XML Schema Languages 1/2

Dongwon Lee, Ph.D.
The Pennsylvania State University

IST 516

http://www.practicingsafetechs.com/TechsV1/XMLSchemas/
Course Objectives

● Understand the purpose of using schemas

● Study regular expression as the framework to formalize schema languages

● Understand details of DTD and XML Schema

● Learn the concept of schema validation
Motivation

• The company “Nittany Vacations, LLC” wants to standardize all internal documents using the XML-based format $\rightarrow$ nvML

• Gather requirements from all employees
  • Informal description (ie, narrative) of how nvML should look like

• Q
  • How to describe formally and unambiguously?
  • How to validate an nvML document?
Motivation: Schema = Rules

- XML Schemas is all about expressing rules:
  - Rules about what data is (not) allowed
  - Rules about how the data must be organized
  - Rules about the relationships between data
<Vacation date="3000-09-09" guide-by="Lee">
  <Trip segment="1" mode="air">
    <Transportation>airplane</Transportation>
  </Trip>
  <Trip segment="2" mode="water">
    <Transportation>boat</Transportation>
  </Trip>
  <Trip segment="3" mode="ground">
    <Transportation>car</Transportation>
  </Trip>
</Vacation>

Example modified from Roger L. Costello’s slides @ xfront.com
Motivation: Validate

<p>&lt;Vacation date="3000-09-09" guide-by="Lee"&gt;
   &lt;Segment id="1" mode="air"&gt;
      &lt;Transportation&gt;airplane&lt;/Transportation&gt;
   &lt;/Segment&gt;
   &lt;Segment id="2" mode="water"&gt;
      &lt;Transportation&gt;boat&lt;/Transportation&gt;
   &lt;/Segment&gt;
   &lt;Segment id="3" mode="ground"&gt;
      &lt;Transportation&gt;car&lt;/Transportation&gt;
   &lt;/Segment&gt;
&lt;/Vacation&gt;</p>

Validate the XML document against the XML Schema

XML Schema = RULES

Rule 1: A vacation has segments.
Rule 2: Each segment is uniquely identified.
Rule 3: There are three modes of transportation: air, water, ground.
Rule 4: Each segment has a mode of transportation.
Rule 5: Each segment must identify the specific mode used.
Schema Languages

- Schema: a formal description of structures / constraints
  - Eg, relational schema describes tables, attributes, keys, ..
- Schema Language: a formal language to describe schemas
  - Eg, SQL DDL for relational model

```sql
CREATE TABLE employees
(
id INTEGER PRIMARY KEY,
first_name CHAR(50) NULL,
last_name CHAR(75) NOT NULL,
dateofbirth DATE NULL
);
```

- Why bother formalizing the syntax with a schema?
  - A formal definition provides a **precise but human-readable** reference
  - Schema processing can be done with **existing implementations**
  - One’s own tools for own language can benefit:
    by piping input documents through a schema processor, one can assume that the input is valid and defaults have been inserted
Schema Processing

- XML document
- Schema

Schema processor

- valid
- invalid

- Normalized XML document
- Error message

Requirements for Schema Lang.

- Expressiveness
- Efficiency
- Comprehensibility
Regular Expressions (RE)

- Commonly used to describe sequences of characters or elements in schema languages.
- RE to capture content models.
- \( \Sigma \): a finite Alphabet
  - \( \alpha \in \Sigma \): set only containing the character \( \alpha \)
  - \( \alpha ? \): matches zero or one \( \alpha \)
  - \( \alpha * \): matches zero or more \( \alpha \)’s
  - \( \alpha + \): matches one or more \( \alpha \)’s
  - \( \alpha \beta \): matches the concatenation of \( \alpha \) and \( \beta \)
  - \( \alpha | \beta \): matches the union of \( \alpha \) and \( \beta \)
RE Examples

- $a|b^*$ denotes $\{\varepsilon, a, b, bb, bbb, \ldots\}$
- $(a|b)^*$ denotes the set of all strings with no symbols other than $a$ and $b$, including the empty string: $\{\varepsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$
- $ab^*(c|\varepsilon)$ denotes the set of strings starting with $a$, then zero or more $b$s and finally optionally a $c$: $\{a, ac, ab, abc, abb, abbc, \ldots\}$
RE Examples

- Valid integers:
  - 0 | -? (1|2|3|4|5|6|7|8|9) (1|2|3|4|5|6|7|8|9) *

- Valid contents of table element in XHTML:
  - caption ? (col * | colgroups *)
    thead ? tbody ? (tfoot * | tr *)
Which Schema Language?

- Many proposals competing for acceptance
- W3C Proposals: DTD, XML Data, DCD, DDML, SOX, XML-Schema, ...
- Non-W3C Proposals: Assertion Grammars, Schematron, DSD, TREX, RELAX, XDuce, RELAX-NG, ...

- Different applications have different needs from a schema language
Family Tree of Schema Languages for Markup Languages

By Rick Jelliffe © 1999, 2006. Permission granted to use this in any way providing this attribution is kept. (v.5)
Expressive Power (content model)

Closure (content model)

- DTD
- XML-Schema
- XDuce, RELAX-NG

Closed under INTERSECT

Closed under INTERSECT, UNION, DIFFERENCE
DTD: Document Type Definition

- XML DTD is a subset of SGML DTD
- XML DTD is the standard XML Schema Language of the past (and present maybe...)
  - It is one of the simplest and least expressive schema languages proposed for XML model
  - It does **NOT** use XML tag notation, but use its own *weird* notation
  - It cannot express relatively complex constraint (eg, key with scope) well
  - It is being replaced by XML-Schema of W3C and RELAX-NG of OASIS
DTD: Elements

- `<!ELEMENT element-name content-model>`
- Associates a content model to all elements of the given name content models

- EMPTY: no content is allowed
- ANY: any content is allowed
- Mixed content: (#PCDATA | e₁ | … | eₙ)*
  arbitrary sequence of character data and listed elements
DTD: Elements

- Eg: “Name” element consists of an
  - optional FirstName, followed by
  - mandatory LastName elements, where
  - Both are text string

```xml
<!ELEMENT Name (FirstName?, LastName)>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT LastName (#PCDATA)>
```
DTD: Attributes

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

Declares which attributes are allowed or required in which elements attribute types:

- CDATA: any value is allowed (the default)
- `(value...)`: enumeration of allowed values
- ID, IDREF, IDREFS: ID attribute values must be unique (contain "element identity"), IDREF attribute values must match some ID (reference to an element)
- ENTITY, ENTITIES, NMTOKEN, NMTOKENS, NOTATION: consider them obsolete…
DTD: Attributes

- Attribute defaults:
  - #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
DTD: Attributes

- Eg: “Name” element consists of an
  - optional FirstName, followed by
  - mandatory LastName attributes, where
  - Both are text string

```xml
<!ELEMENT Name (EMPTY)>  
<!ATTLIST Name FirstName CDATA #IMPLIED  
LastName CDATA #REQUIRED>
```
**DTD: Attributes**

- **ID vs. IDREF/IDREFS**
  - ID: document-wide unique ID (like key in DB)
  - IDREF: referring attribute (like foreign key in DB)

```xml
<!ELEMENT employee (…)>
<!ATTLIST employee
eID ID #REQUIRED
  boss IDREF #IMPLIED>

…

<employee eID="a">…</employee>…
<employee eID="b" boss="a">…</employee>
```
Namespace: History & Def.

- Namespaces in XML 1.0
  - W3C recommendation (2009)
  - [http://www.w3.org/TR/REC-xml-names/](http://www.w3.org/TR/REC-xml-names/)

- Namespaces
  - Multiple schema may have identical element or attribute names
    - Eg, html <title> vs. book <title>
  - A way to distinguish element and attribute names used in XML documents
Namespace Declaration

```
<x xmlns:b='http://pike.psu.edu/book-schema'>
    ...
    // contents of <x> here
</x>
```

- “xmlns”:prefix = namespace name
- Meaning: the prefix “b” is bound to the URL (http://pike.psu.edu/book-schema) for the “x” element and its contents
Where To Point At?

- `xmlns : prefix = “URL”`
- URL as the name of the namespace
- But this URL doesn’t have to exist on Web
- Just a symbolic URL bound to a schema
- W3C decided to use URL as the name of namespaces since:
  - URL contains domain names (eg, pike.psu.edu)
  - Works globally across the Web
Scenario: webster.com

- Suppose (fake) webster.com has 2 schemas:
  - author.dtd defines elements like <author>, <name>, <title>, <contact>
  - book.dtd defines elements like <book>, <title>, <price>, <publisher>

- When writing an XML file using both schemas, some name collision occurs
  - <author><title>Associate Prof</title></>
  - <book><title>Gone with the wind</title></>
  - Machines can’t distinguish two <title> elements
Problem: Name Collision

- For each element/attribute, explicitly specify from which schema they are from.

```xml
<webster>
  <author>
    <title>Associate Prof</title>
  </author>
  <book>
    <title>Gone with the wind</title>
  </book>
</webster>
```

Name Collision
Solution #1

- For each element/attribute, explicitly specify from which schema they are from

```xml
<webster>
      Associate Prof</A:title>
  </A:author>
      Gone with the wind</B:title>
  </B:book>
</webster>
```

No real URL – just symbolic URL to serve as the name of the namespace
Solution #2

- Declare namespace as attributes of the root element by convention

```xml
<webster xmlns:A="http://www.webster.com/author"
  <A:author>
    <A:title>Associate Prof</A:title>
  </A:author>
  <B:book>
    <B:title>Gone with the wind</B:title>
  </B:book>
</webster>
```
Solution #3

- Declare “default” namespace by not specifying the prefix

```xml
<webster xmlns:A="http://www.webster.com/author"
  <A:author>
    <A:title>Associate Prof</A:title>
  </A:author>
  <book>
    <title>Gone with the wind</title>
  </book>
</webster>
```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
<?xml version="1.0"?>

// Reference to schema goes here

<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
3 Ways to Reference a DTD

- Embed DTD within XML file
  - Used only within the XML file
  - Usually, NOT recommended to use
- External: common DTDs shared by many
  - Private
    ```xml
    <!DOCTYPE root_element SYSTEM "DTD_location">
    ```
  - Public
    ```xml
    <!DOCTYPE root_element PUBLIC "DTD_name"
    "DTD_location">
    ```
    URL
    prefix//owner_of_the_DTD//
    description_of_the_DTD//
    ISO 639_language_identifier
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to, from, heading, body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]> 

<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "http://pike.psu.edu/foo/bar/note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
<?xml version="1.0"?>
<!DOCTYPE note PUBLIC "-//PSU//DTD for note//EN" "http://pike.psu.edu/foo/bar/note.dtd">

<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
// event.dtd
<!ELEMENT event (acronym, society*, url?, loc, year)>
<!ATTLIST event eID ID #REQUIRED>
<!ELEMENT acronym (#PCDATA)>
<!ELEMENT society (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT loc (city, state)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT year (#PCDATA)>
Exercise: event.xml

XML document that conforms to “event.dtd”

```xml
<?xml version="1.0"?>
<!DOCTYPE event SYSTEM "../../dir/event.dtd">
<event eID="sigmod02">
  <acronym>SIGMOD</acronym>
  <society>ACM</society>
  <url>www.sigmod02.org</url>
  <loc>
    <city>Madison</city> <state>WI</state>
  </loc>
  <year>2002</year>
</event>
```
XML Editor Examples

- SyncroSoft <oXygen/>
  - 30 day free license, both Mac and Windows
  - http://www.oxygenxml.com/xml_editor/

- Altova XML Spy
  - 30 day free license, both Mac and Windows
  - http://www.altova.com/xml-editor/

- XMLPad
  - Free license, Only Windows version
  - http://www.wmhelp.com/xmlpad3.htm or
  - http://download.cnet.com/XmlPad/3000-7241_4-10252051.html

- XMLGrid
  - Online XML editor: http://xmlgrid.net/
01 <?xml version="1.0" encoding="UTF-8"?>
02 <!-- Enter you external DTD here -->
03 <!ELEMENT event (acronym, society*, url?, loc, year)>
04 <!ATTLIST event eID ID #REQUIRED>
05 <!ELEMENT acronym (#PCDATA)>
06 <!ELEMENT society (#PCDATA)>
07 <!ELEMENT url (#PCDATA)>
08 <!ELEMENT loc (city, state)>
09 <!ELEMENT city (#PCDATA)>
10 <!ELEMENT state (#PCDATA)>
11 <!ELEMENT year (#PCDATA)>
12
13
14
XMLPad 3
XMLPad 3
References


- W3Schools XML Schema Tutorial
  - Much of slides for the XML Schema (2nd half) are modified from W3Schools materials
  - [http://www.w3schools.com/schema/default.asp](http://www.w3schools.com/schema/default.asp)

- Zvon XML Schema Tutorial
  - [http://www.zvon.org/comp/m/schema.html](http://www.zvon.org/comp/m/schema.html)