Semantic Web

Assignment 0. Demo Exam

Johannes Härtel Iryna Dubrovska

Institute of Web Science and Technologies
Department of Computer Science
University of Koblenz-Landau

1 Multiple Choice (10 Points)

1.1 Linked Data Principles (5 points)

Mark correct and incorrect statements. You get 0.5 points for every box correctly checked; -0.5 points otherwise. Negative points do not carry over to the other sub-task.

Correct	Not correct	
☐	☐	Use URIs as names for things.
☐	☐	Include links by using URIs that dereference remote documents.
☐	☐	Do not use relational databases to provide RDF triple persistence.
☐	☐	Use HTTP URIs so those names can be dereferenced.
☐	☐	Use OWL Full as much as possible.
☐	☐	Use the XML serialization to represent RDF data.
☐	☐	Use one vocabulary per data set.
☐	☐	Return useful information upon lookup of those URIs.
☐	☐	Always use DBpedia as a bridge when defining links.
☐	☐	Enable a SPARQL endpoint to make URIs deferenceable.

1.2 Ontology engineering methodology (5 points)

Correct	Not correct	
☐	☐	Draw the ontology graphically before defining axioms.
☐	☐	Consider reusing existing ontologies (or parts of them).
☐	☐	Define the facets of the slots.
☐	☐	Define the properties of the classes (slots).
☐	☐	Define the classes and the class hierarchy.
☐	☐	Create instances (of classes).
☐	☐	Define classes based on potential SPARQL queries.
☐	☐	Enumerate important terms in the ontology.
☐	☐	Determine the domain and scope of the ontology.
☐	☐	Define only foundation (or upper level) ontologies.

2 XML (10 points)

Consider the following document type definition (DTD):

```
<!DOCTYPE a [
    <!ELEMENT a (b*, c)+>
    <!ELEMENT b (c*, d*, e*)>
    <!ELEMENT c (a*, d)+>
    <!ELEMENT d (#PCDATA)>
    <!ELEMENT e EMPTY>
]>
```

For each of the following XML documents, with respect to the above DTD, check whether the document is *well-formed* and *valid*.

If an XML document is not well-formed or not valid with respect to the above DTD, give all the reasons.

2.1 (5 points)

```
<a>
    <b>
    <c><d> Valid? </c></d>
    <c><d></d></c>
    <c><d/><d></c>
    <b>
</a>
```

2.2 (5 points)

```
<a>
    <b><d>Valid</d></b>
    <c>
        <a><c><d>or</d></c></a>
        <d>not</d>
    </c>
</a>
```

3 RDF and RDFS (15 points)

Consider the following RDF/XML document. The document contains information about Vancouver:

```
 1: <?xml version="1.0" encoding="utf-8" ?>
 2: <rdf:RDF
 3:    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 4:    xmlns:xo="http://example.org/ontology#"
 5:
 6:   <rdf:Description rdf:about="http://example.org/resource/Vancouver">
 7:     <rdf:type rdf:resource="http://example.org/ontology#City"/>
 8:     <xo:name>Vanvouver</xo:name>
 9:     <xo:latitude>49.25</xo:latitude>
10:     <xo:longitude>-123.1</xo:longitude>
11:     <xo:population>631,000</xo:population>
12:     <xo:state rdf:resource="http://example.org/resource/British_Columbia"/>
13:     <xo:mayor>
14:          <xo:Person rdf:about="http://example.org/resource/Gregor_Robertson"
15:              <xo:age>53</xo:age>
16:              <xo:assumedOffice>2008-12-08</xo:assumedOffice>
17:          <xo:party rdf:resource="http://example.org/resource/BC_NDP"/>
18:          </xo:Person>
19:     </xo:mayor>
20:    </rdf:Description>
21: </rdf:RDF>
```

3.1 (12 points)

Translate above information into a visual RDF graph representation using the graphical notation presented in the lecture.

3.2 (3 points)

We want to specify districts of Vancouver. Use an RDF container of your choice to group the following 3 districts: `x:Kitsilano`, `x:Gastown` and `x:Yaletown`. Express the information in graphical notation.

4 SPARQL (30 points)

Consider the excerpt of an RDF dataset on the next page containing touristic and geographical information.

Study the data structure apparent from the snippet. Based on that structure, formulate SPARQL queries for each of the following tasks.

Hints:

• Specify your queries as precisely as possible.

• View above listing as just an excerpt. Expect there to be other cities, states, countries and their points of interest.

• You can assume the used prefixes to be known. You don't need to declare them.

• Expect numerical literals to be typed accordingly, even without specific types annotations to them.

4.1 (4 points)

Specify a query that performs a check for existence for a point-of-interest 'Seattle Great Wheel' in the city of Seattle `(x:city2)` of type `xo:FerrisWheel`.

4.2 (5 points)

Specify a query that returns the population of each German city named 'Frankfurt'. Refer to `x:Germany` as country.

4.3 (6 points)

Specify a query that returns the total area of all the states that are part of `x:Germany` or `x:USA`.

4.4 (7 points)

Specify a query that returns the name and population of each of the 5 most populous cities (ordered descendingly) within the state of `x:Hesse`.

4.5 (8 points)

Specify a query that returns all the museums within a (geo-spatial) bounding box of 1 degree in width and height around location (lat: 41.5; long: 12.3).

```
@prefix x:     <http://example.org/resource/> .
@prefix xo:    <http://example.org/ontology#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .

# countries and cities

x:city1      rdf:type         xo:City;
             xo:name          "Frankfurt";
             xo:population    "736,000";
             xo:state         x:Hesse.
x:city2      rdf:type         xo:City;
             xo:name          "Seattle";
             xo:population    "704,352";
             xo:state         x:Washington.
...

# states and regions

x:Hesse          xo:name      "Hesse";
                 xo:area      "21,114.94";
                 xo:country   x:Germany.
x:Washington     xo:name      "Washington";
                 xo:area      "184,665.27";
                 xo:country   x:USA.
...

# points of interest

x:city1      xo:pointOfInterest  x:poi11, x:poi12.
x:poi11      rdf:type         xo:Museum;
             xo:name          "Staedel Museum";
             xo:latitude      "50.173818";
             xo:longitude     "9.001271".
x:poi12      rdf:type         xo:Monument;
             xo:name          "Euro-Sculpture";
             xo:latitude      "50.010556";
             xo:longitude     "8.773361".
...
x:city2      xo:pointOfInterest  x:poi21, x:poi22.
x:poi21      rdf:type         xo:Tower;
             xo:name          "Space Needle";
             xo:latitude      "47.4934893";
             xo:longitude     "-121.923892".
x:poi22      rdf:type         xo:FerrisWheel;
             xo:name          "Seattle Great Wheel";
             xo:latitude      "47.605239";
             xo:longitude     "-122.54923".
...
```

5 Description Logics (15 points)

Consider the following statements:

i) An `administrative unit` may be `municipality`, `city district` or `city` and always has some `residents`.

ii) In administrative terms `municipalities, city districts` and `cities` are to be distinguished from each other (pairwise).

iii) An `administrative official` may be `mayor, councillor` or `clerk`.

iv) Necessarily `mayors govern` over an `administrative unit`.

v) A `lord mayor` is a `mayor` that `governs` a `city having` at least `100,000 residents`.

vi) A `lord mayor elected` in `Hamburg, Bremen` or `Berlin` also is a head-of-state.

5.1 (3 points)

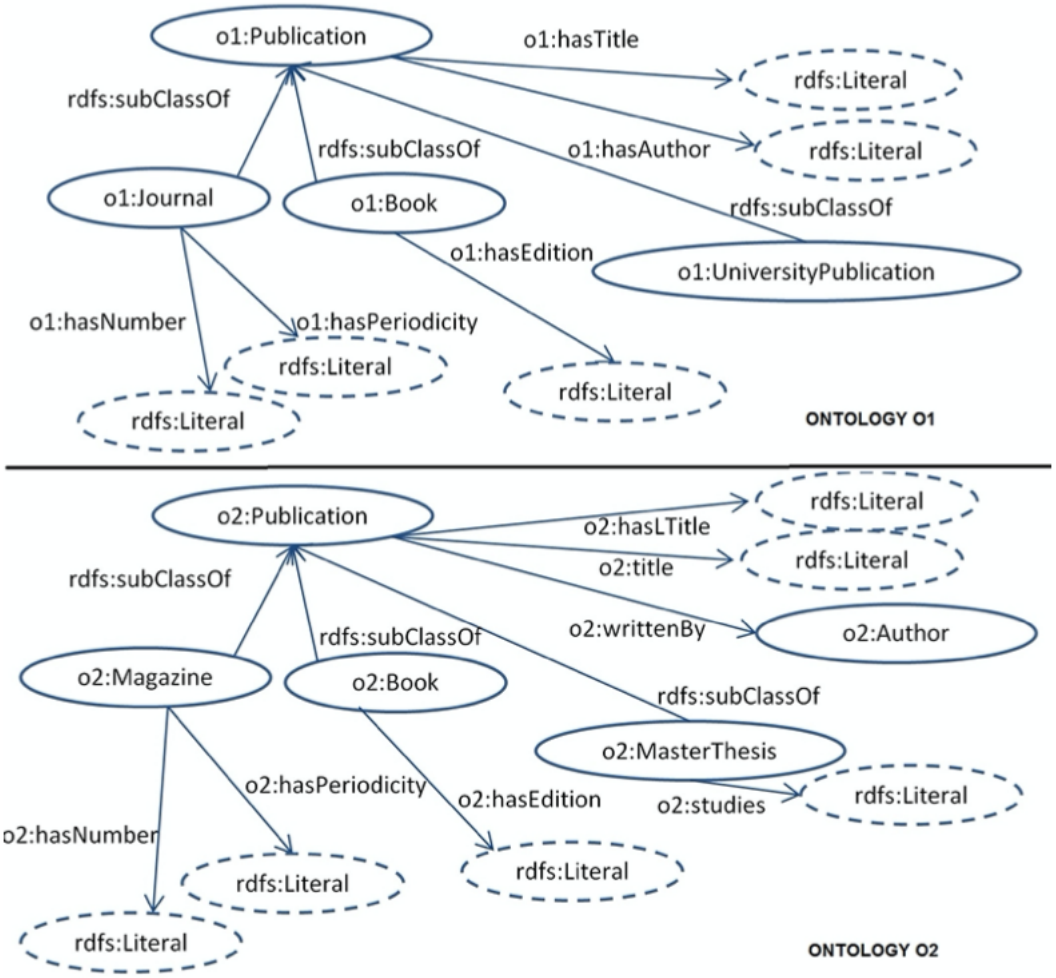
Define a signature for the given sentences.

5.2 (12 points)

Formalize DL axioms for the following simplified city administration domain using its mathematical notation. Formalize your axioms as precisely as possible, using the implied concepts and roles.

6 Ontology Alignment (10 points)

Consider the following two ontologies O1 and O2:



6.1 2.5 points)

Using string-based techniques, identify the mapping between the `o1:hasTitle` property and the corresponding element in O2 with minimum Levenshtein distance. Provide the calculations of the Levenshtein distance.

6.2 (2.5 points)

Using graph-based techniques, identify the mapping between the `o1:Journal` class and the corresponding element in O2.

6.3 (2.5 points)

What kind of mapping would you define between `o1:hasAuthor` and `o2:Author`?

- ☐ subsumption.
- ☐ equivalence.
- ☐ disjointness.
- ☐ does not apply.

6.4 (2.5 points)

Calculate Levenshtein distance between `o1:hasEdition` and `o2:hasPeriodicity`. You can assume that both strings are converted into lowercase.