Chapter 2
Structured Web Documents in XML

Grigoris Antoniou
Frank van Harmelen

An HTML Example

```
<h2>Nonmonotonic Reasoning: Context-
   Dependent Reasoning</h2>
<i>by <b>V. Marek</b> and
   <b>M. Truszczynski</b></i><br>
Springer 1993<br>
ISBN 0387976892
```

The Same Example in XML

```
<book>
    <title>Nonmonotonic Reasoning: Context-
        Dependent Reasoning</title>
    <author>V. Marek</author>
    <author>M. Truszczynski</author>
    <publisher>Springer</publisher>
    <year>1993</year>
    <ISBN>0387976892</ISBN>
</book>
```

HTML versus XML: Similarities

- Both use tags (e.g. <h2> and </year>)
- Tags may be nested (tags within tags)
- Human users can read and interpret both HTML and XML representations quite easily

… But how about machines?

Problems with Automated Interpretation of HTML Documents

An intelligent agent trying to retrieve the names of the authors of the book

- Authors' names could appear immediately after the title
- or immediately after the word by
- Are there two authors?
- Or just one, called "V. Marek and M. Truszczynski"?

HTML vs XML: Structural Information

- HTML documents do not contain structural information: pieces of the document and their relationships.
- XML more easily accessible to machines because
  - Every piece of information is described.
  - Relations are also defined through the nesting structure.
  - E.g., the **<author>** tags appear within the **<book>** tags, so they describe properties of the particular book.

HTML vs XML: Structural Information (2)

- A machine processing the XML document would be able to deduce that
  - the **author** element refers to the enclosing **book** element
  - rather than by proximity considerations
- XML allows the definition of constraints on values
  - E.g. a year must be a number of four digits

HTML vs XML: Formatting

- The HTML representation provides more than the XML representation:
  - The formatting of the document is also described
- The main use of an HTML document is to display information: it must define formatting
- XML: separation of content from display
  - same information can be displayed in different ways

HTML vs XML: Another Example

- In HTML

```
<h2>Relationship force-mass</h2>
<i> F = M × a </i>
```

- In XML

```
<equation>
      <meaning>Relationship force-mass</meaning>
      <leftside> F </leftside>
      <rightside> M × a </rightside>
</equation>
```

HTML vs XML: Different Use of Tags

- In both HTML docs same tags
- In XML completely different

- HTML tags define display: color, lists …
- XML tags not fixed: user definable tags
- XML meta markup language: language for defining markup languages

XML Vocabularies

- Web applications must agree on common vocabularies to communicate and collaborate
- Communities and business sectors are defining their specialized vocabularies
  - mathematics (MathML)
  - bioinformatics (BSML)
  - human resources (HRML)
  - …

Lecture Outline

1. Introduction
2. Detailed Description of XML
3. Structuring
   a) DTDs
   b) XML Schema
4. Namespaces
5. Accessing, querying XML documents: XPath
6. Transformations: XSLT

The XML Language

An XML document consists of

- a prolog
- a number of elements
- an optional epilog (not discussed)

Prolog of an XML Document

The prolog consists of

- an XML declaration and
- an optional reference to external structuring documents

<?xml version="1.0" encoding="UTF-16"?>

<!DOCTYPE book SYSTEM "book.dtd">

XML Elements

- The “things” the XML document talks about
  - E.g. books, authors, publishers
- An element consists of:
  - an opening tag
  - the content
  - a closing tag

<lecturer>David Billington</lecturer>

XML Elements (2)

- Tag names can be chosen almost freely.
- The first character must be a letter, an underscore, or a colon
- No name may begin with the string “xml” in any combination of cases
  - E.g. “Xml”, “xML”

Content of XML Elements

- Content may be text, or other elements, or nothing

```
<lecturer>
    <name>David Billington</name>
    <phone> +61 – 7 – 3875 507 </phone>
</lecturer>
```

- If there is no content, then the element is called empty; it is abbreviated as follows:
  <lecturer/> for **<lecturer></lecturer>**

XML Attributes

- An empty element is not necessarily meaningless
  - It may have some properties in terms of attributes
- An attribute is a name-value pair inside the opening tag of an element

  <lecturer name="David Billington" phone="+61 − 7 − 3875 507"/>

XML Attributes: An Example

```
<order orderNo="23456" customer="John Smith"
        date="October 15, 2002">
        <item itemNo="a528" quantity="1"/>
        <item itemNo="c817" quantity="3"/>
</order>
```

The Same Example without Attributes

```
<order>
    <orderNo>23456</orderNo>
    <customer>John Smith</customer>
    <date>October 15, 2002</date>
    <item>
        <itemNo>a528</itemNo>
        <quantity>1</quantity>
    </item>
    <item>
        <itemNo>c817</itemNo>
        <quantity>3</quantity>
        </item>
</order>
```

XML Elements vs Attributes

- Attributes can be replaced by elements
- When to use elements and when attributes is a matter of taste
- But attributes **cannot** be nested

Further Components of XML Docs

- Comments
  - A piece of text that is to be ignored by parser
  - **<!-- This is a comment -->**
- Processing Instructions (PIs)
  - Define procedural attachments
  - **<?stylesheet type="text/css" href="mystyle.css"?>**

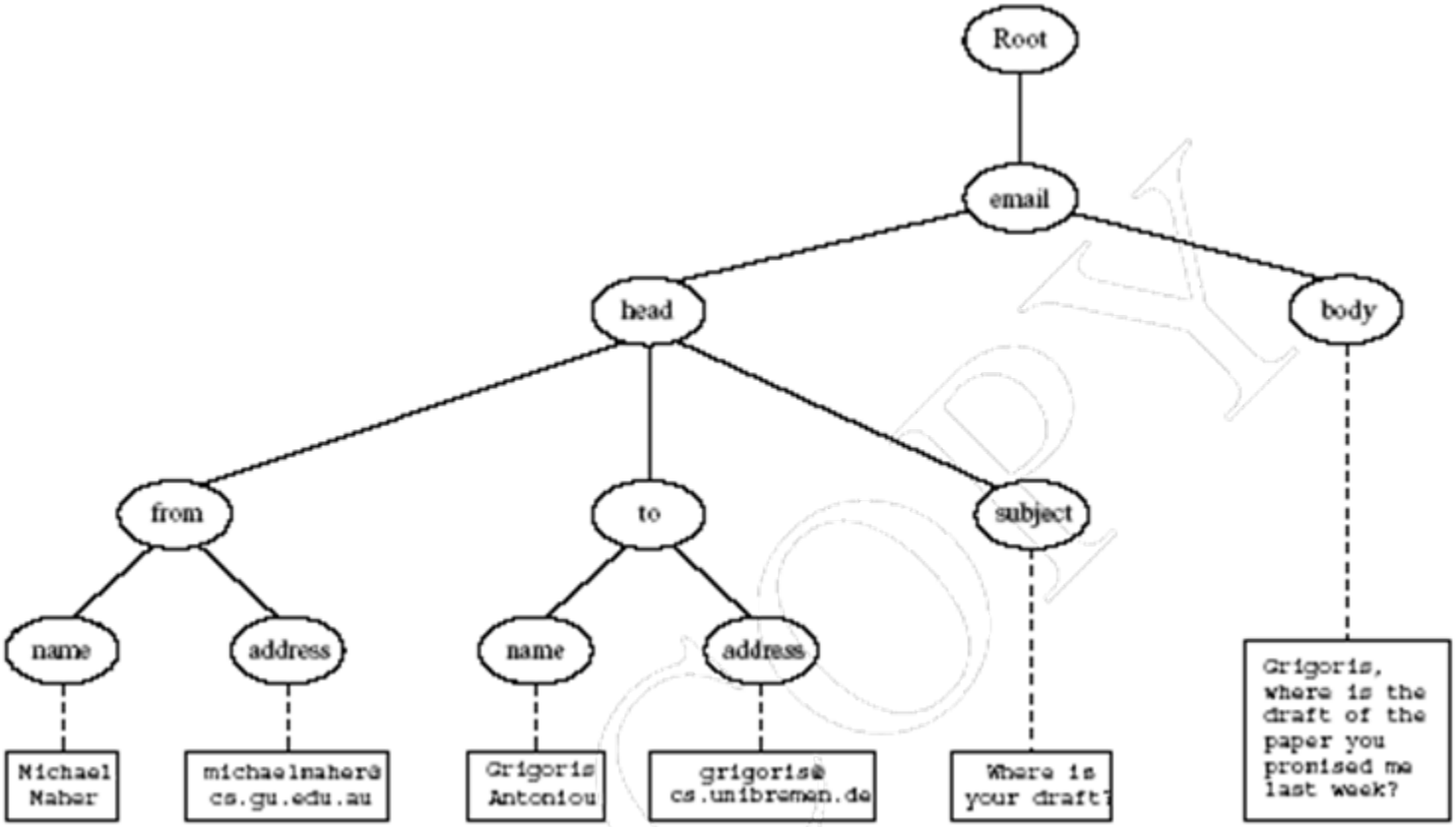
Well-Formed XML Documents

- Syntactically correct documents
- Some syntactic rules:
  - Only one outermost element (called root element)
  - Each element contains an opening and a corresponding closing tag
  - Tags may not overlap
    - **<author><name>Lee Hong</author></name>**
  - Attributes within an element have unique names
  - Element and tag names must be permissible

The Tree Model of XML Documents: An Example

```
<email>
    <head>
        <from name="Michael Maher"
              address="michaelmaher@cs.gu.edu.au"/>
        <to name="Grigoris Antoniou"
              address="grigoris@cs.unibremen.de"/>
        <subject>Where is your draft?</subject>
    </head>
    <body>
        Grigoris, where is the draft of the paper you promised me
        last week?
    </body>
</email>
```

The Tree Model of XML Documents: An Example (2)



The Tree Model of XML Docs

- The tree representation of an XML document is an ordered labeled tree:
  - There is exactly one root
  - There are no cycles
  - Each non-root node has exactly one parent
  - Each node has a label.
  - The order of elements is important
  - … but the order of attributes is not important

Lecture Outline

1. Introduction
2. Detailed Description of XML
3. Structuring
   a) DTDs
   b) XML Schema
4. Namespaces
5. Accessing, querying XML documents: XPath
6. Transformations: XSLT

Structuring XML Documents

- Define all the element and attribute names that may be used
- Define the structure
  - what values an attribute may take
  - which elements may or must occur within other elements, etc.
- If such structuring information exists, the document can be **validated**

Structuring XML Dcuments (2)

- An XML document is valid if
  - it is well-formed
  - respects the structuring information it uses
- There are two ways of defining the structure of XML documents:
  - DTDs (the older and more restricted way)
  - XML Schema (offers extended possibilities)

Lecture Outline

1. Introduction
2. Detailed Description of XML
3. Structuring
   a) DTDs
   b) XML Schema
4. Namespaces
5. Accessing, querying XML documents: XPath
6. Transformations: XSLT

XML Schema

- Significantly richer language for defining the structure of XML documents
- Tts syntax is based on XML itself
  - not necessary to write separate tools
- Reuse and refinement of schemas
  - Expand or delete already existent schemas
- Sophisticated set of data types, compared to DTDs (which only supports strings)

XML Schema (2)

- An XML schema is an element with an opening tag like

<schema
"http://www.w3.org/2000/10/XMLSchema"
version="1.0">

- Structure of schema elements
  - Element and attribute types using data types

Element Types

```
<element name="email"/>
<element name="head" minOccurs="1"
   maxOccurs="1"/>
<element name="to" minOccurs="1"/>
```

Cardinality constraints:

- **minOccurs="x"** (default value 1)
- **maxOccurs="x"** (default value 1)
- Generalizations of *,?,+ offered by DTDs

Attribute Types

<attribute name="id" type="ID“
use="required"/>

< attribute name="speaks" type="Language"
use="default" value="en"/>

- Existence: **use="x"**, where **x** may be **optional** or **required**
- Default value: **use="x" value="..."**, where **x** may be **default** or **fixed**

Data Types

- There is a variety of built-in data types
  - Numerical data types: **integer**, **Short** etc.
  - String types: **string**, **ID**, **IDREF**, **CDATA** etc.
  - Date and time data types: **time**, **Month** etc.
- There are also user-defined data types
  - simple data types, which cannot use elements or attributes
  - complex data types, which can use these

Data Types (2)

- Complex data types are defined from already existing data types by defining some attributes (if any) and using:
  - **sequence**, a sequence of existing data type elements (order is important)
  - **all**, a collection of elements that must appear (order is not important)
  - **choice**, a collection of elements, of which one will be chosen

A Data Type Example

```
<complexType name="lecturerType">
    <sequence>
        <element name="firstname" type="string"
            minOccurs="0“   maxOccurs="unbounded"/>
        <element name="lastname" type="string"/>
    </sequence>
    <attribute name="title" type="string"
        use="optional"/>
</complexType>
```

Data Type Extension

- Already existing data types can be extended by new elements or attributes. Example:

```
<complexType name="extendedLecturerType">
    <extension base="lecturerType">
        <sequence>
                <element name="email" type="string"
                        minOccurs="0" maxOccurs="1"/>
        </sequence>
        <attribute name="rank" type="string" use="required"/>
    </extension>
</complexType>
```

Resulting Data Type

```
<complexType name="extendedLecturerType">
    <sequence>
        <element name="firstname" type="string"
                 minOccurs="0" maxOccurs="unbounded"/>
        <element name="lastname" type="string"/>
        <element name="email" type="string"
                 minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="title" type="string" use="optional"/>
    <attribute name="rank" type="string" use="required"/>
</complexType>
```

Data Type Extension (2)

- A hierarchical relationship exists between the original and the extended type
  - Instances of the extended type are also instances of the original type
  - They may contain additional information, but neither less information, nor information of the wrong type

Data Type Restriction

- An existing data type may be restricted by adding constraints on certain values
- Restriction is not the opposite from extension
  - Restriction is not achieved by deleting elements or attributes
- The following hierarchical relationship still holds:
  - Instances of the restricted type are also instances of the original type
  - They satisfy at least the constraints of the original type

Example of Data Type Restriction

```
<complexType name="restrictedLecturerType">
      <restriction base="lecturerType">
            <sequence>
                  <element name="firstname" type="string"
                        minOccurs="1" maxOccurs="2"/>
             </sequence>
             <attribute name="title" type="string"
                  use="required"/>
            </restriction>
</complexType>
```

Restriction of Simple Data Types

```
<simpleType name="dayOfMonth">
        <restriction base="integer">
                <minInclusive value="1"/>
                <maxInclusive value="31"/>
        </restriction>
</simpleType>
```

Data Type Restriction: Enumeration

```
<simpleType name="dayOfWeek">
        <restriction base="string">
                <enumeration value="Mon"/>
                <enumeration value="Tue"/>
                <enumeration value="Wed"/>
                <enumeration value="Thu"/>
                <enumeration value="Fri"/>
                <enumeration value="Sat"/>
                <enumeration value="Sun"/>
        </restriction>
</simpleType>
```

XML Schema: The Email Example

```
<element name="email" type="emailType"/>

<complexType name="emailType">
    <sequence>
        <element name="head" type="headType"/>
        <element name="body" type="bodyType"/>
    </sequence>
</complexType>
```

XML Schema: The Email Example (2)

```
<complexType name="headType">
    <sequence>
        <element name="from" type="nameAddress"/>
        <element name="to" type="nameAddress"
                 minOccurs="1" maxOccurs="unbounded"/>
        <element name="cc" type="nameAddress"
                 minOccurs="0" maxOccurs="unbounded"/>
        <element name="subject" type="string"/>
    </sequence>
</complexType>
```

XML Schema: The Email Example (3)

```
<complexType name="nameAddress">
        <attribute name="name" type="string"
        use="optional"/>
        <attribute name="address"
        type="string" use="required"/>
</complexType>
```

- Similar for **bodyType**

Lecture Outline

1. Introduction
2. Detailed Description of XML
3. Structuring
   a) DTDs
   b) XML Schema
4. Namespaces
5. Accessing, querying XML documents: XPath
6. Transformations: XSLT

Namespaces

- An XML document may use more than one DTD or schema
- Since each structuring document was developed independently, name clashes may appear
- The solution is to use a different prefix for each DTD or schema
  - **prefix:name**

An Example

```
<vu:instructors   xmlns:vu="http://www.vu.com/empDTD"
                  xmlns:gu="http://www.gu.au/empDTD"
                  xmlns:uky="http://www.uky.edu/empDTD">

     <uky:faculty uky:title="assistant professor"
                  uky:name="John Smith"
                  uky:department="Computer Science"/>

<gu:academicStaff      gu:title="lecturer"
                       gu:name="Mate Jones"
                       gu:school="Information Technology"/>
</vu:instructors>
```

Namespace Declarations

- Namespaces are declared within an element and can be used in that element and any of its children (elements and attributes)
- A namespace declaration has the form:
  - **xmlns:prefix="location"**
  - **location** is the address of the DTD or schema
- If a prefix is not specified: **xmlns="location"** then the **location** is used by default