# 5. Modellierung von Strukturen
# 5.1 Kontextfreie Grammatiken

**Kontextfreie Grammatik (KFG)**: formaler Kalkül, Ersetzungssystem; definiert

- **Sprache** als Menge von Sätzen; jeder **Satz** ist eine **Folge von Symbolen**
- **Menge von Bäumen**; jeder Baum repräsentiert die **Struktur eines Satzes** der Sprache

**Anwendungen**:

- Programme einer **Programmiersprache** und deren Struktur, z. B. Java, Pascal, C
- Sprachen als Schnittstellen zwischen Software-Werkzeugen, **Datenaustauschformate**, z. B. HTML, XML
- Bäume zur Repräsentation **strukturierter Daten**, z. B. in HTML
- Struktur von **Protokollen** beim Austausch von Nachrichten zwischen Geräten oder Prozessen

**Beispiel zu HTML:**

```
<table>
    <tr>
        <td>Mo</td>
        <td>11-13</td>
        <td>AM</td>
    </tr>
    <tr>
        <td>Fr</td>
        <td>9-11</td>
        <td>AM</td>
    </tr>
</table>
```

# Kontextfreie Grammatik

Eine kontextfreie Grammatik G = (T, N, P, S) besteht aus:

| | |
|---|---|
| T | **Menge der Terminalsymbole** (kurz: Terminale) |
| N | **Menge der Nichtterminalsymbole** (kurz: Nichtterminale)<br>T und N sind disjunkte Mengen |
| $S \in N$ | **Startsymbol** (auch Zielsymbol) |
| $P \subseteq N \times V^*$ | **Menge der Produktionen**; $(A, x) \in P$, mit $A \in N$ und $x \in V^*$;<br>statt (A, x) schreibt man A ::= x |
| $V = T \cup N$ | heißt auch **Vokabular**, seine Elemente heißen **Symbole** |

Man sagt „In der Produktion A ::= x steht A auf der **linken Seite** und x auf der **rechten Seite**."
Man gibt Produktionen häufig **Namen**: p1: A ::= x
In Symbolfolgen aus V* werden die Elemente nur durch Zwischenraum getrennt: A ::= B C D

**Beispiel**:

**Terminale** **T** = { `(`, `)` }

**Nichtterminale** **N** = { `Klammern`, `Liste` }

**Startsymbol** **S** = `Klammern`

| **Name** | **Produktionsmenge P**<br>**N** | | **V*** |
|---|---|---|---|
| | { | | |
| `p1:` | `Klammern` | `::=` | `'(' Liste ')'` |
| `p2:` | `Liste` | `::=` | `Klammern Liste` |
| `p3:` | `Liste` | `::=` | |
| | } | | |

# Bedeutung der Produktionen

Eine Produktion A ::= x ist eine **Strukturregel**: A **besteht aus** x

**Beispiele**:

| | | |
|---|---|---|
| DeutscherSatz | ::= | Subjekt Prädikat Objekt |
| *Ein* DeutscherSatz | *besteht aus (der Folge)* | Subjekt Prädikat Objekt |
| Klammern | ::= | '(' Liste ')' |
| Zuweisung | ::= | Variable ':=' Ausdruck |
| Variable | ::= | Variable '[' Ausdruck ']' |

**Produktion graphisch als gewurzelter Baum** mit geordneten Kanten und mit Symbolen als Knotenmarken:

# Ableitungen

Produktionen sind **Ersetzungsregeln**: Ein Nichtterminal A in einer Symbolfolge u A v kann durch die rechte Seite x einer Produktion A ::= x ersetzt werden.
Das ist ein **Ableitungsschritt**; er wird notiert als u A v => u x v

z. B. **`Klammern Klammern Liste`** => **`Klammern ( Liste ) Liste`**
mit Produktion p1

Beliebig viele Ableitungsschritte nacheinander angewandt heißen **Ableitung**; notiert als u =>* v

Eine kontextfreie Grammatik **definiert eine Sprache**; das ist eine Menge von Sätzen.
Jeder Satz ist eine Folge von Terminalsymbolen, die aus dem Startsymbol ableitbar ist:

$$L(G) = \{ w \mid w \in T^* \text{ und } S \Rightarrow^* w \}$$

Grammatik auf Mod-5.2 definiert geschachtelte Folgen paariger Klammern als Sprachmenge:

{ `()`, `(())`, `(()()())`, `((()())(()))`, ... } ⊆ L(G)

Ableitung des Satzes `(()())`:

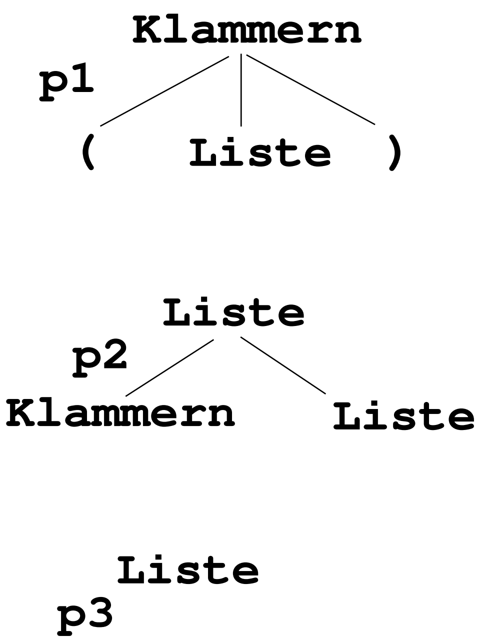
**S = Klammern**
**=> ( Liste )**
**=> ( Klammern Liste )**
**=> ( Klammern Klammern Liste )**
**=> ( KLammern ( Liste ) Liste )**
**=> ( ( Liste ) ( Liste ) Liste )**
**=> ( ( ) ( Liste ) Liste )**
**=> ( ( ) ( ) Liste )**
**=> ( ( ) ( ) )**

# Ableitungsbäume

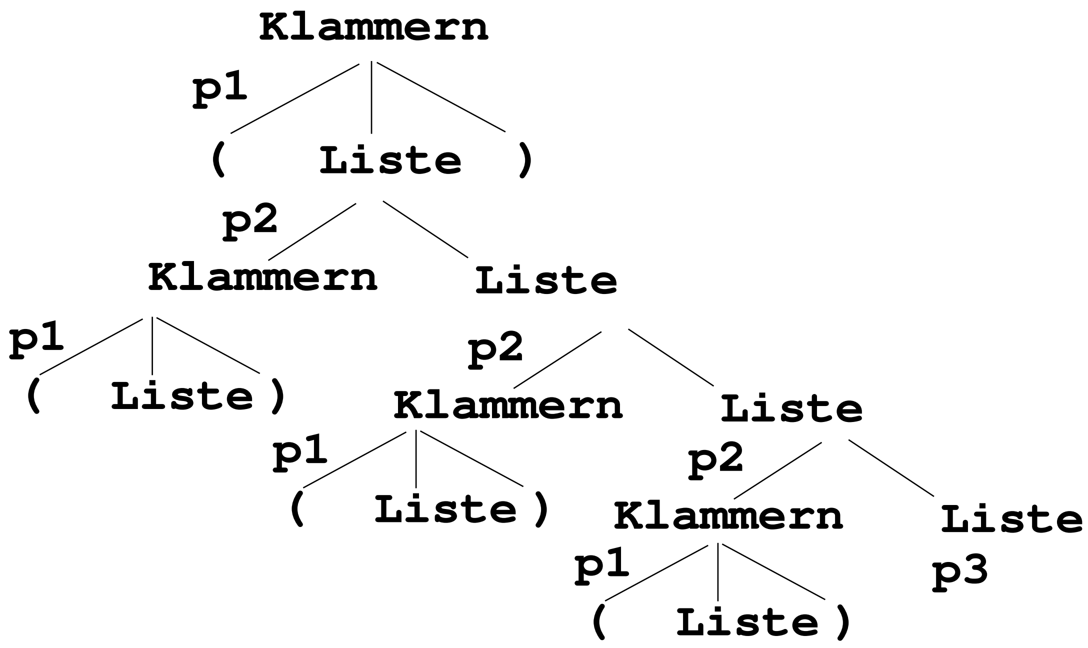
Jede Ableitung kann man als **gewurzelten Baum** darstellen:

Die **Knoten** mit ihren Marken repräsentieren **Vorkommen von Symbolen**.
Ein Knoten mit seinen direkten Nachbarn repräsentiert die **Anwendung einer Produktion**.
Die **Wurzel** ist mit dem **Startsymbol** markiert.
**Terminale** kommen nur an **Blättern** vor.

Produktionen:




ein Ableitungsbaum:




der Satz dazu: `(()()())`

Satz zum Baum: Terminale im links-abwärts Durchgang

# Beispiel: Ausdrucksgrammatik

p1: Ausdruck ::= Ausdruck BinOpr Ausdruck

p2: Ausdruck ::= Zahl

p3: Ausdruck ::= Bezeichner

p4: Ausdruck ::= '(' Ausdruck ')'

p5: BinOpr ::= '+'

p6: BinOpr ::= '-'

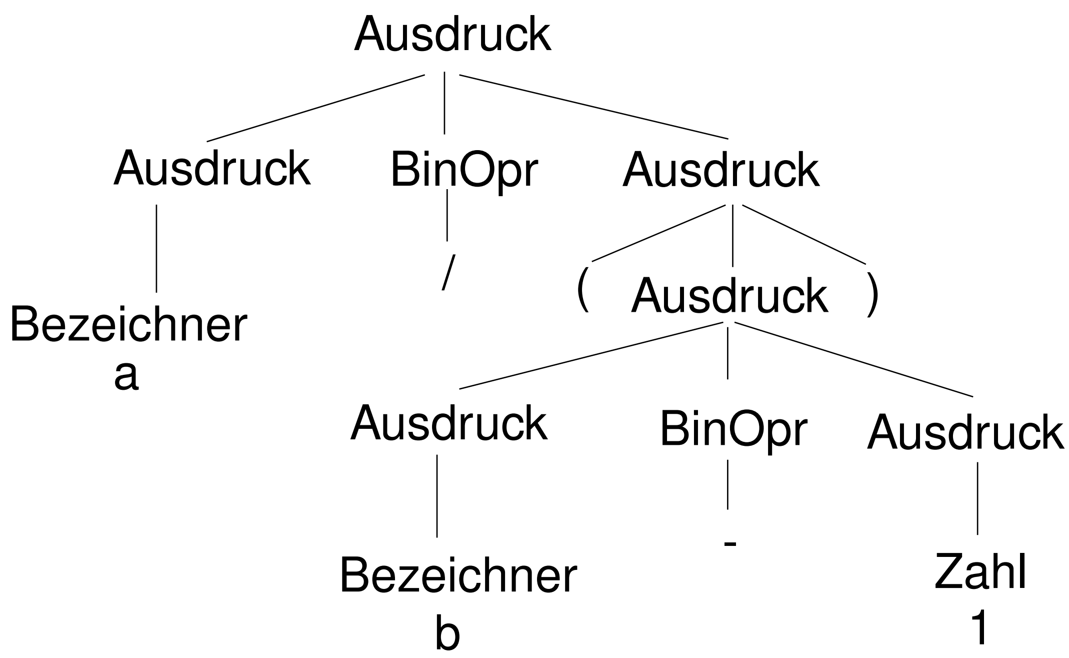
p7: BinOpr ::= '*'

p8: BinOpr ::= '/'

Startsymbol: Ausdruck

Terminale:
{ Zahl, Bezeichner, (, ), +, -, *, / }

Ableitungsbaum zum Ausdruck
a / (b - 1)




Schreibweise der Terminale
Zahl und Bezeichner wird nicht in der KFG definiert.

**Grammatik ist mehrdeutig**: Es gibt **Sätze, die mehrere Ableitungsbäume** haben.

# Beispiel: Tabellen in HTML

**HTML**: Hypertext Markup Language zur Darstellung von verzeigerten Dokumenten, insbesondere im WWW verwendet.
**typisch: geklammerte Strukturen** mit Klammern der Form **<x>...</x>**.

hier: vereinfachter Ausschnitt aus der Sprache zur Darstellung von Tabellen.

**Produktionen der kontextfreien Grammatik**:

```
Table   ::= '<table>' Rows '</table>'

Rows    ::= Row *

Row     ::= '<tr>' Cells '</tr>'

Cells   ::= Cell *

Cell    ::= '<td>' Text '</td>'

Cell    ::= '<td>' Table '</td>'
```

**Beispieltext in HTML**:

```
<table>
    <tr> <td>Tag</td>
         <td>Zeit</td>
         <td>Raum</td></tr>
    <tr> <td>Mo</td>
         <td>11:00-12.30</td>
         <td>AM</td></tr>
    <tr> <td>Fr</td>
         <td>9:15-10:45</td>
         <td>AM</td></tr>
</table>
```

**Darstellung der Tabelle**:

| | | |
|---|---|---|
| Tag | Zeit | Raum |
| Mo | 11:00-12.30 | AM |
| Fr | 9:15-10:45 | AM |

**Erweiterung der Notation vonKFGn:**

**X *** auf der rechten Seite einer Produktion steht für eine **beliebig lange Folge von X**

(gleiche Bedeutung wie bei Wertebereichen)