EINI
LogWing/WiMa/MP

Einführung in die Informatik für Naturwissenschaftler und Ingenieure

Vorlesung 2 SWS WS 20/21

Dr. Lars Hildebrand
Fakultät für Informatik – Technische Universität Dortmund
lars.hildebrand@tu-dortmund.de
http://ls14-www.cs.tu-dortmund.de

Thema

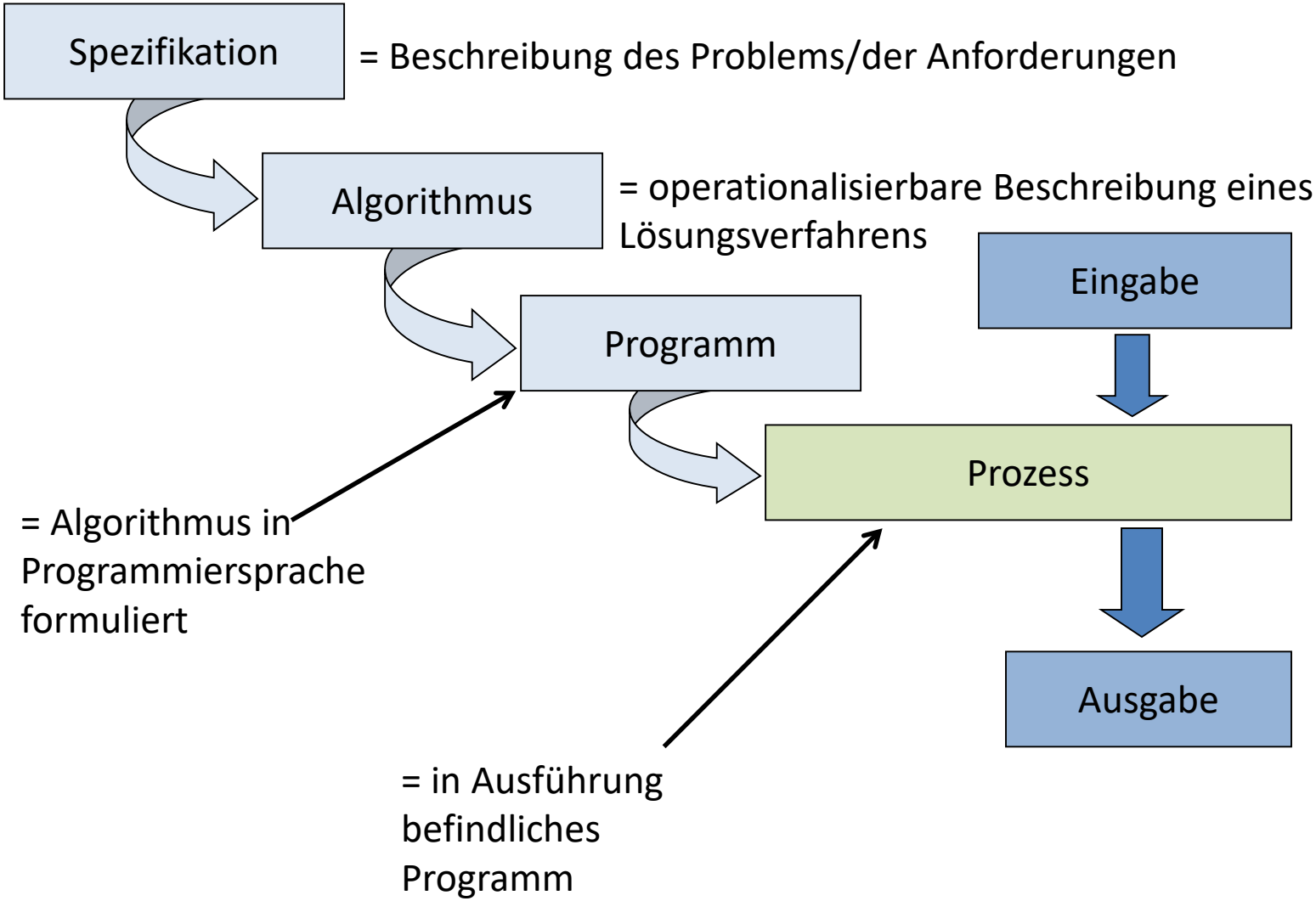
Gliederung

- Stationen im Entwurf von Algorithmen und Programmen
- Spezifikation
- Algorithmus
- Syntax(diagramm)
- Semantik

Unterlagen

- Echtle, Klaus und Michael Goedicke: *Lehrbuch der Programmierung mit Java*. Heidelberg: dpunkt-Verl, 2000. (→ ZB)
- Gumm, Heinz-Peter und Manfred Sommer: *Einführung in die Informatik*, 10. Auflage. München: De Gruyter, 2012. (Kap. 2) (→ Volltext aus Uninetz)

Stationen im Entwurf von Algorithmen und Programmen



In diesem Kapitel:

- **Prolog**
- Spezifikation
- Algorithmus
- Syntax, Semantik

Spezifikationen und Algorithmen

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 (nicht) sein?

„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.“

→ Informell: 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).

(nach A. Schürr, Universität der BW München)

Eigenschaften von Spezifikationen

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.

(nach A. Schürr, Universität der BW München)

Spezifikation der ggT-Berechnung

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

- eine Zahl z berechnet (der Variablen z einen Wert zuweist) und
- 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´ <= z }

Annahme:

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

(A. Schürr, Universität der BW München)

Spezifikation

Artikel im EINI-Wiki:

→ **Probleme** (Spezifikation)

Zwischenstand

Gliederung

- ✓ Stationen im Entwurf von Algorithmen und Programmen
- ✓ Spezifikation
- ➢ Algorithmus
- ► Syntax(diagramme)
- ► Semantik

In diesem Kapitel:

- Prolog
- Spezifikation
- **Algorithmus**
- Syntax, Semantik

Definition des Begriffs Algorithmus

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].“
>
> Gumm/Sommer, S. 91.

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 I

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.

Definition eines Algorithmus nach Echtle/Goedicke II

Weitere Forderungen an Algorithmen:

A6: Für jede (!) Eingabe endet die Ausführung des Algorithmus nach endlich vielen Schritten (**Terminierung)**.

A7 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 (**Begrenzte Schrittanzahl)**.

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

Deterministisch vs. nicht deterministisch

Deterministischer Algorithmus

```
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
```

Echtle/Goedicke, Heidelberg: *Prog. 1–1*, S. 8 © dpunkt 2000.

Indeterministischer Algorithmus

```
Schritt 1:  Lies Eingaben x und y,     weiter mit Schritt 2
                                        oder Schritt 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
```

Echtle/Goedicke, Heidelberg: *Prog. 1–2*, S. 8 © dpunkt 2000.

Freiheiten bei der Festlegung der Reihenfolge

Indeterminismus:

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

Forderung nach Determiniertheit des Ergebnisses:

A8: Die Eingabe-Ausgabe-Relation (siehe A1) ist rechtseindeutig. Dies bedeutet, dass jeder Eingabe genau eine Ausgabe zugeordnet wird (**Determiniertheit**).

A9: In jedem Zustand, der bei Ausführung des Algorithmus erreicht wird, ist jeweils nur ein einziger Folgeschritt als nächster ausführbar (**Determinismus**).

- Die Forderung A9 impliziert A8!

Beispiele

- Die Addition (42+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.
- Algorithmen zur Erzeugung von (Pseudo-) Zufallszahlen sind nicht deterministisch.

Beispiele

Indeterministischer Algorithmus

```
Schritt 1: Lies Eingaben x und y,     weiter mit Schritt 2
                                       oder Schritt 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
```

Echtle/Goedicke, Heidelberg: *Prog. 1–2*, S. 8 © dpunkt 2000.

- **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.

Forderungen an Algorithmen

A10: Ein Algorithmus löst nicht nur ein einziges Problem, sondern eine Klasse von Problemen **(Allgemeinheit)**.

A11: Ein Algorithmus soll sich leicht modifizieren lassen, um ihn an eine veränderte Aufgabenstellung anzupassen **(Änderbarkeit)**.

A12: Für eine gegebene Eingabe soll die Anzahl der benötigten Schritte möglichst gering sein **(Effizienz)**.

A13: 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 (**Robustheit)**.

Weitere Forderungen an Algorithmen

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

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

```
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
```

Echtle/Goedicke, Heidelberg: *Prog. 1–3*, S. 10 © dpunkt 2000.

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

```
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
```

Echtle/Goedicke, Heidelberg: *Prog. 1–4*, S. 10 © dpunkt 2000.

Anmerkungen

- Forderungen A10 - A13 werden manchmal auch als „weich“ bezeichnet. Sie sind deswegen nicht minder wichtig!

- Beide Algorithmen der vorhergehenden Folie lösen – mathematisch gesehen – ein und dasselbe Problem. Aber:

  - Der erste Algorithmus erfüllt Forderung **A11** besser, da er kürzer und übersichtlicher formuliert ist.

  - Für Forderung **A13** ist der zweite Algorithmus die bessere Variante, da große Eingabewerte nicht so schnell Rechner-arithmetikprobleme aufwerfen.

    - Allerdings gilt diese Robustheit nur für kommabehaftete Zahlen. Für ganze Zahlen kann der vermeintlich robuste Algorithmus versagen!

(Un)Verständlichkeit von Algorithmen

Beispiel

- 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,ch
ar*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 die Eigenschaften von Algorithmen I

- **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.

Beispiele für die Eigenschaften von Algorithmen II

- **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.

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 unterstützen.

Vom Algorithmus zum Programm

- 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

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

Die bekanntesten Programmierparadigmen

- **Funktionale Programmierung:**
  - 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.
  - Sprachen: Lisp, Haskell, ML, Scheme, …

Beispiel (Scheme):

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

Die bekanntesten Programmierparadigmen

- **Logische Programmierung:**
  - 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.
  - Sprachen: Prolog, ...

  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),maen
nlich(person1).
%Anfrage1
?-vater_von(Elizabeth,Charles).
No
```


In diesem Kapitel:
- Prolog
- Spezifikation
- **Algorithmus**
- Syntax, Semantik

Die bekanntesten Programmierparadigmen

- **Objektorientierte Programmierung:**
  - 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.
  - Sprachen:
    - Java
    - C++
    - Smalltalk
    - ...

Algorithmus

Artikel im EINI-Wiki:

- → **Algorithmus** (Anforderungen an einen Algorithmus)
- → **Programm**
- → **Programmiersprache**
- → **Imperative Programmierung**
- → **Funktionale Programmierung**
- → **Logische Programmierung**
- → **Objektorientierte Programmierung**

Zwischenstand

Gliederung

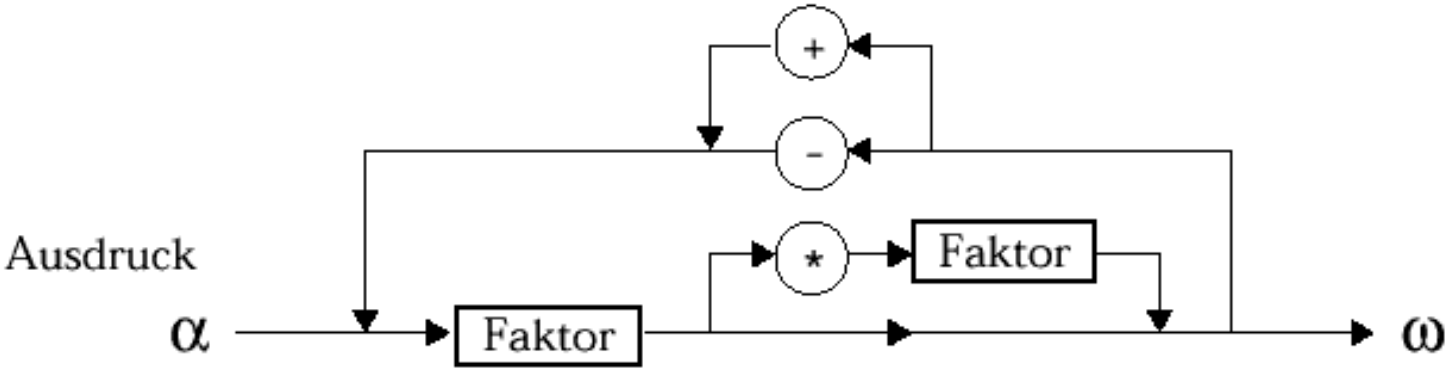
- ✓ Stationen im Entwurf von Algorithmen und Programmen
- ✓ Spezifikation
- ✓ Algorithmus
- ➢ Syntax(diagramme)
- ► Semantik

In diesem Kapitel:

- Prolog
- Spezifikation
- **Algorithmus**
- Syntax, Semantik

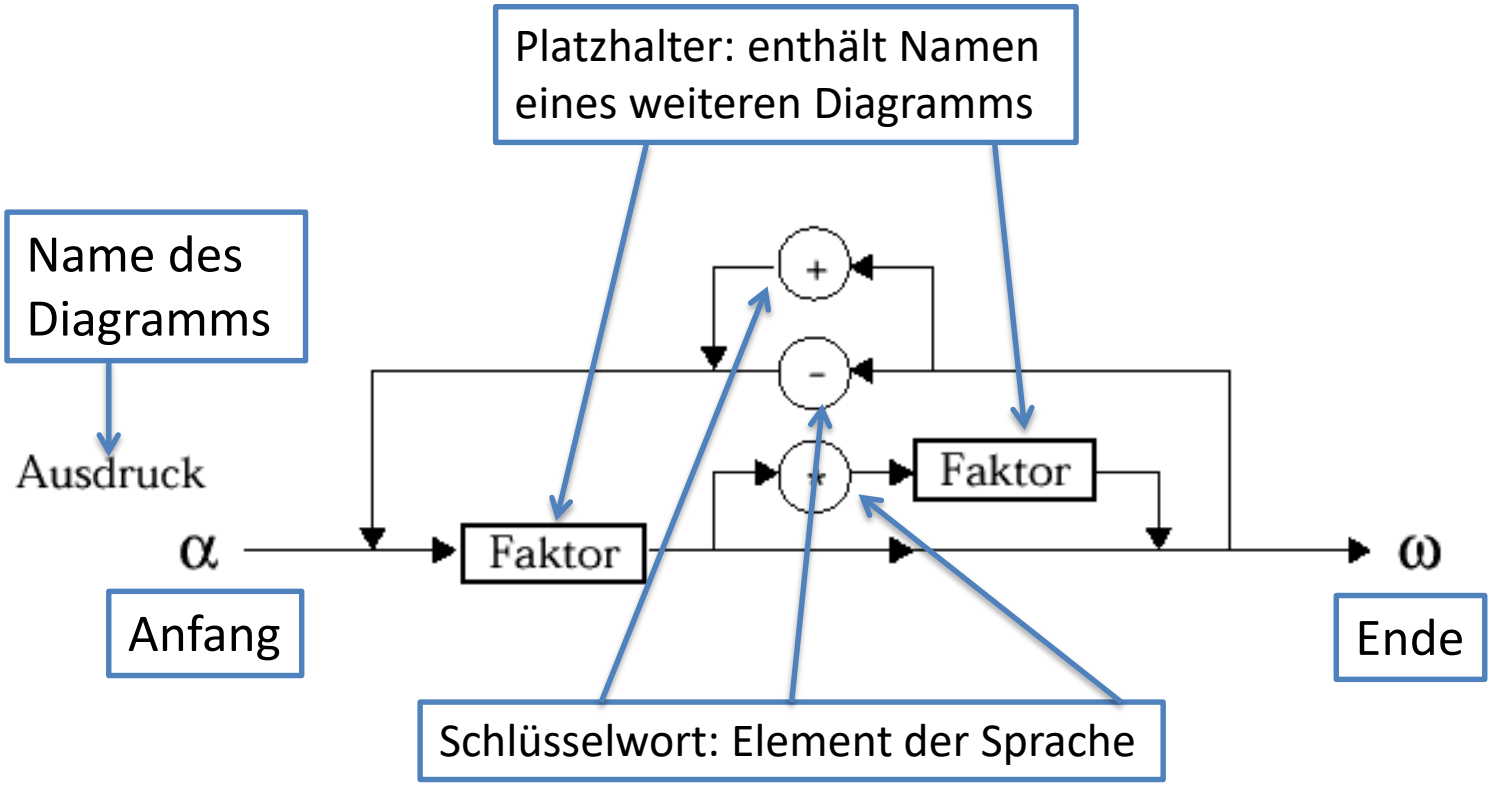
Syntaxdiagramme

- grafische Alternative zur Beschreibung von Programmiersprachen
- bestehen aus
  - zwei unterschiedlichen Arten von **Kästen**
    - **rund = "Schlüsselwörter"**
    - **eckig = "Platzhalter"**
  - **Pfeilen,** die diese Kästen miteinander verbinden



Echtle/Goedicke, Heidelberg: *Abb. 1–3* (Ausschnitt), S. 15 © dpunkt 2000.

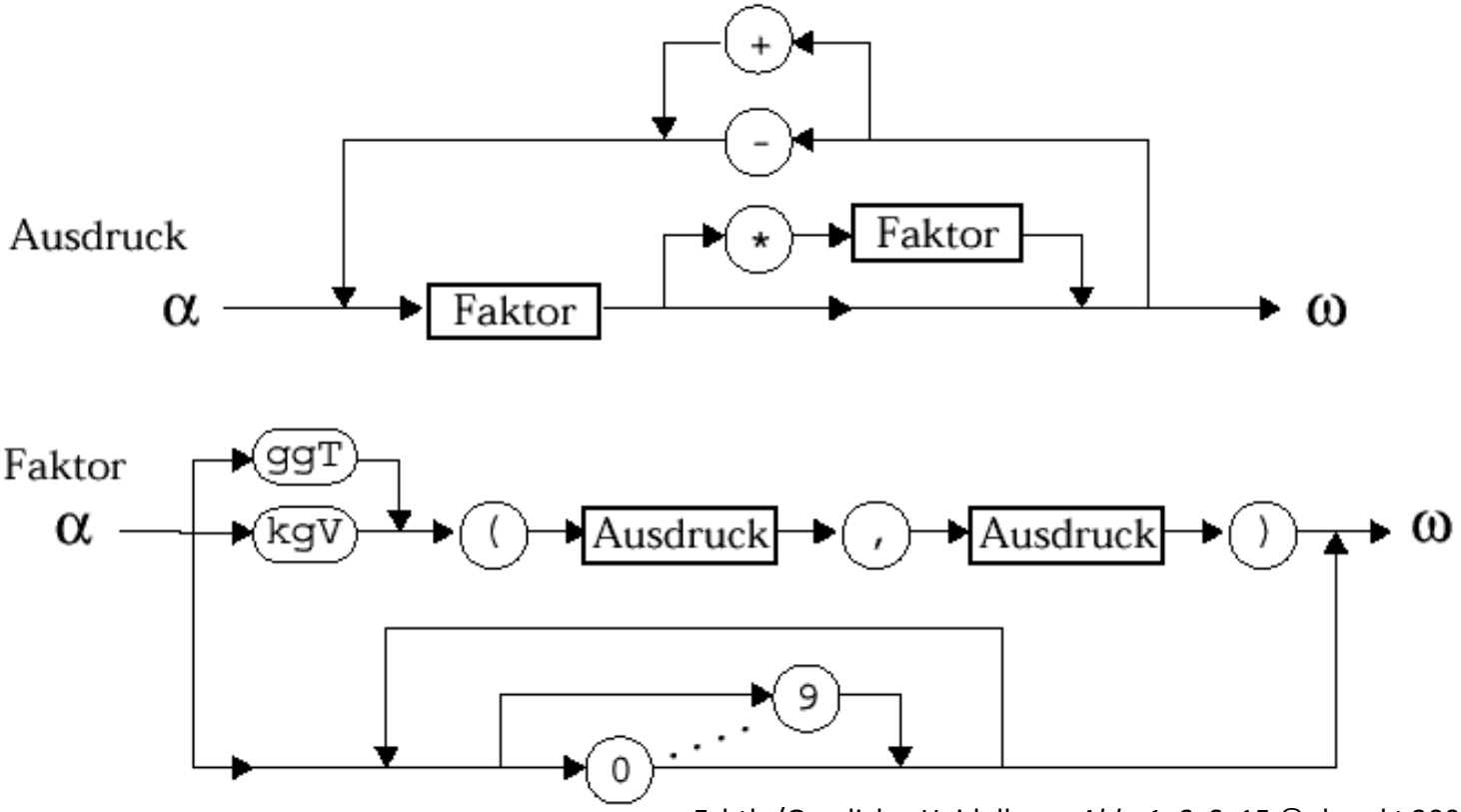
Darstellung von Regel(teil)mengen



Echtle/Goedicke, Heidelberg: *Abb. 1–3* (Ausschnitt, Ergänzungen), S. 15 © dpunkt 2000.

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

Beispielhafte Regelmenge



Echtle/Goedicke, Heidelberg: *Abb. 1–3*, S. 15 © dpunkt 2000.

In diesem Kapitel:
- Prolog
- Spezifikation
- Algorithmus
- Syntax, Semantik

Zwischenstand

Gliederung

- ✓ Stationen im Entwurf von Algorithmen und Programmen
- ✓ Spezifikation
- ✓ Algorithmus
- ✓ Syntax(diagramme)
- ➢ Semantik

In diesem Kapitel:

- Prolog
- Spezifikation
- **Algorithmus**
- Syntax, Semantik

Bemerkung zur Semantik

In der Informatik kann die **Bedeutung** einer formalen **Sprache** (=Programmiersprache)

- operational,
- denotational oder
- verbal

beschrieben werden.

Operationale Semantik

Die operationale Methode definiert schrittweise die Wirkung von Elementaroperationen.

- Schrittweise Beschreibung, wie die Elementaroperationen in den verschiedenen Situationen ausgeführt werden.
- Man unterscheidet also zwischen
  - **Elementaroperationen**
  - **Programmsituationen**

  → Beide zusammen definieren, wie ein Programm schrittweise ausgeführt wird.

- Basis für Softwareentwicklungswerkzeuge (Compiler)

Denotationale Semantik

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 $\rightarrow$ Zustand
- Auf dieser Basis werden formale Korrektheitsbeweise (Macht ein Programm das, was es soll?) geführt.

Verbale 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.
- Für **Java**: „Java Language Specification“
  - eher technisches Dokument (Nachschlagewerk für Hersteller von Softwareentwicklungswerkzeugen)
- Basis für die Einführung der Programmiersprache Java in dieser Vorlesung

Syntax, Semantik

Artikel im EINI-Wiki:

- → **Syntaxdiagramm**
- → **Syntax und Semantik**

Übersicht

Vielen Dank für Ihre Aufmerksamkeit!

Nächste Termine

- Nächste Vorlesung – WiMa 26.11.2020, 08:15
- Nächste Vorlesung – LogWing 27.11.2020, 08:15