UNIVERSITÄT
BERN

Komplexität

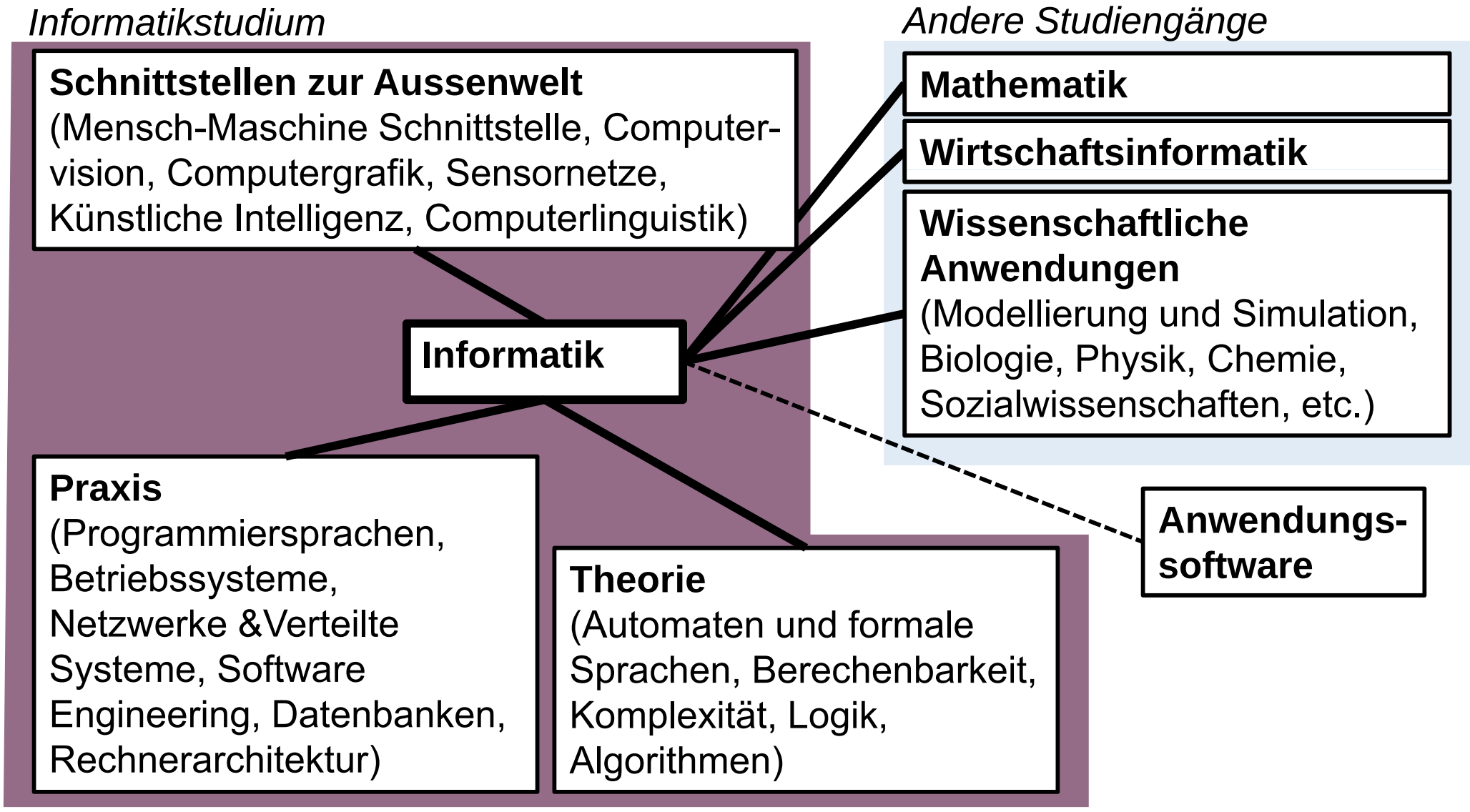
Einführung in die Informatik

Herbstsemester 2009

Thomas Strahm
Institut für Informatik und angewandte Mathematik
Universität Bern

November 2009

Übersicht


Informatikstudium
Andere Studiengänge
Schnittstellen zur Aussenwelt
(Mensch-Maschine Schnittstelle, Computer-vision, Computergrafik, Sensornetze, Künstliche Intelligenz, Computerlinguistik)
Mathematik
Wirtschaftsinformatik
Wissenschaftliche Anwendungen
(Modellierung und Simulation, Biologie, Physik, Chemie, Sozialwissenschaften, etc.)
Informatik
Praxis
(Programmiersprachen, Betriebssysteme, Netzwerke &Verteilte Systeme, Software Engineering, Datenbanken, Rechnerarchitektur)
Theorie
(Automaten und formale Sprachen, Berechenbarkeit, Komplexität, Logik, Algorithmen)
Anwendungs-software

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)
- In vielen Fällen liefert die Turingmaschine auch hier das geeignete formale Modell
- Welches ist die Grenze zwischen praktisch lösbaren und unlösbaren Problemen?

Komplexitätstheorie

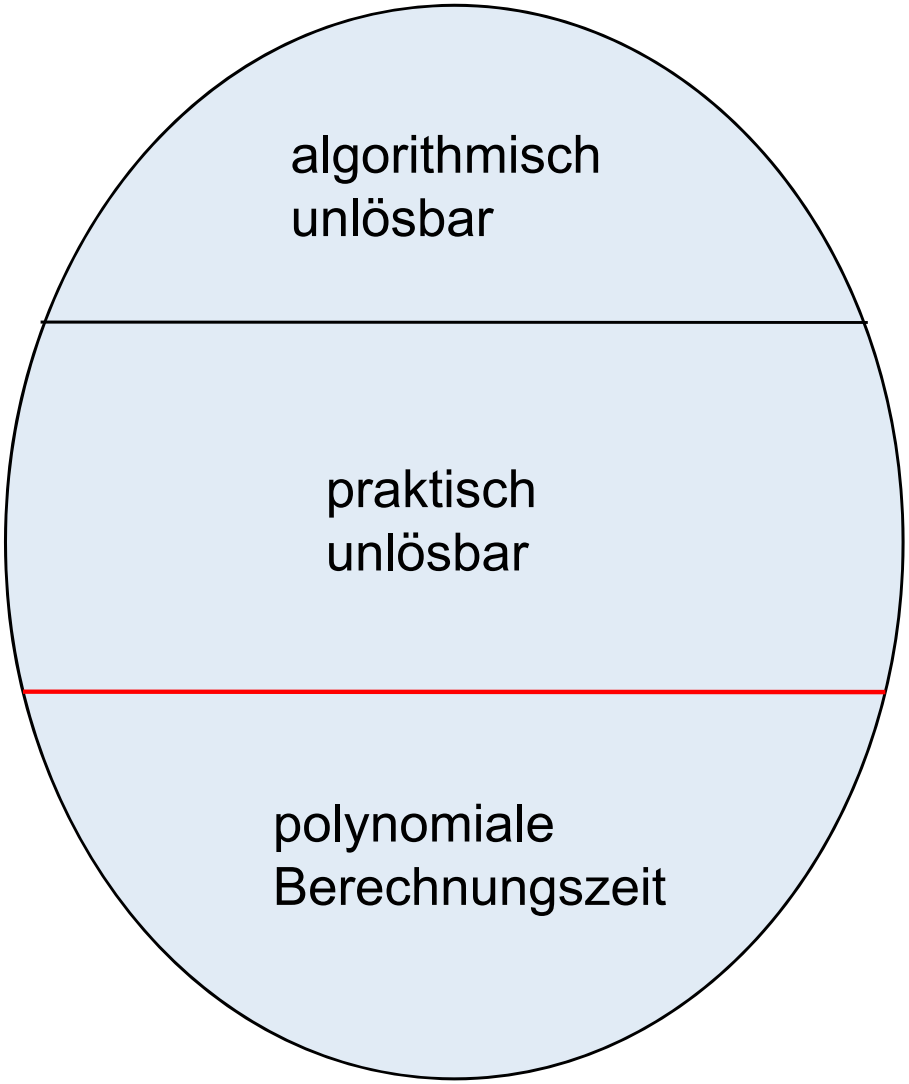
UNIVERSITÄT BERN

- **Polynomiale versus exponentielle Rechnenzeit**
- Polynomiale Entscheidbarkeit versus polynomiale Verifizierbarkeit
- Viele Probleme, die (vermutlich) keine effiziente Lösung besitzen, haben einen **polynomialen Verifikations-algorithmus**, d.h., mögliche Lösungen können effizient auf ihre Korrektheit überprüft werden
- Ein 1‘000‘000 $ Problem: **Die P=NP Frage**; NP-Vollständigkeit

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



Relative Primalität RELPRIM

UNIVERSITÄT BERN

- **Gegeben:**
  - Zwei natürliche Zahlen x und y
- **Gefragt:**
  - Sind x und y relativ prim, d.h., ist 1 die grösste Zahl, die sowohl x wie auch y teilt ?

- Beispiel: 10 und 21 sind relativ prim

- Der naive Algorithmus, welcher dieses Problem entscheidet, ist exponentiell in der Länge der Eingabe (wieso?)
- Bessere (polynomiale) Lösung: der Algorithmus von Euklid

Der Algorithmus von Euklid

UNIVERSITÄT BERN

- Berechnung des **grössten gemeinsamen Teilers (ggT)** zweier Zahlen x und y
- Gegeben sei der Input (x,y) von zwei natürlichen Zahlen
- Wiederhole bis y=0:
  - x := x mod y
  - Vertausche x und y
- Output x
- Dieser Algorithmus ist polynomial und damit ist auch **RELPRIM in polynomialer Zeit entscheidbar**

Das Travelling Salesperson Problem TSP

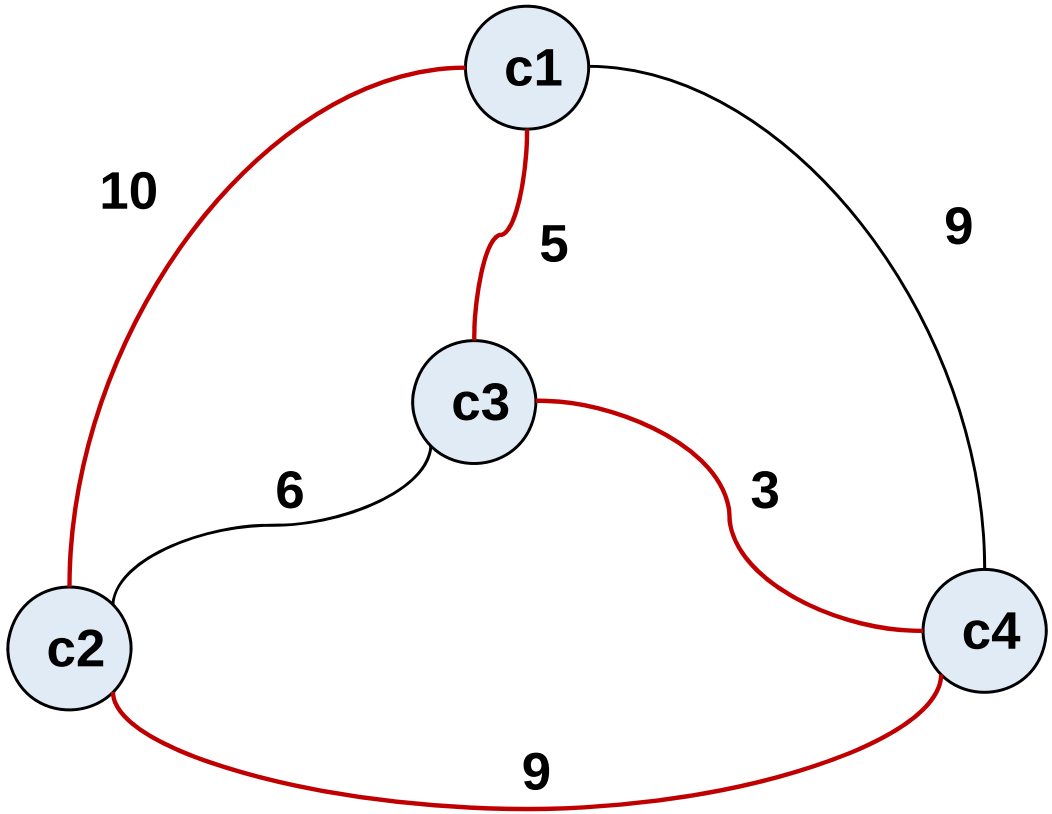
UNIVERSITÄT BERN

- **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:

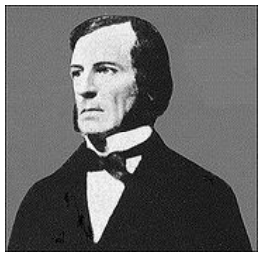


Charakteristiken von TSP

UNIVERSITÄT BERN

- 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

Exkurs/Repetition Aussagenlogik I



UNIVERSITÄT BERN

- Die Aussagenlogik fusst auf einer Menge von Boolschen Variablen x, y, z, ... welche einen Wahrheitswert wahr (1) oder falsch (0) annehmen können
- Boolsche oder aussagenlogische Formeln (Ausdrücke) werden mittels Variablen und den Boolschen Operatoren $\wedge$ (AND), $\vee$ (OR) und $\neg$ (NOT) gebildet.
- Beispiel eines Boolschen Ausdrucks:

$$(\neg x \wedge y) \vee (x \wedge \neg z)$$

- Ein Boolscher Ausdruck repräsentiert eine Boolesche Funktion (in seinen Variablen)

Exkurs/Repetition Aussagenlogik II

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

Exkurs/Repetition Aussagenlogik III

UNIVERSITÄT BERN

- Ein Boolscher Ausdruck Φ in den Variablen x1, x2, ..., xn heisst **erfüllbar**, falls es eine Belegung von x1, x2, ..., xn mit 0 und 1 gibt, unter der der Ausdruck zu 1 auswertet.
- Der Boolsche Ausdruck

$$(\neg x \wedge y) \vee (x \wedge \neg z)$$

ist erfüllbar durch die Belegung

$$x = 0,\ y = 1,\ z = 0$$

Erfüllbarkeitsproblem SAT

- **Gegeben:**
  - Boolscher Ausdruck Φ in den Variablen x1, x2, ..., xn
- **Frage:**
  - Ist Φ erfüllbar ?

Charakteristiken von SAT

UNIVERSITÄT BERN

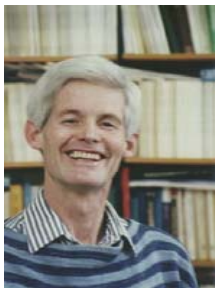
- Der naive Algorithmus, welcher SAT entscheidet, testet alle möglchen Belegungen der Variablen x1, ..., xn durch. Es gibt 2^n Belegungen, also ist der Algorithmus exponentiell in der Anzahl der Variablen
- Von einer festen Belegung der Variablen kann in polynomialer Zeit entschieden werden, ob sie einen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

Die Komplexitätsklassen P und NP

UNIVERSITÄT BERN

- **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):

P ≠ NP



Polynomiale Veriifiziertbarkeit

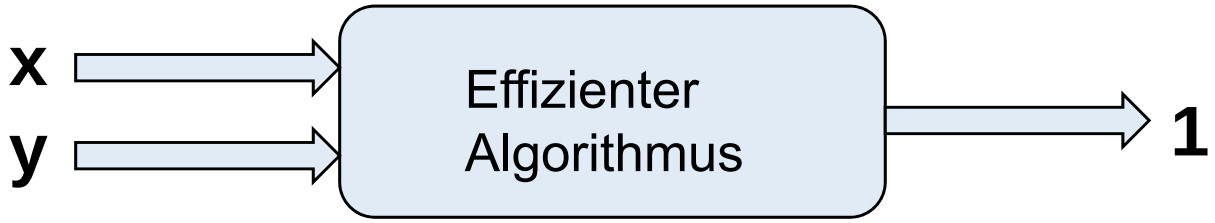
UNIVERSITÄT BERN

- Eine Sprache ***L*** liegt in **NP**, falls es einen effizienten (d.h. polynomialen) Algorithmus gibt, der mit zwei Eingaben arbeitet, so dass die folgenden zwei Bedingungen erfüllt sind:

  - Wenn ***x in L***, dann gibt es eine zweite Eingabe y (ein Wort, dessen Länge polynomial beschränkt in der Länge von x sein muss), so dass dieser effiziente Algorithmus (bei Eingabe von x und y) **1** ausgibt.
  - Wenn ***x nicht in L***, dann wird bei Eingabe von x und jeder beliebigen zweiten Eingabe y immer **0** ausgegeben.

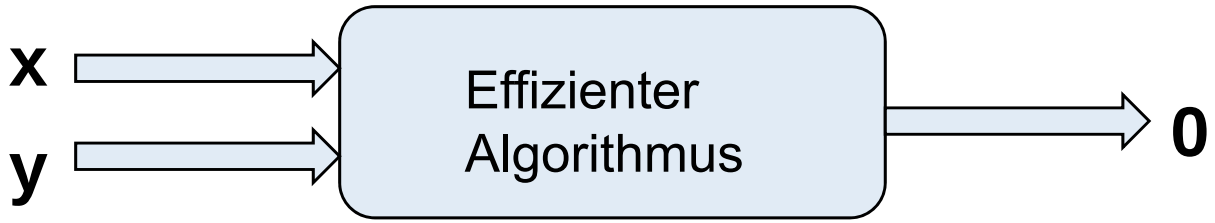
Polynomiale Verifizierbarkeit ff.

Falls x in L



(für mindestens ein y)

Falls x nicht in L

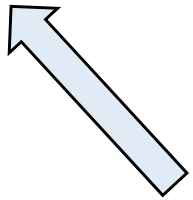


(für alle y)

Formale Definition von NP

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

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

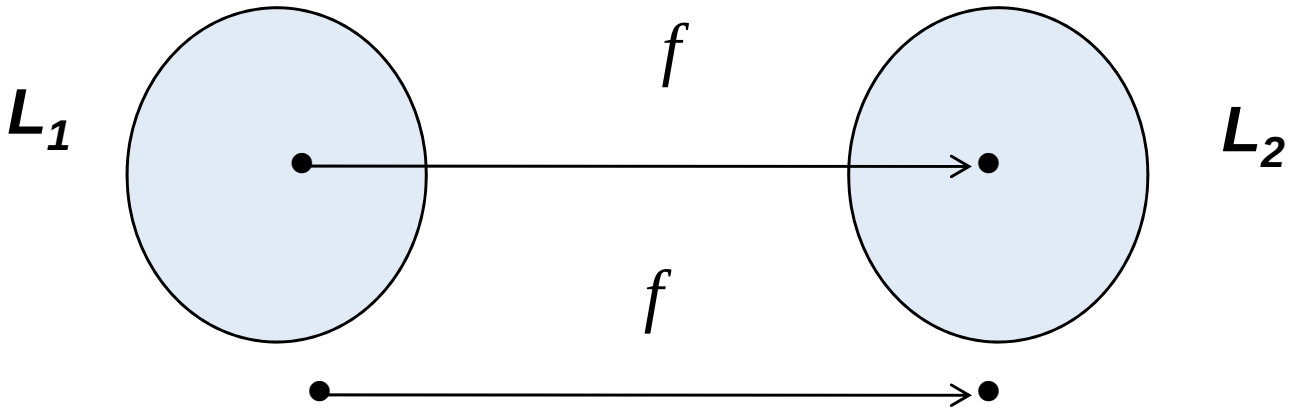


Polynomialer Zeuge, Beweis

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

UNIVERSITÄT BERN

Theorem (Cook, Karp, Levin)

TSP und SAT sind **NP**-vollständig

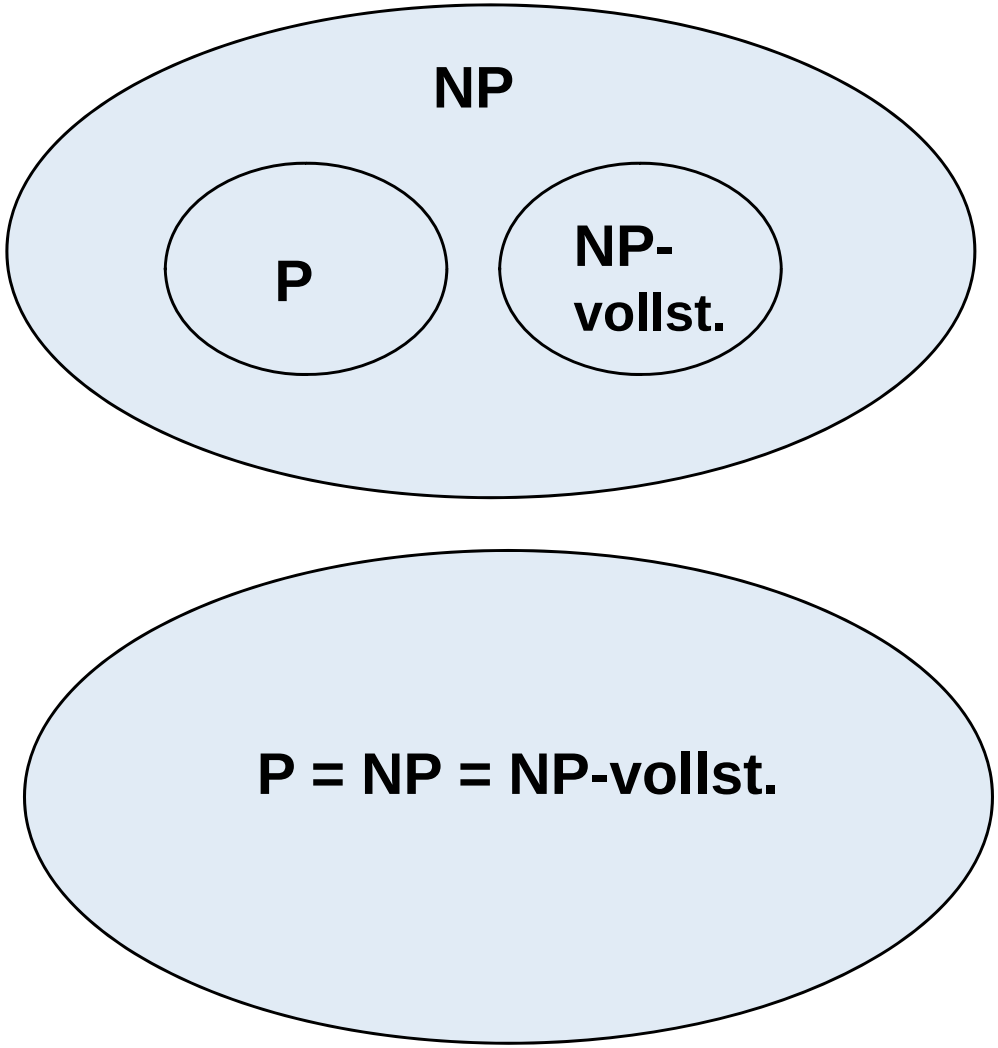
Heute kennt man hunderte von NP-vollständigen Probleme, z.B. aus den folgenden Bereichen:

NP-vollständige Probleme

UNIVERSITÄT BERN

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

Zwei mögliche Welten



Jack Edmonds

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 “

Zusätzlicher Literaturhinweis

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