

# Die $P \neq NP$ -Vermutung

6. Mai 2015

Adrian Hein, Florian Weber

# Einführung

# Turingmaschine

- mathematische Abstraktion eines Computers
- besteht aus
  - Steuerwerk
  - unendlich langes Steuerband
  - Lese- und Schreibkopf

# Turingmaschine

- pro Schritt wird
  - ein Zeichen gelesen
  - ein Zeichen geschrieben
  - eine Bewegung ausgeführt
- jeder Schritt ist nur abhängig von
  - aktuellem Zeichen auf dem Band
  - aktuellem Zustand der TM
- eine TM hat endlich viele Zustände
- man kann Zustände als Endzustände definieren

# Turingmaschine formal

- formal besteht eine TM aus
  - $Q$ , die endlichen Zustandsmenge
  - $\Sigma$ , das endlichen Eingabealphabet
  - $\Gamma$ , das endliche Bandalphabet und es gilt  $\Sigma \subset \Gamma$
  - $\delta: (Q \setminus \{q_f\}) \times \Gamma \rightarrow Q \times \Gamma \times \{L, 0, R\}$  ist die (partielle) Überföhrungsfunktion
  - $q_0 \in Q$  ist der Anfangszustand
  - $\square \in \Gamma \setminus \Sigma$  steht für das leere Feld
  - $q_{accept} \in Q$  ist der akzeptierende Zustand

# Turingmaschine (nichtdeterministisch)

- ähnlich der deterministischen TM
- NDTM hat allerdings zwei Übergangsfunktionen  $\delta_0$  und  $\delta_1$
- endet eine Sequenz von Entscheidungen in  $q_{accept}$  gilt die Eingabe als akzeptiert
- im Gegensatz zur deterministischen TM nicht ohne Weiteres realisierbar

# Die Klasse P

- enthält alle Entscheidungsprobleme die in Polynomialzeit von einer TM lösbar sind
- Probleme in P gelten als praktisch lösbar
- Beispiele sind:
  - Lineare Programmierung/Optimierung
  - PRIMES (AKS-Primzahltest)
  - HORNSAT

# Die Klasse NP (formal)

Eine Sprache  $L \subseteq \{0, 1\}^*$  liegt in NP, wenn es ein Polynom  $p : \mathbb{N} \rightarrow \mathbb{N}$  sowie eine in Polynomialzeit laufende TM  $M$ , den sogenannten Verifizierer für  $L$ , gibt, sodass für jedes  $x \in \{0, 1\}^*$  gilt:

$$x \in L \Leftrightarrow \exists u \in \{0, 1\}^{p(|x|)} \text{ sodass } M(x, u) = 1$$

In diesem Fall nennt man  $u$  ein Zertifikat für  $x$ .



## Die Klasse NP (alternativ)

- alle Entscheidungsprobleme die von einer NDTM  $M$  in Polynomialzeit gelöst werden
- $x$  ist eine Lösung, wenn es eine Sequenz von Entscheidungen gibt, sodass  $M$  in  $q_{accept}$  hält.
  - es gilt in diesem Fall  $M(x) = 1$
- gibt es keine Sequenz für die  $M$  in  $q_{accept}$  gilt  $M(x) = 0$
- ursprüngliche Definition, deswegen auch NP (nondeterministic polynomial time)
- beide Definitionen äquivalent, da die Sequenz von Entscheidungen die zu  $q_{accept}$  führt als Verifizierer betrachtet werden kann

# Die Klasse coNP

- alle Sprachen, deren Komplement in NP liegt
- NICHT das Komplement zu NP
- Beispiel: Kontradiktion

# Reduktion

- $A$  heißt reduzierbar auf  $B$ , wenn es einen Algorithmus gibt, der aus jedem Problem aus  $A$  in Polynomialzeit ein Problem aus  $B$  macht
- gibt es einen Algorithmus zur Lösung von  $B$  und gilt  $A \preceq B$ , so kann dieser auch  $A$  lösen
- man sagt  $B$  ist mindestens so schwer wie  $A$

# NP-Vollständigkeit

- gilt  $L \preceq L'$ ,  $\forall L \in \text{NP}$ , so nennt man  $L'$  NP-schwer
- liegt  $L'$  selber auch in NP nennt man  $L'$  NP-vollständig
- um NP-schwere für  $L'$  zu zeigen genügt es  $L \preceq L'$  für ein NP-schweres  $L$  zu zeigen

# Cook-Levin Theorem

# konjunktive Normalform

- Jede boolsche Funktion lässt sich in konjunktiver Normalform darstellen
- TMs die Sprachen entscheiden, sind boolsche Funktionen
- Die Größe einer KNF für  $n$  Variablen liegt in  $O(n \cdot 2^n)$  o.B.
- Siehe auch: TI1 (Digitaltechnik)

## Reduktion \* auf SAT

- $O(n \cdot 2^n)$  offensichtlich zu groß.
- Sei  $M$  eine TM die eine NP-vollständige Sprache entscheidet und die
  - ein Eingabe- und ein Ausgabe/Arbeitsband habe
  - bei der die Position des Kopfes in Schritt  $i$  nur von der Länge der Eingabe abhängt
  - gültige Annahme, da in  $O(f(n)^2)$  simulierbar
- Sei  $Q$  die Menge der Zustände von  $M$
- Sei  $\Gamma$  das Bandalphabet von  $M$
- Sei  $\langle a, b, q \rangle_i \in Q \times Q \times \Gamma$  der Snapshot der TM in Schritt  $i$

## Reduktion \* auf SAT

- Snapshots können offensichtlich als Zeichenketten konstanter Länge kodiert werden.
- Ein Snapshot  $S_i$  hängt ab von:
  - $S_{i-1}$
  - einem Zeichen fester Position der Eingabe
  - dem letzten Snapshot an der selben Stelle
- Es gibt für jedes  $i$  genau einen korrekten Snapshot
- Daraus folgt: Es gibt eine Funktion  $f$ , die zwei Snapshots und eine Position auf dem Eingabeband auf einen neuen Snapshot abbilden:

$$S_i = f(S_{i-1}, S_{\text{prev}(i)}, E_{\text{inputpos}(i)})$$



## Reduktion \* auf SAT

- Um eine Lösung für die betrachtete Sprache zu finden, muss man eine Abfolge von TM-Schritten finden, die zum Ergebnis führt.
- Die einzelnen Schritte (und damit die Snapshotkette) kodieren eine Lösung, sind aber zunächst unbekannt.
- Die Snapshotkette lässt sich aber als polynomielle KNF schreiben.
- Angenommen, es gäbe einen Polyzeit-Entscheider für SAT, so könnte dieser damit auch die Kette von Snapshots für andere Probleme finden, und damit diese in Polyzeit entscheiden!

## Reduktion SAT auf 3SAT

Um eine SAT-Klausel  $(a_1 \vee a_2 \vee \dots \vee a_n)$  nach 3SAT zu konvertieren, genügt es, sie wie folgt zu schreiben:

$$(a_1 \vee a_2 \vee h_1) \wedge (\overline{h_1} \vee a_3 \vee h_2) \wedge \dots \wedge (\overline{h_{n-2}} \vee a_{n-1} \vee a_n)$$

Hierbei sind  $h_1 \dots h_{n-2}$  neu eingeführte Hilfsvariablen.

## Wichtige NP-vollständige Probleme

# MY HOBBY: EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS

CHOTCHKIES RESTAURANT	
~ APPETIZERS ~	
MIXED FRUIT	2.15
FRENCH FRIES	2.75
SIDE SALAD	3.35
HOT WINGS	3.55
MOZZARELLA STICKS	4.20
SAMPLER PLATE	5.80
~ SANDWICHES ~	
BARBECUE	6.55



Abbildung 1:CC-BY-NC 2.5, Randall Munroe, <https://xkcd.com/287/>

# INDSET

# 0/1 IPROG

- gegeben:  $m$  lineare Ungleichungen über  $n$  Variablen
- gesucht: eine Lösung für das System wobei die Variablen nur 0 oder 1 annehmen können
- in NP: die Belegung der Variablen kann als Zertifikat gesehen werden
- NP-vollständig:  $\text{SAT} \preceq 0/1 \text{ IPROG}$ , da jede Klausel als Ungleichung aufgefasst werden kann
  - $u_1 \vee \overline{u_2} \vee \overline{u_3}$  kann ausgedrückt werden durch
$$u_1 + (1 - u_2) + (1 - u_3) \geq 1$$

## Andere Klassen

# EXP und NEXP



# Sonstige

# Indizien

$P \neq NP$

$\text{coNP} \neq \text{NP}$

# Implikationen

# Philosophisch

# Mathematische Beweise

$$P = NP$$



$$\text{coNP} = \text{NP}$$

# Probleme zwischen P und NP

## Umgang mit NP-vollständigen Problemen



Abbildung 2:<http://everfalling.deviantart.com/art/DON-T-PANIC-15975789>

# Umgang mit NP-vollständigen Problemen

- Existieren vielleicht gute Näherungslösungen?
- Ist der Worst-Case wirklich wahrscheinlich?
- Gibt es andere Modellierungen in P?
- Ist  $n$  wirklich so groß, dass NP-Vollständigkeit ein Problem darstellt?