Praktische Informatik für Wirtschaftsmathematiker, Ingenieure und Naturwissenschaftler I (PIWIN I, 3 V + 1 Ü) WS 2002/03

2. Vorlesungswoche

Spezifikation

Algorithmus

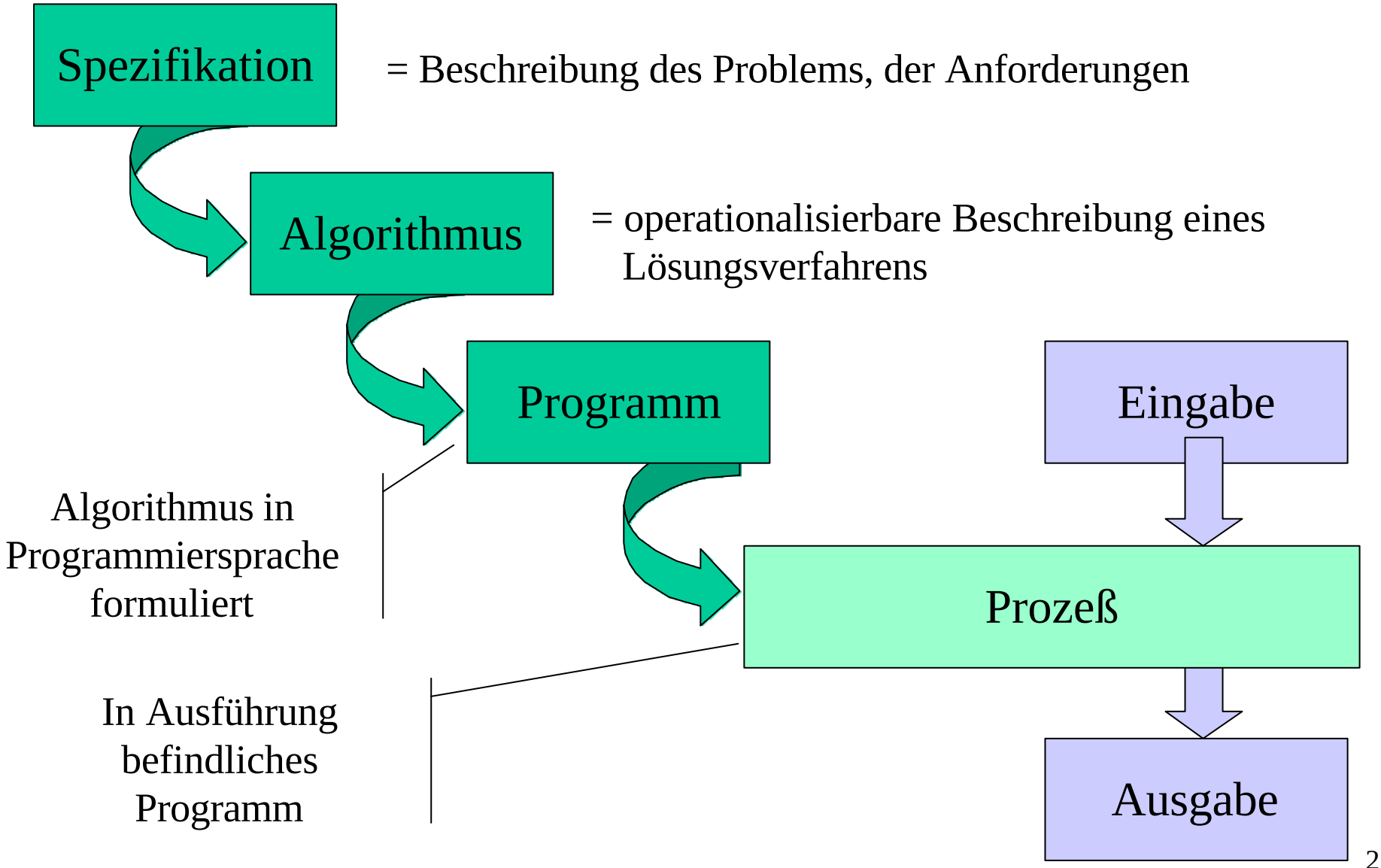
formale Sprache, Grammatik

Unterlagen:

Gumm/Sommer, Einführung in die Informatik, Kapitel 2

Echtle/Goedicke, Einführung in die Programmierung mit Java, dpunkt Verlag

Stationen im Entwurf von Algorithmen und Programmen



Wie geht es weiter ?

- Begriffe
  - Spezifikationen, Algorithmen, formale Sprachen, Grammatik
- Programmiersprachenkonzepte
  - Syntax und Semantik
  - imperative, objektorientierte, funktionale und logische Programmierung
- Grundlagen der Programmierung
  - imperative Programmierung:
    - Verfeinerung, elementare Operationen, Sequenz, Selektion, Iteration, funktionale Algorithmen und Rekursion, Variablen und Wertzuweisungen, Prozeduren, Funktionen und Modularität, Zuweisung, Sequenz
  - objektorientierte Programmierung
- Algorithmen und Datenstrukturen
- Berechenbarkeit und Entscheidbarkeit von Problemen
- Effizienz und Komplexität von Algorithmen
- Programmentwurf, Softwareentwurf

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

Wie sollte eine Problembeschreibung sein ?

- 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)

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

(©A. Schürr, Universität der BW München)

Spezifikation der Aufgabe „ggT-Berechnung“

- 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 < 32767, 0 < n < 32757 }
- 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.

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.“
- 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.“
- Definition nach Echtle/Goedicke über Eigenschaften folgt ...
- Typische Beispiele für Algorithmen aus dem Alltag:
  - Kochrezepte
  - Gebrauchsanweisungen
  - Strickanleitungen
  - ...

Definition eines Algorithmus nach Echtle/Goedicke

A1 Ein Algorithmus beschreibt eine Relation über das 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.

(©M. Goedicke, UGH Essen)

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

(©M. Goedicke, UGH Essen)

Deterministisch vs. nicht deterministisch

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

Bei der Festlegung der Reihenfolge der Elementaroperationen gibt es noch Freiheiten

Siehe oben (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 ...

(©M. Goedicke, UGH Essen)

Beispiele

Z.B.: die Addition (2+2) oder
die Einkommensteuerberechnung
sollten determiniert sein
Achtung: die konkrete Abfolge der Schritte ist damit nicht festgelegt!

Z.B.: Die Reservierung von Flugsitzen von verschiedenen Buchungsterminals aus ist in der Regel nicht determiniert

(©M. Goedicke, UGH Essen)

Beispiele (Forts.)

```
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 involvieren sind in der Regel indeterministisch und müssen mit großem Aufwand zu determinierten Verfahren gemacht werden.

(©M. Goedicke, UGH Essen)

Weitere Forderungen an Algorithmen …

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 sonstige unvorhergesehene Situation auftritt.

(©M. Goedicke, UGH Essen)

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
```

Prog. 1–3 *Berechnung des artihmetischen 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
```

Prog. 1–4 *Berechnung des artihmetischen Mittels nach der Formel 0,5 · x + 0,5 · y*

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

(©M. Goedicke, UGH Essen)

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 Rechnerarithmetikprobleme aufwerfen.

Insgesamt ist die Verstehbarkeit von Software (heute) zentral! Warum?

(©M. Goedicke, UGH Essen)

Beispiele für Eigenschaften von Algorithmen

- 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 beiden Operationen ist zwingend 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 beiden 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 der Programmiersprache Prolog sind, die die sogenannte logische Programmierung unterstützt.

Nichtdeterministische Formulierung des ggT

- Die folgende Charakterisierung läßt sich als nichtdeterministischer Algorithmus mit dennoch determiniertem Ergebnis interpretieren:
  1. z = ggT(z,0)
  2. z = ggT(0,z)
  3. z = ggT(z,z)
  4. z = ggT(m,n) falls gilt m ≥ n und z = ggT(m-n,n)
  5. z = ggT(m,n) falls gilt n ≥ m und z = ggT(m,n-m)
- Grund
  - für ggT(0,0) greift jede der 5 Gleichungen
  - in jedem Fall wird dasselbe Ergebnis berechnet
  - kritisch: Fall 4 und 5 bringen keinen Fortschritt -> Terminierung?

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.

Anmerkung:

Die Unterscheidung „algorithmische Sprache“ und „Programmiersprache“ ist damit auch eine Frage der auf einem Rechner zur Verfügung stehenden Ausführungswerkzeuge.

(©A. Schürr, Universität der BW München)

Die bekanntesten Programmierparadigmen

- imperative (prozedurale) Programmierung (Pascal,C,Fortran, Cobol, PL/1, VisualBasic, ...)
  - Anweisungen verändern Werte von Variablen
  - Kontrollstrukturen regeln Reihenfolge der Ausführung von Anweisungen
  - Prozeduren definieren wiederverwendbare Kontrollstrukturen
- funktionale (applikative) Programmierung (Lisp, Haskell, ML, Scheme, ...)
  - Ein Programm besteht aus Funktionsdefinitionen.
  - Jede Funktion wird durch einen Ausdruck definiert.
  - Die Programmausführung besteht aus der Anwendung (Applikation) von Funktionen auf Ausdrücke (Terme).
- logische Programmierung (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.

Die bekanntesten Programmierparadigmen, fortgesetzt:

- objektorientierte Programmierung (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.
- regelorientierte Programmierung (OPS-5, ...) ist mit der logischen Programmierung verwandt:
  - Es gibt einen großen Datenraum (z.B. als Schachbrett organisiert, auf dem verschiedene Objekte liegen)
  - Regeln suchen bestimmte „Muster“ in diesem Datenraum und ersetzen sie durch andere Muster.
  - Die Programmausführung besteht also auch aus der Anwendung „feuerbereiter“(anwendbarer) Regeln.

Die Formulierung von Algorithmen wird in künstlichen Sprachen abgefasst

Wegen der Eindeutigkeit der Interpretation:

sogenannte Formale Sprachen

Für alle Sprachen gilt (in etwa)

Syntax: bezeichnet die Regeln des formalen Aufbaus einer Sprache

Semantik: bezeichnet die Regeln für die Interpretation eines Wortes/Satzes/Textes, um seine Bedeutung zu definieren

Die Syntax wird durch eine Grammatik festgelegt.

Beispiele für Worte (Ausdrücke) einer Sprache:

(0.0 +(9*-8{ Dem Augen

Die Renten sind sicher

(©M. Goedicke, UGH Essen)

Eine Grammatik besteht aus verschiedenen Elementen

- Alphabet ... ist eine Menge von Symbolen oder Zeichen aus denen die Worte/Sätze einer Sprache geformt werden können
  - Beispiele A= { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  - Oder A= { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, /, ggT, kgV, (,), „“ }
- Ein Wort ist eine Kette von Zeichen über dem gegebenen Alphabet
- Die Menge der Worte, die überhaupt über einem Alphabet gebildet werden können, heißt freier Monoid über A und wird mit A* bezeichnet.

Die Menge der Wörter über einem Alphabet (der freie Monoid) ist immer unendlich groß

- Sei A= { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, /, ggT, kgV, (,), „“ }
  - 11+1+1+0+11111 ∈ A*
  - aber auch 020-(kgV ∈ A*

→ es werden also noch Regeln benötigt „richtige“ Worte zu bilden

(©M. Goedicke, UGH Essen)

Das nächste Element einer Grammatik ist also die Menge der Produktionsregeln

- Produktionsregeln oder Grammatikregeln erlauben auszudrücken, dass
  - in deutschen Sätzen „erst Subjekt dann Prädikat“ kommt
  - die Begriffe deutsche Sätze, Subjekt und Prädikat werden als syntaktische Variablen bezeichnet
  - „erst Subjekt dann Prädikat“ ist eine von vielen grammatikalischen Regeln

  → eine weitere Menge – die syntaktischen Variablen V – und die Regeln P werden benötigt.

Insgesamt besteht eine Grammatik aus 4 Elementen

- V die Menge der syntaktischen Variablen (auch als Nichtterminalsymbole bezeichnet)
- T das Alphabet (bisher mit A bezeichnet) (auch mit „Menge der Terminalsymbole“ bezeichnet)
- P die Menge der Produktionen (oder Grammatik-Regeln)
- S Startsymbol, $S \in V$

$\rightarrow$ Eine formale Sprache $L \subset T^*$
die sich mit Hilfe der Regeln aus der Menge P herleiten lassen

Der Aufbau der Regeln muss noch detailliert werden

- Die Regeln einer Grammatik sind sogenannte Ersetzungsregeln der Form:

$$X \rightarrow Y$$

- das bedeutet, dass ein Vorkommen von X durch Y ersetzt werden darf.
  X und Y sind Teile oder Teilworte
- bei den sogenannten kontextfreien Grammatiken, die hier betrachtet werden sollen gilt

$X \in V$
$Y \in (T \cup V)^*$ (wobei gelten muss: $V \cap T = \emptyset$)

Am Besten ein erstes Beispiel ...

V ={ Ausdruck, Summand, Faktor, Funktion, Zahl }

T ={ „0“, „1“, „2“, „3“, „4“, „5“, „6“, „7“, „8“, „9“, „+“, „-“, „*“,
„ggT“, „kgV“, „(“ , „)“, „ , “ }

P ={Ausdruck → Summand,

Ausdruck → Summand „+“ Ausdruck,

Ausdruck → Summand „-“ Ausdruck,

Summand → Faktor,

Summand → Faktor „*“ Faktor,

Faktor → Zahl,

Faktor → „ggT“ „(“ Ausdruck „ , “ Ausdruck „)“ ,

Faktor → „kgV“ „(“ Ausdruck „ , “ Ausdruck „)“ ,

Zahl → „0“,... , Zahl → „9“, Zahl → Zahl Zahl }

S = Ausdruck

(©M. Goedicke, UGH Essen)

Eine Grammatik ist ein sogenanntes Erzeugendensystem

- $G = (V,T,P,S)$
- $L(G) = \{w \in T^* : \exists\ w_1, \ldots, w_k \in (V \cup T)^*:$
  $S \rightarrow_P w_1,\ w_1 \rightarrow_P w_2, \ldots, w_k \rightarrow_P w\}$
- mit $w_i \rightarrow_P w_j$ aus dem Wort w_i wird w_j mit Hilfe einer Regel aus P
- d.h. sukzessive wird aus dem Startsymbol ein Wort abgeleitet, das nur noch Terminalsymbole enthält.

(©M. Goedicke, UGH Essen)

Hier ein Beispiel wie das funktioniert

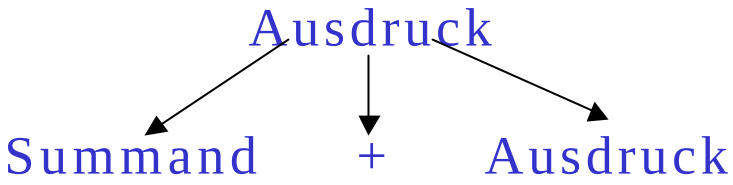
Ausdruck	Regel 2
Summand + Ausdruck	Regel 4
Faktor + Ausdruck	Regel 1
Faktor + Summand	Regel 4
Faktor + Faktor	Regel 6
Zahl + Faktor	Regel 8
Zahl + kgV(Ausdruck,Ausdruck)	Regel 10
Zahl Zahl + kgV(Ausdruck,Ausdruck)	Regel 9‘
2 Zahl + kgV(Ausdruck,Ausdruck)	Regel 9‘
2 2 + kgV(Ausdruck,Ausdruck)	...
...	

(©M. Goedicke, UGH Essen)

Der Ableitungsprozess kann auch grafisch gut dargestellt werden

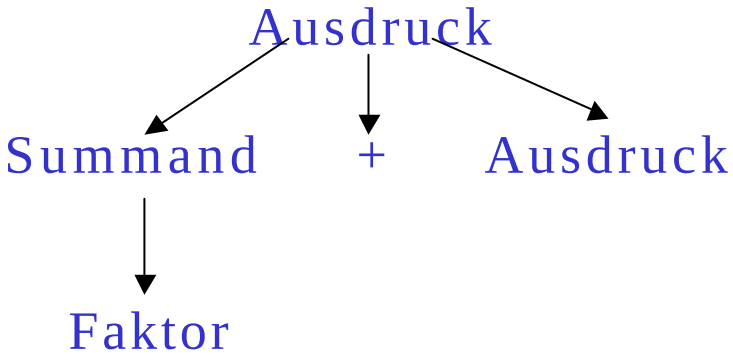
Ausdruck

Ausdruck -> Summand + Ausdruck



Summand + Ausdruck

Summand -> Faktor

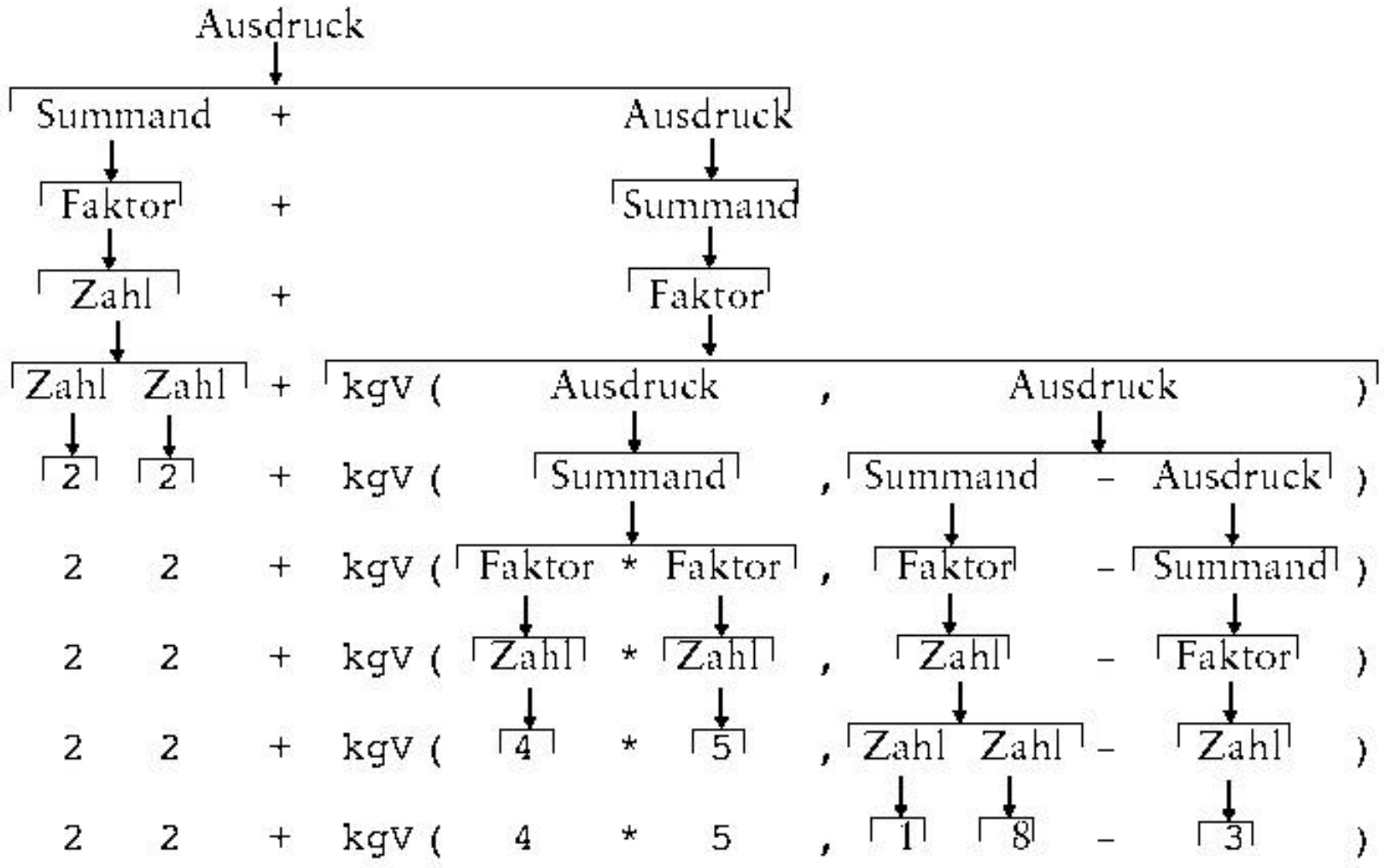


Faktor + Ausdruck

heißt Ableitungsbaum

(©M. Goedicke, UGH Essen)

Ein komplexeres Beispiel ...
der Ausdruck 22+kgV(4*5,18-3)



(©M. Goedicke, UGH Essen)

Für praktische Zwecke ist die erweiterte Backus-Naur Form (EBNF) sinnvoll

Produktionen einer Grammatik $a, b \in V,\ x, y, z, x_1, \ldots, x_k \in (V \cup T)^*$	EBNF-Notation	Bemerkung
$a \rightarrow x$	$a = x$	
$a \rightarrow x_1, \ \ldots, \ a \rightarrow x_k$	$a = (x_1 \mid \ldots \mid x_k)$	Alternativen
$a \rightarrow xz, \quad a \rightarrow xyz$	$a = x \ [\ y \ ] \ z$	y ist optional
$a \rightarrow xa, \quad a \rightarrow y$	$a = \{ \ x \ \} \ y$	x beliebig oft wiederholt
$a \rightarrow xb, \quad b \rightarrow xb, \quad b \rightarrow y$	$a = \{ \ x \ \}_1 \ y$	x mindestens 1 mal

Tab. 1–1 *Erweiterte Backus-Naur-Form EBNF*

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

Nebenbemerkung: das funktioniert nur für kontextfreie Grammatiken

(©M. Goedicke, UGH Essen)

Damit wird die Grammatik für die arithmetischen Ausdrücke sehr kompakt

- Aus

  Ausdruck ® Summand,

  Ausdruck ® Summand ”+” Ausdruck,

  Ausdruck ® Summand ”-” Ausdruck

- Wird

  Ausdruck = Summand [(”+” ç ”-”) Ausdruck]

- Beachte: die Zeichen in **rot** sind sogenannte

  Meta-Zeichen/Meta-Symbole **(=,(,) ,|, [,] ,{,})**

(©M. Goedicke, UGH Essen)

Insgesamt wird die Grammatik für arithmetische Ausdrücke zu einem Dreizeiler

- Also

Ausdruck = Summand [(”+” ç ”-”) Ausdruck]

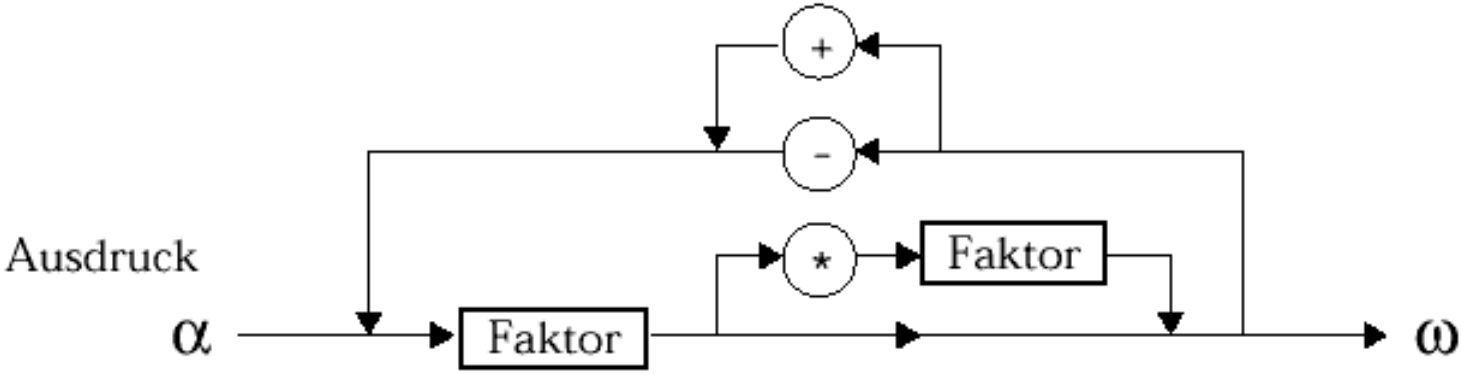
Summand = Faktor [”*” Faktor]

Faktor = ({ (”0” ç ”1” ç ... ç ”9”) }$_1$ ç
(”ggT” ç ”kgV”)
”(” Ausdruck ”,” Ausdruck ”)”)

(©M. Goedicke, UGH Essen)

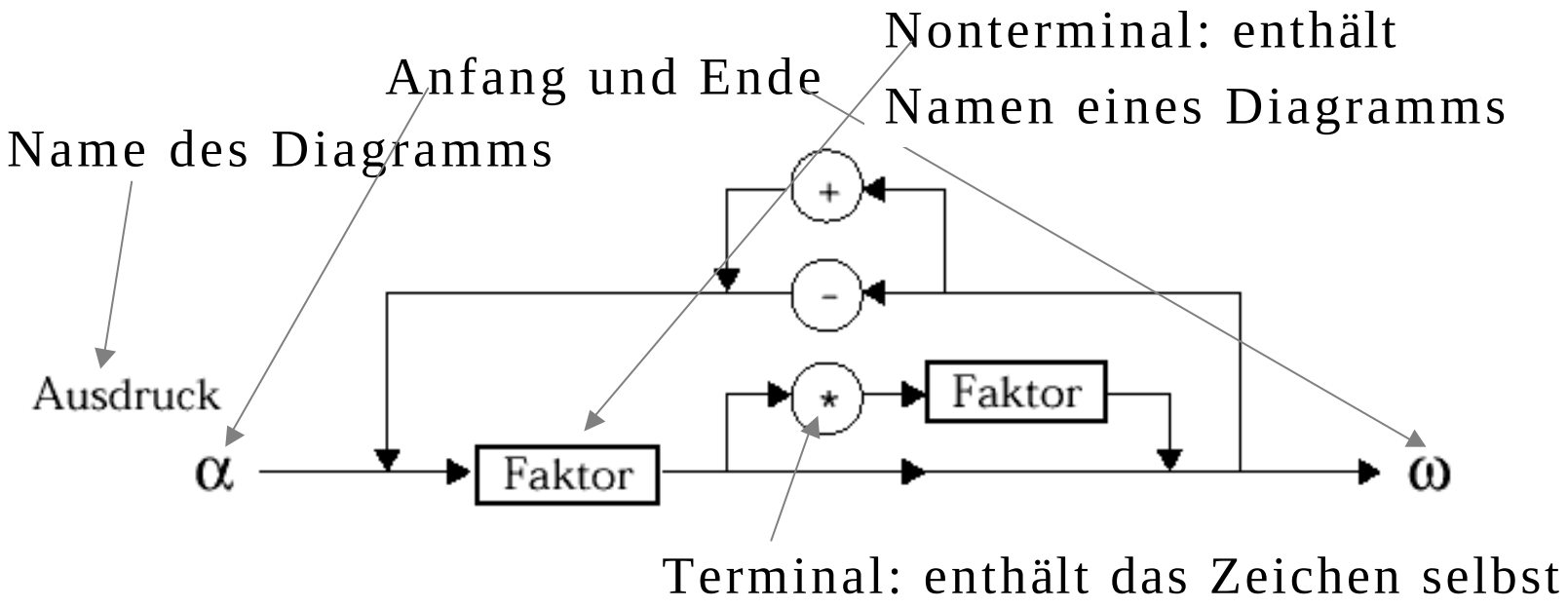
Es gibt eine weitere weitverbreitete grafische Alternative zur EBNF

- Syntax-Diagramme
  bestehen aus
- zwei unterschiedlichen Arten von Kästen (rund = Terminal und eckig = Nonterminal)
- und Pfeilen, die diese Kästen miteinander verbinden



(©M. Goedicke, UGH Essen)

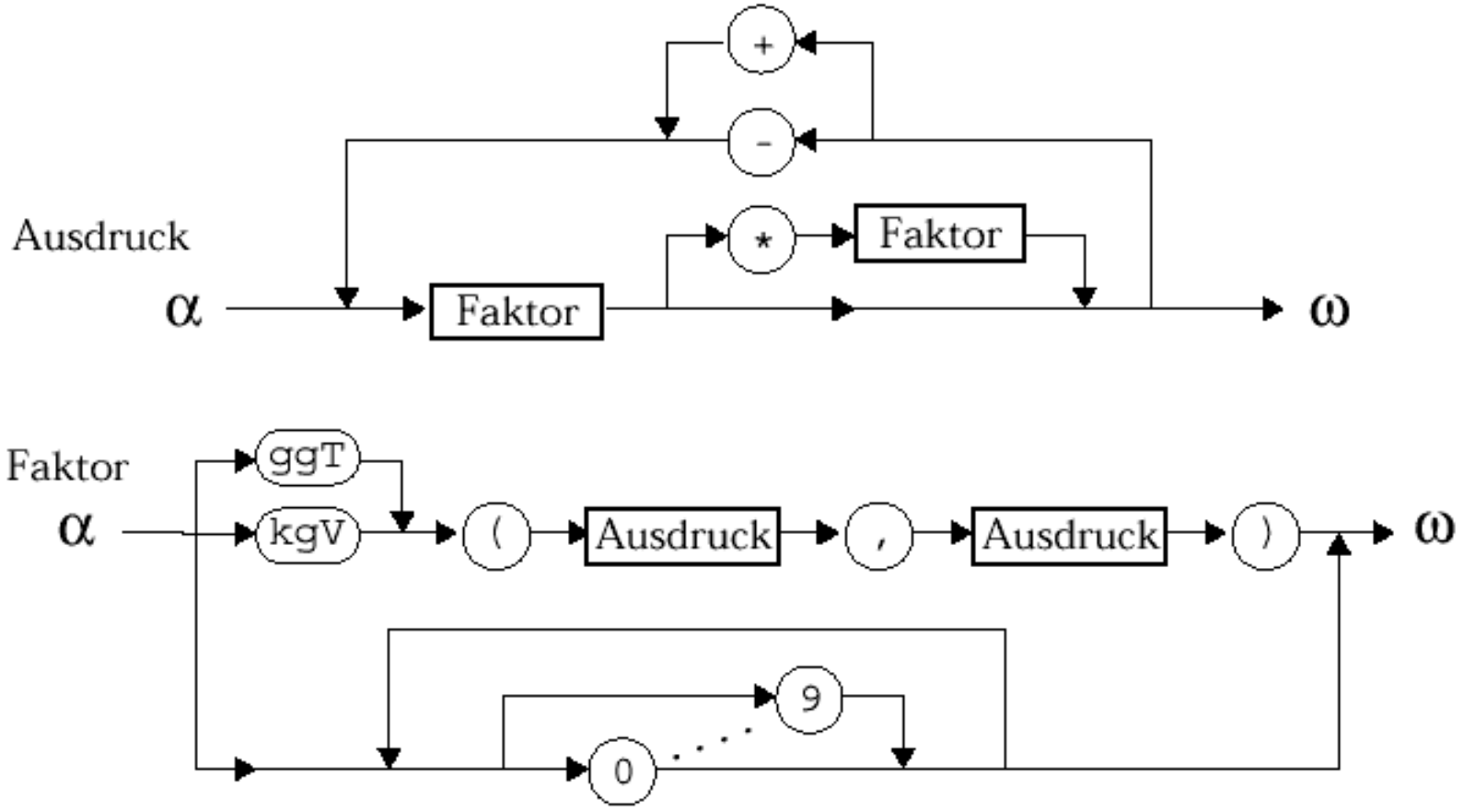
Eine kontextfreie Grammatik besteht damit aus einer Menge solcher Diagramme



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

(©M. Goedicke, UGH Essen)

Das gesamte Syntaxdiagramm lautet dann ...



(©M. Goedicke, UGH Essen)

Grammatiken werden für verschiedenste Zwecke eingesetzt

- Nicht nur für die Struktur und den Aufbau von Programmiersprachen
- Eingabesprachen von Programmen
  - Kommando-Sprachen
  - Datendefinitionssprachen
  - Kassenterminals
  - ...
- Kommunikationsprotokolle
- ...

(©M. Goedicke, UGH Essen)

Eine kurze Bemerkung zur Semantik ist angebracht

- Die Informatik benutzt überlicherweise drei Möglichkeiten die Bedeutung einer formalen Sprache (hier Programiersprache) zu beschreiben

  a) operational

  b) denotational

  c) verbal

(©M. Goedicke, UGH Essen)

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 eine Programm schrittweise ausgeführt wird
- Auf dieser Basis werden Softwareentwicklungswerkzeuge (Compiler) hergestellt

(©M. Goedicke, UGH Essen)

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 (tut ein Programm das was es soll?) geführt

(©M. Goedicke, UGH Essen)

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 Spracheelement der betrachteten Programmiersprache geliefert.

  Java: Die sogenannte Java Language Reference

  ist eher ein technisches Dokument ... gedacht als Nachschlagewerk für Hersteller von Softwareentwicklungswerkzeugen
- Auf dieser Basis wird Programmiersprache Java hier in der Vorlesung eingeführt

(©M. Goedicke, UGH Essen)

Nach all diesen Vorreden sollen nun wesentliche Sprachkonstrukte imperativer Programme betrachtet werden

Rückblick:

- Spezifikationsbegriff: Was sind die Eigenschaften einer Problem-/ Aufgabenbeschreibung
- Algorithmusbegriff: Was sind die Eigenschaften automatischer Verfahrensvorschriften
- Grammatiken: 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