approx$ **Äquivalenzproblem**

Ziele der Vorlesung

Einführung in grundlegende Automatenbegriffe

- auf endlichen Bäumen
- auf unendlichen Wörtern
- auf unendlichen Bäumen

Untersuchung der zugehörigen Sprachklassen

- Abschlusseigenschaften, Determinisierung, Charakterisierungen, Entscheidungsprobleme
- teils einfach, teils anspruchsvoll
- interessante Techniken: Safra-Konstruktion, Paritätsspiele

Herstellung von Bezügen zu Anwendungen

Einsatz dieser Automaten z. B. in XML-Validierung und Verifikation

Übersicht Vorlesung

Einführung ✓

Teil 1: Endliche Automaten auf endlichen Wörtern
(Kurzwiederholung und Anwendungen, ca. 2 Sitzungen)

Teil 2: Endliche Automaten auf endlichen Bäumen

Teil 3: Endliche Automaten auf unendlichen Wörtern

Teil 4: Endliche Automaten auf unendlichen Bäumen