universität innsbruck

Einführung in die Theoretische Informatik

David Drexel
Julian Fodor
Alexander Lochmann
Georg Moser
Alexander Maringele
David Obwaller
Jonas Schöpf

cbr.uibk.ac.at

Frohes Neues!

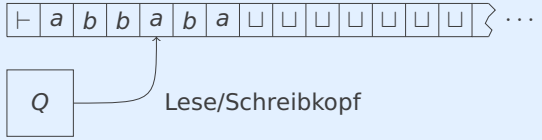
universität innsbruck

Zusammenfassung

Zusammenfassung der letzten LVA

Definition (informell)

deterministische, einbändige Turingmaschine (TM):



- Eine TM verwendet ein einseitig unendliches Band als Speicher
- Zu Beginn der Berechnung steht die Eingabe auf dem Band
- Der Speicher wird mit einem Lese/Schreibkopf gelesen oder beschrieben
- Das Verhalten der TM wird durch die endliche Kontrolle Q kontrolliert

Feedback

Stimmt es, dass die TM als formales Objekt gar kein Speicherband hat, sondern dieses erst durch ihre formale Konfiguration erhält?

Antwort

Nein.

Definition

eine deterministische, einbändige Turingmaschine (TM) M ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

sodass (u.a).

- Γ eine endliche Menge von Bandsymbolen, mit $\Sigma \subseteq \Gamma$,
- $\vdash \in \Gamma \setminus \Sigma$, der linke Endmarker,
- $\sqcup \in \Gamma \setminus \Sigma$, das Blanksymbol,
- $\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ die Übergangsfunktion,

Einführung in die Logik

Syntax & Semantik der Aussagenlogik, Formales Beweisen, KNF und DNF

Einführung in die Algebra

algebraische Strukturen, Beispiele von Algebren, Zusammenhang Boolesche Algebra und Aussagenlogik, Universelle Algebra, Satz von Birkhoff

Einführung in die Theorie der Formalen Sprachen

Grammatiken und Formale Sprachen, Chomsky-Hierarchie, Reguläre Sprachen, Kontextfreie Sprachen, Anwendungen von formalen Sprachen

Einführung in die Berechenbarkeitstheorie und Komplexitätstheorie

Algorithmisch unlösbare Probleme, Turing Maschinen, Registermaschinen, P $\neq$ NP

Einführung in die Programmverifikation

Prinzipien der Analyse von Programmen, Verifikation nach Hoare

Registermaschinen

Definition

Eine Registermaschine (RM) R ist ein Paar $R = ((x_i)_{1 \leqslant i \leqslant n}, \mathrm{P})$ sodass

1. $(x_i)_{1 \leqslant i \leqslant n}$ eine Sequenz von n Registern x_i, die natürliche Zahlen beinhalten
2. P ein Programm

Programme sind endliche Folgen von Befehlen und sind induktiv definiert:

1. Für jedes Register x_i sind die folgenden Instruktionen sowohl Befehle wie Programme: $x_i := x_i + 1$ und $x_i := x_i - 1$
2. wenn P_1, P_2 Programme sind, dann ist $P_1; P_2$ ein Programm
3. wenn P_1 ein Programm und x_i ein Register, dann ist

   $$\text{while } x_i \neq 0 \text{ do } P_1 \text{ end}$$

   sowohl ein Befehl als auch ein Programm

Definition (Semantik von Registermaschinen)

1. Zu Beginn der Berechnung steht die Eingabe (als natürliche Zahlen) in den Registern
2. Die Befehle
   - $x_i := x_i + 1$
   - $x_i := x_i - 1$

   bedeuten, dass der Inhalt des Register x_i entweder um 1 erhöht oder vermindert wird
3. $P_1; P_2$ bedeutet, dass zunächst das Programm P_1 und dann das Programm P_2 ausgeführt wird
4. Der Befehl (und das Programm)

   $$\text{while } x_i \neq 0 \text{ do } P_1 \text{ end}$$

   bedeutet, der Schleifenrumpf P_1 wird ausgeführt, bis die Bedingung $x_i \neq 0$ falsch ist

Beispiel

Sei $R = ((x_i)_{1 \leqslant i \leqslant 5}, \mathrm{P})$ eine RM mit dem folgenden Programm:

Zuweisung $x_i := x_j$

```
while x_i ≠ 0 do
  x_i := x_i − 1
end;
while x_k ≠ 0 do
  x_k := x_k − 1
end
```

```
while x_j ≠ 0 do
  x_i := x_i + 1;
  x_j := x_j − 1;
  x_k := x_k + 1
end;
while x_k ≠ 0 do
  x_j := x_j + 1;
  x_k := x_k − 1
end
```

Multiplikation

```
x_3 := 0;
while x_1 ≠ 0 do
  x_1 := x_1 − 1;
  x_4 := x_2;
  while x_2 ≠ 0 do
    x_2 := x_2 − 1;
    x_3 := x_3 + 1
  end;
  x_2 := x_4
end
```

Bei Eingabe $(m, n, 0, 0, 0)$ berechnet R $(0, n, m \times n, n, 0)$

Definition (Berechenbarkeit mit einer RM)

- Sei $R = ((x_i)_{1 \leqslant i \leqslant n}, \mathrm{P})$ eine RM
- Eine partielle Funktion $f \colon \mathbb{N}^k \to \mathbb{N}$, heißt R-berechenbar, wenn gilt

  $f(n_1, \ldots, n_k) = m$ gdw. R mit n_i in den Registern x_i für $1 \leqslant i \leqslant k$ startet und die Programmausführung mit n_i in den Registern x_i für $1 \leqslant i \leqslant k$ und m im Register x_{k+1} abbricht.
- Eine partielle Funktion $f \colon \mathbb{N}^k \to \mathbb{N}$ heißt berechenbar auf einer RM, wenn eine RM R existiert, sodass f R-berechenbar

Satz

Jede partielle Funktion $f \colon \mathbb{N}^k \to \mathbb{N}$, die berechenbar auf einer RM ist, ist auf einer TM berechenbar und umgekehrt

Church-Turing-These („Naturgesetz“ der Informatik)

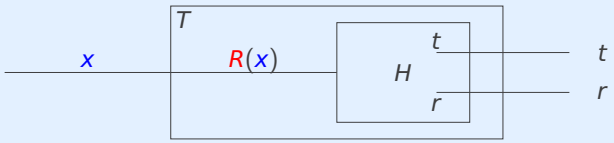
Jedes algorithmisch lösbare Problem ist auch mit Hilfe einer Turingmaschine lösbar.

(Turing-)Reduktion

angenommen

- L, M Sprachen über Σ
- $L \leqslant_T M$ mit $R \colon \Sigma^* \to \Sigma^*$
- die Reduktion R wird von TM T berechnet

$$x \in L \Leftrightarrow R(x) \in M$$



Anwendungen von Reduktionen

Lemma

wenn $L \leqslant_T M$ und M rekursiv, dann ist L rekursiv

Unentscheidbarkeit

Unentscheidbarkeit eines Problems zeigt man mittels Reduktion von einem unentscheidbares Problem (zum Beispiel das Halteproblem) auf das betrachtete Problem

Satz

es kann kein Testprogramm für “hello, world” Programme geben

Beweis.

$\mathrm{HP} \leqslant_T$ “hello, world” Programme

Satz

Sei Σ ein Alphabet und $L \subseteq \Sigma^*$ rekursiv; dann ist $\sim L$ rekursiv

Beweis.

Da L rekursiv ist, gibt es eine totale TM M mit $L = \mathrm{L}(M)$. Wir definieren eine TM M', wobei der akzeptierende und der verwerfende Zustand von M vertauscht werden. Weil M total ist, ist auch M' total. Somit akzeptiert M' ein Wort genau dann, wenn M es verwirft und es folgt $\sim L = \mathrm{L}(M')$, d.h. $\sim L$ ist rekursiv. ∎

Satz

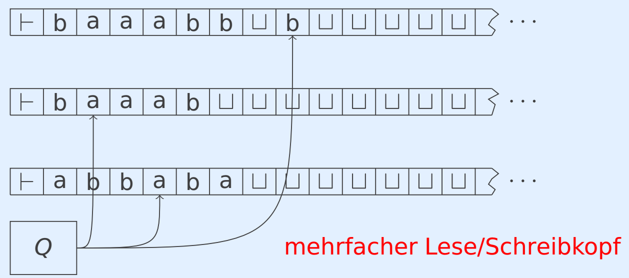
Jede rekursive Menge ist rekursiv aufzählbar. Andererseits ist nicht jede rekursiv aufzählbare Menge rekursiv.

Beweis.

Der erste Teil des Satzes ist eine Konsequenz der Definitionen; der zweite Teil wird in „Diskrete Strukturen“ bewiesen werden. ∎

Definition (informell)

Erweiterung um mehrere Bänder und Lese/Schreibköpfe:



Definition

$$\delta\colon Q \times \Gamma^3 \to Q \times \Gamma^3 \times \{\mathrm{L}, \mathrm{R}\}^3$$

Satz

Sei M eine k-bändige TM. Dann existiert eine (einbändige) TM M', sodass $\mathrm{L}(M) = \mathrm{L}(M')$

Beweisskizze.

- wir simulieren die Bänder übereinander, oBdA sei $k = 2$
- wir erweitern das Alphabet von M'

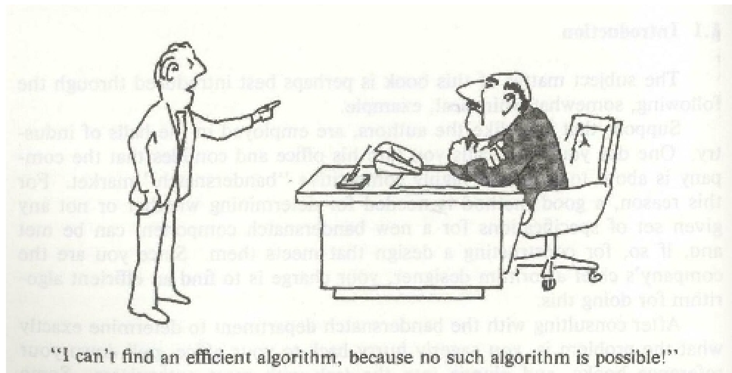
$$\begin{bmatrix} a \\ c \end{bmatrix} \quad \begin{bmatrix} \hat{a} \\ c \end{bmatrix} \quad \begin{bmatrix} a \\ \hat{c} \end{bmatrix} \quad \begin{bmatrix} \hat{a} \\ \hat{c} \end{bmatrix}$$

- Band von M' kann folgende Gestalt haben:

b	$\hat{b}$	a	b	a	a	a	a	b	a	a	a	...
c	c	c	d	d	d	c	$\hat{c}$	d	c	d	c	

- alle Bänder von M sind nun repräsentiert und die Leseköpfe werden durch die Zusatzmarkierung ^ ausgedrückt ∎

Berechenbarkeitstheorie, graphisch



universität innsbruck

Einführung in die Komplexitätstheorie

Definition

Komplexitätstheorie analysiert Algorithmen und Probleme:

Welche Resourcen benötigt ein bestimmter Algorithmus oder ein Problem?

Resourcen

- Speicherplatz
- Rechenzeit
- ...

Problem & Algorithmus

Wir unterscheiden zwischen

- der Komplexität eines Algorithmus — Algorithmus von Quine: $2^{c \cdot n}$ (wobei n die maximale binäre Länge der Eingabe)
- der Komplexität eines Problems — SAT ist in NP

Definition

sei M eine totale TM

- die Laufzeitkomplexität von M ist Funktion $T : \mathbb{N} \to \mathbb{N}$, wobei T wie folgt definiert

$$T(n) := \max\{m \mid M \text{ hält bei Eingabe } x, \ell(x) = n, \text{ nach } m \text{ Schritten}\}$$

- $T(n)$ bezeichnet die Laufzeit von M, wenn n die Länge der Eingabe
- M heißt T-Zeit-Turingmaschine

Definition

sei $T : \mathbb{N} \to \mathbb{N}$ eine numerische Funktion

$$\text{DTIME}(T) := \{L(M) \mid M \text{ ist eine mehrbändige TM mit Laufzeit (ungefähr) in } T\}$$

NB: Formal gilt $\exists c \in \mathbb{R}^+ \, \exists m \, \forall n \geqslant m$ Lauftzeit von M bei Eingabe $x \leqslant T(n)$, wobei $\ell(x) = n$.

Die Klasse P und NP

Definition

$$\text{P} := \bigcup_{k \geqslant 1} \text{DTIME}(n^k)$$

Example

betrachte SAT als Sprache:

$$\text{SAT} = \{F \mid F \text{ Formel mit erfüllbarer Belegung v}\}$$

es gilt SAT $\in$ DTIME(2^n), aber es ist nicht bekannt ob SAT $\in$ P

Definition

- ein Verifikator einer Sprache $L \subseteq \Sigma^*$, ist ein Algorithmus V sodass
$$L = \{x \in \Sigma^* \mid \exists\, c, \text{ sodass } V \text{ akzeptiert Eingabe } (x, c)\}$$
- ein polytime Verifikator ist ein Verifikator mit (ungefährer) Laufzeit n^k wobei $\ell(x) = n$
- Wort c wird Zertifikat genannt

Definition

NP ist die Klasse der Sprachen, die einen polytime Verifikator haben

Example

- Es gilt SAT $\in$ NP.
- Als Zertifikat wählen wir die (erfüllende) Belegung v für F. Gegeben die korrekte Belegung kann leicht (in polynomieller Zeit) nachgewiesen werden, dass v(F) = T.

Reduktionen (in polynomieller Zeit)

Definition

1. $\exists$ k-Band TM M mit Eingabealphabet Σ
2. M läuft in polynomieller Zeit
3. bei Eingabe $x \in \Sigma^*$, schreibt M, $R(x)$ auf das (erste) Band

dann heißt $R\colon \Sigma^* \to \Delta^*$ in polynomieller Zeit berechenbar

Definition

1. $\exists\, R\colon \Sigma^* \to \Delta^*$
2. R berechenbar in polynomieller Zeit
3. für $L \subseteq \Sigma^*$, $M \subseteq \Delta^*$ gilt $x \in L \Leftrightarrow R(x) \in M$

dann ist L in polynomieller Zeit auf M reduzierbar; kurz: $L \leqslant^{\mathrm{p}} M$

Definition

1. $\mathcal{C}$ eine beliebige Komplexitätsklasse
2. L eine Sprache über Σ und
3. $\forall$ Sprachen $M \in \mathcal{C}$ gilt: $M \leqslant^{\mathrm{p}} L$

dann ist L $\leqslant^{\mathrm{p}}$-hart für $\mathcal{C}$

Beispiel

SAT ist $\leqslant^{\mathrm{p}}$-hart für NP

Definition

für eine Sprache L, sei
1. L $\leqslant^{\mathrm{p}}$-hart für $\mathcal{C}$ und
2. $L \in \mathcal{C}$

dann ist L $\leqslant^{\mathrm{p}}$-vollständig für $\mathcal{C}$ oder (kurz) $\mathcal{C}$-vollständig

Komplexitätstheorie, graphisch

"I can't find an efficient algorithm, but neither can all these famous people."