

Durch die einanderreihung solcher Zeichen wird ein Satz (hier = Java Programm) gebildet. Syntaxregeln legen fest, wann das syntaktisch korrekt ist.

verschiedene Darstellungsmöglichkeiten

- Backus Naur Form (bzw. erweiterte BNF)
- Syntaxgraph (Syntaxdiagramme)

Beide bieten Darstellungsmöglichkeiten für

- Option
- Alternative
- (begrenzte) Wiederholung
- einanderreihung (Kombination)

4.1 Backus Naur Form

Man verwendet Metazeichen für die Beschreibung der Konstruktionsmöglichkeiten

[...] für Option

| ... | für Alternative

{ ... } für Wiederholung

Kombination durch einanderreihung

Symbole der Sprache heißen Terminalsymbole
 Daneben kann man noch nicht erklärte Begriffe
 verwenden, die Nonterminalsymbole. Gemeinsamzeit
 durch $\langle \dots \rangle$. Diese werden durch $::=$ definiert

Beispiel: Identifier in Java

$$\langle \text{identifier} \rangle ::= \langle \text{Unicode-Buchstabe} \rangle \{ \langle \text{Unicode-Buchstabe} \rangle | \langle \text{Unicode-Ziffer} \rangle \}$$

$$\langle \text{Unicode-Buchstabe} \rangle ::= A | B | C | \dots | a | b | \dots | \alpha | \beta | \dots | _$$

$\uparrow \quad \quad \uparrow \uparrow$
 Terminalsymbole

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

Nach diesen Regeln korrekte Identifier sind:

a_1 $a_very_long_identifier$ $\alpha\beta\gamma$

nicht korrekt

$1a$ $x-11$ $a*$

Neben der Syntax gibt es die Semantik

\uparrow
 syntaktische Regeln

\downarrow
 legt Bedeutung der
 korrekt geformten Ausdrücke

fest, meist durch zusätzliche Regeln außerhalb des Syntax

Bsp bei Identifier:

- Groß- und Kleinschreibung wird an jeder Position unterschieden

abba, ABBa, aBBa, abBa ...

sind paarweise verschiedene Identifier

- Schlüsselwörter können keine Identifier sein

Beispiel 4.2 if-statement in Java

$\langle \text{if-statement} \rangle ::= \text{if} (\langle \text{Bedingung} \rangle) \langle \text{Anweisung} \rangle$
[else $\langle \text{Anweisung} \rangle$]

$\langle \text{Bedingung} \rangle ::=$

$\langle \text{Anweisung} \rangle ::=$ siehe Java Spezifikation

Ein ausführliches Beispiel: Integer-Literal

↑
Zeichenkette die Zahl vom
Typ int oder long beschreibt

$\langle \text{integer literal} \rangle ::= \langle \text{Zahlenwert} \rangle [\langle \text{Typsuffix} \rangle]$

$\langle \text{Zahlenwert} \rangle ::= \langle \text{Dezimalwert} \rangle / \langle \text{Oktalwert} \rangle / \langle \text{Hexadezimalwert} \rangle$

$\langle \text{Typsuffix} \rangle ::= \ell \mid L$

$\langle \text{Dezimalwert} \rangle ::= 0 \mid \langle \text{erste Dezimalziffer} \rangle \{ \langle \text{Dezimalziffer} \rangle \}$

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

$\langle \text{Oktalwert} \rangle ::= \langle \text{Oktalpräfix} \rangle \langle \text{Oktalziffer} \rangle \{ \langle \text{Oktalziffer} \rangle \}$

$\langle \text{Oktalpräfix} \rangle ::= 0$

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

$\langle \text{Hexadezimalwert} \rangle ::= \langle \text{Hexadezimalpräfix} \rangle \langle \text{Hexadezimalziffer} \rangle \{ \langle \text{Hexadezimalziffer} \rangle \}$

$\langle \text{Hexadezimalpräfix} \rangle ::= 0x \mid 0X$

$\langle \text{Hexadezimalziffer} \rangle ::= 0 \mid 1 \mid 2 \mid \dots \mid 9 \mid a \mid A \mid b \mid B \mid c \mid C \mid d \mid D \mid e \mid E \mid f \mid F$

↑

syntaktische Regeln Bsp. 0x Da Da Cafe

zur Semantik regelt Bedeutung ↗

- Präfix 0 bedeutet Oktaldarstellung

d.h. zur Basis 8 mit Ziffern 0 1 2 ... 7

Literal 045 $\hat{=}$ Zahlenwert $4 \cdot 8^1 + 5 \cdot 8^0 = \underline{\underline{37}}$
dezimal

- Präfix 0x oder 0X bedeutet Hexadezimaldarstellung

d.h. zur Basis 16 mit Ziffern 0 ... 9 a A b B ... f F

Zahlenwerte der Ziffern: $\begin{array}{cccccc} & 0 & 1 & \dots & 9 & a & A & b & B & \dots & f & F \\ & | & | & & | & | & | & | & | & & | & | \\ & 0 & 1 & & 9 & 10 & 11 & 12 & 13 & & 15 & 16 \end{array}$

Literal 0x Da Da Cafe $\hat{=}$ $13 \cdot 16^7 + 10 \cdot 16^6 + 13 \cdot 16^5 + 10 \cdot 16^4$

$$16^7 \quad 16^0$$

$$+ 12 \cdot 16^3 + 10 \cdot 16^2 + 15 \cdot 16^1 + 14 \cdot 16^0$$

- Typsuffix l oder L bezieht sich auf den Java Typ long zur Darstellung ganzer Zahlen

standard für int Zahlen: 32 bit zur Darstellung
im sog. Zweierkomplement

$$\text{Werte von } 2^{-31} \text{ bis } 2^{31} - 1$$

standard für long Zahlen 64 bit im Zweierkomplement

$$\text{Werte von } 2^{-63} \text{ bis } 2^{63} - 1$$

- Liegt Zahlenwert eines Literals außerhalb des durch 32 bit bzw 64 bit festgelegten Bereichs, so soll beim Compilieren (laut Spezifikation) eine Fehlermeldung kommen

↑ ACHTUNG, geschieht in der Regel nicht

Beispiele für integer Literale

13	039	0	f17	000	0xfL	0-3
✓	✗	✓	✗	✓	✓	✗

Beachte: Ver gehört nicht zum Integer-Literal
- ist einstelliger Operator, der auf ein Literal angewendet wird

4.2 erweiterte BNF

$\{ \dots \}_l^k$ mindestens l mal, höchstens k mal wiederholen

$\{ \dots \}_l^*$ mindestens l mal, sonst beliebig oft

Beispiele

$\langle \text{integer literal} \rangle ::= \langle \text{Zahlenwert} \rangle \{ \langle \text{Typsuffix} \rangle \}_0^1$

$\langle \text{Oktalwert} \rangle ::= \langle \text{Oktalpräfix} \rangle \{ \langle \text{Oktalziffer} \rangle \}_1^*$

4.3 Syntaxgraphen



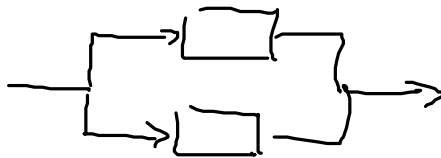
Nonterminalsymbol



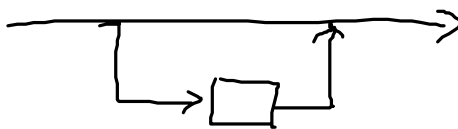
Terminalsymbol



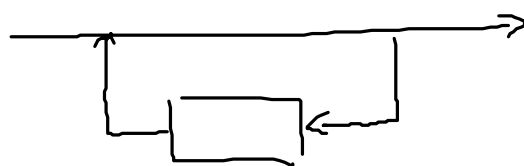
Konkatenation



Alternative

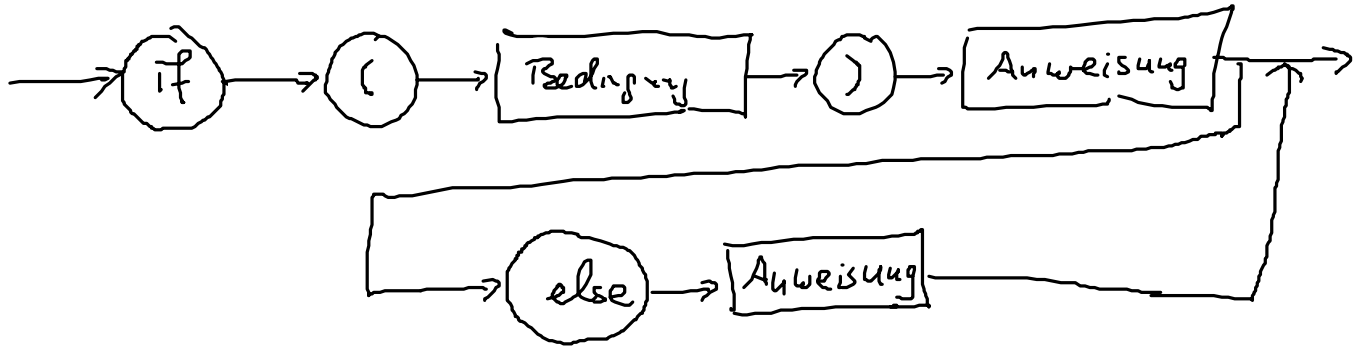


Optional



Wiederholung

Beispiel: if statement



Jeder Weg vom Anfang bis Ende in Pfeilrichtung
 führt ein korrektes Wort der Grammatik

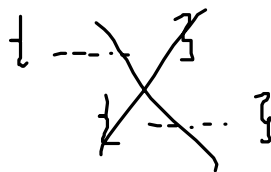
Ein Anwendung außerhalb von Java

Korrekte Klammerausdrücke

{ } { } { } { } { } { } ← Korrekt oder nicht?

↑
 in jeder Klammer { gehört
 eine Klammer }
 das dazwischen heißt Block

Blöcke dürfen sich nicht echt überlappen



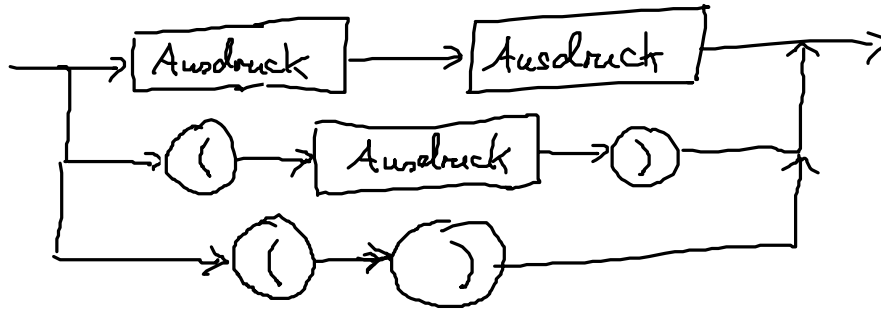
Suchen: Grammatik (= Syntaxregeln) die genau die
 korrekt geschriebenen Klammerausdrücke beschreibt

- kleinstes korrektes (\neq leer) $()$
- wie kann man aus korrekten Klammerausdrücken

größere machen?

- A korrekt \Rightarrow (A) korrekt
- A, B korrekt \Rightarrow A B korrekt
- (), (()) \Rightarrow () (())

festhalten als Syntaxregeln



Frage: beschreibt diese Grammatik alle korrekten Klammerausdrücke?