EINI LW

Einführung in die Informatik für Naturwissenschaftler und Ingenieure

Vorlesung 2 SWS WS 11/12

Dr. Lars Hildebrand
Fakultät für Informatik – Technische Universität Dortmund
lars.hildebrand@udo.edu
http://ls1-www.cs.uni-dortmund.de

technische universität dortmund

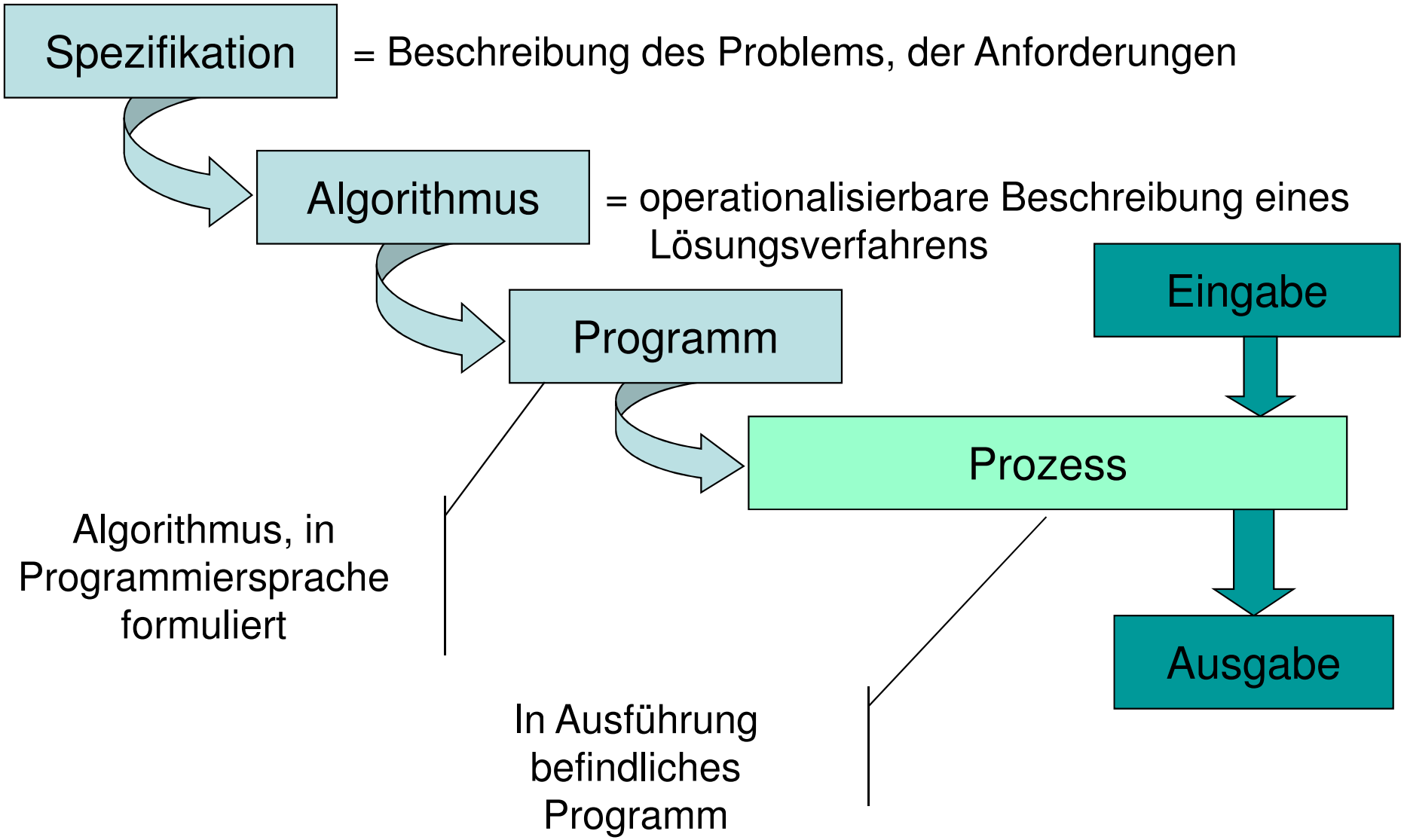
Dr. Lars Hildebrand

Gliederung

- Stationen im Entwurf von Algorithmen und Programmen
- Spezifikation
- Algorithmus
- Syntaxdiagramm
- Semantik

Unterlagen

- Gumm/Sommer: Einführung in die Informatik, Kap. 2
- Echtle/Goedicke: Einfg. in die Progr. mit Java, dpunkt Verlag


Spezifikation
= Beschreibung des Problems, der Anforderungen
Algorithmus
= operationalisierbare Beschreibung eines Lösungsverfahrens
Programm
Eingabe
Prozess
Ausgabe
Algorithmus, in Programmiersprache formuliert
In Ausführung befindliches Programm

Aufgabenstellung

Entwicklung eines Programms (Software), das ein Rechensystem, einen Rechner (Hardware) dazu befähigt, ein gegebenes Problem zu lösen.

Vorgehensweise

1. Das zu lösende Problem wird genau beschrieben

   -> **Spezifikation**

2. Ein Ablauf von Aktionen wird entworfen, der das Problem löst

   -> **Algorithmus**

3. Der entworfene Algorithmus wird in für Rechner ausführbare Form gebracht

   -> **Programm**

(A. Schürr, Universität der BW München)

Wie sollte eine Problembeschreibung sein ?

Dr. Lars Hildebrand

Beispiel: der größte gemeinsame Teiler zweier Zahlen

„Für beliebige Zahlen m und n berechne den größten gemeinsamen Teiler ggT(m,n), d.h. die größte Zahl, die sowohl m als auch n teilt."

Informelle Problembeschreibungen dieser Art haben oft Mängel

- **Vollständigkeit**: die Beschreibung lässt offen, welche Zahlen (als Eingabe) zugelassen sind (natürliche, rationale Zahlen, mit 0 oder ohne?)
- **Detailliertheit**: die Beschreibung lässt offen, welche Operationen (Befehle) zur Lösung des Problems verwendet werden dürfen (nur Addition, Subtraktion oder auch ganzzahlige Division und Restbildung)
- **Unzweideutigkeit**: die Beschreibung lässt offen, was „berechnen" heißt (soll das Ergebnis ausgegeben oder gespeichert werden?)
- **Widerspruchsfreiheit**: oft enthalten in natürlicher Sprache formulierte (informelle) Problembeschreibungen Widersprüche (Inkonsistenzen)

(A. Schürr, Universität der BW München)

Eigenschaften von Spezifikationen

Dr. Lars Hildebrand

„Eine Spezifikation ist eine vollständige, detaillierte, unzweideutige und widerspruchsfreie Problembeschreibung in einer präzise definierten Sprache.“

Sie ist:

- **vollständig**: wenn alle Anforderungen und relevanten Rahmenbedingungen angegeben worden sind.
- **detailliert**: wenn klar ist, welche Hilfsmittel zur Problemlösung zugelassen sind.
- **unzweideutig**: wenn klare Kriterien angegeben sind, wann eine berechnete Lösung zulässig ist.
- **widerspruchsfrei**: wenn verschiedene Teile der Problembeschreibung nicht unvereinbare Anforderungen an die Lösung stellen.

(A. Schürr, Universität der BW München)

technische universität dortmund

Dr. Lars Hildebrand

Gesucht wird eine Funktion ggT(m,n), die

- eine Zahl z berechnet (der Variablen z einen Wert zuweist)
- die die unten aufgeführte Nachbedingung erfüllt
- falls die folgende Vorbedingung für die Eingabewerte erfüllt ist.

Vorbedingung für zulässige Eingabewerte

- { m und n sind ganze Zahlen mit 0 < m < 65536, 0 < n < 65536 }

Nachbedingung für erwartete Ausgabewerte

- { z teilt m und z teilt n und für jedes z´ mit z´ teilt m und z´ teilt n gilt: z´ ist kleiner oder gleich z }

Annahme

- Die genaue Bedeutung von „x teilt y“ ist bekannt.

(A. Schürr, Universität der BW München)

Definition des Begriffs Algorithmus

Dr. Lars Hildebrand

Definition 1 (imperative = befehlsorientierte Variante, nach Gumm/Sommer):

„Ein Algorithmus ist eine detaillierte und explizite Vorschrift zur schrittweisen Lösung eines Problems durch eine Abfolge bekannter Befehle/Operationen."

Definition 2 (funktionale Variante, nach Schürr, UniBW München):

„Ein Algorithmus ist eine Vorschrift, die detailliert beschreibt, wie man allen erlaubten Eingabewerten einer Funktion den „richtigen" Ausgabewert zuordnet."

Typische Beispiele für Algorithmen aus dem Alltag:

- Kochrezepte
- Gebrauchsanweisungen
- Strickanleitungen
- ...

Definition eines Algorithmus nach Echtle/Goedicke

Dr. Lars Hildebrand

A1

- Ein Algorithmus beschreibt eine Relation über dem Kreuzprodukt einer Eingabe- und einer Ausgabemenge. Dadurch werden für jede Eingabe die zulässigen Ausgaben festgelegt.

A2

- Ein Algorithmus setzt sich aus wohldefinierten Elementaroperationen zusammen, die auf einer geeigneten Maschine ausführbar sind.

A3

- Ein Algorithmus legt die Abfolge der Schritte fest, wobei jeder Schritt genau eine Elementaroperation umfasst.

A4

- Ein Algorithmus ist eine Beschreibung endlicher Länge.

A5

- Ein Algorithmus benutzt nur endlich viele Speicherplätze zur Ablage von Zwischenergebnissen.

Es werden in der Regel weitere Forderungen an Algorithmen gestellt

A6 Terminierung

- Für jede (!) Eingabe endet die Ausführung des Algorithmus nach endlich vielen Schritten.

A7 Begrenzte Schrittanzahl

- Für jede (!) Eingabe wird die zugehörige Ausgabe spätestens nach Ausführung einer vorgegebenen Schrittanzahl n geliefert. Wenn ein Rechensystem für jeden Schritt höchstens die Zeit s benötigt, dann wird die Ausgabe spätestens nach Verstreichen der begrenzten Antwortzeit t = s * n geliefert.

Gelegentlich werden die Forderungen A6 bzw. A7 auf einzelne Programmabschnitte beschränkt

Deterministisch vs. nicht deterministisch

Dr. Lars Hildebrand

```
Schritt 1:   Lies Eingaben x und y,   weiter mit Schritt 2
Schritt 2:   Falls x < y:             weiter mit Schritt 3,
             falls x > y:                  weiter mit Schritt 4
Schritt 3:   Berechne a = y - x,      weiter mit Schritt 5
Schritt 4:   Berechne a = x - y,      weiter mit Schritt 5
Schritt 5:   Schreibe Ausgabe a,      beende Ausführung
```

Deterministischer Algorithmus

Lehrbuch der Programmierung mit Java, Echtle Goedicke, Heidelberg, © dpunkt 2000

```
Schritt 1:   Lies Eingaben x und y,   weiter mit Schritt 2 oder 3
Schritt 2:   Berechne a = x - y,      weiter mit Schritt 4
Schritt 3:   Berechne a = y - x,      weiter mit Schritt 4
Schritt 4:   Falls a > 0:                  weiter mit Schritt 5,
             falls a < 0:                  weiter mit Schritt 6
Schritt 5:   Setze b = a,             weiter mit Schritt 7
Schritt 6:   Berechne b = -a,         weiter mit Schritt 7
Schritt 7:   Schreibe Ausgabe b,      beende Ausführung
```

Indeterministischer Algorithmus

Lehrbuch der Programmierung mit Java, Echtle Goedicke, Heidelberg, © dpunkt 2000

Indeterminismus

- es muss nur feststehen, dass irgendeine Elementaroperation ausgeführt werden kann (A3)

Forderung nach Determiniertheit des Ergebnisses:

A8 Determiniertheit:

- Die Eingabe-Ausgabe-Relation (siehe A1) ist rechtseindeutig. Dies bedeutet, dass jeder Eingabe genau eine Ausgabe zugeordnet wird.

A9 Determinismus:

- In jedem Zustand, der bei Ausführung des Algorithmus erreicht wird, ist jeweils nur ein einziger Folgeschritt als nächster ausführbar.

- Die Forderung A9 impliziert A8 …

Beispiele

- Die Addition (2+2) oder die Einkommensteuerberechnung sollten determiniert sein
- Achtung: die konkrete Abfolge der Schritte ist damit nicht festgelegt!
- Die Reservierung von Flugsitzen von verschiedenen Buchungsterminals aus ist in der Regel nicht determiniert

Beispiele (cont.)

Dr. Lars Hildebrand

```
Schritt 1:   Lies Eingaben x und y,   weiter mit Schritt 2 oder 3
Schritt 2:   Berechne a = x - y,      weiter mit Schritt 4
Schritt 3:   Berechne a = y - x,      weiter mit Schritt 4
Schritt 4:   Falls a > 0: weiter mit Schritt 5,
             falls a < 0: weiter mit Schritt 6
Schritt 5:   Setze b = a,             weiter mit Schritt 7
Schritt 6:   Berechne b = -a,         weiter mit Schritt 7
Schritt 7:   Schreibe Ausgabe b,      beende Ausführung
```

Indeterministischer Algorithmus

Lehrbuch der Programmierung mit Java, Echtle Goedicke, Heidelberg, © dpunkt 2000

- ist *indeterministisch* aber trotzdem *determiniert*!
- Softwaresysteme, die die Arbeit mehrerer Rechner einschließen, sind in der Regel indeterministisch und müssen mit großem Aufwand zu determinierten Verfahren gemacht werden.

technische universität dortmund

Dr. Lars Hildebrand

A10 Allgemeinheit

- Ein Algorithmus löst nicht nur ein einziges Problem, sondern eine Klasse von Problemen.

A11 Änderbarkeit

- Ein Algorithmus soll sich leicht modifizieren lassen, um ihn an eine veränderte Aufgabenstellung anzupassen.

A12 Effizienz

- Für eine gegebene Eingabe soll die Anzahl der benötigten Schritte möglichst gering sein.

A13 Robustheit

- Der Algorithmus soll sich möglichst auch dann wohldefiniert verhalten, wenn eine unzulässige Eingabe (die nicht Element der Eingabemenge ist) vorliegt oder eine sonstige unvorhergesehene Situation auftritt.

technische universität dortmund

Dr. Lars Hildebrand

Forderungen A10 - A13 sind nicht immer leicht zu erfüllen und müssen auch gegeneinander abgewogen werden

```
Schritt 1:   Lies Eingaben x und y,   weiter mit Schritt 2
Schritt 2:   Berechne a = x + y,      weiter mit Schritt 3
Schritt 3:   Berechne b = a / 2,      weiter mit Schritt 4
Schritt 4:   Schreibe Ausgabe b,      beende Ausführung
```

Berechnung des arithm. Mittels nach der Formel (x+y)/2

Lehrbuch der Programmierung mit Java, Echtle Goedicke, Heidelberg, © dpunkt 2000

```
Schritt 1:   Lies Eingaben x und y,   weiter mit Schritt 2
Schritt 2:   Berechne a = 0.5 * x,    weiter mit Schritt 3
Schritt 3:   Berechne b = 0.5 * y,    weiter mit Schritt 4
Schritt 4:   Berechne c = a + b,      weiter mit Schritt 5
Schritt 5:   Schreibe Ausgabe c,      beende Ausführung
```

*Berechnung des arithm. Mittels nach der Formel 0.5*x + 0.5*y*

Lehrbuch der Programmierung mit Java, Echtle Goedicke, Heidelberg, © dpunkt 2000

Anmerkungen

- Forderungen A10 - A13 werden gelegentlich auch als „weich“ bezeichnet. Sie sind deswegen nicht minder wichtig!
- Beide Algorithmen oben lösen – mathematisch gesehen – ein und dasselbe Problem.
- Der erste Algorithmus erfüllt Forderung A11 besser, weil kürzer und übersichtlicher formuliert.
- Für Forderung A13 ist der zweite Algorithmus die günstigere Variante, da große Eingabewerte nicht so schnell Rechner-arithmetikprobleme aufwerfen.
- Insgesamt ist die Verständlichkeit von Software (heute) zentral! Warum?

technische universität dortmund

Dr. Lars Hildebrand

Beispiel (nicht wirklich passend)

- Obfuscated C Code Contest: Best one-liner 2001
  - Jens Schweikhardt, Weinstadt

```
main(int
  c,char**v){return!m(v[1],v[2]);}m(char*s,char*
  t){return*t-42?*s?63==*t|*s==*t&&m(s+1,t+1):
  !*t:m(s,t+1)||*s&&m(s+1,t);}
```

- This one-liner program is a glob pattern matcher. It understands the glob characters `*' meaning `zero or more characters' and `?' meaning exactly one character, just like your unix shell.

Beispiele für Eigenschaften von Algorithmen

Dr. Lars Hildebrand

- **Einfache Grundoperation**
  „Schneide das Fleisch in kleine Würfel“; es wird vorausgesetzt, dass der Leser weiß, wie man Fleisch in kleine Würfel schneidet.

- **Sequentieller Algorithmus**
  „Bringe das Wasser zum Kochen, dann gib das Paket Nudeln hinzu“; die Reihenfolge der Ausführung der Operationen ist festgelegt.

- **Nebenläufiger Algorithmus**
  „Schneide Fleisch und Gemüse“; Fleisch und Gemüse können gleichzeitig geschnitten werden oder in beliebiger Reihenfolge.

- **Parallele Ausführung**
  „Ich schneide das Fleisch und Du das Gemüse“; die Operationen werden tatsächlich gleichzeitig (parallel) ausgeführt und nicht hintereinander (sequentiell) in beliebiger Reihenfolge.

- **Nichtdeterministischer/nichtdeterminierter Algorithmus**
  „Man nehme Schweine- oder Kalbfleisch“; je nachdem, wie man sich entscheidet, ist das erzeugte Ergebnis (Gericht) ein anderes.

(A. Schürr, Universität der BW München)

Naheliegende Frage

Ist die folgende Charakterisierung der Rechenvorschrift von Euklid zur Berechnung des ggT bereits ein Algorithmus ?

1. z = ggT(z,z)
2. z = ggT(m,n) falls gilt: m < n und z = ggT(m,n-m)
3. z = ggT(m,n) falls gilt: m > n und z = ggT(m-n,m)

- Nein, da zunächst unklar ist, wie man aus der obigen Beschreibung eine Anleitung zur Ausführung von Rechenoperationen ableitet.
- Ja, da die drei angegebenen Zeilen bereits (fast) ein Programm in den Programmiersprachen Prolog oder Lisp sind, die die sogenannte logische Programmierung unterstützt.

Vom Algorithmus zum Programm

Dr. Lars Hildebrand

- Die Beschreibung eines Algorithmus kann in einer beliebigen Sprache erfolgen.
- Praktisch ausführbare Algorithmen formuliert man in **algorithmischen Sprachen.**
- Ist eine solche (algorithmische) Sprache zusätzlich auf die Bedürfnisse der Ausführung auf einem Rechensystem (z.B. „von Neumann Rechner“) zugeschnitten, so heißt sie **Programmiersprache**.
- Die Formulierung eines Algorithmus in einer Programmiersprache heißt **Programm**, das Entwerfen eines Programms entsprechend **Programmieren**.
- Es gibt verschiedene Klassen von Programmiersprachen, die ein sogenanntes **Programmierparadigma** (Konzept der Programmierung) unterstützen.

Die bekanntesten Programmierparadigmen

Dr. Lars Hildebrand

- Imperative (prozedurale) Programmierung
  - Sprachen: Pascal,C,Fortran, Cobol, PL/1, VisualBasic, ...
  - Anweisungen verändern Werte von Variablen.
  - Kontrollstrukturen regeln Reihenfolge der Ausführung von Anweisungen.
  - Prozeduren definieren wiederverwendbare Kontrollstrukturen.

- Funktionale Programmierung
  - Sprachen: Lisp, Haskell, ML, Scheme, ...
  - Ein Programm besteht aus Funktionsdefinitionen.
  - Jede Funktion wird durch einen Ausdruck definiert.
  - Die Programmausführung besteht aus der Anwendung von Funktionen auf Ausdrücke (Terme), dem sog. Lambda-Kalkül.

Beispiel Scheme:

```
(define (fakultaet n)
  (if (= n 0)
      1
      (* n (fakultaet (- n 1)))))
```

technische universität dortmund

Dr. Lars Hildebrand

- Logische Programmierung
  - Sprache: Prolog, ...
  - Ein logisches Programm besteht aus immer wahren Aussagen und Regeln zur Ableitung weiterer Aussagen.
  - Die Programmausführung wird durch eine Anfrage gestartet, ob (unter welchen Bedingungen) eine bestimmte Aussage wahr ist.

Beispiel Prolog:

```
%Fakten
weiblich(Elizabeth).
maennlich(Philip).
elternteil_von(Elizabeth,Charles).
elternteil_von(Philip, Charles).
%Regeln
kind_von(person1, person2):-
elternteil_von(person2,person1).
vater_von(person1,person2):-
elternteil_von(person1,person2),maennlich(person1).
%Anfrage1
?-vater_von(Elizabeth,Charles).
No
```

Die bekanntesten Programmierparadigmen

Dr. Lars Hildebrand

- Objektorientierte Programmierung
  - Sprachen: Java, C++, Smalltalk, ...
  - ergänzt die imperative Programmierung:
    - Daten (Werte) und Operationen (Prozeduren, Methoden) werden in Objekten zusammengefasst.
    - Objekte schicken sich Botschaften zu, die die Ausführung von Operationen auslösen.
    - Klassen beschreiben Mengen sich gleich verhaltender Objekte.

technische universität dortmund

Dr. Lars Hildebrand

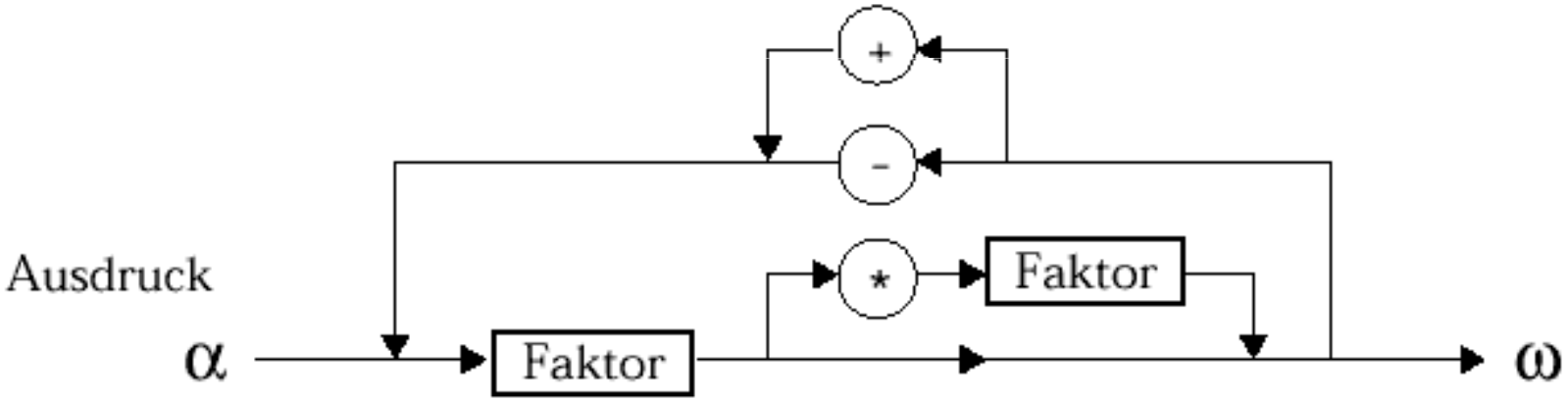
Gliederung

- Stationen im Entwurf von Algorithmen und Programmen
- Wie geht es weiter
- Spezifikation
- Algorithmus
- Syntaxdiagramm
- Semantik

technische universität dortmund
Dr. Lars Hildebrand

als grafische Alternative zur Beschreibung von Programmiersprachen

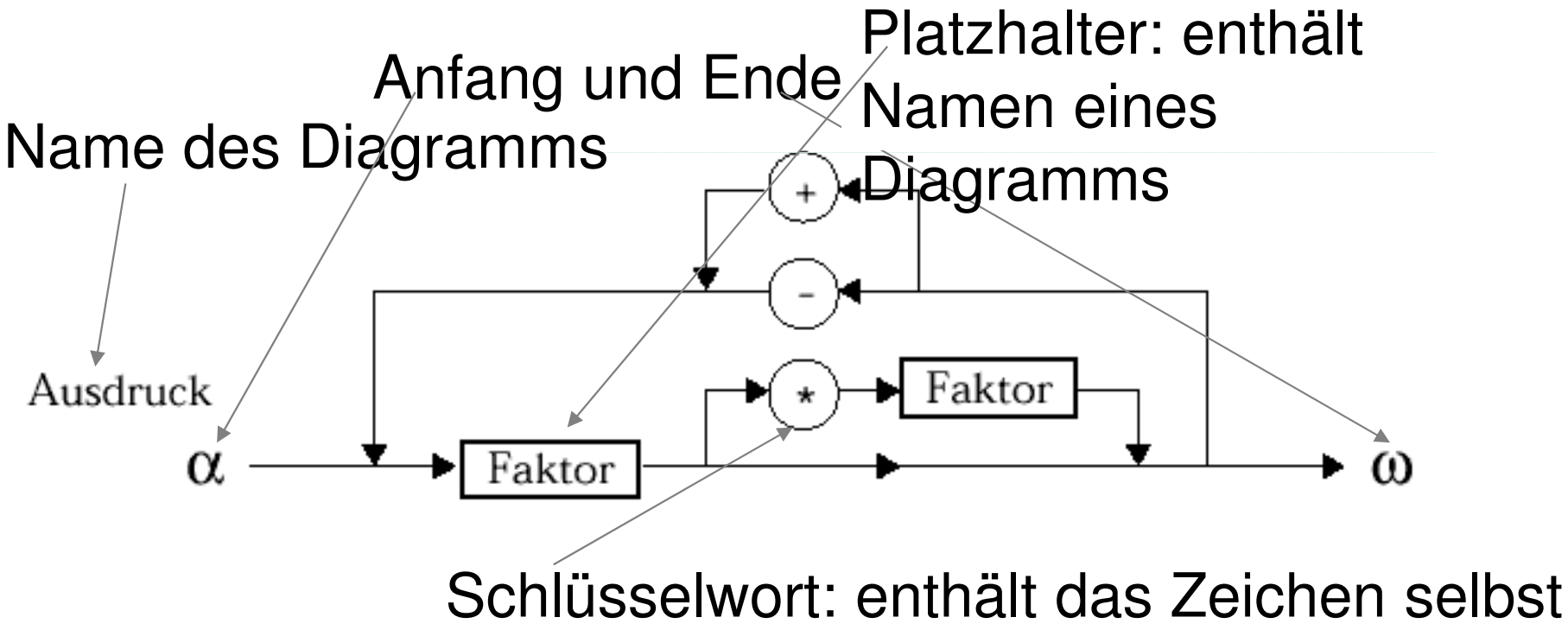
- bestehen aus
  - zwei unterschiedlichen Arten von **Kästen**
    - **rund = "Schlüsselwörter"** und
    - **eckig = "Platzhalter"** und
  - **Pfeilen,** die diese Kästen miteinander verbinden



(©M. Goedicke, UGH Essen)

Darstellung von Regel(teil)mengen

Dr. Lars Hildebrand

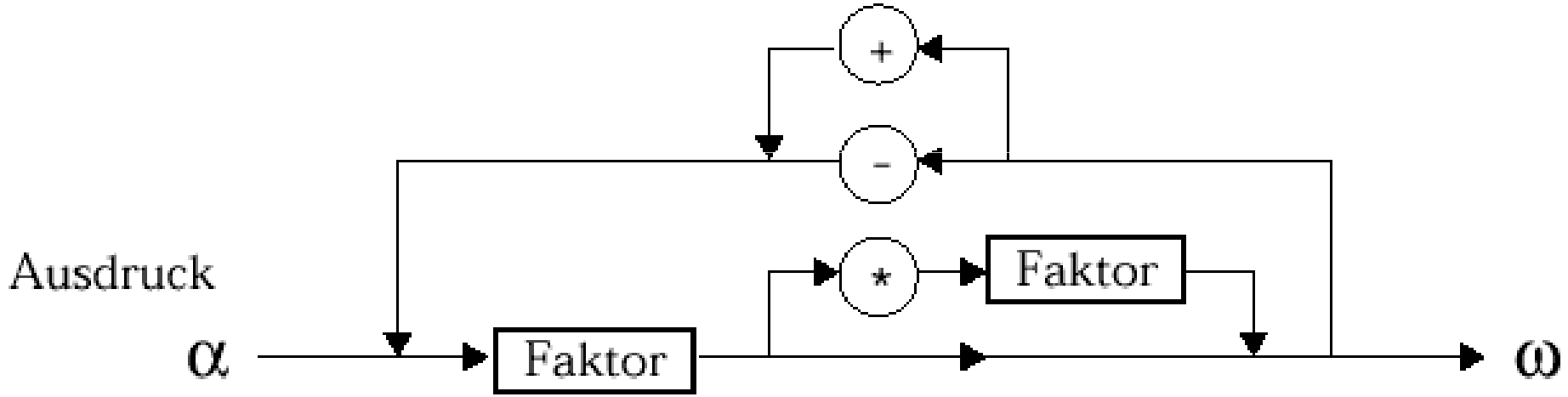


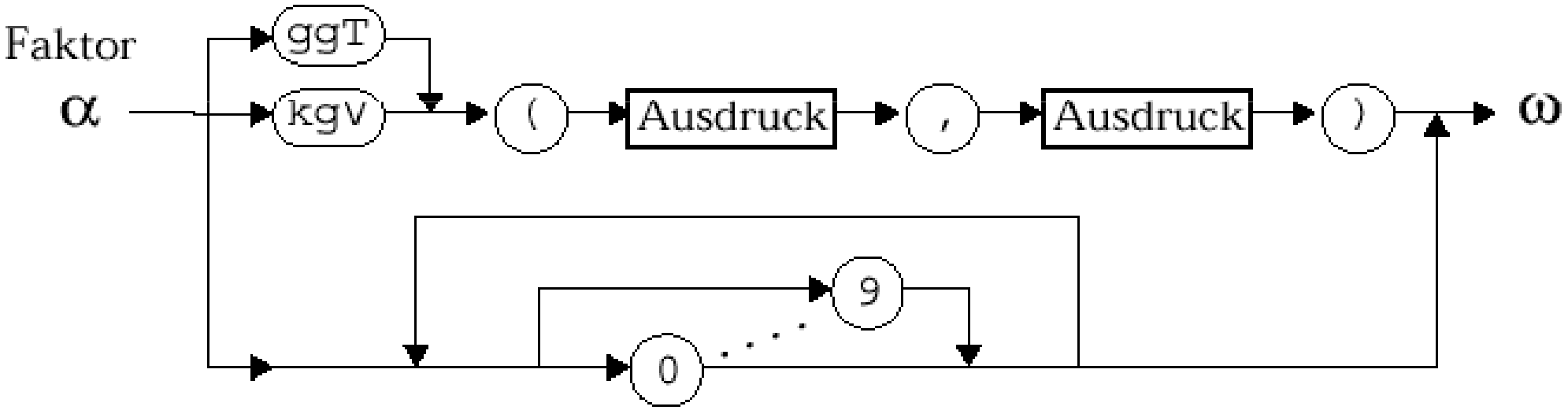
Beim Durchlaufen durch ein Diagramm entlang der Pfeile werden an den Schlüsselwort-Kästen Zeichen aufgesammelt und bei Platzhalter-Kästen zu dem angegebenen Diagramm verzweigt.

(©M. Goedicke, UGH Essen)

Die gesamte Regelmenge

dargestellt über Syntaxdiagramme (ohne Summanden)





(©M. Goedicke, UGH Essen)

Wo stehen wir jetzt?

- **Gliederung**

- Stationen im Entwurf von Algs und Progs
- Wie geht es weiter
- Spezifikation
- Algorithmus
- Syntaxdiagramm
- Semantik

(©M. Goedicke, UGH Essen)

Bemerkung zur Semantik

Die Informatik benutzt üblicherweise drei Möglichkeiten, die **Bedeutung einer formalen Sprache** (hier Programmiersprache) zu **beschreiben.**

- operational
- denotational
- verbal

(©M. Goedicke, UGH Essen)

technische universität dortmund

Dr. Lars Hildebrand

- Die **operationale** Methode definiert schrittweise die Wirkung von Elementaroperationen
  - Beschreibe elementweise, wie die **Elementaroperationen** in den verschiedenen **Situationen** ausgeführt werden
  - D.h. man unterscheidet
    - die Elementaroperationen und
    - Programmsituationen
  - Beides zusammen definiert, wie ein Programm schrittweise ausgeführt wird.
- Auf dieser Basis werden Softwareentwicklungswerkzeuge (Compiler) hergestellt.

(©M. Goedicke, UGH Essen)

technische universität dortmund
Dr. Lars Hildebrand

- Die **denotationale** Methode definiert die Wirkung von Programmen durch eine mathematische Funktion:
  - Die Wirkung (= Bedeutung) eines Programms wird durch die Veränderung von Zuständen beschrieben
  - Programm : Zustand, Eingabe → Zustand
- Auf dieser Basis werden formale Korrektheitsbeweise (tut ein Programm das, was es soll?) geführt.

(©M. Goedicke, UGH Essen)

Verbal Semantik

- Die **verbale** Methode definiert die Wirkung von Programmen durch eine präzise verbale Erklärung
  - Die Wirkung (= Bedeutung) eines Programms wird durch die verbale Beschreibung der einzelnen Sprachelemente der betrachteten Programmiersprache geliefert.
  - Java: Die so genannte Java Language Reference
  - ist eher ein technisches Dokument ... gedacht als Nachschlagewerk für Hersteller von Softwareentwicklungs-werkzeugen

- Auf dieser Basis wird die Programmiersprache Java hier in der Vorlesung eingeführt

(©M. Goedicke, UGH Essen)

Wo stehen wir jetzt?

- **Ziel** demnächst:
  - Betrachtung wesentlicher Sprachkonstrukte imperativer Programme.
- **Rückblick:**
  - **Spezifikation**sbegriff: Was sind die Eigenschaften einer Problem-/ Aufgabenbeschreibung
  - **Algorithmus**begriff: Was sind die Eigenschaften automatischer Verfahrensvorschriften
  - **Syntaxdiagramme & Semantik**: Wie können künstliche Sprachen definiert werden
- **Nächster Schritt:**
  - **Basiskonstrukte** imperativer & objektorientierter Programmierung
    - Zuweisung und Datentypen
    - Sequenz
    - Fallunterscheidung, Alternative
    - Iteration
    - Verfeinerung, Unterprogrammaufrufe, Prozedur
    - Funktion und Rekursion

technische universität dortmund

Dr. Lars Hildebrand

Vielen Dank für Ihre Aufmerksamkeit!

Nächste Termine

- Nächste Vorlesung 4.11.2011, 08:30