Grundlagen der Programmierung (GdP)

Ralf Möller, FH-Wedel

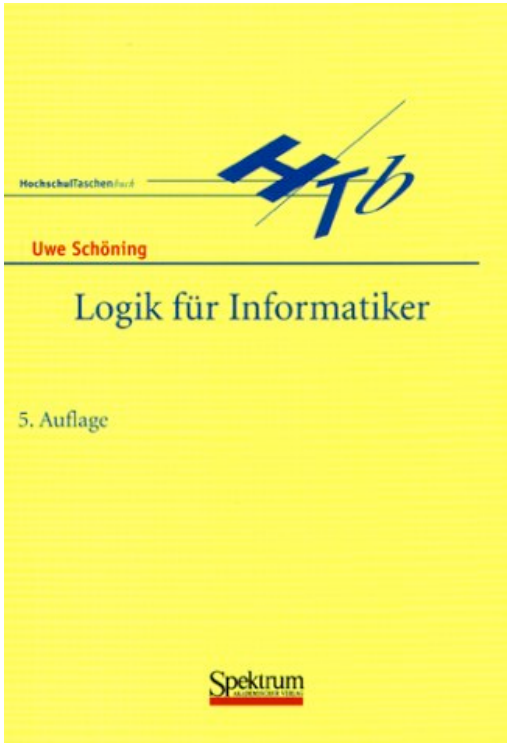
- Voraussetzungen:
  - Mengenlehre, Relationen, Funktionen
- Lernziele allgemein:
  - Fundamente und Grundprinzipien der Programmierung
  - Systematische Entwicklung von Programmen
- Organisation:
  - Vorlesung, Übungen, Tutorien

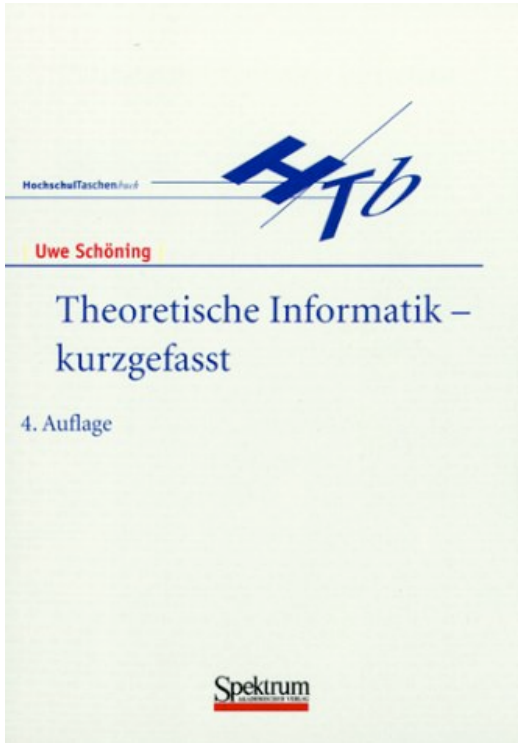
GdP: Das Konzept zur Lehre

- Vorlesung: Mi 8.00 Uhr, 9.30 Uhr, HS 4
  - Vermittlung stofflicher Inhalte
  - Austeilen von Aufgaben
- Übung: Alissa Kaplunova, Mo 12.30 Uhr, HS 4
  - Klären von Fragen, Wiederholung der Vorlesungsinhalte
  - Durchführung von Übungen in Kleingruppen
- Tutorium: Alissa Kaplunova
  - Klären von Fragen

Literatur, Details und Zusatzinformationen

- Infos: http://www.fh-wedel.de/~mo/lectures/gdp-sose-04.html
- Literatur:





- Weitere, ergänzende Literatur auf der obigen Webseite

Überblick über die Vorlesung

- Einführung: Algorithmen, Entwurf von Algorithmen
- Aussagenlogik, Prädikatenlogik,
  Spezifikation der Aufgabe von Algorithmen
- Zuweisungen, Kontrollstrukturen, Bedingungen
  und die systematische Entwicklung von Algorithmen
- Funktionen, Prozeduren, Rekursion
- Komplexität von Algorithmen
- Abstrakte Automaten und Formale Sprachen

Danksagungen

- Die Vorlesung baut auf der gleichnamigen Vorlesung von Uwe Schmidt aus früheren Semestern auf.
- Folien zu dem Buch "Logik für Informatiker" von Uwe Schöning wurden übernommen von Javier Esparza http://wwwbrauer.in.tum.de/lehre/logik/SS99/
- Folien zu dem Buch "Theoretische Informatik kurz gefaßt" wurden übernommen von Angelika Steger http://www14.in.tum.de/lehre/2000SS/info4/

1 Einleitung

1.1 Algorithmus

Computer führt Routineaufgaben aus

- einfache Operationen (Addition, Vergleich)
- hohe Geschwindigkeit

Fragen Welche Operationen?
Welche Reihenfolge der Operationen?

$\hookrightarrow$ Beschreibung durch **Algorithmus**

Algorithmus *(erste Definition)*
beschreibt eine Methode zur Lösung einer Aufgabe,
besteht aus einer endlichen Folge von Schritten (einfache Operationen)

Prozeß	Abarbeitung eines Algorithmus
Prozessor	Einheit die einen Prozeß ausführt
Algorithmus	in der Datenverarbeitung: Ein Algorithmus berechnet aus Eingabedaten Ausgabedaten (Resultate)
formal	Ein Algorithmus berechnet eine Funktion $f : E \longrightarrow A$. E ist der Wertebereich der Eingabedaten, A ist der Wertebereich der Ausgabedaten.
Daten	Werte aus bestimmten Wertebereichen

Computer ein spezieller Prozessor

Komponenten

- Zentraleinheit **Central Processing Unit, CPU**, Ausführung der Basisoperationen
- Speicher
  - Daten mit denen die Basisoperationen manipulieren
  - Operationen des Algorithmus das **Programm**
- Ein– und Ausgabe–Geräte I/O

Geschwindigkeit $10^6 - 10^9$ Operationen/Sekunde.

Zuverlässigkeit	sehr hoch
Fehlerursache	der Algorithmus
Prozeß	berechnet eine Funktion für bestimmte Eingabedaten
Speicher	immer billiger ⇒immer größer 1970 : 64 KByte 1980 : 640 KByte 1990 : 8 MByte 2000 : 256 MByte
Kosten	pro Operation immer billiger

1.2 Programme und Programmiersprachen

Algorithmus in einer Sprache formulieren, die der Prozessor versteht

Interpretation

- verstehen, was jeder Schritt bedeutet
- Operation ausführen

Programmier–sprache Sprache, in der ein Algorithmus für einen Computer formuliert wird

Programm ein in einer Programmiersprache formulierter Algorithmus

programmieren Algorithmen in Programme umsetzen

Elementar–operationen Operationen, die ein Prozessor ausführen kann

Sprachhierarchie maschinennah $\Rightarrow$ problemorientiert

Maschinensprache Programmiersprache, die ein Computer direkt versteht (eine Folge von 0-en und 1-en)

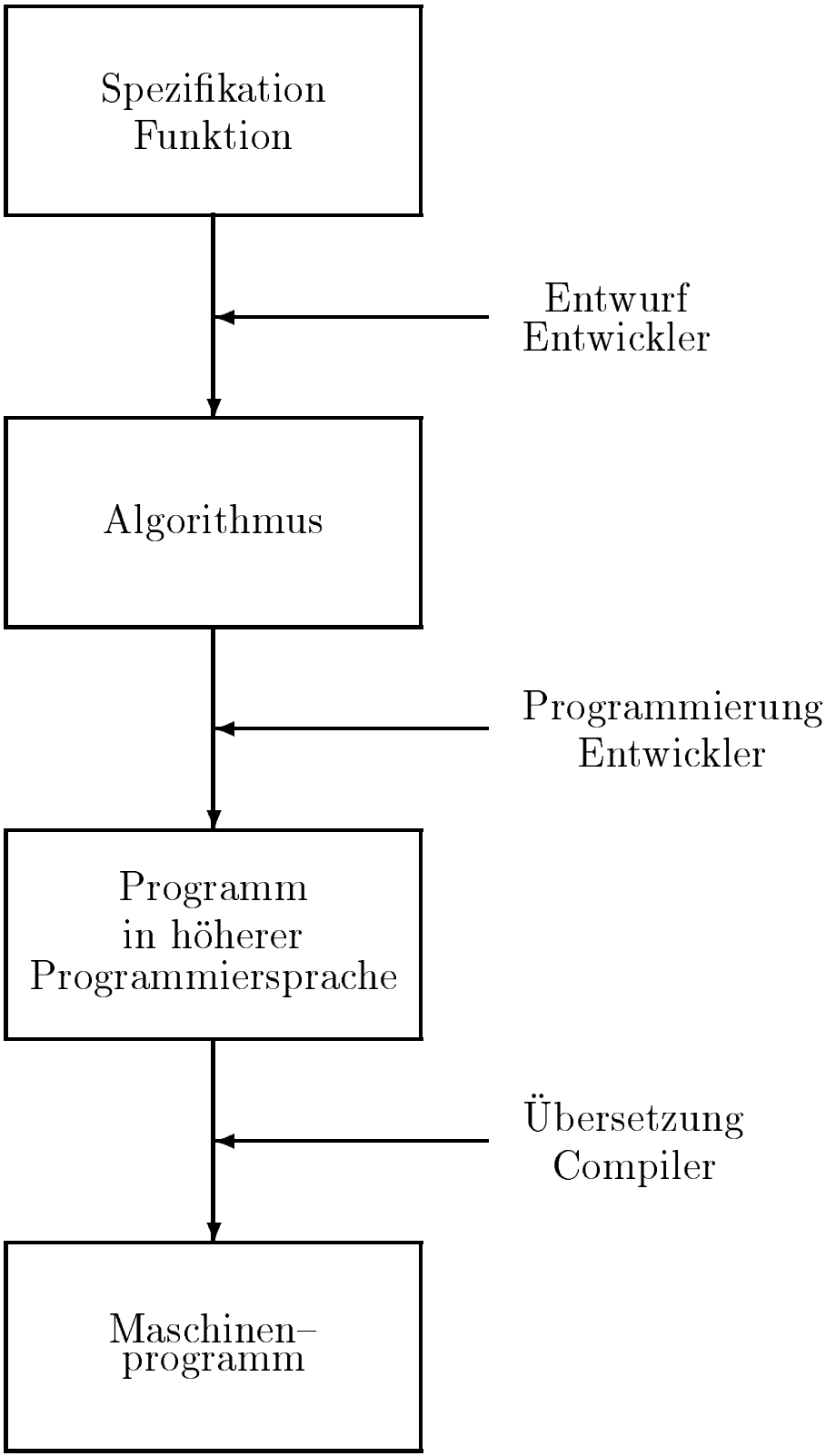
Assemblersprache Maschinensprache in einer für Menschen lesbaren (nicht unbedingt verständlichen) Form: Jede Instruktion erhält einen Namen

Assembler Ein **Programm** zur Transformation einer Assemblersprache in die zugehörige Maschinensprache

höhere Programmier–sprache Zur Vereinfachung der Programmierung Anpassung der Programmiersprache an problem– und aufgabenorientierte Notation

- komplexere Elementaroperationen
- übersichtlichere Anordnung der Anweisungen

Compiler ein **Programm** zur Transformation von Programmen einer höheren Programmiersprache in die Maschinen– oder Assemblersprache eines Computers


Spezifikation
Funktion
Entwurf
Entwickler
Algorithmus
Programmierung
Entwickler
Programm
in höherer
Programmiersprache
Übersetzung
Compiler
Maschinen–
programm

Abgrenzung	Hardware $\Leftrightarrow$ Software: fließend
Hardware	implementiert eine Menge von Elementaroperationen
Betriebssystem	erweitert diese Menge um neue Elementaroperationen, die durch (kurze) Programme implementiert sind
Basis–Software	erweitert diese Menge nochmals, z.B. durch E/A–Operationen
$\hookrightarrow$	Für die Programmentwicklung ist es unwesentlich, wie die Elementaroperationen implementiert sind, entscheidend ist, welche Operationen verfügbar sind

Beispiele

Arithmetik	für reelle Zahlen in Hardware ($\Leftarrow$ Coprozessor) in Software ($\Leftarrow$ Emulation)
Multiplikation	als Instruktion durch Zurückführen auf Addition
$\hookrightarrow$	Algorithmenentwicklung auch für die Hardware–Entwicklung von Bedeutung

Operationen im Rechner und Betriebssystem sind

Funktionen Eine Funktion ist eine eindeutige Zuordnung von Elementen einer Menge D zu den Elementen einer Menge R. Jedem Element aus D darf höchstens ein Element von R zugeordnet sein. Ist f eine solche Funktion, so schreibt man

$$f : D \longrightarrow R$$

Urbildbereich D heißt Urbildbereich

Bildbereich R heißt Bildbereich

Definitionsbereich Nicht jedem Element aus D muß ein Element aus R zugeordnet sein. Ist einem Element d kein Element aus R zugeordnet, so ist f für d nicht definiert. Die Menge der Elemente von D, für die f definiert ist, heißt Definitionsbereich und wird mit

$$Def(f)$$

bezeichnet.

Operationen auf den Wertebereichen sind Funktionen

Alle Operatoren $(+, -, *, \mathsf{div}\ , \mathsf{mod}\ , \wedge, \vee, \ldots)$ sind Namen für Funktionen

totale Funktion f heißt totale Funktion, wenn

$$Def(f) = D$$

ist.

partielle Funktion f heißt partielle Funktion, wenn

$$Def(f) \subset D$$

ist.

Bild Ordnet die Funktion f dem Element $d \in D$ das Element $r \in R$ zu, so heißt r Bild von d unter f. Man schreibt

$$f : d \mapsto r$$

oder

$$f(d) = r$$

einstellig $f : D \longrightarrow R$

n-stellig $f : D_1 \times \ldots \times D_n \longrightarrow R$

Funktion $\Rightarrow$ Algorithmus $\Rightarrow$ Programm $\Rightarrow$ Ausführung

Funktion	zu einer Funktion (Spezifikation) gibt es viele verschiedene Algorithmen
Algorithmus	zu einem Algorithmus gibt es viele verschiedene Programme
Programm	zu einem Programm gibt es viele verschiedene Prozessoren und Maschinenprogramme

zentral **Wie entwirft man Algorithmen ?**

☞ Viel schwieriger als die Programmierung
(Umsetzung: Algorithmus ⇒Programm)

☞ **Es gibt keinen Algorithmus zur Entwicklung von Algorithmen !!!**

aber Prinzipien, Techniken, Richtlinien

Berechenbarkeit Gibt es Funktionen (Prozesse) für die es keinen Algorithmus gibt?

↪ Wenn ja ⇒ nicht alles kann mit einem Computer berechnet werden !!!

• Kann man für eine Funktion (Prozeß) entscheiden, ob es hierfür einen Algorithmus gibt?

Komplexität

Fragen Welche und wieviele Betriebsmittel braucht ein Prozeß zur Ausführung eines Algorithmus?

Betriebsmittel

- Zeit
- Speicher
- Prozessoren
- Geräte

Vergleich Wann ist ein Algorithmus besser als ein anderer?

Komplexität eines Algorithmus. Die Komplexität eines Algorithmus ist der Aufwand an Betriebsmitteln bei der Berechnung.

Maschinenmodell mit den elementaren Operationen und Ablaufsteuerungen bildet die Basis für die Komplexitätsabschätzungen und den Vergleich von Algorithmen

- Turing–Maschine
- Registermaschine
- Parallelrechner

Komplexität einer Funktion = Komplexität des bestmöglichen Algorithmus, der diese Funktion berechnet.

Korrektheit

Frage

Berechnet ein Algorithmus **immer** genau denselben Wert wie die zugehörige Funktion?

Antwort

ist schwierig.

Korrektheitsbeweise für Programme sind sehr umfangreich und schwierig.

Hier ist noch viel Forschung notwendig.

Korrektheitsbeweise und -argumentationen können immer nur relativ zu einer Spezifikation geführt werden !!!

Behauptung: „ *Dieses Programm ist richtig* “ setzt eine Spezifikation voraus.

2 Entwurf von Algorithmen

2.1 Algorithmen, Programme, Programmiersprachen

Algorithmus ist eine Verarbeitungsvorschrift, die aus genau bestimmten Elementaroperationen aufgebaut ist, und bei deren Interpretation die Reihenfolge der Ausführung der Elementaroperationen genau festgelegt ist.

in Daten–verarbeitung Die Elementaroperationen berechnen aus Eingabedaten (Parametern) neue Ausgabedaten (Resultate)

Daten

Werte aus Wertebereichen (Typen)

- Wahrheitswerte (wahr, falsch)
- ganze Zahlen
- reelle Zahlen
- Namen
- Texte
- Tabellen
- ...

Folgerung

Ein Algorithmus beschreibt eine Funktion

$f : E \longrightarrow A$

wobei E der Wertebereich der Eingabedaten ist, A der Wertebereich der Ausgabedaten.

Terminierung Ein Algorithmus terminiert, wenn seine Interpretation nach endlich vielen Schritten ein Ergebnis liefert

Beispiel

nicht terminierend: Sisyphos: mußte einen Felsen auf einen Berg wälzen, von dem der immer wieder herabrollte.

Problem Ist sichergestellt, daß ein Algorithmus für alle möglichen Eingaben terminiert.

$\hookrightarrow$

Korrektheit

(1) ein Algorithmus berechnet immer dengleichen Wert wie die zugehörige Funktion (*partielle* Korrektheit)

(2) der Algorithmus terminiert immer

Sprachen zur Formulierung von Algorithmen

Umgangssprache großes Vokabular, komplizierte Grammatik, mehrdeutig

↪ für Computer unverständlich

↪ für Menschen verständlich

Mathematische Formelsprache großes Vokabular, exakt, eindeutig, ausdruckskräftig, komplexe Operationen

↪ für Computer schon besser geeignet,

↪ durch die hohe Ausdruckskraft nicht immer automatisch in eine für Computer verständliche Sprache umsetzbar

↪ für Menschen nur mit math. Vorbildung verständlich

höhere Programmier–sprache

exakt, eindeutig, einfache Elementaroperationen

$\hookrightarrow$ automatisch umsetzbar (compilierbar)

$\hookrightarrow$ länglicher als Formelsprache

$\hookrightarrow$ noch gut lesbar

Maschinensprache kleines Vokabular, schlecht lesbar, einfache Elementaroperationen

$\hookrightarrow$ gut auf einem Computer auszuführen

$\hookrightarrow$ schlecht zum Entwickeln und Programmieren

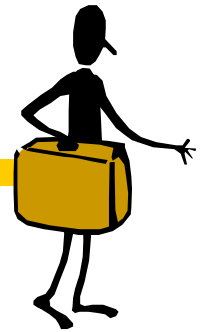
2.2 Syntax und Semantik

Syntax eines Satzes (einer Sprache): grammatikalische Aufbau des Satzes

Semantik eines Satzes (einer Sprache): Interpretation des Satzes, Zuordnung einer Bedeutung zu dem Satz

$\hookrightarrow$ nur sehr wenigen syntaktisch richtigen Sätzen kann eine Bedeutung zugeordnet werden, kann sinnvoll interpretiert werden.

Zusammenfassung, Kernpunkte



- Algorithmusbegriff
- Sprachen zur Formulierung von Algorithmen
  - Begriffe: Syntax, Semantik
- Ausführung von Algorithmen
  - Begriffe: Prozessor, Betriebsmittel, ...

Was kommt beim nächsten Mal?

- Grundlagen zur Formulierung von Algorithmen insbesondere für Bedingungen
- Aussagenlogik