Berechenbarkeit und Komplexität

Seite im Web: www.informatik.uni-leipzig.de/~wolter

Literatur:

- Uwe Schöning: Theoretische Informatik - kurzgefaßt, Spektrum Akademischer Verlag;
- J. Hopcroft und J. Ullman: Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie, Addison Wesley (Englische Version besser!);
- K. Erk und L. Priese: Theoretische Informatik, Springer Verlag;
- I. Wegener: Theoretische Informatik, Teubner Verlag;
- V. Sperschneider und B. Hammer: Theoretische Informatik, Springer Verlag.

Berechenbarkeitstheorie?

- Schreibe ein Programm, das bei Eingabe zweier natürlicher Zahlen n, m deren Summe $n + m$ ausgibt.
- Schreibe ein Programm, das bei Eingabe einer natürlichen Zahl n "ja" ausgibt, falls n eine Primzahl ist – und sonst "nein".
- Schreibe ein Programm, das bei Eingabe einer Liste diese sortiert.

Zur Lösung dieser Probleme trägt die Berechenbarkeitstheorie *nicht* bei.

Berechenbarkeitstheorie?

- Schreibe ein Programm, das bei Eingabe eines Programms P und einer Eingabe x "ja" ausgibt, falls P bei Eingabe von x terminiert – und sonst "nein".
- Schreibe ein Programm, das bei Eingabe eines Programms P und einer Eingabe x genau dann "ja" ausgibt, wenn P bei Eingabe x terminiert.

- Schreibe ein Programm, das bei Eingabe einer endlichen Wissensbasis Γ und einer Formel F (in Prädikatenlogik) "ja" ausgibt, falls $\Gamma \models F$ – und sonst "nein".
- Schreibe ein Programm, das bei Eingabe einer endlichen Wissensbasis Γ und einer Formel F (in Prädikatenlogik) genau dann "ja" ausgibt, wenn $\Gamma \models F$.

- Schreibe ein Programm, das bei Eingabe einer Grammatik G und eines Wortes w "ja" ausgibt, falls $w \in L(G)$ – und "nein" sonst.
- Schreibe ein Programm, das bei Eingabe einer Grammatik G und eines Wortes w genau dann "ja" ausgibt, wenn $w \in L(G)$.

Berechenbarkeitstheorie ist eine Theorie zur Präzisierung des Begriffs "*nicht berechenbarer Funktionen*".

Plan

- Turing-Berechenbarkeit
- LOOP-, WHILE- und GOTO-Berechenbarkeit
- Rekursive Funktionen
- Unentscheidbarkeit
- Postsches Korrespondenzproblem
- Komplexitätsklassen
- NP-Vollständigkeit

Turing-Berechenbarkeit

Eine (deterministische) Turing Maschine (TM) ist eine Tupel

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, E),$$

wobei:

- Q endliche Menge von Zuständen;
- Σ endliche Menge von Eingabesymbolen;
- $\Gamma \supseteq \Sigma$ endliche Menge von Bandsymbolen;
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{N, R, L\}$ Überführungsfunktion;
- $q_0 \in Q$ Startzustand;
- $\Box \in \Gamma - \Sigma$ das "Blank";
- $E \subseteq Q$ Endzustände.

- Eine Konfiguration von M ist ein Wort in $\Gamma^* Q \Gamma^*$;
- Startkonfiguration bei Eingabe $w \in \Sigma^*$: $q_0 w$.

Die Relation $\vdash$

Die Übergangsrelation $\vdash$ ist eine 2-stellige Relation auf dem Konfigurationen:

$$a_1 \cdots a_m q b_1 \cdots b_n \vdash \begin{cases} a_1 \cdots a_m q' c b_2 \cdots b_n & : \quad \delta(q, b_1) = (q', c, N) \\ a_1 \cdots a_m c q' b_2 \cdots b_n & : \quad \delta(q, b_1) = (q', c, R) \\ a_1 \cdots q' a_m c b_2 \cdots b_n & : \quad \delta(q, b_1) = (q', c, L) \end{cases}$$

Berechnete Funktion auf Σ^*

Eine Turing Maschine M berechnet eine partielle Funktion $f : \Sigma^* \to \Sigma^*$, falls für alle $x, y \in \Sigma^*$ gilt:

$$f(x) = y \Leftrightarrow \exists q_e \in E \quad q_0 x \vdash^* q_e y.$$

Ist f für x nicht definiert, so terminiert M nicht.

Bemerkung: Jede durch eine nichtdeterministische TM berechenbare Funktion ist auch durch eine deterministische TM berechenbar.

Berechnete Funktion auf N

Eine Turing Maschine M berechnet eine (partielle) Funktion $f : \mathbf{N}^k \to \mathbf{N}$ falls für alle $n_1, \ldots, n_k, m \in \mathbf{N}$:

$$f(n_1, \ldots, n_k) = m$$

$$\Leftrightarrow$$

$$\exists q_e \in E \quad q_0 bin(n_1) \sharp bin(n_2) \cdots \sharp bin(n_k) \vdash^* q_e bin(m)$$

$bin(n)$ ist die Binärdarstellung von n.

Ist f für $(n_1, \ldots, n_k)$ nicht definiert, so terminiert M nicht.

LOOP-Programme

Induktive Definition der Syntax von LOOP:

- Wertzuweisungen $x_i := x_j + c$, $x_i := x_j - c$ sind LOOP-Programme, wobei $c \in \{0, 1, 2 \ldots\}$ und x_i, x_j Variablen.
- Sind P_1, P_2 LOOP-Programme, so auch $P_1; P_2$.
- Ist P ein LOOP-Programm und x_i eine Variable, so ist auch

  $$\text{LOOP } x_i \text{ DO } P \text{ END}$$

  ein LOOP-Programm.

Semantik

Soll ein LOOP-Programm eine Funktion $f : \mathbf{N}^k \to \mathbf{N}$ berechnen und ist $(n_1, \ldots, n_k)$ das Argument, dann gilt am Anfang:

- $x_i = n_i$ für $1 \leq i \leq k$ und $x_i = 0$ sonst.

Wertzuweisung wird wie üblich interpretiert, wobei x_i auf 0 gesetzt wird, falls für $x_i := x_j - c$ der Wert von x_j kleiner als c ist.

"LOOP x_i DO P END" wird so interpretiert, daß P so oft ausgeführt wird, wie der Wert der Variablen x_i bei Eintritt in die Schleife angibt.

LOOP-Berechenbarkeit

Eine partielle Funktion $f : \mathbf{N}^k \to \mathbf{N}$ heißt LOOP-berechenbar, falls es ein LOOP-Programm P gibt, das gestartet mit $n_1, \ldots, n_k$ in den Variablen $x_1, \ldots, x_k$ genau dann mit dem Wert $f(n_1, \ldots, n_k)$ in x_0 terminiert, wenn f für $(n_1, \ldots, n_k)$ definiert ist. Sonst terminiert P nicht.

Jede LOOP-berechenbare Funktion ist total definiert.

WHILE-Programme

In WHILE-Programmen wird statt "LOOP x_i DO P END" die Schleife

$$\text{WHILE } x_i \neq 0 \text{ DO } P \text{ END}$$

verwendet.

Semantik: P wird ausgeführt, bis $x_i = 0$.

WHILE-Berechenbarkeit

Eine partielle Funktion $f : \mathbf{N}^k \to \mathbf{N}$ heißt WHILE-berechenbar, falls es ein WHILE-Programm P gibt, das gestartet mit $n_1, \ldots, n_k$ in den Variablen $x_1, \ldots, x_k$ genau dann mit dem Wert $f(n_1, \ldots, n_k)$ in x_0 terminiert, wenn f für $n_1, \ldots, n_k$ definiert ist. Sonst terminiert P nicht.

GOTO-Programme

GOTO Programme sind Folgen von markierten Anweisungen

$$M_1 : A_1; \ldots ; M_k : A_k$$

mit Anweisungen A_i der folgenden Form:

- Wertzuweisungen wie oben;
- unbedingter Sprung: GOTO M_i;
- bedingter Sprung: IF $x_i = c$ THEN GOTO M_i;
- Stopanweisung: HALT.

GOTO-Berechenbarkeit

Eine partielle Funktion $f : \mathbf{N}^k \to \mathbf{N}$ heißt GOTO-berechenbar, falls es ein GOTO-Programm P gibt, das gestartet mit $n_1, \ldots, n_k$ in den Variablen $x_1, \ldots, x_k$ genau dann mit dem Wert $f(n_1, \ldots, n_k)$ in x_0 terminiert, wenn f für $(n_1, \ldots, n_k)$ definiert ist. Sonst terminiert P nicht.

Zusammenhänge

Sei $f : \mathbf{N}^k \to \mathbf{N}$ eine partielle Funktion. Die folgenden Aussagen sind äquivalent:

- f ist durch eine TM berechenbar;
- f ist WHILE-berechenbar;
- f ist GOTO-berechenbar.

LOOP-berechenbare Funktionen sind immer total. Die Ackermann-Funktion ist Beipiel einer totalen TM-berechenbaren Funktion, die nicht LOOP-berechenbar ist.

Churchsche These

Die durch den formalen Begriff der TM-Berechenbarkeit erfaßte Klasse von Funktionen stimmt mit der Klasse der intuitiv berechenbaren Funktionen überein.

Ackermann-Funktion

Die Ackermann-Funktion $\alpha : \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ ist definiert durch:

- $\alpha(0, n) = n + 1$;
- $\alpha(n + 1, 0) = \alpha(n, 1)$;
- $\alpha(n + 1, m + 1) = \alpha(n, \alpha(n + 1, m))$.

Es gilt: $\alpha(5, 4) > 10^{10000}$.

Die Ackermann-Funktion ist nicht LOOP-berechenbar.

Zum Beweis

Wir können zeigen (Übung):

- $\alpha(n, m) > m$;
- $\alpha(n, m+1) > \alpha(n, m)$; (Monotonie im 2. Argument)
- $\alpha(n+1, m) \geq \alpha(n, m+1)$;
- $\alpha(n+1, m) > \alpha(n, m)$; (Monotonie im 1. Argument)
- $\alpha(n+2, m) > \alpha(n, 2m)$.

Zum Beweis

Für ein LOOP-Programm P in den Variablen $x_{i_0}, \ldots, x_{i_k}$ bezeichne $\max P(\vec{n})$ den maximalen Wert der $x_{i_0}, \ldots, x_{i_k}$ nach Ausführung von P bei Eingabe $\vec{n} = n_{i_0}, \ldots, n_{i_k}$.

Ackermann-Lemma (Übung): Zu jedem LOOP-Programm P existiert ein $k_P \in \mathbf{N}$ mit

$$\max P(\vec{n}) < \alpha(k_P, \max(\vec{n})) \text{ für alle } \vec{n}.$$

Aus der Annahme der Existenz eines LOOP-Programms, das die Ackermann-Funktion berechnet, folgt nun sofort mit dem Ackermann-Lemma ein Widerspruch.

Fleißige Biber

Ein Biber ist eine TM über $\Sigma = \Gamma = \{\Box, 1\}$, die auf das leere Band angesetzt stoppt. Eine TM heißt *fleißiger Biber* (busy beaver), wenn sie Biber ist und unter den Bibern mit gleicher Anzahl von Zuständen bei Termination die maximale Anzahl von 1en auf dem Band liefert. Die busy beaver Funktion BB ist definiert durch $BB(n)$= Anzahl der gelieferten 1en eines fleißigen Bibers mit n Zuständen.

Fleißige Biber

Es gilt:

- $BB(n) < BB(n+1)$.
- Ist $f : \mathbf{N} \to \mathbf{N}$ TM-berechenbar mit $f(n) < f(n+1)$, so gilt $f(m) < BB(m)$ für alle m größer einem gewissen m_f.
- BB ist nicht TM-berechenbar.

Primitiv rekursive Funktionen

Die Klasse der primitiv rekursiven Funktionen $f : \mathbf{N}^k \to \mathbf{N}$ ist induktiv definiert:

- Alle konstanten Funktionen sind primitiv rekursiv;
- Projektionsabbildungen pr_i^k sind primitiv rekursiv:

$$pr_i^k(n_1, \ldots, n_k) = n_i,\ 1 \leq i \leq k.$$

- Die Nachfolgerfunktion $s : \mathbf{N} \to \mathbf{N}$ ist primitiv rekursiv:

$$s(n) = n + 1.$$

- *Einsetzung*: Jede Funktion, die durch Einsetzung von primitiv rekursiven Funktionen in primitiv rekursiven Funktionen entsteht, ist primitiv rekursiv.

Primitiv rekursive Funktionen

- *Primitive Rekursion*: Seien $g : \mathbf{N}^k \to \mathbf{N}$ und $h : \mathbf{N}^{k+2} \to \mathbf{N}$ primitiv rekursiv. Dann ist auch auch $f : \mathbf{N}^{k+1} \to \mathbf{N}$ mit

  $$f(0, \vec{m}) = g(\vec{m}), \quad f(n+1, \vec{m}) = h(n, f(n, \vec{m}), \vec{m})$$

  primitiv rekursiv.

Die primitiv rekursiven Funktionen bilden also die kleinste Klasse von Funktionen, die alle konstanten Funktionen, die Projektionsfunktionen und die Nachfolgerfunktion enthält und unter Einsetzung und primitiver Rekursion abgeschlossen ist.

Die Klasse der primitiv rekursiven Funktionen stimmt mit der Klasse der LOOP-berechenbaren Funktionen überein.

μ-rekursive Funktionen

Sei $f : \mathbf{N}^{k+1} \to \mathbf{N}$ gegeben. Durch Anwendung des μ-Operators erhält man $g : \mathbf{N}^k \to \mathbf{N}$ mit

$$g(n_1, \ldots, n_k) = \min \left\{ \begin{array}{l} n \mid f(n, n_1, \ldots, n_k) = 0 \ \wedge \\ \forall m < n f(m, n_1, \ldots, n_k) \text{ definiert} \end{array} \right\}$$

(Hier $\min \emptyset =$ undefiniert.)

Die Klasse der μ-rekursiven Funktionen ist die kleinste Klasse von Funktionen, die alle konstanten Funktionen, Projektionen und die Nachfolgerfunktion enthält und abgeschlossen unter Einsetzung, primitiver Rekursion und μ-Rekursion ist.

Die Klasse der μ-rekursiven Funktionen stimmt mit der Klasse der WHILE-berechenbaren Funktionen überein.

Rekursive Prädikate

Ein Prädikat $A \subseteq \mathbf{N}^k$ heißt primitiv rekursiv, falls die charakteristische Funktion

$$\chi_A(\vec{n}) = \begin{cases} 1 & : \quad \vec{n} \in A \\ 0 & : \quad \vec{n} \notin A \end{cases}$$

primitiv rekursiv ist.

Entscheidbarkeit

Eine Menge $A \subseteq \Sigma^*$ heißt *entscheidbar*, falls die charakteristische Funktion

$$\chi_A(w) = \begin{cases} 1 & : \quad w \in A \\ 0 & : \quad w \notin A \end{cases}$$

TM-berechenbar ist.

Eine Menge $A \subseteq \Sigma^*$ heißt *semi-entscheidbar*, falls die Funktion

$$\chi_A^u(w) = \begin{cases} 1 & : \quad w \in A \\ \text{undefiniert} & : \quad w \notin A \end{cases}$$

TM-berechenbar ist.

Entscheidbarkeit

A ist genau dann entscheidbar, wenn A und $\Sigma^* - A$ semi-entscheidbar sind.

$A \subseteq \Sigma^*$ heißt *rekursiv aufzählbar*, falls $A = \emptyset$ oder es gibt eine totale berechenbare Funktion

$$f : \mathbf{N} \to \Sigma^*$$

mit

$$A = \{f(0), f(1), \ldots\}.$$

A ist genau dann rekursiv aufzählbar, wenn A semi-entscheidbar ist.

Unentscheidbare Probleme

Sei *code* eine injektive und berechenbare Funktionen, die jeder TM $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, E)$ ein Wort $code(M) \in \{0,1\}^+$ zuordnet.

Sei N eine beliebige (aber fixierte) TM und setze für $w \in \{0,1\}^*$

$$M_w = \begin{cases} M & : \quad code(M) = w \\ N & : \quad \text{sonst} \end{cases}$$

Das spezielle Halteproblem

$$\{w \in \{0,1\}^* \mid M_w \text{ hält bei Eingabe von } w\}$$

ist nicht entscheidbar.

Reduzierbarkeit

Seien $A, B \subseteq \Sigma^*$. A heißt auf B reduzierbar ($A \preceq B$), falls es eine totale und berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ gibt, so daß für alle $w \in \Sigma^*$:

$$w \in A \Leftrightarrow f(w) \in B.$$

Aus $A \preceq B$ folgt:

- Falls B entscheidbar ist, so ist A entscheidbar.
- Falls B semi-entscheidbar ist, so ist A semi-entscheidbar.

Allgemeines Halteproblem

Die Sprache

$$\{w\sharp x \mid w, x \in \{0,1\}^* \text{ und } M_w \text{ hält bei Eingabe von } x\}$$

ist unentscheidbar.

Satz von Rice

Sei $\mathcal{R}$ die Klasse aller TM-berechenbaren Funktionen und $\emptyset \neq \mathcal{S} \subset \mathcal{R}$. Dann ist

$$\{w \in \{0,1\}^* \mid \text{ die von } M_w \text{ berechnete Funktion liegt in } \mathcal{S}\}$$

unentscheidbar.

Reichlich Anwendungen.

Postsches Korrespondenzproblem (PCP)

Eine Folge von Paaren $(v_1, w_1), \ldots, (v_k, w_k)$ von nichtleeren Wörtern über Σ heißt Instanz (des PCP).

Gibt es eine Folge von Indizes $i_1, \ldots, i_n \in \{1, \ldots, k\}$ mit $n \geq 1$ und

$$v_{i_1} \cdots v_{i_n} = w_{i_1} \cdots w_{i_n},$$

so besitzt die Instanz eine Lösung.

Spiel:

http://www.informatik.uni-leipzig.de/~pcp/pcp4de.html

PCP

PCP ist die Menge der Instanzen, die eine Lösung besitzen.

Theorem: PCP ist unentscheidbar.

Zum Beweis:

- Zunächst Reduktion von MPCP auf PCP.
- Dann Reduktion des Halteproblems auf MPCP.

MPCP

Sei die Folge $(v_1, w_1), \ldots, (v_k, w_k)$ von nichtleeren Wörtern über Σ gegeben.

Falls es eine Folge von Indizes $i_1, \ldots, i_n \in \{1, \ldots, k\}$ mit

$$v_1 v_{i_1} \cdots v_{i_n} = w_1 w_{i_1} \cdots w_{i_n}$$

gibt, so sagen wir, daß die Instanz des PCP eine Lösung mit *fixierter erster Stelle* besitzt.

MPCP ist die Menge der Instanzen, die eine Lösung mit fixierter erster Stelle besitzen.

MPCP ist auf PCP reduzierbar: Beweis

Seien $\sharp, S \notin \Sigma$. Setze für jedes Wort $w = c_1 \cdots c_k$ über Σ:

$$\overline{w} = \sharp c_1 \sharp \cdots \sharp c_k \sharp, \quad w^a = \sharp c_1 \sharp \cdots \sharp c_k, \quad w^e = c_1 \sharp \cdots \sharp c_k \sharp$$

Dann besitzt $(v_1, w_1), \ldots, (v_k, w_k)$ eine Lösung mit fixierter erster Stelle gdw.

$$(\overline{v_1}, w_1^a), (v_1^e, w_1^a), (v_2^e, w_2^a), \ldots (v_k^e, w_k^a), (S, \sharp S)$$

eine Lösung besitzt.

Das Halteproblem ist auf MPCP reduzierbar: Beweis

- Jedem Paar (M, w) mit $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, E)$ und $w \in \Sigma^*$ wird
- effektiv
- eine Instanz $(v_1, w_1), \ldots, (v_k, w_k)$ über $\Gamma \cup Q \cup \{\sharp\}$ zugeordnet,
- die genau dann eine Lösung mit fixierter erster Stelle besitzt, wenn M bei Eingabe w hält.

Die Instanz

Sei also (M, w) gegeben. Die gesuchte Instanz ist dann wie folgt definiert:

- 1. Paar: $(\sharp, \sharp q_0 w \sharp)$;
- 1. Gruppe: (w, w), für $w \in \Gamma \cup \{\sharp\}$;
- 2. Gruppe: fr̈ $a \in \Gamma$
  - $(qa, q'c)$, falls $\delta(q, a) = (q', c, N)$;
  - (qa, cq'), falls $\delta(q, a) = (q', c, R)$;
  - $(bqa, q'bc)$, falls $\delta(q, a) = (q', c, L)$ (für all $b \in \Gamma$);
  - $(\sharp qa, \sharp q' \Box c)$, falls $\delta(q, a) = (q', c, L)$;

Die Instanz

- $(q\sharp, q'c\sharp)$, falls $\delta(q, \Box) = (q', c, N)$;
- $(q\sharp, cq'\sharp)$, falls $\delta(q, \Box) = (q', c, R)$;
- $(bq\sharp, q'bc\sharp)$, falls $\delta(q, \Box) = (q', c, L)$ und $b \in \Gamma$.
- 3. Gruppe: (aq_e, q_e) und $(q_e a, q_e)$ für alle $a \in \Gamma$ und $q_e \in E$;
- 4. Gruppe: $(q_e\sharp\sharp, \sharp)$ für alle $q_e \in E$.

Unentscheidbarkeit für Formale Sprachen

Unter dem *Schnittproblem* für kontextfreie Sprachen versteht man die Frage, ob für vorgegebene kontextfreie Grammatiken G_1 und G_2 die Gleichung $L(G_1) \cap L(G_2) = \emptyset$ gilt.

Theorem. Das Schnittproblem für kontextfreie Sprachen ist unentscheidbar.

Zum Beweis: Jeder Instanz V des PCP wird effektiv ein Paar (G_1, G_2) zugeordnet, so daß V genau dann eine Lösung besitzt, wenn $L(G_1) \cap L(G_2) = \emptyset$.

Das Äquivalenzproblem

Unter dem *Äquivalenzproblem* für kontextfreie Sprachen versteht man die Frage, ob für vorgegebene kontextfreie Grammatiken G_1 und G_2 die Gleichung $L(G_1) = L(G_2)$ gilt.

Theorem. Das Äquivalenzproblem für kontextfreie Sprachen ist unentscheidbar.

Zum Beweis: Durch Reduktion des Schnittproblems.

Unentscheidbarkeit der Prädikatenlogik

Theorem. Die Menge der allgemeingültigen Formeln der Prädikatenlogik ist nicht entscheidbar.

Zum Beweis. Durch Reduktion des PCP. Seien c eine Konstante, f_1, f_2 einstellige Funktionssymbole und p ein zweistelliges Prädikat. Sei die Instanz

$$I = (v_1, w_1), (v_2, w_2), \ldots, (v_k, w_k)$$

gegeben. Abkürzung: $f_{i_1 \ldots i_l}(t) = f_{i_1}(\cdots(f_{i_l}(t)\cdots)$.

Zum Beweis

Setze

$$\varphi_1 = \bigwedge_{1 \leq i \leq k} p(f_{v_i}(c), f_{w_i}(c)),$$

$$\varphi_2 = \forall x \forall y (p(x,y) \rightarrow \bigwedge_{1 \leq i \leq k} p(f_{v_i}(x), f_{w_i}(y))),$$

$$\varphi_3 = \exists x p(x,x).$$

I hat genau dann eine Lösung, wenn $\varphi_1 \wedge \varphi_2 \rightarrow \varphi_3$ allgemeingültig ist.

Der Gödelsche Satz

Sei Σ_A die folgende Signatur der Arithmetik:

- Die Konstanten $\overline{0}, \overline{1}, \ldots;$
- Die zweistelligen Funktionssymbole $\overline{+}$ und $\overline{*}$.

$\mathcal{L}_A$ bezeichnet die prädikatenlogische Sprache über Σ_A (mit $=$).

$\mathcal{L}_A$ wird auf die übliche Weise in der Struktur

$$\mathcal{N} = \langle \{0, 1, \ldots, \}, +, * \rangle$$

interpretiert.

Theorem:

Die Menge der in $\mathcal{N}$ wahren Sätze aus $\mathcal{L}_A$ ist nicht rekursiv aufzählbar (nicht semi-entscheidbar).

Zum Beweis

Eine Funktion $f : \mathbf{N}^k \to \mathbf{N}$ heißt *repräsentierbar*, falls es eine Formel $\varphi(x_1, \ldots, x_k, y) \in \mathcal{L}_A$ gibt, so daß für alle $n_1, \ldots, n_k, m \in \mathbf{N}$

$$f(n_1, \ldots, n_k) = m \quad \Leftrightarrow \quad \mathcal{N} \models \varphi(\overline{n_1}, \ldots, \overline{n_k}, \overline{m}).$$

Theorem:

Jede While-berechenbare Funktion ist repräsentierbar.

Logik 2. Stufe

In Logik 2. Stufe gibt es neben Quantifikation über Individuen auch Quantifikation über Mengen. Syntaktisch wird Logik 1. Stufe dabei um die folgende Klausel erweitert:

- Ist φ eine Formel und P ein Prädikat, so ist auch $\forall P \varphi$ eine Formel.

Beispiel (das Induktionsschema):

$$\forall P[(P(0) \wedge \forall x(P(x) \rightarrow P(x+1))) \rightarrow \forall x P(x)]$$

Theorem:

Die Menge der allgemeingültigen Formeln der Logik 2. Stufe ist nicht rekursiv aufzählbar.

Komplexitätstheorie

Berechenbarkeitstheorie: Klassifikation von Problemen in entscheidbare, semi-entscheidbare und unentscheidbare Probleme.

Theorie der Grade der Unentscheidbarkeit: Untersuchung der Feinstruktur der Klasse der unentscheidbaren Probleme.

Komplexitätstheorie/Effiziente Algorithmen: Untersuchung der Feinstruktur der Klasse der entscheidbaren Probleme gemäß ihres Bedarfs an 'Berechnungsressourcen' (Zeit und Platzbedarf).

Komplexitätstheorie/Effiziente Algorithmen

Effiziente Algorithmen: Feinstruktur der Klasse der in polynomialer Zeit lösbaren Probleme (Stichworte: Vorlesung Algorithmen und Datenstrukturen, O-Notation).

Komplexitätstheorie: Gröbere (von Maschinen und Datenstrukturen unabhängige) Klassifikation von Problemen in polynomiale, exponentielle, nichtdeterministisch polynomiale etc. Probleme.

Klassen für deterministische Berechnungen

Sei M eine det. TM (mit möglicherweise $k > 1$ Bändern) über Σ.

- $time_M : \Sigma^* \to \mathbf{N}$ gibt die Anzahl der Rechenschritte von M bei Eingabe w an (falls M bei Eingabe w terminiert).
- $space_M : \Sigma^* \to \mathbf{N}$ gibt die maximale Anzahl der während der Berechnung von M bei Eingabe w beschriebenen Felder an (falls diese beschränkt ist).

Sei F eine Klasse von Funktion $f : \mathbf{N} \to \mathbf{N}$. Dann bezeichnet

$$TIME(F)$$

die Klasse aller Sprachen L (mit Alphabet Σ), für die eine det. TM M und ein $f \in F$ existiert, so daß

- M berechnet die charakteristische Funktion χ_L von L;
- $time_M(w) \leq f(|w|)$ für alle $w \in \Sigma^*$.

Sei F eine Klasse von Funktion $f : \mathbf{N} \to \mathbf{N}$. Dann bezeichnet

$$SPACE(F)$$

die Klasse aller Sprachen L (mit Alphabet Σ), für die eine det. TM M und ein $f \in F$ existiert, so daß

- M berechnet die charakteristische Funktion χ_L von L;
- $space_M(w) \leq f(|w|)$ für alle $w \in \Sigma^*$.

Bemerkungen

- $TIME(F) \subseteq SPACE(F)$;
- $L \in TIME(F) \Rightarrow \Sigma^* - L \in TIME(F)$.
- $L \in SPACE(F) \Rightarrow \Sigma^* - L \in SPACE(F)$.
- Sind alle Funktionen in F LOOP-berechenbar, so sind alle charakteristischen Funktionen χ_L mit $L \in TIME(F) \cup SPACE(F)$ LOOP-berechenbar.

Wichtige Komplexitätsklassen

- P bezeichnet die Klasse der Polynome der Form

$$p(n) = a_k n^k + a_{n-1} n^{k-1} + \cdots + a_1 n + a_0.$$

  Setze $\mathbf{P} = TIME(P)$ und $\mathbf{PSPACE} = SPACE(P)$.

- E bezeichnet die Klasse der Funktionen $c^{p(n)}$, wobei $p \in P$ und $c \in \mathbf{N}$.

  Setze $\mathbf{EXPTIME} = TIME(E)$, $\mathbf{EXPSPACE} = SPACE(E)$.

Nichtdeterministische TM

Bei einer nichtdeterminische TM $M = (Q, \Sigma, \Gamma, \delta, \Box, E, q_0)$ ordnet δ jedem Paar $(q, a) \in Q \times \Gamma$ eine endliche Teilmenge von $Q \times \Gamma \times \{L, N, R\}$ zu.

Die Übergangsrelation $\vdash$ ist eine 2-stellige Relation auf dem Konfigurationen:

$$a_1 \cdots a_m q b_1 \cdots b_n \vdash \begin{cases} a_1 \cdots a_m q' c b_2 \cdots b_n & : \quad (q', c, N) \in \delta(q, b_1) \\ a_1 \cdots a_m c q' b_2 \cdots b_n & : \quad (q', c, R) \in \delta(q, b_1) \\ a_1 \cdots q' a_m c b_2 \cdots b_n & : \quad (q', c, L) \in \delta(q, b_1) \end{cases}$$

Berechnete charakteristische Funktion

Sei $L \subseteq \Sigma^*$. Eine nichtdeterministische TM M berechnet χ_L, falls für alle $w \in \Sigma^*$ gilt:

- Jede Berechnung bei Eingabe w terminiert mit Endkonfiguration $q_e 1$ oder $q_e 0$, $q_e \in E$;
- $w \in L$ genau dann, wenn es eine Berechnung

$$q_0 w \vdash \alpha_1 q_{i_1} \beta_1 \vdash \cdots \vdash q_e 1$$

gibt.

Sei M eine nichtdet. TM (mit möglicherweise $k > 1$ Bändern) über Σ.

- $time_M : \Sigma^* \to \mathbf{N}$ gibt die maximale Anzahl der Rechenschritte von M bei Eingabe w an (falls M bei Eingabe w immer terminiert).
- $space_M : \Sigma^* \to \mathbf{N}$ gibt die maximale Anzahl der beschriebenen Felder der Berechnung von M bei Eingabe w an (falls diese immer beschränkt ist).

Sei F eine Klasse von Funktionen $f : \mathbf{N} \to \mathbf{N}$. Dann bezeichnet

$$NTIME(F)$$

die Klasse aller Sprachen L (mit Alphabet Σ), für die eine nichtdet. TM M und ein $f \in F$ existiert, so daß

- M berechnet die charakteristische Funktion χ_L von L;
- $time_M(w) \leq f(|w|)$ für alle $w \in \Sigma^*$.

Sei F eine Klasse von Funktion $f : \mathbf{N} \to \mathbf{N}$. Dann bezeichnet

$$NSPACE(F)$$

die Klasse aller Sprachen L (mit Alphabet Σ), für die eine nichtdet. TM M und ein $f \in F$ existiert, so daß

- M berechnet die charakteristische Funktion χ_L von L;
- $space_M(w) \leq f(|w|)$ für alle $w \in \Sigma^*$.

Wichtige Komplexitätsklassen

- Setze $\mathbf{NP} = NTIME(P)$ und $\mathbf{NPSPACE} = NSPACE(P)$.
- Setze $\mathbf{NEXPTIME} = NTIME(E)$ und $\mathbf{NEXPSPACE} = NSPACE(E)$.
- Setze $coNTIME(F) = \{\Sigma^* - L | L \in NTIME(F)\}$. und $coNSPACE(F) = \{\Sigma^* - L | L \in NSPACE(F)\}$.

Beziehungen

- **PSPACE** $=$ **NPSPACE**.
- **P** $\subseteq$ **NP** $\subseteq$ **PSPACE** $\subseteq$ **EXPTIME**.
- **NP** $=$ **coNP** ist ein offenes Problem.
- **NP** $=$ **P** ist ein offenes Problem.

Analoge Aussagen gelten für die exponentiellen Komplexitätsklassen.

Reduzierbarkeit

Seien $L_1 \subseteq \Sigma^*$ und $L_2 \subseteq \Gamma^*$. L_1 heißt polynomial auf L_2 reduzierbar ($L_1 \preceq_p L_2$), falls es eine polynomial berechenbare Funktion

$$f : \Sigma^* \to \Gamma^*$$

gibt, so daß für all $w \in \Sigma^*$:

$$w \in L_1 \Leftrightarrow f(w) \in L_2.$$

Bemerkung: Die Relation $\preceq_p$ ist transitiv.

Reduzierbarkeit

Es gelte $L_1 \preceq_p L_2$. Dann folgt:

- Wenn $L_2 \in \mathbf{P}$, so $L_1 \in \mathbf{P}$;
- Wenn $L_2 \in \mathbf{NP}$, so $L_1 \in \mathbf{NP}$;
- Wenn $L_2 \in \mathbf{PSPACE}$, so $L_1 \in \mathbf{PSPACE}$.
- Wenn $L_2 \in \mathbf{EXPTIME}$, so $L_1 \in \mathbf{EXPTIME}$.
- Wenn $L_2 \in \mathbf{NEXPTIME}$, so $L_1 \in \mathbf{NEXPTIME}$.

NP-Vollständigkeit

Eine Sprache L heißt **NP**-hart, falls für alle Sprachen $L' \in \mathbf{NP}$ gilt:

$$L' \preceq_p L.$$

L heißt **NP**-vollständig, falls $L \in \mathbf{NP}$ und L ist **NP**-hart.

Erfüllbarkeit

Das Erfüllbarkeitsproblem der Aussagenlogik ist wie folgt formuliert: Gibt es für vorgelegte Formel φ eine Belegung V, so daß $V \models \varphi$.

Als formale Sprache:

$$\text{SAT} = \{\varphi \mid \exists V : V \models \varphi\}.$$

Theorem SAT ist **NP**-vollständig.

Zum Beweis

Sei L eine Sprache und

$$M = (Q, \Sigma, \Gamma, \delta, q_0, E, \Box)$$

eine nichtdeterministische TM, die L entscheidet, und deren Rechenzeit durch das Polynom p beschränkt ist. So konstruieren wir für jedes $w \in \Sigma^*$ eine Formel φ_w, so daß

$$\varphi_w \text{ erfüllbar gdw } w \in L.$$

Desweiteren $l(\varphi_w) \in O(|w|^2)$.

Variablen in φ_w für $n = |w|$

Für $t = 0, \ldots, p(n)$, $i = -p(n), \ldots, p(n)$, $q \in Q$, $a \in \Gamma$:

- $Z_{t,q}$: Nach t Schritten befindet sich M in Zustand q;
- $P_{t,i}$: Der Schreib-Lesekopf befindet sich nach t Schritten auf Position i;
- $B_{t,i,a}$: Nach t Schritten befindet sich auf Position i das Zeichen a.

Weitere NP-vollständige Probleme

3-SAT: Gegeben eine aussagenlogische Formel φ in konjunktiver Normalform mit höchstens drei Literalen in jeder Klausel. Ist φ erfüllbar?

CLIQUE: Gegeben ein ungerichteter Graph $\mathcal{G} = (G, E)$ und eine natürliche Zahl n.

Gibt es eine Menge $V \subseteq G$ mit $|V| = k$, so daß für je zwei $u, v \in V$ gilt: $(u, v) \in E$?

Oder: Wie groß ist die größte Clique in $\mathcal{G}$?

Oder: Welches ist die größte Clique in $\mathcal{G}$?

RUCKSACK: Gegeben natürliche Zahlen $a_1, \ldots, a_k$ und b. Gibt es eine Menge $I \subseteq \{1, \ldots, k\}$, so daß

$$\sum_{i \in I} a_i = b.$$

TRAVELING SALESMAN: Gegeben eine $k \times k$-Matrix $(m_{i,j})$ von Zahlen (Kosten für Reisen von i nach j). Gibt es eine Permutation π von $\{1, \ldots, k\}$, so daß

$$\sum_{i=1}^{k-1} m_{\pi(i),\pi(i+1)} + m_{\pi(k),\pi(1)} \leq k.$$

Oder: Was sind die Kosten der billigsten Rundreise?

Oder: Wie sieht die billigste Rundreise aus?

Problem der ganzzahligen linearen Programmierung:

Gegeben eine $m \times n$-Matrix A ganzer Zahlen und ein Vektor b von n ganzen Zahlen. Gibt es einen Vektor x ganzer Zahlen, so daß

$$A \cdot x \geq b.$$

Dieses Problem ist **NP**-vollständig.

coNP versus NP

Definition: $L \in \mathbf{coNP}$ gdw $\Sigma^* - L \in \mathbf{NP}$.

- $\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{coNP} \subseteq \mathbf{NP} \cup \mathbf{coNP} \subseteq \mathbf{PSPACE}$;
- $\mathbf{NP} = \mathbf{coNP}$ gdw das Komplement eines **NP**-vollständigen Problems in **NP** ist;
- PRIMZAHL und ZAHL−PRIMZAHL sind in **NP**.

PSPACE

QBF (Quantifierte boolesche Formeln) ist die Sprache 1. Stufe mit zweistelligen Funktionssysmbolen $\wedge$ und $\vee$, dem einstelligen Funktionssymbol $\neg$ und den Konstanten 0 und 1.

Das QBF-Problem ist wie folgt formuliert:

Gegeben: Satz φ (Formel ohne freie Variablen) aus QBF.

Frage: Ist φ in der 2-elementigen Booleschen Algebra wahr?

Theorem: Das QBF-Problem ist PSPACE-vollständig.

NEXPTIME

Sei Σ_+ die Signatur mit zweistelligen Funktionssymbol $\overline{+}$, der Relation $\overline{<}$ und den Konstanten $\overline{0}$ und $\overline{*}$. S_+ bezeichnet die Menge der Zätze der Logik erster Stufe (mit $=$) über dieser Signatur. Dann gilt:

$$\{\varphi \in \Sigma_+ \mid (\mathbf{R}, +, <, 0, 1) \models \varphi\}$$

ist **NEXPTIME**-vollständig.