UNIVERSITÄT
BERN

Einige Ideen und Probleme der theoretischen Informatik

Kolloquium “Informatik und Unterricht”

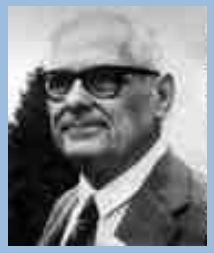
Thomas Strahm
Institut für Informatik und angewandte Mathematik
Universität Bern

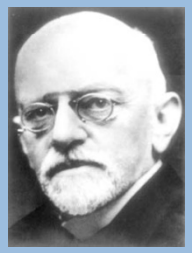
14. Januar 2008

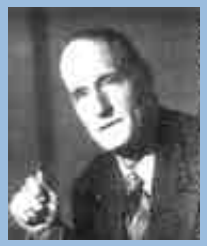
Ideen und Probleme der theoretischen Informatik

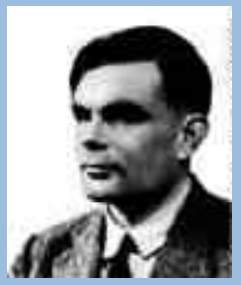
- Die klassischen Kerngebiete der theoretischen Informatik: **Berechenbarkeits- und Komplexitätstheorie**
- Theoretische und praktische **Grenzen der algorithmischen Lösbarkeit von Problemen**
- Zugang zur Informatik über formale Modelle und Konzepte erlaubt eine ästhetische Sichtweise der Informatik
- Informatik als Wissenschaft mit prinzipiellen Fragen, die **unabhängig von Technologie** gestellt werden können
- Theoretische Informatik untrennbar von ihrer Geschichte
- „Theory is good for you because it expands your mind“ (Michael Sipser)
- Befruchtung auch für den gymnasialen Unterricht

Einige Väter der theoretischen Informatik









Inhaltsübersicht

UNIVERSITÄT BERN

- **Berechenbarkeitstheorie**
  - Historische Wurzeln
  - Die Formalisierung des Algorithmusbegriffs
  - Die These von Church
  - Unentscheidbarkeit
  - Das Theorem von Rice
- **Komplexitätstheorie**
  - Polynomiale Entscheidbarkeit versus polynomiale Verfizierbarkeit: die Komplexitätsklassen P und NP
  - Die P-NP-Frage und NP-Vollständigkeit
  - Ein logischer Ansatz zu Fragen der Komplexität

Berechenbarkeit und Komplexität

UNIVERSITÄT BERN

- **Berechenbarkeit:**
  Welches sind die Grenzen der theoretischen und prinzipiellen Berechenbarkeit und Algorithmisierbarkreit ?

- **Komplexität:**
  Welches sind die Grenzen der praktischen Berechenbarkeit ?

- In diesem Vortrag sollen diese beiden Kernthemen der Informatik anhand von Modellen, Konzepten und Beispielen diskutiert werden

Berechenbarkeitstheorie

- Was ist eine berechenbare Funktion bzw. ein algorithmisch lösbares Problem ?
- Wie kann man das informelle Konzept des Algorithmus formalisieren ?
- Gibt es nicht-berechenbare Funktionen ? Falls ja, wo liegt die Grenze zwischen automatisch Lösbarem und automatisch Unlösbarem ?
- Gibt es Problemstellungen, die wohldefiniert, aber algorithmisch nicht lösbar sind ?
- Was können Computer überhaupt berechnen ?

Komplexitätstheorie

UNIVERSITÄT BERN

- Analyse von Algorithmen und algorithmisch lösbaren Problemen hinsichtlich ihres Rechenzeit- und Speicherplatzbedarfs
- Geeignete Modelle zur Messung obiger Grössen
- Welches ist die Grenze zwischen praktisch lösbaren und unlösbaren Problemen
- Polynomiale versus exponentielle Komplexität
- Polynomiale Entscheidbarkeit versus polynomiale Verifizierbarkeit
- Ein 1‘000‘000 $ Problem: Die P=NP Frage; NP-Vollständigkeit

Algorithmus

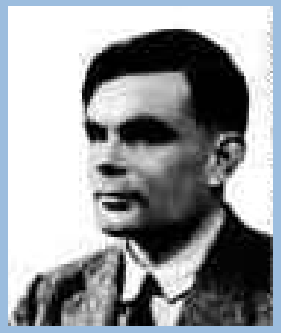
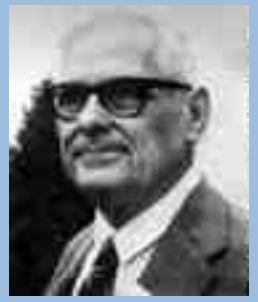
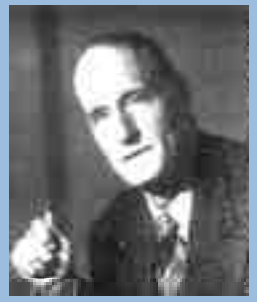
- Ein **Algorithmus** ist eine endliche Beschreibung eines mechanischen Verfahrens zur Lösung eines Problems
- **Informeller, intuitiver Berechenbarkeitsbegriff**
- Beispiele: arithmetische Rechenoperationen, Euklid, n-te Primzahl etc.
- Begriff geht zurück auf den persisch-arabischen Mathematiker **Al-Khowarizmi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

- „Algorithmics – the spirit of computing“ (David Harel)

Die Wurzeln der Berechenbarkeitstheorie

UNIVERSITÄT BERN

- In den dreissiger Jahren des letzten Jahrhunderts wurden zahlreiche **formal-mathematische Berechenbarkeitsmodelle eingeführt und untersucht**
  - Turing-Maschinen
  - Ungetypter Lambdakalkül (Church)
  - Partiell-rekursive Funktionen (Kleene)
  - Gleichheitskalküle (Gödel-Herbrand)

David Hilbert

UNIVERSITÄT BERN

- Motivation zur Formalisierung des Algorithmusbegriffs kam von **David Hilbert** und seinem berühmten Programm
- Hilbert glaubte, dass sich die ganze Mathematik in einem formalen System axiomatisieren lässt und die Frage nach der mathematischen Wahrheit einer Aussage algorithmisch entschieden werden kann
- **Kurt Gödels Unvollständigkeitsätze** zeigten, dass Hilberts Programm nicht realisiert werden kann (siehe später)
- **Church** und **Turing** haben gezeigt, dass die Frage der Gültigkeit in der Prädikatenlogik erster Stufe nicht algorithmisch entschieden werden kann

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

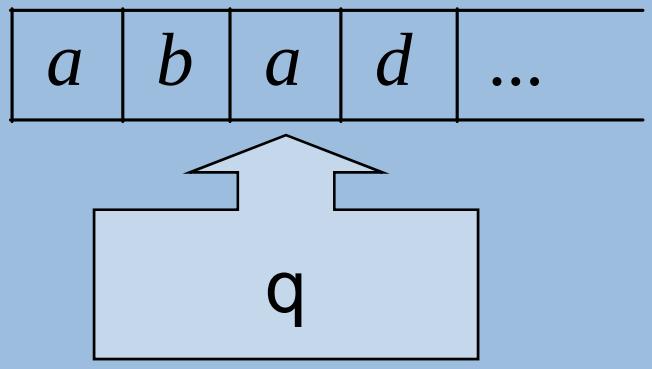
- Welche Funktionen f von Σ^* nach Σ^* sind algorithmisch berechenbar ?

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

Turingmaschinen

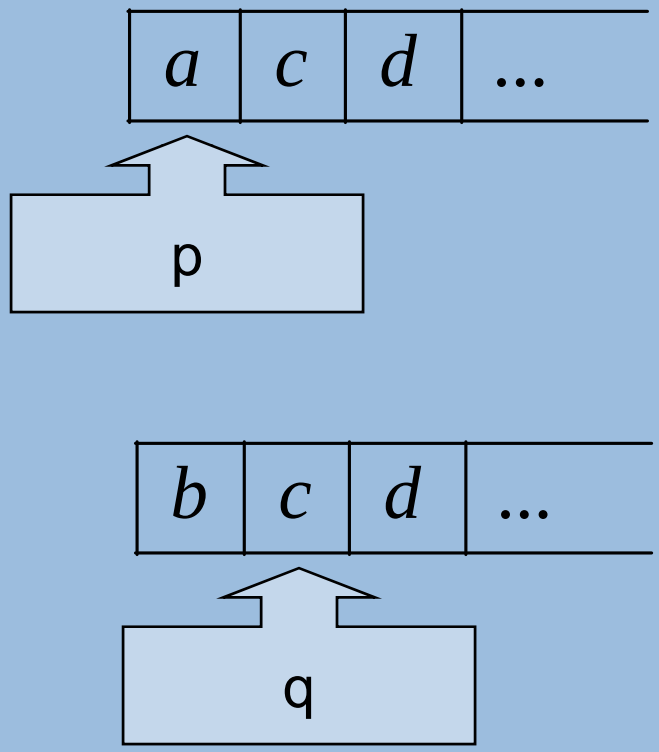
UNIVERSITÄT BERN

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

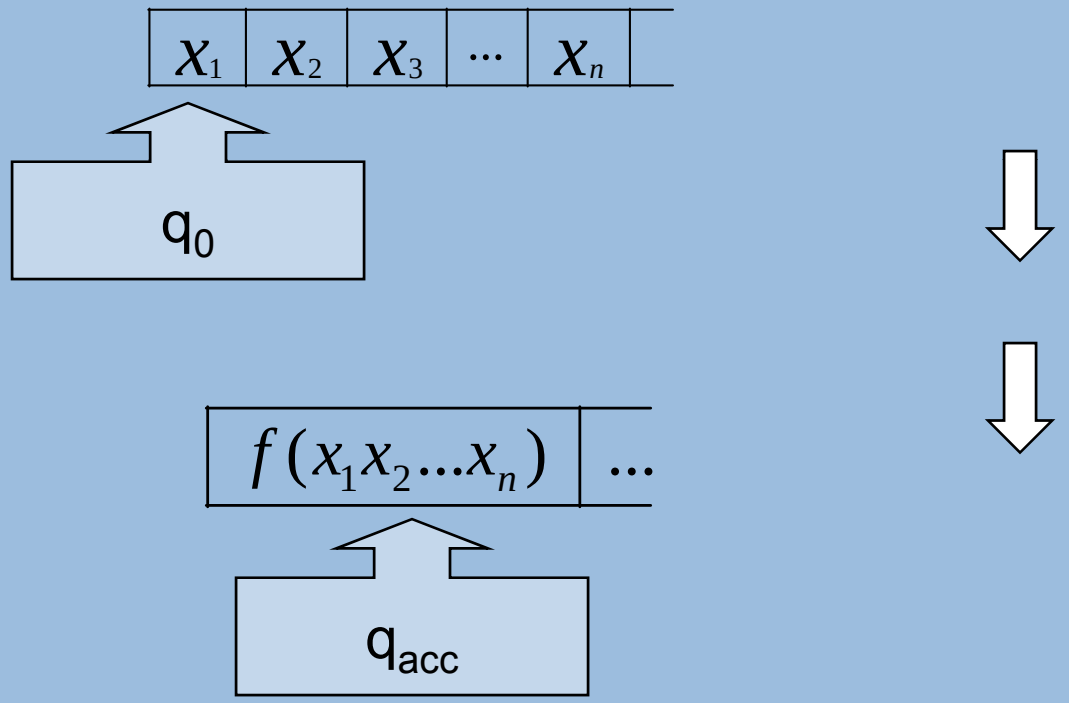


Überführungsfunktion

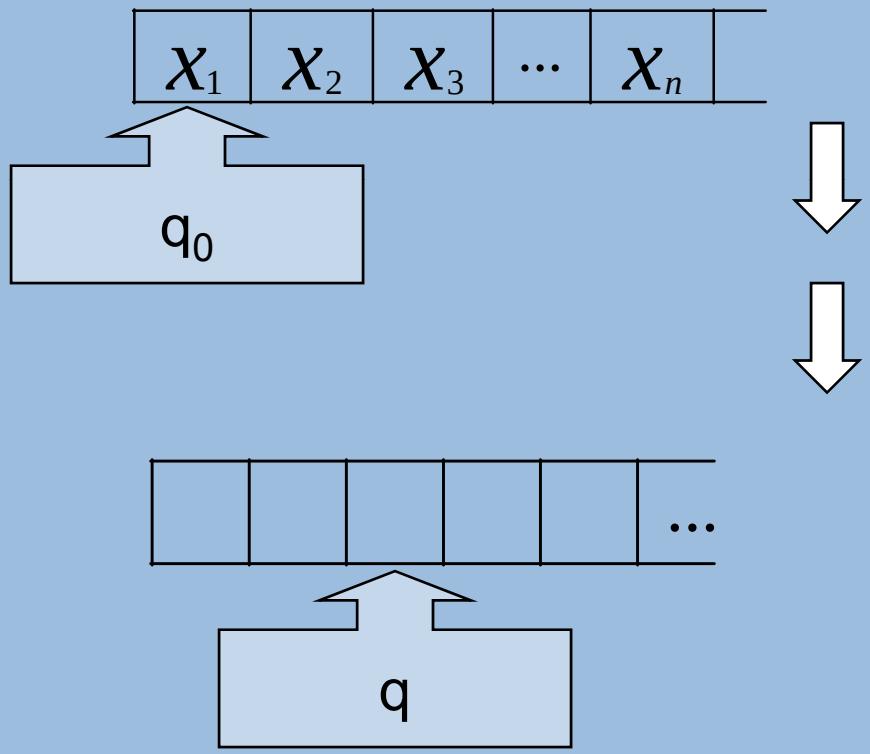
> δ(p, a) = (q, b, R)



Turingberechenbarkeit einer Funktion f



Turingentscheidbarkeit einer Menge L



$$q = \begin{cases} q_{acc}, & \text{falls } x_1 \ldots x_n \text{ in L} \\ q_{rej}, & \text{sonst} \end{cases}$$

Die These von Church

UNIVERSITÄT BERN

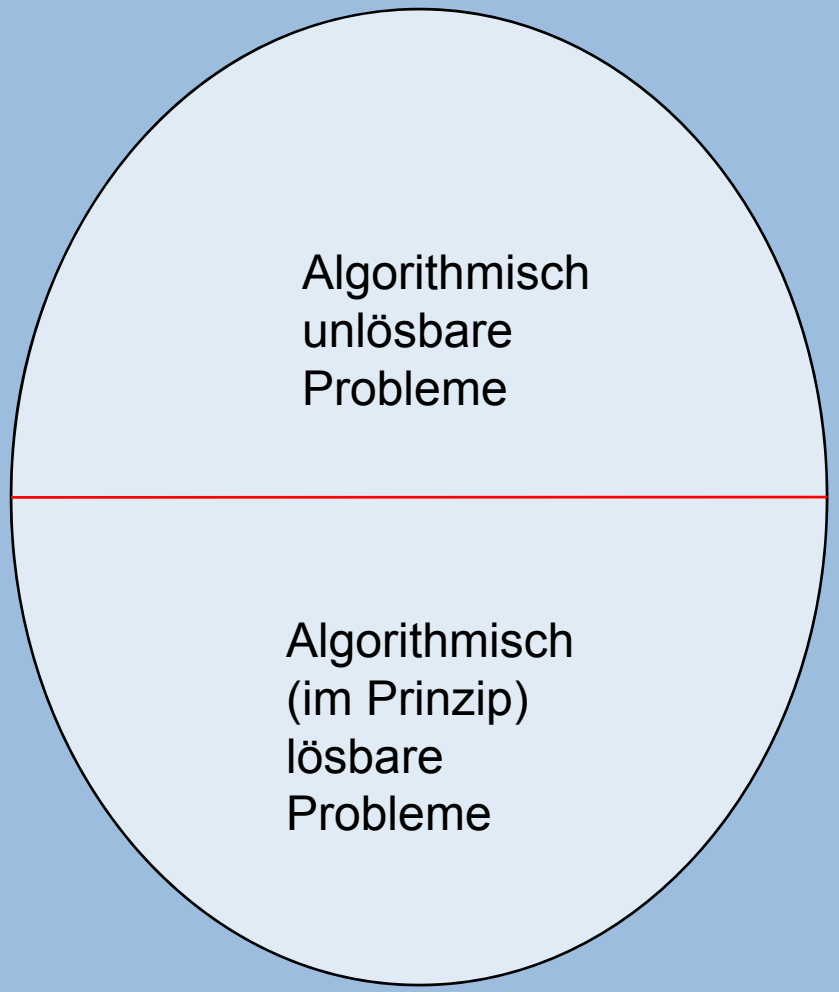
Theorem

Die oben erwähnten Vorschläge zur Formalisierung des Algorithmusbegriffs sind äquivalent, d.h. sie beschreiben dieselbe Klasse von Funktionen.

These von Church

Der intuitive Algorithmusbegriff wird durch jedes dieser Modelle adäquat formalisiert.

Algorithmisch unlösbare Probleme



Entscheidbarkeit vs Semi-Entscheidbarkeit

$$f(x) = \begin{cases} 1 & \text{falls } x \in L \\ 0 & \text{falls } x \notin L \end{cases}$$

$$1 \quad \text{falls } x \in L$$

Das Halteproblem

UNIVERSITÄT BERN

Das Halteproblem H

Stoppt ein beliebiges Programm x angesetzt auf eine Eingabe y nach endlichen vielen Schritten ?

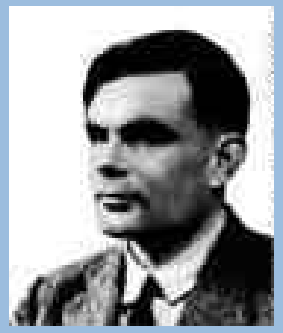
Das spezielle Halteproblem H_0

Stoppt ein beliebiges Programm x angesetzt auf sich selbst nach endlich vielen Schritten ?

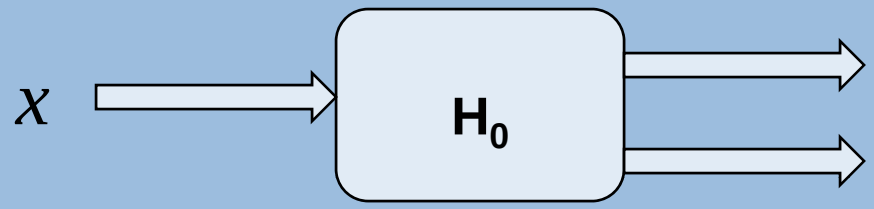
Unentscheidbarkeit des Halteproblems

Unentscheidbarkeit des Halteproblems (Church, Turing)

H und $\mathbf{H_0}$ sind semi-entscheidbar, aber nicht entscheidbar.

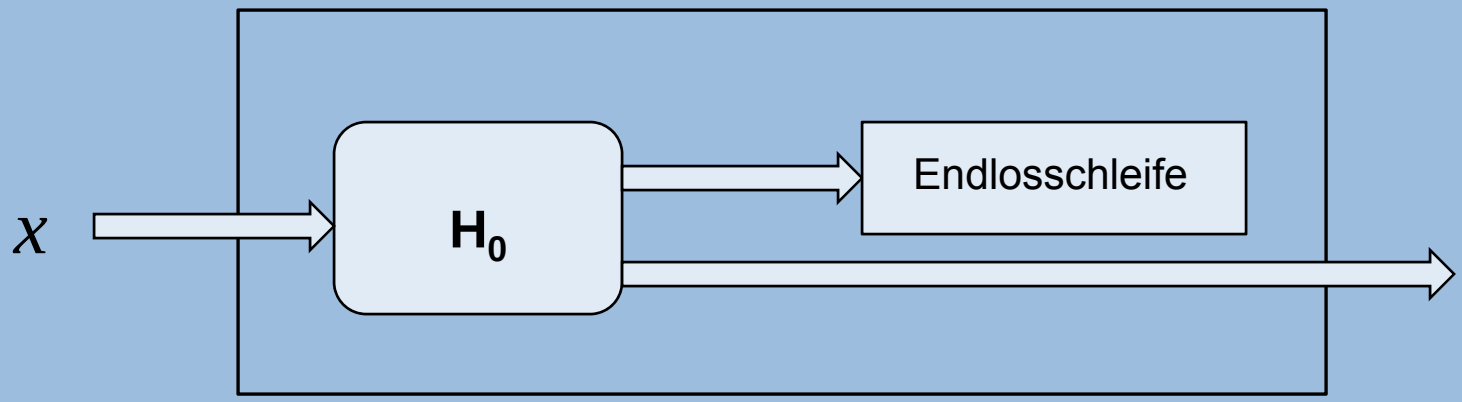


Unentscheidbarkeit von H_0 (Skizze)



1, falls ***x*** bei Eingabe ***x*** stoppt

0, falls ***x*** bei Eingabe ***x*** nicht stoppt



Weitere unentscheidbare Probleme

- Das Gültigkeitsproblem der Prädikatenlogik erster Stufe
- Das Postsche Korrespondenzproblem
- Beide Probleme sind semi-enscheidbar
- Viele weitere Probleme (siehe auch Satz von Rice)

Der Satz von Rice

UNIVERSITÄT BERN

Funktionales Verhalten von Algorithmen

Die von einem gegebenen Programm x berechnete Funktion hat eine gewisse Eigenschaft **E.**

Theorem (Rice)

Obiges Problem ist für beliebige nicht-triviale **E** unentscheidbar.

Hochgradige Unentscheidbarkeit

- Es bezeichne **(N, +, *)** die Struktur der natürlichen Zahlen mit Addition und Multiplikation
- **Th(N, +, *)** sei die Menge aller (erststufigen) Sätze, welche in **(N, +, *)** wahr sind

Theorem (1. Unvollständigkeitssatz von Gödel)

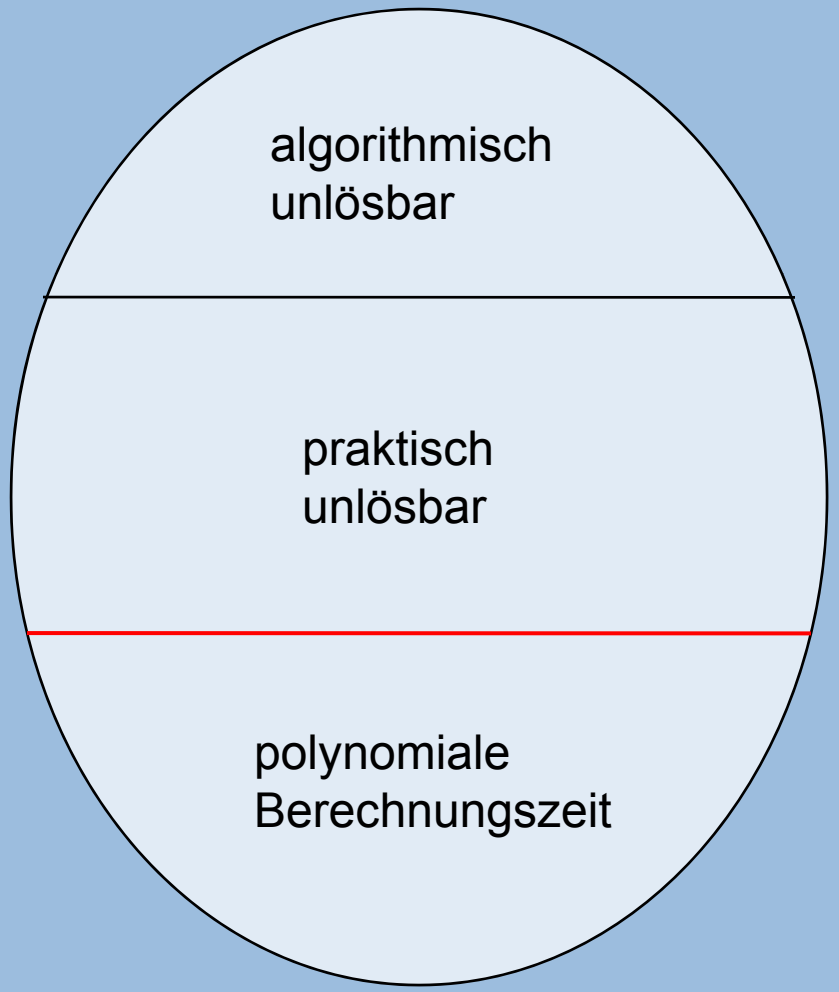
Th(N, +, *) ist nicht semi-entscheidbar.

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
- **polynomiale versus exponentielle Rechnenzeit**
- Viele Probleme, die (vermutlich) keine effiziente Lösung besitzen, haben einen **polynomialen Verifikations-algorithmus**, d.h., mögliche Lösungen können effizient auf ihre Korrektheit überprüft werden

Praktisch unlösbare Probleme



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 ?

Erfüllbarkeitsproblem SAT

UNIVERSITÄT BERN

- **Gegeben:**
  - Boolscher Ausdruck Φ in den Variablen x1, x2, ..., xn
- **Frage:**
  - Gibt es eine Variablenbelegung β aus $\{0,1\}^n$, so dass Φ unter β den Wert 1 annimmt ?

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

UNIVERSITÄT BERN

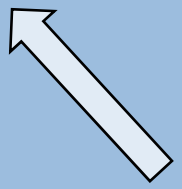
- **P**: Klasse aller Entscheidungsprobleme, welche **durch einen polynomialen Algorithmus entschieden** werden können
- **NP**: Klasse aller Entscheidungsprobleme, für welche **in polynomialer Zeit *verifiziert* werden** kann, ob eine mögliche Lösung korrekt ist
- These von Cook/Levin (1971):

P ≠ NP

Formale Definition von NP

- Ein Entscheidungsproblem ***L*** ist in **NP**, falls es ein Problem L_0 in **P** gibt, so dass

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

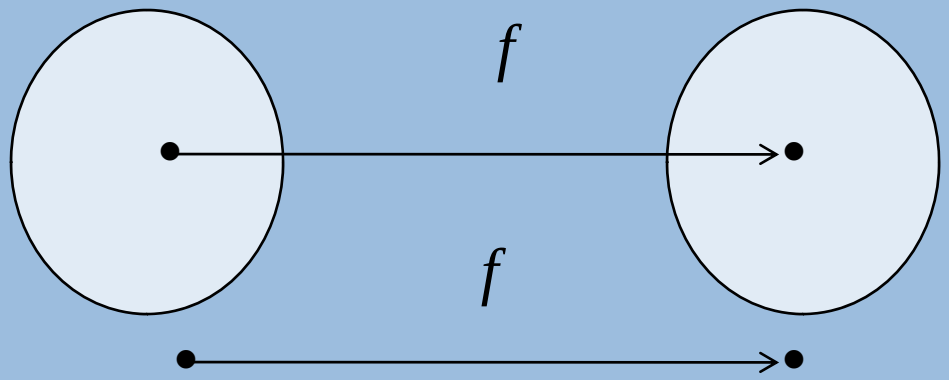


Polynomialer Zeuge, Beweis

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

Theorem (Cook, Karp, Levin)

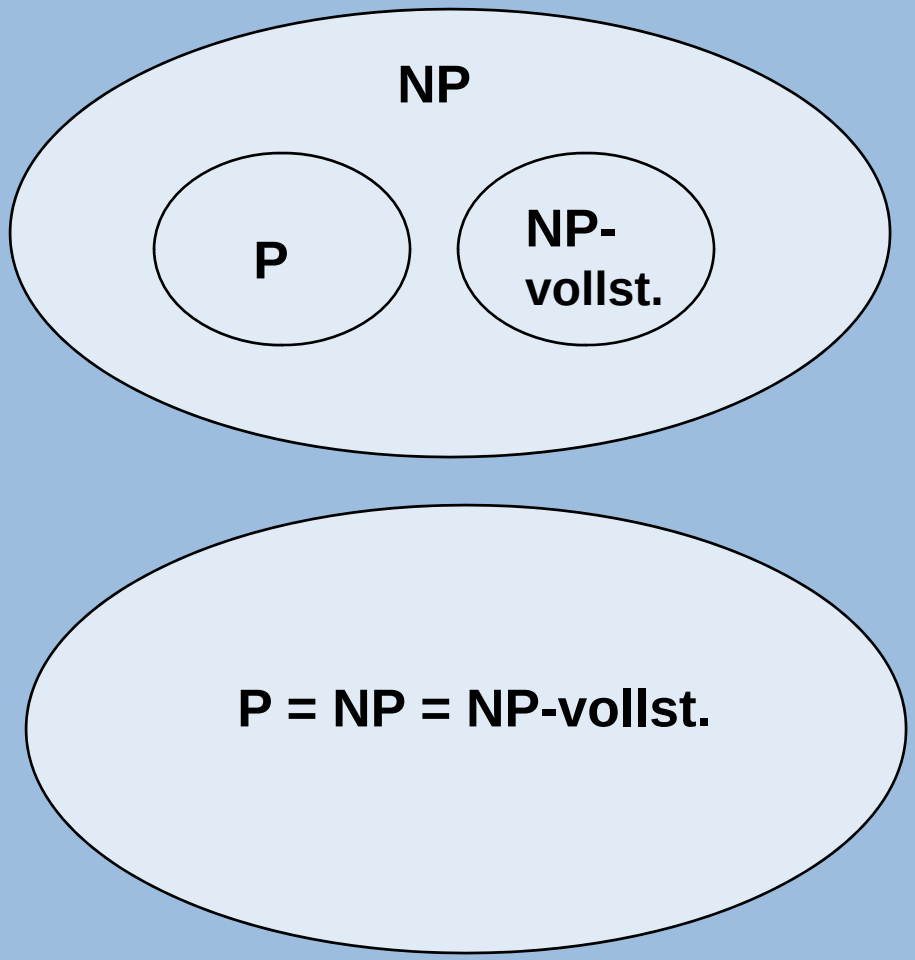
TSP, SAT, BP sind **NP**-vollständig

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

NP-vollständige Probleme

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

Zwei mögliche Welten



Diagonalisierung ?

UNIVERSITÄT BERN

- 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 ist ein Orakel **A**, das **P** und **NP** indentifizerit
- Es gibt ein Orakel **B**, das **P** und **NP** trennt

Logische Methoden zur Trennung von Komplexitätsklassen ?

UNIVERSITÄT BERN

- 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

UNIVERSITÄT BERN

- 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 ?

Jack Edmonds

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 “