

Grundlagen der Programmierung

Vorlesung 1 vom 14.10.2004
Sebastian Iwanowski
FH Wedel

Organisatorisches

Vorlesung

Donnerstags, 08:00 – 09:15

Raum HS 6

Übung

Freitags, 09:30 – 10:45, 1. Übungsstunde am 22.10.2004

betreut von Alissa Kaplunova

Raum HS 6

Aufgabenausteilung in der Vorlesung

Abgabe, Hilfe und Lösung zu den Aufgaben 8 Tage später

Vorlesungsüberblick

Inhaltliche Voraussetzungen:

Logisches Denken, Mathematik bis 9. Klasse (Gymnasium)

Lernziele dieser Vorlesung:

Formalisieren des logischen Denkens

Anwenden des logischen Denkens auf Programmanalysen

Systematischer Programmentwurf

Grundlagen des Entwurfs und der Analyse von Algorithmen

kein Lernziel dieser Vorlesung:

Überblick über Programmiersprachen

Erlernen einer konkreten Programmiersprache

Vorlesungsüberblick

Vorlesungsthemen:

1. Einführung
2. Logik
3. Programmentwicklung und -verifikation
4. Entwurf und Analyse von Algorithmen

Inhaltliche Überschneidungen mit anderen Vorlesungen:

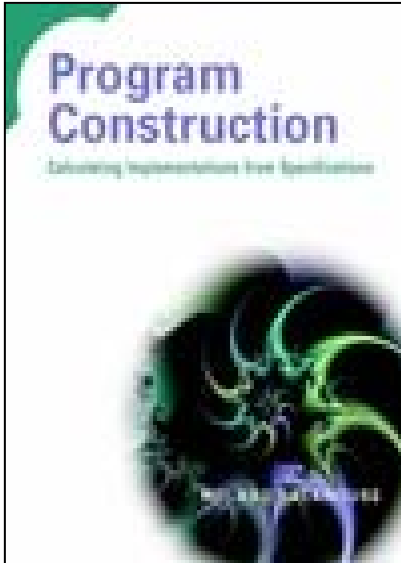
Programmieren I, Diskrete Mathematik

Material zu dieser Vorlesung: <http://www.fh-wedel.de/~iw/Lehrveranstaltungen/WS2004/GdP.html>

Foliensatz von Ralf Möller: <http://www.sts.tu-harburg.de/~r.f.moeller/lectures/gdp-sole-04.html>

Literatur (im Folgenden genannt)

Literatur



Roland Backhouse: *Programmkonstruktion und Verifikation*
Hanser 1989 (vergriffen), ISBN 3-446-15056-0

Englische Neuauflage:

Program Construction: Calculating Implementations from Specifications
Wiley 2003, ISBN 0470848820



Michael Huth / Mark Ryan: *Logic in Computer Science*
Cambridge University Press 2004 (2. Auflage), ISBN 052154310X

Literatur

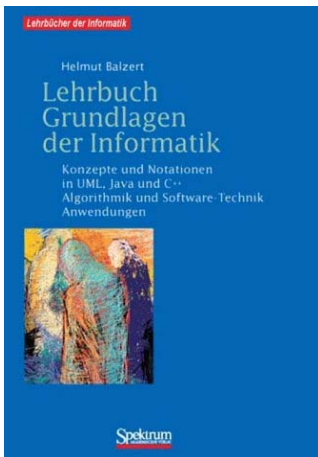


Heinz-Peter Gumm / Manfred Sommer:

Einführung in die Informatik

Oldenbourg 2004 (6. Auflage), ISBN 3-486-27389-2
in unserer Bibliothek:

Oldenbourg 1998 (3. Auflage), ISBN 3-486-24422-1



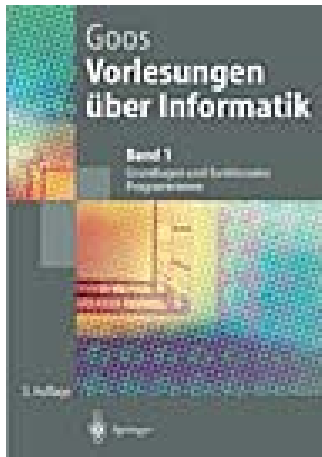
Helmut Balzert: *Lehrbuch Grundlagen der Informatik*

Spektrum 2004 (2. Auflage, erscheint am 25.10.2004),
ISBN 3-8274-1410-5

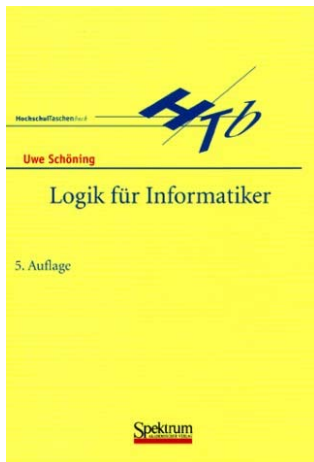
in unserer Bibliothek:

Spektrum 1999 (1. Auflage), ISBN 3-8274-0358-8

Literatur



Gerhard Goos:
Vorlesungen über Informatik,
Band 1: Grundlagen und funktionales Programmieren
Springer 2000 (3. Auflage), ISBN 3-540-67270-2



Uwe Schöning: *Logik für Informatiker*
Spektrum 2000 (5. Auflage), ISBN 3-8274-1005-3

Grundlagen der Programmierung

1. Einführung

➔ Grundlegende Eigenschaften von Algorithmen und Programmen

2. Logik

Aussagenlogik

Prädikatenlogik

3. Programmentwicklung und –verifikation

Grundlagen der Programmverifikation

Verbundanweisungen

Verzweigungen

Schleifen

Modularisierung

Rekursion

4. Entwurf und Analyse von Algorithmen

Klassifikation von Algorithmen

Programmierung von Algorithmen

Bewertung von Algorithmen

Was soll ein Computer machen ?

- **Ausgabe / Wirkung**

 - Anzeigen, ausdrucken**

 - Steuern von Geräten**

 - Abspeichern von Daten**

- **Eingabe / Impuls**

 - Tastendruck, Mausklick, Mausbewegung**

 - Automatisch gemessene Werte**

Was soll ein Computer machen ?

Grundsätzlich ist ein eindeutiger Zusammenhang zwischen Eingabe / Impuls und Ausgabe / Wirkung gefordert

Beschreibungsmöglichkeit: durch Funktionen

Spezifikation $f : D \rightarrow B$

- **Grundsätzlich sind beliebige Definitions- und Bildbereiche möglich**
- **Für die Betrachtung von Software reicht es aus, sich auf Zahlen- und Textbereiche zu beschränken**

Wie soll es der Computer machen ?

Eine **Spezifikation** legt fest, **was** der Computer machen soll.

Ein **Algorithmus** legt fest, **wie** der Computer die Spezifikation erfüllen soll.

- Ein Algorithmus ist die Beschreibung einer Folge von Anweisungen an den Computer, die folgenden Anforderungen genügt:
 - Die Beschreibung ist eindeutig bezüglich der Abfolge und der Einzelanweisungen.
 - Die Parameter, von denen der Algorithmus abhängt, sind vollständig spezifiziert.
 - Die Einzelanweisungen sind entweder elementar oder wieder durch einen Algorithmus beschrieben.
 - Die Einzelanweisungen sind immer ausführbar.
 - Die Beschreibung des Algorithmus ist endlich.
 - Die Anwendung des Algorithmus auf eine zulässige Eingabe der Spezifikation endet immer (Terminiertheit).

Zusammenhang von Spezifikation und Algorithmus

Konstruktionsproblem:

Wie findet man einen Algorithmus zu einer gegebenen Spezifikation ?

Zusammenhang von Spezifikation und Algorithmus

Verifikationsproblem:

Wie beweist man, dass ein gegebener Algorithmus eine gegebene Spezifikation erfüllt ?

Zusammenhang von Spezifikation und Algorithmus

Für folgende Spezifikationen gibt es keinen Algorithmus:

- 1) Gegeben eine beliebige Spezifikation:
Erstelle einen Algorithmus, der diese Spezifikation erfüllt.

→ Das Konstruktionsproblem ist nicht automatisierbar!

- 2) Gegeben eine beliebige Spezifikation und einen beliebigen Algorithmus:
Beweise, dass der Algorithmus die Spezifikation erfüllt

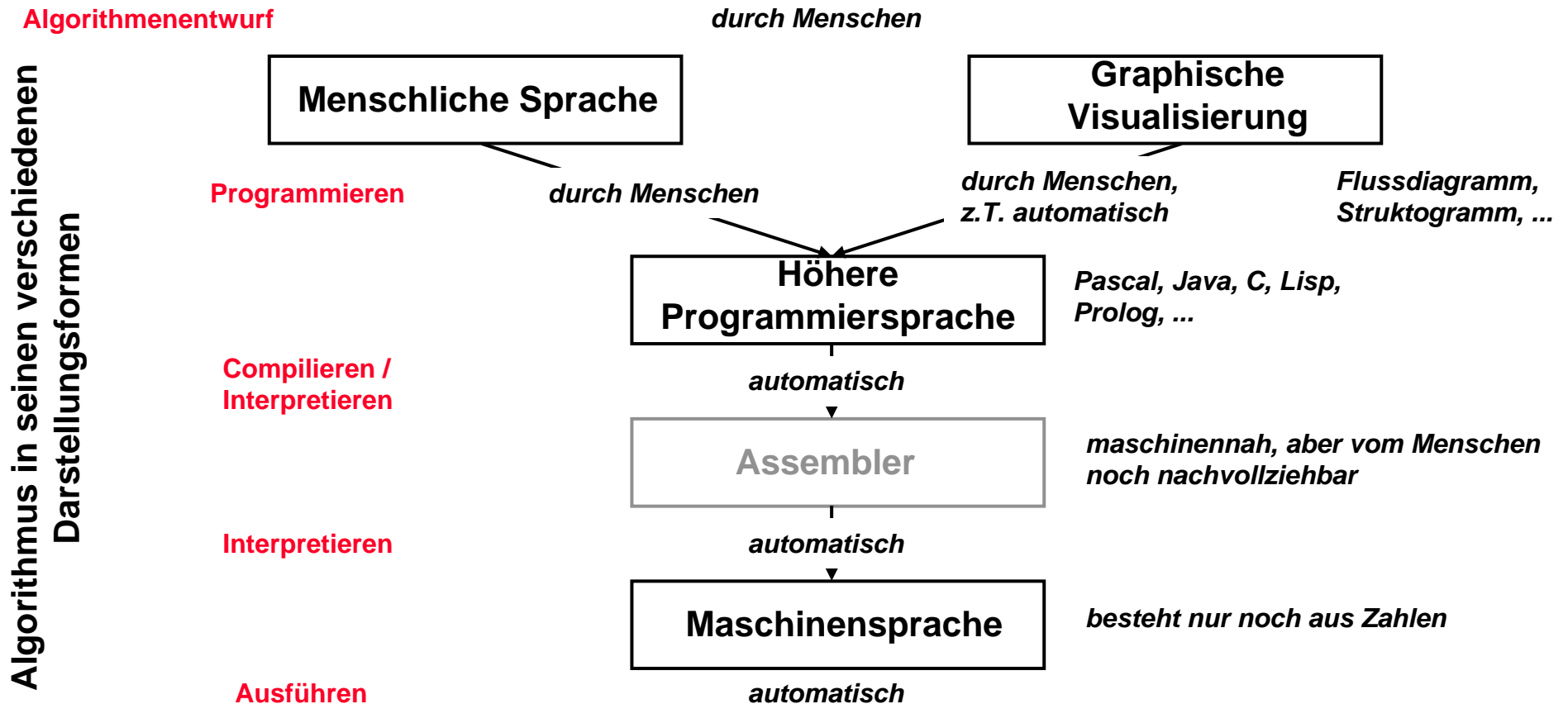
→ Das Verifikationsproblem ist nicht automatisierbar!

Folgerung: Kreativität bei Konstruktion und Verifikation ist unverzichtbar!

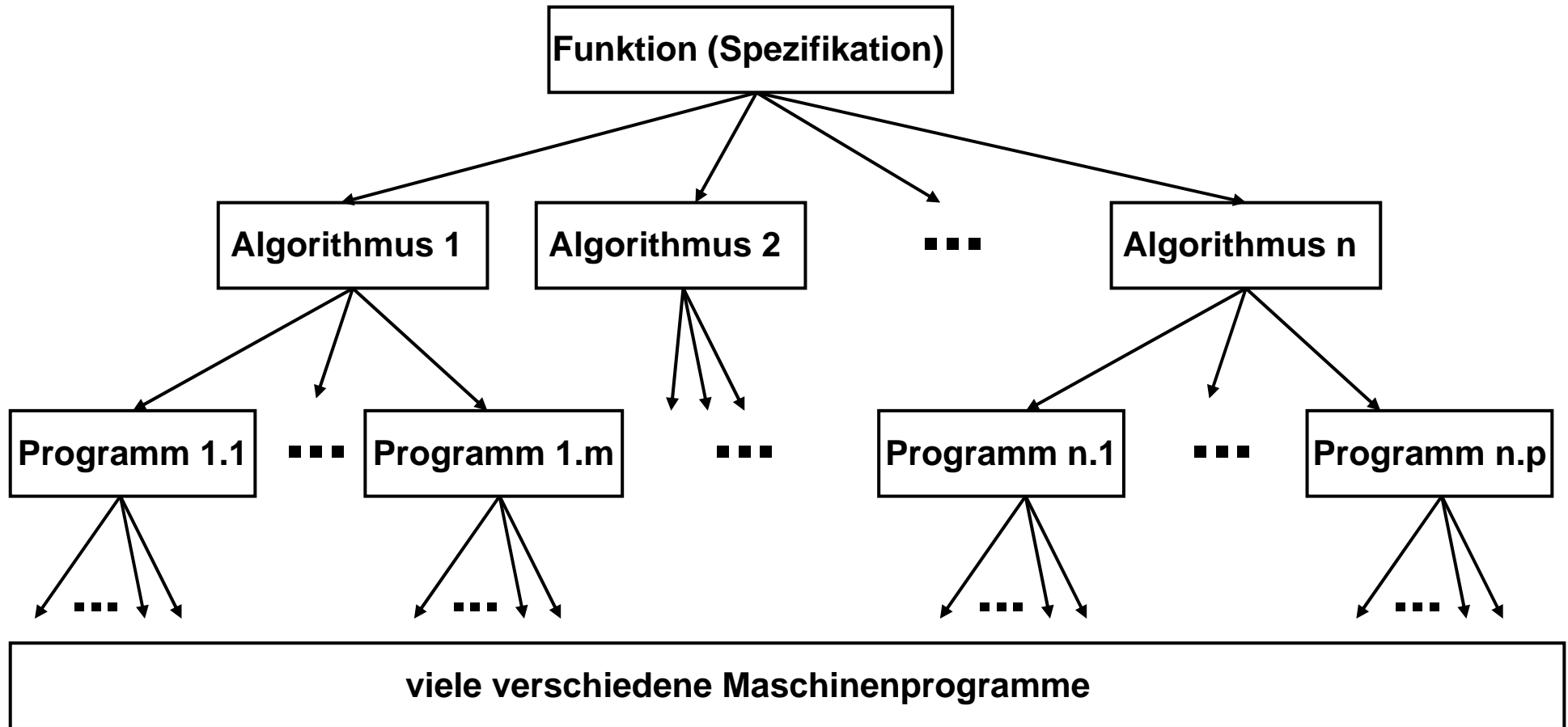
Zusammenhang von Algorithmus und Programm

Ein **Algorithmus** ist ein Konzept

Ein **Programm** ist die Formulierung eines Algorithmus in einer dem Computer verständlichen Sprache



Vielfalt der Möglichkeiten zwischen Spezifikation und Ausführung



Warum sollten wir so viele Möglichkeiten betrachten ?

Vielfalt der Möglichkeiten zwischen Spezifikation und Ausführung

- 1) **Warum sind mehrere Maschinenprogramme für dasselbe Programm sinnvoll ?**
- 2) **Warum sind mehrere Programme für denselben Algorithmus sinnvoll ?**
- 3) **Warum sind mehrere Algorithmen für dieselbe Spezifikation sinnvoll ?**

Wie bewertet man die Qualität von Algorithmen ?



- **Verständlichkeit**

nicht sinnvoll:

bewertet nicht das Konzept, sondern die Darstellung und damit das Programm



- **Schnelligkeit der Umsetzbarkeit (Implementierung)**

schon besser, besonders, wenn Entwicklungskosten teuer sind

hängt aber manchmal von der Wahl der Programmiersprache ab



- **Zeit- und Speicherplatzbedarf**

gut, aber wie ?



- **Laufzeitmessung mit Uhr und Speicherverbrauchszähler**

nicht gut: bewertet das konkrete Programm und auch das nicht einmal objektiv



- **Theoretische Analyse in Abhängigkeit von der Eingabegröße des Problems**

sehr gut: erfordert aber mathematische Abstraktion

➔ Näheres in Kapitel 4 dieser Vorlesung

Bausteine von Algorithmen

Jeder Algorithmus kann aus folgenden Kontrollstrukturen modular aufgebaut werden:

- Sequenz
- Verzweigung
- Schleife

Zum Lösen komplizierterer Probleme benutzt man zur Vereinfachung noch zwei weitere Techniken:

- Teilalgorithmen (Unterprogramme)
- Rekursion

➔ Ein Konzept zur Verifizierung muss nur für diese Kontrollstrukturen erstellt werden.

➔ Details in Kapitel 3 dieser Vorlesung

Beispiel für eine Programmverifikation

Gegeben sei folgender Algorithmus:

```
if (x>0) ∨ ((y+x)≤0)
  then
    z := x • y
  else
    z := x / y
```

Behauptung: Dieser Algorithmus ist für alle $x, y \in \mathbb{R}$ ausführbar

Beweis ?

Frage: Ist der Algorithmus auch korrekt ?

Ausblick auf die nächsten Vorlesungen

Wir haben im letzten Beispiel gesehen:

→ Das Gebiet der Programmverifikation erfordert einen sicheren Umgang mit formalen logischen Schlüssen

Es gilt allgemein für alle Aspekte der Programmierung:

! Grundkenntnisse der Aussagen- und Prädikatenlogik sind unentbehrlich. !

Daher befassen sich die nächsten Vorlesungen mit dem Thema:

Logik

→ Details im folgenden Kapitel 2 dieser Vorlesung