7. Formale Sprachen und Grammatiken

Computer verwenden zur Verarbeitung von Daten und Informationen künstliche, *formale Sprachen* (Maschinenspr., Assemblerspachen, Programmierspr., Datenbankspr., Wissensrepräsentationsspr., ...)
Vorteile gegenüber natürlichen Sprachen:
- **exakte Definition der zulässigen Ausdrücke und ihrer Bedeutung**
- **keine Kontextabhängigkeit der Bedeutung**
- **dadurch erheblich leichtere Verarbeitung**

Eine Sprache L besteht aus Wörtern über einem zugrundeliegenden Alphabet Σ, es gilt also: L Teilmenge von Σ*.
Die Syntax einer Sprache legt fest, welche Ausdrücke zur Sprache gehören.
Die Semantik legt fest, was die Ausdrücke bedeuten.
Endliche Sprachen können durch Aufzählen ihrer Elemente definiert werden, für unendliche Sprachen braucht man endliche Beschreibung.

Formale Sprachen

Formale Sprache

***Alphabet*: geordneter Zeichenvorrat**

***Formale Sprache*: Menge von Zeichenketten, die aus den Symbolen eines beliebigen Alphabets aufgebaut sind.**

***Zeichenkette*: endliche Folge von Symbolen, die ohne Zwischenraum hintereinander geschrieben werden (Verkettung)**

Wohlgeformtheit

Definition der *zulässigen Zeichenketten* einer Sprache durch induktive Definition

***induktive Definition*:**

1. **definitorische Anfangsbestimmung**
2. **generierende Bestimmung**
3. **Abschlußbestimmung**

Beispiel Aussagenlogik

Aussagenlogik - Syntax

Aussagenlogische Variable und die logischen Konstanten *wahr* bzw. *falsch* sind *atomare Formeln.* Es gilt:

(1) Alle atomaren Formeln sind Formeln.

(2a) Für alle Formeln F und G sind (F∧G) und (F∨G) Formeln.

(2b) Für jede Formel F ist ¬F eine Formel.

(3) Nichts ist eine Formel, was nicht als solche durch (1) und (2) bestimmt ist.

Grammatiken

Formale Sprachen können durch Grammatiken beschrieben werden.
Eine Grammatik G = (T, N, P, S) besteht aus folgenden Komponenten:

- **einer endlichen Menge N von Variablen (Nichtterminalsymbolen),**
- **dem endlichen Terminalalphabet T,**
- **einer endlichen Menge P von Regeln $\alpha \rightarrow \beta$, wobei α und β aus N und T gebildet sind,**
- **einer Startvariablen S aus N.**

Die durch G erzeugte Sprache L(G) wird wie folgt definiert:
Die Zeichenkette v heißt direkt ableitbar aus der Zeichenkette u (Notation: u => v) genau dann wenn u = xyz, v = xy'z und $y \rightarrow y' \in P$.
=>* bezeichnet die reflexive und transitive Hülle von =>.

Wir definieren: $L(G) = \{w \in \Sigma^* \mid S =>^* w\}$

Ableitungsbaum

1. **Die Wurzel des Ableitungsbaums ist mit dem Startsymbol markiert.**
2. **Ist ein Knoten mit einem Terminalsymbol markiert, dann ist er ein Blatt und hat keinen Sohn.**
3. **Ist ein Knoten mit einem Nichtterminalsymbol X markiert, und hat er n Söhne, die von links nach rechts mit X1, X2, ...,Xn markiert sind, so enthält die Grammatik die Regel**

   X→ X1, X2, ..., Xn

Die Folge der Blätter von links nach rechts gelesen ergibt ein Wort der Sprache.

Chomsky Grammatik

Induktive Definition. Aufgeführt sind die in der jeweils höheren Sprache zusätzlich gültigen Regeln.

Reguläre Sprachen (Typ 3)

Produktionsregeln der Form

<X> → a|a<Y> **(alternativ <X> → <Y>a)**

Bemerkung: Das Nichtterminalsymbol steht in allen Regeln entweder immer rechts oder immer links

Kontextfreie Sprachen (Typ 2)

Produktionsregeln der Form

<X> → u<Y>v

Kontextsensitive Sprachen (Typ 1)

Produktionsregeln der Form

u<X>v → u<Y>v wobei die Länge der Zeichekette auf der linken Seite nicht größer als die der rechten Seite sein darf.

Unbeschränkte Sprachen (Typ 0)

Typen von Grammatiken

eine Grammatik heißt

- **kontextsensitiv falls für alle Regeln w1 -> w2 gilt: |w1| ≤ |w2| (|w|: Anzahl der Variablen und Terminalzeichen von w)**
- **kontextfrei falls für alle Regeln w1 -> w2 gilt: w1 aus V**
- **regulär falls zusätzlich gilt: w2 Terminalzeichen oder Terminalzeichen gefolgt von Variable**

durch Grammatiktypen erzeugbare Sprachen bilden echte Teilmengen

Chomsky-Hierarchie

Eine Grammatik für die natürliche Sprache

P:

<Satz>	**->**	**<Subjekt> <Prädikat> <Objekt>**
<Subjekt>	**->**	**<Artikel> <Attribut> <Substantiv>**
<Artikel>	**->**	ε
<Artikel>	**->**	**der**
<Artikel>	**->**	**die**
<Artikel>	**->**	**das**
<Attribut>	**->**	ε
<Attribut>	**->**	**<Adjektiv>**
<Attribut>	**->**	**<Adjektiv> <Attribut>**
<Adjektiv>	**->**	**kleine**
<Adjektiv>	**->**	**bissige**
<Adjektiv>	**->**	**große**
<Substantiv>	**->**	**Hund**
<Substantiv>	**->**	**Katze**
<Prädikat>	**->**	**jagt**
<Objekt>	**->**	**<Artikel> <Attribut> <Substantiv>**

N: **{<Satz>, <Subjekt>, <Artikel>, <Attribut>, <Adjektiv>, <Substantiv>, <Prädikat>, <Objekt>}**

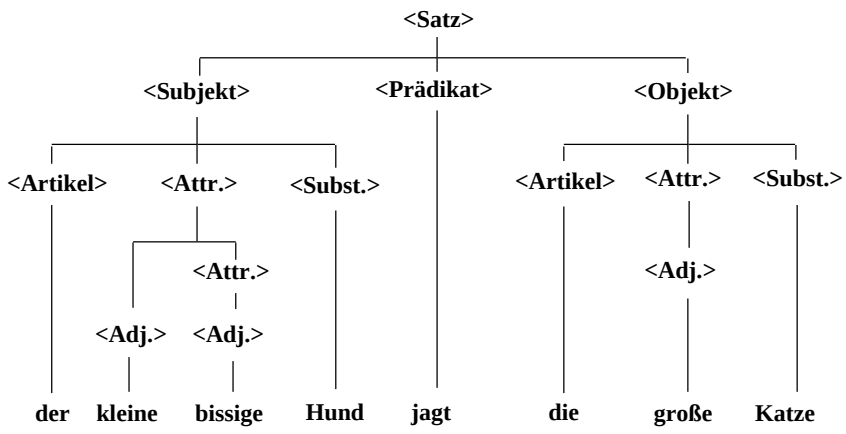
T: **{der, die, das, kleine, bissige, große, Hund, Katze, jagt}**

S: **<Satz>**

Eine Ableitung

<Satz> => <Subjekt> <Prädikat> <Objekt>
=> <Artikel> <Attribut> <Substantiv> <Prädikat> <Objekt>
=> der <Attribut> <Substantiv> <Prädikat> <Objekt>
=> der <Adjektiv> <Attribut> <Substantiv> <Prädikat> <Objekt>
=> der kleine <Attribut> <Substantiv> <Prädikat> <Objekt>
=> der kleine <Adjektiv> <Substantiv> <Prädikat> <Objekt>
=> der kleine bissige <Substantiv> <Prädikat> <Objekt>
=> der kleine bissige Hund <Prädikat> <Objekt>
=> der kleine bissige Hund jagt <Objekt>
=> der kleine bissige Hund jagt <Artikel> <Attribut> <Substantiv>
=> der kleine bissige Hund jagt die <Attribut> <Substantiv>
=> der kleine bissige Hund jagt die <Adjektiv> <Substantiv>
=> der kleine bissige Hund jagt die große <Substantiv>
=> der kleine bissige Hund jagt die große Katze

Ein Ableitungsbaum



Backus - Naur - Formen.

Vereinfachte Darstellung kontextfreier Grammatiken durch BNF oder erweiterte BNF (EBNF) sowie durch Syntaxdiagramme (s.u.)

BNF Notation:

• Nichtterminale Symbole durch spitze Klammern < ... > gekennzeichnet.
• **Metasymbole** sind
  • ::= ist definiert als
  • | trennt Alternativen.

EBNF (erweiterte BNF) Notation:

Im Vergleich zu BNF reichere Metasyntax und damit einfachere und besser lesbare Schreibweise.

• **Metasymbole** sind
  • (...|...) genau eine Alternative kann stehen.
  • [...] Option: Inhalt der Klammer steht einmal oder keinmal.
  • {...} Wiederholung: Inhalt der Klammer steht **mindestens** einmal, d.h. {A}=A|AA|AAA|...

Beispiel: {(a|b)} erzeugt die Menge der Zeichenketten {a,b,aa,ab,ba,bb,...} denn der Inhalt der Klammer ist der Ausdruck A=(a|b), dessen Auswertung entweder a oder b ergibt.

BNF-Beispiel

θ Bezeichner, also Namen für Variablen, Prozeduren oder Konstanten, können wie folgt beschrieben werden:

<Bezeichner> **::=** **<Buchstabe>|**
<Bezeichner><Buchstabe>|
<Bezeichner><Ziffer>

<Buchstabe> **::=** **A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z**
a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

<Ziffer> **::=** **1|2|3|4|5|6|7|8|9|0**

Die Wertebereiche der Metavariablen sind also:

<Ziffer> **:** **{1, ..., 9, 0}**
<Buchstabe> **:** **{A, ..., Z, a, ..., z}**
<Bezeichner> **:** **alle endlichen, nichtleeren Zeichenketten über der Menge {1, ..., 9, 0} ∪ {A, ..., Z, a, ..., z}, die mit einem Buchstaben beginnen, z.B.: "A", "Hans", "A12", "ergo", etc.**

θ Zulässige Variablenvereinbarungen, also zum Beispiel Zeichenketten der Form "VAR x : INTEGER;" , können mit Hilfe der Backus-Naur-Form so spezifiziert werden:

<VarVereinb> **::=** **VAR <Vereinbarungen>**

<Vereinbarungen> **::=** **<Vereinbarung><Vereinbarungen>**

<Vereinbarung> **::=** **<Bezeichnerliste>:<Typ>;**

<Bezeichnerliste> **::=** **<Bezeichner>|<Bezeichnerliste>,<Bezeichner>**

<Typ> **::=** **INTEGER | REAL | BOOLEAN |**

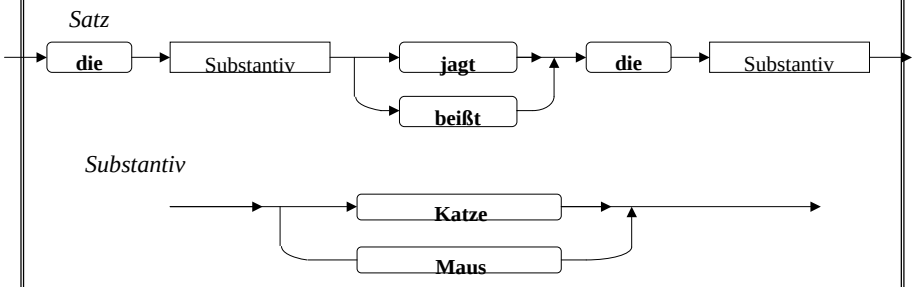
Erweiterte Backus-Naur-Form (EBNF)

Zusätzliche Sprachkonstrukte sind u.a.:

- Ein Metaausdruck in eckigen Klammern, also ein Ausdruck der Gestalt [Metaausdruck], bezeichnet Zeichenketten, die optional sind und somit auch weggelassen werden können.
- Ein Metaausdruck in geschweiften Klammern, also ein Ausdruck der Form {Metaausdruck}, beschreibt Zeichenketten, die beliebig oft wiederholt werden dürfen (0, 1, 2, ...-mal). Dieses Sprachkonstrukt verkörpert also eine Art Iteration.
- Gebräuchlich ist ferner ein Ausdruck der Form $\{\text{Metaausdruck}\}_a^b$, mit dem festgelegt wird, daß die durch Metaausdruck definierten Zeichenketten mindestens a-mal, höchstens b-mal wiederholt werden müssen bzw. dürfen. Dabei wird $0 \le a \le b$ vorausgesetzt, und es ist auch b = * erlaubt, um auszudrücken, daß die Anzahl der Iterationen nach oben unbeschränkt ist.

Syntaxdiagramme

- Die Syntax einer (kontextfreien) Programmiersprache läßt sich durch sog. Syntaxdiagramme beschreiben. Sie erlauben die Ableitung aller syntaktisch korrekten Sätze der Sprache
- Prinzipielle Anwendung
  - ⇨Rechteck entspricht **syntaktischer Variable** (Nicht-Terminal-symbol) in BNF
  - ⇨Rechteck mit <Name> wird bei Durchlauf ersetzt durch Syntax-diagramm mit <Name>
  - ⇨Langrunde/Kreis mit Symbol produziert (gibt aus) bei Durchlauf das entsprechende Symbol
- Beispiel



Umwandlung der Darstellungsform

- Kein prinzieller Unterschied hinsichtlich der Ausdrucksmächtigkeit von Syntaxdiagrammen und EBNF
- **Automatische Umwandlung** möglich

- Vorteile der Syntaxdiagramme
  - ⇨ hohe Anschaulichkeit
  - ⇨ geringe Anzahl der Regeln (Diagramme); typischerweise sind mehrere Produktionen der EBNF in einem Syntaxdiagramm zusammengefaßt

- Vorteile einer BNF-Grammatik
  - ⇨ leichte Erstellung und Änderung
  - ⇨ erzwingt eine klare Struktur
  - ⇨ direkte Verwendung in Übersetzer-erzeugenden Systemen möglich