

Einführung in die Informatik I
Algorithmen und
Textersetzungssysteme

Prof. Bernd Brügge, Ph.D
Technische Universität München

Wintersemester 2000/2001
7.-13. November 2000

Überblick über den Vorlesungsblock

- ❖ Zentraler Begriff der Informatik: Algorithmus
- ❖ Unterschiedliche Definitionen
- ❖ Taxonomie von Algorithmen
- ❖ Textersetzungs-systeme
 - Semi-Thue Systeme
 - Markov-Algorithmen
- ❖ Definitionen: Formales System, Entscheidbarkeit, Berechenbarkeit, Chomsky-Grammatik
- ❖ Backus-Naur-Form, Syntaxdiagramme

Funktionsbeschreibungen

- ❖ Die einfachste Aufgabe eines Informatik-Systems ist die Realisierung einer Funktion $f:A\rightarrow B$. Wir können solche Funktionen auf verschiedene Weisen beschreiben
- ❖ **Statisch und deklarativ**
 - In Form von Gleichungen zur Darstellung der Zusammenhänge von A und B , z.b. $g(a, b) = 0$. Diese Beschreibung hilft nicht als konstruktive Regel, um $f(a)$ zu berechnen, reicht aber oft für die erste Modellierung.
- ❖ **Tabellarisch:**
 - Für alle Werte a des Definitionsbereichs A wird der zugeordnete Wert $f(a)$ in einer Wertetabelle für den Wertebereich B festgehalten.
Werte werden “berechnet”, indem man sie nachschlägt.
- ❖ **Prozedural:**
 - Durch eine **algorithmische** Beschreibung.

Algorithmische Beschreibung

- ❖ Eine algorithmische Beschreibung dient zur Berechnung des Ergebnisses für beliebige Argumente des Definitionsbereichs.
- ❖ Die Beschreibung kann in einer beliebigen Sprache sein (natürliche Sprache, Graph, Pseudosprache, Programmiersprache)
 - Die Beschreibung muss endlich sein und muss die Berechnung als Folge von endlich vielen elementaren Operationen mit hinreichender Präzision spezifizieren.
- ❖ Eine solche Beschreibung heißt **Algorithmus**.
- ❖ Beispiele von Algorithmen:
 - Euklid's Verfahren für die Konstruktion von Dreiecken
 - Rezept zum Herstellen von Semmeln
 - Anweisung für die Einnahme einer Kopfschmerztablette

Beschreibung eines Algorithmus

- ❖ An alle Teilnehmer von Info I:
- ❖ **Wenn Sie keinen linken Nachbarn haben:**
 - Wenn ich die linke Hand hebe, und die Zahl 1 anzeige: Stehen Sie auf; heben Sie Ihre Hände; setzen Sie sich wieder hin.
 - Jedesmal, wenn jemand, der rechts keinen Nachbarn hat, aufsteht, dann stehen Sie auf; setzen Sie sich gleich wieder hin.
- ❖ **Wenn Sie einen linken Nachbarn haben:**
 - Wenn Ihr linker Nachbar aufsteht, stehen Sie auf.
- ❖ **Wenn ich beide Hände hebe**
 - Stehen Sie auf oder bleiben Sie stehen und rufen Sie das Mantra "Es lebe Info I".

Algorithmus: Definition

❖ **Definition Algorithmus:**

– Ein Algorithmus ist ein Verfahren zur Verarbeitung von Daten mit einer **präzisen, endlichen** Beschreibung unter Verwendung **effektiver** Arbeitsschritte

❖ Präzise: In einer eindeutigen Sprache abgefasst

❖ Endlich: In einer endlichen Form beschrieben

❖ Effektiv: Die Arbeitsschritte sind tatsächlich ausführbar

❖ Ein Algorithmus muss nicht terminieren

❖ **Definition Terminierender Algorithmus:** Das Verfahren hat *endlich* viele Schritte

◆ Broy (S.31) verwendet diese Definitionen.

Was ist Präzise, Endlich, Effektiv?

❖ *Präzise: in einer eindeutigen Sprache abgefasst*

- maschinell lesbar und ausführbar
- Eine Beschreibung ist unpräzise, wenn der Schritt nicht ausgeführt werden kann, weil eine Bedingung nicht eindeutig ausgewertet werden kann.

Beispiel: **Wenn der Mond im Innern grün ist, dann addiere 5.**

❖ *Endlich: In einer endlichen Form beschrieben*

- Ich brauche nur endlich viel Papier (oder Elektronen:-).

❖ *Effektiv: Die Arbeitsschritte sind tatsächlich ausführbar*

- Beispiel eines nicht effektiven Arbeitsschrittes:
Nimm Deinen Zauberstab und schau in die Kristallkugel, um den nächsten Arbeitsschritt zu finden.

Algorithmus: Zweite Definition

Definition (Knuth): Ein Algorithmus ist

- 1) Eine Beschreibung eines Verfahrens und seiner Daten mit Hilfe einer Sprache
- 2) Eine *präzise* Beschreibung der einzelnen Schritte des Verfahrens
- 3) Eine *endliche* Aufschreibung für die Darstellung eines endlichen, aber im Allgemeinen unbeschränkten Verfahrens.
- 4) Jeder Einzelschritt des Verfahrens ist in *endlicher* Zeit ausführbar
- 5) Das Verfahren hat nur *endlich* viele Schritte.

Aus 4) und 5) folgt, daß das Verfahren terminieren muss, sonst ist es kein Algorithmus

- ◆ Goos (S.29) verwendet diese alternative Definition.

Algorithmus Definition: Diskussion

- ❖ Einer der fundamentalen Begriffe der Informatik ist nicht allgemein akzeptiert. Warum?
 - ◆ Die zweite Definition (“Ein Algorithmus muss terminieren”) bezieht sich auf die Berechnung einer einzelnen Funktion, und da ist es schon besser, wenn wir für eine Eingabe auch eine Ausgabe bekommen.
 - ◆ Die erste Definition (“Ein Algorithmus muss nicht terminieren”) bezieht auch andere interessante Informatik-Systeme ein, von denen wir erwarten, dass sie nicht terminieren (z.B. ein Betriebssystem oder ein Web-Server)
- ❖ Was sagt das über den Reifegrad der Informatik?
 - Informatik ist immer noch ein junges Gebiet.
 - Unterschiedliche Definitionen in vielen Bereichen der Informatik sind leider gang und gäbe.
- ❖ In Info I benutzen wir die erste Definition, d.h. nicht jeder Algorithmus terminiert.

Geschichte des Gleichheitszeichens

- ❖ **Bezeichnungen für Gleichheit (bis ins 17. Jahrhundert):**
 - Worte: "aequales, esgale, gleich, aeq"
 - Zeichen: ∞ [| 2|2 (Herigone in 1634)
 - Beispiele: $4 \text{ aeq } 4$ $4 \infty 4$ $4[4$ $4|4$ $4 \text{ 2|2 } 4$
- ❖ **Robert Recorde (1577): $7+6=\text{foo}$**
 - Das Gleichheitszeichen ist eine Abkürzung (so haben wir es bisher benutzt)
- ❖ **Gottfried Wilhelm Leibniz (1694): $3+4 = 7$**
 - Das Gleichheitszeichen bezeichnet eine Funktion
- ❖ **George Boole (1850):**
 - Das Gleichheitszeichen bezeichnet eine Boolesche Funktion
- ❖ Heute ist das Zeichen "=" in der Mathematik universell akzeptiert, leider nicht in der Informatik:-((Scott Fahlmann, CMU, 1985)
 - Fortran: "=" bezeichnet auch die Zuweisung
 - C, Java: Gleichheit wird durch "==" ausgedrückt

War die Spezifikation der Welle präzise ?

- ❖ Definition eines Nachbars
 - Wer ist Ihr Nachbar, wenn der Platz neben Ihnen frei ist?
- ❖ Was genau ist eine Welle?
 - Wie schnell muß man aufstehen und sich wiederhinsetzen, damit die Welle funktioniert?
 - Kann man eine Welle mit 10 Leuten durchführen? Mit 5?
 - Wieviele Leute braucht man mindestens?
- ❖ Was ist falsch mit dem folgenden Arbeitsschritt?
 - **Wenn Sie einen linken Nachbarn haben:**
 - ◆ Wenn Ihr linker Nachbar aufsteht, stehen Sie auf.
- ❖ Kann man auch 2 Wellen gleichzeitig machen?
 - Wie müssten dann die Operationen “Hinsetzen” und “Aufstehen” beschrieben sein?

Zeitliche Ausführungen von Operationen in Algorithmen

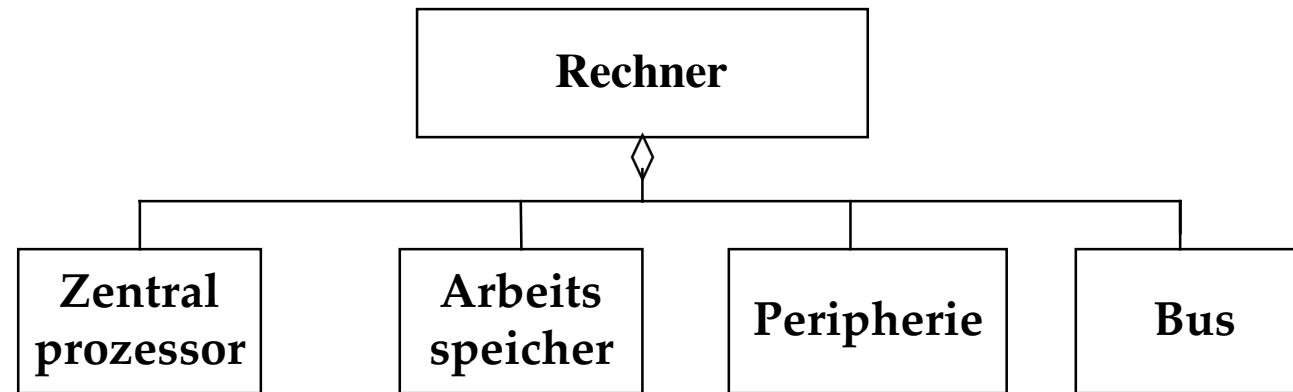
- ❖ *Gegeben: Operationen a und b , und Bedingung B .*
- ❖ **Sequentielle Ausführung:**
 - a muss vor b ausgeführt werden.
- ❖ **Parallele Ausführung:**
 - a kann gleichzeitig mit b ausgeführt werden.
- ❖ **Bedingte Ausführung:**
 - a oder b kann nur ausgeführt werden, wenn Bedingung B wahr ist.
- ❖ **Ausführung in einer Schleife:**
 - a wird für eine bestimmte Zeitspanne, oder bis eine vorgegebene Bedingung B eintritt, ausgeführt.
- ❖ **Ausführung eines Unterprogramms:**
 - Führe Operation b aus, die anderswo beschrieben ist, und verwende das Ergebnis in a weiter.
- ❖ Weitere Angaben zur zeitlichen Ausführung von Operationen gibt es nicht!

Wie kann man Algorithmen realisieren?

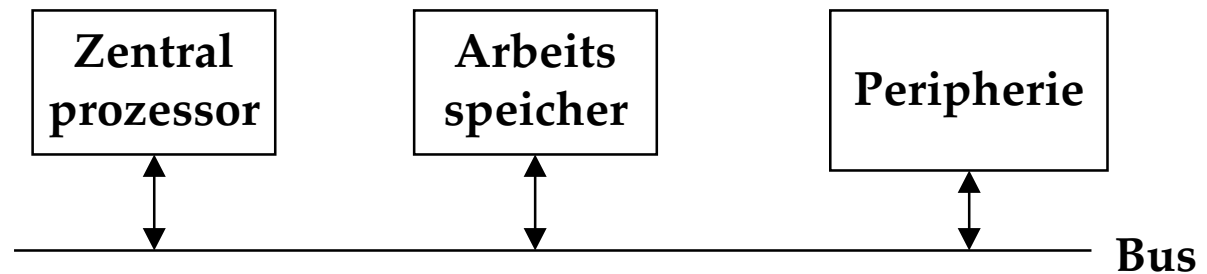
- ❖ Zur Realisierung (**Implementation**) von Algorithmen verwendet man Rechner.
- ❖ Ein **Rechner** besteht aus 4 Komponenten (John von Neumann, 1946):
 - **Arbeitsspeicher:** Enthält den Algorithmus und die Daten
 - **Zentralprozessor:** Führt Operationen (Befehle) des Algorithmus aus
 - **Peripherie:** Ein- und Ausgabegeräte, die die Verbindung zwischen dem Menschen und dem Zentralprozessor sowie Arbeitsspeicher herstellen.
 - **Bus:** Verbindungsweg, der die anderen Komponenten miteinander verknüpft.

Modell Repräsentationen eines von-Neumann Rechners

Als Aggregation
von Komponenten
(Klassendiagramm)



Informelle Darstellung
(Info II \Rightarrow Sequenzdiagramm)



Geschichte der Rechner

❖ **Addiermaschinen:**

- 1623: Schickard, 1641: Pascal, **1673: Leibniz**

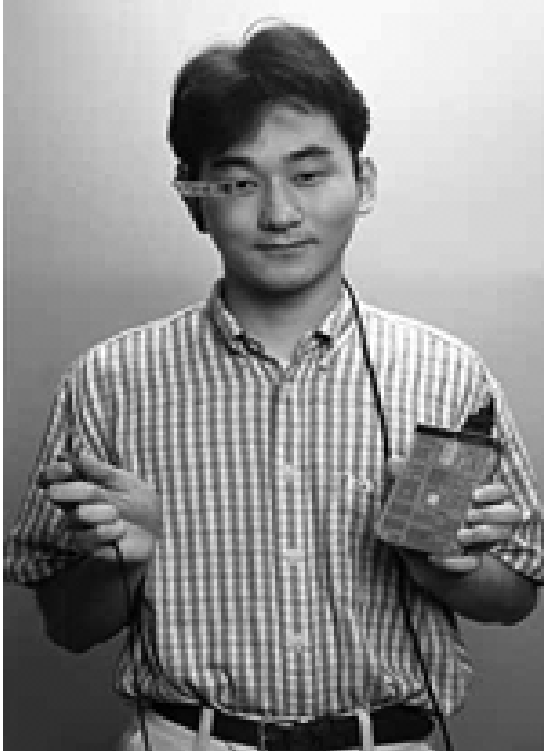
❖ **Programmgesteuerte Rechner**

- **Babbage (1833)**: Erste Idee eines programmgesteuerten Rechners mit mechanischen Hilfsmitteln: Difference Engine, Analytical Engine
- **Hollerith (1890)**: Lochkarte, Tabulator für Volkszählungen
- **Zuse (1937)**: Erster programmgesteuerter Rechner Z3 mit elektromechanischer Realisierung der Komponenten.
- **Aiken (1939)**: Erster elektronischer Rechner Mark I
- **Mauchly und Eckert (1959)**: ENIAC: 18000 Röhren, 167m²

❖ **Und heute?**

Hardware

"*Hardware*": *Tragbare Rechner*



- ❖ Prototyp von IBM
- ❖ Dieselbe Rechenleistung wie ein ThinkPad 560X
- ❖ Gewicht:
 - System Unit mit Batterie: 299g
 - Kopfdisplay: 50 g
 - “Maus”: 20 g
- ❖ Peripherie: 1 GByte MicroDrive: 5mm Dicke, 20g
- ❖ Li-Ion Batterie reicht für 1.5-2 Stunden



Was kommt als nächstes?

❖ **Aiken (ca 1945):**

- Der Bedarf an Rechnern für die gesamten USA: 6.

❖ **Thomas J. Watson (IBM, ca 1950):**

- Der Bedarf an Rechnern für die gesamte Welt: 45.

❖ **Ken Olson (Digital Equipment Corporation, ca. 1980):**

- Es besteht kein großer Bedarf an persönlichen Rechnern

❖ **Bill Gates (Microsoft, ca. 1985):**

- Es besteht kein Bedarf an Programmen, die nicht in einen 16 Bit Addressraum passen.

❖ „Anzahl der Rechner“-Zeitlinie (aus „Rechenmaschinen“, Deutsches Museum, F.L. Bauer, Herausgeber sd&m)

❖ Allgegenwärtiges Rechnen (Ubiquitous Computing):

- Java Ring (Sun, 1996): Rechner am Finger
- Smart Labels (Gemplus, 1999): Rechner im Hemdkragen

Der Algorithmus als Grundlage aller Informatik-Systeme

- ❖ Was haben wir soweit gelernt?
 - Es gibt mehrere Definitionen für Algorithmus
 - ◆ Hält (terminiert) er oder hält er nicht?
 - Geschichte der Rechner ist rasant
- ❖ Was kommt jetzt?
 - **Zeichenketten**
 - **Textersetzungs-systeme** als einfache Algorithmen
 - ◆ Beispiele von Textersetzungs-systemen
 - ◆ Markov-Algorithmen als deterministische Algorithmen
 - **Formale Sprachen** und **Formale Systeme**
 - **Grammatiken** zur Spezifikation von Textersetzungs-systemen
- ❖ Was kommt in den nächsten 2 Wochen?
 - **Termersetzungs-systeme, Boolesche Algebra**

Zeichensequenzen

- ❖ Die Darstellung von Information durch gesprochene Sprache (durch eine Folge von Lauten) und durch Schrift (durch eine Folge von Zeichen) ist eine fundamentale Errungenschaft der Menschheit.
- ❖ Auch in der Informatik verwenden wir sehr oft Zeichensequenzen bzw. Zeichenfolgen.
 - Ein Algorithmus kann beispielsweise als eine endliche Zeichenfolge angesehen werden.

Definitionen

- ❖ **Definition Wort:** Für eine gegebene Menge V von Zeichen bildet eine Folge $x = x_1, \dots, x_n$ mit $x_i \in V$ ein Wort der Länge n . Wir schreiben auch $x = x_1..x_n$ für das Wort und sprechen von einem n -Tupel.
- ❖ V^n bezeichnet die Menge aller n -Tupel
 - $V^n =_{\text{def}} V \times V \dots \times V$ (n -faches Kreuzprodukt über V)
- ❖ ε bezeichnet die leere Sequenz („leeres Wort“)
- ❖ V^+ bezeichnet die Menge aller Worte über V ohne das leere Wort.
- ❖ V^* bezeichnet die Menge aller endlichen Zeichensequenzen von Zeichen aus V .

Zeichensequenz Operationen

- ❖ Über V^* definieren wir als Operation die zweistellige Abbildung **Konkatenation:**

$$\text{conc}: V^* \times V^* \rightarrow V^*$$

- ❖ Die Sequenz $\text{conc}(s, t)$ mit $s, t \in V^*$ ist diejenige Sequenz, die sich durch Aneinanderreihen („Konkatenieren“) der Sequenzen s und t gebildet wird.
- ❖ Konkatenation wird oft in Infixweise geschrieben:

$$- v \circ w =_{\text{def}} \text{conc}(v, w)$$

- ❖ Für die Konkatenation gelten folgende Gesetze:

$$\begin{aligned} - (u \circ v) \circ w &= u \circ (v \circ w) && \text{(Assoziativität)} \\ w \circ \varepsilon &= w && \text{(Neutrales Element)} \end{aligned}$$

- ❖ V^* ist also ein Monoid (eine Halbgruppe mit neutralem Element) bezüglich der Konkatenation.

Textersetzungssystem

- ❖ Ein Textersetzungssystem ist eine sehr einfache Form eines Algorithmus. Mit Hilfe von Regeln werden Worte in andere Worte übergeführt.
- ❖ Sei V ein endlicher Zeichenvorrat.
- ❖ Gegeben sei eine Folge $x = x_1 \dots x_n$ von Zeichen x_i aus V .
- ❖ **Definition Wort, Länge:**
 - x ist ein **Wort** über dem Zeichenvorrat.
 - $|x| = n$ heißt die **Länge** eines solchen Wortes.
 - Ein Wort der Länge 0 bezeichnen wir als leeres Wort ε .
- ❖ **Definition Regel**
 - Seien l und r Worte aus dem Zeichenvorrat V^* .
 - Eine **Regel** $l \rightarrow r$ bedeutet, dass wir das Wort l durch das Wort r ersetzen können.

Textersetzungssystem

- ❖ Definition: Eine Menge $T = \{a \rightarrow b, c \rightarrow d, \dots\}$ von Regeln über einem Zeichenvorrat V heißt **Textersetzungssystem** (auch **Semi-Thue System**), wenn die folgenden Metaregeln gelten:
 - Sei x ein Wort über V mit dem Teilwort $a \circ \dots \circ b$, d.h.
$$x = x_1 \circ x_2 \circ a \circ \dots \circ b \circ x_n$$
 - Wenn $a \circ \dots \circ b \rightarrow c \circ \dots \circ d$ eine Regel von T ist, dann kann man das Teilwort $a \circ \dots \circ b$ in x durch $c \circ \dots \circ d$ ersetzen:
$$x_1 \circ x_2 \circ a \circ \dots \circ b \circ x_n \Rightarrow x_1 \circ x_2 \circ c \circ \dots \circ d \circ x_n$$
 - Wenn $a \circ \dots \circ b$ mehrfach vorkommt oder mehrere Regeln anwendbar sind, so kann man das Teilwort bzw. die Regel beliebig wählen.
 - Die Regeln können beliebig oft angewandt werden.

Beispiel eines Textersetzungssystems: Addition von Strichzahlen

- ❖ Ein Algorithmus für die Addition von natürlichen Zahlen, dargestellt durch Folgen von Strichen |
 - z. B. die 5 wird durch |||| dargestellt.

- ❖ Ein Textersetzungssystem für diesen Algorithmus lautet:

Zeichenvorrat $V = \{|, +\}$

Regelmenge T:

$$(1) \quad +| \rightarrow |+$$

$$(2) \quad + \rightarrow \varepsilon$$

- ❖ Beispiele für Additionen:

$$| + | = ||$$

$$|| + || = ||||$$

Ableitung von III + II

$III+II \Rightarrow IIII+I$ -- Anwendung von Regel (1)

$\Rightarrow IIIII+$ -- Anwendung von Regel (1)

$\Rightarrow IIIII$ -- Anwendung von Regel (2)

Der Übergang $ulv \Rightarrow urv$ beschreibt die Texttransformation, die durch Anwendung einer Regel $l \rightarrow r$ auf dem Wort ulv entsteht. urv heißt aus ulv **abgeleitet** oder **erzeugt**.

Definition Direkte Ableitung: Eine Texttransformation durch Anwendung einer Regel. Wir benutzen das Symbol \Rightarrow zur Bezeichnung von direkten Ableitungen.

Definition Indirekte Ableitung:

Wir schreiben $l \overset{+}{\Rightarrow} r$, wenn r aus l durch mehrere direkte Ableitungen gewonnen werden kann. Also, beispielsweise $III+II \overset{+}{\Rightarrow} IIIII$. Wir schreiben $l \overset{*}{\Rightarrow} r$, wenn entweder $l = r$ ist oder wenn $l \overset{+}{\Rightarrow} r$ gilt, d.h., wenn r aus l indirekt ableitbar ist.

Multiplikation von Strichzahlen

- ❖ Eine Strichzahl besteht aus öffnender Klammer, Strichen und schließender Klammer: z. B. <llll> stellt die 5 dar.
- ❖ Multiplikator * Multiplikand = Resultat
- ❖ **Grundidee des Algorithmus:**
- ❖ 1. Ich schiebe nacheinander jeweils einen Strich vom Multiplikator in den Multiplikanden (roter Strich).
- ❖ 2. Dann schiebe ich den roten Strich durch alle Striche des Multiplikanden, und male ein kleines m für jeden blauen Strich im Multiplikanden.
- ❖ 3. Ich mache das, bis keine Striche mehr im Multiplikator sind.
- ❖ 4. Dann räume ich auf:
Ich lösche das Zeichen * und alle l's.
- ❖ 5. Damit habe ich das Resultat:
Ich konvertiere alle m's in l's.

Beispiel: <ll> * <lll>

⇒ <l> * <llll >

⇒ <l> * <mllll>

⇒ <l> * <m|mlll>

⇒ <l> * <m|m|mll>

⇒ <> * <mm|mm|mmlll>

⇒ <mmmmmm>

⇒ <llllll>

Regeln für das Textersetzungssystem

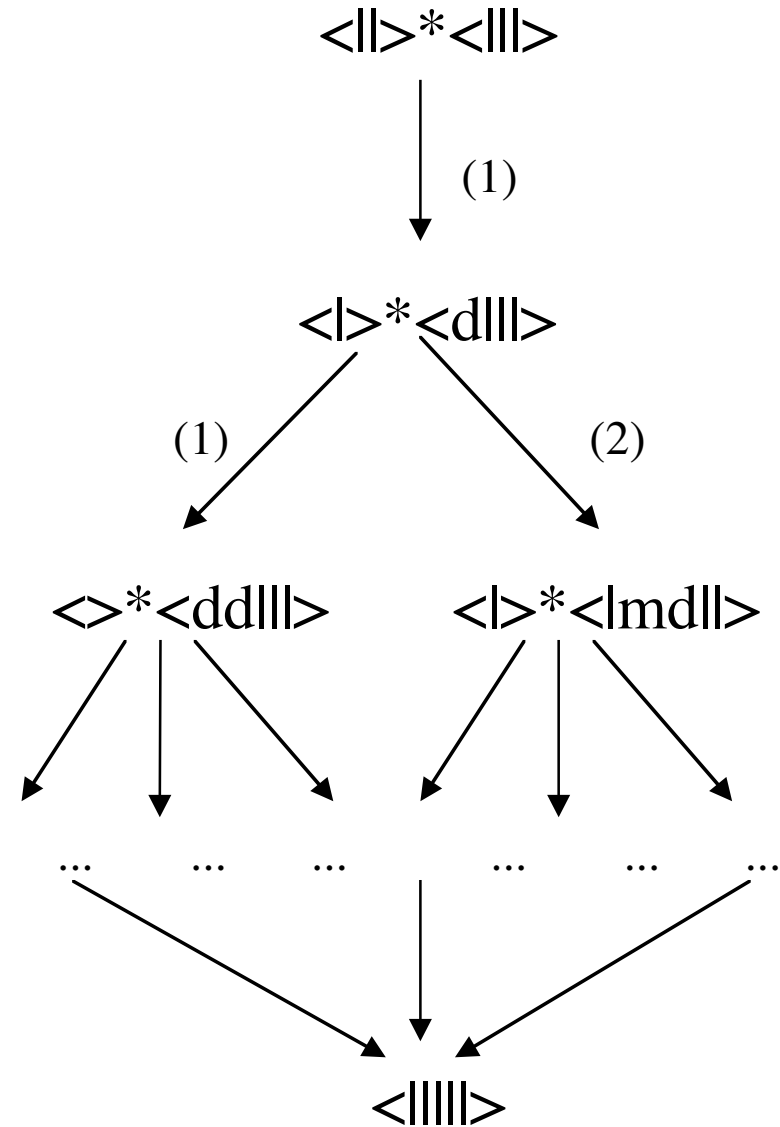
Strichzahlenmultiplikation

- ❖ Das Rüberziehen der | aus dem Multiplikator und das Verdoppeln der | im Multiplikanden machen wir mit 3 Hilfszeichen d, e, m (siehe Broy S. 39):
 - d (Duplikator, steht für den „roten Strich“), e (Eliminator) und m (Marker):

$ >*< \rightarrow >*<d$	(1) Rüberziehen eines aus dem Multiplikator
$d \rightarrow lmd$	(2) Verdoppeln der s im Multiplikanden
$dm \rightarrow md$	(3) Weiterschieben „des roten Strichs“
$d> \rightarrow >$	(4) Räume auf: Der „rote Strich“ hat seine Schuldigkeit getan
$\langle >*< \rightarrow <e$	(5) Wächter für die Zeichen $\langle >*<$, d.h. Erkennung der Aufräumbedingung: Setze den Eliminator ganz links!
$e \rightarrow e$	(6) Aufräumregel für das erste ganz links
$em \rightarrow le$	(7) Ersetzt jeweils einen Marker m durch ein und schiebt den Eliminator nach rechts
$e> \rightarrow >$	(8) Räume auf: Der Eliminator hat seine Schuldigkeit getan!

Textersetzungssysteme und Determinismus

- ❖ **Wichtig:** Die Folge der Anwendung von Regeln des Textersetzungssystems braucht nicht immer dieselbe zu sein, d.h. der Ableitungsgraph (die Menge aller möglichen Ableitungen) kann mehr als einen Pfad enthalten.
- ❖ **Definition: Ein Textersetzungssystem** heißt **deterministisch**, wenn zu jeder Zeit immer nur eine Regel anwendbar ist.
- ❖ **Definition: Ein Textersetzungssystem** heißt **determiniert**, wenn dieselbe Eingabe immer dieselbe Ausgabe ergibt.
- ❖ Die Strichzahlenmultiplikation ist also determiniert, aber nicht deterministisch.



Formale Sprache:

❖ **Definition Formale Sprache:**

- ❖ Sei l ein Wort über V^* und sei T ein Textersetzungssystem. Die Menge aller Worte r , die aus l abgeleitet werden können, heißt die formale Sprache $L_l = L(T, l)$ von l .
 - l heißt manchmal auch das Startsymbol oder Axiom des Textersetzungssystems.

❖ **Definition Alphabet:**

- ❖ Ist V endlich und ist auf V eine lineare Ordnung definiert, dann heißt V auch Alphabet.

Textersetzungssystem und Algorithmen

- ❖ Ein Textersetzungssystem zeigt die Grundform eines Algorithmus:
 - Die Eingabe ist ein Wort x
 - Es gibt nur endlich viele Operationen o_1, o_2, \dots, o_n , nämlich die Regeln des Textersetzungssystems.
 - Der Algorithmus ist ausführbar, indem man Regeln auf die Eingabe x anwendet.

Taxonomie von Algorithmen

Algorithmen unterscheiden sich nach

❖ **der Anwendbarkeit von Regeln**

- ◆ Terminierender Algorithmus
- ◆ Nichtterminierender Algorithmus
- ◆ Deterministischer Algorithmus
- ◆ Indeterministischer Algorithmus

❖ **der Beziehung zwischen Ein- und Ausgabe**

- ◆ Determinierter Algorithmus
- ◆ Indeterminierter Algorithmus

Algorithmenarten: Anwendbarkeit von Regeln

❖ **Terminierender Algorithmus:**

- Keine Regel ist mehr anwendbar.

❖ **Nichtterminierender Algorithmus:**

- Es ist immer mindestens eine Regel anwendbar.

❖ **Deterministischer Algorithmus:**

- Zu jeder Zeit ist immer nur eine Regel anwendbar, d.h. es gibt genau eine Berechnung.
- Falls er terminiert, liefert ein deterministischer Algorithmus genau eine Ausgabe.

❖ **Indeterministischer Algorithmus:**

- Zu einer gegebenen Eingabe x sind mehrere Regeln auf Teilwörter von x anwendbar.

Algorithmenarten: Beziehung zwischen Ein-und Ausgabe

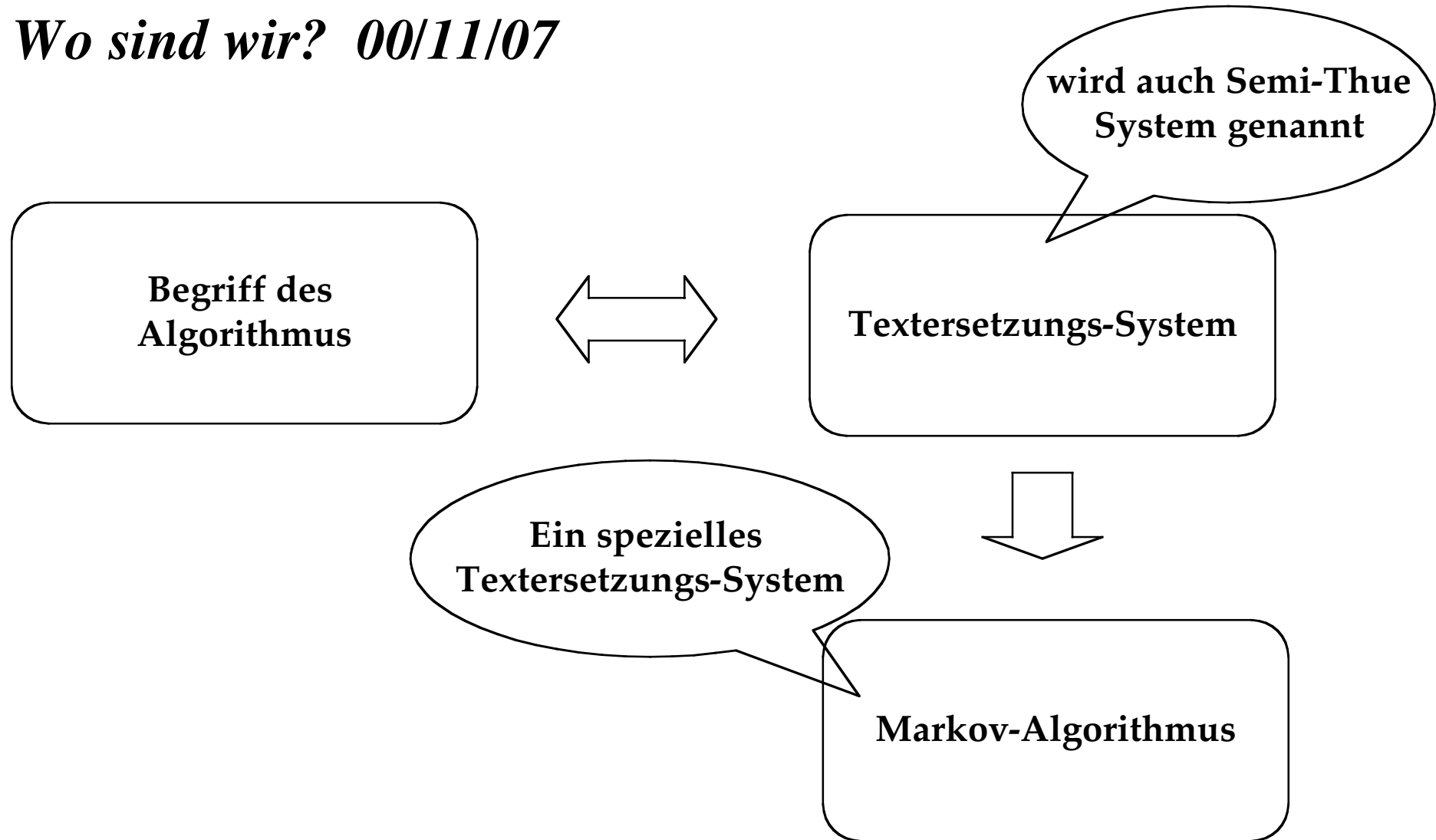
❖ **Determinierter Algorithmus:**

- Zu jeder Eingabe gibt es immer dieselbe Ausgabe. Der Algorithmus kann indeterministisch sein, d.h. es gibt verschiedene Berechnungspfade für dieselbe Eingabe.
 - ◆ Beispiel: Ich stehe an der alten Pinakothek, und will zum AudiMax. Ich kann entweder durch das Hauptgebäude gehen oder den Eingang in der Theresienstrasse oder den Eingang in der Gabelsbergerstrasse benutzen.

❖ **Indeterminierter Algorithmus:**

- Liefert bei wiederholter Anwendung auf die gleiche Eingabe unterschiedliche Ergebnisse.
 - ◆ Beispiel: Ein Zufallszahlengenerator ist ein indeterminierter Algorithmus.

Wo sind wir? 00/11/07



Überblick über den Vorlesungsblock

- ✓ Zentraler Begriff der Informatik: Algorithmus
- ✓ Unterschiedlichen Definitionen
- ✓ Taxonomie von Algorithmen
 - ✓ Terminierend, Nichtterminierend, Deterministisch, Nichtdeterministisch, Determiniert, Nichtdeterminiert
- Textersetzungs-systeme
 - ✓ Semi-Thue Systeme
 - ➡ Markov Algorithmen
- Definitionen: Formales System, Entscheidbarkeit, Berechenbarkeit
- Chomsky-Grammatik
- Backus-Naur-Form
- Syntaxdiagramm

Rubrik Sonstiges

- ❖ Skript: Die Vorlesungsfolien sind auf der Info1 Homepage und werden auch als Skript herausgebracht.
 - ◆ Ab heute ist der Berechtigungsschein zum Info1 Skript im Skriptenverkauf erhältlich
 - ◆ Kosten diese Woche: 12 DM
 - ◆ Kosten ab nächster Woche: höher
 - ◆ Gedruckt werden erst einmal die ersten 4 Vorlesungen, verfügbar ab Montag nächster Woche
 - ◆ Dann regelmäßige Updates
- ❖ Definition von Algorithmen und das Neueste aus der Politik
 - Am 7. November fanden die Präsidentschaftswahlen in den USA statt. Der neue Präsident ist also gewählt.
 - Sende eine Glücksbotschaft an den gewählten Präsidenten der USA
 - Ist das ein Algorithmus?

Definition des Algorithmus

- ❖ Ist dies ein Algorithmus: "Sende eine Glücksbotschaft an den gewählten Präsidenten der USA"
- ❖ **Definition Algorithmus (Siehe Folie 6)**
 - Ein Algorithmus ist ein Verfahren zur Verarbeitung von Daten mit einer **präzisen, endlichen** Beschreibung unter Verwendung **effektiver** Arbeitsschritte
- ❖ Präzise: In einer eindeutigen Sprache abgefaßt
- ❖ Endlich: In einer endlichen Form beschrieben
- ❖ Effektiv: Die Arbeitsschritte sind tatsächlich ausführbar

- ❖ Ein Algorithmus muß nicht terminieren

Markov-Algorithmen

- ❖ **Definition:** Ein Markov-Algorithmus ist ein *deterministisches Textersetzungssystem* mit endlichen vielen Regeln, sowie einigen speziellen Regeln, Zeichen und 2 Metaregeln:
 - Markov-Algorithmen enthalten spezielle Regeln, auch **haltende Regeln** $x \rightarrow \cdot$ genannt, die durch einen Punkt nach dem Pfeil \rightarrow gekennzeichnet sind.
 - Markov-Algorithmen enthalten zusätzliche Zeichen a, b, c..., sogenannte Schiffchen.
- ❖ Markov-Algorithmen haben immer 2 spezielle Anweisungen ("**Metaregeln**") für die Ausführung von Regeln:
 - 1. Wähle in jedem Schritt die erste anwendbare Regel. Falls sie auf mehrere Teilwörter anwendbar ist, wende sie auf das am weitesten links stehende Teilwort an.
 - 2. Wende die Regeln solange an, bis eine haltende Regel angewandt wurde, oder bis keine Regel mehr anwendbar ist.

Markov Algorithmus Beispiel

Zeichenvorrat $V = \{0,L\}$

Schiffchen: a,b

Regelsystem:

(1) $aL \rightarrow La$

(2) $a0 \rightarrow 0a$

(3) $a \rightarrow b$

(4) $Lb \rightarrow b0$

(5) $0b \rightarrow \cdot L$

(6) $b \rightarrow \cdot L$

(7) $\varepsilon \rightarrow a$

❖ Anwendung der Regeln auf das Eingabewort LOLL

LOLL \Rightarrow aLOLL (7)

\Rightarrow LaOLL (1)

\Rightarrow L0aLL (2)

\Rightarrow L0LaL (1)

\Rightarrow L0LLa (1)

\Rightarrow L0LLb (3)

\Rightarrow L0Lb0 (4)

\Rightarrow L0b00 (4)

\Rightarrow LL00 (5)

Frage: Welches Problem löst der Algorithmus?

Noch ein Beispiel

Zeichenvorrat $V = \{0,L\}$

Schiffchen: a,b

Regelsystem:

$$(1) \quad \varepsilon \rightarrow a$$

$$(2) \quad a0 \rightarrow 0a$$

$$(3) \quad a \rightarrow b$$

$$(4) \quad Lb \rightarrow b0$$

$$(5) \quad 0b \rightarrow \cdot L$$

$$(6) \quad b \rightarrow \cdot L$$

$$(7) \quad aL \rightarrow La$$

❖ Anwendung der Regeln auf das Eingabewort LOLL

$$L0LL \Rightarrow aL0LL \quad (1)$$

$$aaL0LL \quad (1)$$

Frage: Was macht dieser Algorithmus?

Vergleich der Algorithmen

Algorithmus 1 terminiert

$$(1) \quad aL \rightarrow La$$

$$(2) \quad a0 \rightarrow 0a$$

$$(3) \quad a \rightarrow b$$

$$(4) \quad Lb \rightarrow b0$$

$$(5) \quad 0b \rightarrow \cdot L$$

$$(6) \quad b \rightarrow \cdot L$$

$$(7) \quad \varepsilon \rightarrow a$$

Algorithmus 2 terminiert nicht

$$(1) \quad \varepsilon \rightarrow a$$

$$(2) \quad a0 \rightarrow 0a$$

$$(3) \quad a \rightarrow b$$

$$(4) \quad Lb \rightarrow b0$$

$$(5) \quad 0b \rightarrow \cdot L$$

$$(6) \quad b \rightarrow \cdot L$$

$$(7) \quad aL \rightarrow La$$

Formale Systeme

- ❖ Definition Formales System:
- ❖ Wir nennen (U, \Rightarrow) ein formales System,
 - wenn U eine endliche (abzählbare unendliche) Menge ist
 - wenn die Ableitung \Rightarrow eine Relation ist über $U \times U$ ist, d.h.
 $\Rightarrow \subseteq U \times U$
 - wenn es einen Algorithmus gibt, der für jedes $l \Rightarrow r$
 - ◆ r in endlich vielen Schritten aus l berechnen kann.
 - ◆ r heißt dann **effektiv berechenbar** oder kurz **berechenbar**.
- ❖ Beispiel: Sei W eine Menge von Wörtern über einem Zeichenvorrat V , und \Rightarrow die Ableitungsrelation über den Regeln eines Markov-Algorithmus. Dann gilt
 - (W, \Rightarrow) ist ein formales System.

Entscheidbarkeit

❖ **Definition: Entscheidbarkeit**

❖ Ein formales System U heißt **entscheidbar**, wenn für beliebige Worte $l, r \in U$ effektiv festgestellt werden kann, ob $l \xRightarrow{*} r$ gilt oder nicht, d.h. ob r aus l ableitbar ist.

❖ **Definition: Kalkül**

❖ Ein formales System (U, \Rightarrow) heißt ein **Kalkül**, wenn die Ableitungsrelation \Rightarrow durch eine **endliche Menge von Regeln** zusammen mit einer **endlichen Menge von Metaregeln** definiert ist.

– **Beispiel:** Ein Markov-Algorithmus ist ein Kalkül, denn er hat nur 2 Metaregeln (Siehe Folie 34) und ist immer durch eine endliche Menge von Regeln definiert.

Chomsky Grammatiken

- ❖ Die natürliche Sprache enthält Regeln wie
 - Ein „Satz“ besteht aus „Substantiv“ und „Verb“
 - Wenn es stimmt, dass „Fisch“ ein Substantiv ist, und „schwimmt“ ein Verb, dann ist „Fisch schwimmt“ ein „Satz“.
- ❖ Die natürliche Sprache hat viele solcher Regeln
- ❖ Noam Chomsky war der erste, der versucht hat, die Regeln der natürlichen Sprache in Form eines Textersetzungssystems (Semi-Thue Systems) zu formulieren.
 - Chomsky nannte diese Semi-Thue Systeme dann **Grammatiken** und die Regeln **Produktionen**

Syntaktische Variable und Terminale

- ❖ In einer Grammatik unterscheidet man syntaktische Begriffe wie *Satz, Substantiv, Verb* von den Wörtern der zu beschreibenden Sprache wie *Fisch, schwimmen*.
- ❖ Die Einzelzeichen des Zeichenvorrates T einer Grammatik heißen **Terminale**.
- ❖ Die syntaktischen Begriffe bilden den Zeichenvorrat N der **Nichtterminale** oder **syntaktische Variablen** einer Grammatik.
- ❖ Ziel einer Grammatik ist es, die terminalen Zeichenreihen zu beschreiben, die aus einer speziellen syntaktischen Variablen abgeleitet werden können, dem Startsymbol, **Axiom** oder **Ziel** Z der Grammatik.

Definition Chomsky Grammatik

- ❖ Die Vereinigung $V = T \cup N$ heißt das **Vokabular** der Grammatik G , bzw. der formalen Sprache $L = L(G, Z)$
- ❖ **Definition Phrase:** Eine Zeichenreihe x aus V^* , die durch endliche viele Anwendungen von Regeln aus dem Ziel Z abgeleitet werden kann:

$$Z \xRightarrow{*} x$$

- ❖ Die Sprache $L(G, Z)$ oder kurz $L(G)$ besteht aus allen terminalen Phrasen.
- ❖ **Definition Chomsky-Grammatik:**
 - Eine Grammatik $G = (T, N, P, Z)$ in der T ein Zeichenvorrat von Terminalen, N ein Zeichenvorrat von Nichtterminalen, P eine endliche Menge von Produktionen und Z ein Axiom ist, heißt eine Chomsky-Grammatik.
- ❖ **Definition Produktion:** Eine Regel in einer Chomsky-Grammatik

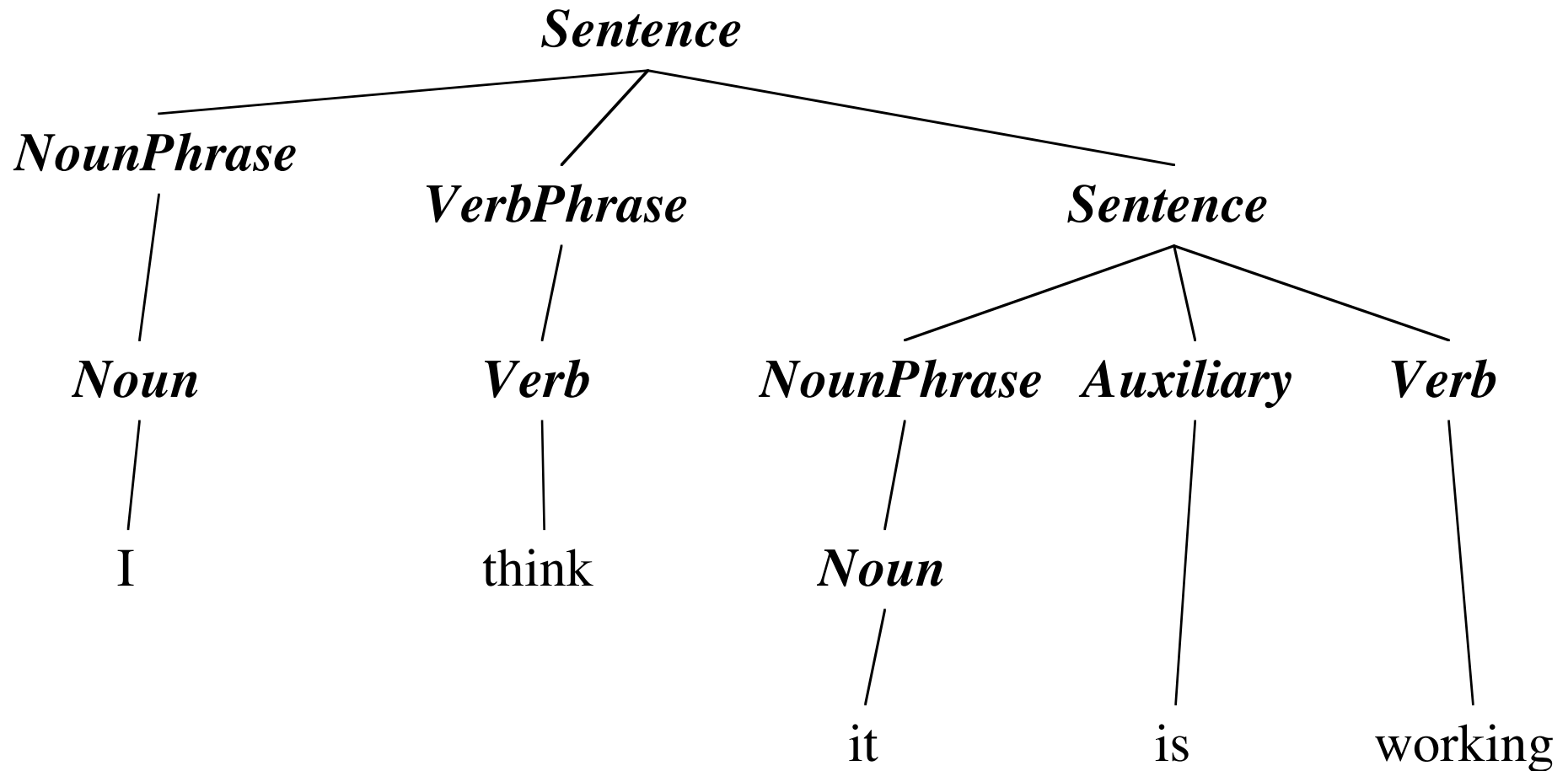
Einfaches Beispiel einer Chomsky Grammatik

- ❖ Gegeben sei eine Chomsky-Grammatik $G_A = (T, N, P, Z)$ mit
- ❖ **Nichtterminale** $N = \{\text{Sentence, NounPhrase, VerbPhrase, Noun, Verb, Auxiliary}\}$
- ❖ **Terminale** $T = \{\text{I, is, it, think, working}\}$
- ❖ **Axiom** $Z = \text{Sentence}$
- ❖ **Produktionen** $P = \{$
 - Sentence \rightarrow NounPhrase VerbPhrase Sentence
 - Sentence \rightarrow NounPhrase Auxiliary Verb
 - NounPhrase \rightarrow Noun
 - VerbPhrase \rightarrow Verb
 - Verb \rightarrow think
 - Verb \rightarrow working
 - Noun \rightarrow I
 - Noun \rightarrow it
 - Auxiliary \rightarrow is

Ableitungsbaum

Ein Ableitungsbaum ist die graphische Darstellung einer Ableitung.

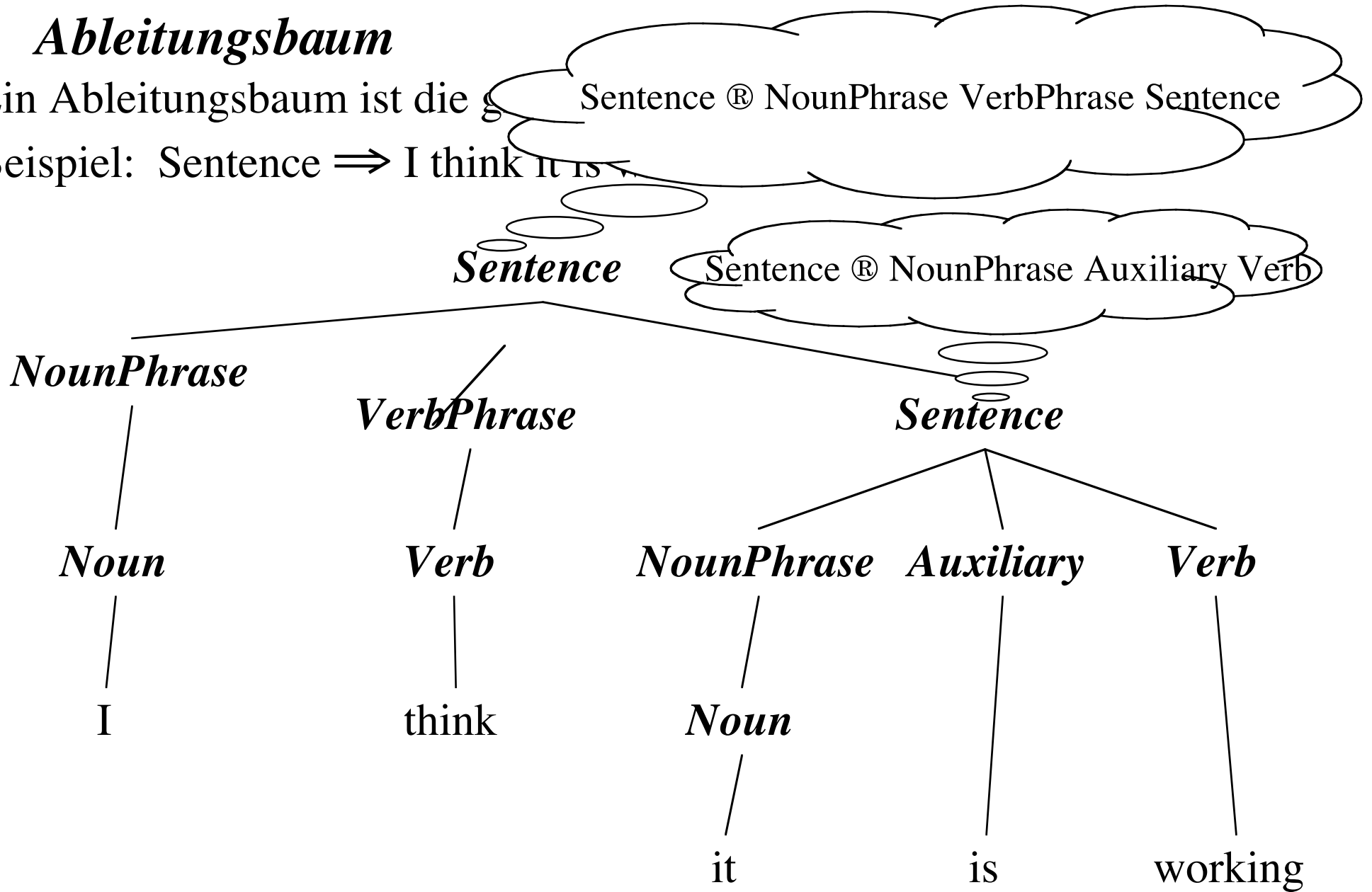
Beispiel: Sentence \Rightarrow I think it is working



Ableitungsbaum

Ein Ableitungsbaum ist die

Beispiel: Sentence \Rightarrow I think it is



Ein etwas komplizierteres Beispiel

- ❖ Die Menge aller arithmetischen Ausdrücke mit den Bezeichnern a , b , c und den Operatoren $+$ und $*$ ist gegeben durch die folgende Chomsky-Grammatik $G_A = (T, N, P, Z)$ mit
- ❖ $T = \{a, b, c, +, *, (,)\}$
- ❖ $N = \{\text{Ausdruck, Term, Faktor, Bezeichner}\}$
- ❖ $P = \{ \text{Ausdruck} \rightarrow \text{Term},$
 $\text{Ausdruck} \rightarrow \text{Ausdruck} + \text{Term},$
 $\text{Term} \rightarrow \text{Faktor},$
 $\text{Term} \rightarrow \text{Term} * \text{Faktor},$
 $\text{Faktor} \rightarrow \text{Bezeichner},$
 $\text{Faktor} \rightarrow (\text{Ausdruck}),$
 $\text{Bezeichner} \rightarrow a, \text{Bezeichner} \rightarrow b, \text{Bezeichner} \rightarrow c \}$
- ❖ $Z = \text{Ausdruck}$

Beispiel: $(a+b) * c + d$ ist ein Ausdruck

Produktionen P =

{ Ausdruck \rightarrow Term -- 1
 Ausdruck \rightarrow Ausdruck + Term --2
 Term \rightarrow Faktor --3
 Term \rightarrow Term * Faktor --4
 Faktor \rightarrow Bezeichner --5
 Faktor \rightarrow (Ausdruck) --6
 Bezeichner \rightarrow a | b | c | d --7
 }

Axiom Z = Ausdruck

Terminale T = {a, b, c, d, +, (,), *}

Nichtterminale N = {Ausdruck, Term, Faktor, Bezeichner}

Vokabular V = {a, b, c, d, (,), +, *,
Ausdruck, Term, Faktor, Bezeichner}

❖ Abkürzungen: A = Ausdruck, T = Term, F = Faktor, B = Bezeichner

❖ $A \Rightarrow A+T$ --2
 ❖ $\Rightarrow T+ T$ --1
 ❖ $\Rightarrow T*F+T$ --4
 ❖ $\Rightarrow F*F+T$ --3
 ❖ $\Rightarrow (A)*F+T$ --6
 ❖ $\Rightarrow (A+T)*F+T$ --2
 ❖ $\Rightarrow (T+T)*F+T$ --1
 ❖ $\Rightarrow (F+T)*F+T$ --3
 ❖ $\Rightarrow (B+T)*F+T$ --5
 ❖ $\Rightarrow (B+F)*F+T$ --3
 ❖ $\Rightarrow (B+B)*F+T$ --5
 ❖ $\Rightarrow (B+B)*B+T$ --5
 ❖ $\Rightarrow (B+B)*B+F$ --3
 ❖ $\Rightarrow (B+B)*B+B$ --5
 ❖ $\Rightarrow (a+b)*c+d$ --7,7,7,7

Kompaktere Notation für Grammatiken

- ❖ Grammatiken verwendet man zur Beschreibung der Syntax von Programmiersprachen. Um die Beschreibung kompakt zu halten, werden folgende Konventionen benutzt.

- Statt \rightarrow schreibt man $::=$
- Nichtterminale werden in $\langle \rangle$ eingeschlossen, z.B. $\langle \text{Verb} \rangle$
- Der senkrechte Strich "|" trennt rechte Seiten zur gleichen linken Seite von Produktionen.

- ❖ Beispiel: Die beiden Produktionen

Verb \rightarrow think

Verb \rightarrow working

schreibt man als

$\langle \text{Verb} \rangle ::= \text{think} \mid \text{working}$

- ❖ Die Notation stammt von John Backus (1959) und wurde von Peter Naur bei der Definition der Grammatik für Algol60 eingesetzt. Deshalb auch Backus-Naur-Form (BNF) genannt.

Konvertierung von Produktionen in Backus-Naur-Form

P = { Sentence \rightarrow NounPhrase VerbPhrase Sentence
Sentence \rightarrow NounPhrase Auxiliary Verb
NounPhrase \rightarrow Noun, VerbPhrase \rightarrow Verb
Verb \rightarrow think, Verb \rightarrow working
Noun \rightarrow I, Noun \rightarrow it
Auxiliary \rightarrow is
}

<Sentence> ::= <NounPhrase> <VerbPhrase> <Sentence> |
 <NounPhrase> <Auxiliary> <Verb>
<NounPhrase> ::= <Noun>
<VerbPhrase> ::= <Verb>
<Verb> ::= think | working
<Noun> ::= I | it
<Auxiliary> ::= is

Backus-Naur-Form für Ausdrücke

Produktionen

$P = \{$
Ausdruck \rightarrow Term ,
Ausdruck \rightarrow Ausdruck + Term
Term \rightarrow Faktor
Term \rightarrow Term * Faktor
Faktor \rightarrow Bezeichner
Faktor \rightarrow (Ausdruck)
Bezeichner \rightarrow a | b | c
 $\}$

Backus-Naur-Form

$\langle \text{Ausdruck} \rangle ::= \langle \text{Term} \rangle \mid$
 $\langle \text{Ausdruck} \rangle + \langle \text{Term} \rangle$
 $\langle \text{Term} \rangle ::= \langle \text{Faktor} \rangle \mid$
 $\langle \text{Term} \rangle * \langle \text{Faktor} \rangle$
 $\langle \text{Faktor} \rangle ::= \langle \text{Bezeichner} \rangle \mid$
 $(\langle \text{Ausdruck} \rangle)$
 $\langle \text{Bezeichner} \rangle ::= a \mid b \mid c$

Erweiterungen in der Backus Naur Form

- ❖ Gleich nach Erscheinen der Backus-Naur Form wurden zusätzliche Notationen eingeführt:
 - [...] bezeichnet optionale Teile auf einer rechten Seite
 - (...) umschließt eine Gruppe von Zeichen
 - | in einer runden oder eckigen Klammer trennt Alternativen
 - Ein Stern * nach einem Zeichen oder einer Gruppe in einer runden Klammer bezeichnet die optionale oder n-fache Wiederholung des Zeichens oder der Gruppe
 - Ein Pluszeichen + nach einem Zeichen oder einer Gruppe bezeichnet die n-fache Wiederholung des Zeichens oder der Gruppe.
- ❖ Die Sprache, die durch diese Notationen definiert wird, heißt auch die Sprache der **regulären Ausdrücke**. Wir geben jetzt eine rekursive Definition für reguläre Ausdrücke.

Rekursive Definition von Regulären Ausdrücken

00/11/13

- ❖ Sei T ein Zeichenvorrat von Terminalen.
- ❖ Jedes Terminalzeichen $t \in T$ ist ein regulärer Ausdruck mit Sprache $\{t\}$
- ❖ Seien R und Q reguläre Ausdrücke mit Sprachen X und Y . Dann gilt

Vereinigung: $R \mid Q$ ist ein regulärer Ausdruck mit der Sprache $X \cup Y$.

Konkatenation: RQ ist ein regulärer Ausdruck mit der Sprache
 $\{x \circ y: x \in X \text{ und } y \in Y\}$

Option: $\{R\}$ ist ein regulärer Ausdruck mit der Sprache $X \cup \varepsilon$

Iteration: $\{R\}^*$ ist ein regulärer Ausdruck mit der Sprache
 $\{x_1 \circ x_2 \dots \circ x_n: x_1, x_2 \dots, x_n \in X\}$

$\{R\}^+$ ist ein regulärer Ausdruck mit der Sprache
 $\{x_1 \circ x_2 \dots \circ x_n: n \in \mathbb{N} \text{ und } x_1, x_2 \dots, x_n \in X\}$

Klammerung: $[R]$ ist ein regulärer Ausdruck mit der Sprache X .

Leere Sprache: $\{\}$ ist ein regulärer Ausdruck mit der Sprache \emptyset

Grammatik für Dezimalzahlen in der erweiterten Backus-Naur-Form

Beispiele: 0.34 3.4 -3.4

Vorschlag einer Grammatik:

$\langle \text{Ziffer} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\langle \text{Zahl} \rangle ::= (\langle \text{Ziffer} \rangle)^* . (\langle \text{Ziffer} \rangle)^*$

Ist . oder .4 eine Dezimalzahl?

Iteration 1: Vor dem Punkt muß mindestens eine Ziffer stehen

$\langle \text{Ziffer} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\langle \text{Zahl} \rangle ::= \langle \text{Ziffer} \rangle (\langle \text{Ziffer} \rangle)^* . (\langle \text{Ziffer} \rangle)^*$

Ist 4. eine Dezimalzahl?

Iteration 2: Nach dem Punkt muß mindestens auch eine Ziffer stehen

$\langle \text{Ziffer} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\langle \text{Zahl} \rangle ::= \langle \text{Ziffer} \rangle (\langle \text{Ziffer} \rangle)^* . (\langle \text{Ziffer} \rangle)^* \langle \text{Ziffer} \rangle$

Weitere Iterationen ...

Wie ist es mit - 3.4?

Iteration 3: Wir erlauben negative Dezimalzahlen

$\langle \text{Ziffer} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\langle \text{Zahl} \rangle ::= [-] \langle \text{Ziffer} \rangle (\langle \text{Ziffer} \rangle)^* . (\langle \text{Ziffer} \rangle)^* \langle \text{Ziffer} \rangle$

Sind 0 oder 03.4 oder 03.04 Dezimalzahlen?

Iteration 4: Wir verbieten führende und folgende Nullen (indem wir zwischen Ziffern und positiven Ziffern unterscheiden):

$\langle \text{Pziffer} \rangle ::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\langle \text{Ziffer} \rangle ::= 0 \mid \langle \text{Pziffer} \rangle$

$\langle \text{Zahl} \rangle ::= (0 \mid ([-] \langle \text{Pziffer} \rangle (\langle \text{Ziffer} \rangle)^*)) . ((\langle \text{Ziffer} \rangle)^* \langle \text{Pziffer} \rangle)$

Arithmetische Ausdrücke in BNF

Beispiele für arithmetische Ausdrücke:

$$(4+5)*6$$

4

(4)

(4*5)

4*(5/7)+(7*8)

- ❖ Die Sprache der arithmetischen Ausdrücke über den ganzen Zahlen $0, 1, 2, \dots$ mit den Operationen $+, -, *, /$ ist eine formale Sprache.

<Arith_Ausdruck> ::=

<Zahl> |

<Arith_Ausdruck> (+ | - | * | /) <Arith_Ausdruck> |

(<Arith_Ausdruck>)

<Zahl> ::= (<Ziffer> (<Ziffer> | 0)*) | 0

<Ziffer> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Frage: Wie können wir erreichen, daß $4+5*6$ dieselbe Ableitung wie $4+(5*6)$ ergibt, aber nicht dieselbe wie $(4+5)*6$?

Antwort: Einführung von Prioritäten für die Operatoren $*$ und $/$.

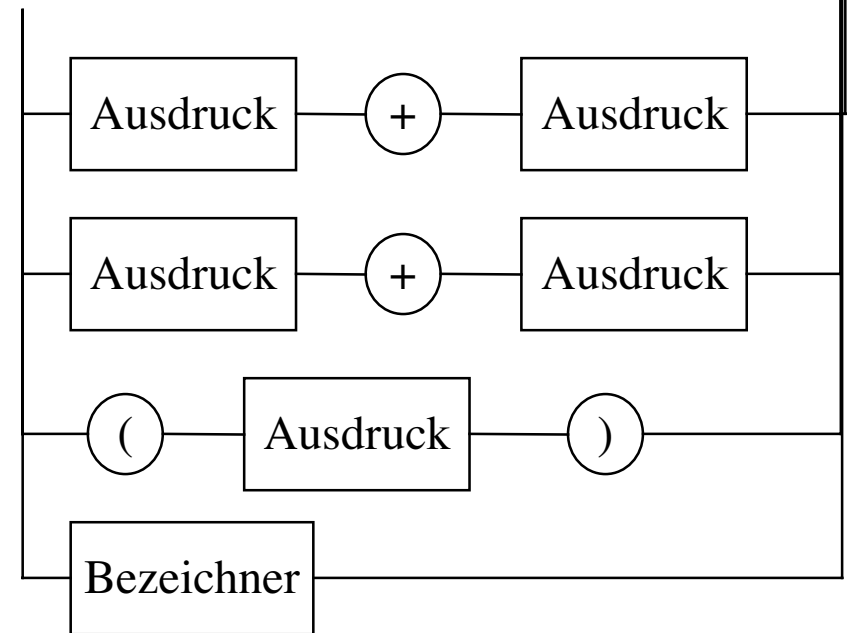
Syntaxdiagramm

- ❖ Statt in BNF kann man die Produktionen auch graphisch durch Syntaxdiagramme darstellen.
- ❖ Jedes Diagramm hat einen Namen, nämlich das Nichtterminal, das es repräsentiert
 - Es hat genau einen Eingang und einen Ausgang.
 - Jeder Weg vom Eingang zum Ausgang ergibt eine gültige Ableitung.
- ❖ Ein Syntaxdiagramm besteht aus Ovalen (Kreisen) und Rechtecken verbunden durch Kanten.
 - Ovale oder Kreise bezeichnen Terminale
 - Rechtecke bezeichnen Nichtterminale

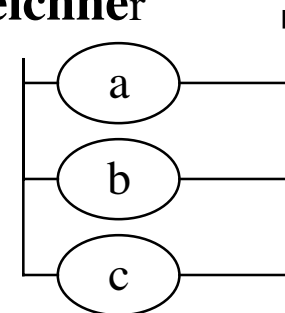
Beispiel: Konvertierung von Produktionen in ein Syntaxdiagramm

- ❖ $G_A = (T, N, P, Z)$ (Siehe Folie 50)
- ❖ $T = \{a, b, c, +, *\}$
- ❖ $N = \{\text{Ausdruck, Term, Faktor, Bezeichner}\}$
- ❖ $P = \{ \text{Ausdruck} \rightarrow \text{Term},$
 $\text{Ausdruck} \rightarrow \text{Ausdruck} + \text{Term},$
 $\text{Term} \rightarrow \text{Faktor},$
 $\text{Term} \rightarrow \text{Term} * \text{Faktor},$
 $\text{Faktor} \rightarrow \text{ID}$

Ausdruck



Bezeichner



- Syntaxdiagramme sind im allgemeinen rekursiv
- Syntaxdiagramme benötigen weniger Nichtterminale als BNF

Zusammenfassung

- ❖ Der Begriff des Algorithmus ist eine der wichtigsten Säulen der Informatik (Leider gibt es mehrere Definitionen).
- ❖ Klassifikation von Algorithmen nach Anwendbarkeit der Regeln und nach der Beziehung zwischen Ein- und Ausgabe
- ❖ Textersetzungs-systeme
- ❖ Markov-Algorithmen als Beispiel für deterministische Algorithmen
- ❖ Definitionen: Formales System, Entscheidbarkeit, Berechenbarkeit, Grammatik
- ❖ Die Backus-Naur-Form und Syntaxdiagramme sind nützlich für die kompakte Beschreibung von (kontextfreien) Grammatiken