XML in a Nutshell

Note

In these notes we abridge the XML Recommendation and extract from it the most important parts.

XML Requirements

The following high-level non-functional requirement have been specified for XML by the XML Working Group at W3C.

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

Well-Formed XML Documents

XML Recommendation specifies (in Backus-Naur form) the grammar for construction of an XML document. In addition, it provides some extra constraints that must be satisfied by well-formed and valid XML documents.

```
document    ::= prolog element Misc*

prolog      ::= XMLDecl Misc* (doctypedecl Misc*)?
XMLDecl     ::= '<?xml' VersionInfo EncodingDecl? SDecl? S? '?>'
VersionInfo ::= S 'version' S? '=' S? ('1.1'|'1.0'|``1.1",|``1.0")

element     ::= EmptyElemTag|
                STag content ETag
Stag        ::= '<' Name (S Attribute)* S? '>'
Etag        ::= '</' Name S? '/>'
EmptyElemTag::= '<' Name (S Attribute)* S? '/>'

Attribute   ::= Name S? '=' S? AttValue

content     ::= CharData? ((element|Reference|CDSect|PI|Comment) CharData?)*

Misc        ::= Comment | PI | S
```

PI = processing instruction
S = whitespace
Comment = comment
doctypedecl = XML DTD description/reference
Char = Unicode character data
RestrictedChar = restricted character data (?)
EncodingDecl = declaration of an encoding
SDecl = standalone document declaration
Name = allowable XML name - consists of "allowable" XML name characters
AttValue = XML attribute values
CDSect = section of unparsed character data

Notes

- An XML document consists of a prolog and one XML element, called the **root** element. The root does not appear in the content of any other element.
- For any **non-root** element, its start tag appears in the content of the same element as its end tag. That is, all elements *are properly nested.*

- Content of XML elements can consists of parsed character data (PCDATA), comments, processing instructions, elements, and sections of unparsed character data (CDATA) and character or entity references.
- XML elements can be empty, in which case they do not have any content and are represented by a special empty element tag.
- Non-empty XML elements are represented by a pair of a start tag and an end tag. Start tag can contain information about attributes associated with the element and their values.
- Attributes appear in start tags of XML elements. Their values are always unparsed characted data (CDATA).

Well-Formedness Constraints

Well-formedness constraints are constraints, which must be satisfied, in addition to the document production, for a string of text to be considered a well-formed XML document. Most important well-formedness constraints are shown below.

- **Element Type Match**. The Name in the element's end tag MUST match the element type in the start tag.
- **Unique Attribute Spec.** An attribute name MUST NOT appear more than once in the same start tag or empty element tag.
- **No < in Attribute Values.** The replacement text of any entity referred to directly or indirectly in an attribute value MUST NOT contain a '<'.

Document Type Declaration and Validity

Informally, **validity** means *correctness of an XML document w.r.t. a specification of its format.* Format specifications are not schemas in the relational database sense. Rather, they describe a set of formats - all considered correct.

Document Type Declarations, **DTDs** are a way to specify such formats included into the XML Recommendation. Other means of format specification exists, e.g., XML Schema.

DTD Syntax

```
doctypedecl   ::= '<!DOCTYPE' S Name (S ExternalID)? S?
                   ( '[' intSubset ']' S?)?

intSubset     :: = (markupdecl | DeclSep)*

DeclSep       ::= S | PEReference
```

```
 markupdecl    ::= elementdecl | AttlistDecl | EntityDecl |
                   NotationDecl | PI | Comment
elementdecl    ::= '<!ELEMENT' S Name S contentspec S? '>'
contentspec    ::= 'EMPTY' | 'ANY' | Mixed | children

children       ::= (choice|seq) ('?' | '*' | '+')?
choice         ::= '(' S? cp (S? '|' S? cp)+ S? ')'
cp             ::= (Name | choice | seq) ('?' | '*' | '+')?
seq            ::= '(' cp ( S? ',' S? cp)* S? ')'

Mixed          ::= '(' S? '#PCDATA' ( S? '|' Name)* S? ')*' |
                    '(' S? '#PCDATA' S? ')'

AttlistDecl    ::= '<ATTLIST' S Name AttDef* S? '>'
AttDef         ::= S Name S AttType S DefaultDecl
AttType        ::= 'CDATA' | 'ID' | 'IDREF' | 'IDREFS' |
                   'ENTITY' | 'ENTITIES' | 'NMTOKEN' | 'NMTOKENS' |EnumeratedType
EnumeratedType ::= NotationType | Enumeration
NotationType   ::= 'NOTATION' S '(' S? Name (S? '|' S? Name)* S? ')'
Enumeration    ::= '(' S? Nmtoken (S? '|' S? Nmtoken)* S? ')'

DefaultDecl    ::= '#REQUIRED' | '#IMPLIED' | ( ('#FIXED' S)? AttValue
```

ExternalID - pointer to an external XML DTD document
Nmtoken - name token
EntityDecl - declaration of an entity
NotationDecl - declaration of a notation
AttValue - allowable attribute values.

Notes

- Generally speaking, a DTD is either defined within the XML document itself via the intSubset non-terminal, or, an external file containing the DTD is referenced via the ExtrenalID non-terminal.

  However, the production for doctypedecl allows for a well-formed XML document to contain neither.

Validity

DTDs define a context-free grammar, whose productions the XML document must match.

However, just as well-formedness is not a context-free property, neither is validity. Validity constraints are associated both with the DTD and with the XML proper.

Well-formedness constraints on DTDs

Validity constraints on DTDs

- **Root Element Type.** The Name in the `<!DOCTYPE` declaration must match the root element of the XML document.
- **No Duplicate Types.** The same name MUST NOT appear more than once in a single mixed-content declaration.
- **ID.** Attribute Values of the type `ID` must match the Name production. A name MUST NOT appear more than once in an XML document as a value of this type.
- **One ID per element type.** An element type MUST NOT have more than one ID attribute.
- **ID Attribute Default.** An ID attribute must have a declared default of #IMPLIED or #REQUIRED.
- **IDREF.** Values of type IDREF MUST match the Name production and values of type IDREFS MUST match Names; each Name MUST match the value of an ID attribute on some element in the XML document.
- **Entity Name.** Values of type ENTITY must match the Name production, values of type ENTITIES MUST match Names; each name MUST match the name of an unparsed entity declared in the DTD.
- **Name Token.** Values of type NMTOKEN must match the Nmtoken production; values of type NMTOKENS must match the Nmtokens.
- **Notation Attributes.** Values of type 'notation' must match one of the notation names included in the declaration.
- **One notation per element type.** An element type MUST NOT have more than one NOTATION attribute.
- **No Notation on Empty Element**. For compatibility, an attribute of type NOTATION MUST NOT be declared on an element of type EMPTY.
- **No Duplicate Tokens.** The notation names in a single NotationType attribute declaration, as well as the NmTokens in a single Enumeration declaration MUST all be distinct.
- **Enumeration.** Values of this type MUST all match the Nmtoken production.
- **Root Element Type.** The Name in the document type declaration MUST match the element type of the root element.
- **Unique Element Type Declaration.** An element type MUST NOT be declared more than once in a DTD.
- **No Duplicate Types.** The same name MUST NOT appear more than once in a single mixed-content declaration.

Validity of XML Elements

Element Valid. An element is **valid** if there is a declaration matching elementdecl production in the DTD where the Name matches the element type and one of the following holds:

EMPTY: The declaration matches the EMPTY production and the element has not content (`<x/>` or `<x></x>`).

children: The declaration matches children and the sequence of child elements belongs to the language generated by the regular expression on the right-hand side of the production.

Mixed: The declaration matches Mixed production and the content consists of character data, comments, PIs and child elements mentioned in the production.

ANY: The declaration matches ANY, and the content consists of character data and child elements whose types have been declared.

Other Validity Constraints

1. **Attribute Type Value.** Attributes found in start tags and empty tages of XML elements MUST be declared in the the DTD, their value MUST be of declared type.

2. **Required Attribute.** If the default declaration is #REQUIRED, the the attribute MUST be specified for all elements of the type.

3. **Attribute Default Value Syntactically Correct.** The declared default value MUST meet the syntactic constraints of the declared attribute type.

4. **Fixed Attribute Default.** In an attribute has a default value declared with the #FIXED keyword, instances of that attribute MUST match the default value.