Kapitel 3: Berechnungstheorie

Gliederung

Einordnung

- zur Erinnerung
  - Unser Ziel war es, besser zu verstehen
    - dass es und weshalb es algorithmische Probleme gibt, die man nicht lösen kann,
  - Diesem Ziel haben wir uns genähert, indem wir
    - ein „ leistungsfähiges“ Berechnungsmodell definiert haben und
    - uns davon überzeugt haben, dass es aus Gründen der Mächtigkeit vollständig definierte Funktionen gibt, die man mit Programmen dieses Modells nicht berechnen kann.

- Hilft uns diese Annäherung an unser Ziel?
  - erste mögliche Antwort
    - nicht viel (Es könnte ja „leistungsfähigere“ Berechnungsmodelle geben, d.h. Berechnungsmodelle, deren Programme es erlauben, vollständig definierte Funktionen zu berechnen, die man mit Turing-Maschinen nicht berechnen kann.)
  - zweite mögliche Antwort
    - ja (… wenn es gute Gründe dafür gibt, dass es solche „leistungs-fähigeren“ Berechnungsmodelle gar nicht geben kann.)

... und es gibt wirklich gute Gründe ...

Alternative Berechnungsmodelle

- Seitdem A. Turing sein Berechnungsmodell definiert hat, sind eine Vielzahl von unterschiedlicher weiterer Berechnungsmodelle definiert worden, u.a. …
  - partiell-rekursive Funktionen
  - Markov-Algorithmen
  - Random-Access-Maschinen
  - Lambda-Kalkül
  - Petri-Netze mit Verbotskanten
  - Zweikellerautomat
  - While-Programme
  - Goto-Programme
  - Quantencomputer
  - DNA-Computer

*Diesen Berechnungsmodellen liegen **ganz unterschiedliche** Vorstellungen zugrunde, wie Programme beschrieben werden, wie Programme ihre Eingaben verarbeiten, welches Verarbeitungsmodell (und damit welche „Hardware“) zugrunde liegt.*

Eine zentrale Beobachtung

- Für alle bisher definierten alternativen Berechnungsmodelle, wie auch für in beliebigen Sprachen programmierte Computer, konnte nachgewiesen werden, dass sie „gleichwertig“ und insbesondere nicht „leistungsfähiger“ als das Berechnungsmodell Turing-Maschine sind.
- Genauer heißt das ...
  - Ist P irgendein Programm im alternativen Berechnungsmodell,
  - dann kann man anhand von P eine Turing-Maschine M definieren, die dasselbe Eingabe/Ausgabe-Verhalten wie P hat, d.h. dieselbe Funktion wie P berechnet.

*Für viele der gleichwertigen Berechnungsmodelle bestehen auch analoge **Komplexitätszusammenhänge**:*
Was in einem Modell mit gewisser Komplexität berechnet werden kann, kann im anderen Modell mit der gleichen Komplexität berechnet werden.

Die Churchsche These

- Die Churchsche These kann wie folgt formuliert werden:
  - Sei Σ ein Alphabet, und
  - sei f eine vollständig definierte Funktion über Σ^*;
  - dann gilt:

> Wenn die Funktion f überhaupt in einem Berechnungsmodell berechnet werden kann, dann ist f auch Turing-berechenbar.

> *... Wenn es ein Computerprogramm gibt, das das durch f präzisierte Eingabe/Ausgabe-Verhalten hat, so gibt es auch eine Turing-Maschine mit diesem Eingabe/Ausgabe-Verhalten*

- Die Churchsche These wird allgemein akzeptiert.

Churchsche These

- Die Churchsche These, frei formuliert

Die Klasse der „intuitiv berechenbaren“ Funktionen über den natürlichen Zahlen stimmt mit der Klasse der Funktionen überein, die mit Turing-Maschinen berechnet werden können.

Für jedes überhaupt lösbare algorithmische Problem kann man eine Turing-Maschine angeben, die dieses Problem löst.

Churchsche These

- Einordnung

Die Churchsche These

- *ist informell und **kann daher nicht bewiesen werden**.*
- *Es kann nur Evidenz für ihre Richtigkeit gesammelt werden.*

Problematik

- *auf der einen Seite eine „informelle“ Begriffsbildung*
- *auf der anderen Seite eine „formale“ Begriffsbildung*

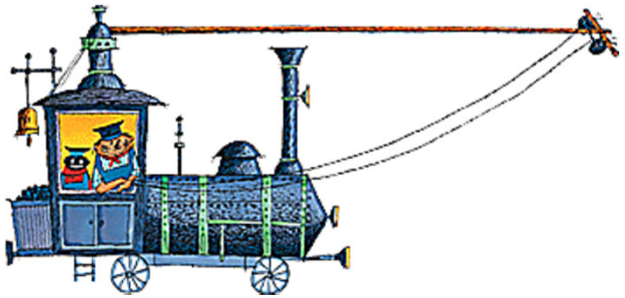
Evidenz für die Richtigkeit:

Alle bisher studierten „formalen“ Begriffsbildungen (d.h. alle Berechenbarkeitsmodelle) sind „äquivalent“.

Anmerkungen

- allgemein Akzeptiertes anderswo

  - Wir betrachten ein Beispiel, das etwas mit Physik zu tun hat:
  - Es geht um sogenannte „Perpetuum mobile“, also Geräte
    - die – einmal in Gang gesetzt – ohne weitere Energiezufuhr ewig in Bewegung bleiben
      und dabei möglicherweise auch noch Arbeit verrichten.



→ *Jim Knopf und die Wilde 13*

Anmerkungen

- allgemein Akzeptiertes anderswo (cont.)

  - Aus dem ersten Hauptsatz der Thermodynamik (dem so genannten Energieerhaltungssatz) folgt, dass es unmöglich ist, ein „Perpetuum mobile“ zu konstruieren.

    Physiker gehen davon aus, dass der Energieerhaltungssatz korrekt ist – auch wenn er nicht mathematisch bewiesen werden kann.

    Diese Akzeptanz hat Einzug ins Leben gehalten: Patentanträge zum Thema „Perpetuum mobile“ werden unter Verweis auf die mangelnde Ausführbarkeit der Erfindung ohne Prüfung abgelehnt.

  - An anderer Stelle in der theoretischen Informatik (genauer: in der Komplexitätstheorie) gibt es ein fast genauso breit akzeptierte Vermutung:

    $P \neq NP$.

Ausblick

- Woran werden wir **konkrete** Nicht-Berechenbarkeit „aufhängen“?
  (im nächsten Abschnitt)
  - Dass es auch „interessante“ Funktionen gibt, die man nicht berechnen kann, liegt – wie wir sehen werden – daran, dass es in jedem „leistungsfähigsten“ Berechnungsmodell
    - Programme gibt, die „partiell“ definierte Funktionen berechnen
    - Programme gibt, die andere Programme als „Unterprozedur“ verwenden

... ersteres ist eine Art „Naturkonstante“ (sonst wäre es kein „leistungsfähigstes“ Berechnungsmodell ...)

... letzteres ist gewünscht (sonst könnten wir keine Programmbibliotheken benutzen ...)