Stand: 12. Oktober 2000

7 XML und Scriptprogrammierung

7.8 XML-Syntax

7.8.1 EBNF

Eine *Syntax* ist ein Regelwerk, mittels dessen korrekte sprachliche Ausdrücke konstruiert werden können, und mit dem festgestellt werden kann, ob ein sprachlicher Ausdruck korrekt ist oder nicht. Eine Syntax kann ihrerseits eine Sprache definieren, d.h. syntaktisches Regelwerk für eine Syntax sein. Eine solche Sprache, die eine Sprache definiert, wird als *Metasprache* bezeichnet. Zur Festlegung der Syntax einer Computersprache, z.B. einer Programmier- oder Markupsprache, werden *Produktionen* verwendet, d.h. *Regeln*, nach denen die Bestandteile eines Strings (oder eines Wortes oder eines Satzes oder einer anderen Konstruktion) durch andere Strings (oder ...) ersetzt werden können. Die Menge aller syntaktischer Regeln beschreibt eine *Sprache*.
Eine einfache Syntax ist die nach Backus und Naur benannte EBNF, die Extended Backus-Naur Form, die in derjenigen Variante vorgestellt wird, mittels derer die Syntax von XML definiert ist. EBNF selbst ist eine Metasprache, die hier der Definition der Metasprache XML dient.
Die Buchstaben `A`, `B`, `C` etc. sollen *Symbole* für Strings (oder Wörter, Sätze etc.) sein. Dann wird die *Regel* für die *Verkettung* zweier Symbole `A` und `B` zu einem neuen Symbol `C` beschrieben durch

```
C ::= A B .
```

In ähnlicher Weise wird mittels

D ::= A | B | C

eine *Auswahl* aus A, B und C getroffen und das Resultat mit dem Symbol D bezeichnet (dafür steht das Symbol ::=). Das Symbol D kann also entweder den Wert A oder B oder C haben (exklusives Oder). In Mengenschreibweise: D ∈ {A, B, C}.
Bei komplexen Regeln kann in Ausdrücken auf die übliche Weise *geklammert* werden. Damit steht bereits das Grundgerüst für die Formulierung syntaktischer Regeln fest. Im Beispiel:

```
D ::= A (B | C) (E | F)          d.h.
D ∈ {ABE, ABF, ACE, ACF}         in Mengenschreibweise
```

Verwendet wird außerdem noch eine „Differenzbildung“ zwischen zwei Symbolen,

C ::= A - B ,

d.h. C besteht aus all den Bestandteilen des Symbols A, die nicht in B enthalten sind, in Kurzform also „A ohne B“.
Hinzu kommen einige Vereinbarungen über effizientere Schreibweisen. So sind bislang Optionen und Wiederholungen von Symbolen nur umständlich darstellbar, z.B.:

```
D ::= A | AA | AAA | AAAA ...        d.h.
D ∈ {A, AA, AAA, AAAA ...}           in Mengenschreibweise
```

Zu solchen Zwecken werden Symbole definiert, die als Postfix an das Symbol geheftet werden. Das gleiche Beispiel schreibt sich dann mit dem *Pluszeichen* als Postfix so:

D ::= A+ statt z.B. `D ::= ∅ | A | AA | AAA | ...`

Ein *Fragezeichen* als Postfix bezeichnet das Symbol als optional, es kommt also entweder einmal oder es kommt gar nicht vor:

D ::= A? statt z.B. `D ::= ∅ | A`
`D ∈ {∅, A}`

Der *Asteriskus* entspricht dem postfixen Pluszeichen, das voranstehende Symbol ist aber darüber hinaus optional (d.h. A* und A+? meinen das Gleiche).

D ::= A* statt z.B. `D ::= A | AA | AAA | ...`
`D ∈ {∅, A, AA, AAA, ...}`

Zusammen entspricht dies etwa dem *Erweiterten Backus-Naur-Formalismus* (EBNF, Extended Backus-Naur Form), dem noch einige wenige Konventionen über String- und Zeichenliterale hinzugefügt werden, um so als Grundlage für die Definition von XML zu dienen.
Es ist nicht unüblich, zwischen terminalen und nicht-terminalen Symbolen zu unterscheiden. Das Adjektiv „terminal“ meint „am Ende stehend“, womit in EBNF das Ende einer Kette von Regeln bezeichnet wird. Rechts neben ::= stehende Symbole für weitere Regeln sind zwangsläufig nicht-terminal, während String- oder Zeichenliterale immer terminal sind.
Zeichen-/Zahlenliterale sind insbesondere:

`#xN`	ist das Literal für eine positive ganze Zahl in Hexadezimalnotation. N steht für eine oder mehrere so notierte Stellen, z.B. `#xf`, `#x300f` etc.
`[...]`	ist das Symbol für eines der Zeichen, die in der Klammer stehen und
`[^...]`	das Symbol für eines der Zeichen, die *nicht* in der Klammer enthalten sind.

Einige Beispiele:

`[abc]`	Eines der Zeichen a, b, oder c.
`[^abc]`	Ein Zeichen, das weder a noch b noch c ist.
`[a-bA-B]`	Einer der Klein- oder Großbuchstaben von a bis z.
`[^a-bA-B]`	Ein Zeichen, das weder ein Klein- noch ein Großbuschstabeist.
`[^#xN#xM#xL]`	Ein beliebiges Zeichen, aber nicht `#xN`, `#xM` und `#xL`.
`[#xN-#xM]`	Ein Zeichen, das einer Zahl zwischen #xN und #xM entspricht.
`[^#xN-#xM]`	Ein beliebiges Zeichen, aber keine der Zahlen zwischen #xN und #xM.

Stringliterale schreiben sich wie folgt:

```
"dies ist ein Stringliteral"
'und dies ist auch eines'
```

Beide Notierungen sind gleichwertig, können also alternativ verwendet werden. Der Grund für die beiden Formen ist, daß ein "..."-String auch ' und ein '...'-String " enthalten kann, ohne Zuflucht zu Sonderschreibweisen wie `\'` oder `\"` suchen zu müssen:

```
"dies ist 'ein String' in einem String"
'dies ist auch "ein String" im String'
```


7.8.2 Die XML-Syntax

EBNF ist eine so genannte kontextfreie Syntax, d.h. sie hat Beschränkungen in der Beschreibungsfähigkeit von XML. Beispielsweise ist der Doppelpunkt gemäß EBNF ein erlaubtes Zeichen in XML-Namen, tatsächlich ist er gebunden an Namensraumbezeichnungen und wird in anderen Zusammenhängen in der Regel als fehlerhaft bemängelt. Ebenso kann EBNF nicht ausdrücken, daß zusammengehörige Start- und Endemarke den gleichen Namen beinhalten müssen, usw. Um solche Sachverhalte für XML festzuhalten, sind die Regeln im Bedarfsfall wie folgt kommentiert:

`[WFC]`	*Well-Formedness Constraint*, d.h. Regeln für die Wohlgeformtheit eines Ausdruckes. Verletzungen solcher Regeln sind kritische Fehler.
`[VC]`	*Validity Constraint*, also Zusatzerfordernisse für die Validität. Verstöße müssen validierenden XML-Prozessoren (Parsern) auf Anforderung hin gemeldet werden.

Jede syntaktische Regel hat die Schreibweise

```
symbol ::= ausdruck .
```

Symbole mit großgeschriebenem ersten Buchstaben bezeichnen endliche Regelfolgen, an deren Ende nur noch terminale Symbole stehen (Text- und Zeichenliterale). Dies korrespondiert mit dem Standard, in dem es heißt, daß ein solches Symbol durch einen regulären Ausdruck definiert sei. Ein kleingeschriebener erster Buchstabe dagegen signalisiert, daß die Regel grundsätzlich nicht endlich ist, d.h. Rekursionen bzw. Zyklen enthält.

Anmerkung: Alle Sprachterme wurden per „copy-and-paste“ direkt aus dem Standard entnommen. Jeder Term ist mit der laufenden Nummer aus versehen. Zusätzlich ist jeder Term ergänzt durch die Verweise auf diejenigen Terme, in denen sie verwendet werden.

Außerdem gibt es eine Liste mit Querverweisen (7.8.3), in der für jeden Term sein Vorkommen in anderen Termen aufgeführt ist.

```
[01]      document ::=                                    22, 39, 27
                   prolog element Misc*
[02]          Char ::=
                   #x9 | #xA | #xD | [#x20-#xD7FF]
                   | [#xE000-#xFFFD] | [#x10000-#x10FFFF]
[03]             S ::=
                   (#x20 | #x9 | #xD | #xA)+
[04]      NameChar ::=                          84, 88, 87, 89
                   Letter | Digit | '.' | '-' | '_' | ':'
                   | CombiningChar | Extender
[05]          Name ::=                                  84, 04
                   (Letter | '_' | ':') (NameChar)*
[06]         Names ::=                                  05, 03
                   Name (S Name)*
[07]       Nmtoken ::=                                      04
                   (NameChar)+
[08]      Nmtokens ::=                                  07, 03
                   Nmtoken (S Nmtoken)*
[09]   EntityValue ::=                                  69, 67
                   '"' ([^%&"] | PEReference | Reference)* '"'
                   |  "'" ([^%&'] | PEReference | Reference)* "'"
[10]      AttValue ::=                                       67
                   '"' ([^<&"] | Reference)* '"'   |  "'" ([^<&']
                   | Reference)* "'"
[11] SystemLiteral ::=
                   ('"' [^"]* '"') | ("'" [^']* "'")
[12] PubidLiteral ::=                                      13
                   '"' PubidChar* '"' | "'" (PubidChar - "'")* "'"
[13]     PubidChar ::=
                   #x20 | #xD | #xA | [a-zA-Z0-9]
                   | [-'()+,./:=?;!*#@$_%]
[14]      CharData ::=
                   [^<&]* - ([^<&]* ']]>' [^<&]*)
[15]       Comment ::=                                    '02
                   '<!--' ((Char - '-') | ('-' (Char - '-')))* '-->'
[16]            PI ::=                                 17, 03, 02
                   '<?' PITarget (S (Char* - (Char* '?>' Char*)))?
                   '?>'
[17]      PITarget ::=                                      05
                   Name - (('X' | 'x') ('M' | 'm') ('L' | 'l'))
[18]        CDSect ::=                                  19, 20, 21
                   CDStart CData CDEnd
[19]       CDStart ::=
                   '<![CDATA['
[20]         CData ::=                                       02
                   (Char* - (Char* ']]>' Char*))
[21]         CDEnd ::=
                   ']]>'
[22]        prolog ::=                                  23, 27, 28
                   XMLDecl? Misc* (doctypedecl Misc*)?
```

```
[23]      XMLDecl ::=                                    24, 80, 32, 03
                  '<?xml' VersionInfo EncodingDecl? SDDecl? S? '?>
[24]  VersionInfo ::=                                       03, 25, 26
                  S 'version' Eq (' VersionNum ' | " VersionNum ")
[25]           Eq ::=                                             03
                  S? '=' S?
[26]   VersionNum ::=
                  ([a-zA-Z0-9_.:] | '-')+
[27]         Misc ::=                                    15, 16, 03
                  Comment | PI |  S
[28]  doctypedecl ::=                              03, 05, 75, 29, 69
                  '<!DOCTYPE' S Name (S  ExternalID)? S?
                  ('[' (markupdecl | PEReference | S)* ']' S?)? '>'
[29]   markupdecl ::=                          45, 52, 70, 82, 16, 15
                  elementdecl | AttlistDecl | EntityDecl
                  | NotationDecl | PI | Comment
                  [ VC: Proper Declaration/PE Nesting ]
                  [ WFC: PEs in Internal Subset ]
[30]    extSubset ::=                                       77, 31
                  TextDecl? extSubsetDecl
[31] extSubsetDecl ::=                                 29, 61, 69, 03
                  ( markupdecl | conditionalSect | PEReference
                  | S )*
[32]       SDDecl ::=                                       03, 25
                  S 'standalone' Eq (("'" ('yes' | 'no') "'")
                  | ('"' ('yes' | 'no') '"'))
[33]   LanguageID ::=                                       34, 38
                  Langcode ('-' Subcode)*
[34]     Langcode ::=                                 35, 36, 37
                  ISO639Code |  IanaCode |  UserCode
[35]   ISO639Code ::=
                  ([a-z] | [A-Z]) ([a-z] | [A-Z])
[36]     IanaCode ::=
                  ('i' | 'I') '-' ([a-z] | [A-Z])+
[37]     UserCode ::=
                  ('x' | 'X') '-' ([a-z] | [A-Z])+
[38]      Subcode ::=
                  ([a-z] | [A-Z])+
[39]      element ::=                                  44, 40, 43, 42
                  EmptyElemTag | STag content ETag
                  [ WFC: Element Type Match ] [ VC: Element Valid ]
[40]         STag ::=                                       05, 03, 41
                  '<' Name (S Attribute)*S? '>'
                  [ WFC: Unique Att Spec ]
[41]    Attribute ::=                                       05, 25, 10
                  Name Eq AttValue
                  [ VC: Attribute Value Type ]
                  [ WFC: No External Entity References ]
                  [ WFC: No < in Attribute Values ]
[42]         ETag ::=                                             05, 03
                  '</' Name S? '>'
```

```
[43]      content ::=                          39, 14, 67, 18, 16, 15
                  (element | CharData | Reference | CDSect
                  | PI | Comment)*
[44] EmptyElemTag ::=                                  05, 03, 41
                  '<' Name (S Attribute)* S? '/>'
                  [ WFC: Unique Att Spec ]
[45]  elementdecl ::=                                  03, 05, 46
                  '<!ELEMENT' S Name S contentspec S? '>'
                  [ VC: Unique Element Type Declaration ]
[46]  contentspec ::=                                        51, 47
                  'EMPTY' | 'ANY' | Mixed | children
[47]     children ::=                                        59, 50
                  (choice | seq) ('?' | '*' | '+')?
[48]           cp ::=                                  05, 49, 50
                  (Name | choice | seq) ('?' | '*' | '+')?
[49]       choice ::=                                        03, 48
                  '(' S? cp ( S? '|' S? cp )* S? ')'
                  [ VC: Proper Group/PE Nesting ]
[50]          seq ::=                                        03, 48
                  '(' S? cp ( S? ',' S? cp )* S? ')'
                  [ VC: Proper Group/PE Nesting ]
[51]        Mixed ::=                                        03, 05
                  '(' S? '#PCDATA' (S? '|' S? Name)* S? ')*'
                  | '(' S? '#PCDATA' S? ')'
                  [ VC: Proper Group/PE Nesting ]
                  [ VC: No Duplicate Types ]
[52]  AttlistDecl ::=                                  03, 05, 53
                  '<!ATTLIST' S Name AttDef* S? '>'
[53]       AttDef ::=                               03, 05, 54, 60
                  S Name S AttType S DefaultDecl
[54]      AttType ::=                                  55, 56, 57
                  StringType | TokenizedType | EnumeratedType
[55]   StringType ::=
                  'CDATA'
[56] TokenizedType ::=
                  'ID' [ VC: ID ] [ VC: One ID per Element Type ]
                  [ VC: ID Attribute Default ]
                  | 'IDREF' [ VC: IDREF ]
                  | 'IDREFS' [ VC: IDREF ]
                  | 'ENTITY' [ VC: Entity Name ]
                  | 'ENTITIES' [ VC: Entity Name ]
                  | 'NMTOKEN' [ VC: Name Token ]
                  | 'NMTOKENS' [ VC: Name Token ]
[57] EnumeratedType ::=                                      58, 59
                  NotationType | Enumeration
[58] NotationType ::=                                        05, 03
                  'NOTATION' S '(' S? Name (S? '|' S? Name)* S? ')'
                  [ VC: Notation Attributes ]
[59]  Enumeration ::=                                        03, 07
                  '(' S? Nmtoken (S? '|' S? Nmtoken)* S? ')'
```

```
[60]  DefaultDecl ::=                                              10
                  '#REQUIRED' | '#IMPLIED' | (('#FIXED' S)?
                  AttValue) [ VC: Required Attribute ]
                  [ VC: Attribute Default Legal ]
                  [ WFC: No < in Attribute Values ]
                  [ VC: Fixed Attribute Default ]
[61] conditionalSect ::=                                       62, 63
                  includeSect | ignoreSect
[62]  includeSect ::=                                          03, 31
                  '<![' S? 'INCLUDE' S? '[' extSubsetDecl ']]>'
[63]   ignoreSect ::=                                          03, 64
                  '<![' S? 'IGNORE' S? '['
                  ignoreSectContents* ']]>'
[64] ignoreSectContents ::=                                    65, 64
                  Ignore ('<![' ignoreSectContents ']]>'
                  Ignore)*
[65]       Ignore ::=                                          02
                  Char* - (Char* ('<![' | ']]>') Char*)
[66]      CharRef ::=
                  '&#' [0-9]+ ';' | '&#x' [0-9a-fA-F]+ ';'
                  [ WFC: Legal Character ]
[67]    Reference ::=                                          68, 66
                  EntityRef | CharRef
[68]    EntityRef ::=                                          05
                  '&' Name ';' [ WFC: Entity Declared ]
                  [ VC: Entity Declared ] [ WFC: Parsed Entity ]
                  [ WFC: No Recursion ]
[69]  PEReference ::=                                          05
                  '%' Name ';' [ VC: Entity Declared ]
                  [ WFC: No Recursion ] [ WFC: In DTD ]
[70]   EntityDecl ::=                                          71, 72
                  GEDecl | PEDecl
[71]       GEDecl ::=                                    03, 05, 73
                  '<!ENTITY' S Name S EntityDef S? '>'
[72]       PEDecl ::=                                    03, 05, 74
                  '<!ENTITY' S '%' S Name S PEDef S? '>'
[73]    EntityDef ::=                                    09, 75, 76
                  EntityValue | (ExternalID NDataDecl?)
[74]        PEDef ::=                                          09, 75
                  EntityValue | ExternalID
[75]   ExternalID ::=                                    03, 11, 12
                  'SYSTEM' S SystemLiteral | 'PUBLIC' S
                  PubidLiteral S SystemLiteral
[76]    NDataDecl ::=                                          03, 05
                  S 'NDATA' S Name [ VC: Notation Declared ]
[77]     TextDecl ::=                                    24, 80, 03
                  '<?xml' VersionInfo? EncodingDecl S? '?>'
[78] extParsedEnt ::=                                          77, 43
                  TextDecl? content
[79]        extPE ::=                                          77, 31
                  TextDecl? extSubsetDecl
```

```
[80] EncodingDecl ::=                                    03, 25, 81
                  S 'encoding' Eq ('"' EncName '"' |  "'" EncName
                  "'" )
[81]      EncName ::=
                  [A-Za-z] ([A-Za-z0-9._] | '-')*   /* Encoding
                  name contains only Latin characters */
[82] NotationDecl ::=                                03, 05, 75, 83
                  '<!NOTATION' S Name S (ExternalID |  PublicID)
                  S? '>'
[83]     PublicID ::=                                        03, 12
                  'PUBLIC' S PubidLiteral
[84]       Letter ::=                                        85, 86
                  BaseChar | Ideographic
[85]     BaseChar ::=
                    [#x0041-#x005A] | [#x0061-#x007A]
                  | [#x00C0-#x00D6] | [#x00D8-#x00F6]
                  | [#x00F8-#x00FF] | [#x0100-#x0131]
                  | [#x0134-#x013E] | [#x0141-#x0148]
                  | [#x014A-#x017E] | [#x0180-#x01C3]
                  | [#x01CD-#x01F0] | [#x01F4-#x01F5]
                  | [#x01FA-#x0217] | [#x0250-#x02A8]
                  | [#x02BB-#x02C1] | #x0386 | [#x0388-#x038A]
                  | #x038C | [#x038E-#x03A1] | [#x03A3-#x03CE]
                  | [#x03D0-#x03D6] | #x03DA | #x03DC | #x03DE
                  | #x03E0 | [#x03E2-#x03F3] | [#x0401-#x040C]
                  | [#x040E-#x044F] | [#x0451-#x045C]
                  | [#x045E-#x0481] | [#x0490-#x04C4]
                  | [#x04C7-#x04C8] | [#x04CB-#x04CC]
                  | [#x04D0-#x04EB] | [#x04EE-#x04F5]
                  | [#x04F8-#x04F9] | [#x0531-#x0556] | #x0559
                  | [#x0561-#x0586] | [#x05D0-#x05EA]
                  | [#x05F0-#x05F2] | [#x0621-#x063A]
                  | [#x0641-#x064A] | [#x0671-#x06B7]
                  | [#x06BA-#x06BE] | [#x06C0-#x06CE]
                  | [#x06D0-#x06D3] | #x06D5 | [#x06E5-#x06E6]
                  | [#x0905-#x0939] | #x093D | [#x0958-#x0961]
                  | [#x0985-#x098C] | [#x098F-#x0990]
                  | [#x0993-#x09A8] | [#x09AA-#x09B0] | #x09B2
                  | [#x09B6-#x09B9] | [#x09DC-#x09DD]
                  | [#x09DF-#x09E1] | [#x09F0-#x09F1]
                  | [#x0A05-#x0A0A] | [#x0A0F-#x0A10]
                  | [#x0A13-#x0A28] | [#x0A2A-#x0A30]
                  | [#x0A32-#x0A33] | [#x0A35-#x0A36]
                  | [#x0A38-#x0A39] | [#x0A59-#x0A5C] | #x0A5E
                  | [#x0A72-#x0A74] | [#x0A85-#x0A8B] | #x0A8D
                  | [#x0A8F-#x0A91] | [#x0A93-#x0AA8]
                  | [#x0AAA-#x0AB0] | [#x0AB2-#x0AB3]
                  | [#x0AB5-#x0AB9] | #x0ABD | #x0AE0
                  | [#x0B05-#x0B0C] | [#x0B0F-#x0B10]
                  | [#x0B13-#x0B28] | [#x0B2A-#x0B30]
                  | [#x0B32-#x0B33] | [#x0B36-#x0B39] | #x0B3D
                  | [#x0B5C-#x0B5D] | [#x0B5F-#x0B61]
                  | [#x0B85-#x0B8A] | [#x0B8E-#x0B90]
```

```
                    | [#x0B92-#x0B95] | [#x0B99-#x0B9A] | #x0B9C
                    | [#x0B9E-#x0B9F] | [#x0BA3-#x0BA4]
                    | [#x0BA8-#x0BAA] | [#x0BAE-#x0BB5]
                    | [#x0BB7-#x0BB9] | [#x0C05-#x0C0C]
                    | [#x0C0E-#x0C10] | [#x0C12-#x0C28]
                    | [#x0C2A-#x0C33] | [#x0C35-#x0C39]
                    | [#x0C60-#x0C61] | [#x0C85-#x0C8C]
                    | [#x0C8E-#x0C90] | [#x0C92-#x0CA8]
                    | [#x0CAA-#x0CB3] | [#x0CB5-#x0CB9] | #x0CDE
                    | [#x0CE0-#x0CE1] | [#x0D05-#x0D0C]
                    | [#x0D0E-#x0D10] | [#x0D12-#x0D28]
                    | [#x0D2A-#x0D39] | [#x0D60-#x0D61]
                    | [#x0E01-#x0E2E] | #x0E30 | [#x0E32-#x0E33]
                    | [#x0E40-#x0E45] | [#x0E81-#x0E82] | #x0E84
                    | [#x0E87-#x0E88] | #x0E8A | #x0E8D
                    | [#x0E94-#x0E97] | [#x0E99-#x0E9F]
                    | [#x0EA1-#x0EA3] | #x0EA5 | #x0EA7
                    | [#x0EAA-#x0EAB] | [#x0EAD-#x0EAE] | #x0EB0
                    | [#x0EB2-#x0EB3] | #x0EBD | [#x0EC0-#x0EC4]
                    | [#x0F40-#x0F47] | [#x0F49-#x0F69]
                    | [#x10A0-#x10C5] | [#x10D0-#x10F6] | #x1100
                    | [#x1102-#x1103] | [#x1105-#x1107] | #x1109
                    | [#x110B-#x110C] | [#x110E-#x1112] | #x113C
                    | #x113E | #x1140 | #x114C | #x114E | #x1150
                    | [#x1154-#x1155] | #x1159 | [#x115F-#x1161]
                    | #x1163 | #x1165 | #x1167 | #x1169
                    | [#x116D-#x116E] | [#x1172-#x1173] | #x1175
                    | #x119E | #x11A8 | #x11AB | [#x11AE-#x11AF]
                    | [#x11B7-#x11B8] | #x11BA | [#x11BC-#x11C2]
                    | #x11EB | #x11F0 | #x11F9 | [#x1E00-#x1E9B]
                    | [#x1EA0-#x1EF9] | [#x1F00-#x1F15]
                    | [#x1F18-#x1F1D] | [#x1F20-#x1F45]
                    | [#x1F48-#x1F4D] | [#x1F50-#x1F57] | #x1F59
                    | #x1F5B | #x1F5D | [#x1F5F-#x1F7D]
                    | [#x1F80-#x1FB4] | [#x1FB6-#x1FBC] | #x1FBE
                    | [#x1FC2-#x1FC4] | [#x1FC6-#x1FCC]
                    | [#x1FD0-#x1FD3] | [#x1FD6-#x1FDB]
                    | [#x1FE0-#x1FEC] | [#x1FF2-#x1FF4]
                    | [#x1FF6-#x1FFC] | #x2126 | [#x212A-#x212B]
                    | #x212E | [#x2180-#x2182] | [#x3041-#x3094]
                    | [#x30A1-#x30FA] | [#x3105-#x312C]
                    | [#xAC00-#xD7A3]
[86]  Ideographic ::=
                    [#x4E00-#x9FA5] | #x3007 | [#x3021-#x3029]
[87]CombiningChar ::=
                    [#x0300-#x0345] | [#x0360-#x0361]
                    | [#x0483-#x0486] | [#x0591-#x05A1]
                    | [#x05A3-#x05B9] | [#x05BB-#x05BD] | #x05BF
                    | [#x05C1-#x05C2] | #x05C4 | [#x064B-#x0652]
                    | #x0670 | [#x06D6-#x06DC] | [#x06DD-#x06DF]
                    | [#x06E0-#x06E4] | [#x06E7-#x06E8]
                    | [#x06EA-#x06ED] | [#x0901-#x0903] | #x093C
                    | [#x093E-#x094C] | #x094D | [#x0951-#x0954]
```

```
                    | [#x0962-#x0963] | [#x0981-#x0983] | #x09BC
                    | #x09BE | #x09BF | [#x09C0-#x09C4]
                    | [#x09C7-#x09C8] | [#x09CB-#x09CD] | #x09D7
                    | [#x09E2-#x09E3] | #x0A02 | #x0A3C | #x0A3E
                    | #x0A3F | [#x0A40-#x0A42] | [#x0A47-#x0A48]
                    | [#x0A4B-#x0A4D] | [#x0A70-#x0A71]
                    | [#x0A81-#x0A83] | #x0ABC | [#x0ABE-#x0AC5]
                    | [#x0AC7-#x0AC9] | [#x0ACB-#x0ACD]
                    | [#x0B01-#x0B03] | #x0B3C | [#x0B3E-#x0B43]
                    | [#x0B47-#x0B48] | [#x0B4B-#x0B4D]
                    | [#x0B56-#x0B57] | [#x0B82-#x0B83]
                    | [#x0BBE-#x0BC2] | [#x0BC6-#x0BC8]
                    | [#x0BCA-#x0BCD] | #x0BD7 | [#x0C01-#x0C03]
                    | [#x0C3E-#x0C44] | [#x0C46-#x0C48]
                    | [#x0C4A-#x0C4D] | [#x0C55-#x0C56]
                    | [#x0C82-#x0C83] | [#x0CBE-#x0CC4]
                    | [#x0CC6-#x0CC8] | [#x0CCA-#x0CCD]
                    | [#x0CD5-#x0CD6] | [#x0D02-#x0D03]
                    | [#x0D3E-#x0D43] | [#x0D46-#x0D48]
                    | [#x0D4A-#x0D4D] | #x0D57 | #x0E31
                    | [#x0E34-#x0E3A] | [#x0E47-#x0E4E] | #x0EB1
                    | [#x0EB4-#x0EB9] | [#x0EBB-#x0EBC]
                    | [#x0EC8-#x0ECD] | [#x0F18-#x0F19] | #x0F35
                    | #x0F37 | #x0F39 | #x0F3E | #x0F3F
                    | [#x0F71-#x0F84] | [#x0F86-#x0F8B]
                    | [#x0F90-#x0F95] | #x0F97 | [#x0F99-#x0FAD]
                    | [#x0FB1-#x0FB7] | #x0FB9 | [#x20D0-#x20DC]
                    | #x20E1 | [#x302A-#x302F] | #x3099 | #x309A
[88]        Digit ::=
                    [#x0030-#x0039] | [#x0660-#x0669]
                    | [#x06F0-#x06F9] | [#x0966-#x096F]
                    | [#x09E6-#x09EF] | [#x0A66-#x0A6F]
                    | [#x0AE6-#x0AEF] | [#x0B66-#x0B6F]
                    | [#x0BE7-#x0BEF] | [#x0C66-#x0C6F]
                    | [#x0CE6-#x0CEF] | [#x0D66-#x0D6F]
                    | [#x0E50-#x0E59] | [#x0ED0-#x0ED9]
                    | [#x0F20-#x0F29]
[89]     Extender ::=
                   #x00B7 | #x02D0 | #x02D1 | #x0387 | #x0640
                   | #x0E46 | #x0EC6 | #x3005 | [#x3031-#x3035]
                   | [#x309D-#x309E] | [#x30FC-#x30FE]
```

7.8.3 Querverweise

Der Abschnitt enthält ein alphabetisches Verzeichnis aller Terme bzw. Symbole mit Querverweisen. Vor jedem Symbol ist die Nummer der zugehörigen Regel in eckiger Klammer festgehalten, und rechts davon sind die Nummern aller Regeln aufgeführt, in denen das Symbol auf der rechten Seite von `::=` vorkommt. Beispiel:

```
[43]    content             39, 78
```

Symbol `[43]`, `content`, kommt in den Regeln `39` (`element`) und `78` (`extParseEnt`) vor.


```
[53]    AttDef              52
[52]    AttlistDecl         29
[41]    Attribute           40, 44
[54]    AttType             53
[10]    AttValue            41, 60
[85]    BaseChar            84
[20]    CData               18
[21]    CDEnd               18
[18]    CDSect              43
[19]    CDStart             18
[02]    Char
[14]    CharData            43
[66]    CharRef             67
[47]    children            46
[49]    choice              48
[15]    Comment             27, 29, 43
[48]    cp                  49, 50
[87]    CombiningChar       04
[61]    conditionalSect     31
[43]    content             39, 78
[46]    contentspec         45
[88]    Digit               04
[60]    DefaultDecl         53
[28]    doctypedecl         22
[01]    document            15, 16, 20, 65
[39]    element             01, 43
[45]    elementdecl         29
[44]    EmptyElemTag        39
[81]    EncName             80
[80]    EncodingDecl        23, 77
[70]    EntityDecl          29
[73]    EntityDef           71
[68]    EntityRef           67
[09]    EntityValue         73, 74
[57]    EnumeratedType      54
[59]    Enumeration         47, 57
[25]    Eq                  24, 32, 41, 80
[42]    ETag                39
[75]    ExternalID          28, 73, 74, 82
[78]    extParsedEnt
[79]    extPE
```