Lektion 10

Die bisher betrachteten kontextfreien Grammatiken sind, wie Sie nun sehen werden, eine spezielle Ausprägung eines allgemeineren Konzepts von Grammatiken, der sog. *Chomsky-Grammatiken*. In dieser Lektion wird nochmal das Thema XML aufgegriffen. Mit sog. *Document Type Definitions* (DTD) lernen Sie eine Methoden kennen, wie anwendungsbezogen Strukturvorgaben für den Aufbau von XML-Dokumenten festgelegt werden können.

6.4 Chomsky-Grammatiken

Grenzen kontextfreier Grammatiken

Kontextfreie Grammatiken sind zwar mächtiger als reguläre Ausdrücke und viele der praktisch relevanten syntaktischen Eigenschaften von Programmiersprachen lassen sich damit beschreiben. Es gibt aber auch Sprachen, die nicht durch kontextfreie Grammatiken dargestellt werden können:

- Ein einfaches Beispiel dafür ist die Sprache $L = \{a^n b^n c^n \mid n > 0\}$, die nicht kontextfrei ist (das Standardbeispiel für eine nicht kontextfreie Sprache).
- Auch viele "kontextsensitive" Eigenschaften in Programmiersprachen lassen sich nicht durch kontextfreie Grammatiken ausdrücken: z.B. dass Variablen deklariert werden müssen, bevor sie verwendet werden dürfen, oder dass Variablen passend typisiert sein müssen.

```
int var1;
...
while (x < 7) {
    ...
    var1 = 2 * x;
    ...
}
```

6.4.1 Typ-i-Grammatiken

Es gibt, als Verallgemeinerung des Konzepts der kontextfreien Grammatiken, verschieden Varianten, die teilweise mächtiger als kontextfreie Grammatiken sind, teilweise auch weniger ausdrucksstark. Allen gemeinsam ist das Konzept der Ersetzungsregeln, dass die linke Seite einer Produktion durch die rechte Seite ersetzt werden kann. Unterschiede gibt es darin, wie die linke und die rechte Seite einer Produktion aufgebaut sein dürfen.

Definition 6.44 - Chomsky-Grammatik

Eine **Chomsky-Grammatik** $G = (N, T, P, S)$ des **Typs** i ($i = 0, 1, 2, 3$) besteht aus

- einer endlichen Menge N nichtterminaler Symbole,
- einer endlichen Menge T terminaler Symbole (wobei $N \cap T = \varnothing$)
- einem Startsymbol $S \in N$

sowie einer endlichen Menge P von **Produktionen des Typs** i ($i = 0, 1, 2, 3$), die folgendermaßen definiert sind:

- **Typ 0** (***uneingeschränkt***): Produktion haben die Form $\boldsymbol{u \rightarrow w}$ mit
  $u \in (N \cup T)^*$
  $w \in (N \cup T)^*$
- **Typ 1** (***kontextsensitiv, monoton***): Produktion haben die Form $\boldsymbol{u \rightarrow w}$ mit
  $u \in (N \cup T)^*$,
  $w \in (N \cup T)^*$
  wobei $\boldsymbol{|u| \leq |w|}$
  Zusätzlich ist die Produktion $S \rightarrow \varepsilon$ erlaubt, sofern S nicht auf der rechten Seite einer Produktion vorkommt.
- **Typ 2** (***kontextfrei***): Produktion haben die Form $\boldsymbol{A \rightarrow w}$ mit
  $A \in N$,
  $w \in (N \cup T)^*$
- **Typ 3** (***rechtslinear***): Produktionen können entweder die Form
  $\boldsymbol{A \rightarrow aB}$
  oder die Form
  $\boldsymbol{A \rightarrow \varepsilon}$
  haben (mit $A, B \in N$, $a \in T$)

Ableitungen in Chomsky-Grammatiken

- Der Begriff der (schrittweisen) Ableitung ist für Chomsky-Grammatiken analog wie bei kontextfreien Grammatik definiert: Die linke Seite einer Produktion kann in einem Ableitungsschritt durch deren rechte Seite ersetzt werden, d.h. ist $\boldsymbol{u \rightarrow w}$ eine Produktion und sind x und y beliebige Folgen von Symbolen, d.h. $y \in (N \cup T)^*$, $y \in (N \cup T)^*$, dann ist ein direkter Ableitungsschritt
  $$\boldsymbol{xuy} \Rightarrow \boldsymbol{xwy}$$
  möglich.
- Die Sprache einer Chomsky-Grammatik ist (genauso wie schon bei den

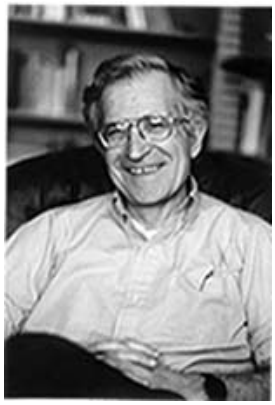
kontextfreien Grammatiken) die Menge aller Wörter aus terminalen Symbolen, die in endlich vielen Schritten aus dem Startsymbol abgeleitet werden können.

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

Anmerkungen 6.45

- Bei Typ-0-Grammatiken können linke und rechte Seite von Produktionen beliebige Folgen aus terminalen und nichtterminalen Symbolen sein.
- Typ-1-Grammatiken werden auch als "*monotone* Grammatiken" bezeichnet. Linke und rechte Seite sind Folgen terminaler und nichtterminaler Symbole, aber die rechte Seite muss mindestens so lange sein wie die linke Seite. Die Länge der Wörter bei Ableitungen ist monoton wachsend, d.h. wird nie kürzer. Damit auch das leere Wort abgeleitet werden kann, ist als Ausnahme die Regel $S \rightarrow \varepsilon$ möglich.
- Typ-2-Grammatiken sind genau die üblichen kontextfreien Grammatiken.
- Typ-3-Grammatiken werden auch als "reguläre Grammatiken" bezeichnet. Die rechte Seite von Produktionen ist entweder leer oder besteht aus einem Terminal und einem Nichtterminal am Ende (es darf auch das gleiche wie auf der linken Seite sein, z.B. A → aA).

  Analog zu *rechtslinearen* Grammatiken können symmetrisch auch *linkslineare* Grammatiken mit Produktionen der Form $A \rightarrow Ba$ definiert werden



Noam Chomsky
(geb. 1928)
amerikanischer Linguist

Folgendes Beispiel zeigt, dass Typ-1-Grammatiken mächtiger als kontextfreie Grammatiken (Chomsky-Typ-2) sind.

Beispiel 6.46 - Typ-1-Grammatik für $L = \{a^n b^n c^n \mid n > 0\}$

- Produktionen (Startsymbol S)

  $S \rightarrow aSBC$
  $S \rightarrow aBC$
  $CB \rightarrow BC$
  $aB \rightarrow ab$
  $bB \rightarrow bb$
  $bC \rightarrow bc$
  $cC \rightarrow cc$

- Ableitung für $a^3b^3c^3$:

$\underline{S} \Rightarrow a\underline{S}BC$
$\Rightarrow aa\underline{S}BCBC$
$\Rightarrow aaaB\underline{C}B\underline{C}BC$
$\Rightarrow aaaBB\underline{C}\underline{C}BC$
$\Rightarrow aaaBB\underline{C}\underline{B}CC$
$\Rightarrow aaa\underline{a}\underline{B}BCCC$
$\Rightarrow aaab\underline{B}BCCC$
$\Rightarrow aaabb\underline{B}CCC$
$\Rightarrow aaabbb\underline{C}CC$
$\Rightarrow aaabbbc\underline{C}C$
$\Rightarrow aaabbbcc\underline{C}$
$\Rightarrow aaabbbccc$

Die Sprache dieser Grammatik ist die nicht kontextfreie Sprache
$\{ a^n b^n c^n \mid n > 0 \}$.

Beispiel 6.47 - Typ-3-Grammatik

- Folgendes ist eine Chomsky-Typ-3-Grammatik:

S	→	aA
A	→	aA
	|	bB
B	→	bB
	|	ε

- Mit dieser Grammatik kann z.B. das Wort **aaabb** abgeleitet werden:

$S \Rightarrow aA \Rightarrow aaA \Rightarrow aaaA \Rightarrow aaabB \Rightarrow aaabbB \Rightarrow aaabb$

Man sieht hier, dass bei Ableitungen von Typ-3-Grammatiken die Symbolfolge immer (bis auf den letzten Schritt) genau ein Nichtterminal am Ende der Folge hat, das dann im nächsten Schritt ersetzt werden kann. Jeder Ableitungsschritt, bei dem keine ε-Produktion verwendet wird, produziert ein weiteres terminales Symbol. Nur dann, wenn eine ε-Produktion angewendet wird, verschwindet das Nichtterminal und es bleibt eine Folge von terminalen Symbolen übrig, so dass der Ableitungsvorgang endet.

- Diese Grammatik erzeugt die Sprache $L_3 = \{ a^n b^k \mid n > 0, k > 0 \}$.

Aufgabe 6.48 - Typ-3-Grammatik

Gegeben ist folgende Chomsky-Typ-3-Grammatik.

S	→	aA
	|	ε
A	→	bB
B	→	cS

Welche Sprache wird durch diese Grammatik erzeugt?

Aufgabe 6.49 - Typ-3-Grammatik

- Definieren Sie eine Chomsky-Typ-3-Grammatik, die die Sprache des regulären Ausdrucks

  (a|b)*c

  generiert.
- Geben Sie eine Ableitung für das Wort **abbac** an.

6.4.2 Die Chomsky-Hierarchie

Der Zusammenhang zwischen *Sprachen* (als Menge von Wörtern) und der Beschreibung durch *Chomsky-Grammatiken vom Typ i* ist wie folgt definiert (analog wie bei kontextfreien Grammatiken):

Definition 6.50 Chomsky-Typ-i-Sprachen

Eine **Sprache ist vom Chomsky-Typ *i*** (*i* = 0, 1, 2, 3), wenn es eine Chomsky-Grammatik *G* vom Typ *i* gibt, die die Sprache definiert.

Beispiel 6.51 - Chomsky-Typ-i-Sprachen

- Die Sprache $L_1 = \{a^n b^n c^n \mid n > 0\}$ ist eine Chomsky-Typ-1-Sprache, da es eine Chomsky-Typ-1-Grammatik dafür gibt, s.o.
- Die Sprache $L_2 = \{a^n b^n \mid n > 0\}$ ist eine Chomsky-Typ-2-Sprache, da es eine kontextfreie Grammatik (Typ-2-Grammatik) dafür gibt:

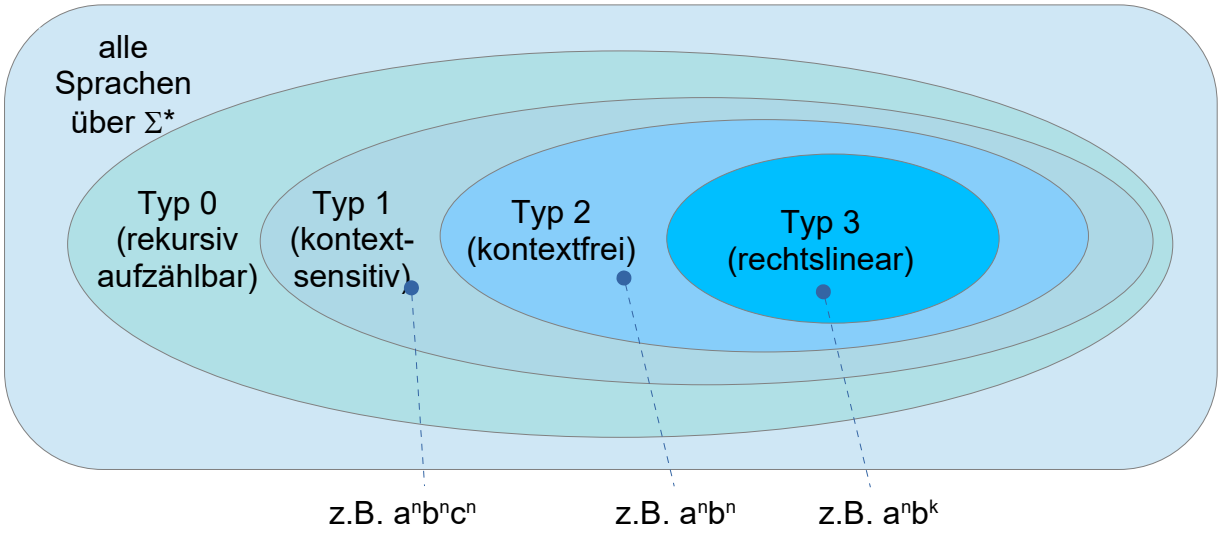
  S → aSb | ab
- Die Sprache $L_3 = \{a^n b^k \mid n > 0, k > 0\}$ (a^+b^+ als regulärer Ausdruck) ist eine Chomsky-Typ-3-Sprache, da die Grammatik aus Beispiel 6.47 sie erzeugt.

Eigenschaft 6.52 - Chomsky-Hierarchie

Die Chomsky-Sprachen vom Typ 0, 1, 2, 3 (über einem gegebenen Alphabet Σ) bilden eine Hierarchie, jede Sprachklasse ist in der nächsten enthalten.

$$\textit{Typ 3} \subset \textit{Typ 2} \subset \textit{Typ 1} \subset \textit{Typ 0} \subset \textit{alle Sprachen}$$

Die Beziehung der Typ-i-Sprachklassen und einige typische Beispielsprachen sind in folgender Abbildung dargestellt.



Anmerkungen

- Für Sprachen, die nicht Typ 0 sind, oder die zwar Typ 0, aber nicht Typ 1 sind, gibt es leider keine leicht verständlichen Beispiele.
- In praktischen Anwendungen spielen nur Typ-2-Sprachen und Typ-3-Sprachen eine Rolle.
- In einem der folgenden Kapitel werden wir sehen, dass die Typ-3-Sprachen mit den Sprachen übereinstimmen, die durch reguläre Ausdrücke beschrieben werden können.

6.5 Exkurs: Strukturdefinition für XML-Bäume

In Kapitel 4.3 haben Sie XML als Austauschformat für strukturierte Daten kennengelernt. Eine Eigenschaft, die *jedes* XML-Dokument erfüllen muss, ist die *Wohlgeformtheit.* Das bedeutet letztendlich nur, dass das Dokument eine baumartige Struktur repräsentieren muss. Werden in einem Anwendungsbereich XML-Dokumente verwendet, haben die Dokument üblicherweise noch anwendungs-spezifische Vorgaben zu erfüllen, wie sie strukturell aufgebaut sein können.

Beispiel 6.53 - XML-Addressbuch-Dokument

```
<adressBuch>
    <adresse>
        <name>
            <vorname>Thomas</vorname>
            <nachname>Teufel</nachname>
        </name>
        <anschrift>
            <strasse>In der Höll 13</strasse>
            <plz>89073</plz>
            <ort>Ulm</ort>
        </anschrift>
    </adresse>
    <adresse>
        <name>
            <vorname>Elvira</vorname>
            <nachname>Engel</nachname>
        </name>
        <anschrift>
            <strasse>Im Himmelreich 42</strasse>
            <plz>09421</plz>
            <ort>Wolkenstein</ort>
        </anschrift>
    </adresse>
</adressBuch>
```

Wenn Sie dieses Beispiel aus Kapitel 4.3 betrachten, erkennen Sie, dass das Dokument offensichtlich nach einem Schema aufgebaut ist:

- Ein Adressbuch enthält eine Folge von Adressen
- Jede Adresse besteht aus einem Namen und einer Anschrift
- Der Name besteht aus Vorname und Nachname
- …

Um derartige anwendungsbezogene Strukturvorgaben für XML-Dokumente definieren zu können, gibt es bei XML mehrere Methoden. Eine davon sind sog. *Document Type Definitions* (DTD).

6.5.1 Document Type Definition (DTD)

DTDs sind formale Beschreibungen für den strukturellen Aufbau von XML-Dokumenten, die Ähnlichkeiten zu EBNF haben.

- Eine DTD-Spezifikation hat die Form

```
<!DOCTYPE wurzelelement [
   … Definition der XML-Elemente …
]>
```

- Jedes XML-Element `<elementname>...</elementname>` wird durch einen Eintrag folgender Form definiert:

  ```
  <!ELEMENT elementname regAusdruck>
  ```

- Der Ausdruck *regAusdruck*, der in der Art wie reguläre Ausdrücke aufgebaut sein kann, beschreibt, welchen Inhalt das XML-Element haben kann. Er kann aus folgenden Teilen gebildet werden:

`#PCDATA`	elementarer Text (*parsed character data*), enthält keine weiteren XML-Elemente
`EMPTY`	leeres Element (ohne Inhalt)
A , *B* , *C*	Sequenz: `A` gefolgt von *B* gefolgt von `C`
A | *B* | *C*	alternativ *A* oder *B* oder *C*
*Y**	(evtl. leere) Folge von *Y*, beliebig oft
Y+	nicht-leere Folge von *Y*, d.h. mindestens einmal
A?	*A* ist optional

Beispiel 6.54 - DTD für Adressbücher

Der Aufbau von Adressbuch-Dokumenten sei durch folgende DTD definiert:

```
<!DOCTYPE addressBuch [
    <!ELEMENT adressBuch (adresse)*>                    (1)
    <!ELEMENT adresse (name, anschrift)>
    <!ELEMENT name  (vorname+, nachname, titel?)>       (2)
    <!ELEMENT nachname (#PCDATA)>                       (3)
    <!ELEMENT vorname (#PCDATA)>
    <!ELEMENT titel (#PCDATA)>
    <!ELEMENT anschrift (strasse, plz, ort)>
    <!ELEMENT strasse (#PCDATA)>
    <!ELEMENT plz (#PCDATA)>
    <!ELEMENT ort (#PCDATA)>
]>
```

Schauen Sie sich einige der gekennzeichneten Elementdefinitionen genauer an:

(1) Ein `addressBuch`-Element kann beliebig viele `adresse`-Elemente enthalten, es kann also auch leer sein.

(2) Ein `name`-Element kann einen oder mehrere `vorname`-Elemente enthalten, genau ein `nachname`-Element und optional kann auch ein `titel`-Element enthalten sein. Die Reihenfolge ist fest, erst Vornamen, dann Nachname, dann ggf. Titel.

(3) `name`, `vorname` und `titel` können nur Text enthalten, keine weiteren XML-Elemente.

Definition 6.55 - Validität von XML-Dokumenten

Ein XML-Dokument heißt **valide** (**gültig**), wenn es die strukturellen Vorgaben der zugeordneten Strukturbeschreibung (DTD) einhält.

Beispiel 6.56 - Validität von XML-Dokumenten

Das oben angegebene Addressbuch-Beispiel 6.53 ist valide bezüglich der Adressbuch-DTD aus Beispiel 6.54.

Aufgabe 6.57 - Validität von XML-Dokumenten

Ist das folgende XML-Dokument valide bezüglich der Adressbuch-DTD aus Beispiel 6.54 ? Geben Sie ggf. alle Fehler an.

```
<adressBuch>
    <adresse>
        <name>
            <vorname>Detlef</vorname>
            <vorname>Dussel</vorname>
            <titel>Dr.</titel>
        </name>
        <anschrift>
            <strasse>Dorfstraße 42</strasse>
            <ort>Dimpflingen</ort>
            <plz>12345</plz>
        </anschrift>
        <plz>89073</plz>
    </adresse>
</adressBuch>
```

6.5.2 DTDs und kontextfreie Grammatiken

Eine DTD ist gewissermaßen eine Grammatik für die erlaubte Baumstruktur der XML-Dokumente. Aus jeder DTD kann direkt eine Grammatik in EBNF für die serialisierte textuelle Form abgeleitet werden, wie an folgendem Beispiel verdeutlicht wird.

Beispiel 6.58 - DTD und abgeleitete EBNF

Geben sind folgende Element-Definitionen aus der DTD:

```
<!DOCTYPE addressBuch [
    <!ELEMENT adressBuch (adresse)*>
    <!ELEMENT adresse (name, anschrift)>
    <!ELEMENT name  (vorname+, nachname, titel?)>
    ...
```

Indem für jedes XML-Element ein entsprechendes nichtterminales Symbol eingeführt wird, können die DTD-Definitionen naheliegend in EBNF übersetzt werden:

AdressBuch → `<adressBuch>` *Adresse** `</adressBuch>`
Adresse → `<adresse>` *Name Anschrift* `</adresse>`
Name → `<name>` *Vorname*$^+$ *Nachname* [*Titel*] `</name>`
…

Das in der DTD angegebene Wurzelelement `adressBuch` legt dann auch das Startsymbol *AdressBuch* der Grammatik fest.

6.5.3 Ausblick: Mehr Details zu DTDs

[nicht klausurrelevant, aber wissenswert für die Praxis] Für die praktische Arbeit mit DTDs ist auch folgendes von Bedeutung:

- Ist ein XML-Dokument entsprechend einer DTD aufgebaut, verweist das XML-Dokument in seinem Kopfteil üblicherweise durch eine DOCTYPE-Deklaration auf diese DTD. Die DTD kann z.B. als separate Datei auf dem System vorhanden sein, meist ist sie aber über eine URL angegeben und die DTD-Definition steht irgendwo im Internet.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE adressBuch SYSTEM "adressBuch.dtd">
…
```

- Mit einer DTD können auch die *Attribute* von XML-Elementen definiert werden:

```
<ELEMENT buch (title, autor+, info)>

<ATTLIST buch
    isbn CDATA #REQUIRED
    jahr CDATA "2020"   >
```

  Hier wird angegeben, dass ein Buch ein obligatorisches `isbn`-Attribut hat und dass es ein Attribut `jahr` mit Default-Wert "2020" geben kann.

Alternativen zu DTDs

Außer DTDs gibt es noch andere Möglichkeiten, um die Struktur von XML-Dokumenten zu definieren. DTDs bieten z.B. keine Möglichkeit, den Textinhalt eines XML-Elements genauer zu spezifizieren. Eine neuere, ausdrucksstärkere Beschreibungsform sind sog. **XML-Schema-Definitionen (XSD)**. In XSD ist es beispielsweise möglich, für den textuellen Inhalt eines XML-Elements zu spezifizieren, wie er aufgebaut sein kann, z.B. ob es eine Zahl sein muss, eine Datums- oder Zeitangabe oder ob per regulärem Ausdruck spezifizierter String.

Fazit zu Lektion 10

Diese Fragen sollten Sie nun beantworten können

- Was sind Chomsky-Typ-i-Grammatiken?
- Was sagt die Chomsky-Hierarchie aus?
- Was sind DTDs für XML-Dokumente? Welcher Zusammenhang besteht zwischen DTDs und Grammatiken?