XML DTD and Schemas

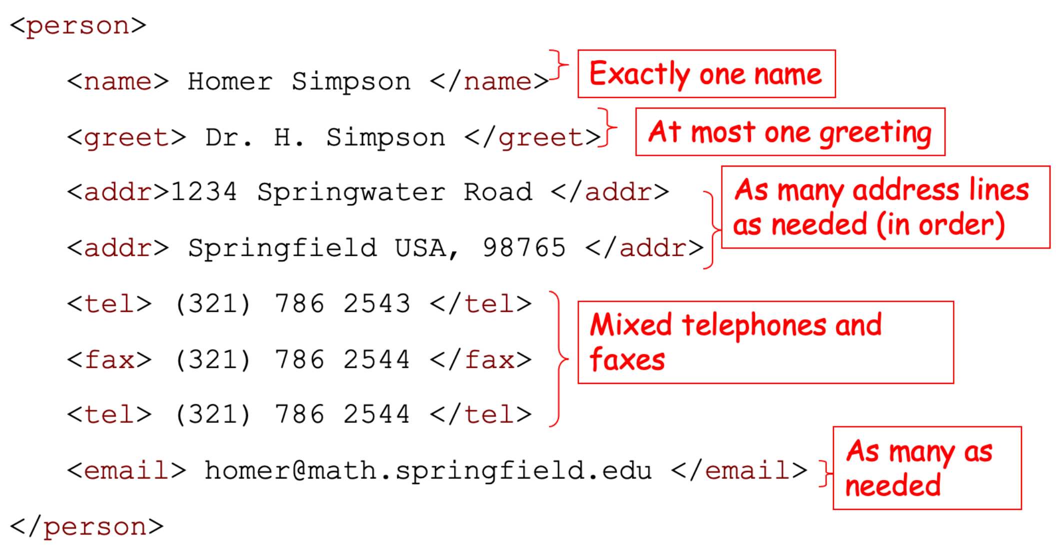
Type system to enforce data constraints.

Document Type Definitions (DTDs)

- A way to specify the structure of XML documents.
- DTD adds syntactical requirements in addition to the well-formed requirement.
- DTDs help in
  - Eliminating errors when creating or editing XML documents.
  - Clarifying the intended semantics.
  - Simplifying the processing of XML documents.

- DTDs
  - Use “regular expression” like syntax to specify a grammar for the XML document.
  - Have limitations such as weak data types, inability to specify complex constraints, no support for schema evolution, etc.

Example: An Address Book



```
<person>
   <name> Homer Simpson </name>
   <greet> Dr. H. Simpson </greet>
   <addr>1234 Springwater Road </addr>
   <addr> Springfield USA, 98765 </addr>
   <tel> (321) 786 2543 </tel>
   <fax> (321) 786 2544 </fax>
   <tel> (321) 786 2544 </tel>
   <email> homer@math.springfield.edu </email>
</person>
```

Exactly one name
At most one greeting
As many address lines as needed (in order)
Mixed telephones and faxes
As many as needed

Specifying the Structure

Regular expression syntax (inspired from UNIX regular expressions)

expression	denotes
`name`	a name element
`greet?`	an optional (0 or 1) greet elements
`name, greet?`	a name followed by an optional greet
`addr*`	0 or more address lines
`tel | fax`	a tel or a fax element
`(tel | fax)*`	0 or more repeats of tel or fax
`email*`	0 or more email elements

So the whole structure of a person entry is specified by

`name, greet?, addr*, (tel | fax)*, email*`

- Each element type of the XML document is described by an expression
- the leaf level element types are described by the data type (#PCDATA) - parsed character data
- Each attribute of an element type is also described in the DTD by enumerating some of its properties (OPTIONAL, etc.)

Element Type Definition

For each element type `E`, a declaration of the form:

```
<!ELEMENT   E   content-model>
```

where the `content-model` is an expression:

```
content-model ::=
  EMPTY   | ANY | #PCDATA |  P1, P2 | P1 | P2 |  P1?  | P1+  | P1* | (P)
```

expression	denotes
`P1 , P2`	concatenation
`P1 | P2`	disjunction
`P?`	optional
`P+`	one or more occurrences
`P*`	the Kleene closure
`(P)`	grouping

The definition of an element consists of exactly one of the following:

- #PCDATA
- A regular expression (as defined earlier)
- EMPTY: element has no content
- ANY: content can be any mixture of PCDATA and elements defined in the DTD

Mixed content is described by a repeatable OR group

```
(#PCDATA | element-name | …)*
```

Inside the group, no regular expressions – just element names; i.e. `#PCDATA` must be first followed by 0 or more element names, separated by `|`; The group can be repeated 0 or more times

Address Book Document with an Internal DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE addressbook [
   <!ELEMENT addressbook (person*)>
   <!ELEMENT person (name, greet?, address*, (fax | tel)*, email*)>
   <!ELEMENT name    (#PCDATA)>
   <!ELEMENT greet   (#PCDATA)>
   <!ELEMENT address (#PCDATA)>
   <!ELEMENT tel     (#PCDATA)>
   <!ELEMENT fax     (#PCDATA)>
   <!ELEMENT email   (#PCDATA)>
]>
<addressbook>
  <person>
    <name>Jeff Cohen</name>
    <greet>Dr. Cohen</greet>
    <email>jc@penny.com</email>
  </person>
</addressbook>
```

Some Difficult Structures

Each employee element should contain name, age and ssn elements in some order

```
<!ELEMENT employee
    ((name, age, ssn) |
     (age, ssn, name) |
     (ssn, name, age) |
      ...
      ...
    )>
```

Too many permutations!

Attribute Specification in DTDs

```
<!ELEMENT height (#PCDATA)>
<!ATTLIST height
      dimension CDATA #REQUIRED
      accuracy  CDATA #IMPLIED >
```

- The dimension attribute is required
- The accuracy attribute is optional
- CDATA is the "type" of the attribute – character data

The format of an Attribute Definition

```
<!ATTLIST element-name attr-name attr-type attr-default>
```

The default value is given inside quotes

Attribute types:

- CDATA
- ID, IDREF, IDREFS

ID, IDREF, IDREFS are used for references

Attribute Default

- #REQUIRED: the attribute must be explicitly provided
- #IMPLIED: attribute is optional, no default provided
- "value": if not explicitly provided, this value inserted by default
- #FIXED "value": as above, but only this value is allowed

Recursive DTDs

```
<DOCTYPE genealogy [
    <!ELEMENT   genealogy (person*)>
    <!ELEMENT   person (
        name,
        dateOfBirth,
        person,         -- mother
        person   ) >    -- father
]>
```

Problem with this DTD: Parser does not see the recursive structure and looks for “person” sub-element indefinitely!

```
<DOCTYPE genealogy [
    <!ELEMENT   genealogy (person*)>
    <!ELEMENT   person (
        name,
        dateOfBirth,
        person?,        -- mother
        person?  ) >    -- father
    ...
]>
```

The problem with this DTD is if only one “person” sub-element is present, we would not know if that person is the father or the mother.

Using ID and IDREF Attributes

```
  <!DOCTYPE family [
   <!ELEMENT family   (person)* >
   <!ELEMENT person  (name) >
   <!ELEMENT name    (#PCDATA) >
   <!ATTLIST  person
        id ID #REQUIRED
        mother IDREF #IMPLIED
        father IDREF #IMPLIED
        children IDREFS #IMPLIED >
 ]>
```

IDs and IDREFs

- ID attribute: unique within the entire document.
    - An element can have at most one ID attribute.
    - No default (fixed default) value is allowed.
        - #required: a value must be provided
        - #implied: a value is optional

- IDREF attribute: its value must be some other element’s ID value in the document.
- IDREFS attribute: its value is a set, each element of the set is the ID value of some other element in the document.

```
<person id="898" father="332" mother="336" children="982 984 986">
```

Some Conforming Data

```
<family>
    <person  id="lisa"  mother="marge" father="homer">
      <name> Lisa Simpson </name>
    </person>
    <person  id="bart"  mother="marge" father="homer">
      <name> Bart Simpson </name>
    </person>
    <person id="marge" children="bart lisa">
        <name> Marge Simpson </name>
    </person>
    <person id="homer" children="bart lisa">
        <name> Homer Simpson </name>
    </person>
</family>
```

Limitations of ID References

- The attributes mother and father are references to IDs of other elements.
- However, those are not necessarily person elements!
- The mother attribute is not necessarily a reference to a female person.

An Alternative Specification

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE family [
    <!ELEMENT family (person)* >
    <!ELEMENT person (name, mother?, father?, children?) >
       <!ATTLIST person id ID #REQUIRED >
    <!ELEMENT name (#PCDATA) >
    <!ELEMENT mother EMPTY >
       <!ATTLIST mother idref IDREF #REQUIRED >
    <!ELEMENT father EMPTY >
       <!ATTLIST father idref IDREF #REQUIRED >
    <!ELEMENT children EMPTY >
       <!ATTLIST children idrefs IDREFS #REQUIRED >
]>
```

Empty sub-elements instead of attributes

The Revised Data

```
<family>
  <person id="marge">
    <name>Marge Simpson</name>
    <children idrefs="bart lisa"/>
  </person>
  <person id="homer">
    <name>Homer Simpson</name>
    <children idrefs="bart lisa" />
  </person>
 <person id="bart">
   <name>Bart Simpson</name>
   <mother idref="marge"/>
   <father idref="homer"/>
 </person>
 <person id="lisa">
   <name>Lisa Simpson</name>
   <mother idref="marge"/>
   <father idref="homer"/>
 </person>
</family>
```

Consistency of ID and IDREF Attribute Values

- If an attribute is declared as ID
    - The associated value must be distinct, i.e., different elements (in the given document) must have different values for the ID attribute.
    - Even if the two elements have different element names

- If an attribute is declared as IDREF
    - The associated value must exist as the value of some ID attribute (no dangling “pointers”)

- Similarly for all the values of an IDREFS attribute
- ID, IDREF and IDREFS attributes are not typed

Adding a DTD to the Document

A DTD can be

- *internal*: The DTD is part of the document file
- *external*: The DTD and the document are on separate files
- An external DTD may reside
    - In the local file system (where the document is)

    - In a remote file system

Connecting a Document with its DTD

An internal DTD

```
<?xml version="1.0"?>
<!DOCTYPE db [<!ELEMENT ...> … ]>
<db> ... </db>
```

A DTD from the local file system:

```
    <!DOCTYPE db SYSTEM "schema.dtd">
```

A DTD from a remote file system:

```
<!DOCTYPE db SYSTEM "http://www.schemaauthority.com/schema.dtd">
```

Well-Formed XML Documents

An XML document (with or without a DTD) is **well-formed** if

- Tags are syntactically correct
- Every tag has an end tag
- Tags are properly nested
- There is a root tag
- A start tag does not have two occurrences of the same attribute

Valid Documents

A well-formed XML document is **valid** if it conforms to its DTD, that is,

- The document conforms to the regular-expression grammar
- The attributes types are correct, and
- The constraints on references are satisfied