

Kapitel 2

Methoden zur Beschreibung von Syntax

***„Grammatik,
die sogar Könige zu kontrollieren weiß ...“***

– aus Molière, Les Femmes Savantes (1672), 2. Akt

Ziele

Zwei Standards zur Definition der Syntax von Programmiersprachen kennenlernen:

- Backus-Naur-Form (BNF)
sowie deren Erweiterung EBNF
- Syntaxdiagramme

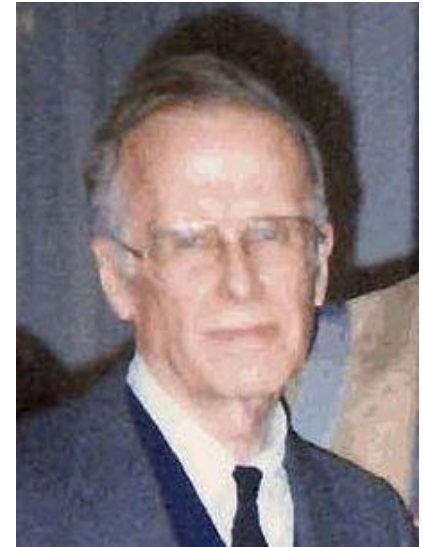


Peter Naur

*1928

Mitwirkung bei ALGOL 60

Turingpreis 2005



John Backus

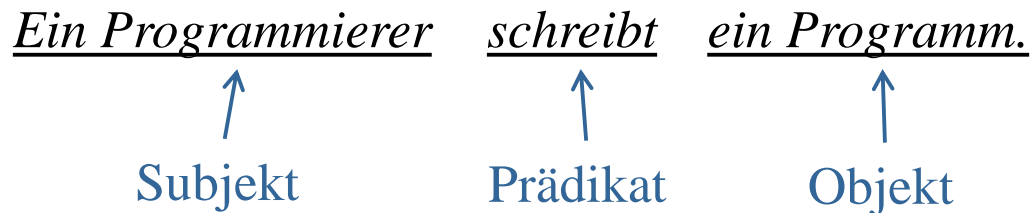
1924-2007

Entwicklung von FORTRAN

Turingpreis 1977

Syntax und Programme

- Java-Programme sind Texte, die sich wie natürlichsprachliche Texte in verschiedenartige Einzelteile zerlegen lassen.
- Natürliche Sprache:



- Java-Programm:

```
public class Test {  
    public static void main(String[] args) {  
        System.out.print("Hal");  
        System.out.println("lo!");  
    }  
}
```

Methodenname String-Literal

Anweisungen
(Methodenaufrufe)

Methoden-
deklaration

BNF und Syntaxdiagramme: Grundideen

- Beispiel für eine Grammatikregel:

Als Syntaxdiagramm:

In BNF-Form: $\langle \text{Satz} \rangle ::= \langle \text{Subjekt} \rangle \langle \text{Prädikat} \rangle \langle \text{Objekt} \rangle$

In EBNF-Form: $\text{Satz} = \text{Subjekt Prädikat Objekt}$

- Das Syntaxdiagramm und die BNF (*Backus-Naur-Form*) erlauben das Bilden folgender syntaktisch korrekter Sätze:

. *Ein Programmierer schreibt ein Programm*

. *Ein Programmierer* Subjekt

. *schreibt* Prädikat

. *ein Programm* Objekt

. *Ein Programm schreibt einen Programmierer*

Dieser Satz ist zwar semantischer Unsinn, aber syntaktisch korrekt.

- Der Compiler benützt die Grammatik um ein gegebenes Programm auf syntaktische Richtigkeit zu überprüfen

Backus-Naur-Form

- Die **Backus-Naur-Form (BNF)** wurde erstmals zur Beschreibung der Syntax von Algol 60 verwendet.
- Die BNF ist eine Notation für Grammatiken, die vor allem für die Beschreibung von Programmiersprachen verwendet wird.
- Heute ist die BNF (in notationellen Varianten) die Standardbeschreibungstechnik für Programmiersprachen und andere strukturierte Texte.
- Wir verwenden in der Vorlesung die „Erweiterte Backus-Naur-Form“ EBNF (eingeführt zur Beschreibung von PL1, 60er Jahre).
- Auch die Syntax von Java ist in einer Variante der Backus-Naur-Form beschrieben.
- Wichtige Begriffe: Symbole, Regeln, Grammatik, Ableitung

Symbole

- **Nichtterminalsymbole:**

Begriffe, die durch Regeln definiert werden.

Beispiele

in BNF: *<Digit>*, *<Sign>*

in EBNF: *Digit*, *Sign*

- **Terminalsymbole:**

Zeichen oder Folgen von Zeichen, die genau so in der zu definierenden Sprache vorkommen.

Beispiele

in BNF: *0*, *1*, *class*

in EBNF: *"0"*, *"1"*, *"class"*

- **Operatorsymbole:**

in BNF: *|* für Auswahl

in EBNF: *|* sowie *[]* für Optionen und *{ }* für Wiederholung

Regeln

Jede **EBNF-Regel** hat die Form

$$\textit{Nichtterminalsymbol} = \textit{Ausdruck}$$

Ein Ausdruck ist entweder

- ein Terminalsymbol oder
- ein Nichtterminalsymbol oder
- ein zusammengesetzter Ausdruck.

Aus gegebenen Ausdrücken E , $E1$ und $E2$ können zusammengesetzte Ausdrücke durch Anwendung der Operatorsymbole gebildet werden:

Sequentielle Komposition $E1 E2$ („ $E2$ folgt direkt auf $E1$ “)

Auswahl $E1 | E2$ („ $E1$ oder $E2$ “)

Option $[E]$ („ E kann weggelassen werden“)

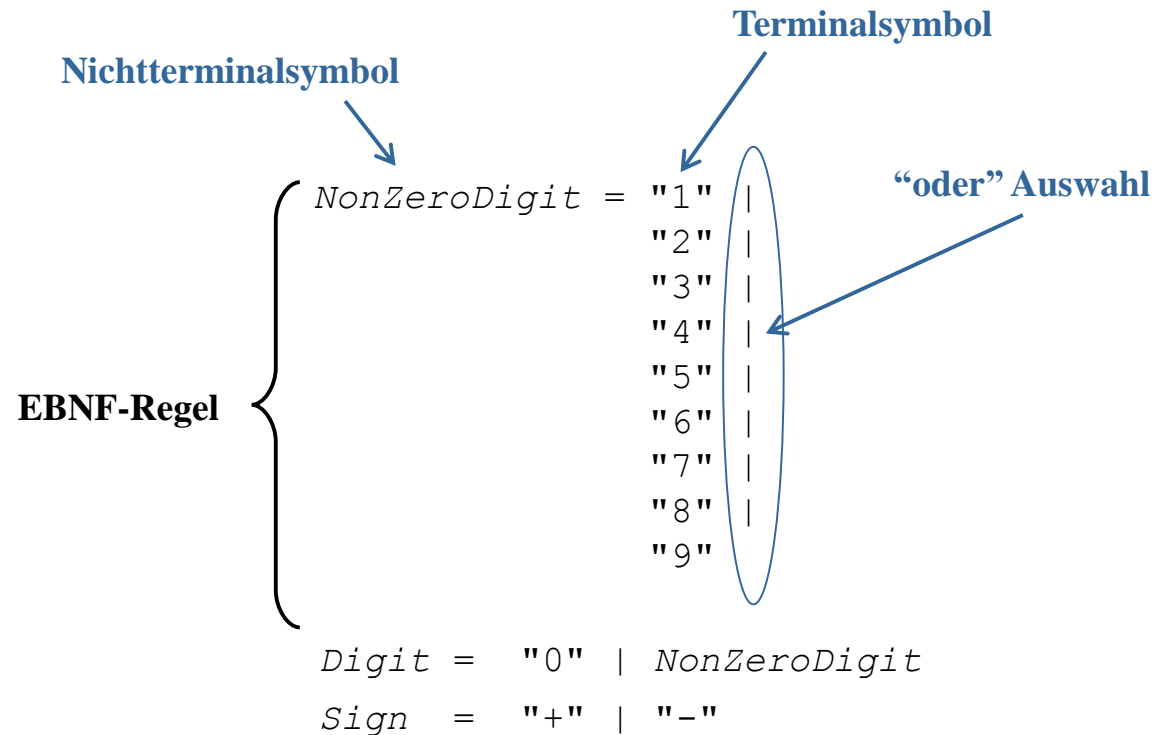
Wiederholung $\{E\}$ („ E kann 0-mal oder mehrmals hintereinander vorkommen“)

Grammatik

- Eine **Grammatik** besteht aus
 - einer Menge von **Regeln** für jedes Nichtterminal sowie
 - einem Startsymbol (Nichtterminalzeichen)
- Jede Grammatik G definiert eine **Menge von Wörtern**, die als **Sprache von G** bezeichnet wird.
 - Ein **Wort** ist eine Folge von Terminalzeichen.
 - Wir schreiben $L(G)$ für die Sprache von G . (L für Language)
 - Die Sprache $L(G)$ besteht genau aus den Wörtern, die vom Startsymbol der Grammatik **abgeleitet** werden können.

Beispiele für EBNF-Grammatiken

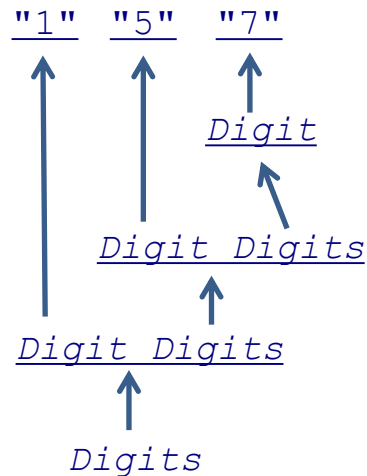
- Grammatik für Ziffern und Vorzeichen



Beispiele für EBNF-Grammatiken

- Grammatik für ganze Zahlen (*Integers*)
- Informelle Beschreibung: Eine ganze Zahl besteht aus einer nichtleeren Folge von Ziffern ohne führende „0“, evtl. mit einem vorangestellten Vorzeichen.

z.B.



Eine nichtleere Ziffernfolge ist eine Ziffer evtl. gefolgt von einer nichtleeren Ziffernfolge.

EBNF-Regel:

$Digits = Digit [Digits]$

oder mit Wiederholung: $Digits = Digit \{Digit\}$

Eine ganze Zahl beginnt mit einem optionalen Vorzeichen, gefolgt von einer nichtleeren Ziffernfolge ohne führende Null.

EBNF-Regeln:

$DecimalNumeral = "0" | NonZeroDigit [Digits]$

$IntegerValue = [Sign] DecimalNumeral$

Wie wendet man die Regeln an?

Ist **+31** eine *GanzeZahl*?

Wir bilden folgende Ableitung:

<i>IntegerValue</i>	→ (Regel für <i>IntegerValue</i>)
[<i>Sign</i>] <i>DecimalNumeral</i>	→ (Ausführen des Operators [])
<i>Sign</i> <i>DecimalNumeral</i>	→ (Regel für <i>Sign</i>)
("+" "-") <i>DecimalNumeral</i>	→ (Ausführen des Operators)
"+" <i>DecimalNumeral</i>	→ (Regel für <i>DecimalNumeral</i>)
"+" ("0" <i>NonZeroDigit</i> [<i>Digits</i>])	→ (Ausführen des Operators)
"+" <i>NonZeroDigit</i> [<i>Digits</i>]	→ (Regel für <i>NonZeroDigit</i>)
"+" ("1" ... "9") [<i>Digits</i>]	→ (Ausführen des Operators)
"+" "3" [<i>Digits</i>]	→ (Ausführen des Operators [])
"+" "3" <i>Digits</i>	→ (nicht rek. Regel für <i>Digits</i>)
"+" "3" <i>Digit</i> { <i>Digit</i> }	→ (Regel für <i>Digit</i>)
"+" "3" ("0" <i>NonZeroDigit</i>) { <i>Digit</i> }	→ (Ausführen des Operators)
"+" "3" <i>NonZeroDigit</i> { <i>Digit</i> }	→ (Regel für <i>NonZeroDigit</i>)
"+" "3" ("1" ... "9") { <i>Digit</i> }	→ (Ausführen des Operators)
"+" "3" "1" { <i>Digit</i> }	→ (Ausführen des Operators { })
"+" "3" "1"	

Ableitung von Worten

Ein Wort w kann vom Startsymbol der Grammatik abgeleitet werden (und ist dann in $L(G)$), falls es eine **Ableitung** der Form

$$E_0 \rightarrow E_1 \rightarrow \dots \rightarrow E_k$$

gibt, wobei:

- E_0 das Startsymbol der Grammatik ist,
- E_k zum Wort w identisch ist,
- E_{i+1} aus E_i entsteht durch
 - 1) Ersetzung eines oder mehrerer Nichtterminale durch die rechte Seite ihrer Regeln oder durch
 - 2) Ausführung von Operatoren, d.h.
 - $[E]$ darf durch E ersetzt oder gelöscht werden,
 - $E | F$ darf durch E oder durch F ersetzt werden,
 - $\{E\}$ darf gelöscht werden oder durch $E\{E\}$ ersetzt werden.

Häufig wird Schritt 1) gefolgt von Schritt 2) in einem Schritt zusammengefasst. Man spricht dann von einer „kurzen Ableitung“.

EBNF-Grammatik für Bezeichner

Ein **Bezeichner** (*Identifizier*) ist eine nichtleere Folge von Buchstaben oder Ziffern, beginnend mit einem Buchstaben.

Bezeichner sind z.B. A, A2D2, Passau

Keine Bezeichner sind 007, 1A, O.K. (*Punkte sind keine Buchstaben.*)

EBNF-Grammatik:

$$\begin{aligned} \textit{Letter} &= \text{"A"} | \\ &\quad \text{"B"} | \\ &\quad \dots \\ &\quad \text{"Z"} | \\ &\quad \text{"a"} | \\ &\quad \dots \\ &\quad \text{"z"} \end{aligned}$$
$$\textit{Identifizier} = \textit{Letter} \{ \textit{Letter} | \textit{Digit} \}$$

In Java müssen alle Variablennamen, Klassennamen usw. Bezeichner sein. Die Grammatik für Bezeichner ist etwas allgemeiner als oben.

BNF-Variante für Java

Bemerkung: Die Java-Spezifikation verwendet eine andere Variante der BNF.

- Nichtterminalsymbole *kursiv*
- Terminalsymbole Schreibmaschinenschrift (`if` statt `"if"`)
- Regeln `A: E` statt `A = E`
- Das Auswahlsymbol, d.h. `|`, wird weggelassen und durch neue Zeile ersetzt. Oder man verwendet *one of*.
- Die Option wird durch tiefgestelltes `opt` gekennzeichnet: `Eopt` statt `[E]`

Beispiele aus der Java-Spezifikation:

DecimalNumeral:

0
NonZeroDigit Digits_{opt}

Digits:

Digit
Digits Digit

Digit:

0
NonZeroDigit

NonZeroDigit: one of

1 2 3 4 5 6 7 8 9

Syntaxdiagramme

Ein **Syntaxdiagramm** ist ein einfacher grafischer Formalismus zur Definition von Sprachen. Es besteht aus

- Rechtecken, in denen die Nichtterminale stehen,
- Ovalen, in denen die Terminale stehen,
- Pfeilen, die die Elemente verbinden,
- Eingangs- und Ausgangspfeilen

Beispiele für Syntaxdiagramme

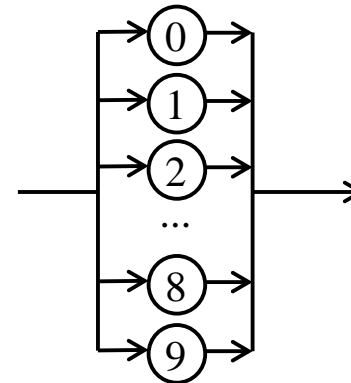
Satz :



Beispiel:

Digit = "0" |
 "1" |
 "2" |
 ...
 "9"

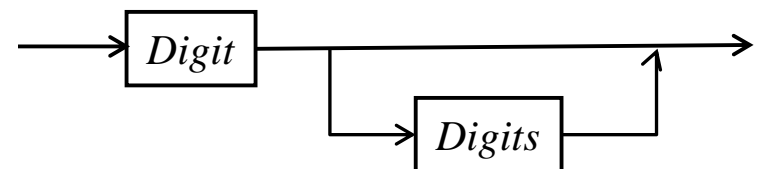
Digit :



Beispiel:

Digits = *Digit* [*Digits*]

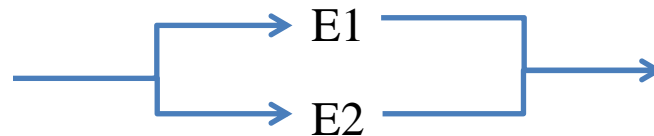
Digits :



Korrespondenz von Syntaxdiagrammen und EBNF (1)

Jeder EBNF-Operator lässt sich durch ein Syntaxdiagramm ausdrücken:

- **Auswahl:** $E1 \mid E2$ wird repräsentiert durch eine Verzweigung.



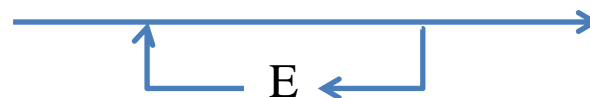
- **Sequentielle Komposition:** $E1 E2$ wird repräsentiert durch Aneinanderhängen



- **Option:** $[E]$ wird repräsentiert durch



- **Wiederholung:** $\{E\}$ wird repräsentiert durch



Korrespondenz von Syntaxdiagrammen und EBNF (2)

Umgekehrt lässt sich jedes Syntaxdiagramm durch eine EBNF-Grammatik ausdrücken.

Folgerung:

EBNF und Syntaxdiagramme sind äquivalent in dem Sinne, dass sie die gleiche Klasse von (formalen) Sprachen beschreiben.

Bemerkung:

Man nennt sie die Klasse der **kontextfreien Sprachen**, da Nichtterminalsymbole ohne Berücksichtigung ihrer benachbarten Symbole (d.h. ohne Berücksichtigung des Kontexts) durch Ausdrücke (nämlich durch die rechten Seiten der zugehörigen Regeln) ersetzt werden.

Beispiel für eine **nicht-kontextfreie** Sprache:

Die Sprache bestehend aus allen Wörtern der Form $a^n b^n c^n$ mit $n \geq 1$, d.h. alle Wörter der Form

abc, aabbcc, aaabbccc, aaaabbbbccccc, ...

Beispiel: Palindrome

Ein **Palindrom** ist ein nichtleeres Wort, das von links wie von rechts gelesen das Gleiche ergibt.

(griechisch: Παλίνδρομος (*palíndromos*) „rückwärts laufend“ [Wikipedia])

Palindrome:

„lege an eine brandnarbe nie naegel“

(wenn man Leerzeichen ignoriert)

ANNA

ANANA

NN

A

37873

Keine Palindrome:

ANANAS

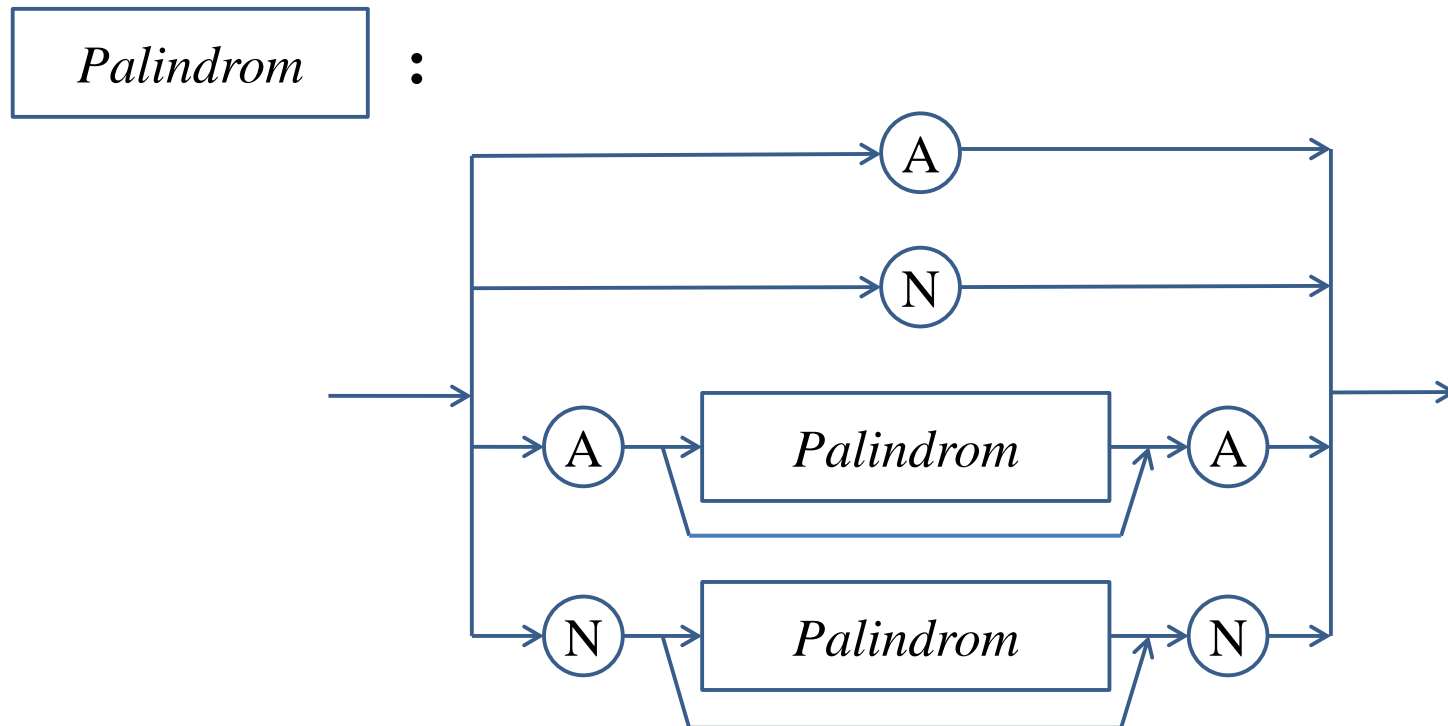
ANAN

ANAAA

37863

Syntaxdiagramm für Palindrome

Syntaxdiagramm für Palindrome, die mit den Buchstaben A und N gebildet werden können:



EBNF-Grammatik für Palindrome

EBNF-Grammatik für Palindrome, die mit den Buchstaben A und N gebildet werden können:

```
Palindrom = "A" |  
            "N" |  
            "A" [Palindrom] "A" |  
            "N" [Palindrom] "N"
```