Theoretische Informatik 1

Sabine Kuske
Linzer Str. 9a, OAS 3005
Tel.: 2335, 8794
kuske@informatik.uni-bremen.de
www.informatik.uni-bremen.de/theorie

24. Oktober 2007

Termine

- Vorlesung: 2 SWS
  - Mo von 10:00 - 12:00 NW1 H1 (H0020)
- Tutorien: 2 SWS (ab nächster Woche)
  - Mo von 08:00 - 10:00 MZH 7250,
  - Mo von 08:00 - 10:00 MZH 4194,
  - Di von 10:00 - 12:00 MZH 7050,
  - Di von 13:00 - 15:00 MZH 7250,
  - Do von 08:00 - 10:00 MZH 7230,
  - Do von 10:00 - 12:00 MZH 7210,
  - Mi von 10:00 - 12:00 MZH 7250

TutorInnen

- Sven Bertel (bertel@informatik.uni-bremen.de)
- Laura Bieker (lbieker@informatik.uni-bremen.de)
- Frank Dylla (dylla@informatik.uni-bremen.de)
- Marcus Ermler (maermler@informatik.uni-bremen.de)
- Friederike Jolk (rikej@informatik.uni-bremen.de)
- Sabine Kuske (kuske@informatik.uni-bremen.de)

Scheinkriterien

Es gibt zwei Möglichkeiten, den Leistungsnachweis zu erwerben:

1. Regelmäßige Bearbeitung von Übungsaufgaben und Fachgespräch
   - Übungsblätter werden in Gruppen bearbeitet; die Gruppengröße soll 3 nicht überschreiten.
   - Jedes Blatt muss mindestens zu 50% richtig bearbeitet werden. Ein Blatt darf nachgebessert werden.
   - Das Fachgespräch dauert ca. 10 Minuten pro Person und dient der Überprüfung der individuellen Leistungsfähigkeit. Es findet i.d.R. gegen Ende der Vorlesungszeit statt.

2. Mündliche Prüfung

- Eine mündliche Prüfung dauert 20-30 Minuten.

Weitere Infos unter

http://studienzentrum.informatik.uni-bremen.de/

Theoretische Informatik ist...

wichtig

und

interessant,

denn sie beantwortet Fragen, wie z.B.

Macht es Sinn, für ein gegebenes Problem eine Lösung zu entwickeln?

Theoretische Informatik ist...

wichtig

und

interessant,

denn sie beantwortet Fragen, wie z.B.

Welches Modellierungswerkzeug ist für welchen Zweck geeignet?

Theoretische Informatik ist...

wichtig

und

interessant,

denn sie beantwortet Fragen, wie z.B.

Wie findet man eine möglichst fehlerfreie Lösung?

Theoretische Informatik ist...

wichtig

und

interessant,

denn sie beantwortet Fragen, wie z.B.

Wie lange muss man höchstens oder mindestens auf ein Ergebnis warten?

Theoretische Informatik ist...

wichtig

und

interessant,

denn sie beantwortet Fragen, wie z.B.

Für welche Probleme existieren bisher nur viel zu langsame Lösungen?

Lernziele

- Grundlagen der Theoretischen Informatik
- Abstraktes Denken
- Formalisierung von Sachverhalten
- Beweisen

Themen dieses Kurses

- Automatentheorie
- Formale Sprachen
- Berechenbarkeit

Automatentheorie

- Was sind Automaten?
- Was haben Automaten mit Informatik zu tun?
- Welche Automaten gibt es?
- Wofür kann man Automaten (nicht) einsetzen?
- Wie entwirft man korrekt funktionierende Automaten?
- Wie "schnell" sind Automaten?

Formale Sprachen

- Was sind Formale Sprachen?
- Was haben Formale Sprachen mit Informatik zu tun?
- Welche Beschreibungsmittel gibt es für Formale Sprachen?
- Was kann man mit Formalen Sprachen ausdrücken?
- Wie hängen Automaten und Formale Sprachen zusammen?

Berechenbarkeit

- Was ist berechenbar (und was nicht)?
- Was hat Berechenbarkeit mit Computern zu tun?
- Wie kann man Berechenbares beschreiben?
- Wie zeigt man, dass etwas (nicht) berechenbar ist?

Literatur

1. Skript (`www.informatik.uni-bremen.de/theorie/teach/thi1/`; dieser Kurs basiert auf dem Skript.)
2. John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie. Addison-Wesley, 2002. (Das Buch gibt es auch auf Englisch)
3. A.J. Kfoury, Robert N. Moll, and Michael A. Arbib. A Programming Approach to Computability. Springer, 1982. (Berechenbarkeit)

Literatur

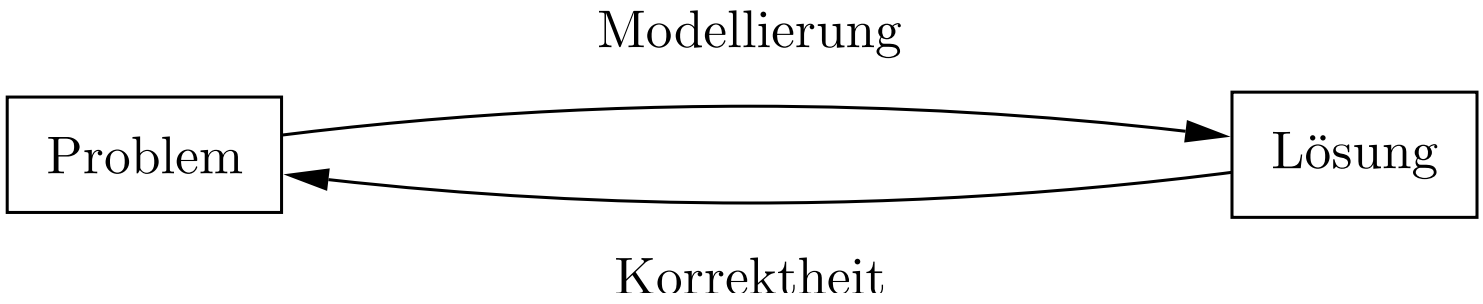
4. Uwe Schöning. Theoretische Informatik – kurz gefasst, 4. Auflage. Spektrum Akademischer Verlag, 2003.
5. Gottfried Vossen and Kurt-Ulrich Witt. Grundlagen der Theoretischen Informatik mit Anwendungen. Vieweg, 2000.

Weiteres Material:

- Folien zur Vorlesung (`www.informatik.uni-bremen.de/theorie/teach/thi1/`)
- Literaturhinweise im Skript

Informatik beinhaltet...

die Modellierung von Problemen und deren (korrekte) Lösungen.



Modellierung einer Heizung

Modelliere eine Heizung, die

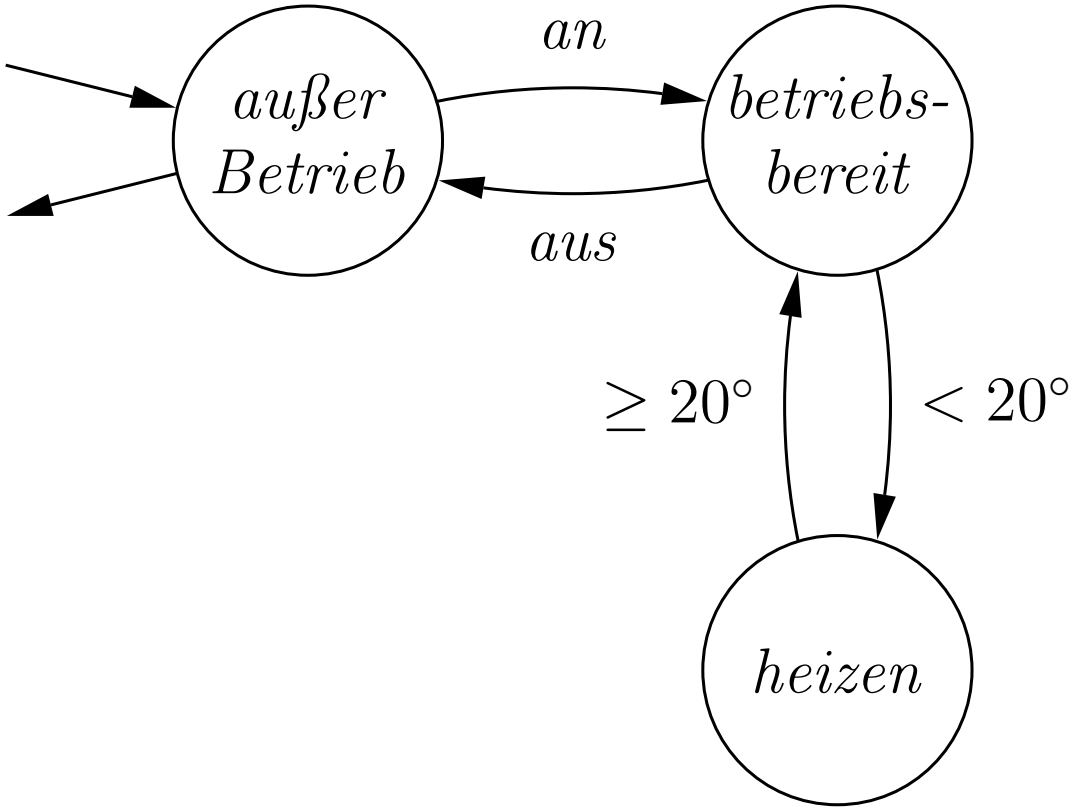
(1) angeschaltet werden kann und dann betriebsbereit ist,

(2) bei einer Temperatur unter 20° heizt,

(3) wieder betriebsbereit wird, wenn die Temperatur mindestens 20° erreicht hat, und

(4) ausgeschaltet werden kann, wenn sie nicht gerade heizt.

Zustandsgraph Heating



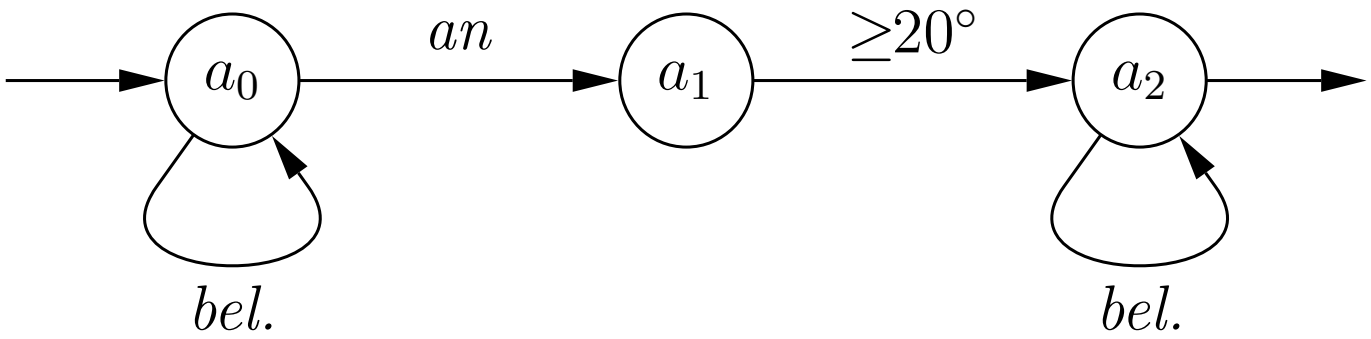
Abläufe (Ereignisfolgen)

- $an\ aus\ an\ aus\ldots$
- $an < 20^\circ \geq 20^\circ < 20^\circ \geq 20^\circ\ aus$
- usw.

$L(heating)$: Menge aller möglichen Abläufe

Verbotene Teilfolgen

(a) $an \geq 20^\circ$

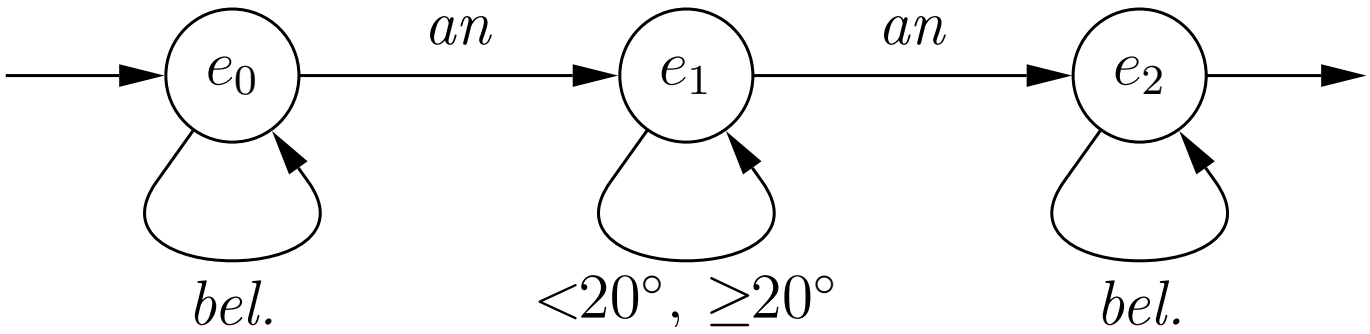


(b) $< 20^\circ$ aus (Zustandsgraph analog)

(c) $< 20^\circ$ $< 20^\circ$ (Zustandsgraph analog)

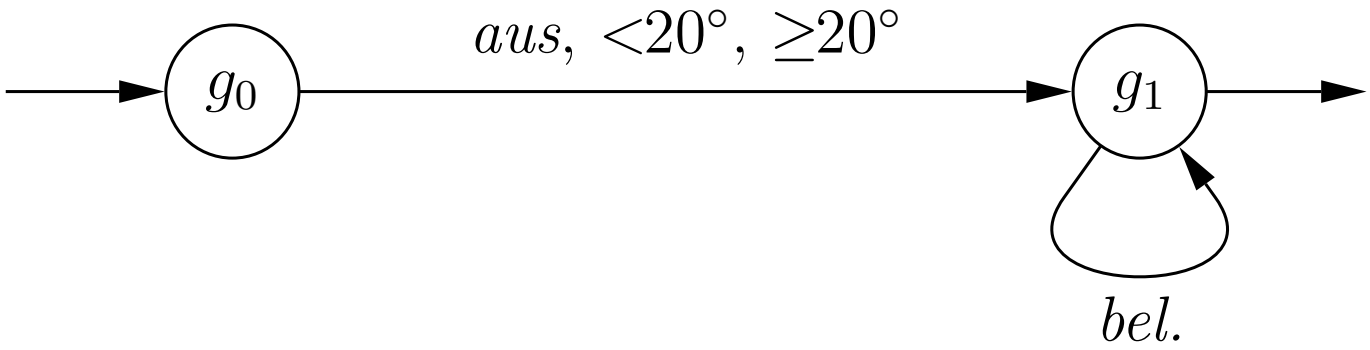
(d) $\geq 20^\circ$ $\geq 20^\circ$ (Zustandsgraph analog)

(e) $an\ u\ an$ (u: Ablauf, der nur $<20°$ und $\geq 20°$ enthält.)



(f) $aus\ u\ aus$ (Zustandsgraph analog)

(g) Abläufe, die nicht mit an beginnen



(h) Abläufe, die nicht mit aus enden (Zustandsgraph analog)

$L_{forbidden}$: Alle verbotenen Abläufe

Korrektheit

Heating ist korrekt bezüglich $L_{forbidden}$, falls

$$L(heating) \cap L_{forbidden} = \emptyset.$$

Wörter

Wörter sind Zeichenketten

- Programmiersprachen:

  Namen, Ausdrücke, Datentypen, Programme,. . .

- Systemmodellierung:

  mögliche Abläufe, Ereignisfolgen, verbotene Folgen,. . .

- Natürliche Sprachen:

  Wörter, Sätze, Texte, . . .

- Textsysteme:

  Schreiben und Lesen, Verändern und analysieren: Cut & Paste, Zeichen zählen, Vergleichen, Zusammensetzen, . . .

- Eine der wichtigsten Datenstrukturen

Erzeugung von Wörtern

- A : Alphabet (Menge von Zeichen)
- A^*: Menge aller Wörter über A.

Rekursive Definition von A^*

- $\lambda \in A^*$ (leeres Wort, enthält keine Zeichen)
- mit $v \in A^*$ und $x \in A$ ist auch $xv \in A^*$ (Linksaddition)

Schreibweise: $x\lambda$ wird mit x abgekürzt.

Länge

1. $length(\lambda) = 0$
2. $length(xv) = length(v) + 1$ für $x \in A, v \in A^*$

Beispiel

$length(aba) = 1 + length(ba) = 1 + 1 + length(a) = 2 + 1 + length(\lambda) = 3 + 0 = 3$

Induktionsprinzip

- Induktionsanfang (IA):
  Zeige THEOREM für $v = \lambda$.
- Induktionsvoraussetzung (IV):
  Nimm THEOREM für v an.
- Induktionsschluss (IS):
  Zeige THEOREM für xv mit $x \in A$.