Grundlagen der Informatik

- Einführung in Berechenbarkeit und Komplexität -

Prof. Dr. Klaus Volbert

Hochschule für angewandte Wissenschaften
Fakultät Informatik und Mathematik

Wintersemester 2010/11
Regensburg, 14. Dezember 2010

Vorläufiger Prüfungstermin

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Prüfung
  - 90 min. Klausur am Ende des Semesters
  - Keine Hilfsmittel (kein Taschenrechner/Laptop/Mobiltelefon o.ä.)
  - Gesamter Stoff der Vorlesung ist relevant
  - Voraussetzung zur Teilnahme:
    - Regelmäßige und aktive Teilnahme an den Übungen
    - Erfolgreiche Bearbeitung der Übungsblätter (min. 50 % der Punkte)
- Vorläufiger Termin
  - Dienstag, 01.02.2011:

    08:15 - 09:45 Uhr (90 Minuten)

  - Ort:

    Räume U212, U213 und U311

Berechenbarkeit und Komplexität

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Teilgebiete der Theoretischen Informatik
- Berechenbarkeitstheorie
  - Welche Aufgaben kann ein Computer lösen, welche nicht?
  - Welche Funktionen sind berechenbar, welche nicht?
  - Was ist formal ein Algorithmus?
  - Church'sche These (1936, auch Church-Turing-These):

    Die Klasse der Turing-berechenbaren Funktionen ist genau die Klasse der intuitiv berechenbaren Funktionen
- Komplexitätstheorie
  - Wieviel Rechenzeit und Speicherplatz ist zur Berechnung einer Funktion notwendig? (Zeit: Anzahl Schritte, Platz: Anzahl Zellen)
  - Klassifizierung von lösbaren Problemen nach Zeit-/Platzkomplexität
  - Einführung zweier wesentlicher Komplexitätsklassen:
    - Klasse P : Praktisch lösbar (polynomiell deterministisch)
    - Klasse NP : Praktisch undurchführbar (polynomiell nicht-deterministisch)

Intuitiv Berechenbar

- Beim Programmieren entwickelt sich ein „Gefühl“ für Funktionen (auf den natürlichen Zahlen), die berechenbar sind
- Eine (partielle) Funktion $f: \mathrm{IN}^k \to \mathrm{IN}$ heißt berechenbar, wenn es einen Algorithmus gibt, der f berechnet
  - Bei Eingabe von $x \in \mathrm{IN}^k$ liefert der Algorithmus (das Rechenverfahren, die Verarbeitung) nach endlich vielen Schritten die Ausgabe $f(x) \in \mathrm{IN}$
  - Falls f partiell ist, so muss der Algorithmus bei Eingabe des entsprechenden Wertes nicht terminieren (unendliche Schleife)
  - Der Algorithmus kann z.B. als C/C++-, Java-, C#-, Assembler- oder Registermaschinen-Programm oder … eingegeben werden
- Beispiele

$$f(n) = \begin{cases} 1, \text{falls } n \text{ ein Anfangsabschnitt der Darstellung von } \pi \text{ ist} \\ 0, \text{sonst} \end{cases}$$

$$g(n) = \begin{cases} 1, \text{falls } n \text{ irgendwo in der Darstellung von } \pi \text{ vorkommt} \\ 0, \text{sonst} \end{cases}$$

Berechenbare Funktion

- Eingabe: $n \in \mathrm{IN}$

Algorithmus 1

```
i = 1; s = 0;
Solange i ≤ n:
   s = s + i;
   i = i + 1;
Gib s aus
```

Algorithmus 2

```
s = n;
s = s * (s + 1);
s = s / 2;
Gib s aus
```

Algorithmus 3

```
i = 1; s = 0;
Solange i ≤ (n/2):
   s = s + (n+1);
   i = i + 1;
Falls n ungerade
   s = s + (n+1)/2;
Gib s aus
```

- Ausgabe: Summe der ersten n Zahlen:

$$s = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

- Wie viele verschiedene Algorithmen gibt es? ∞-viele!

Goldbach-Vermutung (~ 1742)

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Jede gerade Zahl größer gleich 4 lässt sich als Summe zweier Primzahlen darstellen?! (ungelöstes Problem der Mathematik)
- Darstellung aus Sicht der Informatik:

Gibt es einen Algorithmus, der bzgl. einer beliebigen geraden natürlichen Zahl entscheidet, ob sie sich als Summe zweier Primzahlen darstellen lässt (d.h. die Goldbach-Eigenschaft besitzt)?

Ja!

- Eingabe: Gerade, natürliche Zahl
- Ausgabe: Ja oder Nein (Goldbach-Eigenschaft)
- Idee des Algorithmus:
  - Prim(*x*) liefert für eine Zahl *x*, ob sie eine Primzahl ist
  - Bei Eingabe der Zahl werden alle möglichen Summanden getestet

Existenz nicht-berechenbarer Funktionen

- Wie viele berechenbare Funktionen gibt es?
  - Algorithmus ist ein Wort über $\sum^*$ mit $\sum = \{0,1\}$
  - D.h. es gibt abzählbar viele Algorithmen
  - D.h. es gibt abzählbar viele berechenbare Funktionen
- Wie viele Funktionen $f: \mathrm{IN}^k \to \mathrm{IN}$ gibt es?
  - Anzahl $f: \mathrm{IN}^k \to \mathrm{IN}$ ist größer als Anzahl $f: \mathrm{IN} \to \mathrm{IN}$
  - Annahme: Es gibt abzählbar viele Funktionen $f: \mathrm{IN} \to \mathrm{IN}$

Eingaben

Funktionen	1	2	3	4	5	...
f_1	2	4	6	8	10	
f_2	1	2	3	4	5	
f_3	3	6	9	12	15	
f_4	1	2	3	0	1	
...	...					

Definiere $f_x: \mathrm{IN} \to \mathrm{IN}$ mit

$$f_x(i) = 0, \text{ wenn } f_i(i) > 0$$
$$f_x(i) = 1, \text{ sonst}$$

Dann gilt: f_x ist eine neu konstruierte Funktion, die bisher nicht enthalten war.

$\Rightarrow$ Es gibt nicht-berechenbare Funktionen

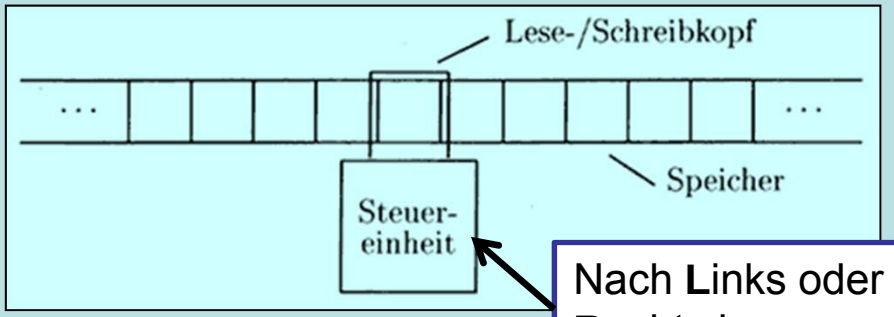
Turingmaschine (Alan M. Turing, 1936)

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Eine deterministische Turingmaschine ist beschrieben durch ein 7-Tupel $\mathrm{TM} = (Q, \sum, \Gamma, \delta, q_0, \#, F)$ mit:
  - Q ist eine endliche, nichtleere Menge von Zuständen
  - $\sum \subseteq \Gamma$ ist ein endliches, nicht leeres Eingabealphabet
  - Γ ist ein endliches, nicht leeres Bandalphabet
  - $\delta: Q \backslash \mathrm{F} \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$ ist die Übergangsfunktion
  - $q_0 \in Q$ ist der Startzustand
  - $\# \in \Gamma \backslash \sum$ ist das Blank-Symbol

    (leeres Feld, Initialwert)

  - $F \subseteq Q$ ist die Menge der akzeptierenden Endzustände



- Akzeptanz
  - Eine TM akzeptiert eine Eingabe $x_1, \ldots, x_n$, wenn gilt:

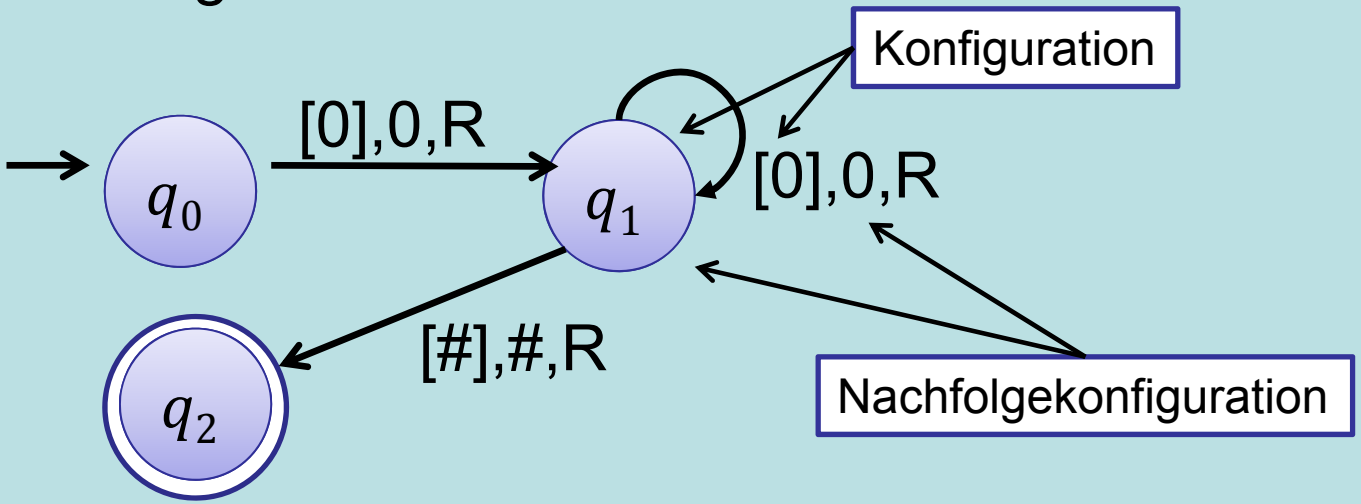
$$q_0 x_1, \ldots, x_n \overset{*}{\rightarrow} \alpha q \beta \text{ mit } q \in F \text{ und } \alpha, \beta \in \Gamma^*$$

Beispiel (Turingmaschine verstehen)

- $\mathrm{TM} = (Q, \Sigma, \Gamma, \delta, q_0, \#, F)$ mit
  - $Q = \{q_0, q_1, q_2\}$
  - $\Sigma = \{0,1\},\ \Gamma = \{\#, 0,1\}$
  - $F = \{q_2\}$

δ	q_0	q_1	q_2
#	–	$(q_2, \#, R)$	-
0	$(q_1, 0, R)$	$(q_1, 0, R)$	-
1	-	-	-

- Grafische Darstellung:



- Die TM hält, wenn keine Kante vorhanden ist ("-" in Tabelle)
  (Beachte: DFA musste vollständig sein!)
- Welche Sprache akzeptiert die TM? $L = \{0^n \mid n \geq 1\}$

Beispiel (Turingmaschine entwerfen)

$$L = \{ 1^n 0^n \mid n \geq 1 \}$$

- Idee:
  - Prüfe, ob die Eingabe aus 1…10…0 besteht
  - Wenn ja, dann prüfe ob (Anzahl Einsen) = (Anzahl Nullen)

δ	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
#	-	$(q_2, \#, L)$	-	$(q_4, \#, R)$	-	$(q_7, \#, N)$	$(q_2, \#, L)$	-
0	$(q_1, 0, R)$	$(q_1, 0, R)$	$(q_3, \#, L)$	$(q_3, 0, L)$	-	$(q_6, 0, R)$	$(q_6, 0, R)$	-
1	$(q_0, 1, R)$	-	-	$(q_3, 1, L)$	$(q_5, \#, R)$	$(q_6, 1, R)$	$(q_6, 1, R)$	-

Zunächst nur 1en

Dann nur 0en

Rechteste 0 streichen

Ganz nach links

Linkeste 1 streichen

Ganz nach rechts und Schleife, wenn noch Zahlen da sind, sonst akzeptieren

- $\text{TM} = (Q, \sum, \Gamma, \delta, q_0, \#, F)$ mit $Q = \{q_0, q_1, \dots, q_7\}, \sum = \{0,1\}, \Gamma = \{\#, 0,1\}, F = \{q_7\}$