University of Washington
Department of Computer Science & Engineering

# Managing XML and Semistructured Data

Lecture 12: XML Schema

Prof. Dan Suciu

Spring 2001

## In this lecture

- XML Schemas
  - Elements v. Types
  - Regular expressions
  - Expressive power

Resources

W3C Draft: www.w3.org/TR/2001/REC-xmlschema-1-20010502

## XML Schemas

- http://www.w3.org/TR/xmlschema-1/10/2000
- generalizes DTDs
- uses XML syntax
- two documents: structure and datatypes
  - http://www.w3.org/TR/xmlschema-1
  - http://www.w3.org/TR/xmlschema-2
- XML-Schema is very complex
  - often criticized
  - some alternative proposals

## XML Schemas

```
<xsd:element name="paper" type="papertype"/>
<xsd:complexType name="papertype">
   <xsd:sequence>
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="author" minOccurs="0"/>
      <xsd:element name="year"/>
      <xsd: choice> < xsd:element name="journal"/>
                    <xsd:element name="conference"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:element>
```

DTD: <!ELEMENT paper (title,author*,year, (journal|conference))>

## Elements v.s. Types in XML Schema

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="name"
                   type="xsd:string"/>
      <xsd:element name="address"
                   type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="person"
             type="ttt">
<xsd:complexType name="ttt">
  <xsd:sequence>
    <xsd:element name="name"
                 type="xsd:string"/>
    <xsd:element name="address"
                 type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

DTD: <!ELEMENT person (name,address)>

## Elements v.s. Types in XML Schema

- Types:
  - Simple types (integers, strings, ...)
  - Complex types (regular expressions, like in DTDs)
- Element-type-element alternation:
  - Root element has a complex type
  - That type is a regular expression of elements
  - Those elements have their complex types...
  - ...
  - On the leaves we have simple types

## Local and Global Types in XML Schema

- Local type:

```
<xsd:element name="person">
        [define locally the person's type]
</xsd:element>
```

- Global type:

```
<xsd:element name="person" type="ttt"/>

<xsd:complexType name="ttt">
        [define here the type ttt]
</xsd:complexType>
```

Global types: can be reused in other elements

## Local v.s. Global Elements in XML Schema

- Local element:

```
<xsd:complexType name="ttt">
    <xsd:sequence>
        <xsd:element name="address" type="..."/>...
    </xsd:sequence>
</xsd:complexType>
```

- Global element:

```
<xsd:element name="address" type="..."/>

<xsd:complexType name="ttt">
    <xsd:sequence>
        <xsd:element ref="address"/>  ...
    </xsd:sequence>
</xsd:complexType>
```

Global elements: like in DTDs

## Regular Expressions in XML Schema

Recall the element-type-element alternation:

```
<xsd:complexType name="....">
        [regular expression on elements]
</xsd:complexType>
```

Regular expressions:

- <xsd:sequence> A B C </...> = A B C
- <xsd:choice> A B C </...> = A | B | C
- <xsd:group> A B C </...> = (A B C)
- <xsd:... minOccurs="0" maxOccurs="unbounded"> ..</...> = (...)*
- <xsd:... minOccurs="0" maxOccurs="1"> ..</...> = (...)?

## Local Names in XML-Schema

**name** has different meanings in **person** and in **product**

```
<xsd:element name="person">
  <xsd:complexType>
      . . . .
    <xsd:element name="name">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="firstname" type="xsd:string"/>
                <xsd:element name="lastname" type="xsd:string"/>
            </xsd:sequence>
    </xsd:element>
    . . . .
  </xsd:complexType>
</xsd:element>


<xsd:element name="product">
  <xsd:complexType>
      . . . . .
    <xsd:element name="name"   type="xsd:string"/>

  </xsd:complexType>
</xsd:element>
```

## Subtle Use of Local Names

```
<xsd:element name="A" type="oneB"/>


<xsd:complexType name="onlyAs">
  <xsd:choice>
    <xsd:sequence>
        <xsd:element name="A" type="onlyAs"/>
        <xsd:element name="A" type="onlyAs"/>
    </xsd:sequence>
    <xsd:element name="A" type="xsd:string"/>
  </xsd:choice>
</xsd:complexType>
```

```
<xsd:complexType name="oneB">
  <xsd:choice>
    <xsd:element name="B" type="xsd:string"/>
    <xsd:sequence>
        <xsd:element name="A" type="onlyAs"/>
        <xsd:element name="A" type="oneB"/>
    </xsd:sequence>
    <xsd:sequence>
        <xsd:element name="A" type="oneB"/>
        <xsd:element name="A" type="onlyAs"/>
    </xsd:sequence>
  </xsd:choice>
</xsd:complexType>
```

Arbitrary deep binary tree with A elements, and a single B element

## Attributes in XML Schema

```
<xsd:element name="paper" type="papertype"/>
<xsd:complexType name="papertype">
    <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        . . . . . .
</xsd:sequence>
 <xsd:attribute name="language" type="xsd:NMTOKEN" fixed="English"/>
</xsd:complexType>
```

Attributes are associated to the *type*, not to the element
Only to *complex types*; more trouble if we want to add attributes to *simple types*.

## "Mixed" Content, "Any" Type

```
<xsd:complexType mixed="true">
  . . . .
```

- Better than in DTDs: can still enforce the type, but now may have text between any elements

```
<xsd:element name="anything" type="xsd:anyType"/>
  . . . .
```

- Means anything is permitted there

## "All" Group

```
<xsd:complexType name="PurchaseOrderType">
 <xsd:all>  <xsd:element name="shipTo" type="USAddress"/>
            <xsd:element name="billTo" type="USAddress"/>
            <xsd:element ref="comment" minOccurs="0"/>
            <xsd:element name="items"  type="Items"/>
 </xsd:all>
 <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>
```

- A restricted form of & in SGML
- Restrictions:
  - Only at top level
  - Has only elements
  - Each element occurs at most once
- E.g. "comment" occurs 0 or 1 times

## Derived Types by Extensions

```
<complexType name="Address">
 <sequence>  <element name="street" type="string"/>
             <element name="city"   type="string"/>
 </sequence>
</complexType>

<complexType name="USAddress">
 <complexContent>
  <extension base="ipo:Address">
  <sequence>  <element name="state" type="ipo:USState"/>
              <element name="zip"   type="positiveInteger"/>
  </sequence>
  </extension>
 </complexContent>
</complexType>
```

Corresponds to inheritance

## Derived Types by Restrictions

```
<complexContent>
   <restriction base="ipo:Items">
    … [rewrite the entire content, with restrictions]...
   </restriction>
</complexContent>
```

- (*): may restrict cardinalities, e.g. (0,infty) to (1,1); may restrict choices; other restrictions…

Corresponds to set inclusion

## Simple Types

- String
- Token
- Byte
- unsignedByte
- Integer
- positiveInteger
- Int (larger than integer)
- unsignedInt
- Long
- Short
- ...
- Time
- dateTime
- Duration
- Date
- ID
- IDREF
- IDREFS

## Facets of Simple Types

•Facets = additional properties restricting a simple type

•15 facets defined by XML Schema

Examples

- length
- minLength
- maxLength
- pattern
- enumeration
- whiteSpace
- maxInclusive
- maxExclusive
- minInclusive
- minExclusive
- totalDigits
- fractionDigits

## Facets of Simple Types

- Can further restrict a simple type by changing some facets
- Restriction = subset

## Not so Simple Types

- List types:

```
<xsd:simpleType name="listOfMyIntType">
  <xsd:list itemType="myInteger"/>
</xsd:simpleType>
```

```
<listOfMyInt>20003 15037 95977 95945</listOfMyInt>
```

- Union types
- Restriction types

## Summary of XML Schema

- Formal Expressive Power:
  - Can express precisely the regular tree languages (over unranked trees)
- Lots of other stuff
  - Some form of inheritance
  - A "null" value
  - Large collection of data types

## Summary of Schemas

- in SS data:
  - graph theoretic
  - data and schema are decoupled
  - used in data processing
- in XML
  - from grammar to object-oriented
  - schema wired with the data
  - emphasis on semantics for exchange