# 7. Syntax: Grammatiken, EBNF

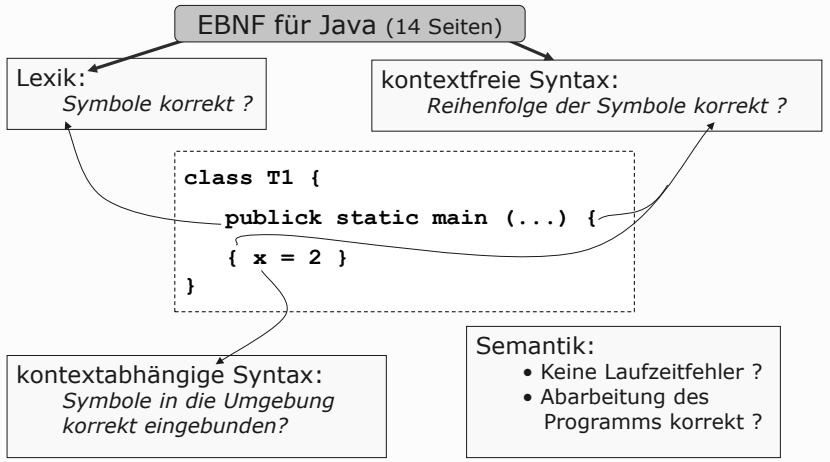
## Teil 2



# Java-Syntax



# Aspekte der Korrektheit von Programmen




https://www2.informatik.hu-berlin.de/swt/lehre/GdP-WS-15/folien/java_syntax.html



# GdP: Web–Seiten

vorige Seite - übergeordnete Seite - nächste Seite - Inhaltsverzeichnis - Index

**Syntax -- kompakt**

***The Unicode Standard: Worldwide Character Encoding*** ::=

ftp://unicode.org .

***Unicode-Zeichen*** ::=

*The Unicode Standard: Worldwide Character Encoding* .

***Unicode-Ziffer*** ::=

*The Unicode Standard: Worldwide Character Encoding* .

***Unicode-Buchstabe*** ::=

*The Unicode Standard: Worldwide Character Encoding* .

***Zeichen*** ::=

\ u { u } *Hexadezimalziffer* *Hexadezimalziffer* *Hexadezimalziffer* *Hexadezimalziffer* | *Unicode-Zeichen* .

***Eingabezeichen*** ::=

< *Zeichen*, **aber nicht** ASCII-CR, ASCII-LF > .

***Zeilenendekennzeichen*** ::=

ASCII-CR ASCII-LF | ASCII-CR | ASCII-LF .

***Quelltext*** ::=

{ *Kommentar* | *Leerzeichen* | *Sprachelement* } .

***Leerzeichen*** ::=

ASCII-SP | ASCII-HT | ASCII-FF | *Zeilenendekennzeichen* .

***Sprachelement*** ::=

*Schlüsselwort* | *Identifikator* | *Literal* | *Separator* | *Operator* .

***Synchronisationsanweisung*** ::=

synchronized ( *Ausdruck* ) *Anweisung* .

***Ausnahmeanweisung*** ::=

try *Block* { catch ( *Ausnahme* ) *Block* } ( finally | catch ( *Ausnahme Identifikator* ) ) *Block* .

***Ausnahme*** ::=

*Klassentyp* .

***Argumentliste*** ::=

*Ausdruck* { , *Ausdruck* } .

***Referenzausdruck*** ::=

*Referenzzugriff* | *Objekterzeugung* .

***Grundausdruck*** ::=

*Referenzausdruck* | *Literal* .

***Referenzzugriff*** ::=

*Bezeichner* | *Feldzugriff* | *Komponentenzugriff* | *Methodenaufruf* | ( *Ausdruck* ) .

***Bezeichner*** ::=

[ *Qualifikator* . ] *Identifikator* | this | super | null .

***Qualifikator*** ::=

{ *Identifikator* . } *Identifikator* .

***Feldzugriff*** ::=

*Referenzausdruck* [ *Ausdruck* ] .

***Komponentenzugriff*** ::=

[ ( *Referenzausdruck* | *Zeichenkette-Literal* ) . ] *Identifikator* .

**Im Detail: kleine Abweichungen zur aktuellen Java-Version**

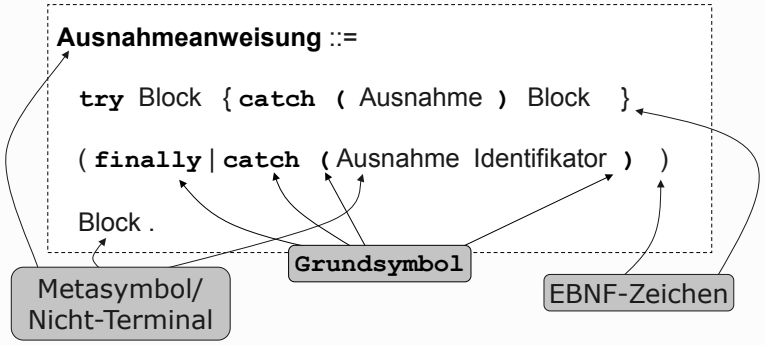
**Sehr gut für Navigation, Prinzip der EBNF**

# Überblick zur Java-Syntax

→ siehe GdP-Web-Seiten (14 Seiten)
https://www2.informatik.hu-berlin.de/swt/lehre/GdP-WS-15/folien/java_syntax.html

in EBNF angegeben

unterschiedliche Schriftsätze regeln Mehrdeutigkeiten !






# Java:
# EBNF-Syntax → Grammatik

$G_{Java} = [ A , M , s , R ]$

| | |
|---|---|
| **Grundsymbole A :** | Sprachelement (ableitbar aus …) |
| **Metasymbole M :** | alle Metasymbole auf linker Seite einer Regel<br>z.B.: Ausnahmeanweisung ::= . . . |
| **Satzsymbol s :** | Quelltextdatei |
| **Regeln R :** | wie angegeben (als EBNF) |



# Regel für das Satzsymbol

Quelltextdatei ::= [Paketfestlegung] {Import} {Typdeklaration} .

Paketfestlegung ::= package Paketname ; .

Import ::= Typimport | Paket-Import .

Typimport ::= import Paketname . Identifikator ; .

Paket-Typimport ::= import Paketname .* ; .

Typdeklaration ::= Klassendeklaration | Schnittstellendeklaration .

Klassendeklaration ::=

[ Abstraktionsebene ] class Identifikator
[ extends Klassentyp ]
[ implements Schnittstellentyp { , Schnittstellentyp } ]
Klassenkörper .

Klassensichtbarkeit ::= public .

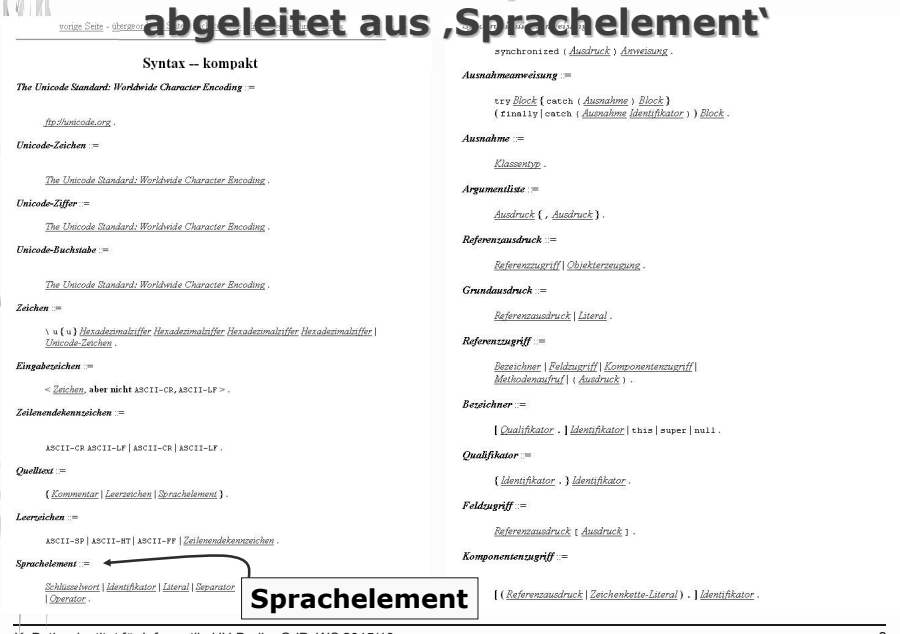
Abstraktionsebene ::= abstract | final .

Mit welchem Symbol kann ein korrektes Java-Programm beginnen?

**u.a.**



# Java-Grundsymbole: abgeleitet aus ‚Sprachelement`



# Java-Grundsymbole (1)

(s. Web-Seite)

Grundsymbole in Java: "Sprachelemente"
→ auch sie haben Syntax (lexikalische Struktur)

Sprachelement ::= Schlüsselwort | Identifikator | Literal | Separator | Operator .

Schlüsselwort ::= **`abstract`** | **`boolean`** | **`break`** | **`byte`** | **`case`** | **`cast`** | **`catch`** | **`char`** | **`class`** | **`const`** | **`continue`** | **`default`** | **`do`** | **`double`** | **`else`** | **`extends`** | **`final`** | **`finally`** | **`float`** | **`for`** | **`future`** | **`generic`** | **`goto`** | **`if`** | **`implements`** | **`import`** | **`inner`** | **`instanceof`** | **`int`** | **`interface`** | **`long`** | **`native`** | **`new`** | **`null`** | **`operator`** | **`outer`** | **`package`** | **`private`** | **`protected`** | **`public`** | **`rest`** | **`return`** | **`short`** | **`static`** | **`super`** | **`switch`** | **`synchronized`** | **`this`** | **`throw`** | **`throws`** | **`transient`** | **`try`** | **`var`** | **`void`** | **`while`** .

Identifikator ::= Unicode-Buchstabe { Unicode-Buchstabe | Unicode-Ziffer | **`_`** | **`$`** } .

z.B.: x1, Matrix, M_1



# Java-Grundsymbole (2)

25 — 1.1 e-2

Literal ::= Zahl-Literal | Gleitkommazahl-Literal | Wahrheitswert-Literal | Zeichen-Literal | Zeichenkette-Literal | Klassen-Literal . **Konstanten**

Operator ::= Inkrementoperator | Dekrementoperator | Vorzeichenoperator | Negationsoperator | Multiplikationsoperator | Additionsoperator | Schiebeoperator | Ordnungsoperator | Vergleichsoperator | **`? ... :`** | **`&&`** | **`||`** | **`&`** | **`|`** | **`^`** | Zuweisungsoperator .

Separator ::= **`(`** | **`)`** | **`{`** | **`}`** | **`[`** | **`]`** | **`;`** | **`,`** | **`.`** .

Inkrementoperator ::= **`++`** .

...

Zuweisungsoperator ::= **`=`** | **`+=`** | **`*=`** | **`-=`** | **`/=`** | **`%=`** | **`&=`** | **`|=`** | **`^=`** | **`<<=`** | **`>>=`** .



# Kann es auch ein andere Grammatik für Java geben?



# Äquivalenz von Grammatiken

Zwei Grammatiken $G_1$ und $G_2$ sind **äquivalent**, falls

$$L_{G1} = L_{G2}$$

(die erzeugten Sprachen sind identisch)

→ "**die**" Java-Grammatik ?

Unterschiedliche Varianten für Java-Grammatiken:
- Metasymbole umbenennen (z.B. deutsch → englisch)
- Statt Metasymbol in die Regel direkt die Alternativen eintragen (z.B. Inkrementoperator, Separator …)
- u.v.a.

**Zu einem Programm wird es im allg. immer mehrere Ableitungen geben:**

**Ist das ein Problem – z.B. für einen Compiler?**

| | |
|---|---|
| expr → term exprrest<br>→ term ε<br>→ factor termrest<br>→ factor<br>→ a | expr → term exprrest<br>→ factor termrest exprrest<br>→ a termrest exprrest<br>→ a exprrest<br>→ a |



# Syntaxbäume:

## alternative Repräsentation der syntaktischen Struktur

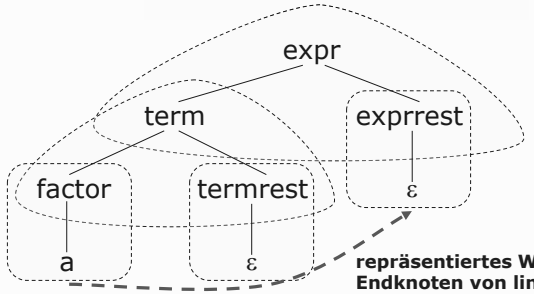
Repräsentation der syntaktischen Syntax:

- Ableitungen mit Regeln
- Syntaxbäume

expr → term exprrest
→ term ε
→ factor termrest
→ factor
→ a



# Syntaxbäume



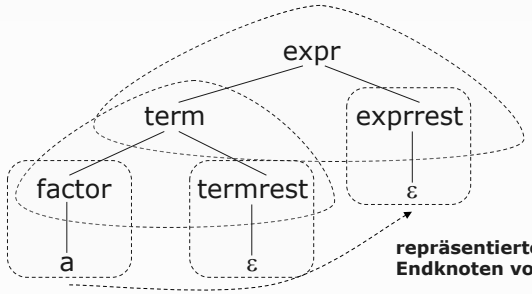

**repräsentiertes Wort:**
**Endknoten von links nach rechts**

entspricht in $G_1$ den Ableitungen:

| | | |
|---|---|---|
| expr → term exprrest<br>→ term ε<br>→ factor termrest<br>→ factor<br>→ a | und | expr → term exprrest<br>→ factor termrest exprrest<br>→ a termrest exprrest<br>→ a exprrest<br>→ a |
| **Rechtsableitung** | | **Linksableitung** |



# Syntaxbaum: Regelanwendungen entwickeln Baum von oben nach unten (1)

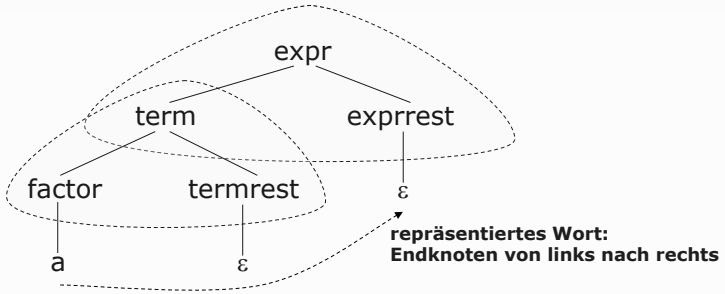



**repräsentiertes Wort:**
**Endknoten von links nach rechts**

entspricht in $G_1$ der Ableitung:

expr → term exprrest
→ term ε
→ factor termrest
→ factor ε
→ a

# Syntaxbaum: Regelanwendungen entwickeln Baum von oben nach unten (2)




entspricht in $G_1$ der Ableitung:

**Identischer Baum**
**- nur in anderer Reihenfolge aufgebaut.**

expr → term exprrest
→ factor termrest exprrest
→ a termrest exprrest
→ a exprrest
→ a



# Syntaxbäume: Aufgabe

- Repräsentieren syntaktische Struktur von Programmen
  (lesbarer und aussagefähiger als Ableitungen)
- Abstraktion von konkret gewählter Ableitungsfolge
- Eindeutigkeit von Grammatiken definiert
- Sind Ergebnis der Syntaxanalyse von Compilern
  (interne Repräsentation von Programmen zur weiteren Analyse und Optimierung)



# Eindeutigkeit von Grammatiken

Def.:

Eine Grammatik G heißt **eindeutig**:

Für jedes Wort x aus $L_G$ führen alle Ableitungen von x zum selben Syntaxbaum.

m. a. W.: Unterschiedliche Ableitungen desselben Wortes dürfen keine unterschiedliche syntaktische Struktur des Wortes definieren!

Beispiel:
$G_1$ ist eindeutig (ohne Beweis),
aber ein Beispiel „a“ gegeben:
mehrere Ableitungen von a → derselbe Syntaxbaum



# Eine zu $G_1$ äquivalente Grammatik G'

G' mit den Regeln:

E = E + E |
E * E |
a |
( E ) .

- Erzeugte Sprachen gleich: a, a+a, a*a, (a+a)*a …
- G' kürzer (lesbarer / übersichtlicher).

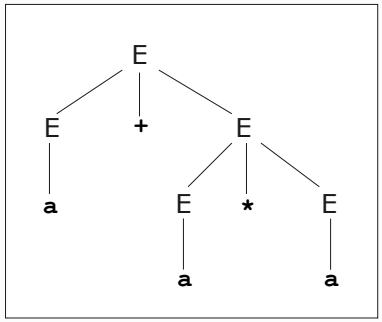
G' bevorzugen anstelle von G1?

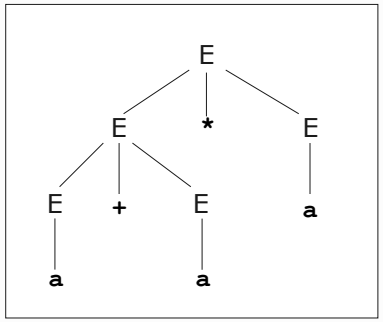
**a + a * a** ist Element der Sprache

... und damit G' nicht eindeutig!

# G' nicht eindeutig:

**a + a * a**
hat zwei Ableitungen mit unterschiedlichen Syntaxbäumen



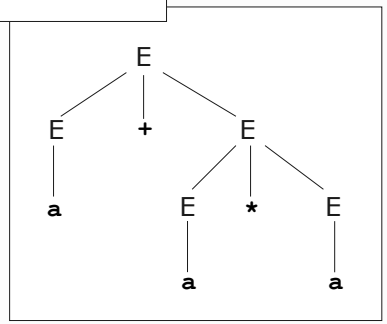



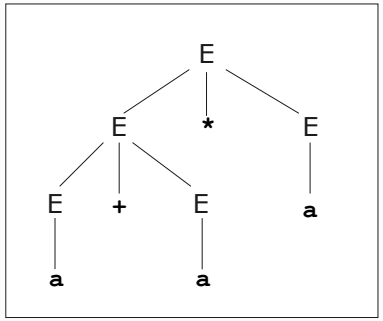



# G' nicht eindeutig:

E = E + E |
E * E |
a |
( E ) .

**a + a * a**
hat zwei Ableitungen mit unterschiedlichen Syntaxbäumen



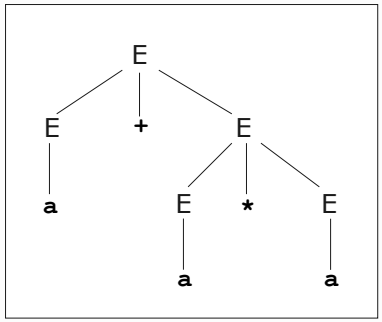

E → E + E → E + E * E → a + E * E → a + a * E → a + a * a

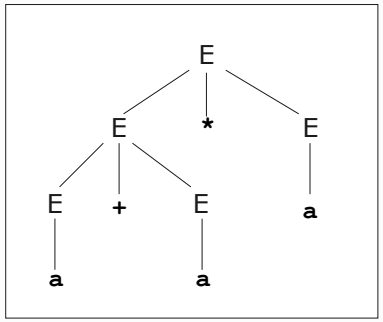
E → E * E → E + E * E → a + E * E → a + a * E → a + a * a



# G' nicht eindeutig:

**a + a * a**
hat zwei Ableitungen mit unterschiedlichen Syntaxbäumen




Ist Nicht-Eindeutigkeit ein Problem?

**JA: Semantik ist unterschiedlich (erst „+" oder erst „*")**



# Grammatik von Java ist eindeutig

… bis auf eine Ausnahme

# Grammatiken zu Sprachen finden



# Bisherige Aufgabe

***Gegebene Grammatik definiert Sprache***

**Grammatik G → erzeugte Sprache $L_G$**

Anwendung: EBNF für Java

→ alle syntaktisch korrekten Java-Programme



# Umgekehrte Aufgabe

Anwendung: InformatikerIn möchte eine (Fach-)Sprache L einführen:

1. Schritt: kontextfreie Grammatik für L definieren

Für eine vorgegebene Sprache wird eine kf. Grammatik gesucht.

*Problem:*
**Existiert überhaupt eine kontextfreie Grammatik ?**



# Formale Sprachen

(Formale) **Sprache** L über einem Alphabet A $=_{def}$ $L \subseteq A^*$

(Definition unabhängig vom Grammatikbegriff!)

→ Programme sind Folgen von Grundsymbolen

L heißt **kontextfreie Sprache** $=_{def}$

Formale Sprache L lässt sich durch eine kontextfreie Grammatik G beschreiben (d.h. $L = L_G$)

# Beispiele: formale Sprachen

formale Sprachen über $A_2 = \{a, b, c\}$

- $L_1 = \{a, b, c\} \subseteq A_2^*$
- $L_2 = \{a^n \mid n > 0\}$ = {a, aa, aaa, aaaa, ...}
- $L_3 = \{a^n b^n \mid n \geq 0\}$ = {ε, ab, aabb, aaabbb, ...}
- $L_4 = \{a^n b^n c^n \mid n > 0\}$ = {abc, aabbcc, aaabbbccc, ...}

Sind $L_1 ... L_4$ auch kontextfreie Sprachen?

- $G_1$ = [ {a, b, c}, {s}, s, ... ]
- $G_2$ = [ {a}, {s}, s, ... ]
- $G_3$ = [ {a, b}, {s}, s, ... ]
- $G_4$ = [ {a, b, c}, {s}, s, ... ]



# Beispiele: kontextfreie Grammatiken für die Sprachen

formale Sprachen über $A_2 = \{a, b, c\}$

- $L_1 = \{a, b, c\}$
- $L_2 = \{a^n \mid n > 0\}$ = {a, aa, aaa, aaaa, ...}
- $L_3 = \{a^n b^n \mid n \geq 0\}$ = {ε, ab, aabb, aaabbb, ...}
- $L_4 = \{a^n b^n c^n \mid n > 0\}$ = {abc, aabbcc, aaabbbccc, ...}

Sind $L_1 ... L_4$ auch kontextfreie Sprachen?

- $G_1$ = [ {a, b, c}, {s}, s, {(s, a), (s, b), (s, c)}]
- $G_2$ = [ {a}, {s}, s, {(s, a), (s, as)}]
- $G_3$ = [ {a, b}, {s}, s, {(s, ), (s, asb)}]
- $G_4$ : es ex. keine kontextfreie Grammatik mit $L_{G4} = L_4$



# Historie:

Untersuchung zum Aufbau *natürlicher* Sprachen
→ erster (formaler) Grammatik-Begriff (Chomsky 1959)

Satzsymbol

| | | |
|---|---|---|
| SATZ | : | SUBJEKT PRAEDIKAT OBJEKT . |
| SUBJEKT | : | ARTIKEL SUBSTANTIV . |
| OBJEKT | : | ARTIKEL SUBSTANTIV . |
| PRAEDIKAT | : | VERB . |
| SUBSTANTIV | : | Hund \| Brieftraeger . |
| VERB | : | beisst . |
| ARTIKEL | : | den \| der . |

Metasymbole = Fachbegriffe der Grammatik

Regeln der Grammatik

Grundsymbole = Worte der Sprache

SATZ → SUBJEKT PRAEDIKAT OBJEKT → ...
... → der Hund beisst den Brieftraeger

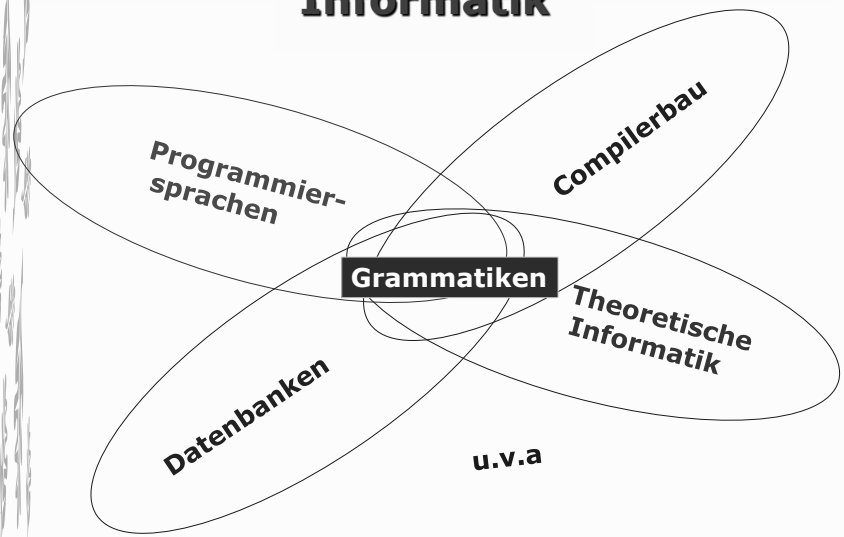
aber auch:
SATZ →* den Brieftraeger beisst der Hund
SATZ →* den Hund beisst der Brieftraeger

Problem: Reichhaltigkeit der Grammatik einer natürlichen Sprache (Konjugation, Deklination u. v. a.)

→ erweiterte Grammatikbegriffe nötig



# Grammatiken in Disziplinen der Informatik



# Nachfolgend

# Zusatzmaterial: am 2. 11.2015 in der Vorlesung behandelt (zur Information)



# Allgemeiner Grammatik-Begriff



# Beispiele: kontextfreie Grammatiken für die Sprachen

formale Sprachen über $A_2 = \{a, b, c\}$

- $L_1 = \{a, b, c\}$
- $L_2 = \{a^n \mid n > 0\}$ = {a, aa, aaa, aaaa, ...}
- $L_3 = \{a^n b^n \mid n \geq 0\}$ = {ε, ab, aabb, aaabbb, ...}
- $L_4 = \{a^n b^n c^n \mid n > 0\}$ = {abc, aabbcc, aaabbbccc, ...}

Sind $L_1 ... L_4$ auch kontextfreie Sprachen?

- $G_1 = [\{a, b, c\}, \{s\}, s, \{(s, a), (s, b), (s, c)\}]$
- $G_2 = [\{a\}, \{s\}, s, \{(s, a), (s, as)\}]$
- $G_3 = [\{a, b\}, \{s\}, s, \{(s, ), (s, asb)\}]$
- $G_4$ : es ex. keine kontextfreie Grammatik mit $L_{G4} = L_4$

ABER: Es gibt allgemeine Grammatik für $L_4$



# (Allgemeine) Grammatik

G = [A, M, s, R] heißt **Grammatik**, falls

- A, M, s wie bisher (bei kf. Grammatiken)
- $R \subseteq (A \cup M)^+ \times (A \cup M)^*$ Menge von Regeln (l,r), wobei l mindestens ein Metasymbol enthält

**$(A \cup M)^+$ beliebige nicht-leere Symbolfolgen**

**Kontextfreie Grammatik:**
**Kontextfreie Regeln**

$X_2 = \mathbf{b}\ \mathbf{c}\ .$
$X_1 = X_1\ \mathbf{c}\ .$

Metasymbole können beliebig ersetzt werden

**Allgemeine Grammatik:**
**Kontextabhängige Regeln**

$X_2 X_1 = \mathbf{b}\ X_2\ \mathbf{c}\ .$
$\mathbf{c}\ X_1 = X_1\ \mathbf{c}\ .$

Metasymbole können nur im Kontext ersetzt werden

# (Allgemeine) Grammatik

G = [A, M, s, R] heißt **Grammatik**, falls

- A, M, s wie bisher (bei kf. Grammatiken)
- $R \subseteq (A \cup M)^+ \times (A \cup M)^*$ Menge von Regeln (l,r), wobei l mindestens ein Metasymbol enthält

**$(A \cup M)^+$ beliebige nicht-leere Symbolfolgen**

Kontextfreie Grammatik: Spezialfall einer allgemeinen Grammatik

Allg. Grammatiken leistungsfähiger als kf. Grammatiken
→auch kontextabhängige Syntax beschreibbar!
*Also auch: benutzte Variablen müssen vorher vereinbart sein*

aber: allg. Regeln schwer zu lesen



# Praxis: Beschreibung von Programmiersprachen

- Kontextfreie Syntax:
  kontextfreie Grammatik (EBNF)
- Kontextabhängige Syntax:
  verbal (umgangssprachlich)
  z. B.: "*Jede benutzte Variable muss vereinbart werden.*"



# Beispiel für allgemeine Grammatik

$L_4 = \{a^n b^n c^n \mid n > 0\}$

Mögliche Regeln einer (allg.) Grammatik, die $L_4$ erzeugt:

**kontextfrei: Ersetzung eines Metasymbols unabh. von Umgebung**

| | |
|---|---|
| $s = \mathbf{a}\ s\ X_1\ .$ | (r1) |
| $s = \mathbf{a}\ X_2\ .$ | (r2) |
| $X_2 X_1 = \mathbf{b}\ X_2\ \mathbf{c}\ .$ | (r3) |
| $\mathbf{c}\ X_1 = X_1\ \mathbf{c}\ .$ | (r4) |
| $X_2 = \mathbf{b}\ \mathbf{c}\ .$ | (r5) |

r1, r2, r5: kontextfreie Regeln
r3, r4: kontextabhängige Regeln

Beispielableitung:

| | |
|---|---|
| $s \rightarrow asX_1$ | (r1) |
| $\rightarrow aaX_2X_1$ | (r2) |
| $\rightarrow aabX_2c$ | (r3) |
| $\rightarrow aabbcc$ | (r5) |