"Grammatik,
die sogar Könige zu kontrollieren weiß...„
– aus Molière, Les Femmes Savantes (1672), 2. Akt

Syntax von Programmiersprachen

Prof. Dr. Martin Wirsing

in Zusammenarbeit mit
Michael Barth, Philipp Meier und Gefei Zhang

10/04

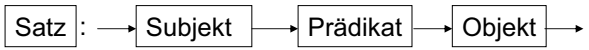
Ziele

- Zwei Standards zur Definition der
  Syntax von Programmiersprachen kennen lernen:
  - Backus-Naur-Form (BNF)
  - Syntaxdiagramme

Syntax von Programmiersprachen und Syntaxdiagramme

- **Beispiel für eine Grammatik**

Als Syntaxdiagramm:



In BNF-Form: Satz = Subjekt Prädikat Objekt

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

  • *Ein Programmierer schreibt ein Programm.*
  - *Ein Programmierer* Subjekt
  - *schreibt* Prädikat
  - *ein Programm* Objekt

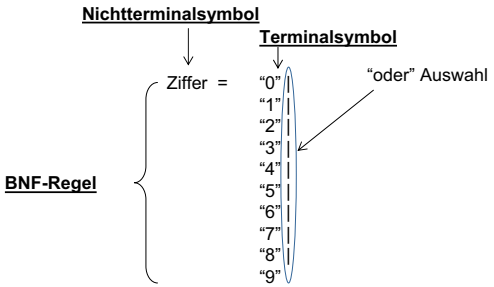
  • *Ein Student liest einen Text.*
  Dieser Satz ist syntaktisch korrekt.

Syntax von Programmiersprachen und Syntaxdiagramme

- Die **BNF (Backus-Naur-Form),** benannt nach **John Backus** und **Peter Naur**, wurde erstmals zur Beschreibung der Syntax von Algol 60 verwendet.
- Heute ist die BNF (in notationellen Varianten) die Standardbeschreibungstechnik für Programmiersprachen.
- Wir verwenden in der Vorlesung die „erweiterte Backus-Naur-Form“ EBNF.
- Auch die Syntax von Java wird in der Backus-Naur-Form beschrieben.

Backus-Naur-Form

Beispiel: EBNF für Ziffern



Vorzeichen = "+" | "-"

Backus-Naur-Form

Zahl ist eine nicht-leere Folge von **Ziffern**

Zahlen werden also durch **Aneinanderhängen von Zahlen und Ziffern** gebildet.
Spezielle Beispiele für Zahlen sind:

- "1" → **Zahl**
- "1" "5" → **Zahl** *Ziffer*
- "1" "5" "7" → **Zahl** *Ziffer* → **Zahl**

- Zahl = Ziffer | Zahl Ziffer *Rekursion*
- GanzeZahl (engl. Integer) ist eine Zahl mit möglichen **Vorzeichen**.

  GanzeZahl = [Vorzeichen] Zahl ***BNF-Regel***

Backus-Naur-Form

Wir unterscheiden 2 Arten von Symbolen:

- **Nichtterminalsymbole** stehen für Begriffe, die durch **BNF-Regeln** erklärt werden (z.B Ziffer, Zahl, …)
- **Terminalsymbole** (geschrieben in Anführungszeichen) stehen für Symbole und Namen, die so in den Programmtext übernommen werden (z.B. "0", "1", "+", …)

Jede BNF-Regel hat die Form: **A = Ausdruck**

wobei
A ein Nichtterminalsymbol ist und
ein **Ausdruck** gebildet wird aus:

- Terminal- und Nichtterminalsymbolen
- Operatoren zum Bilden von Ausdrücken

Auswahl	E1 | E2	für "E1 oder E2"
Sequenzielle Komposition	E1 E2	(direkt hintereinander schreiben) für "E2 folgt direkt auf E1"
Option:	[E1]	für "E1 kann 1-mal oder 0-mal (d.h. nicht) vorkommen"

Wie erhält man aus der Grammatik korrekte Wörter?

Ist +31 eine GanzeZahl?
Wir bilden folgende Ableitung:

GanzeZahl	→ (rechte Seite der Def. von GanzeZahl)
[Vorzeichen] Zahl	→ Ausführen des Operators [])
Vorzeichen Zahl	→ (Ersetzen von Vorzeichen durch rechte Seite der Def.)
("+"|"-") Zahl	→ (Ausführen der Auswahl)
"+" Zahl	→ (Def. von Zahl)
"+" (Ziffer | Zahl Ziffer)	→ (Ausführen von |)
"+" (Zahl Ziffer)	→ (Def. von Zahl)
"+" ((Ziffer | Zahl Ziffer) Ziffer)	→ (Auswahl)
"+" Ziffer Ziffer	→ (Def. von Ziffer)
"+"("0"|"1"|...|"9") Ziffer	→ (mehrfache Auswahl)
"+" "3" Ziffer	→ (Def. von Ziffer)
"+" "3" ("0"|"1"|...|"9")	→ (mehrfache Auswahl)
"+" "**3**" "**1**"	

BNF-Grammatik

- Eine **BNF-Grammatik** besteht aus einer Menge von **BNF-Regeln zusammen mit einem Startsymbol** (Nichtterminalzeichen für den betrachteten Begriff z.B. GanzeZahl).
- Jede solche **Grammatik G** definiert eine **Menge von Wörtern**, die als Sprache von G, geschrieben **L(G)** , bezeichnet wird. (L für Language)

Ein **Wort** ist eine **Folge von Terminalzeichen**.

BNF-Grammatik

Man erhält jedes Wort w von L(G) durch eine **Ableitung** der Form

$S \rightarrow E_1 \rightarrow E_2 \rightarrow ... \rightarrow E_k$

wobei

- S das Startsymbol von G ist,
- E_{i+1} aus E_i dadurch entsteht, dass
  - ein oder mehrere Nichtterminale aus E_i durch die rechten Seiten ihrer Definition ersetzt werden oder dass
  - Operatoren ausgewertet werden.
- $E_k \equiv w$

  ↓

  syntaktisch gleich

  Hier bedeutet "≡" die syntaktische Gleichheit.

Bei dem abgeleiteten Wort w verzichtet man im Allgemeinen auf die Anführungszeichen.

BNF-Grammatik für Java

Bemerkung:

Die Java-Spezifikation verwendet eine etwas andere Schreibweise:

- *Nichtterminalsymbole* — *kursiv* (z.B. *Ziffer* statt Ziffer)
- Terminalsymbole — Schreibmaschinenschrift (z.B. `if` statt "if")
- Statt A = E schreibt man A: E
- Die Auswahl wird weggelassen und durch neue Zeile ersetzt.
  z.B. *Vorzeichen*: +
  -
- Die Option wird durch tiefgestelltes opt gekennzeichnet.
  z.B. E_{opt} statt [E]

Beispiel:

Zahl : *Ziffer*
Zahl Ziffer

Vorzeichen : +
−

GanzeZahl: $Vorzeichen_{opt}$ *Zahl*

Abgeleitete Operationen

Eine **Folge von Ausdrücken <u>gleicher</u> Bauart** läßt sich so beschreiben:

Sei E ein Ausdruck

{E} bedeutet: E kann 0-mal oder mehrmals auftreten.

Man kann {E} folgendermaßen definieren:
{E} = [E] | *E kann beliebig (aber endlich) oft auftreten.*
E {E}

Äquivalent dazu ist die Definition:
{E} = [E] |
{E} E

Beispiel: Bezeichner

Ein **Bezeichner** besteht aus einer nichtleeren Folge von Buchstaben oder Ziffern, beginnend mit einem Buchstaben.

Bezeichner sind z.B. A, A2D2, Wirsing
Keine Bezeichner sind 007, 1B, F.D.P *(Punkte sind keine Buchstaben.)*

EBNF-Grammatik:

Buchstabe = "A"|
"B"|
...
"Z"|
"a"|
...
"z"

Buzi = Buchstabe|
Ziffer

Bezeichner = Buchstabe {Buzi}

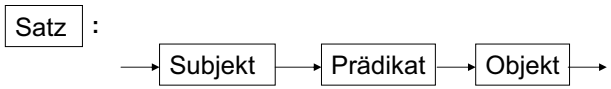
Syntaxdiagramme

Ein **Syntaxdiagramm** ist ein einfacher grafischer Formalismus zur Beschreibung der Syntax von Programmen.

Ein **Syntaxdiagramm** erklärt (wie eine BNF-Grammatik) einen syntaktischen Begriff.
Es besteht aus Rechtecken, Ovalen und Pfeilen.

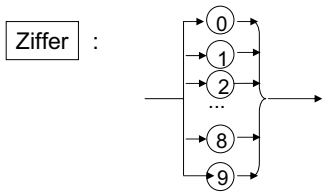
- Nichtterminale stehen in Rechtecken
- Terminale stehen in Ovalen
- Pfeile führen von einem Sprachelement zum anderen
- Jedes Syntaxdiagramm besitzt genau einen Eingangs- und genau einen Ausgangspfeil.

Beispiele für Syntaxdiagramme



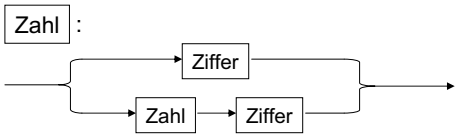
Beispiel:

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



Beispiel:

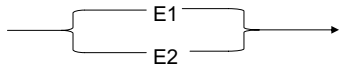
Zahl = Ziffer |
Zahl Ziffer



Korrespondenz von Syntaxdiagrammen und BNF

Jeder BNF-Operator läßt sich durch ein Syntaxdiagramm ausdrücken:

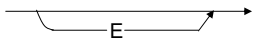
- Auswahl: E1 | E2 wird repräsentiert durch eine Verzweigung.



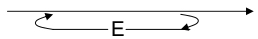
- Komposition: E1 E2 wird repräsentiert durch Hintereinanderfügen

E1 → E2

- Option: [E] wird repräsentiert durch



- Wiederholung:



Korrespondenz von Syntaxdiagrammen und BNF

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

Folgerung:
BNF 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 rechte 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>=1, d.h. alle Wörter der Form
abc, aabbcc, aaabbbccc, aaaabbbbcccc, ...

Beispiel für BNF und Syntaxdiagramme: Palindrome

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

Palindrome sind:

"Ein Neger mit Gazelle zagt im Regen nie"
ANNA
ANANA
NN
A

Kein Palindrom:

ANANA**S**
ANA**N**
A**NAA**A

Beispiel für BNF und Syntaxdiagramme: Palindrome

BNF-Form:

Die folgende Grammatik beschreibt alle Palindrome, die mit den Terminalzeichen "A" und "N" gebildet werden können.

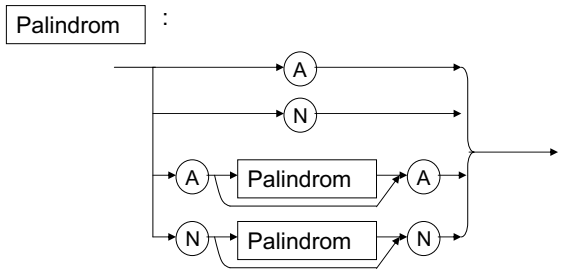
Palindrom — *Nichtterminalsymbol*
"A", "N" — Terminalsymbole

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

} "einfachste Fälle, Abbruchfälle"

Beispiel für BNF und Syntaxdiagramme: Palindrome

Syntaxdiagramm:



Zusammenfassung

- Die **Backus-Naur Form (BNF)** und **Syntaxdiagramme** sind Formalismen zur **Definition der Syntax von Programmiersprachen.**
- Eine **BNF-Grammatik** besteht aus
  - einer Menge von **BNF-Regeln** und
  - einem **Startsymbol.**
- Jede **BNF-Grammatik G** definiert eine **Menge L(G) von Wörtern**, die als Sprache von G bezeichnet wird.
- Ein **Syntaxdiagramm** ist ein zu BNF äquivalenter grafischer Formalismus, bei dem
  - Nichtterminalsymbole in Rechtecken
  - Terminalsymbole in Ovale repräsentiert werden und
  - Pfeile von einem Sprachelement zum anderen führen;
  - dabei besitzt jedes Syntaxdiagramm genau einen Eingangs- und genau einen Ausgangspfeil.