EBNF
Extended Backus Naur Form

Eine Metasyntax

Grammatik-Notation

- Informatik benutzt formale Sprachen
- Eine formale Sprache besteht aus Worten über einem endlichen Alphabet von Zeichen.
- In der Informatik nur Sprachen interessant, die durch (einfach zu handhabende) Grammatiken erzeugt werden können.
- EBNF ist eine einfache und doch „mächtige“ Grammatik

  John Backus 1924, Fortran & FP, Peter Naur 1928

Definition EBNF

- N = endliche Menge von ***Nonterminals***
  Nonterminals sind Grammatikbegriffe
- T = endliche Menge ***Terminalsymbole*** (Zeichen)
  Die durch EBNF definierte Sprache besteht aus aneinandergereihten Terminalsymbolen (Worte aus den Terminalzeichen)
- P = endliche Menge von ***Produktionen*** der Form
  A ::= EBNF-Ausdruck
  A∈N ist Nonterminal
  EBNF-Ausdruck besteht aus Nonterminals, Terminals und Meta-Zeichen **{}[]()|**
- Ein ***Startsymbol*** S∈N

Die definierte Sprache besteht aus allen Worten aus Zeichen aus T, die mit Regeln aus P aus dem Startsymbol hergeleitet werdeb können.

EBNF-Beispiel

Zahl-Literale in Java durch EBNF definieren

- **Menge der Nonterminals**
  N = {IntegerLiteral, OctalLiteral, DeciLiteral, HexLiteral,Digit, Digit1, OctDigit, HexDigit}

IntegerLiteral

- **Menge der Terminalsymbole**
  T={‘0‘,‘1‘,‘2,‘3,‘4‘,‘5‘,‘6‘,‘7‘,‘8‘,‘9‘,‘a‘,‘b‘,‘c,‘d‘,‘e‘,‘f‘,‘ l,‘‘x‘,‘A‘,‘B‘,‘C‘,‘D‘,‘E‘,‘F‘,‘L‘,‘X‘,‘.‘,‘+‘,‘-‘}

- **Startsymbol: IntegerLiteral**

EBNF-Regeln für Integer-Literale in Java

$$IntegerLiteral ::= ['+'|'-'](OctalLiteral | DeciLiteral | HexLiteral)['l'|'L']$$

$$OctalLiteral ::= '0' OctDigit\{OctDigit\}$$

$$OctDigit ::= '0'|...|'7'$$

$$DeciLiteral ::= Digit1\{Digit\}|'0'$$

$$Digit1 ::= '1'|...|'9'$$

$$Digit ::= '0'|Digit1$$

$$HexLiteral ::= ('0x'|'0X')HexDigit\{HexDigit\}$$

$$HexDigit ::= Digit|'a'|'A'|...|'f'|'F'$$

Metasymbol der EBNF

- **Aneinanderreihung**
  A ::= BCD
  A wird durch die 3 aufeinaderfolgenden Symbole ersetzt
- **| Alternative**
  A ::= B | C | D
  A kann durch B oder C oder D ersetzt werden
- **[] Option**
  A ::= [B]C
  A kann durch BC oder C ersetzt werden.
- **{} Wiederholung**
  A ::= {B}C
  A kann durch C, BC, BBC, B....BC ersetzt werden.

- Ableiten der Dezimalzahl 10

IntegerLiteral ::= DeciLiteral

DeciLiteral ::= Digit1{Digit}
‘1‘Digit
‘1‘‘0‘

- Ableiten der Oktalzahl 077 = 63

IntegerLiteral ::= OctLiteral

OctLiteral ::=‘0‘ OctDigit{OctDigit}
‘0‘ ‘7‘{OctDigit}
‘0‘ ‘7‘OctDigit
‘0‘ ‘7‘ ‘7‘