Theoretische Informatik I
Vorlesung vom 14.04.2000 von 13:30 bis 15:00 Uhr

Literaturhinweise

- K. Erk, L. Priese: „Theoretische Informatik„, Springer Verlag, 2000
- U. Schöning: „Theoretische Informatik – kurzgefasst„, Spektrum Verlag, 1994

1. Motivation für die Theoretische Informatik

Kurzgesagt beschäftigt sich die Theoretische Informatik mit dem Weg von einem Problem zum Algorithmus, vom Algorithmus zu einem Programm und schliesslich mit der Implementierung des Programms auf einer Maschine, also mit dem Ablauf:

Problem → Algorithmus → Programm → Maschine

Bei der Betrachtung von **Problemen** stehen folgende Fragen im Vordergrund:
- Wie beschreibt man Probleme?
- Ist das Problem (maschinell) lösbar?
- Ist ein Problem „schwer„ oder „leicht„ lösbar – Effizienz?

Folgende Fragen sind bei der Betrachtung von **Algorithmen** von Interesse:
- Was ist ein Algorithmus – Definition?
- Wodurch kann man Algorithmen charakterisieren?
- Wie beschreibt man einen Algorithmus?
- Welche Merkmale haben Algorithmen?

Bei der Betrachtung von **Programmen** beschäftigt man sich mit den Fragen:
- Welche Sprachen und Sprachklassen gibt es?
- Wie kann man Sprache Erkennen und Erzeugen?
- Wie effizient ist eine Sprache?
- Wie ist die Semantik einer Sprache?

Maschinen schließlich werden unter folgenden Gesichtspunkten betrachtet:
- Welche Maschinentypen gibt es – Typisierung?
- Was sind die Merkmale dieser Maschinentypen?
- Wie groß ist die Leistungsfähigkeit von Maschinen?

1.2. Inhalt des Studiums

Das Grundstudium der Theoretischen Informatik umfasst vier Themenbereiche:
- **Formale Sprachen** machen ca. 40% des Kurses „Theoretische Informatik I„ und wiederum 40% des Kurses „Theoretische Informatik II„ aus
- die **Automatentheorie** umfasst 50% der „Theoretischen Informatik I„
- die Frage nach der **Berechenbarkeit** von Funktionen geht mit 10% in die „Theoretische Informatik I„ und zu 20% in die „Theoretische Informatik II„ ein
- 40% der „Theoretischen Informatik II„ beschäftigen sich mit dem Thema **Semantik**

Theoretische Informatik I
Vorlesung vom 14.04.2000 von 13:30 bis 15:00 Uhr

Das Hauptstudium der Theoretischen Informatik beschäftigt sich mit **Komplexitätstheorie**. Hier stehen Themen wie Compilerbau, Effiziente Algorithmen, Parallele Systeme und Verteilte Algorithmen im Vordergrund. Die theoretische Betrachtung befasst sich mit Grundsatzfragen, wie:

- Erschließen und Strukturieren des Problemraums
- Methoden der Problemlösung
- Lösungssystematik und gegebenenfalls Lösungsgarantie
- Was geht? Was geht nicht? Wie schnell?

Diese Fragen sind bei allen Anwendungsgebieten der Informatik (Programmierung, Künstliche Intelligenz, Betriebssysteme, Datenbanken, Grafik, etc.) von grundsätzlicher Bedeutung.

2. Grenzen der Algorithmisierung

Die zentrale Frage dieses Abschnitts ist: **Gibt es algorithmisch unlösbare Probleme?** Nachfolgend wird gezeigt werden, dass solche Probleme existieren. Dies wiederum lenkt die Betrachtung darauf, um was für Probleme es sich handelt und wieviele Probleme dieser Art es gibt. Vorher soll jedoch noch einmal auf grundlegende mathematische Begriffe eingegangen werden.

2.1. Mathematische Grundbegriffe

- **Kartesisches Produkt**

  Das Kartesische Produkt (auch Kreuzprodukt genannt) zweier Mengen M_1 und M_2 ist definiert: $\mathbf{M_1 \times M_2 = \{(x,y) \mid x \in M_1, y \in M_2\}}$.

  Für Mengen M_1 bis M_n ist das Kartesische Produkt $\mathbf{M_1 \times M_2 \times \ldots \times M_n}$ als die Menge aller geordneten n-Tupel, die sich aus den Mengen M_1 bis M_n bilden lassen, definiert.

  Sind verschiedene Mengen an der Bildung des Kartesischen Produktes beteiligt, so spricht man von einem **inhomogenen** Kartesischen Produkt.

  Ist dagegen $\mathbf{M_1 = M_2 = \ldots = M_n}$ so ist folgende Schreibweise vereinbart:
  $\mathbf{M \times M \times \ldots \times M}$ (n-mal) $= \mathbf{M^n}$. Man spricht von einem **homogenen** Kartesischen Produkt.

Theoretische Informatik I
Vorlesung vom 14.04.2000 von 13:30 bis 15:00 Uhr

- **Wörtermonoide**

  Sei M eine Menge, so ist M^* die Menge aller Wörter mit Elementen aus M definiert mit:

  $\mathbf{M^* = M^0 \cup M^1 \cup M^2 \cup \ldots \cup M^n = \bigcup_{i=0}^{\infty} M^i}$.

  Für $x_1, x_2, \ldots, x_n \in M$ mit $(x_1, x_2, \ldots, x_n) \in M^n$ bezeichnet man $x_1x_2...x_n$ als **Wort**.

  ε steht dabei für das **leere Wort**. Man vereinbart: $\varepsilon \in \mathbf{M^0}$.

  Mit **|w|** wird die **Länge** eines Wortes $w \in M^*$ bezeichnet.

  Als Operation auf einer Menge von Worten dient die **Konkatenation**. Sie ist auf der gesamten Menge von Wörtern definiert. Es gilt:
  $\mathbf{u, v \in M^* \Rightarrow u \bullet v \in M^*}$.
  Beispiel: u = Baum, v = Haus $\Rightarrow$ u • v = BaumHaus

  M^* bildet mit der Konkatenation als Operation eine algebraische Struktur die als **freies Monoid** bezeichnet wird. Das bedeutet:
  - Die Operation ist stets ausführbar. Das Ergebnis ist eindeutig bestimmt und liegt in der Ausgangsmenge.
  - Das **Assoziativgesetz** gilt: $\forall$ **u, v, w** $\in$ **M : u • (v • w) = (u • v) • w**
  - Es existiert (genau) ein **neutrales Element**: $\forall$ **x** $\in$ **M : x** • ε **= x =** ε • **x**

  Die Menge M, mit deren Elementen Wörter gebildet werden, bezeichnet man als **Alphabet**. Durch Definition einer Syntax $\mathbf{L \subseteq M^*}$ erhält man eine **formale Sprache**.

  Beispiel für die Definition einer Menge von Wörtern:
  $M = \{a, b\}$, $M^* = \{w \mid w = x_1...x_n \mid \text{für } 1 \le i \le n\ x_i \in M\}$

- **Potenzmenge**

  Unter der Potenzmenge $\mathbf{2^M}$ einer Menge M versteht man die Menge aller ihrer Teilmengen. Es gilt: $\mathbf{2^M = \{K \mid K \subseteq M\}}$.

  Die Potenzmenge der leeren Menge $\varnothing$ ist definiert: $2\varnothing = \{\varnothing\}$, $_2 2\varnothing = 2^{\{\varnothing\}} = \{\varnothing, \{\varnothing\}\}$

Theoretische Informatik I
Vorlesung vom 14.04.2000 von 13:30 bis 15:00 Uhr

- **Funktionsbegriffe**

  Für alle Funktionen gilt, dass sie **nacheindeutig** sind.

  Eine Funktion f: M → K ist **total**, wenn es zu jedem x ∈ M ein y ∈ K mit f(x) = y gibt, sie also auf der gesamten Menge M definiert ist. Ansonsten ist eine Funktion **partiell**.

  Eine Funktion ist **injektiv**, wenn sie voreindeutig ist. D.h.: $\mathbf{f(x_1) = f(x_2) \Rightarrow x_1 = x_2}$

  Eine Funktion ist **surjektiv**, wenn es zu jedem y ∈ K ein x ∈ M gibt mit y = f(x), also jedes y ∈ K Bild von (mindestens) einem x ∈ M ist.

  Eine Funktion ist **bijektiv**, wenn sie total, injektiv und surjektiv ist.

  Zwei Mengen M und K sind **isomorph**, wenn es eine bijektive Abbildung zwischen M und K gibt.

  Sei f: M → K eine partielle Funktion. Dann heisst **D(f) = {x ∈ M | f(x) definiert}** **Domain** oder **Definitionsbereich** von f. **R(f) = {y ∈ K | ∃ x ∈ M : f(x) = y }** heisst **Range** oder **Zielbereich** von f.

  In der Informatik ist es üblich, von einer partiellen zu einer totalen Funktion überzugehen, indem man den Quell- und den Zielbereich einer Funktion um das sogenannte **„bottom„-Element** ⊥ erweitert. ⊥ stellt dabei eine „fiktive„ Ausgabe für einen nichtterminierenden Algorithmus dar. Es gilt:

  - $\mathbf{M_\perp := M \cup \perp}$, $\mathbf{K_\perp := K \cup \perp}$
  - $\mathbf{f_\perp : M_\perp \rightarrow K_\perp}$
  - durch $\mathbf{f(x)} = \begin{cases} \mathbf{f(x),\ falls\ f(x)\ definiert} \\ \perp\mathbf{,\ sonst} \end{cases}$

  Die Definition von $f_\perp$ wird auch auf das Kartesische Produkt erweitert. Man vereinbart:

  $\mathbf{f_\perp : A_1 \times \ldots \times A_n \rightarrow B}$

  $\mathbf{f_\perp(a_1, \ldots, a_n)} = \begin{cases} \mathbf{f(a_1, \ldots, a_n),\ falls\ a_i \neq \perp\ für\ 1 \leq i \leq n} \\ \perp\mathbf{,\ falls\ a_i = \perp\ für\ ein\ i \in \{1, \ldots, n\}} \end{cases}$

  Man bezeichnet dies als **Strikte Fortsetzung**. Das bedeutet, dass keine undefinierten Werte zugelassen werden.