Grundlagen der Informatik

- Einführung in Berechenbarkeit und Komplexität -

Prof. Dr. Klaus Volbert

Hochschule für angewandte Wissenschaften
Fakultät Informatik und Mathematik

Wintersemester 2010/11
Regensburg, 21. Dezember 2010

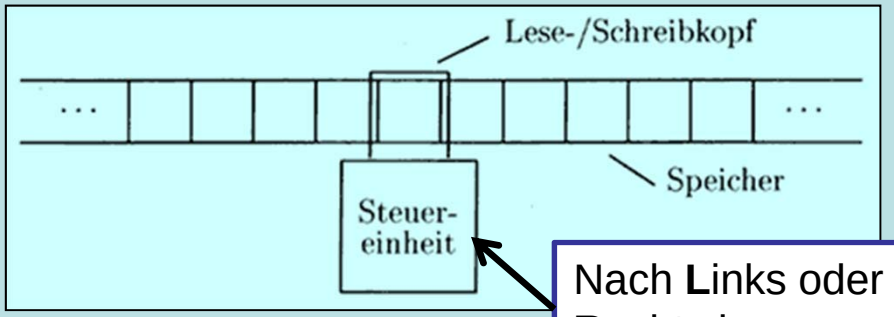
Turingmaschine (Alan M. Turing, 1936)

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



Nach **L**inks oder **R**echts bewegen oder **N**ichts tun!

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

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

Turing-Berechenbarkeit

- Eine (partielle) Funktion $f: \mathrm{IN}^k \rightarrow \mathrm{IN}$ heißt Turing-berechenbar, falls es eine (deterministische) Turingmaschine TM gibt, die bei Eingabe von $x_1, \ldots, x_n$ die Ausgabe $f(x_1, \ldots, x_n)$ liefert und hält, d.h.

  $q_0 x_1, \ldots, x_n \overset{*}{\rightarrow} qy$ mit $q \in F$ und y $\in \Gamma^*$ und $y = f(x_1, \ldots, x_n)$

- Falls TM nicht hält, ist $f(x_1, \ldots, x_n)$ nicht definiert
- TM-äquivalente Berechenbarkeitsmodelle
  - Registermaschinen (RAM)
  - Goto-Programme (IF…GOTO…)
  - While-Programme (IF…THEN…, WHILE …DO …)
  - μ-rekursive Funktionen

  } Beweis durch Simulation

- Ausdrucksschwächere Berechenbarkeitsmodelle
  - Loop-Programme (IF…THEN…, FOR …TO … DO…)
  - Primitiv-rekursive Funktionen

  } äquivalent

Akzeptanz und Entscheidbarkeit

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Im Gegensatz zu einem DFA kann es bei einer TM zu einer Endlosschleife kommen
  - Erinnerung DFA: Endliche Eingabe wird einmal von links nach rechts gelesen und verarbeitet, d.h. die Verarbeitung terminiert
- Akzeptanz
  - Eine TM M akzeptiert eine Sprache L, falls M alle $x \in L$ akzeptiert (d.h. M gestartet mit $x \in L$ hält in einem akzeptierenden Endzustand)
  - Anmerkung: Es kann $x \notin L$ existieren mit M gestartet mit x hält nicht
- Entscheidbarkeit
  - Eine TM M entscheidet eine Sprache L, falls M die Sprache L akzeptiert und für alle $x \notin L$ nach endlich vielen Schritten in einem nicht akzeptierenden Endzustand hält
- Eine Sprache L heißt
  - rekursiv aufzählbar (semi-entscheidbar) $\Leftrightarrow$ Es gibt eine TM, die L akzeptiert
  - rekursiv oder entscheidbar $\Leftrightarrow$ Es gibt eine TM, die L entscheidet

Bemerkungen

- Sei L eine Sprache, dann ist $\bar{L}$ die Komplementärsprache
  - $x \notin L \Leftrightarrow x \in \bar{L}$
- Eine Sprache L ist entscheidbar genau dann, wenn L und $\bar{L}$ aufzählbar sind
- Für zwei Sprachen L_1 und L_2 gelten:
  - L_1 und L_2 aufzählbar, dann auch $L_1 \cap L_2$ und $L_1 \cup L_2$

    Wie „arbeitet“ eine TM für die Schnitt- bzw. Vereinigungsmenge?

  - L_1 entscheidbar, dann auch $\overline{L_1}$
  - L_1 und L_2 entscheidbar, dann auch $L_1 \cap L_2$ und $L_1 \cup L_2$

Chomsky-Hierarchie

Bez.	Typ	Grammatik	Sprachen	Minimaler Automat
L_0	Typ-0	uneingeschränkt	aufzählbar	TM
L_1	Typ-1	kontextsensitiv	kontextsensitiv	Linear beschränkte TM (Eingabebereich wird nicht verlassen)
L_2	Typ-2	kontextfrei	kontextfrei	Kellerautomat
L_3	Typ-3	regulär	regulär	DFA

Entscheidbar

$$L_3 \subseteq L_2 \subseteq L_1 \subseteq L_0$$

- Beispiele
  - Die Sprache $\{ a^n b^n \mid n \geq 1 \}$ ist kontextfrei, aber nicht regulär
  - Die Sprache $\{ a^n b^n c^n \mid n \geq 1 \}$ ist kontextsensitiv, aber nicht kontextfrei
  - Das Halteproblem ist vom Typ-0, aber nicht kontextsensitiv

Anmerkungen zu Turing-Maschinen

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Erweiterungen der Turing-Maschine sind nicht mächtiger als Turing-Maschinen selbst
- Beispiele (Nachweis durch Simulationen)
  - Mehrere Spuren auf einem Band
    (z.B. mehr als ein Zeichen an einer Position)
  - Mehrere Lese- /Schreibköpfe auf einem Band
  - Ein Lese- /Schreibkopf auf mehreren Bändern
  - Mehrere Lese- /Schreibköpfe für mehrere Bänder
- Satz: Jede k-Band TM kann durch eine 1-Band TM ohne „kritischen/merkbaren“ Zeit- und Platzverlust simuliert werden
- Anmerkung
  - Zeit- und Platzkomplexitätsklassen bzgl. TM und RAM sind im Wesentlichen ebenfalls identisch

Universelle Turingmaschine

- Bisher: TM ist definiert über ein 7-Tupel, das von vornherein festlegt, welche Operationen ausgeführt werden sollen
- Eine TM M_0 heißt universell, wenn sie als Eingabe eine geeignete (Binär-)Codierung $< M >$ (Gödelnummer) einer TM M und ein $x \in \{0,1\}^*$ erhält, und folgendes gilt:

M_0 gestartet mit $< M > x$

verhält sich genauso wie

M gestartet mit x

(Programmierung der TM über ein Programm)

- Satz: Es gibt eine universelle TM die jede zeit- und platz-beschränkte TM ohne „kritischen/merkbaren“ Zeit- und Platzverlust simuliert

Unentscheidbare Sprachen I

- Gibt es Probleme/Sprachen, für die zwar ein Algorithmus angegeben werden kann, der aber nicht mit einem Computer ausgeführt werden kann?
- Spezielles Halteproblem

  $H_{\text{spez}} = \{< M > | \; M \text{ ist TM und } M \text{ gestartet mit } < M > \text{ hält nicht}\}$
- Beweis: Annahme H_{spez} ist entscheidbar
  - Dann ist H_{spez} aufzählbar und es existiert eine universelle TM M_0, die H_{spez} akzeptiert
  - Was macht M_0 mit Eingabe $< M_0 >$?
    - 1. Fall: $< M_0 > \in H_{\text{spez}}$, dann: M_0 akzeptiert $< M_0 >$, d.h. M_0 gestartet mit $< M_0 >$ hält nicht
    - 2. Fall: $< M_0 > \notin H_{\text{spez}}$, dann: M_0 akzeptiert $< M_0 >$ nicht, d.h. M_0 gestartet mit $< M_0 >$ hält
  - Folglich ist H_{spez} nicht aufzählbar und nicht entscheidbar

Unentscheidbare Sprachen II

- Allgemeines Halteproblem
  - $H = \{< M > x \mid M \text{ ist TM und } M \text{ gestartet mit } x \text{ hält}\}$
- Komplement des Halteproblems
  - $\overline{H} = \{< M > x \mid M \text{ ist TM und } M \text{ gestartet mit } x \text{ hält nicht}\}$
- Satz: Das Halteproblem ist nicht entscheidbar
- Beweisidee (Reduktion: $H_{\text{spez}} \leq \overline{H}$):
  - Annahme: Das Halteproblem ist entscheidbar
  - Dann ist $\overline{H}$ aufzählbar und es existiert eine universelle TM M_0, die $\overline{H}$ akzeptiert
  - Erhält M_0 nun die Eingabe $< M_0 >< M_0 >$, dann kann ermittelt werden, ob M_0 gestartet mit $< M_0 >$ nicht hält
  - Widerspruch, da H_{spez} nicht aufzählbar ist
  - Folglich ist das Halteproblem nicht entscheidbar
- Bemerkung: Das allgemeine Halteproblem ist aufzählbar

Anwendung/Interpretation

- Gibt es folgendes Programm (z.B. programmiert in C/C++)?
- Eingabe:
  - Quelltext eines Programms
  - Eingaben zu dem Programm
- Ausgabe:
  - Das Programm ausgeführt mit den Eingaben terminiert bzw. terminiert nicht (gerät/gerät nicht in eine Endlosschleife)

So ein Programm kann es nach der Church‘schen These auch zukünftig in keiner Programmiersprache geben, da es kein TM-Programm mit der gleichen Funktionalität gibt!

Informaler Widerspruchsbeweis in C

```
bool ProgrammMitEingabeHaelt(Programm p, Eingabe e)
{
      if (p(e) terminiert) return true;
      else return false;
}

void TesteProgramm(Programm p)
{
      while (ProgrammMitEingabeHaelt(p,p) == true) ;
}
```

- TesteProgramm(TesteProgramm):
  - 1. Fall: TesteProgramm(TesteProgramm) terminiert, dann geht TesteProgramm in eine Endlosschleife (while) und terminiert nicht
  - 2. Fall: TesteProgramm(TesteProgramm) terminiert nicht, dann wird die while-Schleife sofort beendet und TesteProgramm terminiert

Weitere nicht entscheidbare/berechenbare Probleme

- Äquivalenzproblem
  - Berechnen zwei Algorithmen A und B die gleiche Funktion?

$$A = \{\{< M_1 >, < M_2 >\} \mid M_1, M_2 \text{ sind TM und akzept. die gleiche Sprache}\}$$

- Totalitätsproblem
  - Ist die Funktion an allen Stellen definiert?

$$T = \{< M > \mid M \text{ hält für jede Eingabe}\}$$

- Endlichkeitsproblem
  - Ist die Funktion für endliche viele Eingaben definiert?

$$E = \{< M > \mid M \text{ hält für endlich viele Eingaben}\}$$

- Satz von Rice: Probleme der Art „Eingabe: Algorithmus A, Eigenschaft E, Ausgabe: E ist erfüllt/nicht erfüllt“ sind nicht berechenbar