

6 Modellierung von Strukturen

6.1 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>
```

Vorlesung Modellierung WS 2011/12 / Folie 601

Ziele:

Einsatz von KFGn kennenlernen

in der Vorlesung:

Erläuterungen zu den Anwendungen

nachlesen:

Kastens, Kleine Übung: Modellierung, Abschnitt 6.1

Kontextfreie Grammatik

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

- T** Menge der **Terminalsymbole** (kurz: Terminale)
N Menge der **Nichtterminalsymbole** (kurz: Nichtterminale)
 T und N sind disjunkte Mengen
S ∈ N **Startsymbol** (auch Zielsymbol)
P ⊆ N × V* **Menge der Produktionen**; (A, x) ∈ P, mit A ∈ N und x ∈ V*;
 statt (A, x) schreibt man A ::= x
 V = T ∪ 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:

Terminalsymbole T = { (,) }

Nichtterminalsymbole N = { Klammern, Liste }

Startsymbol S = Klammern

Produktionsmenge P =

```
Name N V*
{
p1: Klammern ::= '(' Liste ') '
p2: Liste ::= Klammern Liste
p3: Liste ::=
}
```

Vorlesung Modellierung WS 2011/12 / Folie 602

Ziele:

KFG Definition lernen

in der Vorlesung:

- Erläuterung der Begriffe an dem Beispiel
- Erläuterung der Notation von Produktionen
- Unbenannte Terminale werden gekennzeichnet, um Verwechslungen mit KFG-Zeichen zu vermeiden: '('

nachlesen:

Kastens, Kleine Übung: Modellierung, Abschnitt 6.1

Bedeutung der Produktionen

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

Beispiele:

DeutscherSatz	::=	Subjekt Prädikat Objekt
EinDeutscherSatz		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:



Vorlesung Modellierung WS 2011/12 / Folie 603

Ziele:

Produktionen verstehen

in der Vorlesung:

Erläuterungen der beiden Rollen von Produktionen:

- Definition von Struktur: "besteht aus"
- Definition von Ersetzungen
- Siehe auch Mod-6.5 zur graphischen Darstellung von Produktionen.

nachlesen:

Kastens, Kleine Büning: Modellierung, Abschnitt 6.1

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 $u A v \Rightarrow u x v$

z. B. **Klammern Klammern Liste** \Rightarrow **Klammern (Liste) Liste**
mit Produktion p1

Beliebig viele **Ableitungsschritte nacheinander** angewandt heißen **Ableitung**;
notiert als $u \Rightarrow^* v$

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-6.2 **definiert** geschachtelte Folgen paariger Klammern als **Sprache**:

$$\{ (), (()), ((() ()), (((()) (())), \dots \} \subseteq L(G)$$

Ableitung des Satzes (() ()): S = Klammern
 \Rightarrow (Liste)
 \Rightarrow (Klammern Liste)
 \Rightarrow (Klammern Klammern Liste)
 \Rightarrow (Klammern (Liste) Liste)
 \Rightarrow ((Liste) (Liste) Liste)
 \Rightarrow (() (Liste) Liste)
 \Rightarrow (() () Liste)
 \Rightarrow (() ())

Vorlesung Modellierung WS 2011/12 / Folie 604

Ziele:

Ableitungsbegriff verstehen

in der Vorlesung:

Erläuterungen dazu

- Beispiele für Ableitungen
- Beispiele für Sprachen

nachlesen:

Kastens, Kleine Büning: Modellierung, Abschnitt 6.1

Ableitungsbäume

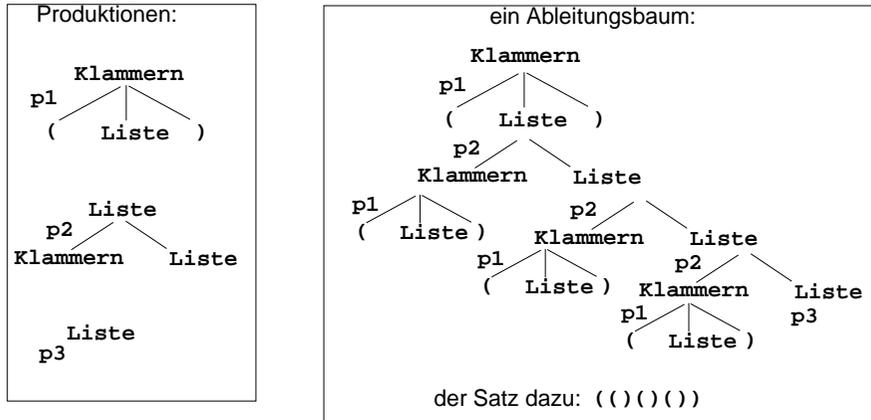
Jede Ableitung kann man als **gewurzelten Baum** darstellen:

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.



Satz zum Baum: Terminale im links-abwärts Durchgang

Ziele:

Ableitungsbaum verstehen

in der Vorlesung:

- Konstruktion des Baumes durch Zusammensetzen von Produktionsanwendungen am "Bastelbogen" zeigen,
- Zusammenhang zum Satz der Sprache

nachlesen:

Kastens, Kleine Büning: Modellierung, Abschnitt 6.1

Beispiel: Ausdrucksgrammatik

p1: Ausdruck ::= Ausdruck BinOpr Ausdruck

p2: Ausdruck ::= Zahl

p3: Ausdruck ::= Bezeichner

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

p5: BinOpr ::= '+'

p6: BinOpr ::= '-'

p7: BinOpr ::= '*'

p8: BinOpr ::= '/'

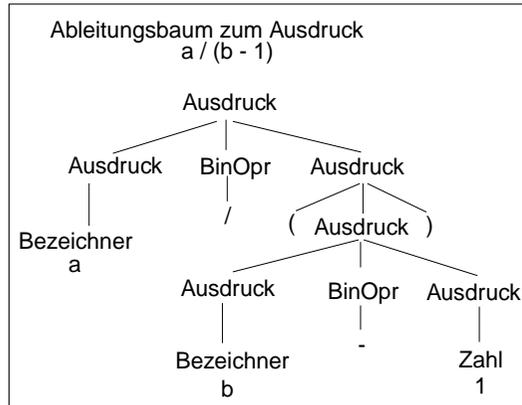
Startsymbol: Ausdruck

Terminale:

T = { Zahl, Bezeichner, (,), +, -, *, / }

Schreibweise der Terminale

Zahl und Bezeichner wird nicht in der KFG definiert.



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

Ziele:

Vollständiges Beispiel sehen

in der Vorlesung:

- Erläuterungen dazu.
- Vergleich mit Kantorowitsch-Bäumen.
- Diese Grammatik ist mehrdeutig: z. B. hat der Satz a+b+c mehrere Ableitungsbäume.

nachlesen:

Kastens, Kleine Büning: Modellierung, Abschnitt 6.1

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 von KFGn:

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

(gleiche Bedeutung wie bei Wertebereichen)

Vorlesung Modellierung WS 2011/12 / Folie 607

Ziele:

HTML-Ausschnitt verstehen

in der Vorlesung:

Erläuterungen

- zum *-Operator (siehe Mod-2.8b),
- zur Struktur von HTML,
- zum Beispiel,
- zur Baumdarstellung

Übungsaufgaben:

Beschreiben Sie die Operationsfolgen zur Bedienung des Getränkeautomaten durch eine KFG.

6.2 Baumstrukturen in XML Übersicht

XML (Extensible Markup Language, dt.: Erweiterbare Auszeichnungssprache)

- seit 1996 vom W3C definiert, in Anlehnung an SGML
- Zweck: Beschreibungen **allgemeiner Strukturen** (nicht nur Web-Dokumente)
- **Meta-Sprache** ("erweiterbar"): Die Notation ist festgelegt (Tags und Attribute, wie in HTML), Für beliebige Zwecke kann **jeweils eine spezielle syntaktische Struktur** definiert werden (DTD) Außerdem gibt es Regeln (XML-Namensräume), um XML-Sprachen in andere **XML-Sprachen zu importieren**
- **XHTML** ist so als XML-Sprache definiert
- Viele **Sprachen sind aus XML abgeleitet**, z.B. SVG, MathML, SMIL, RDF, WML
- **individuelle XML-Sprachen** werden definiert, um strukturierte Daten zu speichern, die von **Software-Werkzeugen geschrieben und gelesen** werden
- XML-Darstellung von strukturierten Daten kann mit verschiedenen Techniken in **HTML transformiert** werden, um sie **formatiert anzuzeigen**: XML+CSS, XML+XSL, SAX-Parser, DOM-Parser

Dieser Abschnitt orientiert sich eng an **SELFHTML** (Stefan Münz), <http://de.selfhtml.org>

Vorlesung Modellierung WS 2011/12 / Folie 607a

Ziele:

Rolle von XML verstehen

in der Vorlesung:

Die Aspekte werden einführend erklärt.

3 elementare Prinzipien

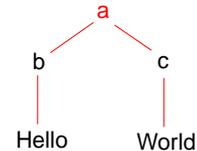
Mod-6.7b

Die XML-Notation basiert auf 3 elementaren Prinzipien:

A: Vollständige Klammerung durch Tags

```
<a>  
  <b>Hello</b>  
  <c>World</c>  
</a>
```

B: Klammerstruktur ist äquivalent zu gewurzeltem Baum



C: Kontextfreie Grammatik definiert Bäume;
eine DTD ist eine KFG

```
a ::= b c  
b ::= PCDATA  
c ::= PCDATA
```

Vorlesung Modellierung WS 2011/12 / Folie 607b

Ziele:

Prinzipien der XML-Notation

in der Vorlesung:

Kurze Erklärung der Prinzipien.

Notation und erste Beispiele

Mod-6.7c

Ein Satz in einer XML-Sprache ist ein Text, der durch **Tags** strukturiert wird.

Tags werden immer in Paaren von **Anfangs- und End-Tag** verwendet:

```
<ort>Paderborn</ort>
```

Anfangs-**Tags** können Attribut-Wert-Paare enthalten:

```
<telefon typ="dienst">05251606686</telefon>
```

Die **Namen von Tags und Attributen** können für die XML-Sprache **frei gewählt** werden.

Mit **Tags** gekennzeichnete Texte können geschachtelt werden.

```
<adressBuch>  
<adresse>  
  <name>  
    <nachname>Mustermann</nachname>  
    <vorname>Max</vorname>  
  </name>  
  <anschrift>  
    <strasse>Hauptstr 42</strasse>  
    <ort>Paderborn</ort>  
    <plz>33098</plz>  
  </anschrift>  
</adresse>  
</adressBuch>
```

(a+b)² in MathML:

```
<msup>  
  <mfenced>  
    <mrow>  
      <mi>a</mi>  
      <mo>+</mo>  
      <mi>b</mi>  
    </mrow>  
  </mfenced>  
  <mn>2</mn>  
</msup>
```

Vorlesung Modellierung WS 2011/12 / Folie 607c

Ziele:

Notation von XML verstehen

in der Vorlesung:

An den Beispielen wird erklärt:

- Tags und Attribute werden für den speziellen Zweck frei erfunden,
- ein Tag-Paar begrenzt ein Element und benennt seine Rolle,
- geschachtelte Strukturen.
- Wir entwerfen eigene Sprachen!!

Ein vollständiges Beispiel

Mod-6.7d

Kennzeichnung des Dokumentes als XML-Datei

Datei mit der Definition der Syntaktischen Struktur dieser XML-Sprache (DTD)

Datei mit Angaben zur Transformation in HTML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE adressBuch SYSTEM "adressBuch.dtd">
<?xml-stylesheet type="text/xsl" href="adressBuch.xsl" ?>
<adressBuch>
<adresse>
  <name>
    <nachname>Mustermann</nachname>
    <vorname>Max</vorname>
  </name>
  <anschrift>
    <strasse>Hauptstr 42</strasse>
    <ort>Paderborn</ort>
    <plz>33098</plz>
  </anschrift>
</adresse>
</adressBuch>
```

Ziele:

Technische Angaben sehen

in der Vorlesung:

Am Beispiel wird erklärt:

- die 3 technischen Angaben,
- XML-Text.

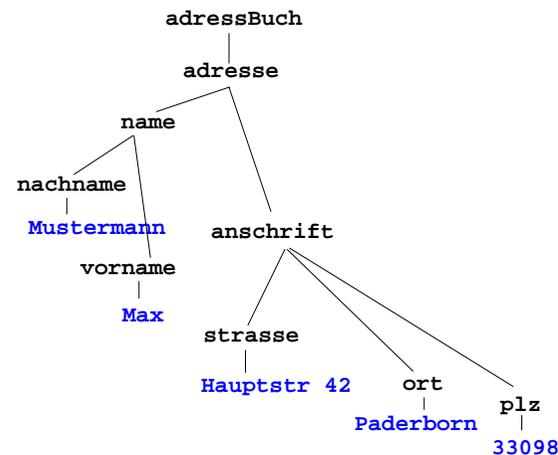
Baumdarstellung von XML-Texten

Mod-6.7e

Jeder XML-Text ist durch Tag-Paare **vollständig geklammert** (wenn er *wohlgeformt* ist).

Deshalb kann er eindeutig **als Baum dargestellt** werden. (Attribute betrachten wir hier nicht)
Wir markieren die inneren Knoten mit den Tag-Namen; die **Blätter** sind die elementaren Texte:

```
<adressBuch>
<adresse>
  <name>
    <nachname>Mustermann
    </nachname>
    <vorname>Max
    </vorname>
  </name>
  <anschrift>
    <strasse>Hauptstr 42
    </strasse>
    <ort>Paderborn</ort>
    <plz>33098</plz>
  </anschrift>
</adresse>
</adressBuch>
```



XML-Werkzeuge können die Baumstruktur eines XML-Textes ohne weiteres ermitteln und ggf. anzeigen.

Vorlesung Modellierung WS 2011/12 / Folie 607e

Ziele:

XML-Text als Baum verstehen

in der Vorlesung:

Am Beispiel wird erklärt:

- vollständige Klammerung durch Tags,
- definiert einen Baum,
- aus dem Baum kann man den Text wiederherstellen

Wohlgeformte XML-Texte

XML-Texte sind **wohlgeformt** (well-formed), wenn sie folgende Regeln erfüllen:

1. Ein Element beginnt mit einem Anfangs-Tag und endet mit einem gleichnamigen End-Tag. Dazwischen steht eine evtl. leere Folge von Elementen und elementaren Texten.
2. Elementare Texte können beliebige Zeichen, aber keine Tags enthalten.
3. ein XML-Text ist ein Element.

wohlgeformt	wohlgeformt	nicht wohlgeformt
<pre><a> <c>1</c> <d>2</d> <e>3</e> </pre>	<pre><a> 1 2 <c>3</c> 4 <d>5</d> <e>6</e> </pre>	<pre><a> <c>1 </c> </pre>

Vorlesung Modellierung WS 2011/12 / Folie 607f

Ziele:

Regeln für wohlgeformte XML-Texte kennenlernen

in der Vorlesung:

Regeln und Beispiele werden erklärt.

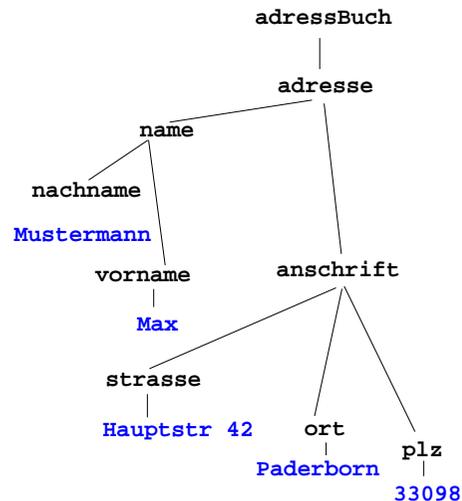
Grammatik definiert die Struktur der XML-Bäume

Mit **kontextfreien Grammatiken (KFG)** kann man **Bäume definieren**.

Folgende KFG definiert korrekt strukturierte Bäume für das Beispiel Adressbuch:

```

adressBuch ::= adresse*
adresse   ::= name anschrift
name      ::= nachname vorname
Anschrift ::= strasse ort plz
nachname  ::= PCDATA
vorname   ::= PCDATA
strasse   ::= PCDATA
ort       ::= PCDATA
plz       ::= PCDATA
    
```



Vorlesung Modellierung WS 2011/12 / Folie 607g

Ziele:

Definition durch KFG verstehen

in der Vorlesung:

Am Beispiel wird erklärt:

- Tag-Namen werden Nichtterminale,
- PCDATA ist das Terminal für die elementaren Texte,
- weiteren Baum skizzieren.

Document Type Definition (DTD) statt KFG

Die Struktur von XML-Bäumen und -Texten wird in der **DTD-Notation** definiert. Ihre Konzepte entsprechen denen von KFGn:

KFG	DTD
adressBuch ::= adresse *	<!ELEMENT adressBuch (adresse)* >
adresse ::= name anschrift	<!ELEMENT adresse (name , anschrift) >
name ::= nachname vorname	<!ELEMENT name (nachname , vorname)>
Anschrift ::= strasse ort plz	<!ELEMENT anschrift (strasse , ort , plz)>
nachname ::= PCDATA	<!ELEMENT nachname (#PCDATA) >
vorname ::= PCDATA	<!ELEMENT vorname (#PCDATA) >
strasse ::= PCDATA	<!ELEMENT strasse (#PCDATA) >
ort ::= PCDATA	<!ELEMENT ort (#PCDATA) >
plz ::= PCDATA	<!ELEMENT plz (#PCDATA) >

weitere Formen von DTD-Produktionen:

(Y) ⁺	nicht-leere Folge
(A B)	Alternative
(A)?	Option
EMPTY	leeres Element

Vorlesung Modellierung WS 2011/12 / Folie 607h

Ziele:

DTD-Notation als KFG verstehen

in der Vorlesung:

Am Beispiel wird erklärt:

- Zuordnung der KFG- zu DTD-Konstrukten,
- Erklärung der weiteren Formen an Beispielen.
- Hinweis: Die DTD-Notation zur Definition von Attributlisten in Anfangs-Tags wird hier nicht beschrieben.