Das P-NP-Problem und die Grenzen der praktischen Berechenbarkeit

Kantonale Fachschaftstagung Informatik, Mathematik und Physik

Thomas Strahm
Institut für Informatik und angewandte Mathematik
Universität Bern

4. November 2010

P-NP: Die 1 Mio $ Frage

- Die Clay Mathematics Institute Millenium Prize Probleme
  - Birch and Swinnerton-Dyer Conjecture
  - Hodge Conjecture
  - Navier-Stokes Equations
  - **P vs NP**
  - Poincaré Conjecture
  - Riemann Hypothesis
  - Yang-Mills Theory

“P versus NP – a gift to mathematics from computer science”. Steve Smale

In diesem Vortrag

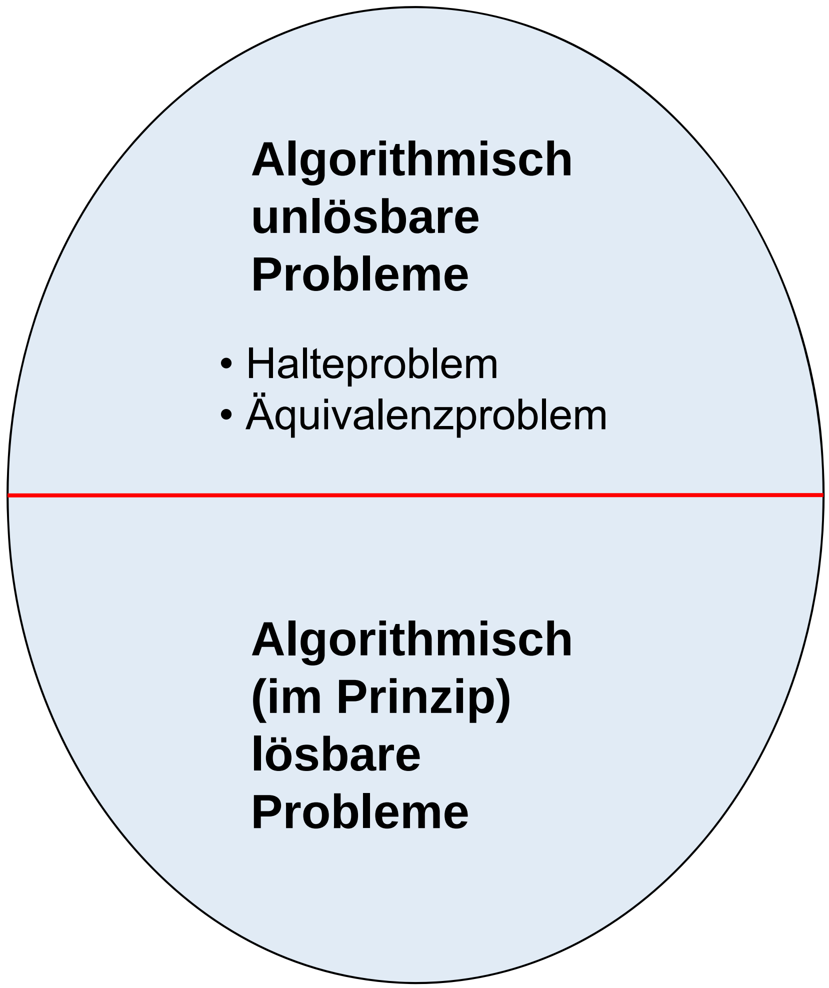
- Einleitung
- Informelle Beschreibung des P-NP-Problems
- Mathematische Präzisierung der P-NP-Frage
- NP-Vollständigkeit
- Einige Ansätze zur Lösung der P-NP-Frage

Algorithmus

UNIVERSITÄT BERN

- Ein **Algorithmus** ist eine endliche Beschreibung eines mechanischen Verfahrens zur Lösung eines Problems
- **Informeller, intuitiver Berechenbarkeitsbegriff**
- Beispiele: arithmetische Rechenoperationen, Euklid, n-te Primzahl etc.
- Begriff geht zurück auf den persisch-arabischen Mathematiker **Al-Khowarizmi (ca. 780-850)**
- „Algorithmics – the spirit of computing“ (David Harel)

Algorithmisch unlösbare Probleme



- Church
- Gödel
- Kleene
- Turing

Komplexitätstheorie

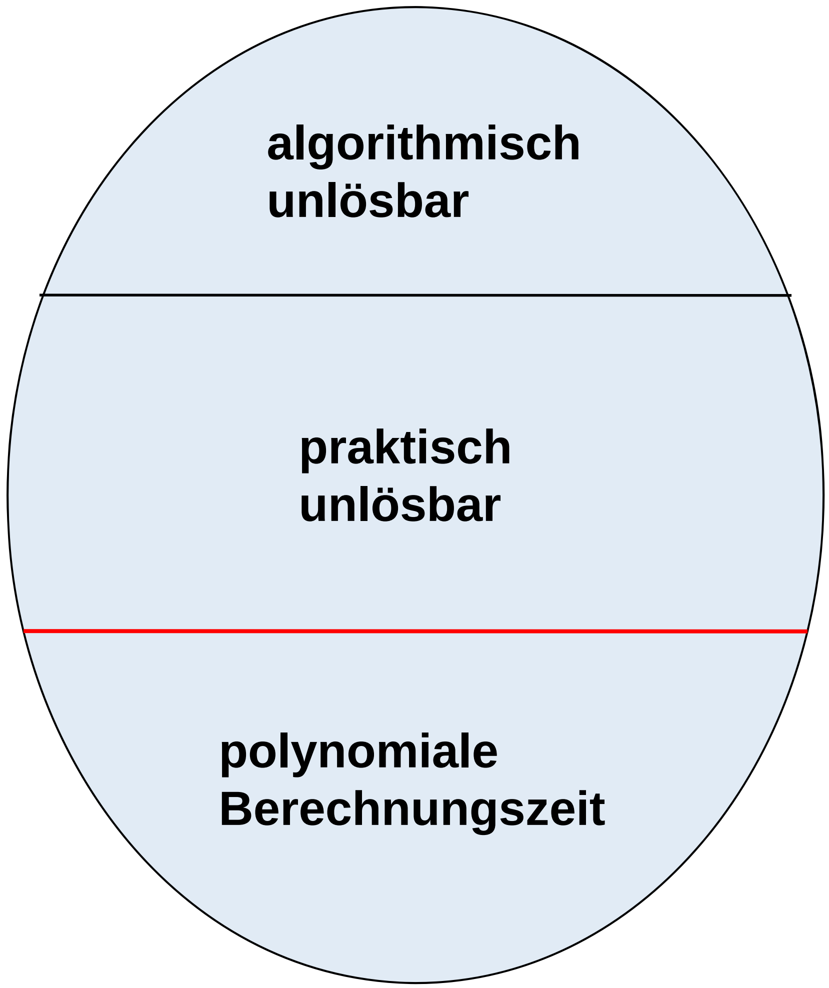
UNIVERSITÄT BERN

- **Im Prinzip algorithmisch lösbare Probleme sind i.d.R. nicht praktisch lösbar**
- Klassifikation von entscheidbaren algorithmischen Problemen nach ihrer Komplexität (Rechenzeit, Speicherplatz)
- Welches ist die Grenze zwischen praktisch lösbaren und unlösbaren Problemen?
- **Polynomiale versus exponentielle Rechenzeit**
- Viele Probleme, die (vermutlich) keine effiziente Lösung besitzen, haben einen **polynomialen Verifikations-algorithmus**, d.h., mögliche Lösungen können effizient auf ihre Korrektheit überprüft werden

Polynomial versus exponentiell

	20	40	60	100	300
n	10^{-8} sec	10^{-8} sec	10^{-7} sec	10^{-7} sec	10^{-7} sec
n^2	10^{-7} sec	10^{-6} sec	10^{-6} sec	10^{-5} sec	10^{-4} sec
n^3	10^{-5} sec	10^{-4} sec	10^{-4} sec	10^{-3} sec	10^{-2} sec
2^n	10^{-3} sec	19 min	37 J	10^{13} J	10^{73} J

Praktisch unlösbare Probleme



In diesem Vortrag

UNIVERSITÄT BERN

- Einleitung
- **Informelle Beschreibung des P-NP-Problems**
- Mathematische Präzisierung der P-NP-Frage
- NP-Vollständigkeit
- Einige Ansätze zur Lösung der P-NP-Frage

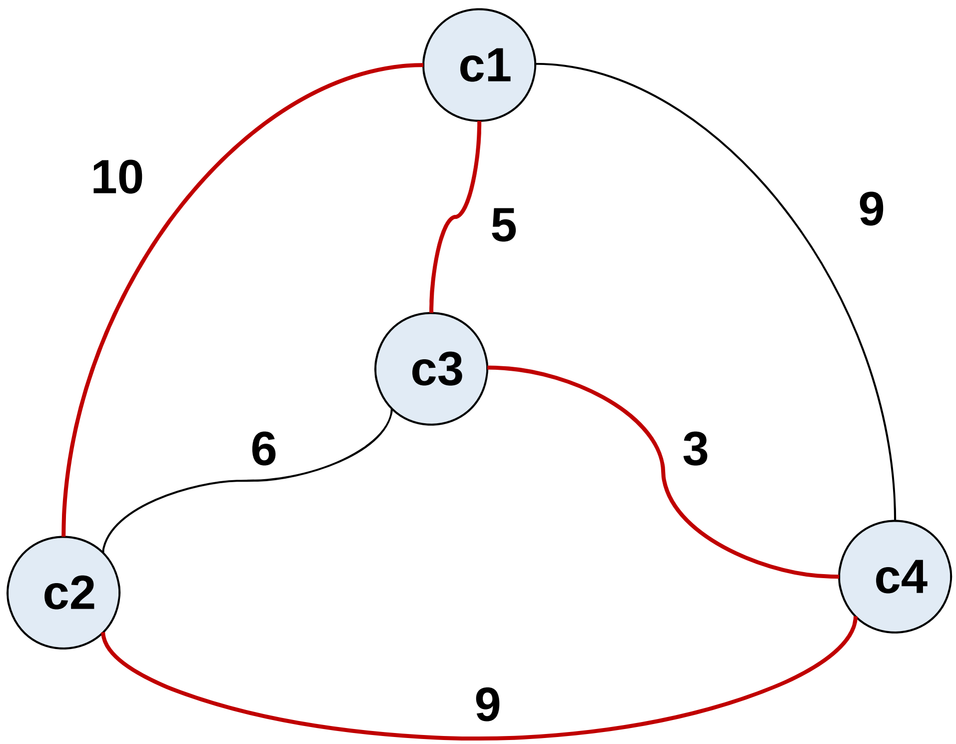
Das Travelling Salesperson Problem TSP

- **Gegeben:**
  - — Eine Menge $C = \{c1, c2, ..., cn\}$ von Städten und eine „Distanz“ $d(ci, cj)$ für alle ci, cj aus C; eine natürliche Zahl B („Budget“)

- **Frage:**
  - — Gibt es eine Tour der Städte in C, deren Länge durch B beschränkt ist ?

TSP Beispiel

- Im folgenden Beispiel mit 4 Städten hat die minimale Rundreise die Länge 27:



Charakteristiken von TSP

> Der naive Algorithmus, welcher TSP entscheidet, testet alle möglchen Touren von n Städten durch. Es gibt n! viele Touren und $n! \geq 2^n$ für $n \geq 4$.
> Von einer möglichen Tour kann in polynomialer Zeit überprüft werden, ob sie eine Lösung des TSP ist.
> Es ist nicht bekannt, ob es einen polynomialen Algorithmus für TSP gibt.
> Vermutung: NEIN

Erfüllbarkeitsproblem SAT

- **Gegeben:**
  - — Boolscher Ausdruck Φ in den Variablen x1, x2, ..., xn
- **Frage:**
  - — Ist Φ erfüllbar, d.h. gibt es eine Variablenbelegung β aus $\{0,1\}^n$, so dass Φ unter β zu 1 auswertet?

Boolesche Ausdrücke

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

x	$\neg x$
0	1
1	0

Charakteristiken von SAT

- Der naive Algorithmus, welcher SAT entscheidet, testet alle möglchen Belegungen der Variablen x1, ..., xn durch. Es gibt 2^n Belegungen, also ist der Algorithmus exponentiell in der Anzahl der Variablen
- Von einer festen Belegung der Variablen kann in polynomialer Zeit entschieden werden, ob sie einen gegebenen Booleschen Ausdruck erfüllt
- Es ist nicht bekannt, ob es einen polynomialen Algorithmus für SAT gibt.
- Vermutung: NEIN

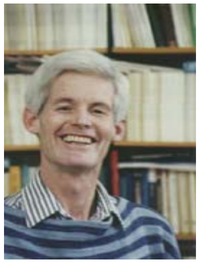
Binpacking Problem BP

> **Gegeben:**
> — Eine endliche Menge U von Gegenständen der Grösse s(u) aus **N** für jedes u aus U; eine Kübelkapazität B und eine natürliche Zahl K (Anzahl Kübel)

> **Frage:**
> — Kann man U in K Kübel der Grösse B verpacken, so dass keiner der Kübel überläuft?

Die Komplexitätsklassen P und NP

- **P**: Klasse aller Entscheidungsprobleme, welche **durch einen polynomialen Algorithmus entschieden** werden können
- **NP**: Klasse aller Entscheidungsprobleme, für welche **in polynomialer Zeit *verifiziert* werden** kann, ob eine mögliche Lösung korrekt ist
  - These von Cook/Levin (1971): (bis dato unbewiesen)




$$P \neq NP$$

In diesem Vortrag

UNIVERSITÄT BERN

- Einleitung
- Informelle Beschreibung des P-NP-Problems
- **Mathematische Präzisierung der P-NP-Frage**
- NP-Vollständigkeit
- Einige Ansätze zur Lösung der P-NP-Frage

Formalisierung des Algorithmusbegriffs

UNIVERSITÄT BERN

> Codierung von algorithmischen Problemen und Berechnungen in endlichen Alphabeten, z.B.
> $\Sigma = \{a, b, c, \ldots, z\}$ oder $\Sigma = \{0, 1\}$
> Σ^* : endliche Wörter über Σ

> Welche Funktionen f von Σ^* nach Σ^* sind algorithmisch berechenbar ?

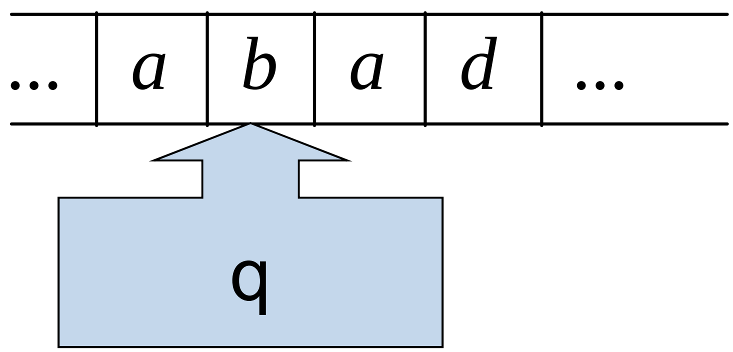
> Welche Teilmengen L von Σ^* sind algorithmisch entscheidbar ?

> Wie werden Komplexitätsmasse (Zeit, Platz) formalisiert ?

Turingmaschinen (Turing, 1936)

UNIVERSITÄT BERN

- Eine **Turingmaschine M** besteht aus
  - Q endliche Menge von Zuständen
  - Γ endliches Alphabet
  - δ: Q x Γ $\longrightarrow$ Q x Γ x {L, R, N} : Überführungsfunktion
  - q_0 Anfangszustand
  - q_{accept}, q_{reject}: akzeptierender / verwerfender Endzustand



Formale Definition von NP

> Ein Entscheidungsproblem ***L*** ist in **NP**, falls es ein Problem $\boldsymbol{L_0}$ in **P** und ein Polynom **p(x)** gibt, so dass

$$L = \{x \mid \exists y \; |y| \le p(|x|) : \; (x,y) \in L_0\}$$

Polynomialer Zeuge, Beweis

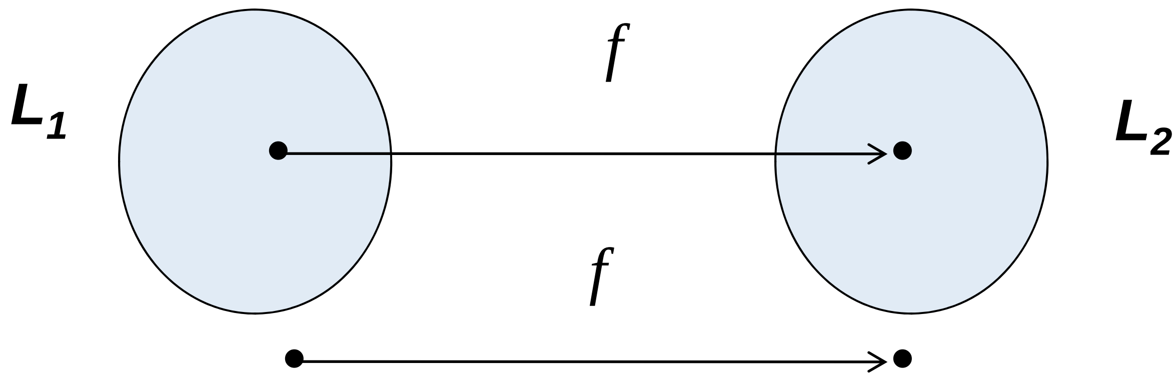
In diesem Vortrag

- Einleitung
- Informelle Beschreibung des P-NP-Problems
- Mathematische Präzisierung der P-NP-Frage
- **NP-Vollständigkeit**
- Einige Ansätze zur Lösung der P-NP-Frage

NP-Vollständigkeit

- Identifikation der schwierigsten Probleme in **NP**
- Begriff der **polynomialen Reduktion**: ein Problem L_1 heisst *polynomial reduzierbar* auf L_2, falls es eine in polynomialer Zeit berechenbare Funktion f gibt, so dass gilt:

$$\forall x \ (x \in L_1 \Leftrightarrow f(x) \in L_2)$$



NP-Vollständigkeit (ff.)

> Ein Problem ***L*** heisst **NP-vollständig**, falls
  — ***L*** gehört zu **NP**
  — Jedes beliebige L_1 aus **NP** lässt sich polynomial auf ***L*** reduzieren

Sei ***L*** **NP**-vollständig. Dann gilt: **P** ist verschieden von **NP** genau dann, wenn ***L*** nicht zu **P** gehört.

Erste NP-vollständige Probleme

Theorem (Cook, Karp, Levin)

TSP, SAT und BP sind **NP**-vollständig

Heute kennt man tausende von NP-vollständigen Probleme, z.B. aus den folgenden Bereichen:

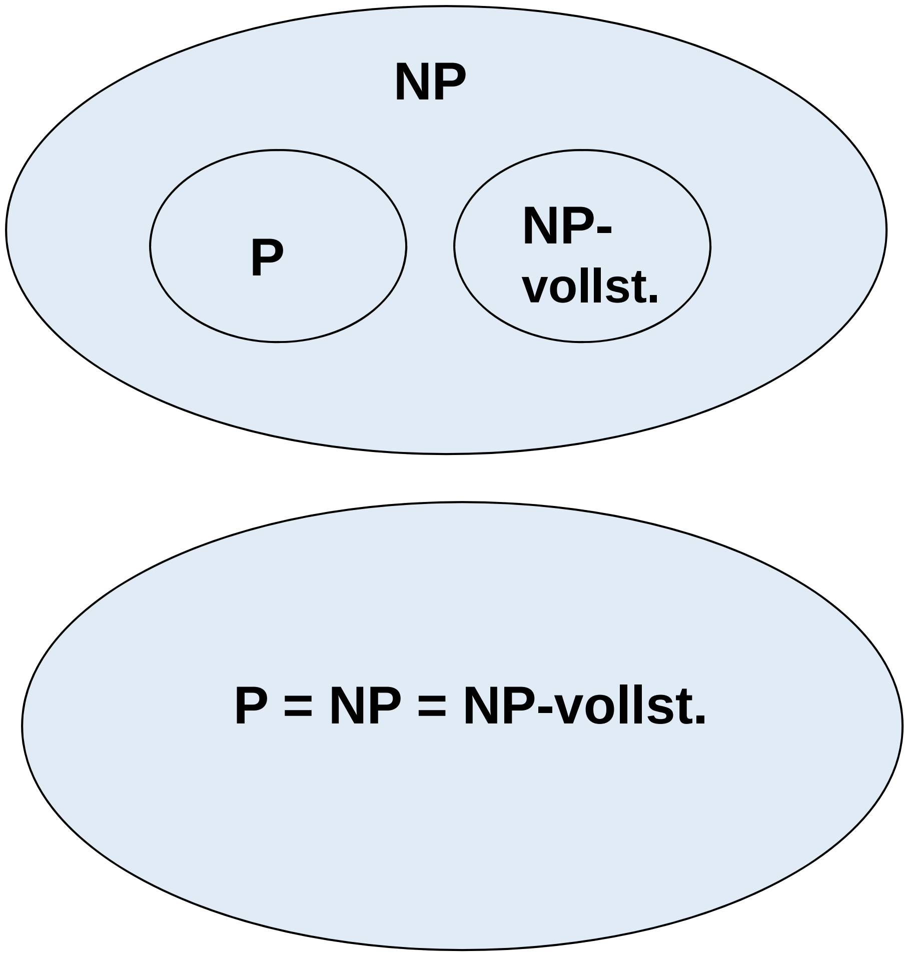
NP-vollständige Probleme

> Graphentheorie
> Netzwerkdesign
> Mengentheorie
> Datenbanken
> Scheduling
> Zahlentheorie
> Logik
> Automatentheorie
> etc.

Ein Klassiker

> **M. Garey, D. Johnson**, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979

Zwei mögliche Welten



In diesem Vortrag

UNIVERSITÄT BERN

- Einleitung
- Informelle Beschreibung des P-NP-Problems
- Mathematische Präzisierung der P-NP-Frage
- NP-Vollständigkeit
- **Einige Ansätze zur Lösung der P-NP-Frage**

Diagonalisierung ?

> Kann die Diagonalisierungstechnik helfen, **P** und **NP** zu trennen ?
> Dies ist sehr unwahrscheinlich, da die **P-NP**-Frage abhängig ist von sogenannten Orakeln (Baker, Gill, Solovay)
> Es gibt ein Orakel **A**, das **P** und **NP** identifiziert
> Es gibt ein Orakel **B**, das **P** und **NP** trennt

Logische Methoden zur Trennung von Komplexitätsklassen ?

- Nach dem **Gödelschen ersten Unvollständigkeitssatz** wissen wir, dass es in jedem widerspruchsfreien Axiomensystem für die elementare Arithmetik Aussagen gibt, welche **wahr aber nicht beweisbar** sind
- Typische **„Unvollständigkeitsaussagen“** betreffen die Totalität von Funktionen, bzw. die Terminierung von Algorithmen
  - Klassifikation von Axiomensystemen nach ihren **beweisbar terminierenden Algorithmen**

Beweisbarkeit und Komplexität

> Sei **F** eine berechenbare Funktion und bezeichne $\mathbf{Gr_F}$ ihren Graphen. **F** heisse repräsentierbar im Axiomensystem **Ax**, falls

$$\mathbf{Ax} \quad \text{beweist} \quad \forall x\, \exists y\, \mathbf{Gr_F(x,y)}$$

> Seien $\mathbf{C_1}$ und $\mathbf{C_2}$ Klassen von Funktionen einer bestimmten Komplexität; ferner charakterisiere $\mathbf{Ax_1}$ die Klasse $\mathbf{C_1}$ und $\mathbf{Ax_2}$ die Klasse $\mathbf{C_2}$

> Fragen:

— Können $\mathbf{Ax_1}$ und $\mathbf{Ax_2}$ getrennt werden ?

— Hilft eine solche Trennung für einen Beweis, dass $\mathbf{C_1}$ und $\mathbf{C_2}$ verscheiden sind ?

Aussagenlogische Beweissysteme

- **Frage:** Gibt es ein **polynomiales aussagenlogisches Beweissystem**, d.h. einen Kalkül für die Aussagenlogik, in dem jede Tautologie einen Beweis polynomialer Grösse hat?
- Es gibt ein polynomiales Beweissystem genau dann, wenn NP = co-NP (Komplemente der Sprachen in NP)
- Aus NP $\neq$ co-NP folgt P$\neq$ NP!
- Studium der Komplexität von speziellen und allgemeinen aussagenlogischen Beweissystemen
- Untere Schranken
- Tiefe Zusammenhänge zur sog. beschränkten Arithmetik

Deskriptive Komplexitätstheorie

- Welches ist die **logische Komplexität**, eine bestimmte Eigenschaft auszudrücken?
- Charakterisierung von zahlreichen Komplexitätsklassen in verschiedenen Sprachen/Dialekten der Logik
- Ziel: Einsatz logischer/semantischer Methoden zur Trennung von Komplexitätsklassen
- **Endliche Modelltheorie**

Jack Edmonds

b UNIVERSITÄT BERN

Jack Edmonds (1966)

„The classes of problems which are respectively known and not known to have good algorithms are of great theoretical interest [...]. I conjecture that there is no good algorithm for the travelling salesman problem. My reasons are the same as for any mathematical conjecture:
(1) It is a legitimate mathematical possibility, and
(2) I do not know “