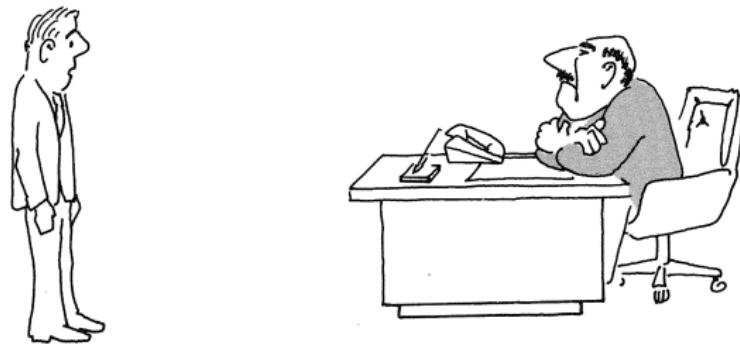


# Die $P \neq NP$ -Vermutung

Adrian Hein, Florian Weber

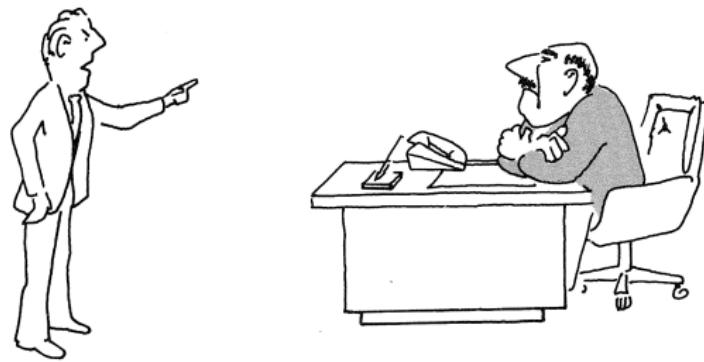
6. Mai 2015

# Motivation



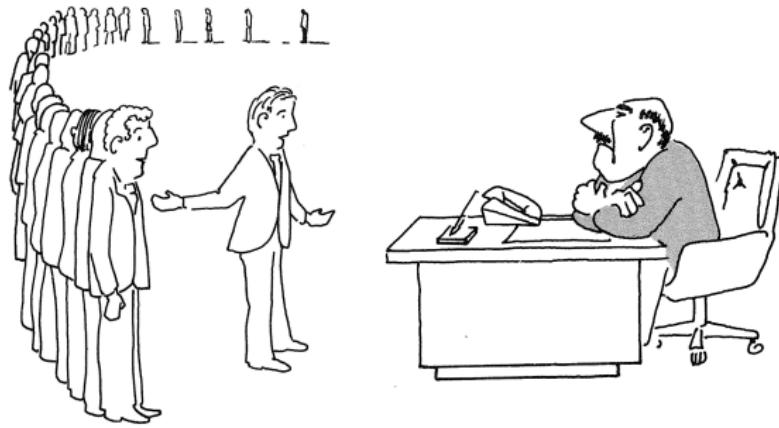
„I can't find an efficient algorithm, I guess I'm just too dumb.“

# Motivation



„I can't find an efficient algorithm, because no such algorithm is possible.“

# Motivation



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

# Themen

- Einführung
- Das Cook-Levin Theorem
- Wichtige NP-vollständige Probleme
- Andere Komplexitätsklassen
- Indizien
- Implikationen
- Umgang mit NP-vollständigen Problemen
- Zusammenfassung
- Quellen

# Einführung

## Turingmaschine

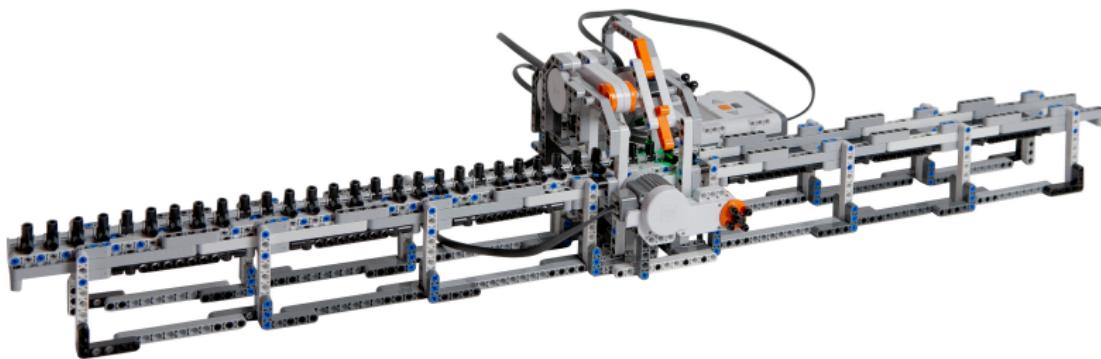
- Mathematische Abstraktion eines Computers
- Besteht aus:
  - Steuerwerk
  - unendlich langes Steuerband
  - Lese- und Schreibkopf

# Einführung

## 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

# Einführung Turingmaschine



# Einführung

## Turingmaschine formal

- Formal besteht eine TM aus einem Tupel  
 $\mathcal{M} := \{Q, \Sigma, \Gamma, \delta, q_0, F\}$  mit:
  - $Q$ , der endlichen Zustandsmenge
  - $\Sigma$ , dem endlichen Eingabealphabet
  - $\Gamma$ , dem endliche Bandalphabet und es gilt  $\Sigma \subset \Gamma$
  - $\delta: (Q \setminus F) \times \Gamma \rightarrow Q \times \Gamma \times \{L, 0, R\}$ , der (partiellen) Überführungsfunktion
  - $q_0 \in Q$ , dem Anfangszustand
  - $F \subseteq Q$ , der Menge der akzeptierenden Zustände
- $\square \in \Gamma \setminus \Sigma$  bezeichnet das leere Feld

# Einführung

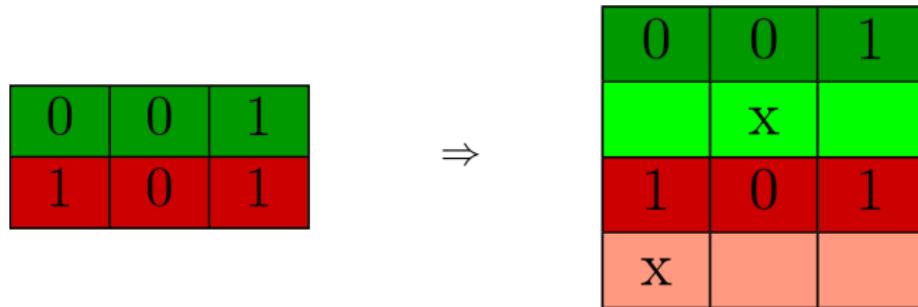
## Turingmaschine (nichtdeterministisch)

- Ähnlich der deterministischen TM
- NDTM hat allerdings mehrere Übergangsfunktionen
- Endet eine Sequenz von Entscheidungen in  $F$  gilt die Eingabe als akzeptiert
- Im Gegensatz zur deterministischen TM nicht ohne Weiteres realisierbar

# Einführung

## Turingmaschine (Mehrband)

- Hat anstatt einem Band mehrere mit jeweils einem Lese- und Schreibkopf



- Kann durch eine TM mit einem Band simuliert werden

0 | 1 | x | 0 | x | 0 | 1 | 1 |

- Mehrband TMs sind genauso mächtig wie normale TMs aber evt. anschaulicher

# Einführung

## Die Klasse P

- Enthält alle Entscheidungsprobleme die in Polynomialzeit von einer TM lösbar sind
- Probleme in P gelten als praktisch lösbar
- Ist unter Komplementbildung abgeschlossen
- Beispiele sind:
  - Lineare Programmierung/Optimierung
  - PRIMES (AKS-Primzahltest)
  - SET-COVER

# Einführung

## 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  $\mathcal{M}$ , den sogenannten Verifizierer für  $L$ , gibt
  - sodass für jedes  $x \in \{0, 1\}^*$  gilt:
    - $x \in L \Leftrightarrow \exists u \in \{0, 1\}^{p(|x|)}$ , sodass  $\mathcal{M}(x, u) = 1$
- In diesem Fall nennt man  $u$  ein Zertifikat für  $x$ .

# Einführung

## Die Klasse NP (alternativ)

- Alle Entscheidungsprobleme die von einer NDTM  $\mathcal{M}$  in Polynomialzeit gelöst werden
- $x$  ist eine Lösung, wenn es eine Sequenz von Entscheidungen gibt, sodass  $\mathcal{M}$  in einem der akzeptierenden Zustände ( $F$ ) hält.
- Ursprüngliche Definition, deswegen auch NP (nondeterministic polynomial time)
- Beide Definitionen äquivalent, da die Sequenz von Entscheidungen, die zu  $F$  führt als Zertifikat betrachtet werden kann

# Einführung

## Die Klasse coNP

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

# Einführung

## Reduktion

- $A$  heißt reduzierbar auf  $B$ , wenn es einen Algorithmus gibt, der aus jedem Problem aus  $A$  in Polynomialzeit ein Problem aus  $B$  macht
- Man schreibt dann  $A \preceq B$
- 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  $A$

# Einführung

## 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
- Ist ein Problem  $A$  NP-schwer, so ist das entsprechende Problem  $\overline{A}$  in coNP logischerweise coNP-schwer

# Das Cook-Levin Theorem



Stephen Cook, Leonid Levin

# Das Cook-Levin Theorem

## Konjunktive Normalform

- Boolesche Funktionen der Form

$$(a \vee b \vee c) \wedge (d \vee e) \wedge (f \vee g \vee h \vee i)$$

stehen in **konjunktiver Normalform**

- Lösungen stellen alle Belegungen dar, die zu 1 evaluieren
- Das Entscheidungsproblem, ob es eine Lösung gibt, ist als **SAT** (von „satisfiable“) bekannt
- Der Spezialfall, bei dem jede Teilklausel genau drei Variablen beinhaltet, heißt **3SAT**:

$$(a \vee b \vee c) \wedge (d \vee e \vee f) \wedge \dots$$

# Das Cook-Levin Theorem

## Boolsche Funktionen

- 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  $O(n \cdot 2^n)$

# Das Cook-Levin Theorem

Reduktion \* auf SAT

- $O(n \cdot 2^n)$  offensichtlich zu groß
- Sei  $\mathcal{M}$  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  $\mathcal{M}$
- Sei  $\Gamma$  das Bandalphabet von  $\mathcal{M}$
- Sei  $\langle a, b, q \rangle_i \in \Gamma \times \Gamma \times Q$  der Snapshot von  $\mathcal{M}$  in Schritt  $i$

# Das Cook-Levin Theorem

## 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:

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

# Das Cook-Levin Theorem

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!

# Das Cook-Levin Theorem

## Reduktion SAT auf 3SAT

- Wir möchten die Klausel  $F_1 = (a \vee b \vee c \vee d)$  mit Lösung  $L_1$  in 3SAT-Klauseln zerlegen
- Sei  $h$  eine neu eingeführte Hilfsvariable
- Dann ist  $F_2 = (a \vee b \vee h) \wedge (\bar{h} \vee c \vee d)$  mit Lösung  $L_2$  eine äquivalente Formel
- Erfüllbarkeit ist äquivalent:
  - $F_1 \Rightarrow F_2$ : 2 Fälle:
    - $a \vee b = 1$ :  $L_2$  mit  $h = 0$  erfüllt
    - $c \vee d = 1$ :  $L_2$  mit  $h = 1$  erfüllt
  - $F_2 \Rightarrow F_1$ : 2 Fälle:
    - $h = 0 \Rightarrow (a \vee b) = 1$
    - $h = 1 \Rightarrow (c \vee d) = 1$

# Das Cook-Levin Theorem

## Reduktion SAT auf 3SAT

- Allgemein gilt: 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:

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

- $h_1 \dots h_{n-2}$  sind hierbei neu eingeführte Hilfsvariablen
- SAT liegt in NP  $\Rightarrow$  3SAT liegt als Spezialfall von SAT in NP
- Laufzeit der Reduktion SAT  $\rightarrow$  3SAT ist polynomiell
- $\Rightarrow$  3SAT ist NP-vollständig

# 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



# Wichtige NP-vollständige Probleme

## INDSET

*Besitzt ein Graph  $G$  mindestens  $k$  paarweise nicht über eine Kante verbundene Knoten (=stabile Menge)?*

INDSET ist NP-vollständig. Hierzu definieren wir eine Transformation beliebiger 3SAT-Instanzen zu Graphen:

- Erzeuge für jede Klausel einen vollständig verbundenen Teilgraph (=Clique), dessen Knoten jeweils eine gültige Belegung repräsentieren.
- Verbinde alle nicht verbundenen Knoten, die gemeinsam zu einer widersprüchlichen Belegung führen würden.
- Bestimme nun eine stabile Menge der Größe  $k$ . Deren Knoten kodieren nun eine gültige Belegung für die 3SAT-Instanz.

# Wichtige NP-vollständige Probleme

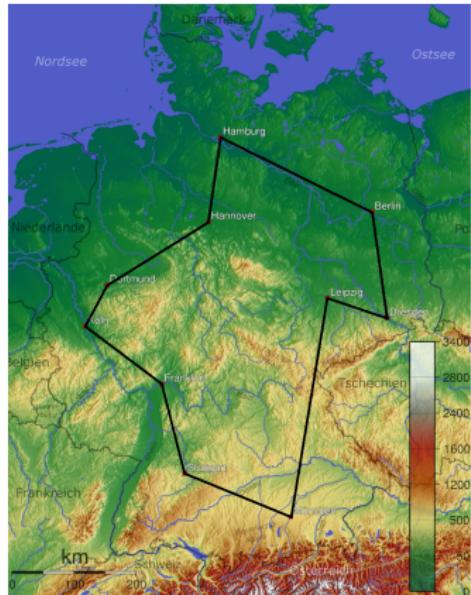
0/1 IPROG

- 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: SAT  $\preceq$  0/1 IPROG, da jede Klausel als Ungleichung aufgefasst werden kann
  - $u_1 \vee \overline{u_2} \vee \overline{u_3}$  kann ausgedrückt werden durch
$$u_1 + (1 - u_2) + (1 - u_3) \geq 1$$

# Wichtige NP-vollständige Probleme

## Traveling Salesman (TSP)

*Gibt es zu  $n$  Städten einen  
Rundweg der kürzer ist als  $b$ ?*



# Andere Klassen

## NP-Intermediate

- Gilt  $P \neq NP$  so gibt es eine Klasse NP-Intermediate (NPI) für die gilt:
  - $A \in NPI \Leftrightarrow A \in NP, A \notin P$  und  $A$  ist nicht NP-Schwer
- Wurde von Richard Ladner 1975 bewiesen
- Er konstruierte ein künstliches Problem welches unter der Annahme  $P \neq NP$  in NPI liegt
- Es ist nicht sicher ob es "natürliche" Probleme in NPI gibt

# Andere Klassen

## Vermutete Probleme in NPI

- Man vermutet, dass die Primfaktorzerlegung in NPI liegt
- Bisher noch kein Algorithmus in Polynomialzeit gefunden
- Noch kein Beweis für NP-Schwere
- Aktuelle Kryptographie baut darauf auf (RSA)

# Andere Klassen

## EXP und NEXP

- Probleme deren Zertifikat in exponentieller Zeit verifiziert bzw. gefunden werden kann.
- In vieler Hinsicht analog zu P und NP, aber in der Praxis weniger interessant.

# Andere Klassen

## Platzbasierte

- $L \subseteq NL$ 
  - logarithmischer Platz
- $PSPACE = NPSPACE$ 
  - polynomieller Platz
- $EXPSPACE = NEXPSPACE$ 
  - exponentieller Platz

Es gilt:

- $L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE$
- $NL \subset PSPACE$

# Andere Klassen

## Sonstige

- PP, BPP
  - Propabilistische Klassen
- BQP (Bounded Error Quantum Polynomial), QMA (Quantum Merlin Arthur), ...
  - Analoge Klassen für Quantencomputer

# Indizien

$P \neq NP$

- Unüberschaubar viele Probleme in P und NP. Trotz enormem Aufwand nicht eine einzige Reduktion.
- Reduktionen oft um ein  $\varepsilon$  nicht polynomiell. Warum, wenn  $P = NP$ ?
- Existenzbeweise meist leichter als Nichtexistenzbeweise. Deswegen schwer?
- $NL \subset PSPACE$ , eine der Untermengenrelationen dazwischen **muss** also echt sein.

# Indizien

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

- Bisher wurde noch für kein coNP-schweres Problem ein polynomielles Zertifikat gefunden
- Es konnte noch für kein NP-schweres Problem nachgewiesen werden, dass es in coNP liegt
- Es konnte noch für kein Problem aus  $\text{NP} \cap \text{coNP}$  NP-schwere bzw. coNP-schwere nachgewiesen werden

# Indizien

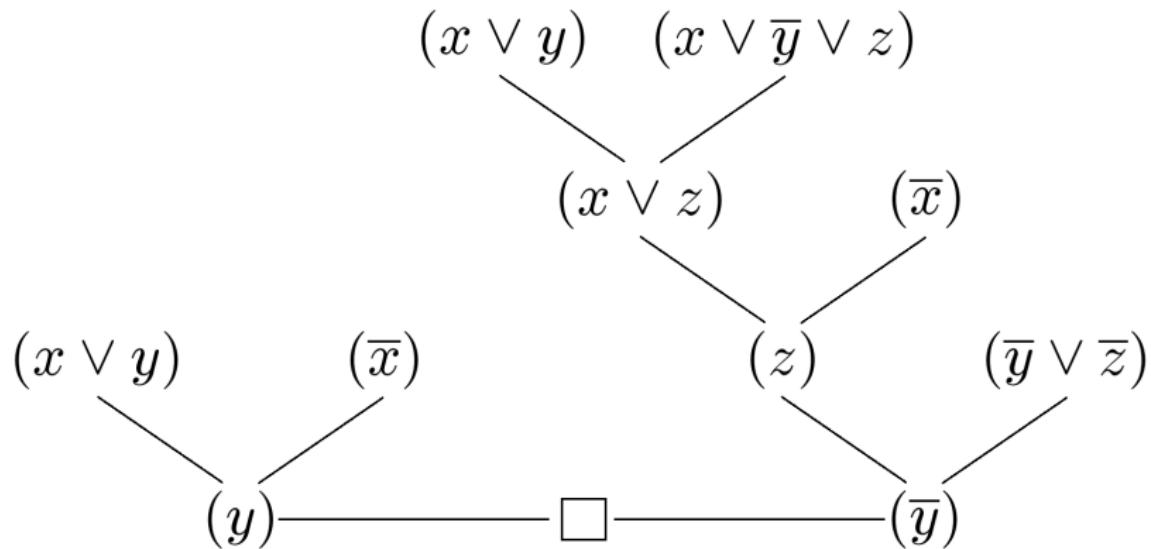
Resolution als Indiz für  $\text{coNP} \neq \text{NP}$

- Resolution prüft ob eine KNF eine Kontradiktion ist
- Wähle 2 Klauseln  $C_1, C_2$ , sodass ein Literal  $u$  in  $C_1$  und seine Negierung  $\bar{u}$  in  $C_2$  vorkommen
- Bilde eine neue Klausel  $C_3 = C_1 \setminus \{u\} \vee C_2 \setminus \{\bar{u}\}$
- $C_1 = (x \vee \bar{y} \vee z)$  und  $C_2 = (y \vee z)$  werden zu  $C_3 = (x \vee z)$
- Es gilt  $C_3 = \text{false} \Rightarrow C_1 \wedge C_2 = \text{false}$
- Kann man eine leere Klausel herleiten ist die Formel eine Kontradiktion

# Indizien

## Resolution Beispiel

$$F = (x \vee y) \wedge (x \vee \bar{y} \vee z) \wedge (\bar{x}) \wedge (\bar{y} \vee \bar{z})$$



# Indizien

Resolution als Indiz für  $\text{coNP} \neq \text{NP}$

- 1985 bewies Haken eine untere Schranke für die Größe des Resolutionsbeweises für das Taubenschlagprinzip
- Diese ist  $(1.49^{0.01})^n \approx (1,004)^n$  und unabhängig vom gewählten Algorithmus
- Aufbauend auf Hakens Beweis wurden exponentielle untere Schranken auch für andere Probleme gezeigt
  - mittels Resolution ist  $\text{coNP} = \text{NP}$  nicht beweisbar
  - ⇒ Indiz für  $\text{coNP} \neq \text{NP}$

# Implikationen

## Philosophisch

- In den Naturwissenschaften wären Hypothesen mit vergleichbarer Faktenlage als Theorien anerkannt.
  - „nur“ ein Problem weil Informatik eine Strukturwissenschaft ist
- Folgen oft völlig unintuitiv:
  - Warum sollte es keine Suchprobleme geben, die sich nicht besser als mit brute-force lösen lassen?
  - Insbesondere bei nichtdeterministischen TMs: Polyzeitreduktion praktisch nicht vorstellbar.
  - Alle Probleme in NP, die nicht auch in leichteren Klassen als P liegen, vergleichbar schwer, es gäbe keine Klasse NPI

# Implikationen

## Kryptographisch

- Nicht **zwingend** katastrophal

- Feste aber große Exponenten reichen vermutlich auch:  
 $2^{512} \ll 512^{100}$
- Heutige Krypto meist nicht NP-vollständig (Faktorisierung in NPI vermutet)
- Quantencomputer sind hier eine **viel** realere Bedrohung.  
(→ Shor-Algorithmus)

- Andererseits: Passwort raten, leicht gemacht?

- Gibt es ein Passphrase der Länge  $\leq n$  die diese Datei entschlüsselt?  $\in \text{NP}$

# Implikationen

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

- Aus  $P = NP$  folgt automatisch  $\text{coNP} = NP$ 
  - Umkehrung gilt nicht automatisch
- Allerdings folgt aus  $\text{coNP} \neq NP$  automatisch  $P \neq NP$ :
  - man wählt ein  $NP$ -vollständiges Problem  $L$
  - das Komplement  $\bar{L}$  von  $L$  liegt laut Definition in  $\text{coNP}$
  - würde  $P = NP$  gelten, gilt  $\bar{L} \in P$
  - da  $P$  unter Komplementbildung abgeschlossen ist gilt  $\bar{L} \in P$
  - da gilt  $P = NP$  gilt  $\bar{L} \in NP$
  - daraus würde folgen  $\text{coNP} = NP$  was ein Widerspruch ist

# Umgang mit NP-vollständigen Problemen



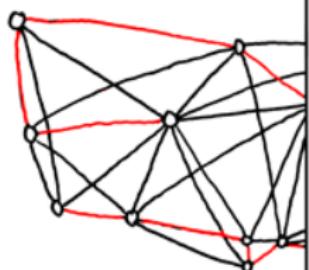
# Umgang mit NP-vollständigen Problemen

- Existieren vielleicht gute Näherungslösungen?
- Ist der Worst-Case wirklich wahrscheinlich?
- Ist  $n$  wirklich so groß, dass NP-Vollständigkeit ein Problem darstellt?

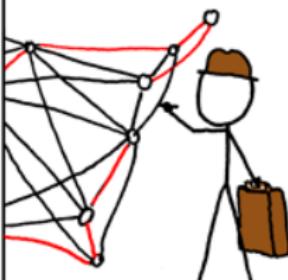
# Umgang mit NP-vollständigen Problemen

- Gibt es vielleicht bessere, nicht NP-schwere Modelierungen?

BRUTE-FORCE  
SOLUTION:  
 $O(n!)$



DYNAMIC  
PROGRAMMING  
ALGORITHMS:  
 $O(n^2 2^n)$



SELLING ON EBAY:  
 $O(1)$

STILL WORKING  
ON YOUR ROUTE?

SHUT THE  
HELL UP.



# Zusammenfassung

- Alle NP-Probleme können in Polyzeit auf NP-vollständigen Probleme reduziert werden.
- Wichtige Beispiele für NP-vollständige Probleme sind SAT, 3SAT, INDSET, 0/1-PROG und TSP
- Analoge Probleme existieren auch in diversen anderen Klassen
- $P \neq NP$  ist zwar *unbewiesen* aber es gibt sehr gute *Indizien* dafür
- Die Implikationen von  $P = NP$  sind gravierend, aber nicht zwingend katastrophal
- In der Realität sind NP-vollständige Probleme bisweilen gut zu meistern

# Quellen

## Literatur

- „Aussagenlogik: Deduktion und Algorithmen“ von Hans Kleine Büning und Theodor Lettmann
- „Computational Complexity: A Modern Approach“ von Sanjeev Arora und Boaz Barak

- <http://de.wikipedia.org/wiki/>
  - AKS-Primzahltest
  - Satz\_von\_Ladner
  - NP-intermediate
- <https://complexityzoo.uwaterloo.ca/>
  - Complexity\_Zoo:B#bqp
  - Complexity\_Zoo:B#bpp
  - Complexity\_Zoo:C#conp
  - Complexity\_Zoo:P#p
  - Complexity\_Zoo:Q#qma

- https://homepages.uni-tuebingen.de/~student/monika.gehweiler/Applets/html/resolutionIndex.html
- http://www.ti.inf.ethz.ch/ew/lehre/extremal04/raemy.pdf
- http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/46927-f97/slides/Lec4/sld033.htm
- http://www.cosy.sbg.ac.at/~held/teaching/wiss\_arbeiten/slides\_02-03/kmrr.pdf
- http://hackvalue.de/files/tsp.pdf
- http://www.scottaaronson.com/blog/?p=1720

- Motivations-Comics: „Computers and Intractability – A Guide to the Theory of NP-Completeness“ von Michael R. Garey und David S. Johnson
- Lego-Turingmaschine: <http://www.legoturingmachine.org>
- Stephen Cook (CC-BY-SA 3.0, Jiří Janíček): <https://commons.wikimedia.org/wiki/File:Prof.Cook.jpg>
- Leonid Levin (CC-BY-SA 3.0, Sergio01):  
<https://commons.wikimedia.org/wiki/File:LeonidLevin2010.jpg>

- TSP-Karte (CC-BY-SA 3.0):  
[http://commons.wikimedia.org/wiki/File:Deutschland\\_topo.png](http://commons.wikimedia.org/wiki/File:Deutschland_topo.png)
- „My Hobby“ (CC-BY-NC 2.5, Randall Munroe):  
<https://xkcd.com/287/>
- „Don't Panic“: <http://everfalling.deviantart.com/art/DON-T-PANIC-15975789>
- „Traveling Salesman“ (CC-BY-NC 2.5, Randall Munroe):  
<https://xkcd.com/399/>