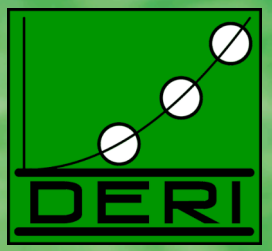

Advanced Studies in IT
CT433

Review of Assignments,
Lecture 5

sfi science foundation ireland
fondúireacht eolaíochta éireann

National University of Ireland, Galway
Ollscoil na hÉireann, Gaillimh

Exercise

1) Play around: Check the examplesDTDXSD.zip file attached to this exercise sheet and understand the example DTDs and XML Schemas therein. Check validity of the instance files with respect to their Grammars, create/modify other valid instances of the xsd and dtd files.

Some changes in bsp1.xml:


```
<?xml version="1.0" ?>
<!--
 <!DOCTYPE course [
<!ENTITY % xyz "lecture">
 <!ELEMENT course (lecture)+>
 <!ELEMENT lecture (#PCDATAltitle)*>
 <!ATTLIST lecture date CDATA #REQUIRED>
 <!ELEMENT title (#PCDATA)>
]>
-->
<!DOCTYPE course SYSTEM "/Users/axepol/Documents/teaching/ws2007/AIT_3exercises/examplesDTDXSD/bsp1.dtd">
<course>
 <lecture date="04/03/2004" lvanr="nr4711">
  <title>Introduction</title>
 </lecture>
  <lecture date="11/03/2004" lvanr="nr0815">
  This lecture has the title <title>What is Telecooperation? Concepts, History and Application Areas</title>.
 </lecture>
 <lecture date="18/03/2004">
  <title>From the current Web towards a Semantic Web, Introduction XML &amp;amp; Semistructured Data</title>
 </lecture>
</course>
```

An example for bsp2.xml:


```
<?xml version="1.0" encoding="UTF-8"?>
<xyz:purchaseOrder xmlns:xyz="http://www.xyz.com"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.xyz.com
file:/Users/axepol/Documents/teaching/ws2007/AIT_3exercises/examplesDTDXSD/bsp2.xsd">
   <xyz:shipTo country="US">
      <xyz:name>John Doe</xyz:name>
      <xyz:street>Doestreet</xyz:street>
      <xyz:city>Doetown</xyz:city>
      <xyz:state>TX</xyz:state>
      <xyz:zip>12345</xyz:zip>
   </xyz:shipTo>
   <xyz:billTo country="IE">
      <xyz:name>Axel Polleres</xyz:name>
      <xyz:street>DERI, Lower Dangan</xyz:street>
      <xyz:city>Galway</xyz:city>
      <xyz:state>Galway</xyz:state>
      <xyz:zip>0000</xyz:zip>
   </xyz:billTo>
   <xyz:items>
      <xyz:item partNum="012-XY">
      <xyz:productName>Semantic Web Primer</xyz:productName>
      <xyz:quantity>1</xyz:quantity>
      <xyz:USPrice>10</xyz:USPrice>
     </xyz:item>
   </xyz:items>
</xyz:purchaseOrder>
```

Attention: I had changed the fixed="US" for attribute country!

A slight modification of bsp3.xml


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<lehre xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="beispiel.xsd">
    <veranstaltung jahr="2003">
        <schlagwort>XML</schlagwort>
        <schlagwort>XSLT</schlagwort>
        <schlagwort>Wrapper</schlagwort>
        <vorbesprechung>
            <teilnehmer>15</teilnehmer>
            <ort>Seminarraum 184/2</ort>
        </vorbesprechung>
    </veranstaltung>
</lehre>
```

Also check slight modifications in beispiel.xsd

Exercise

2) Create an XML-Schema for the following XML document, using:

- Global types (type referencing)
  --> for instance "DepartmentType"
- Restricted types
  --> for instance simpleSalaryType restricting to strings with 4 digits, a comma and 2 digits.
- Extended types
  --> for instance SalaryType extending simpleSalaryType by an attribute Contracttype
- Documentation-Annotations

Part 1:


```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.example.com/Staff"
    xmlns="http://www.example.com/Staff"
    elementFormDefault="qualified">

    <xs:annotation>
        <xs:documentation xml:lang="en">
            This is my very unimportant annotation for the staff schema for Example.com.
            Copyright 2008 Axel Polleres. All rights reserved.
        </xs:documentation>
    </xs:annotation>

    <xs:element name="Staff">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Marketing" type="DepartmentType"/>
                <xs:element name="Development" type="DepartmentType"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="DepartmentType">
        <xs:sequence>
            <xs:element ref="Employee" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

Part 2:


```
<xs:simpleType name="simpleSalaryType">
    <xs:restriction base="xs:string">
        <xs:pattern value="\d{4},\d{2}"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="SalaryType">
    <xs:simpleContent>
        <xs:extension base="simpleSalaryType">
            <xs:attribute name="ContractType" type="xs:string"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<xs:element name="Employee">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Name" type="xs:string"/>
            <xs:element name="Salary" type="SalaryType"/>
            <xs:element name="Email" type="xs:string"/>
            <xs:element name="Address">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Street" type="xs:string"/>
                        <xs:element name="City" type="xs:string"/>
                        <xs:element name="Zip" type="xs:string"/>
                        <xs:element name="Country" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="ID" type="xs:integer"></xs:attribute>
    </xs:complexType>
</xs:element>

</xs:schema>
```

Xpath: 3) Taking as input the students.xml file (in the zip file), provide XPath expressions for the following statements.

a) Select all the elements in the document

```
//*
```

b) Select all students

```
//student
```

c) Select the last name of the second student of the document

```
//student[2]/lastname/text()
```

d) Select the name of the student with id “2028”

```
//student[@id="2028"]/name/text()
```

e) Select the last student of the document

```
//student[last()]
```

f) Select the name and last name of all the students

```
//student/concat(./name," ",./lastname)
//student/*[string(node-name(.)) = "lastname" or string(node-name(.)) = "name"]
```

g) Select all the students’ ids

```
//@id
```

h) Select the name, last name and telephone of the male students

```
//student[./gender="Male"]/*[string(node-name(.)) = "lastname" or string(node-name(.)) = "name" or string(node-name(.)) = "tel"]
```

i) Select the name and last name of the students that are males or study the master degree

```
//student [./gender="Male” or ./master=”Yes”] /concat(./name," ",./lastname)
```

j) Select all the students who have a telephone number

~~//student[./tel]~~ //student[data(./tel)]

4) Write down XPath expressions for the following queries:

a) all courses of type "projectlab" where Thomas Strang was not a lecturer

~~//course[@type="projectLab" and ./lecturer/text() != "Thomas Strang"]~~

//course[@type="projectLab" and not(./lecturer/text()= "Thomas Strang")

b) the first course which has more than one exam

//course[count(./exam)>1][1]

c) all titles of lectures before 2005

//title[../@year < 2005]

5) Given the following XML and XSLT, write the resulting document:

Note: There were some quotes missing in the stylesheet:
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" …

Result:

```
<?xml version="1.0" encoding="utf-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">

 <title>Management Director, Widget Inc.</title>
 <body>
  <h2 id="123">Max Muster</h2>

  <p>email: <a href="mailto:max.muster@widget.com”>
                    <tt>max.muster@widget.com</tt></a>
  </p>
 </body>
</html>
```

Exercise Sheet 2

Write down the following queries in SPARQL and their result tables:

1) a) Which are the names (indicated by the predicate `http://xmlns.com/foaf/0.1/knows`) and FOAF files (indicated by the property `http://www.w3.org/2000/01/rdf-schema#seeAlso`) of persons known by Axel Polleres?

Solution: q1.sparql:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: < http://www.w3.org/2000/01/rdf-schema#>
SELECT ?N ?F
FROM <http://www.polleres.net/foaf.rdf>
WHERE { <http://www.polleres.net/foaf.rdf#me> foaf:knows ?P .
        ?P foaf:name ?N . ?P rdfs:seeAlso ?F.
                }
```

- Call with: arq -query q1.sparql

  Among other answers, we get:

?N	?F
"Jos De Bruijn"	<http://www.debruijn.net/foaf.rdf>
"Giovambattista Ianni"	<http://www.gibbi.com/foaf.rdf>

1) b) Based on the answers from a) : Which are the names of persons known by Giovambattista Ianni OR Jos De Bruijn?


```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?N
FROM <http://www.debruijn.net/foaf.rdf>
FROM <http://www.gibbi.com/foaf.rdf>
WHERE { ?P foaf:knows ?P1. ?P1 foaf:name ?N }
```

Alternatively:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?N
FROM NAMED <http://www.debruijn.net/foaf.rdf>
FROM NAMED <http://www.gibbi.com/foaf.rdf>
WHERE { GRAPH ?G {?P foaf:knows ?P1. ?P1 foaf:name ?N } }
```

You can additionally order result, limit the number of results, etc.

1) c) Write a query querying ALL those FOAF files found in a) (using several FROM and/or FROM NAMED clauses) that states: Which are the names of all persons who don't know anybody called "Jos De Bruijn"?


```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N
FROM <http://www.polleres.net/foaf.rdf>
FROM NAMED <http://danbri.org/foaf.rdf>
FROM NAMED <http://nets.ii.uam.es/~rlara/foaf.rdf>
FROM NAMED <http://www.debruijn.net/foaf.rdf>
FROM NAMED <http://hometown.aol.com/chbussler/foaf/chbussler.foaf>
FROM NAMED <http://members.deri.at/~jamess/foaf_james.xml>
FROM NAMED <http://page.mi.fu-berlin.de/mochol/foaf.rdf>
FROM NAMED <http://sw.deri.org/~haller/foaf.rdf>
FROM NAMED <http://www.postsubmeta.net/foaf.rdf>
FROM NAMED <http://page.mi.fu-berlin.de/~nixon/foaf.rdf>
FROM NAMED <http://www.kr.tuwien.ac.at/staff/roman/foaf.rdf>
FROM NAMED <http://www.johnbreslin.com/foaf/foaf.rdf>
FROM NAMED <http://www.gibbi.com/foaf.rdf>
FROM NAMED <http://members.deri.at/~holgerl/foaf.rdf>
FROM NAMED <http://tinf2.vub.ac.be/~sheymans/foaf.rdf>
FROM NAMED <http://www.scharffe.fr/foaf.rdf>
FROM NAMED <http://www.harth.org/~andreas/foaf.rdf>
FROM NAMED <http://www.ee.surrey.ac.uk/Personal/A.Zhdanova/foaf.rdf>
FROM NAMED <http://sw.deri.org/~ina/foaf.rdf>
FROM NAMED <http://g1o.net/foaf.rdf>
FROM NAMED <http://www-cdr.stanford.edu/~petrie/foaf.rdf>
FROM NAMED <http://www.aifb.uni-karlsruhe.de/Personen/viewPersonFOAF/foaf_2084.rdf>
FROM NAMED <http://eyaloren.org/foaf.rdf>
WHERE { GRAPH ?G {
          ?P foaf:name ?N; foaf:knows _:someone .
          OPTIONAL {?P foaf:knows ?J. ?J foaf:name "Jos de Bruijn" .}
          FILTER (!bound(?J)) }
      }
```

Problem with this query: Not all files accessible, some corrupt, tools do not yet Properly deal with this… compare web browsers!

Slight variation of 1) d) Return the names of all persons who *do* have a URI as identifier (needs FILTERs, particularly the functions isBlank( ) or isIRI( ) can be used)?


```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N
FROM <http://www.polleres.net/foaf.rdf>
FROM NAMED <http://danbri.org/foaf.rdf>
FROM NAMED <http://nets.ii.uam.es/~rlara/foaf.rdf>
FROM NAMED <http://www.debruijn.net/foaf.rdf>
FROM NAMED <http://hometown.aol.com/chbussler/foaf/chbussler.foaf>
FROM NAMED <http://members.deri.at/~jamess/foaf_james.xml>
FROM NAMED <http://page.mi.fu-berlin.de/mochol/foaf.rdf>
FROM NAMED <http://sw.deri.org/~haller/foaf.rdf>
FROM NAMED <http://www.postsubmeta.net/foaf.rdf>
FROM NAMED <http://page.mi.fu-berlin.de/~nixon/foaf.rdf>
FROM NAMED <http://www.kr.tuwien.ac.at/staff/roman/foaf.rdf>
FROM NAMED <http://www.johnbreslin.com/foaf/foaf.rdf>
FROM NAMED <http://www.gibbi.com/foaf.rdf>
FROM NAMED <http://members.deri.at/~holgerl/foaf.rdf>
FROM NAMED <http://tinf2.vub.ac.be/~sheymans/foaf.rdf>
FROM NAMED <http://www.scharffe.fr/foaf.rdf>
FROM NAMED <http://www.harth.org/~andreas/foaf.rdf>
FROM NAMED <http://www.ee.surrey.ac.uk/Personal/A.Zhdanova/foaf.rdf>
FROM NAMED <http://sw.deri.org/~ina/foaf.rdf>
FROM NAMED <http://g1o.net/foaf.rdf>
FROM NAMED <http://www-cdr.stanford.edu/~petrie/foaf.rdf>
FROM NAMED <http://www.aifb.uni-karlsruhe.de/Personen/viewPersonFOAF/foaf_2084.rdf>
FROM NAMED <http://eyaloren.org/foaf.rdf>
WHERE { GRAPH ?G {
          ?P a foaf:Person . ?P foaf:name ?N.
          FILTER (isIRI(?P)) }
      }
```

Problem with this query: Not all files accessible, some corrupt, tools do not yet Properly deal with this… compare web browsers!

CONSTRUCT queries…

… next time!