0 VL0

- David Hilbert
- Travelling Salesman Problem (TSP)

Person: David Hilbert (1862-1943)

- Deutscher Mathematiker
- "Wir müssen wissen, wir werden wissen."

TSP Travelling Salesman Problem

- Eingabe: vollständiger Graph G, alle Kantenlängen (Gewichtung)
- Ausgabe: eine Rundreise, die alle Knoten in G besucht und dabei so kurz wie mögl. ist.
- $P \neq NP \implies$ kein effizienter Algorithmus existiert

1 VL1 - Turing Maschinen I

- Probleme
- Turingmaschinen
- rekursive / berechenbare Funktionen
- rekursive / entscheidbare Sprachen
- Konfigurationen
- Programmiertechniken: Speicher im Zustandsraum
- Programmiertechniken: Mehrspurmaschinen
- Programmiertechniken: Weiteres

Definition: Probleme

- **Problem als Relation** $R \subseteq \Sigma^* \times \Gamma^*$ für Alphabete Σ, Γ
- Es ist $(x, y) \in R \iff y$ ist zulässige Ausgabe zur Eingabe x
- Beispiel Primfaktorbestimmung:

$$R := \{(x, y) \in \{0,1\}^* \times \{0,1\}^* \mid x = \text{bin}(q), y = \text{bin}(p), q, p \in \mathbb{N}, q \geq 2, p \text{ prim}, p \mid q\}$$

- Bei eindeutiger Lösung **Problem als Funktion** $f : \Sigma^* \to \Gamma^*$
- Beispiel Multiplikation inklusive Trennzeichen:

$$f : \{0,1,\#\}^* \to \{0,1,\#\}, f(\text{bin}(i_1)\#\text{bin}(i_2)) = \text{bin}(i_1 \cdot i_2)$$

- **Problem als Entscheidungsproblem**: Form $f : \Sigma^* \to \{0,1\}$
- $L := f^{-1}(1) \subseteq \Sigma^*$ ist Sprache vom durch f definiertem Entscheidungsproblem.
- Beispiel Graphzusammenhang: Bestimme zur Eingabe $G = (V, E)$ ob G zsmhgd.
- Wenn Graph G codiert in Σ durch $code(G)$, so ist

$$L := \{w \in \Sigma^* \mid \exists \text{ zusammenhängender Graph } G : w = code(G)\}$$

  die zu diesem Entscheidungsproblem gehörende Sprache.

Definition: Turingmaschine (TM)

Eine Turingmaschine M ist gegeben durch $M = (Q, \Sigma, \Gamma, B, q_0, \overline{q}, \delta)$, wobei

- Q endliche Zustandsmenge
- Σ endliches Eingabealphabet
- $\Gamma \supsetneq \Sigma$ endliches Bandalphabet
- $B \in \Gamma \setminus \Sigma$ Leerzeichen, Blank
- $q_0 \in Q$ Anfangszustand
- $\overline{q}$ Endzustand
- $\delta : (Q \setminus \{\overline{q}\}) \times \Gamma \to Q \times \Gamma \times \{R, L, N\}$ Zustandsüberführungsfunktion

Weiteres:

- Startet in q_0, Kopf über (1. Symbol vom) Eingabewort eingerahmt von Blanks
- TM stoppt, sobald Endzustand $\overline{q}$ erreicht.
- Ausgabewort $y \in \Sigma^*$ beginnt Kopfposition und endet vor erstem Symbol in $\Gamma \setminus \Sigma$

- akzeptiert $\iff$ terminiert und Ausgabe beginnt mit 1
- verwirft $\iff$ terminiert und Ausgabe beginnt nicht mit 1

- **Laufzeit** ist Anzahl von Zustandsübergängen bis zur Terminierung
- **Speicherbedarf** Anzahl während Berechnung besuchter Bandzellen

- TM M **entscheidet** $L \subset \Sigma^*$ wenn M $w \in L$ akzeptiert und $w \notin L$ verwirft (hält stets)

Definition: rekursive / T-berechenbare Funktionen

$f : \Sigma^* \to \Sigma^*$ heißt rekursiv bzw (T-)berechenbar,
wenn es eine TM gibt welche bei Eingabe $x \in \Sigma^*$ stets $f(x)$ berechnet.

Definition: rekursive / T-entscheidbare Sprachen

Eine Sprache $L \subseteq \Sigma^*$ heißt rekursiv bzw (T-)entscheidbar,
wenn es eine TM gibt welche stets terminiert und $w \in \Sigma^*$ akzeptiert gdw. $w \in L$.

Definition: Konfiguration

Eine Konfiguration einer TM ist ein String $\alpha q \beta$, wobei $\alpha, \beta \in \Gamma^*, q \in Q$ wobei $\beta \neq \varepsilon$.
α entspricht dem Wort links vom Kopf, 1. Symbol von β unter dem Kopf (evtl. B), rest rechts.

$\alpha' q' \beta'$ ist **direkte Nacholgerkonfiguration** von $\alpha q \beta$,
wenn sie in einem Rechenschritt aus $\alpha q \beta$ entsteht. Man schreibt $\alpha q \beta \vdash \alpha' q' \beta'$.

Analog schreibt man für endlich viele (auch 0) Rechenschritte $\alpha q \beta \vdash^* \alpha'' q'' \beta''$.

Techniken zur Programmierung TM's (1): Speicher im Zustandsraum

Zu $k \in \mathbb{N}_{>0}$ können wir k Symbole des Bandalphabets Γ im Zustand abspeichern,
indem wir den Zustandsraum um den Faktor $|\Gamma|^k$ vergrößern, d.h.

$$Q_{neu} := Q \times \Gamma^k$$

Bspw sind neue Zustände für $k = 2$ dann (q_0, BB) oder $(q_1, 01)$ (wenn $0, 1 \in \Gamma$).

Techniken zur Programmierung TM's (2): Mehrspurmaschinen

k-spurige TM: TM mit zusätzlich k-Vektoren als Symbole für $k \in \mathbb{N}$. Man schreibt

$$\Gamma_{neu} := \Gamma \cup \Gamma^k$$

Techniken zur Programmierung TM's (3): Weiteres

- Variablen: pro Variable eine Spur
- Arrays: ebenfalls in einer Spur möglich
- Unterprogramme: eine Spur als Prozedurstack benutzen

2 VL2 - Turing Maschinen II

- k-Band-TM's
- Simulation von k-Band-TM's mit 1-Band-TM's
- Gödelnummern
- Universelle TM
- Alan Turing
- Alonzo Church
- Church-Turing-These

Definition: k-Band-TM

Besitzt $k \in \mathbb{N}_{>0}$ Arbeitsbänder mit unabhängigen Köpfen. Zustandsübergangsfkt ist dann

$$\delta : (Q \setminus \{\overline{q}\}) \times \Gamma^k \to Q \times \Gamma^k \times \{R, L, N\}^k$$

Dabei ist Band 1 das Eingabe / Ausgabeband. Die anderen sind zunächst leer (Blanks).

Satz: Simulation von k-Band TM's durch 1-Band-TM's

Eine k-Band TM M mit Zeitbedarf $t(n)$ und Platzbedarf $s(n)$ kann mit einer 1-Band-TM M' in Zeitbedarf $\mathcal{O}(t^2(n))$ und Platzbedarf $\mathcal{O}(s(n))$ simuliert werden.

Also **quadratischer Zeitverlust** und **konstanter Speicherverlust**.

Bewies vie $2k$ Spuren; Inhalt der Bänder und Positionen der Köpfe (markiert mit #).
Jeder Rechenschritt von M wird wie folgt durch M' simuliert:

- Kopf steht auf linkestem #, M' kennt Zustand von M.
- Laufe nach rechts und speichere alle Zeichen and den Kopfpositionen auf den zugeh. Bändern im Zustand.
- Werte damit δ_M aus
- Laufe zurück und verändere entsprechend Kopfpositionen / Bandinhalte

- Nach t Schritten von M können #'s höchstens $2t$ Positionen auseinanderliegen
- Simulation eines Schrittes also in $\mathcal{O}(t(n))$
- Für $t(n)$ Schritte damit $\mathcal{O}(t(n)^2)$

Definition: Gödelnummer

Die Gödelnummer einer TM M wird durch $\langle M \rangle$ bezeichnet.

- Eindeutige, **präfixfreie** Kodierung über $\{0, 1\}$.
- $\langle M \rangle$ beginnt und endet stets mit 111, enthält sonst 111 nicht.
- Man beschränkt sich auf TM's mit $Q = \{q_1, q_2, \cdots, q_t\}, t \geq 2$
  wobei q_1, q_2 Anfangs-/Endzustand sind. Ferner soll $\Gamma = \{0, 1, B\}$.
  Man kodiert den t-ten Übergang mit $code(t)$ in der Form $0^a10^b10^c10^d10^e$.
  Dann kodiert man die TM M mit s Übergängen durch:

$$\langle M \rangle = 111code(1)11code(2)11 \ldots 11code(s)111$$

Definition: Universelle Turingmaschine

Eingabe ist ein Wort der Form $\langle M \rangle w$ für $w \in \{0, 1\}^*$.

Simulation via 3-Spur TM in **konstanter Zeit** möglich: Gödelnr auf Spur 2, Zustand auf 3.

Person: Alan Turing (1912-1954)

- Englischer Mathematiker, Informatiker, Logiker, Philospher
- Angesehen als Vater der theoretischen Informatik

Person: Alonzo Church (1903-1995)

- Amerikanischer Mathematiker, Logiker
- Erfinder des Lambda-Calculus

Behauptung: Church-Turing-These (1930)

Die Klasse der TM-berechenbaren Funktionen stimmt mit der Klasse der
"intuitiv berechenbaren" Funktionen überein.

Daher (in dieser Vorlesung)

berechenbare Funktion = TM-berechenbare Funktion = rekursive Funktion

enstschiedbare Sprache = TM-entscheidbare Sprache = rekursive Sprache

3 VL3 - Registermaschinen

- Registermaschinen
- Kostenmaße
- Simulation RAM durch TM

Definition: Registermaschine (RAM)

Besteht aus Befehlszähler, Akkumulator (c(0)), unbeschränkter Speicher c(1),c(2),...
Programme haben Befehlssatz:

(IND/C)LOAD, (IND)STORE, (IND/C)ADD, (IND/C)SUB, (IND/C)MULT, (IND/C)DIV

IF c(0) ? x THEN GOTO j wobei j Zeile im Programm und ? $\in \{=, <, \leq, \geq, >\}$

GOTO, END

- Inhalt des Speichers sind Elemente von $\mathbb{N}$ (beliebig groß)
- Eingabe ebenfalls in $\mathbb{N}^*$, zu Beginn in den ersten Registern
- Andere Register mit 0 initialisiert.
- Befehlszähler startet mit 1. Als nächstes wird immer die Zeile, auf die der Befehlszähler verweist, ausgeführt.
- Rechnung stoppt sobald END ausgeführt wird.
- Ausgabe befindet sich dann in den ersten Registern.

Definition: Kostenmaße für RAM's

Uniformes Kostenmaß: Jeder Schritt / Befehl zählt eine Zeiteinheit

Logarithmisches Kostenmaß: Die Laufzeitkosten eines Schrittes sind Maximum der Logarithmen der involvierten Zahlen. (Maximale Zahlenlänge)

Satz: Simulation von RAM durch TM

Für jede im logarithmischen Kostenmass $t(n)$-zeitbeschränkte RAM R gibt es ein Polynom q und eine $q(n + t(n))$-zeitbeschränkte TM M, welche R simuliert.

Simulation hat also **polynomiellen Overhead**.