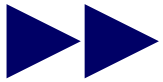
Theoretische Informatik
Kap 2: Berechnungstheorie

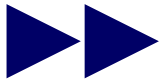


Gliederung der Vorlesung

0. Grundbegriffe
1. Formale Sprachen/Automatentheorie
    - 1.1. Grammatiken
    - 1.2. Reguläre Sprachen
    - 1.3. Kontextfreie Sprachen

2. Berechnungstheorie
    - 2.1. Berechenbarkeitsmodelle
    - 2.2. Die Churchsche These
    - 2.3. Unentscheidbarkeit
3. Komplexitätstheorie
    - 3.1. Nicht-deterministische Turing Maschinen
    - 3.2. Komplexitätsmaße
    - 3.3. Das P=NP? Problem

Theoretische Informatik
Kap 2: Berechnungstheorie



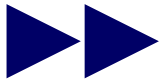
Motivation

Ausgangspunkt: die Vermutung, daß es Probleme gibt, die man „algorithmisch“ nicht in den Griff bekommen kann

... zu deren Lösung man keinen Algorithmus angeben kann

aber... was heißt Problem ... ?
was heißt Algorithmus ... ?

Theoretische Informatik
Kap 2: Berechnungstheorie



Motivation

ein Problem P ist charakterisiert durch

- Menge der infrage kommenden Probleminstanzen
- Menge der infrage kommenden Lösungen

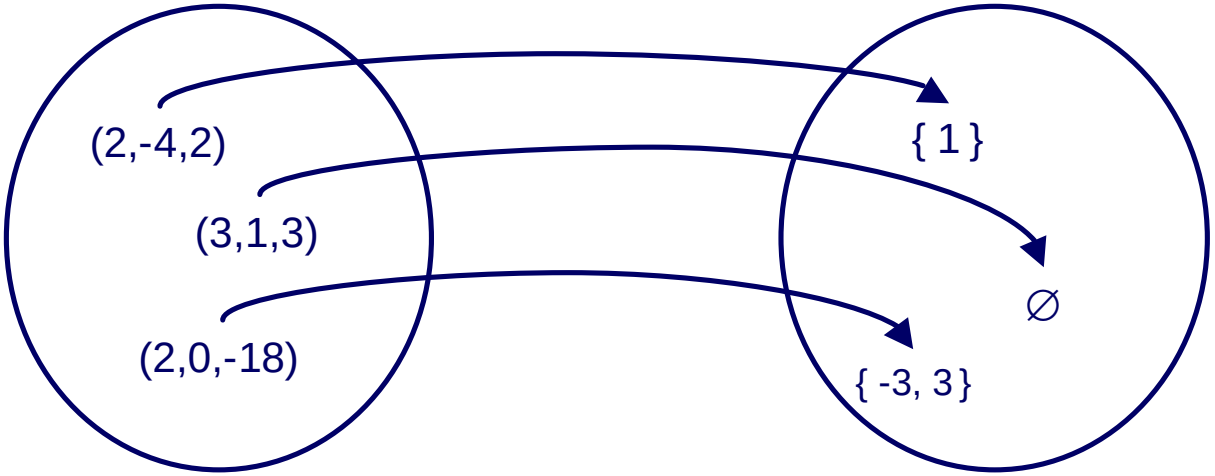
Beispiel: Nullstellen quadratischer Funktionen

- Funktion vom Typ $f(x) = ax^2 + bx + c$
- Menge der infrage kommenden Probleminstanzen
  - alle Tripel (a,b,c)
- Menge der infrage kommenden Lösungen
  - $\varnothing$ (/* keine Nullstelle */)
  - alle Mengen $\{ x_0 \}$ (/* eine Nullstelle */)
  - alle Mengen $\{ x_1, x_2 \}$ (/* zwei Nullstellen */)

Theoretische Informatik
Kap 2: Berechnungstheorie

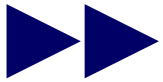
Motivation

Beispiel: quadratische Funktionen



- Probleminstanz (a,b,c) hat die Lösung $\{ x_0 \}$, falls $ax_0^2 + bx_0 + c = 0$ und $ax^2 + bx + c \neq 0$ für alle $x \neq x_0$
- ....

Theoretische Informatik
Kap 2: Berechnungstheorie



Motivation

zugrunde liegende Annahme

uns interessieren nur Probleme, die folgende Eigenschaften haben

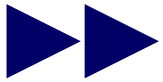
- jede Probleminstanz ist endlich beschreibbar
- jede Lösung ist endlich beschreibbar

Konsequenzen

ein Algorithmus zur Lösung eines Problems

- akzeptiert als Eingabe endliche Beschreibungen von Probleminstanzen
- produziert als Ausgaben endliche Beschreibungen von Lösungen

Theoretische Informatik
Kap 2: Berechnungstheorie



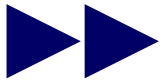
Motivation

zusätzliche Forderungen

ein Algorithmus A zur Lösung eines Problems

- soll korrekt arbeiten, d.h. für jede Probleminstanz soll A eine korrekte Lösung ausgeben
- soll immer terminieren, d.h. für jede Probleminstanz soll A nur endlich viele Schritte rechnen
- A soll endlich beschreibbar sein

„Gretchen-Frage“: Sind diese Forderungen umsetzbar?



Motivation

Abstraktion

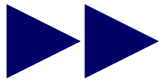
setzt man ein geeignetes Kodierungs-Schema voraus, so gilt für jedes Problem

- jede infrage kommende Probleminstanz kann als natürliche Zahl kodiert werden
- jede infrage kommende Lösung kann als natürliche Zahl kodiert werden

Hinweis

- im allgemeinen wird nicht jede Zahl Kodierung einer Probleminstanz sein
- solange das Kodierungs-Schema berechenbar ist, kann man davon abstrahieren

Theoretische Informatik
Kap 2: Berechnungstheorie



Motivation

Konsequenzen

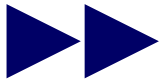
- Zu jedem Problem P gehört eine vollständig definierte Funktion f_P: N → N.
- Jeder Algorithmus A berechnet eine vollständig definierte Funktion f_A: N → N.

... f_A ist eine berechenbare Funktion

Hinweis:

vollständig definiert bedeutet, daß für alle $x \in N$ gilt, daß $f_P(x)$ bzw. $f_A(x)$ definiert ist

Theoretische Informatik
Kap 2: Berechnungstheorie



Motivation

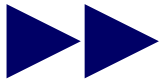
Zielstellung

wir wollen einige erste Konsequenzen aus den bisher getroffenen Annahmen ziehen

im Mittelpunkt stehen die folgenden Punkte:

- jeder Algorithmus A soll endlich beschreibbar sein
- jeder Algorithmus A soll für jede Eingabe terminieren, d.h. die von A berechnete Funktion f_A: N → N soll vollständig definiert sein

Theoretische Informatik
Kap 2: Berechnungstheorie



Motivation

Eine erste Beobachtung

es sei irgendein Algorithmen-Begriff präzisiert, d.h. es ist präzisiert, welche Funktionen f: N → N berechenbar sind

Es gibt vollständig definierte Funktionen f: N → N, die im präzisierten Sinn nicht berechenbar sind.

- es gibt überabzählbar viele vollständig definierte Funktionen f: N → N
- es gibt nur abzählbar viele im präzisierten Sinn berechenbare Funktionen f: N → N

... Diagonalisierung

Theoretische Informatik
Kap 2: Berechnungstheorie

Motivation

Eine zweite Beobachtung

es sei irgendein Algorithmen-Begriff präzisiert, d.h. es ist präzisiert, welche Funktionen f: N → N berechenbar sind

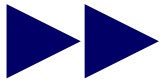
Es gibt nur die folgenden zwei Möglichkeiten:

(1) Es gibt „intuitiv“ berechenbare Funktionen f: N → N, die im präzisierten Sinn nicht berechenbar sind.
(2) Es gibt partiell definierte Funktionen f: N → N, die im präzisierten Sinn berechenbar sind.

... Diagonalisierung

Hinweis

partiell definiert bedeutet, daß es $x \in N$ gibt, für die f (x) nicht definiert ist



Motivation

Zwischenfazit

es sei irgendein hinreichend ausdrucksstarker Algorithmen-Begriff präzisiert

Dann muß es berechenbare Funktionen f: N → N geben,
die partiell definiert sind.

... es muß Algorithmen geben, die nicht für jede Eingabe terminieren

... das liegt an der Forderung, daß Algorithmen endlich beschreibbar sein sollen