UNIVERSITÄT
BERN

Berechenbarkeit

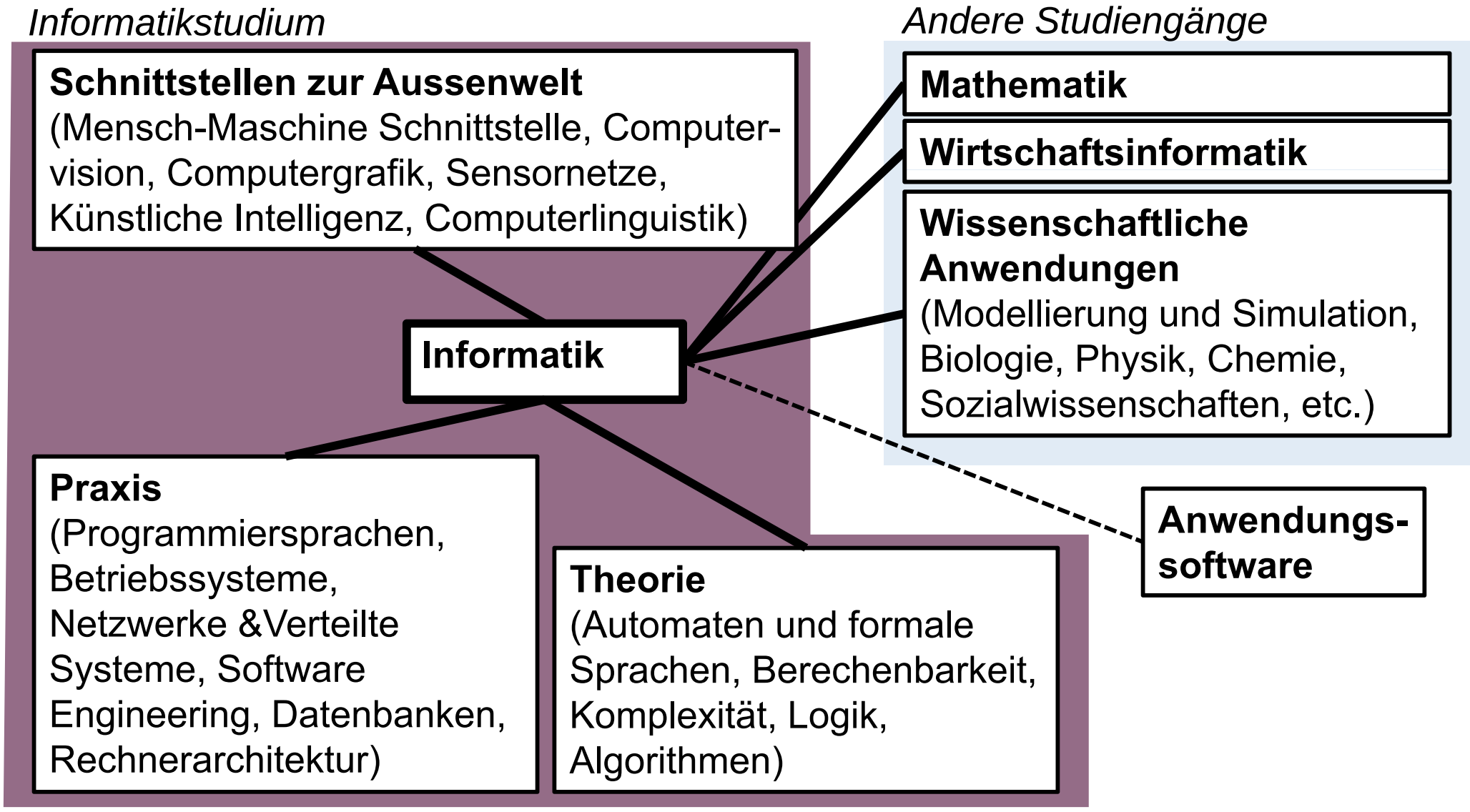
Einführung in die Informatik

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

Ideen und Probleme der theoretischen Informatik

UNIVERSITÄT BERN

- 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)

Einige Väter der theoretischen Informatik

Al-Khowarizmi

W. Ackermann

A. Church

K. Gödel

D. Hilbert

S. C. Kleene

E. Post

A. M. Turing

S. Cook

L. Levin

Inhaltsübersicht

UNIVERSITÄT BERN

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

Berechenbarkeit und Komplexität

UNIVERSITÄT BERN

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

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

- In diesem Vorlesungblock 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 ?

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)**

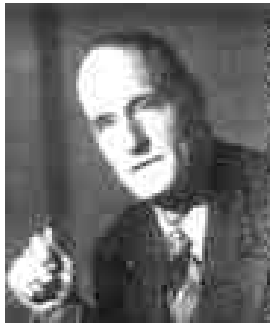
- „Algorithmics – the spirit of computing“ (David Harel)

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

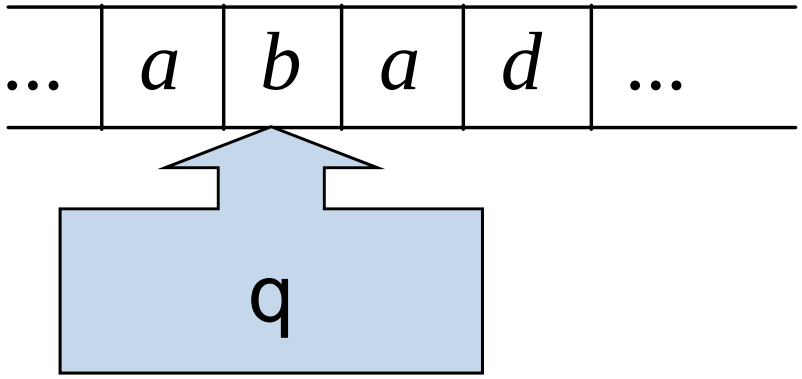
- Welche Funktionen f von Σ^* nach Σ^* sind algorithmisch berechenbar ?

- 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, N} : Überführungsfunktion
  - q_0 Anfangszustand
  - q_{accept}, q_{reject}: akzeptierender / verwerfender Endzustand



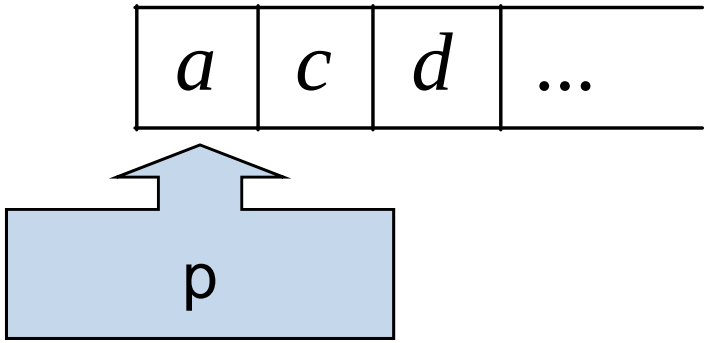
Überführungsfunktion

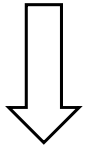
UNIVERSITÄT BERN

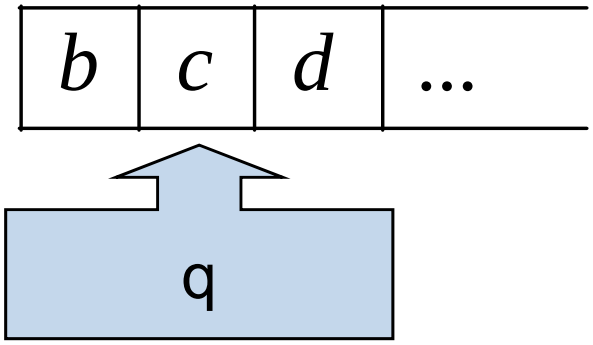
- **δ(z, a) = (z', b, x)** bedeutet Folgendes:
- wenn sich die Maschine im Zustand **z** befindet und **a** das Zeichen ist, welches unter dem Schreib-Lese-Kopf steht, so geht sie im nächsten Schritt in den Zustand **z'** über, überschreibt das Zeichen **a** mit **b**, wobei **a** und **b** Elemente des Arbeitsalphabets Γ sind
- danach führt die Turingmaschine eine Bewegung des Schreib-Lese-Kopfs aus, die durch x = **L**, **R**, oder **N** angegeben wird
- hierbei bedeutet **L** "ein Feld nach **links**", **R** "ein Feld nach **rechts**" und **N** "**neutral**" (keine Bewegung)

Überführungsfunktion (Beispiel)

> δ(p, a) = (q, b, R)

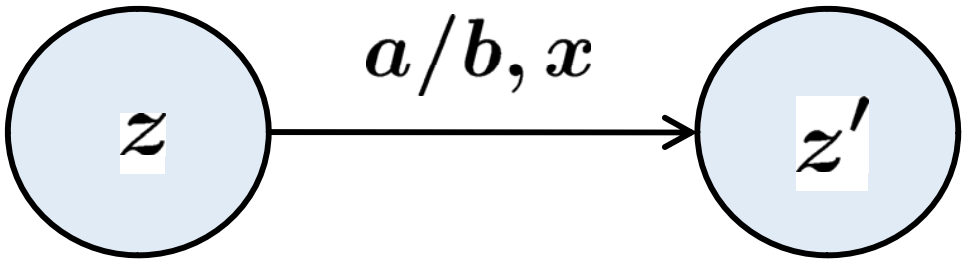




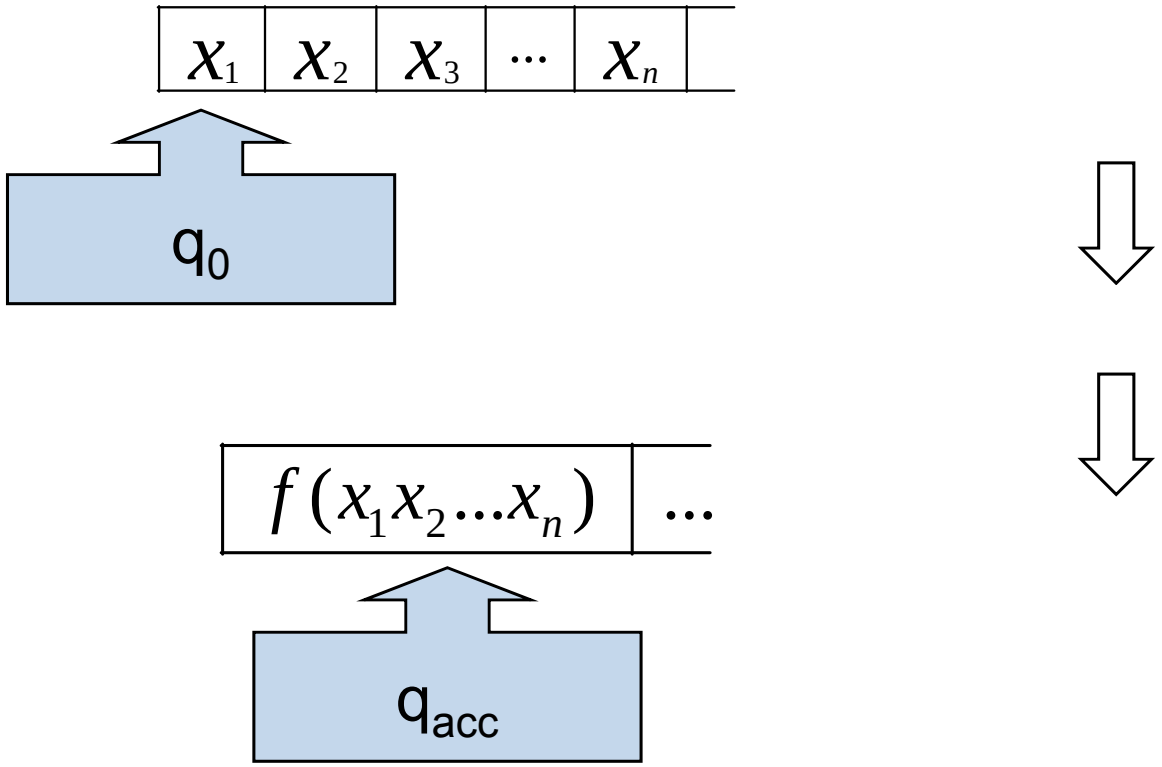


Zustandsübergangsdiagramme

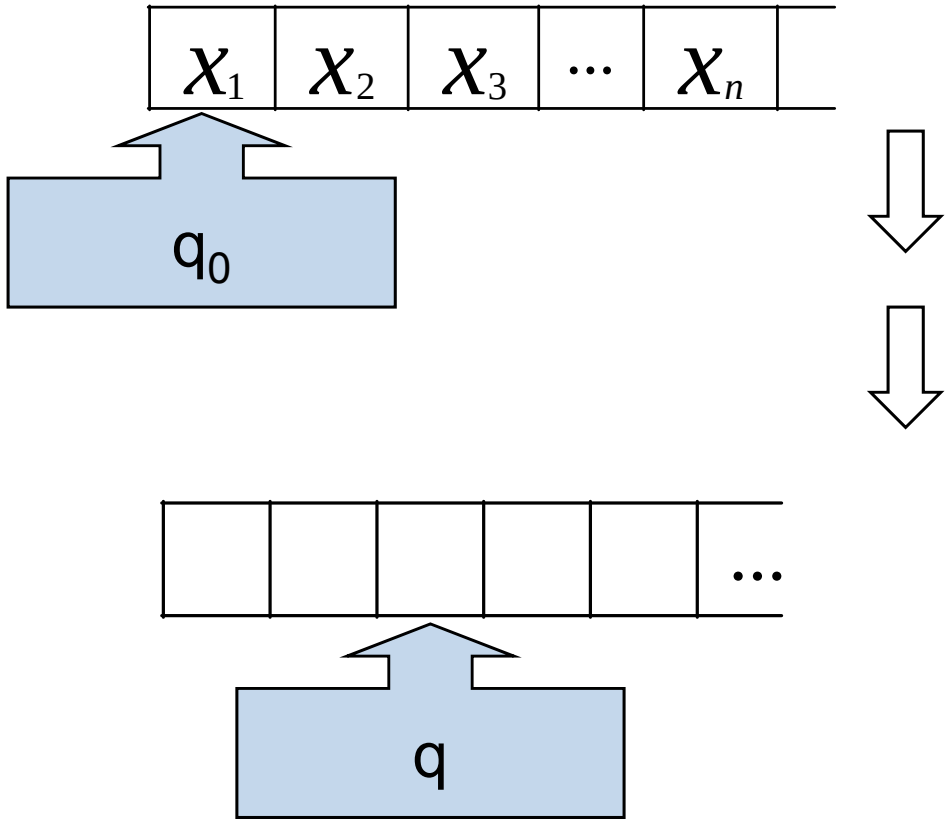
- Graphisch kann man Turingmaschinen ähnlich wie endliche Automaten darstellen
- Die Zeile **δ(z, a) = (z', b, x)** kann in einem Diagramm durch die folgende Kante beschrieben werden:



Turingberechenbarkeit einer Funktion f



Turingentscheidbarkeit einer Menge L



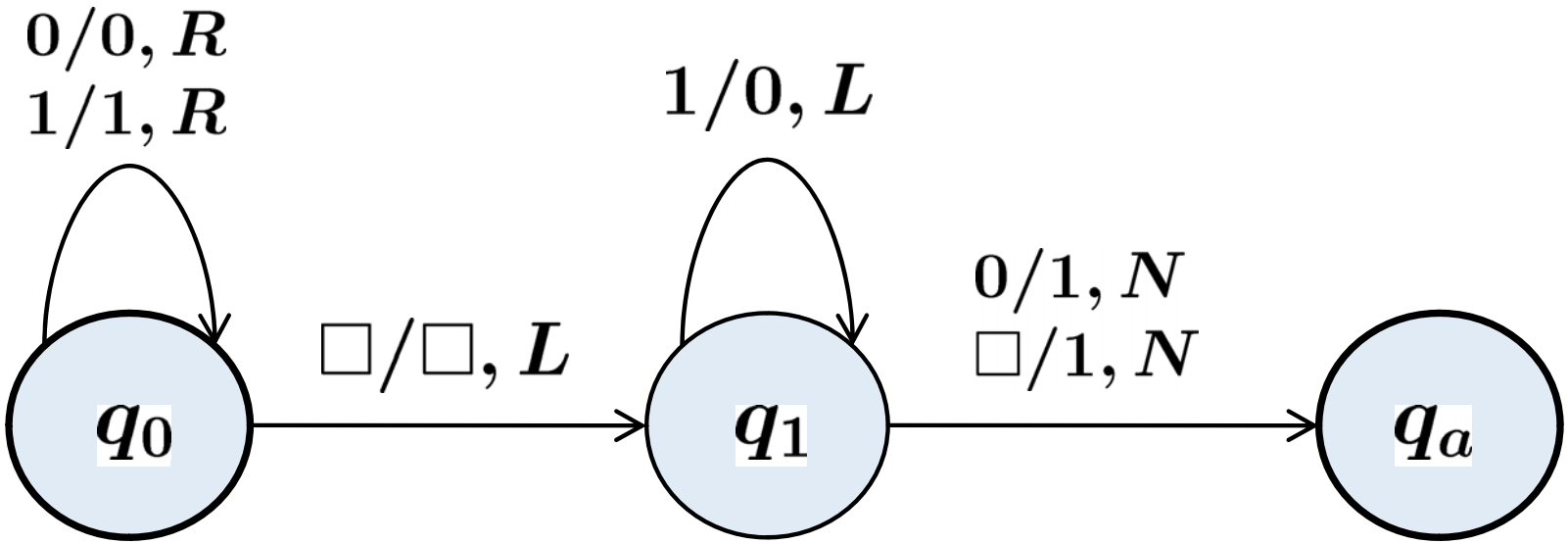
$$q = \begin{cases} q_{acc}, & \text{falls } x_1 \ldots x_n \text{ in L} \\ q_{rej}, & \text{sonst} \end{cases}$$

Beispiel (binäre Nachfolgerfunktion)

UNIVERSITÄT BERN

- Wir wollen zeigen, dass die Funktion **f(x) = x+1**, welche jeder natürlichen Zahl x ihren Nachfolger zuordnet, durch eine Turingmaschine berechenbar ist
- Wir nehmen an, dass eine natürliche Zahl x durch ihre Binärdarstellung gegeben ist
- Wenn man die folgende Maschine auf eine Zahl x (in Binärdarstellung) ansetzt, so stoppt diese nach endlich vielen Schritten in einem akzeptierenden Zustand, so dass auf dem Arbeitsband dann die Zahl x+1 (ebenfalls in Binärdarstellung) steht.
- Im folgenden steht □ für das Leerzeichen

Beispiel (Zustandsübergangsdiagramm)



Beispiel für x = 101

UNIVERSITÄT BERN

> Ablauf für die Eingabe **101**

1 0 1	1 **0** 1	1 0 **1**	1 0 1 □
q_0	q_0	q_0	q_0
1 0 **1**	1 **0** 0	1 **1** 0	
q_1	q_1	q_a	

Universelle Turingmaschine

- Eine **universelle Turingmaschine UTM** ist eine spezielle Turingmaschine, welche jede andere Turingmaschine **T** ausführen kann
- Die Eingabe einer **UTM** ist das Programm $\boldsymbol{\delta_T}$ von **T** sowie eine Eingabe **w** für **T**; das Programm $\boldsymbol{\delta_{UTM}}$ von **UTM** führt dann das Programm $\boldsymbol{\delta_T}$ von **T** aus
- Die universelle Turingmaschine **(= Turingmaschinen-Interpreter**) ist damit ein theoretisches Konzept eines universellen Rechners

Die These von Church

UNIVERSITÄT BERN

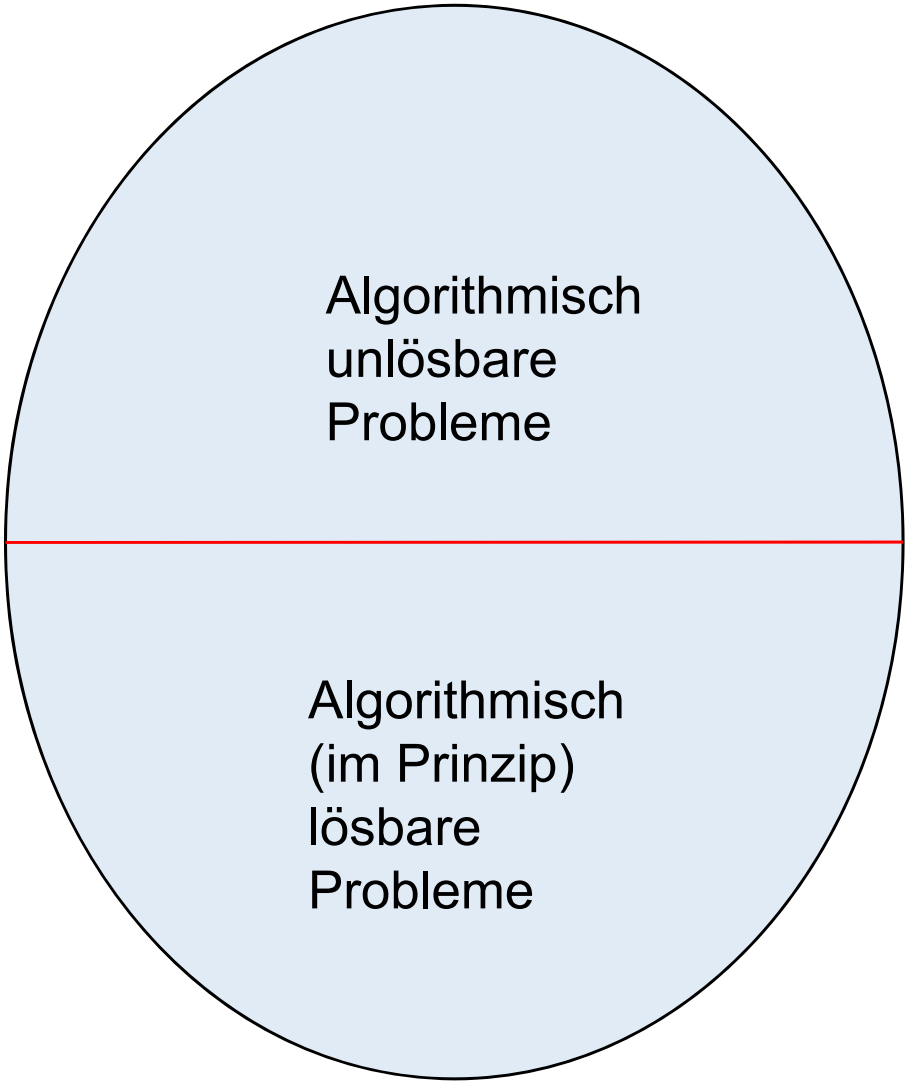
Theorem

Die auf Folie 9 erwähnten Vorschläge zur Formalisierung des Algorithmusbegriffs sind äquivalent, d.h. sie beschreiben dieselbe Klasse von Funktionen.

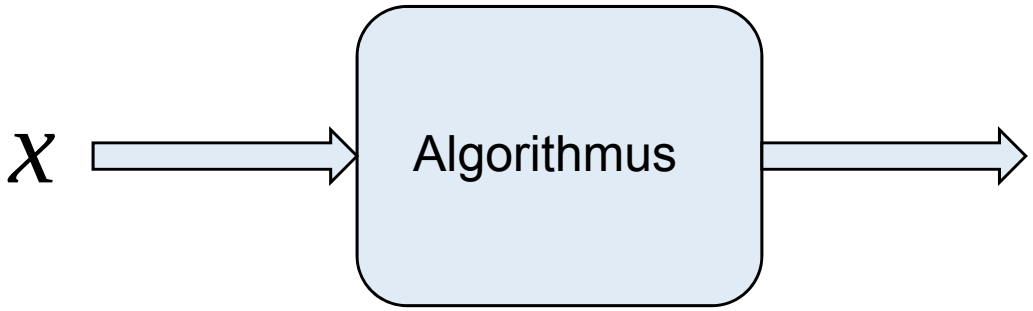
These von Church

Der intuitive Algorithmusbegriff wird durch jedes dieser Modelle adäquat formalisiert.

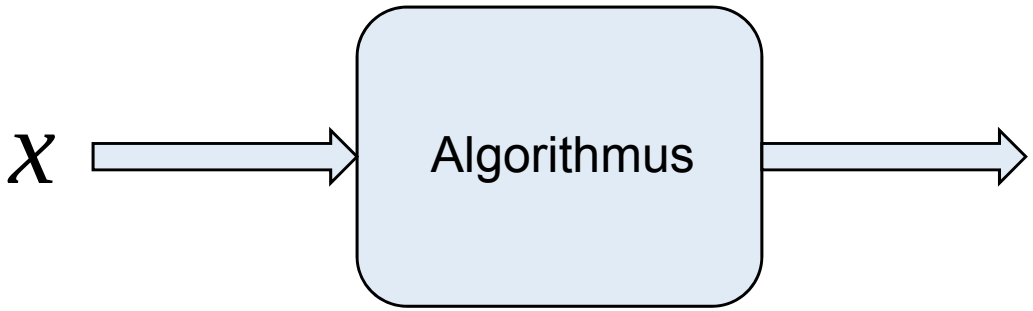
Algorithmisch unlösbare Probleme



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

Das Halteproblem H

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

Das spezielle Halteproblem H_0

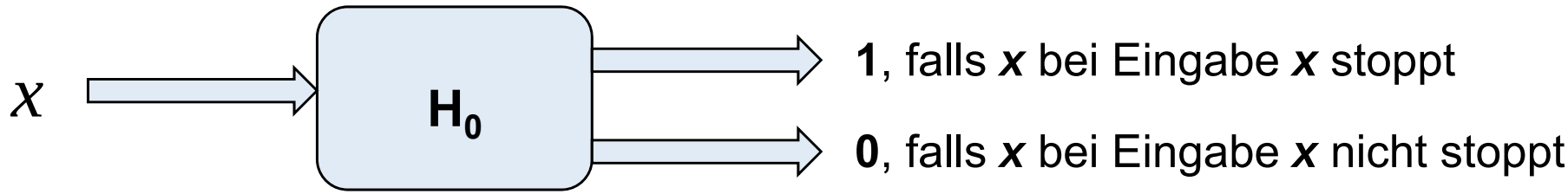
Stoppt ein beliebiges Programm x angesetzt auf sich selbst nach endlich vielen Schritten ?

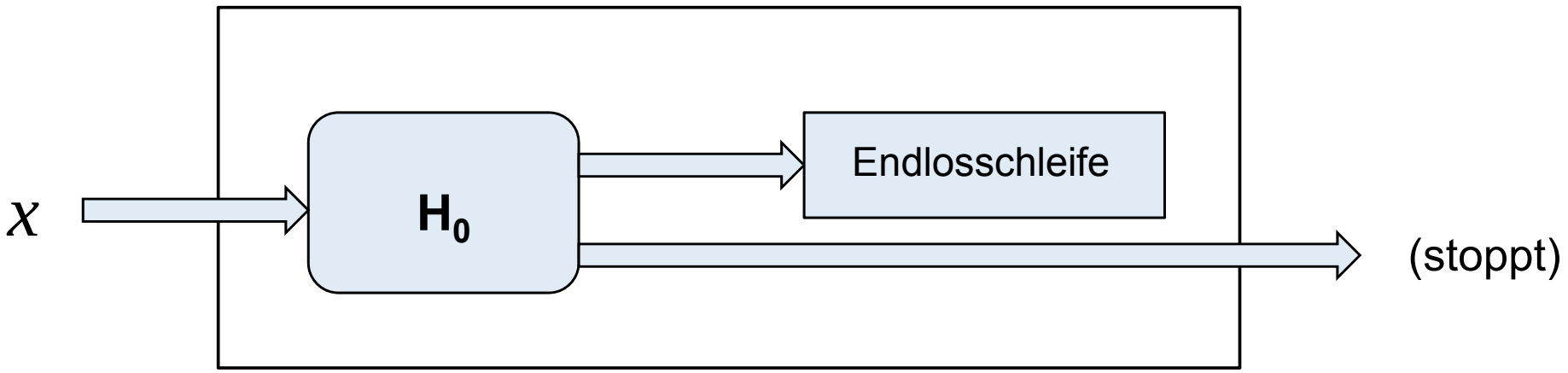
Unentscheidbarkeit des Halteproblems

Unentscheidbarkeit des Halteproblems (Church, Turing)

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

Unentscheidbarkeit von H_0 (Skizze)





Details an der Tafel und in der Übungsserie

Weitere unentscheidbare Probleme

UNIVERSITÄT BERN

- 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

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.

(Nicht-trivial heisst, dass E für mindestens eine, aber nicht für alle berechenbaren Funktionen zutrifft.)

Beispiele zum Satz von Rice

- Es ist nicht entscheidbar, ob ein gegebenes Programm x die Funktion konstant 0 berechnet
- Es ist nicht entscheidbar, ob ein gegebenes Programm x eine überall definierte (= totale) Funktion berechnet
- Es ist nicht entscheidbar, ob ein gegebenes Programm x eine fest vorgegebene Funktion f berechnet (**Unentscheidbarkeit des Spezifikationsproblems**)
- Etc.

Hochgradige Unentscheidbarkeit

UNIVERSITÄT BERN

- 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
- Beispiele:
  - Wahrer Satz: $\forall x \forall y (x * y = y * x)$
  - Falscher Satz: $\forall x \exists y (x = y + y)$

Theorem (1. Unvollständigkeitssatz von Gödel)

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

Literaturhinweise

- U. Schöning, Ideen der Informatik: Grundlegende Modelle und Konzepte der Theoretischen Informatik, Oldenbourg, 2008
- J. Hromkovic, Sieben Wunder der Informatik, Vieweg+Teubner, 2008
- D. Harel, Computers Ltd.: What They Really Can't Do, Oxford University Press, 2003
- A. Dewdney, New Turing Omnibus, Henry Holt, 2003
- M. Sipser, Introduction to the Theory of Computation, Thomson, 2006