

# **Grundlagen der Programmierung**

Dr. Christian Herzog  
Technische Universität München

Wintersemester 2016/2017

## **Kapitel 4: Algorithmen und Textersetzungssysteme**

### **Überblick über dieses Kapitel**

- ❖ Zentraler Begriff der Informatik: Algorithmus
- ❖ Unterschiedliche Definitionen
- ❖ Taxonomie von Algorithmen
- ❖ Textersetzungssysteme
  - Semi-Thue Systeme
  - Markov-Algorithmen
- ❖ Chomsky-Grammatik
- ❖ Backus-Naur-Form, Syntaxdiagramme

### **Arten der Funktionsbeschreibung**

- ❖ 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.  $g(a, b) = 0$ , falls  $b = f(a)$ . 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 elementaren Operationen mit hinreichender Präzision spezifizieren.
- ❖ Eine solche Beschreibung heißt **Algorithmus**.
- ❖ Beispiele von Algorithmen:
  - Verfahren zur schriftlichen Addition, Subtraktion, Multiplikation, Division
  - Rezept zum Herstellen von Semmeln
  - Spielregel für Dame oder Mühle oder Halma ...

## *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 verwendet diese Definitionen.

## *Was ist präzise, endlich, effektiv?*

### ❖ **Präzise:** in einer eindeutigen Sprache abgefasst

- maschinell lesbar und ausführbar
- Eine Beschreibung ist unpräzise, wenn eine Bedingung nicht eindeutig ausgewertet werden kann.  
Beispiel: **Lasse den Braten schmoren, bis er gut durch ist.**
- ❖ **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:  
Wenn bei optimalem Spiel Weiß bei Schach immer gewinnt, ...

## *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 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 dieser Vorlesung benutzen wir die erste Definition, d.h. nicht jeder Algorithmus terminiert.

## 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?
- ❖ Was kommt jetzt?
  - **Zeichenfolgen**
  - **Textersetzungssysteme** als einfache Algorithmen
    - ◆ Beispiele von Textersetzungssystemen
    - ◆ Markov-Algorithmen als deterministische Algorithmen
  - **Grammatiken** zur Spezifikation von Textersetzungssystemen
  - Backus-Naur-Form und Syntax-Diagramme

## Zeichenfolgen

- ❖ 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.
- ❖ Mit  $V$  bezeichnen wir im folgenden den zugrunde liegenden **Zeichenvorrat**: eine endliche Menge von Zeichen.
- ❖ Gibt es auf  $V$  eine lineare Ordnung (d.h. eine feste Reihenfolge der Zeichen), so nennt man  $V$  ein **Alphabet**.
- ❖ Eine Zeichenfolge über einem Zeichenvorrat  $V$  wird auch als **Wort** über  $V$  bezeichnet.

## Definitionen

- ❖ **Definition Wort:** Für eine gegebene Menge  $V$  von Zeichen bildet eine Folge  $x = X_1, X_2, \dots, X_n$  mit  $X_i \in V$  ein Wort der Länge  $n$  über  $V$ . Wir schreiben auch  $x = X_1 X_2 \dots X_n$  für das Wort und sprechen von einem  $n$ -Tupel.
- ❖  $|x|$  bezeichnet die Länge von  $x$ , d.h.  $|X_1 X_2 \dots X_n| = n$
- ❖  $\varepsilon$  bezeichnet die leere Zeichenfolge bzw. das leere Wort,  $|\varepsilon| = 0$
- ❖  $V^n$  bezeichnet die Menge aller  $n$ -Tupel bzw. die Menge aller Worte der Länge  $n$  über  $V$ .
  - $V^n =_{\text{def}} V \times V \times \dots \times V$  ( $n$ -faches Kreuzprodukt über  $V$ )
  - $V^0 = \{\varepsilon\}$
- ❖  $V^*$  bezeichnet die Menge aller Worte über  $V$ , d.h. aller endlichen Folgen von Zeichen aus  $V$ .
  - $V^* = V^0 \cup V^1 \cup V^2 \cup V^3 \cup \dots \cup V^n \cup \dots$  (unendliche Vereinigung)
- ❖  $V^+$  bezeichnet die Menge aller Worte über  $V$  ohne das leere Wort, d.h.  $V^+ = V^* \setminus V^0 = V^* \setminus \{\varepsilon\}$ .

## Konkatenation von Zeichenfolgen

- ❖ Über  $V^*$  definieren wir als Operation die zweistellige Abbildung **Konkatenation**:
  - $\text{conc}: V^* \times V^* \rightarrow V^*$
- ❖ Das Wort  $\text{conc}(v, w)$  mit  $v, w \in V^*$  ist dasjenige Wort, das durch Aneinanderreihen („Konkatenerieren“) der Zeichenfolgen  $v$  und  $w$  gebildet wird.
  - Ist  $v = V_1 V_2 \dots V_m$  und  $w = W_1 W_2 \dots W_n$ , so ist  $\text{conc}(v, w) = V_1 V_2 \dots V_m W_1 W_2 \dots W_n$
- ❖ Konkatenation wird oft in Infixweise geschrieben:
  - $v \circ w =_{\text{def}} \text{conc}(v, w)$
- ❖ Für die Konkatenation gelten folgende Gesetze:
  - $(u \circ v) \circ w = u \circ (v \circ w)$  (Assoziativität)
  - $w \circ \varepsilon = w = \varepsilon \circ w$  (Neutrales Element)
- ❖  $V^*$  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.
- ❖ **Definition Regel**
  - Seien  $l$  und  $r$  Worte aus  $V^*$ .
  - Eine **Regel**  $l \rightarrow r$  bedeutet, dass wir das Auftreten des Wortes  $l$  durch das Auftreten des Wortes  $r$  ersetzen können.
  - $l$  heißt die *linke Seite* der Regel,  $r$  die *rechte Seite*.

## Definition 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  $w$  ein Wort über  $V$  mit dem Teilwort  $a$ , d.h.  $w$  ist von der Form  $w = v \circ a \circ x$  mit geeigneten  $v, x \in V^*$
  - Wenn  $a \rightarrow b$  eine Regel von  $T$  ist, dann kann man das Teilwort  $a$  in  $w$  durch  $b$  ersetzen:
 
$$v \circ a \circ x \Rightarrow v \circ b \circ x$$
  - Wenn  $a$  in  $w$  mehrfach vorkommt oder mehrere Regeln anwendbar sind, so kann man das Teilwort bzw. die Regel beliebig wählen.
  - Die Regeln können beliebig oft angewandt werden.
  - $v \circ b \circ x$  heißt (**direkt**) **abgeleitet** aus  $v \circ a \circ x$  über die Regel  $a \rightarrow b$ .
    - ◆ Wir verwenden für die direkte Ableitung das Symbol  $\Rightarrow$ .

## Beispiel eines Textersetzungssystems: Addition von Strichzahlen

- ❖ Ein Algorithmus für die Addition von natürlichen Zahlen, dargestellt durch Folgen von Strichen  $|$ , die durch die Sonderzeichen  $<$  und  $>$  begrenzt sind.
  - z. B. die 5 wird durch  $<|||>$  dargestellt.
- ❖ Ein Textersetzungssystem für diesen Algorithmus lautet:
  - Zeichenvorrat  $V = \{<, |, >, +\}$
  - Die Regelmengemenge  $T$  besteht nur aus einer einzigen Regel:
 
$$>+< \rightarrow \varepsilon$$
- ❖ Beispiele für Ableitungen:
 
$$<|>+<|> \Rightarrow <||>$$

$$<|||>+<||> \Rightarrow <|||>$$

## Multiplikation von Strichzahlen

Beispiel:

$$\begin{aligned} &<||>^* <|||> \\ \Rightarrow &<|>^* <|||> \\ \dots \Rightarrow &<|>^* <|m||> \\ \dots \Rightarrow &<|>^* <|m|m||> \\ \dots \Rightarrow &<|>^* <|m|m|m||> \\ \dots \Rightarrow &<>^* <|mm|mm|mm||> \\ \dots \Rightarrow &<mmmmmm> \\ \dots \Rightarrow &<|||> \end{aligned}$$

- ❖ **Grundidee des Algorithmus:**
  1. Ich schiebe einen Strich vom linken Faktor in den rechten (roter Strich).
  2. Dann schiebe ich den roten Strich durch alle Striche des rechten Faktors, und schreibe ein  $m$  für jeden gefundenen blauen Strich.
  3. Ich wiederhole 1. und 2., bis keine Striche mehr im linken Faktor sind.
  4. Dann räume ich auf: Ich lösche den (leeren) linken Faktor, das Zeichen  $*$  und alle  $|$ .
  5. Damit habe ich das Resultat: Ich muss nur noch alle  $m$  in  $l$  konvertieren.



## *Textersetzungs-systeme und Algorithmen*

- ❖ Ein Textersetzungs-system 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 Textersetzungs-systems.
  - 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
  - Nichtdeterministischer Algorithmus
- ❖ **der Beziehung zwischen Ein- und Ausgabe**
  - Determinierter Algorithmus
  - Nichtdeterminierter Algorithmus

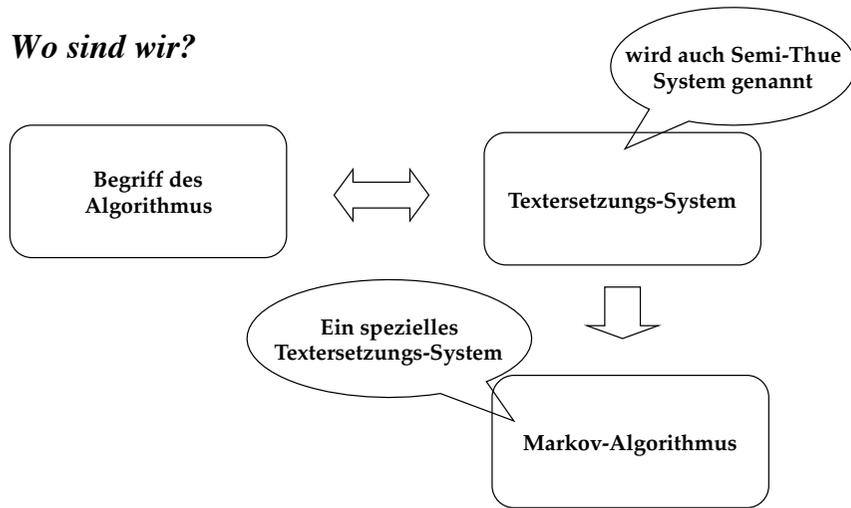
## *Algorithmenarten: Anwendbarkeit von Regeln*

- ❖ **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 auf genau ein Teilwort anwendbar, d.h. es gibt genau eine Berechnung.
  - Falls er terminiert, liefert ein deterministischer Algorithmus genau eine Ausgabe.
- ❖ **Nichtdeterministischer Algorithmus:**
  - Zu einer gegebenen Eingabe  $x$  sind mehrere Regeln bzw. eine Regel auf mehrere Teilwörter von  $x$  anwendbar.

## *Algorithmenarten: Beziehung zwischen Ein- und Ausgabe*

- ❖ **Determinierter Algorithmus:**
  - Zu jeder Eingabe gibt es immer dieselbe Ausgabe. Der Algorithmus kann nichtdeterministisch sein, d.h. es gibt verschiedene Berechnungspfade für dieselbe Eingabe.
    - ◆ Beispiel: Das Textersetzungs-system für die Multiplikation von Strichzahlen
- ❖ **Nichtdeterminierter Algorithmus:**
  - Liefert bei wiederholter Anwendung auf die gleiche Eingabe unterschiedliche Ergebnisse.
    - ◆ Beispiel: Ein Zufallszahlengenerator ist ein nichtdeterminierter Algorithmus.

## Wo sind wir?



## Markov-Algorithmen

❖ **Definition:** Ein Markov-Algorithmus ist ein Textersetzungssystem mit folgenden *zusätzlichen Eigenschaften*:

- Die Menge der Regeln ist linear geordnet: die Reihenfolge der Aufschreibung der Regeln ist wesentlich.
- Markov-Algorithmen enthalten spezielle Regeln der Form  $a \rightarrow \cdot b$ , die durch einen Punkt nach dem Pfeil gekennzeichnet sind. Sie werden *haltende Regeln* genannt.
- Für die Anwendung von Regeln gelten die folgenden beiden *Metaregeln*:
  - ♦ Wähle in jedem Schritt die (von der Aufschreibung her) erste anwendbare Regel. Falls sie auf mehrere Teilwörter anwendbar ist, wende sie auf das am weitesten links stehende Teilwort an.
  - ♦ Wende die Regeln solange an, bis eine haltende Regel angewandt wurde, oder bis keine Regel mehr anwendbar ist.

❖ **Bemerkung:** Durch die erste der beiden Metaregeln sind Markov-Algorithmen deterministisch und damit auch immer determiniert.

## Markov-Algorithmus: Beispiel

Zeichenvorrat  $V = \{0, L, a, b\}$   
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)  $\varepsilon \rightarrow a$

Sonderzeichen wie a und b werden auch *Schiffchen* genannt.

❖ Anwendung der Regeln auf das Eingabewort LOLL

- LOLL  $\Rightarrow$  aLOLL (7)
- $\Rightarrow$  La0LL (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?**

**Antwort: Addition von L!**

## Noch ein Beispiel

Zeichenvorrat  $V = \{0, L, a, b\}$   
Regelsystem:

- (1)  $\varepsilon \rightarrow a$
- (2)  $aL \rightarrow La$
- (3)  $a0 \rightarrow 0a$
- (4)  $a \rightarrow b$
- (5)  $Lb \rightarrow b0$
- (6)  $0b \rightarrow \cdot L$
- (7)  $b \rightarrow \cdot L$

❖ Anwendung der Regeln auf das Eingabewort LOLL

- LOLL  $\Rightarrow$  aLOLL (1)
- $\Rightarrow$  aaLOLL (1)
- $\Rightarrow$  aaaLOLL (1)
- $\Rightarrow$  aaaaLOLL (1)
- $\Rightarrow \dots$

**Frage: Was macht dieser Algorithmus?**

## Vergleich der Algorithmen

Algorithmus 1 terminiert

- (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)  $\varepsilon \rightarrow a$

Algorithmus 2 terminiert nicht

- (1)  $\varepsilon \rightarrow a$
- (2)  $a0 \rightarrow 0a$
- (3)  $a \rightarrow b$
- (4)  $Lb \rightarrow b0$
- (5)  $0b \rightarrow \cdot L$
- (6)  $b \rightarrow \cdot L$
- (7)  $aL \rightarrow La$

Die Reihenfolge der Regeln ist hier wesentlich!

## 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

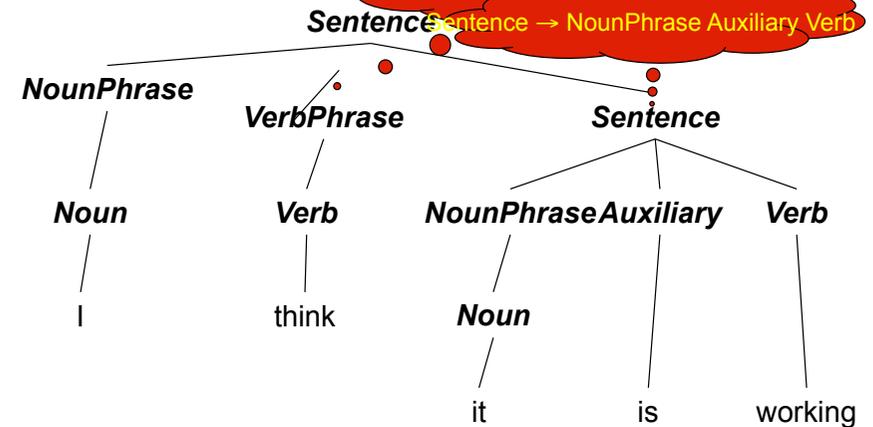
- ❖ Definition **Chomsky-Grammatik**:
  - Eine Grammatik  $G = (T, N, P, Z)$  in der  $T$  ein Zeichenvorrat von Terminalen,  $N$  ein Zeichenvorrat von Nichtterminalen,  $P$  eine endliche Menge von Regeln und  $Z$  ein Zeichen aus  $N$  ist, heißt eine Chomsky-Grammatik.
- ❖ Eine Regel in einer Chomsky-Grammatik heißt **Produktion**.
- ❖  $Z$  ist das **Axiom** oder **Ziel** von  $G$ .
- ❖ Die Vereinigung  $V = T \cup N$  heißt das **Vokabular** der Grammatik  $G$ .
- ❖ Eine Zeichenreihe  $x$  aus  $V^*$ , die durch endlich viele Anwendungen von Regeln aus dem Ziel  $Z$  abgeleitet werden kann, heißt **Satzform** oder **Phrase** von  $G$ .
  - Satzformen können also terminale **und** nichtterminale Zeichen enthalten.
- ❖ Satzformen von  $G$ , die **nur aus terminalen** Zeichen bestehen, heißen **Sätze** von  $G$ .
- ❖  $L(G)$ , die Menge der Sätze von  $G$ , heißt **Sprachschatz** von  $G$ .

## 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**  $T = \{I, is, it, think, working\}$
- ❖ **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  $\Rightarrow \dots \Rightarrow$  **Sentence  $\rightarrow$  NounPhrase VerbPhrase Sentence**



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

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

- |   |   |            |
|---|---|------------|
| <b>Produktionen</b> $P =$   | $A \Rightarrow A+T$                       | -- 2       |
| { Ausdruck $\rightarrow$ Term   | $\Rightarrow T+T$                         | -- 1       |
| Ausdruck $\rightarrow$ Ausdruck + Term  | $\Rightarrow T*F+T$                       | -- 4       |
| Term $\rightarrow$ Faktor   | $\Rightarrow F*F+T$                       | -- 3       |
| Term $\rightarrow$ Term * Faktor  | $\Rightarrow (A)*F+T$                     | -- 6       |
| Faktor $\rightarrow$ Bezeichner   | $\Rightarrow (A+T)*F+T$                   | -- 2       |
| Faktor $\rightarrow$ (Ausdruck)   | $\Rightarrow (T+T)*F+T$                   | -- 1       |
| Bezeichner $\rightarrow a$  | $\Rightarrow (F+T)*F+T$                   | -- 3       |
| Bezeichner $\rightarrow b$  | $\Rightarrow (B+T)*F+T$                   | -- 5       |
| Bezeichner $\rightarrow c$  | $\Rightarrow (B+F)*F+T$                   | -- 3       |
| }   | $\Rightarrow (B+B)*F+T$                   | -- 5       |
| <b>Axiom</b> $Z = \text{Ausdruck}$  | $\Rightarrow (B+B)*B+T$                   | -- 5       |
| <b>Terminale</b> $T = \{a, b, c, (, ), +, *\}$  | $\Rightarrow (B+B)*B+F$                   | -- 3       |
| <b>Nichtterminale</b> $N = \{\text{Ausdruck, Term, Faktor, Bezeichner}\}$                 | $\Rightarrow (B+B)*B+B$                   | -- 5       |
| <b>Vokabular</b> $V = \{a, b, c, (, ), +, *, \text{Ausdruck, Term, Faktor, Bezeichner}\}$ | $\Rightarrow \dots \Rightarrow (a+b)*a+c$ | -- 7,8,7,9 |

## 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 wird sie 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
    }
```

```
 $\langle \text{Sentence} \rangle ::= \langle \text{NounPhrase} \rangle \langle \text{VerbPhrase} \rangle \langle \text{Sentence} \rangle \mid$   
                   $\langle \text{NounPhrase} \rangle \langle \text{Auxiliary} \rangle \langle \text{Verb} \rangle$   
 $\langle \text{NounPhrase} \rangle ::= \langle \text{Noun} \rangle$   
 $\langle \text{VerbPhrase} \rangle ::= \langle \text{Verb} \rangle$   
 $\langle \text{Verb} \rangle ::= \text{think} \mid \text{working}$   
 $\langle \text{Noun} \rangle ::= \text{I} \mid \text{it}$   
 $\langle \text{Auxiliary} \rangle ::= \text{is}$ 
```

## 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
  Bezeichner  $\rightarrow$  b
  Bezeichner  $\rightarrow$  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
  - Ein Stern \* nach einem Zeichen oder einer Gruppe 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.

## Grammatik für Dezimalzahlen in der erweiterten Backus-Naur-Form

Beispiele: 0.34 3.4 -3.4

Erster Vorschlag für eine Grammatik:

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

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

Ist 34 eine Dezimalzahl?

**Iteration 1:** Punkt und Dezimalstellen sind optional

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

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

Ist .34 eine Dezimalzahl?

**Iteration 2:** Vor dem Punkt muss mindestens eine Ziffer stehen

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

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

## Weitere Iterationen ...

Ist 4. eine Dezimalzahl?

**Iteration 3:** Nach dem Punkt muss mindestens auch eine Ziffer stehen

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

$\langle \text{Zahl} \rangle ::= \langle \text{Ziffer} \rangle^+ [ . \langle \text{Ziffer} \rangle^+ ]$

Wie ist es mit -3.4?

**Iteration 4:** Wir erlauben negative Dezimalzahlen

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

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

Sind 0 oder 03.4 oder -03.04 Dezimalzahlen?

**Iteration 5:** Wir verbieten führende Nullen, indem wir zwischen Ziffern und positiven Ziffern unterscheiden:

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

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

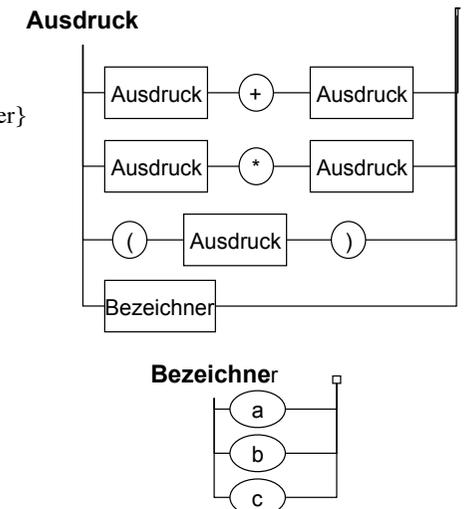
$\langle \text{Zahl} \rangle ::= [-] (0 | \langle \text{Pziffer} \rangle \langle \text{Ziffer} \rangle^* ) [ . \langle \text{Ziffer} \rangle^+ ]$

## 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.
- ❖ Eine Syntaxdiagramm besteht aus Ovalen (Kreisen) und Rechtecken, verbunden durch Kanten.
  - Ovale oder Kreise bezeichnen Terminale.
  - Rechtecke bezeichnen Nichtterminale.

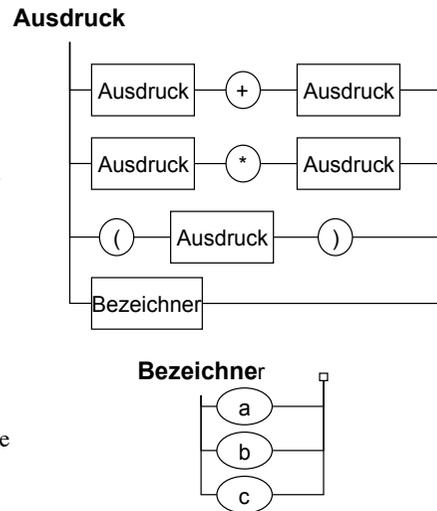
## Beispiel: Konvertierung von Produktionen in ein Syntaxdiagramm

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



## Eine einfachere Grammatik $G'_A$ für arithmetische Ausdrücke

- ❖  $G'_A = (T, N, P, Z)$
- ❖  $T = \{a, b, c, +, *\}$
- ❖  $N = \{\text{Ausdruck}, \text{Bezeichner}\}$
- ❖  $P = \{ \text{Ausdruck} \rightarrow \text{Ausdruck} + \text{Ausdruck}, \text{Ausdruck} \rightarrow \text{Ausdruck} * \text{Ausdruck}, \text{Ausdruck} \rightarrow (\text{Ausdruck}), \text{Bezeichner} \rightarrow a, \text{Bezeichner} \rightarrow b, \text{Bezeichner} \rightarrow c \}$
- ❖  $Z = \text{Ausdruck}$
- ❖ Der **Sprachschatz** ist derselbe.
- ❖ **Strukturinformation** ging verloren (keine Unterscheidung von Ausdruck, Term und Faktor).



## 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
- ❖ Textersetzungssysteme
- ❖ Markov-Algorithmen als Beispiel für deterministische Algorithmen
- ❖ Definitionen: Grammatik
- ❖ Die Backus-Naur-Form und Syntaxdiagramme sind nützlich für die kompakte Beschreibung von Grammatiken