Exam Preparation

Lecture 1. Intro

Be able to define:

- The meaning of Semantic Web and its difference from traditional web.
- Semantic Web layered architecture, meaning and importance of each component.
- Linked data concept and principles, 5* schema.
- URI, URL, URN.

Lecture 2. Description Logic

Be able to define:

- A signature.
- Terminological and assertional axioms (T-box and A-box).
- A model (interpretation).
- Main idea of the tableau algorithm, how it works and how to interpret the results.

Lecture 2. Warm-up*

For each of the concepts below, build an appropriate ALC description:

```
Concept: Human, Lucky, Pet, Cat, Old, Parrot
Property: owns
```

Task:

1. lucky human
2. lucky pet owner
3. human who owns only cats
4. unlucky pet owners who own an old cat
5. pet owners who only own cats and parrot

* Solutions can be found in the end of the presentation

Lecture 2. Warm-up*

Formalise the knowledge described in the given sentences using DL syntax:

i) An image is a special kind of mediaResource.

ii) A video is a special kind of mediaResource.

iii) A video is not an image.

iv) A title is a string of characters used to specify the name of images and videos.

v) Only videos have a frameRate (the frames per second in a video). One video can only have one frameRate.

vi) A mediaResource has at least one creator which is a kind of Person.

vii) A photographer is a kind of creator.

viii) The creator of an image must be a photographer.

ix) A portraitFoto is a kind of image, depicting exactly one person.

* Solutions can be found in the end of the presentation

Lecture 3. XML

Be able to:

- Define well-formed and valid XML.
- Identify from an XML document whether it is well-formed and validate it against its schema.
- Explain why we need namespaces.
- Outline differences of XMLSchema vs DTD.
- Understand DTD elements and attributes!

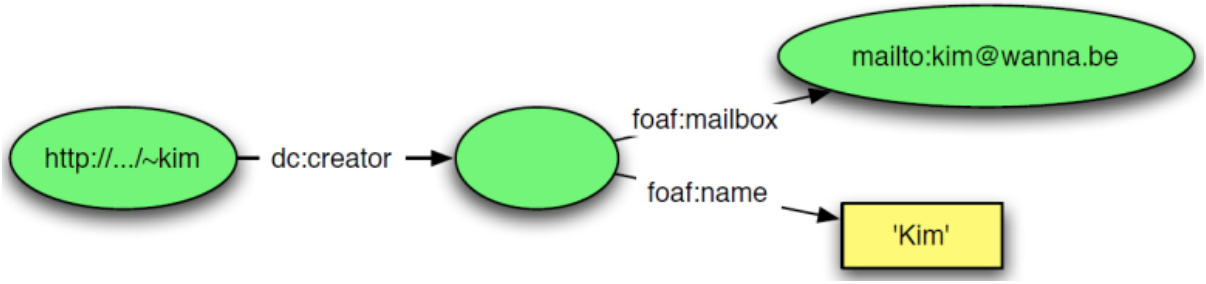
Lecture 4-5. RDF/RDFS/OWL

Be able to:

- Understand the definition of RDF.
- Explain the need for RDFS (e.g., RDF adds no restrictions on how predicates combine with subjects and objects).
- Explain the need for OWL (e.g., RDFS cannot express the statement that two sets are disjoint).
- Use vocabulary and main constructs of RDF.
- Translate RDF/XML to a labeled graph and the other way around.
- Derive RDF/XML from a given scenario.
- Represent blank nodes in RDF/XML.
- Recognise blank nodes in RDF/XML .
- Represent complex statements in a form of a graph.
- Understand the main difference between different languages of OWL family.

Lecture 4-5. Warm-up

Represent the graph as RDF/XML:



by https://www.inf.ed.ac.uk/

Lecture 4-5. Warm-up

Represent in OWL following object **properties**:

1. `ancestor` such as if person A is an ancestor of person B and B of C, then A is also an ancestor of C.
2. `akin` such as if a Person A is akin to a Person B, then B is also akin to A.
3. `hasFather` such as a child has always the same (biological) Father.
4. `hasChild` such as if a Person A hasChild a Person B, then B hasFather A.

Represent in OWL following **concepts**:

1. A `Mother` is a Woman that has a child (some Person).
2. The set of `parents that only have daughters` (female children).
3. A `half Orphan` (i.e. a person that has only one Parent)

by http://www-inf.it-sudparis.eu/

Lecture 4-5. Warm-up

Create a visual representation in a form of a graph of the following RDF/XML:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:xo="http://example.org/ontology#">
<rdf:Description rdf:about="http://example.org/res/cecil">
   <rdf:type rdf:resource="http://example.org/ontology#Lion"/>
   <xo:name>Cecil</xo:name> <xo:dateOfBirth>2011-06-27</xo:dateOfBirth> <xo:weight>156</xo:weight>
   <xo:status>recovering from rival attack</xo:status>
   <xo:habitat rdf:resource="http://example.org/res/KrugerNationalPark"/>
   </rdf:Description>
   <xo:Lion rdf:about="http://example.org/res/edmund">
      <xo:dateOfBirth>2012-03-21</xo:dateOfBirth>
      <xo:status>contesting cecil for leadership</xo:status>
      <xo:subordinateTo rdf:resource="http://example.org/res/cecil"/>
   </xo:Lion>
   <xo:Lioness rdf:about="http://example.org/res/diane">
      <xo:subordinateTo rdf:resource="http://example.org/res/cecil"/>
   </xo:Lioness>
   </rdf:RDF>
```

by http://www-inf.it-sudparis.eu/

Lecture 6. Ontology Engineering

Be able to:

- Name the steps of ontology engineering methodology.
- Understand what exactly happens on each particular step.

Lecture 7. Ontology Matching

Be able to:

- Name the dimensions of ontology matching and explain what they are referring to.
- Understand the mapping that can exist between ontologies (subsumption, equivalence etc.)
- Describe how ontology alignment approaches are classified.
- Calculate Levenshtein distance and normalised Levenshtein distance.
- Name properties of Levenshtein distance.
- Use graph-based techniques.

Lecture 8. SPARQL

Be able to:

- Understand different types of queries and the difference between the output produced by those types.
- Understand and use main operators and query modifiers in SELECT queries.
- **Read and understand** provided data and write queries.

Use available data excerpts (from assignments, demo-exam) to practice queries.

Lecture 9. Ontology-Based Data Access

Be able to:

- Define the goal of OBDA.
- Name components of OBDA.
- Perform GAV (global-as-view) and LAV (local-as-view) mappings.
- Write queries.

Answers to warm ups

Lecture 2. Warm-up

For each of the concepts below, build an appropriate ALC description:

```
Concept: Human, Lucky, Pet, Cat, Old, Parrot
Property: owns
```

Task:

1. lucky human
2. lucky pet owner
3. human who owns only cats
4. unlucky pet owners who own an old cat
5. pet owners who only own cats and parrot

1. $Human \cap Lucky$
2. $Human \cap Lucky \cap \exists owns\,.\,Pet$
3. $Human \cap \forall owns\,.\,Cat$
4. $Human \cap \neg Lucky \cap \exists owns\,.\,(Pet \cap Old \cap Cat)$
5. $Human \cap \exists owns\,.\,Pet \cap \forall owns\,.\,(Cat \cup Parrot)$

Lecture 2. Warm-up

Formalize the knowledge described in the given sentences using DL syntax:

i) $Image \sqsubseteq MediaResource$

ii) $Video \sqsubseteq MediaResource$

iii) $Image \sqcap Video \sqsubseteq \emptyset$

iv) $Image \sqcup Video \sqsubseteq \exists title.xsd : string$

v) $Video \equiv =_1 frameRate.nonNegativeInteger$

vi) $MediaResource \geq_1 creator.Creator.\ \ Creator \sqsubseteq Person.$

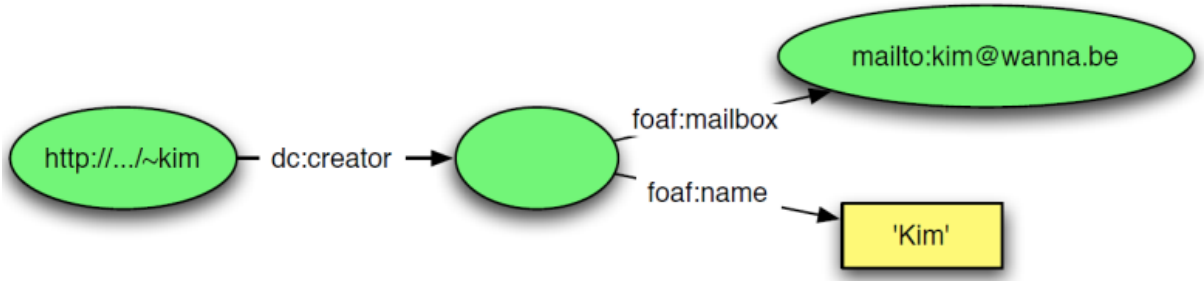
vii) $Photographer \sqsubseteq Creator$

viii) $Image \equiv \forall creator.Photographer$

ix) $PortraitPhoto \sqsubseteq Image \sqcap =_1 depicts.Persons$

Solutions can be found in the end of the presentation

Lecture 4-5. Warm-up



XML version of blank node

```
<rdf:Description rdf:about="http://.../~kim">
  <dc:creator rdf:nodeID="abc"/>
</rdf:Description>
<rdf:Description rdf:nodeID="abc">
  <foaf:mailbox  rdf:resource="mailto:kim@wanna.be"/>
  <foaf:name>Kim</foaf:name>
</rdf:Description>
```

by https://www.inf.ed.ac.uk/

Lecture 4-5. Warm-up

Represent in OWL following object **properties**:

1.
```
<owl:ObjectProperty rdf:ID="ancesotor">
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdfs:domain rdf:resource="#Person" />
  <rdfs:range rdf:resource="#Person" />
</owl:ObjectProperty>
```

2.
```
<owl:ObjectProperty rdf:ID="akin">
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdfs:domain rdf:resource="#Person" />
  <rdfs:range rdf:resource="#Person" />
</owl:ObjectProperty>
```

3.
```
<owl:ObjectProperty rdf:ID="hasFather">
  <rdf:tyoe rdf:resource="&owl;FunctionalProperty"/>
</owl:ObjectProperty>
```

4.
```
<owl:ObjectProperty rdf:ID="hasChild">
  <owl:inverseOf rdf:resource="hasParent" />
</owl:ObjectProperty>
```

by http://www-inf.it-sudparis.eu/

Lecture 4-5. Warm-up

Represent in OWL following **concepts**:

1.
```
<owl:Class rdf:ID="Mother">
  <rdfs:subClassOf rdf:resource="#Woman" />
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasChild" />
        <owl:someValuesFrom rdf:resource="#Person" />
     </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

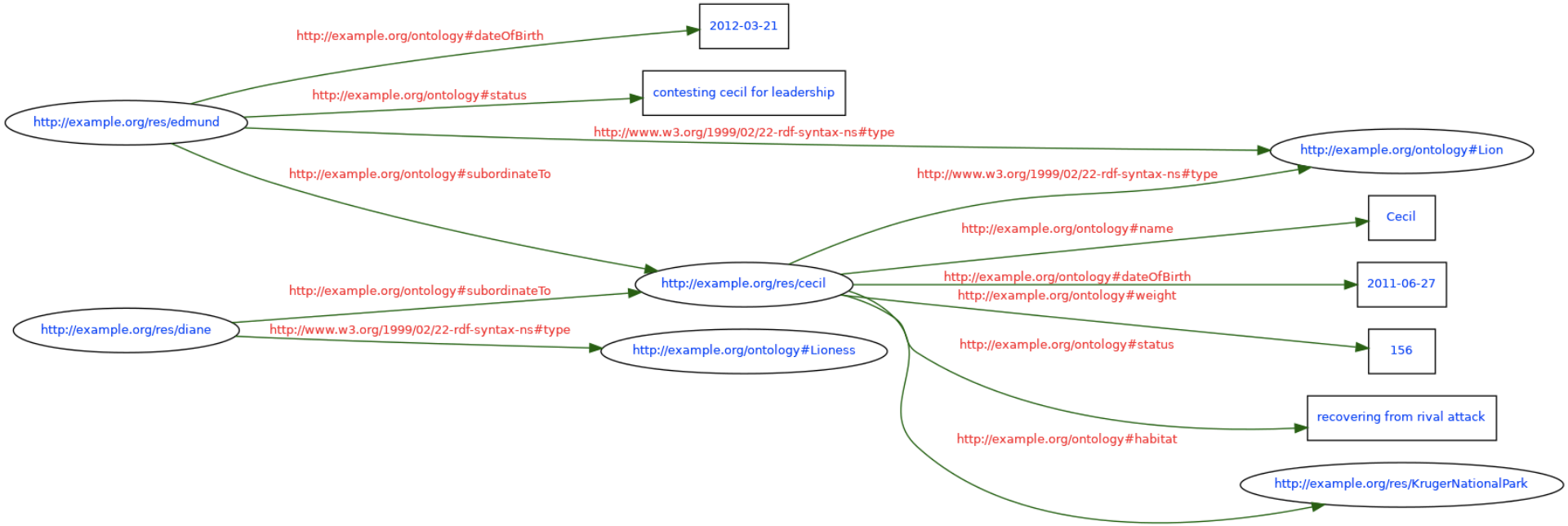
2.
```
<owl:Class rdf:ID="ParentsWithOnlyDaughters">
 <rdfs:subClassOf rdf:resource="#Person" />
   <rdfs:subClassOf>
     <owl:Restriction>
        <owl:onProperty rdf:resource="#hasChild" />
        <owl:allValuesFrom rdf:resource="#Woman" />
     </owl:Restriction>
   </rdfs:subClassOf>
    ...
</owl:Class>
```

3.
```
<owl:Class rdf:ID="HalfOrphan">
  <rdfs:subClassOf rdf:resource="#Person" />
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasParent"/>
        <owl:cardinality rdf:datatype="&xsd;NonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </rdfs:subClassOf>
  …
</owl:Class>
```

by http://www-inf.it-sudparis.eu/

Lecture 4-5. Warm-up

Create a visual representation in a form of a graph of the following RDF/XML:



by http://www-inf.it-sudparis.eu/

Questions?